UNISINOS

Programa de Pós-Graduação em

Computação Aplicada

Mestrado/Doutorado Acadêmico

Carlos Eduardo Carbonera

ProMerge: Uma abordagem para auxiliar desenvolvedores na detecção e resolução proativa de conflitos de integração de código fontes

São Leopoldo, 2024

Carlos Eduardo Carbonera

ProMerge: Uma abordagem para auxiliar desenvolvedores na detecção e resolução proativa de conflitos de integração de código fontes

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre, pelo Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos – UNISINOS

Orientador: Dr. Kleinner Silva Farias de Oliveira

São Leopoldo – 2024.

S264p Carbonera, Carlos Eduardo.
ProMerge : uma abordagem para auxiliar desenvolvedores na detecção e resolução proativa de conflitos de integração de código fontes / Carlos Eduardo Carbonera. – 2024.
155 f. : il. ; 30 cm.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, 2024.
"Orientador: Prof. Dr. Kleinner Silva Farias de Oliveira."

1. Artifacts. 2. Effort. 3. Integration. 4. Merge. 5. Software. I. Título.

CDU 004.4

Dados Internacionais de Catalogação na Publicação (CIP)
(Bibliotecária: Silvana Dornelles Studzinski – CRB 10/2524)

Dedico esse trabalho ao meus pais Ignácio Carbonera (em memória) e Carolina Segeuka Carbonera, que sempre me incetivaram (cada qual a sua maneira) para que eu nunca desistisse dos meus sonhos e sempre seguisse em frente.

AGRADECIMENTOS

Agradeço a Deus por ter me proporcionado esta oportunidade de crescimento pessoal e profissional.

Agradeço imensamente ao meu professor orientador Dr. Kleinner Silva Farias de Oliveira por ter me acolhido e me conduzido durante essa jornada, que com muita dedicação e paciência me orientou nos muitos momentos de dúvidas, meu sincero muito obrigado por tudo.

Agradeço aos colegas Daniel Bruno Armino, Vinícius Lima Ville, Carlos Eduardo Liedtke Borges e César Augusto Graeff por todo apoio e atenção. Agradeço especialmente à minha namorada, Marselha Vianna Altmann que teve muita paciência comigo durante essa caminhada. E, por fim, meu muito obrigado a todos(as) que de alguma maneira me auxiliaram durante essa caminhada.

"*Lembra-te de **Deus** em tudo o que fizeres, e **Ele** te mostrará o caminho certo.*".
(Provérbios 3:6)

RESUMO

A integração de arquivos fontes desempenha um papel fundamental em várias tarefas de desenvolvimento de software como, por exemplo, durante a acomodação de novas funcionalidades desenvolvidas, ou mesmo na reconciliação de trechos de código conflitantes alterados em paralelo por times distribuídos de desenvolvimento de software. Os conflitos surgem quando trechos de código recebem modificações divergentes implementadas em paralelo por diferentes desenvolvedores, afetando a estrutura e/ou semântica do código. Tais modificações podem afetar um mesmo trecho de código (conflito direto) ou trechos diferentes (conflito indireto). Embora a temática de integração de arquivos fontes tenham sido amplamente investigada e explorada na indústria e academia nas últimas décadas, a detecção e a resolução de conflitos são ainda consideradas tarefas altamente propensas a erros e que exigem um alto esforço de desenvolvedores. Essa pesquisa, portanto, propõe a ProMerge, a qual é uma abordagem para auxiliar desenvolvedores na detecção e resolução proativa de conflitos diretos e indiretos gerados à medida que trechos de código são modificados em paralelo. A ProMerge introduz o conceito de histórico de contexto de conflitos entre trechos de código, detecta conflitos em branches diferentes (ou não), auxilia desenvolvedores na avaliação da severidade de conflitos, bem como suporta o conceito de committing time. A ProMerge foi projetada a partir dos resultados obtidos de um mapeamento sistemático da literatura, o qual investigou duas décadas de trabalhos publicados sobre o tema de integração de software e explorou nove questões de pesquisa. A ProMerge foi implementada como um plug-in da plataforma Eclipse. A abordagem proposta foi avaliada através de um experimento controlado com trinta e dois profissionais da indústria, as quais executaram dez tarefas experimentais dividida em dois cenários para avaliar o esforço de integração, a corretude das integrações, bem como a taxa de erro ao realizar as integrações, gerando trezentos e vinte cenários de avaliação. Os resultados, suportados por testes estatísticos, indicam que a taxa de corretude encontrada foi superior em relação à abordagem tradicional e por fim, a taxa de erro encontrada na avaliação de todas as tarefas que integravam o experimento foi superior a abordagem tradicional. Além disso, uma avaliação qualitativa foi realizada, aplicando o questionário TAM, para entender o grau de aceitação da abordagem proposta. No total, trinta e um participantes são profissionais da indústria e responderam ao questionário. Os resultados indicam que a utilização da ProMerge reduziu de maneira significativa o esforço (tempo) para a resolução das tarefas, as informações de contexto geradas a partir da execução dos experimentos auxiliaram no melhor entendimento da taxa de erro e corretude. Por fim, a utilização da ProMerge contribuiu para a melhoria de desempenho dos desenvolvedores e na compreensão e aplicação dos novos conceitos implementados bem como para a geração de indicadores de desempenho e produtividade.

Palavras-chaves: Merge. Software. Branches.

ABSTRACT

The source files integration caracterize as a fundamental role in various software development tasks, for example, when it was accommodated new functionalities or it was when reconcilited conflicting code snippets changed in parallel by distributed software development teams. The conflicts are detected when code snippets receive divergent modifications implemented in parallel by different developers, affecting the structure and/or source code files semantic. This modifications can affect the same code section (direct conflict) or different sections (indirect conflict). Although the topic "source file integration" has been widely investigated and explored by the industry and academia in recent decades, conflict detection and resolution are still considered tasks that are highly prone to errors and require high effort from developers. This research, proposes ProMerge, an approach to assist developers to detect and resolve proactively direct and indirect conflicts generated as code snippets are modified in parallel. The ProMerge introduces the context history conflicts concept between code snippets, detects conflicts in different branches (or not), helps developers to evaluating the conflicts severity and supports the committing time concept. The ProMerge it was designed based on the results obtained from a systematic literature mapping, investigated published work at the last two decades about software integration topic evaluating nine research questions. The ProMerge was implemented as a plug-in for the Eclipse platform. The proposed approach it was evaluated through a controlled experiment with thirty-two industry professionals, that performed ten experimental tasks divided into two evaluate the integration effort scenarios, the integrations correctness and the error rate when integrations taske it was performed, generating three hundred and twenty evaluation scenarios. So, the results it was supported by statistical tests, indicating that the accuracy rate found was higher than the traditional approach. Finally, the error rate founded in the evaluation tasks that were part of the experiment it was higher than the traditional approach. Furthermore, a qualitative assessment it was executed, applying the TAM questionnaire, to understand the proposed approach acceptance degree. In total, thrity-one participants that answered the questionnaire are industry professionals. The results indicated that the ProMerge use presented a significantly reduced of the effort (time) to resolve the tasks. The context information generated from the experiments execution helped the developers to understand better the error and correctness rate . Finally, the ProMerge use contributed to improving developers performance and also in understanding and applying the new concepts implemented and generating performance and productivity indicators.

Keywords: Merge. Software. Branches.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE SIGLAS

API	Interface de Programação de Aplicativos
CE	Critério de Exclusão
CI	Critério de Inclusão
CO	Taxa de Corretude
CVS	Sistema de Controle e Versionamento
DAO	Objeto de Acesso a Dados
DML	Linguagem de Manipulação de Dados
ET	Esforço (Tempo)
GA	Grau de Abstração
JDBC	Conectividade Java com Banco de Dados
MVC	Modelo / Visão / Controlador
NC	Nível de Complexidade
NGS	Nível Grau de Severidade
OG	Objetivo Geral
OE	Objetivo Específico
Pn	Estudos Primários (n = Número de identificação)
QP	Questão de Pesquisa
SQL	Linguagem de Consulta Estruturada
SGDB	Sistema Gerenciador de Banco de Dados
SMS	Mapeamento Sistemático da Literatura
SOA	Estado da Arte
TE	Taxa de Erro
VO	Valor de Objeto

LISTA DE SÍMBOLOS

●	Aplica
◗	Aplica parcialmente
❍	Não se aplica
>	Maior
$\geq$	Maior ou igual
<	Menor
$\leq$	Menor ou igual
$\leftarrow$	Atribuição
$\neq$	Diferente

SUMÁRIO

1 INTRODUÇÃO

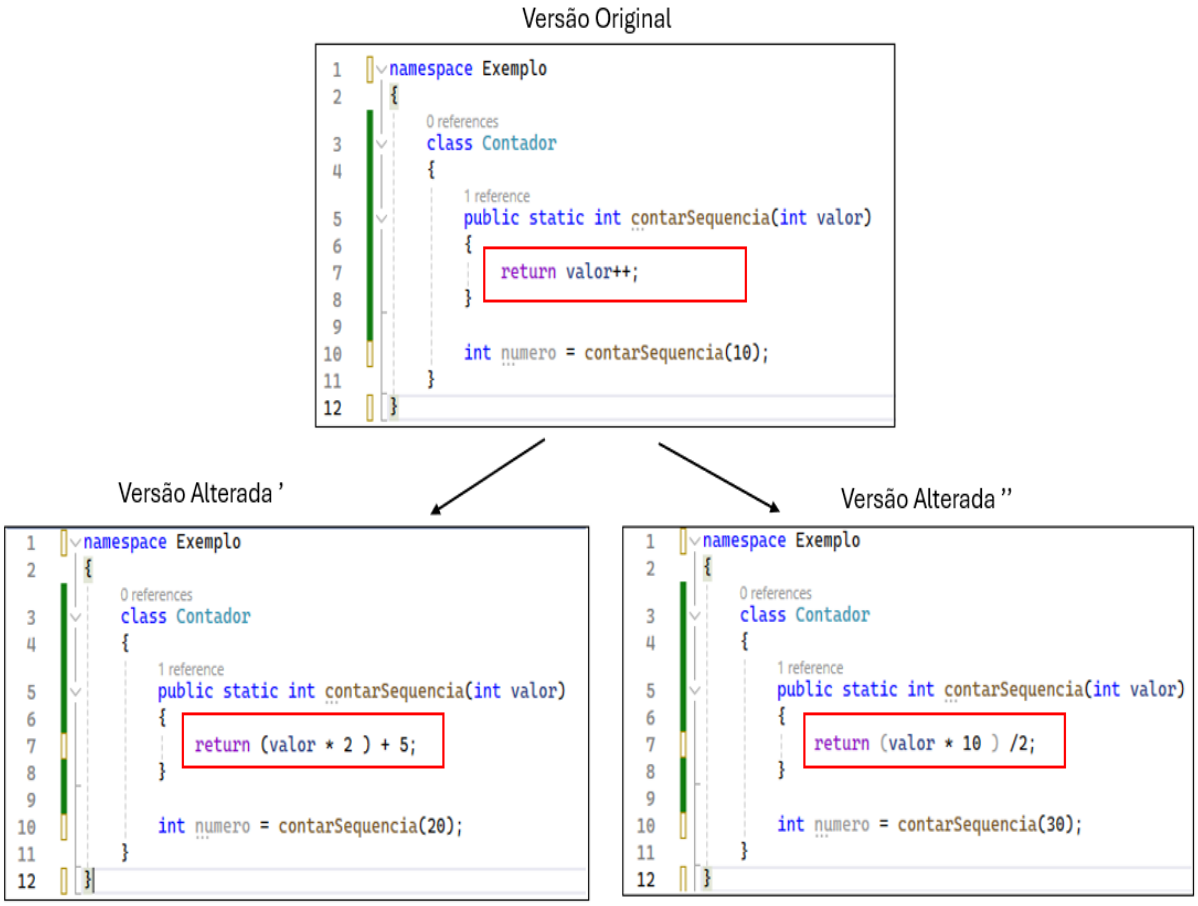
As atividades inerentes à integração de trechos de códigos-fontes executam importantes tarefas de gerenciamento das alterações realizadas. Auxiliando os desenvolvedores durante a inclusão ou acomodação das modificações com segurança e transparência. Outro importante aspecto é o armazenamento dos históricos das modificações realizadas para possíveis e necessárias avaliações [P20,P29],(MENS, 2002). As equipes de desenvolvimento de software, distribuídas ou não, conseguem desempenhar suas atividades e executar suas tarefas em partes distinta do código-fonte, possibilitando aos desenvolvedores manter a atenção nos requisitos, aumentando a produtividade das equipes [P03]. Assim, todos os softwares são passíveis de serem realizadas manutenções durante o seu ciclo de vida com a finalidade de aplicar correções, evolução técnica ou agregando novas funcionalidades [P18,P55]. Nas últimas duas décadas, a indústria de software e a comunidade acadêmica vêm tentando desenvolver novas funcionalidades e no aprimoramento das abordagens existentes nessa importante área da engenharia de software moderna, aumento a confiabilidade e qualidade dos resultados das atividades de integração de trechos de códigos-fontes [P03],(CARBONERA et al., 2023).

Esse Capítulo tem como objetivo de apresentar a estrutura dessa pesquisa. A Seção 1.1 detalha as motivações para o desenvolvimento dessa pesquisa, fornecendo uma visão generalista dos aspectos investigados. A Seção 1.2 detalha os problemas existentes e relatados na literatura atual. A Seção 1.3 especifica as questões de pesquisas que serão exploradas. A Seção 1.4 detalha quais são os principais objetivos dessa pesquisa. A Seção 1.5 detalha a metodologia de pesquisa utilizada para serem exploradas por essa pesquisa. Por fim, a Seção 1.6 apresenta como esta dissertação encontra-se organizada.

1.1 Motivação

A partir da distribuição das equipes em escala global, a utilização da ramificação ou de branches tornou-se uma prática muito recorrente, facilitando o desenvolvimento das atividades pelos desenvolvedores[P01,P03,P08]. Contudo, quando vários desenvolvedores realizarem alterações nos mesmos trechos de códigos-fonte ou em iguais funcionalidades, ao integrar as alterações junto ao CVS, o primeiro desenvolvedor executará o processo sem interrupções. Portanto, os demais desenvolvedores passam a ter problemas, definidos tecnicamente como conflitos entre trechos de códigos-fontes [P20,P32,P63]. Em [P19,P29,P36] os autores afirmam que os conflitos ocorrem quando alterações de trechos de códigos-fontes em arquivos distintos foram acomodadas no mesmo local, e as ferramentas integração baseadas na análise de texto não conseguem realizar a integração automaticamente. A Figura 2 demonstra a ocorrência de conflito de versionamento.

As abordagens de integração de trechos de códigos-fontes inicialmente propostas eram baseadas na análise puramente textual do tipo linha a linha [P03],(MENS, 2002). Entretanto, com



Fonte: Criado pelo Autor.

Figura 1: Exemplo de Conflito de Trechos de Códigos-Fonte.

a necessidade de integrações mais precisas, foram sendo desenvolvidas novas abordagens e ferramentas que realizassem a análise sintática e semântica mais detalhadas e precisas dos artefatos [P05,P06,P07],(MENS, 2002). O principal problema está no entendimento e resolução desses conflitos gerados pelas integrações. Atualmente, existem duas classificações para os sistemas de versionamento de arquivos fontes: (1) Os que consideram os artefatos na forma de texto puro e (2) E sistemas que operam em representações de documentos de forma mais abstrata e estruturada. Segundo [P08,P09], os autores afirma que as abordagens que atuam na revisão de texto puro ainda são as mais utilizadas, pois agem de forma independente à qualquer tipo de linguagem de programação. Portanto, como parte desse processo as alterações podem ser feitas nas mesmas funcionalidades ou em iguais trechos de códigos-fonte que outros desenvolvedores, surgindo assim os conflitos de versão. Esses conflitos devem ser solucionados durante a etapa de integração [P11,P12]. Isso vem atraindo a atenção de pesquisadores e profissionais da área de engenharia de software a muito tempo, pois a solução pode ser uma tarefa demorada, complexa muito propensa a erros [P53,P70],(MENS, 2002).

Segundo [P17,P18],(SMIL PRUTCHI; SOUZA CAMPOS JUNIOR; GRESTA PAULINO MURTA, 2020), no mundo real esse cenário não é um cenário que ocorre facilmente, e tem a necessidade da interação humana com a finalidade de definir como os conflitos serão solucionados da melhor maneira possível, resultando em uma tarefa demorada e complexa, geralmente devido a grande quantidade de modificações realizadas. A evolução dos softwares de integração de arquivos fontes, a análise dos resultados tornou-se mais precisas e confiáveis [P54,P18]. Embora que tenha um forte impacto na evolução e no gerenciamento dos projetos de software, ainda é uma questão em aberto a ser investigada.

1.2 Definição do Problema

A respeito das abordagens de integração de arquivos fontes, existem dois tipos de cenários possíveis: otimista e não otimista. O cenário definido como otimista, é aquele em que vários desenvolvedores a partir de uma cópia local dos códigos-fontes, realizam modificações nos mesmos artefatos e alterando as mesmas funcionalidades ou em iguais trechos de códigos-fontes, o que resultará em conflitos durante o integração (ARMINO, 2016). Assim, é necessário a integração dos trechos de códigos-fontes alterados onde poderá ocorrer conflitos de versionamento devido a mais de um desenvolvedor realizar alterações nos arquivos fontes.

No cenário definido como não otimista, todos os desenvolvedores de uma equipe realizam alterações de trechos de códigos-fontes distintos ou a inclusão de novas funcionalidades nos arquivos fontes, não existindo assim conflitos durante a integração [P48],(ARMINO, 2016; MENS, 2002). Isso trata-se de um cenário muito pouco provável de acontecer regularmente, principalmente quando existirem equipes composta por desenvolvedores trabalhando nos mesmos códigos-fontes de forma distribuída ou em paralelo [P13],(ARMINO, 2016). Na atualidade, as ferramentas de controle e versionamento de software permitem que os desenvolvedo-

res de uma determinada equipe efetuem uma cópia dos códigos-fontes nas unidades de trabalho, onde que cada um dos desenvolvedores possa efetuar as alterações necessárias e ao final, realizem a integração das alterações com o repositório principal onde encontra-se os códigos-fontes originalmente organizados [P14,P39].

Essa pesquisa será focada no cenário não otimista. Pois, devido ao surgimento de conflitos após a integração de trechos de códigos-fontes, ações devem ser executadas para a resolução dos conflitos. Logo, de acordo com as escolhas realizadas, outros problemas podem vir à surgir, dentre eles: erros de compilação da versão existente no repositório, arquivos ou trechos de códigos-fontes faltantes entre outras possibilidades. Portanto, os problemas encontrados são abordados abaixo:

- **P–1: Carência de uma visão geral do estado da arte.** Nas últimas duas décadas, muitos estudos avaliaram o desempenho humano e das abordagens existentes na área de integração de arquivos fontes, principalmente utilizando equipes distribuídas ou em paralelo [P42],(CARBONERA et al., 2023). No entanto, existe uma questão em aberto sobre a avaliação dos estudos e como foram realizadas as avaliações dos resultados obtidos, quais estratégias de pesquisa foram aplicadas e os aspectos comportamentais avaliados, ou ainda como foi realizado a avaliação do conhecimento técnico dos desenvolvedores que os estudos consideraram [P43,P52]. Assim, por todos esses aspectos apresentados é relevante explorar as lacunas encontradas e as tendências atuais e futuras no ambiente acadêmico e na indústria de software existentes na literatura atual. Portanto, com realização de um SMS evidenciou-se a inexistência de indicadores de avaliação de resultados sobre a definição de período de execução dos experimentos, quantidade de desenvolvedores e o respectivo nível de conhecimento técnico, inexistência de indicadores de avaliação de tempo aplicados na resolução desses conflitos.

- **P–2: Ausência de desenvolvimento de novas abordagens para integração de arquivos fontes.** Ausência de desenvolvimento ou pesquisa de novas abordagens que auxiliem proativamente na resolução de conflitos, ou ainda na definição de critérios avaliativos para a análise de resultados de experimentos empíricos. Caracterizando as técnicas atuais que não auxiliam na detecção de conflitos diretos e indiretos, que não avaliam o nível de severidade dos conflitos, taxas de erros e o esforço (tempo) para a resolução proativa de conflitos. Portanto, torna-se fundamental investigar essa importante lacuna encontrada quanto ao desenvolvimento de novas abordagens de integração de arquivos fontes estabelecendo critérios robustos para a análise dos resultados obtidos.

- **P–3: Escassez de conhecimento empírico sobre o esforço (tempo), taxa de corretude e taxa de erro em tarefas de integração.** A realização de um experimento controlado com uma quantidade determinada de desenvolvedores executando atividades relacionadas a integração de códigos-fontes forneceu dados empíricos possibilitando a análise dos

resultados apresentando indicadores e boas práticas que irão auxiliar na compreensão e resolução de conflitos.

1.3 Questões de Pesquisa

Portanto, evidenciou-se a necessidade de serem propostas melhorias nas abordagens de integração de software existentes, no aprimoramento das práticas de avaliação dos resultados e no desenvolvimento de soluções ou abordagens para auxiliar as equipes a apresentarem resultados mais assertivos. As pesquisas desenvolvidas pela comunidade acadêmica e indústria de software representam um avanço importante na melhoria do processo de modo generalista. No entanto, existem várias limitações técnicas e empíricas nas abordagens utilizadas. Por fim, a questão de pesquisa principal é definida abaixo:

Questão de Pesquisa Principal: Como auxiliar desenvolvedores na detecção e na resolução de conflitos críticos para manutenção de sistemas?

Após a definição da questão de pesquisa principal, outras três questões de pesquisas foram definidas a partir da questão de pesquisa principal e que também serão investigadas por essa pesquisa, conforme definidas abaixo:

- **QP–1:** Como avaliar o estado da arte e as abordagens investigadas nas últimas duas décadas referente a integração de trechos de códigos-fonte?
- **QP–2:** Como investigar o esforço (tempo), taxa de corretude e erro utilizando informação de contexto no que diz respeito à execução de um experimento controlado em dois cenários distintos?
- **QP–3:** Como disponibilizar o conhecimento empírico adquirido a partir da realização de um experimento controlado em dois cenários distintos auxiliando na resolução de conflitos?

1.4 Objetivos

Essa Seção tem como finalidade apresentar os principais objetivos que serão investigados por essa pesquisa. Por fim, o objetivo geral está detalhado abaixo:

Objetivo Geral (OG): Propor abordagens colaborativas para auxiliar desenvolvedores na resolução proativa de conflitos gerados a partir das integração de trechos de códigos-fontes.

Portanto, com a finalidade de responder às três questões de pesquisas definidas na Secção 1.3, foram identificados três objetivos específicos (OE), derivados do objetivo geral (OG) e conforme definidos abaixo:

- **OE–1: Apresentar um mapeamento sistemático da literatura sobre integração de software.** O desenvolvimento de um SMS tem o objetivo de solucionar a falta de visão geral do estado da arte na área de integração de arquivos de código-fonte durante as duas últimas décadas, período compreendido entre 2002 e 2023. Sendo direcionados para a coleta de informações dos estudos e pesquisas desenvolvidos, apresentando os resultados obtidos e as oportunidades de pesquisas. Assim, esse objetivo será explorado no Capítulo 3.

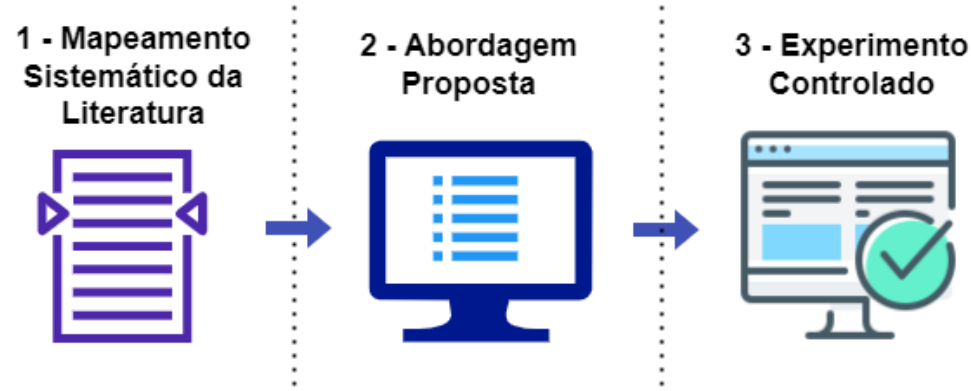
- **OE–2: Implementar uma abordagem para acompanhamento e avaliação do esforço (tempo) e da Taxa de Corretude e Erro durante integrações de trechos de códigos-fontes**. O propósito desse objetivo específico fundamentado na literatura atual, está em desenvolver novas abordagens de análise e avaliação de resultados obtidos a partir de um experimento controlado. Efetuar o acompanhamento e investigar a validade dos resultados das integrações efetuadas gerando informação de contexto, mitigando falhas e assegurando a integridade dos códigos-fontes apresentando indicadores de produtividade. Portanto, esse objetivo será explorado no Capítulo 4.

- **OE–3: Disponibilizar conhecimento empírico para auxiliar desenvolvedores na detecção e resolução proativa de conflitos.** O propósito desse objetivo específico é a produção de conhecimento empírico e estatístico quanto ao uso da *ProMerge*, visando auxiliar desenvolvedores na detecção e resolução proativa de conflitos entre trechos de códigos-fontes. Buscando investigar o impacto do tempo utilizado para detecção, compreensão e resolução de conflitos, na taxa de corretude bem como redução da taxa de erro durante a resolução dos conflitos. Logo, esse objetivo será explorado no Capítulo 5.

  Particularmente a comunidade acadêmica e a indústria de software a partir do desenvolvimento irão se beneficiar com os resultados obtidos da aplicação de novas abordagens para integração de arquivos fontes, tendo a disposição informações importantes que contribuem significativamente para essa importante área da engenharia de software moderna. Pois, serão avaliados os resultados obtidos relativos ao esforço (tempo) aplicado na resolução de conflitos, taxas de corretude e erro avaliadas após a execução de um experimento controlado apresentando indicadores que podem contribuir para o desenvolvimento de novas abordagens.

1.5 Metodologia

Essa Seção detalhada a metodologia utilizada com a finalidade de atingir os objetivos específicos apresentados na Seção 1.4. Serão investigadas detalhadamente as QP conforme definidas na Seção 1.3. A Figura 2 apresenta as metodologias utilizadas para o desenvolvimento dessa pesquisa, as quais foram aplicadas em três etapas distintas.



Fonte: Criado pelo Autor.
Figura 2: Etapas da Pesquisa.

- **Etapa 1: Mapeamento Sistemático da Literatura.** A literatura foi avaliada buscando construir uma visão ampla e irrestrita dos estudos publicados sobre integração de arquivos fontes, o método utilizado foi o mapeamento sistemático. Assim, foram identificadas tendências futuras e novas oportunidades de pesquisa dessa importante área da engenharia de software moderna. Essa etapa foi executada utilizando abordagens que são amplamente utilizadas pela comunidade acadêmica e muito bem definidas e fundamentadas em (GONÇALES et al., 2015a; WOHLIN et al., 2012).

- **Etapa 2: Abordagem Proposta.** A partir do conhecimento adquirido com a realização de um mapeamento sistemático da literatura, foi definida a *ProMerge*, que trata-se de uma abordagem de avaliação empírica que responderá ampla e irrestritamente todas as questões de pesquisas em aberto investigadas por essa pesquisa, para isso foram utilizadas metodologias muito bem definidas e fundamentadas em (GONÇALES et al., 2015a; WOHLIN et al., 2012).

- **Etapa 3: Experimento Controlado.** Foi realizado um experimento controlado seguindo diretrizes muito bem definidas em (WOHLIN et al., 2012) e coletadas as informações relativas à usabilidade e a eficácia da *ProMerge*. A avaliação consistiu em um método de análise quantitativo que investigou as seguintes variáveis dependentes: esforço (tempo), taxa de corretude e de erro. A análise dos resultados será efetuada a partir dos resultados dos experimentos controlados realizados. Por fim, serão apresentados indicadores

estatísticos e qualitativos que irão auxiliar na definição de boas práticas a serem adotadas mitigando falhas durante à integração de arquivos fontes.

1.6 Organização do Trabalho

O restante dessa pesquisa está organizada da seguinte forma: O Capítulo 2 apresenta os principais conceitos sobre integração de arquivos fontes, as abordagens existentes e as linhas de pesquisas investigadas na atualidade, quais são as variáveis consideradas e as configurações adotadas durante a execução dos experimentos. Aspectos para a formalização das equipes e seus aspectos técnicos, além dos termos onde a compreensão foi necessária para entendimento do mapeamento sistemático da literatura realizado. No Capítulo 3 foram apresentados vários aspectos técnicos, os principais trabalhos relacionados sobre integração de software nas últimas duas décadas, obtidos através de um estudo de mapeamento sistemático da literatura. O Capítulo 4 detalha o modelo e as estratégias utilizadas para avaliação dos resultados obtidos, incluindo as abordagens desenvolvidas para detecção proativa de conflitos por essa pesquisa. O Capítulo 5 avaliou a *ProMerge* e discutir os resultados obtidos e as ameaças encontradas quanto a validade e a integridade dos resultados. Por fim, o Capítulo 6 apresenta as considerações finais, limitações encontradas e os desafios apontados de desenvolvimento dessa pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Este Capítulo apresenta a fundamentação teórica necessária ao entendimento e compreensão dos demais Capítulos dessa pesquisa. Serão abordados temas importantes, além dos termos técnicos amplamente utilizados e que servem de base para o desenvolvimento das demais seções desse Capítulo. A Seção 2.1 são detalhadas as principais abordagens existentes para a integração dos arquivos de códigos-fontes. Por fim, a Seção 2.2 apresenta os principais conceitos relacionados aos sistemas de versionamento centralizados ou distribuídos.

2.1 Integração de Arquivos Fontes

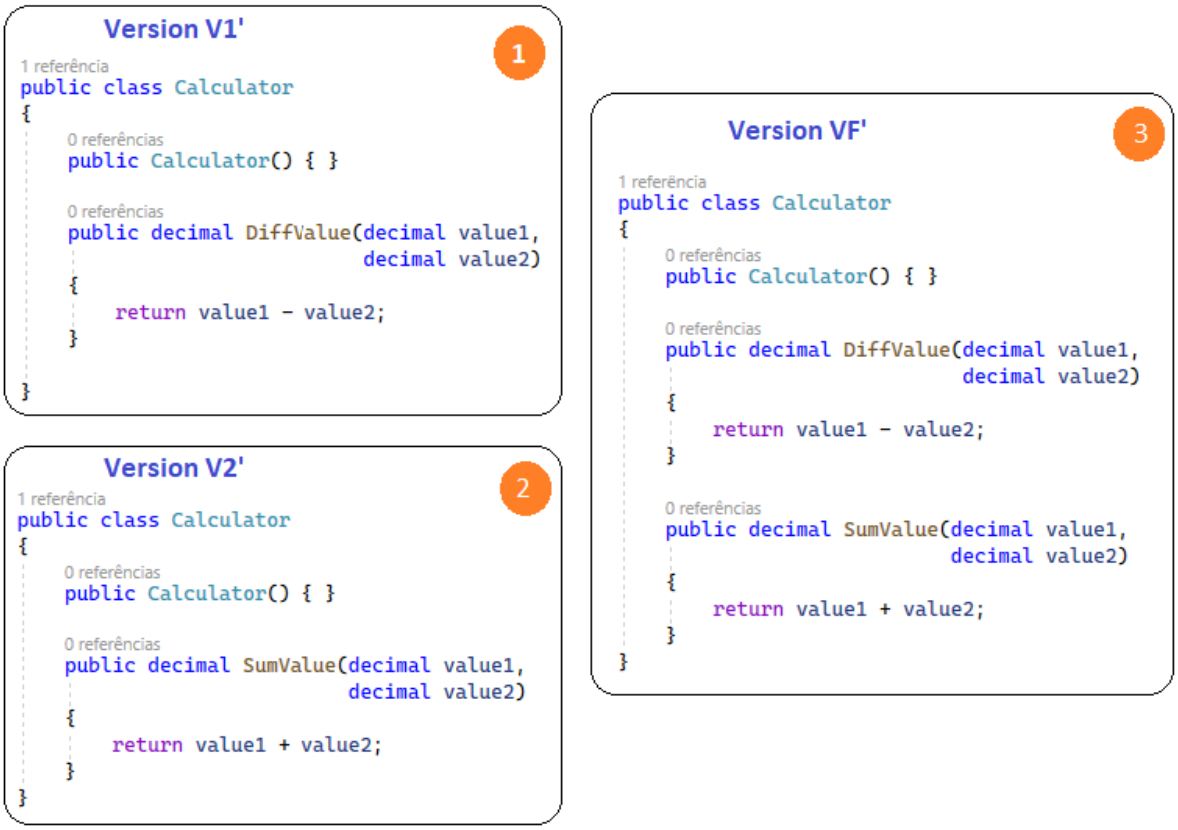
Essa Seção apresenta os principais termos utilizados ao longo dessa pesquisa. A Seção 2.1.1 apresenta os conceitos sobre os diferentes tipos de integração existentes, as Figuras 3 e 4 descrevem as principais abordagens existentes na atualidade. A Seção 2.1.2 detalha aspectos relacionados aos tipos de integração de arquivos fontes, sendo: integração textual, sintática, semântica ou estrutural. A Seção 2.1.3 aborda outro importante aspecto da integração de arquivos fontes, que são a granularidade da comparação dos artefatos. Na Seção 2.1.4 será abordado aspectos relacionados a mudança de estado dos códigos-fontes. Por fim, na Seção 2.1.5 serão analisados os dois tipos existentes de automatização da integração existentes (automática e semiautomática).

2.1.1 Tipos de Integração de Arquivos Fontes

Segundo [P03,P29,P38],(MENS, 2002), existem dois tipos de abordagens de integração de arquivos de código-fonte: Two Way Merge (Fast-Forward Merge) e o Three Way Merge, os quais são detalhados abaixo:

2.1.1.1 Two Way Merge (Fast–Forward Merge)

Sendo também conhecida como integração de duas vias ou bidirecional, em [P32],(MENS, 2002) os autores afirmam que essa foi a primeira abordagem a ser desenvolvida e compara duas versões: V1' e V2' e realiza a integração de forma única em uma nova versão denominada versão VF', sendo originada a partir das versões V1' e V2'. A comparação bidirecional não a identifica, caso uma funcionalidade tenha sido incluída nas versões V1' ou excluída da versão V2', ou ainda se uma funcionalidade ou se um elemento foi modificado em qualquer uma das versões, pelo fato de que nenhuma informação sobre as versões anteriores de V1' e V2' estivessem disponíveis no momento da integração. É importante ressaltar que a possibilidade de comparação das diferenças existentes entre as versões é muito importante para a resolução de conflitos que possam vir a ocorrer no momento da integração, conforme demostrado na Figura 3.

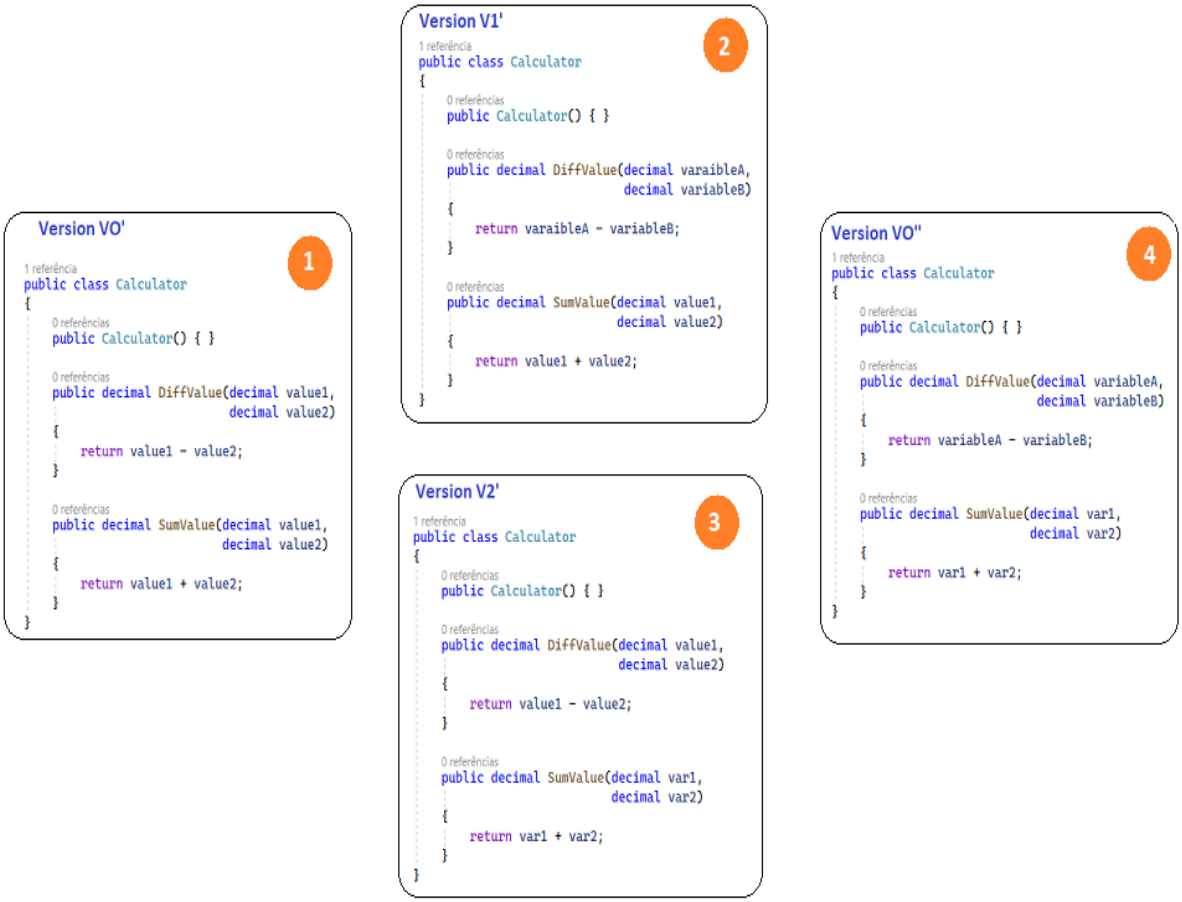


Fonte: Criado pelo Autor.

Figura 3: Two Way Merge Adaptado de [P32],(MENS, 2002).

2.1.1.2 Three Way Merge

A integração de três vias, também conhecida como Three Way Merge é definida como a evolução direta da versão da duas vias conforme detalhado na Seção 2.1.1.1. Tendo como principal característica a inclusão da versão original (VO') nas ramificações criadas em V1' e V2'. Tentando integrar as duas versões que estão sendo avaliadas ao invés de apenas combiná-las [P32],(MENS, 2002). O comparativo entre as versões é realizado a partir da avaliação das versões V1' com a sua versão ancestral VO' e da versão V2' com a sua versão ancestral VO'. A versão ancestral VO' encontra-se disponível para comparação com as versões V1' e V2'. Assim, torna-se possível identificar facilmente as alterações realizadas pelos desenvolvedores em cada uma das ramificações. Portanto, os conflitos são detectados fornecendo maior suporte à sua resolução. Logo, devido à essa evolução essa tornou-se a mais utilizada na atualidade conforme demonstrado na Figura 4.



Fonte: Criado pelo Autor.
Figura 4: Three Way Merge Adaptado de [P32],(MENS, 2002).

2.1.1.3 Estratégias de Integração

Assim, dentre as várias estratégias que os algoritmos de integração de arquivos fontes utilizam, segundo [P28,P41,P46],(MENS, 2002) a integração baseada em linha a linha do código-fonte ainda é a abordagem mais aplicada na atualidade. Contudo, ainda não se consegue aproveitar as informações da estrutura hierárquica dos códigos-fontes, e os conflitos de integração tornam-se muito comuns. Em [P41] os autores desenvolveram a ferramenta **Spork** que realiza a integração estruturada em níveis de árvore de sintaxe abstratas, reduzindo a incidência desses tipos de conflitos e segundo os autores, dentre os vários aspectos a serem melhorados, o mais importante é quanto ao tempo de execução, alteração e na padronização da formatação dos arquivos de código-fonte.

Em [P27], os autores afirmam que a ferramenta de versionamento **GIT** tem sua enorme popularidade em partes devido ao desenvolvimento de novos padrões de usabilidade, que auxiliam os desenvolvedores a serem mais eficientes em suas tarefas. Entretanto, as mudanças resultantes nos comportamentos dos seus usuários e na forma de como são armazenadas as alterações afetam diretamente os procedimentos de mineração e análise de dados, e podem levar à incapacidade de auditar alterações dos arquivos de código-fonte. Em [P48], os autores apresentam a ferramenta **SoManyConflicts**, que é uma extensão independente de linguagem (realiza a análise linha à linha), com a finalidade de auxiliar os desenvolvedores nas tarefas de integração. O principal deles está a capacidade de resolução de conflitos relacionados à clusteres baseado na conectividade de grafos, e por conseguir sugerir estratégias de resolução dos conflitos a partir em outros exemplos já resolvidos.

2.1.2 Aspectos de Integração (Textual, Sintático, Semântico e Estrutural)

Segundo [P04,P06,P69],(MENS, 2002), outro importante tipo de avaliação entre as ferramentas de integração é como os arquivos de código-fonte são representados. E conforme os autores, a melhor alternativa está em utilizar ferramentas que estruturem os arquivos de código-fonte através de arvores de aspectos, assim toda a estruturação hierárquica do artefato será mantida e considerada durante a avaliação e integração, que podem ser assim detalhadas:

Integração Textual. Esse tipo de integração avalia os códigos-fontes apenas como arquivos de texto ou binários, a integração é baseada em linhas de texto puro e que são consideradas como unidades indivisíveis. As linhas de texto podem ser detectadas em modificações paralelas, bem como as linhas que foram incluídas, excluídas, modificadas, alteradas ou reposicionadas. Outro aspecto importante, é a baixa granularidade desta técnica, tendo como desvantagem de não conseguir integrar duas linhas conflitantes, mas apenas realizar a integração entre elas, onde apenas uma das duas linhas modificadas será a selecionada (última integrada). No entanto, ainda é a abordagem mais utilizada na atualidade, devido a sua grande eficiência e por ser independente de linguagem de desenvolvimento. Em [P32],(MENS, 2002) so autores afirmam que

aproximadamente 90% dos arquivos podem ser integrados através da integração textual, e os 10% restantes necessitam de abordagens mais automatizadas, pois as ferramentas de integração textual não consideram nenhuma informação de nível sintático ou semântico dos artefatos.

Integração Sintática. Trata-se de uma abordagem mais evoluída em relação às abordagens de integrações textuais, pois considera a árvore sintática dos artefatos de software. Os conflitos gerados pela integração textual, como por exemplos comentários inseridos ou excluídos no desenvolvimento ou manutenção dos códigos-fontes, podem ser totalmente ignorados através da integração sintática [P09,P10,P69]. Os conflitos serão apresentados quando os resultados da integração não estiverem sintaticamente corretos, por exemplo: se uma determinada funcionalidade for alterada ao mesmo tempo por diferentes desenvolvedores, mesmo que o resultado seja igual, mas sintaticamente divergentes, um conflito será apresentado para que ações sejam executadas na sua resolução. Portanto, para os autores as abordagens são categorizadas de acordo com a estrutura de dados subjacente dos artefatos, existindo as abordagens que se utilizam de árvores de hierarquia (como estrutura de dados) e as que utilizam abordagens de grafos.

Integração Semântico. Conflitos semânticos são causados quando desenvolvedores realizam alterações em diferentes artefatos que são dependentes entre si, por exemplo, uma variável que possui referência em vários trechos de um arquivo de código-fonte ou ainda, uma funcionalidade que foi realizada alteração da sua assinatura em um determinado artefato [P04,P06] e que possui várias referências. A análise sintática não detecta esse tipo de erro, pois o artefato de software está sintaticamente correto. O problema será detectado quando o compilador da linguagem for requisitado, que e irá apresentar uma mensagem de erro de variável ou da funcionalidade não estiver corretamente declarada. Logo, as abordagens baseadas em análise de árvore sintática não conseguem detectar esses tipos de conflitos, devido a relação entre declaração de um procedimento e a sua invocação não serem explicitas.

Assim, as abordagens de integração baseadas em grafos conseguem detectar esses tipos de conflitos devido a representação sintática manter a relação entre a definição e as requisições conseguindo verificar incompatibilidades. Por fim, segundo os autores os conflitos comportamentais podem surgir devido à essa abordagem, e não se consegue assegurar quanto ao comportamento da execução do programa em comparação ao comportamento dos arquivos de código-fonte avaliados. Portanto, esse tipo de problema poderá ser resolvido através de abordagens de integração semânticas mais sofisticadas e que dependem da avaliação da semântica do código-fonte em tempo de execução.

Integração Estrutural. Esse tipo de integração ocorre quando um artefato de software passa por uma modificação organizacional ou de reestruturação, o algoritmo responsável pela integração não consegue decidir qual a melhor forma de realizar a integração estruturada desse artefato. Na atualidade, a maioria das ferramentas e abordagens de integração não oferecem nenhum suporte de detecção desse tipo de conflito, deixando mais uma lacuna em aberto para o desenvolvimento de novas pesquisas pela comunidade acadêmica e pela indústria de software. As informações necessárias para definição do tipo de conflito que está ocorrendo geralmente

são implícitas ao software utilizado, e não foi identificado nenhuma opção para configuração dos algoritmos possíveis de serem utilizados. Portanto, para os autores não foi possível realizar nenhuma inferência ou qualquer tipo de avaliação pelo usuário através do código-fonte desse artefato integrado.

2.1.3 Tipos de Granularidades

Foi possível identificar os diferentes níveis de granularidades de acordo com as abordagens de integração, investigadas o nível de detalhamento da análise quanto a detecção de conflitos. Em [P32],(MENS, 2002), os autores afirmam que a melhor análise consiste na utilização de caracteres únicos (ou bytes) como unidades indivisíveis. No entanto, isso tornaria qualquer algoritmo de comparação muito ineficiente na prática, outra maneira seria a de utilizar substrings ou blocos de bytes. As abordagens de integração sintática também podem ter diferentes níveis de granularidade, dependendo da quantidade de informações levadas em consideração. Segundo [P28,P38,P41] a ferramenta de integração eficiente deverá ter a capacidade de realizar o integração de arquivos fontes em qualquer nível de granularidade. Assim, quando for avaliado a eficiência, será necessário restringir a quantidade e o nível de detalhamento das informações que do contrário o tempo de análise torna-se muito elevado.

2.1.4 Técnicas de Integração Baseadas em Estado, Mudanças e Operações

Outro aspecto importante a ser avaliado, é o integração de arquivos fontes baseado em alteração de estados ou baseada em mudanças de artefatos. Em [P32],(MENS, 2002), os autores afirmam que a integração baseada em mudança de estados não considera as informações da versão original e de suas ramificações, e utilizam algoritmos do tipo Two Way Merge durante a integração, realizando apenas avaliações bidirecionais. A integração baseada em mudanças da árvore de objetos avalia a evolução histórica dos artefatos durante a integração, utilizando algoritmos do tipo Three Way Merge. Assim, trata-se de um caso especial da integração baseada em mudanças, pois apresenta as alterações como operações. A sequência é conhecida como histórico de mudanças, podendo melhorar a detecção e resolução dos conflitos. Em [P04,P06,P69] os autores afirmam que a integração baseada em operações, tem como característica a capacidade de detecção de erros sintáticos, estruturais e vários tipos de erros semânticos em relação as abordagens baseadas em mudanças de artefatos. Por fim, essas abordagens facilitam o desenvolvimento de mecanismos de múltiplos desfazer e refazer, que consiste em aplicar as últimas operações integradas ou aplicar as últimas alterações realizadas.

2.1.5 Grau de Automatização

Segundo [P39,P52,P53,P62] o grau de automatização pode ser representado dois tipos: (1) Semi automatizado e (2) Automatizado. O primeiro necessita da iteração humana apenas a partir da ocorrência de pontos conflitantes, o usuário deverá escolher como resolver os conflitos detectados. Em [P43],(MENS, 2002) os autores afirma que os conflitos são detectados pelos softwares utilizados para a integração dos códigos-fontes, onde na maioria dos casos a resolução desses conflitos ainda requerem a iteração humana. O segundo caracteriza-se por não ter a necessidade da iteração humana durante a integração dos códigos-fontes, uma vez que os conflitos existentes são resolvidos automaticamente pelo algoritmos responsáveis pela integração dos arquivos fontes, essas abordagens caracteriza-se como o procedimento mais rápido e preciso existente na atualidade.

Em [P62] ou autores propuseram um mecanismo para integração automática que reduz a carga de trabalho dos desenvolvedores na resolução dos conflitos de integração comportamental. Sendo apresentada uma técnica automatizada que corrige automaticamente automática as falhas expostas pelos testes unitários realizados. Produzindo os programas iniciais a serem corrigidos combinando os membros de classe dentro do trecho de código-fonte a ser avaliado. Portanto, a técnica proposta absorve os fragmentos de códigos-fontes dentro de arquivos fontes para serem integrados. Os resultados experimentais apresentados demonstraram o sucesso da técnica proposta na resolução de conflitos comportamentais sem a necessidade de intervenção humana.

Em [P67] os autores investigam as causas e efeitos do merge automatizado, e chegaram na conclusão de três importantes aspectos. (1) os conflitos textuais dos mesmos tipos nos commits realizados geralmente eram resolvidos a partir da mesma estratégia de resolução, mesmo que sendo possível prever a estratégia dos desenvolvedores. Assim, se a estratégia adotada for a melhor, a resolução tenderá a ser a mais precisa e correta. (2) a integração baseada em texto poderá produzir conflitos de ordem superiores quando integrados automaticamente alterações semanticamente conflitantes. Isso configura a eminente necessidade de ferramentas melhores auxiliando na detectação de todos os tipos de conflitos, em vez de detectar apenas poucos tipos de conflitos baseados em conflitos semelhantes. E por fim, (3) os conflitos de ordem superior geralmente são resolvidos pelos desenvolvedores aplicando consistentemente alterações semelhantes em locais muito semelhantes, reforçando a adoção de um padrão técnico pelos vendedores. Assim, futuramente as ferramentas poderão resolver vários tipos de conflitos de ordem superior.

2.1.6 Conflitos Entre Arquivos Fontes

Essa Seção tem como finalidade definir o que são conflitos entre arquivos de códigos-fontes. Segundo [P09], são alterações simultâneas de códigos-fonte podem introduzir problemas pró-

prios durante a integração, muitas vezes manifestando-se como conflitos de integração. Ocorrendo durante a integração de alterações feitas por um ou mais desenvolvedores no mesmo trecho de códigos-fonte. Profissionais e pesquisadores procuram minimizar o número de conflitos de integração, pois resolvê-los é uma tarefa difícil, demorada e muitas propensa a erros. Para [P35] os conflitos de integração são um aspecto inevitável e disruptivo do desenvolvimento colaborativo de software. Em [P14], os autores afirmam que embora duas simples alterações simultâneas possam ser individualmente corretas, a sua combinação pode tornar-se inconsistentes. Esses conflitos exigem escolhas conciliadoras e esses cenários necessitam de muito empenho e tempo na sua resolução. Segundo [P18] a maioria dos commits conseguem ser integrados normalmente, alterações paralelas podem ser sobrepostas levando a conflitos de integração.

Em [P46] ao autores afiram que o período compreendido entre a detecção, compreensão e resolução, os conflitos podem evoluir e tornarem-se muito difíceis de serem resolvido. Assim, os desenvolvedores podem evitar a integração temendo que os conflitos possam ser difíceis de resolver. Para [P18] solucionar conflitos de integração não se caracteriza como uma tarefa trivial, especialmente quando as alterações divergem significativamente dificultando a sincronização. Os sistemas de versionamento são ferramentas populares que suportam trabalho paralelo em projetos compartilhados e oferecem suporte para sincronização de alterações paralelas nesses projetos [P46]. O processo de resolução de conflitos poderá ser tedioso e demandar muito tempo para definir a melhor abordagem. Portanto, resoluções de conflitos de integração de trechos de código-fonte executadas erroneamente ocasionam erros de integração, interrupções no fluxo de desenvolvimento das equipes comprometendo a eficiência e os prazos de um projeto. A Seção 2.1.6.1 detalha algumas abordagens quanto a detecção de conflitos. Por fim, a Seção 2.1.6.2 detalha também algumas abordagens direcionadas à resolução de conflitos.

2.1.6.1 Detecção de Conflitos

Nesta seção serão discutidas algumas abordagens quanto a detecção de conflitos e serão identificados quais são os principais tipos de conflitos que podem ser evidenciados.

- **Matriz de Integração.** Muitas ferramentas de integração utilizam uma abordagem ad hoc na detecção de conflitos. A partir de uma abordagem baseada em operações, os conflitos são detectados mais facilmente, avaliando as alterações que foram aplicadas em paralelo por diferentes desenvolvedores. Assim, todos os pares de operações que levam a uma inconsistência são agrupados na tabela de conflitos ou matriz de integração. Isso torna possível detectar os conflitos de integração realizando uma simples consulta de tabela [P17,P19].

  Segundo [P35] é possível a utilização de um formalismo independente de domínio, onde os artefatos de software alterados são representados como elementos gráficos independentes. A evolução das alterações são representadas pelas reescritas e os gráficos de tipo são usados para detalhar as restrições de domínio. Assim, baseando-se em propriedades

formais de reescrita de grafos, pode ser definido formalmente quais pares de operações de modificação produziram conflitos de integração sintática, definindo uma caracterização formal de domínio.

- **Conjunto de Conflitos.** Os conjuntos de conflitos são utilizados no contexto de aplicações colaborativas, que agrupam combinações de operações potencialmente conflitantes tendo como base a semântica fornecida. Dependendo do tipo de aplicação, os tipos de operações e os conflitos de integração associados podem ser divergentes. Sendo definidos estaticamente, permanecendo fixos enquanto a semântica da aplicação não for alterada [P39]. Assim, as operações que pertencem ao mesmo conjunto de conflitos podem causar conflitos quando integradas, e qualquer tipo de operação pode participar em múltiplos conjuntos de conflitos. Portanto, abordam a escalabilidade, pois estão restritas a quantidade de operações que serão considerar ao investigar os conflitos detectados [P34].

- **Conflitos Semânticos.** Essa categoria de conflito utiliza como base a semântica da linguagem de programação utilizada em atribuições muito simples. Segundo (ARMINO, 2016) o resultado de uma integração de código-fonte realizada erroneamente pode acarretar erros de compilação, tanto sintáticos como semânticos que se identificados antecipadamente podem gerar menos retrabalhos para os desenvolvedores. Em [P04,P06], os autores afirmam que significado das informações dos grafos devem ser compartilhados, e a ontologia adotada pela linguagem de desenvolvimento deve ser utilizadas como uma interlíngua para mapear os conceitos utilizados pelas diferentes aplicações, permitindo a correta interpretação dos trechos de código-fonte.

- **Conflitos Diretos e Indiretos.** Em (ARMINO, 2016) o autor apresentou os conceitos para resolução proativa de conflito direto e indireto em arquivos fontes. O conflito direto ocorre quando mais de um desenvolvedor realizar alterações em iguais trechos de códigos-fonte no mesmo arquivo, notificando o desenvolvedor antecipadamente da possível ocorrência de conflitos. A deteção proativa evidência que quando a integração dos trechos de códigos-fontes for realizada, irá ocorrer conflitos entre as diferentes versões existentes integradas. O conflito indireto é caracterizado pela dependência hierarquia das funcionalidades, se uma funcionalidade possuir em seu código-fonte referência à outra(s) funcionalidade(s). Quando a funcionalidade principal for alterada por desenvolvedor e alguma da(s) funcionalidades referenciadas nessa funcionalidade for alterada por outro desenvolvedor o conflito indireto será detectado, pois a funcionalidade subjacente alterada e que está referenciada na funcionalidade principal poderá apresentar algum erro semântico.

2.1.6.2 Resolução de Conflitos

As abordagens para resolução para os diferentes tipos de conflitos podem variar desde um processo manual e muitas vezes demorado propenso a erros até a um processo interativo que o algoritmo requer a interação com o desenvolvedor, ou por fim uma abordagem de resolução automatizada. Isso depende do tipo de conflito ocorrido e do nível de precisão necessário, podendo ser necessárias diferentes estratégias para resolução [P09]. Em [P62] os autores afirmam que a resolução automática de conflitos refere-se a situações em que alterações paralelas em iguais arquivos fontes mas em trechos de códigos-fonte diferentes, nesses casos a integração será realizada automaticamente. Assim, um caso particular de conflito ocorre quando duas ou mais alterações realizadas em paralelas necessitam ser integradas numa determinada ordem, pois se forem aplicadas na ordem inversa dará origem a uma inconsistência.

Segundo(MENS, 2002) , essas situações acontecem normalmente nos casos em que um arquivo fontes for renomeado, sendo uma operação global podendo afetar outros trechos em diferentes locais no código-fonte, que deverá ser aplicada depois de qualquer outro tipo de modificações paralelas garantindo que as referências dos demais artefatos renomeado também sejam alteradas. Portanto, trata-se de uma particularidade em que é necessário definir a ordem de execução. Outra particularidade acontece quando iguais trechos de códigos-fonte podem ser acomodado ou removido duas vezes. Esse tipo de conflito poderá ser resolvido apenas ignorando uma das duas modificações.

Outro exemplo de conflito que não pode ser solucionado automaticamente, acontece quando uma funcionalidade for renomeada diferentemente em alterações paralelas. O algoritmo de integração não possui autonomia para decidir automaticamente qual das alterações é a mais apropriada. Como alternativa à tais situações, o desenvolvimento de uma abordagem para fornecer assistência automatizada para negociar a resolução de conflitos. Assim, outro aspecto importante refere-se a consolidação, utilizada quando duas revisões de uma versão base necessitam serem integradas existindo o conhecido que a maioria das alterações paralelas são complementares, para os casos de exclusões, a integração prossegue automaticamente. Assim, para os casos de alterações sobrepostas, uma das alterações deverá ser a escolhida interativamente pelo desenvolvedor [P36,P38,P49].

2.2 Sistemas de Versionamento de Arquivos Fontes

Durante as etapas de desenvolvimento de um projeto de software, serão criadas várias versões dos códigos-fontes pelos desenvolvedores que podem atuar em uma ou mais atividades. Pois, entre as atividades realizadas poderá ser necessária a correção de erros, atualização ou a criação de novas funcionalidades ou até de novos versionamentos para realização de atividades especificas isoladamente. Gerenciar e registrar a evolução de um projeto e suas atividades caracteriza-se como as principais atividades de uma ferramenta de versionamento de códigos-

fontes. Isso viabiliza o desenvolvimento das tarefas e da atualização dos seus status, fornecendo um histórico das alterações realizadas que são essenciais ao desenvolvimento das atividades de maneira distribuída ou em paralelo [P63],(ARMINO, 2016; MENS, 2002). Segundo [P13,P35] dentre as várias ferramentas existentes na atualidade, pode-se apresentar dois tipos de ferramentas: Sistemas de versionamento de arquivos fontes centralizados e distribuídos.

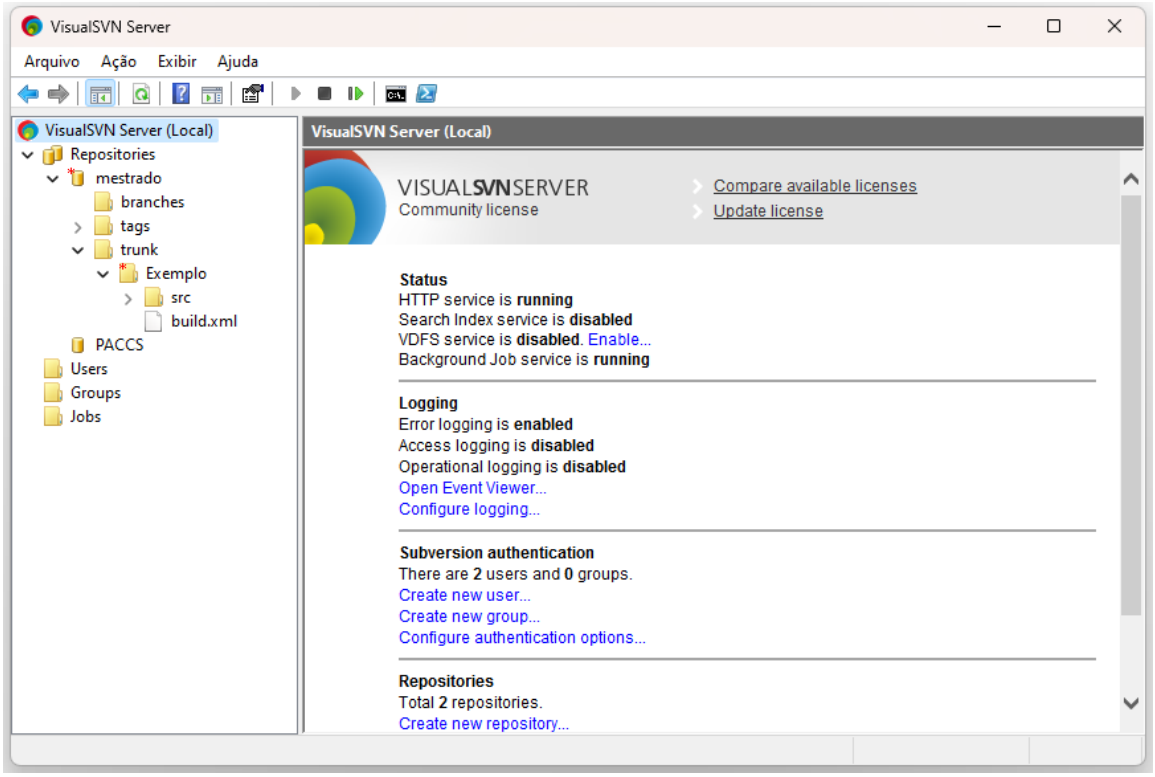
2.2.1 Sistemas de Versionamento de Arquivos Fontes Centralizados

Os sistemas de versionamento centralizados caracterizam-se por apresentarem dois segmentos: (1) Servidor – Software responsável pelo gerenciamento e pela organização estrutural dos códigos-fontes como uma unidade única, onde todas as operações de versionamento serão realizadas, entre as quais pode-se citar: update, merge, delete e commit e, (2) Cliente – Software instalado em cada um dos clientes com acesso ao servidor, com a finalidade de realizar o versionamento e a integração dos trechos de códigos-fontes modificados na cópia local (workspace) do desenvolvedor. A Figura 5 exemplifica o repositório principal do sistema gerenciador de códigos-fontes. A finalidade de sistemas de versionamento centralizados está em disponibilizar os códigos-fontes através de um sistema gerenciador para todos os seus clientes com acesso ao repositório central, onde a cada um dos seus clientes é permitido realizar uma cópia local (na estação de trabalho) dos códigos-fontes para realizar as alterações necessárias e, que possam ser novamente integrados com o repositório central. A Figura 6 exemplifica o cliente do sistema gerenciador de códigos-fontes instalado nas estações de trabalho dos desenvolvedores.

Na atualidade, esse modelo de versionamento ainda é muito utilizado, devido a algumas vantagens: (1) Gerenciamento centralizado do projeto – Aspecto muito importante na segurança e proteção contra tentativas de invasões ou furto de códigos-fontes e, (2) Bloqueio dos códigos-fontes – Permitindo que apenas um desenvolvedor da equipe possa realizar alterações por vez. Desvantagens: (1) Dependente de conexão com o repositório para poder realizar o versionamento dos códigos-fontes e, (2) Necessidade de instalação do software cliente em cada uma das estações de trabalho e do software servidor para conseguir disponibilizar acesso aos códigos-fontes pelos seus clientes. Como exemplos de sistemas de versionamento centralizados pode-se citar: TortoiseSVN (TORTOISESVN, 2024), Mercurial SCM (MERCURIALSCM, 2024) e Subversion (SUBVERSION, 2024). Sendo o primeiro deles, objeto de avaliação dessa pesquisa. A partir da cópia local dos códigos-fontes são permitidas várias operações, a Figura 7 detalha as operações possíveis de serem realizadas.

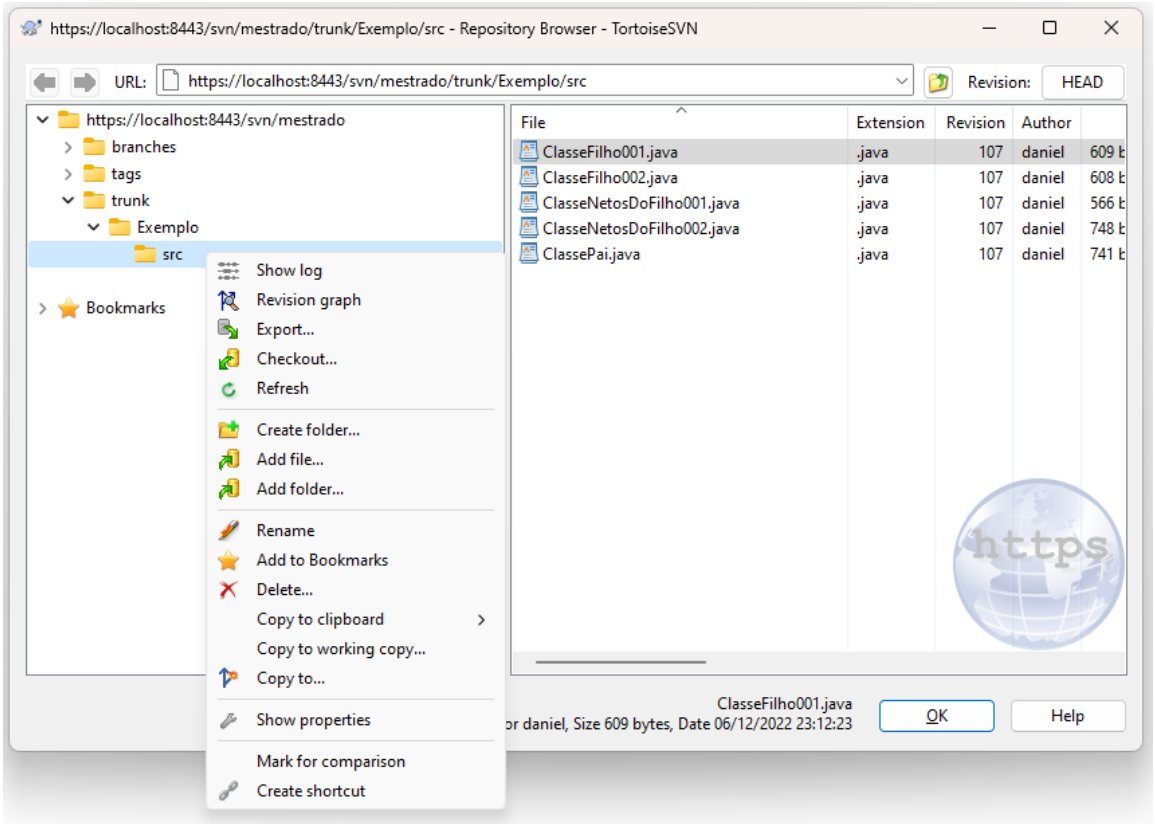
Abaixo estão definidos os principais termos técnicos (em inglês) utilizados pelos sistemas de versionamento de arquivos fontes centralizados e que foram utilizados por essa pesquisa:

- **Checkout:** Realiza uma cópia local dos arquivos ou projeto existentes no repositório.
- **Update:** Atualiza os arquivos ou projeto com a última versão dos arquivos no repositório.
- **Commit:** Envia as alterações realizadas na cópia local dos códigos-fontes ao repositório.



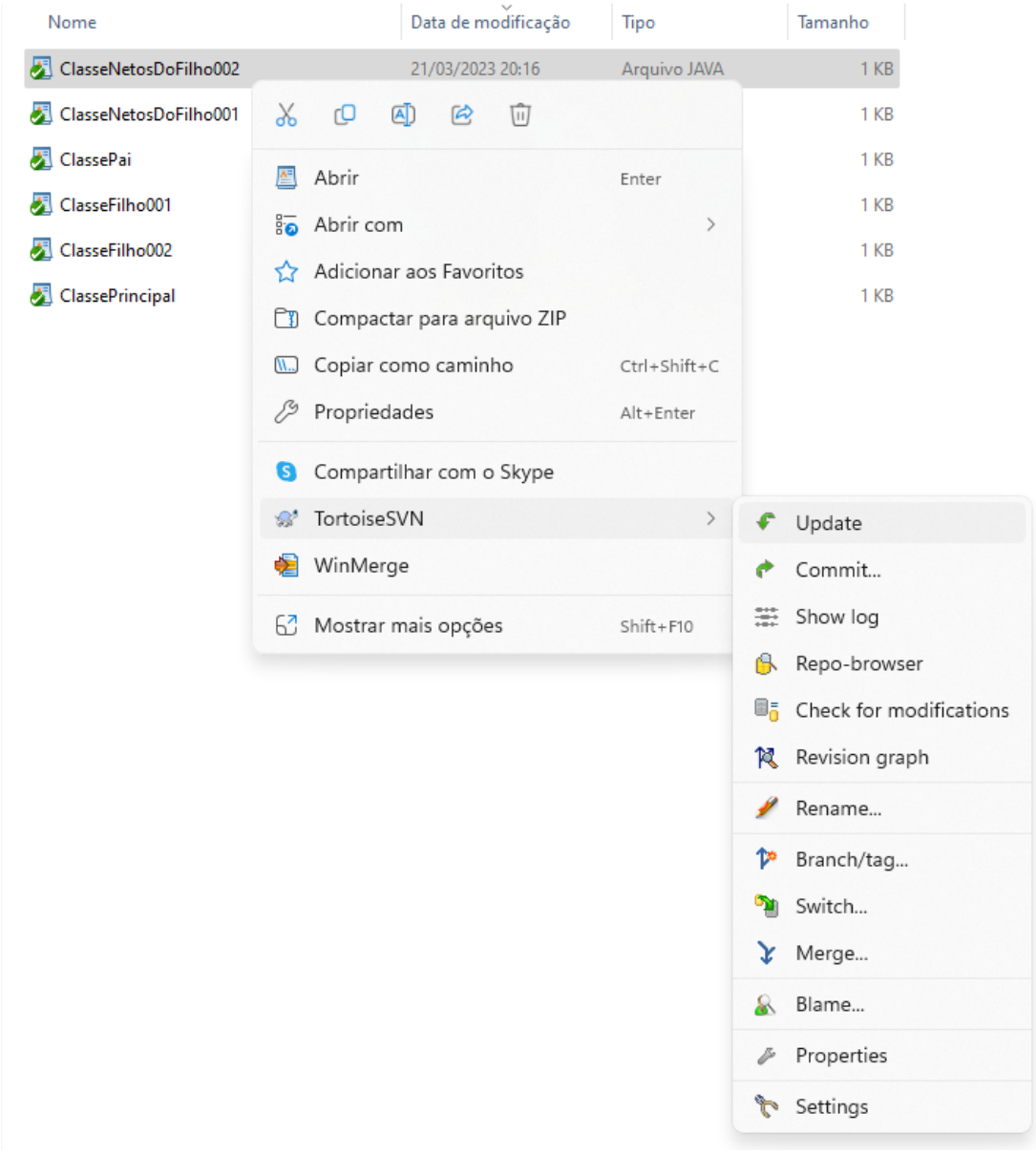
Fonte: Criado pelo Autor.

Figura 5: Sistema de Versionamento de Arquivos Fontes Centralizado.



Fonte: Criado pelo Autor.

Figura 6: Cliente do Sistema de Versionamento de Arquivos Fontes Centralizado.



Fonte: Criado pelo Autor.

Figura 7: Comandos do Sistema Centralizado de Versionamento.

- **Branch/Tag:** Ramificações criadas a partir dos códigos-fontes existentes no repositório principal.

- **Diffs:** Apresenta as diferenças nos códigos-fontes da cópia local com os códigos-fontes do repositório.

- **Merge:** Realiza a integração das alterações locais com a versão do repositório, quando existir conflitos os mesmos devem ser solucionados com a participação de integrantes da equipe e, na inexistência de conflitos a integração é automática.

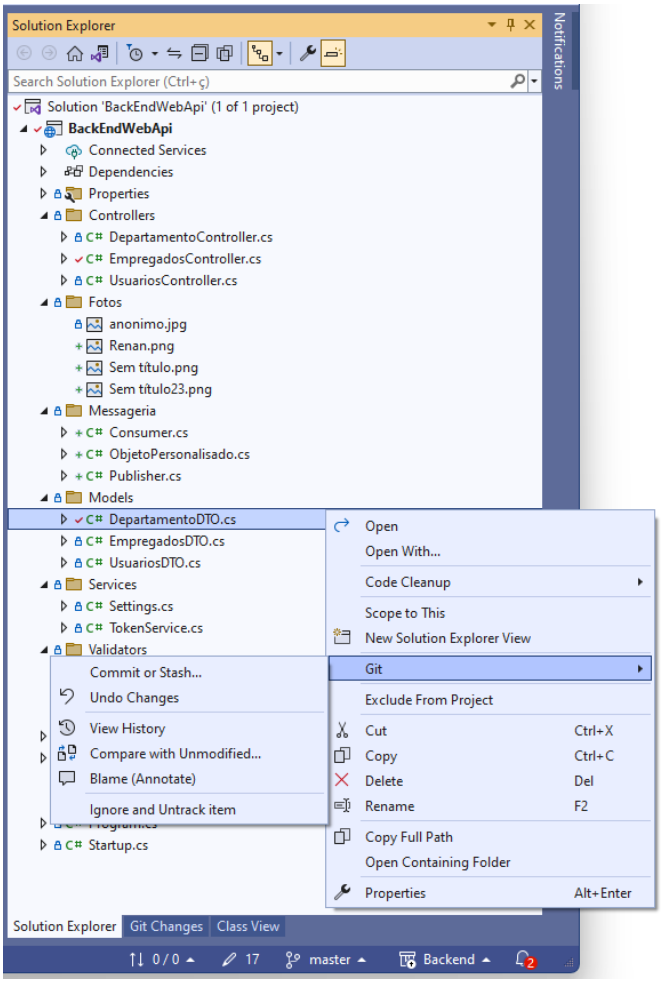
- **Revert:** Cancela as alterações e sobrescreve a versão local dos códigos-fontes com a existente no repositório.

2.2.2 Sistemas de Versionamento de Arquivos Fontes Distribuído

Os sistemas de versionamento de arquivos fontes distribuídos diferente dos sistemas de versionamento de arquivos fontes centralizados, tem a possibilidade de os desenvolvedores permanecerem desconectados do repositório central a maior parte do tempo ou durante o desenvolvimento de suas atividades. Quanto a maioria das operações, é possível que elas sejam realizadas localmente e de modo independente, uma vez que o repositório de trabalho também está localizado na própria estação de trabalho. Dentre as várias ferramentas desse segmento, pode-se destacar: (1) TFS – Team Foundation Server (TFS, 2024) e, (2) GITHub (GITHUB, 2024). O TFS caracteriza-se por ser uma ferramenta híbrida, podendo ser utilizada em equipes centralizadas ou distribuídas, e não apresenta nenhuma limitação técnica quanto ao crescimento de uma equipe e, possui muitas características relacionadas com a aplicação das metodologias ágeis nas suas funcionalidades.

O Git destaca-se como a ferramenta de versionamento de arquivos fontes mais utilizada na atualidade, devido em grande parte pela enorme facilidade de utilização do GITHub (GITHUB, 2024), que configura-se como uma plataforma de hospedagem e gerenciamento de arquivos fontes e projetos na web. Assim, o Git passou a ser utilizado como uma opção de ferramenta com conexão nativa para o versionamento de arquivos de códigos-fontes para várias ferramentas e ambientes de desenvolvimento de software. A Figura 8 demonstra um exemplo da sua utilização no versionamento de arquivos de códigos-fonte.

Como vantagens na utilização de sistemas distribuídos pode-se citar: (1) Independência de acesso ao repositório central, (2) Maior rapidez no gerenciamento dos códigos-fontes, (3) Indisponibilidade do repositório central, assim qualquer desenvolvedor da equipe poderá, a partir da cópia local dos códigos-fontes, reestabelecer novamente o repositório central e por fim, (4) Utilização de múltiplas branches possibilitando a alternância de desenvolvimento com o gerenciamento de cada ramificação dos códigos-fontes alterados separadamente, não interferindo no desenvolvimento e evolução do projeto.



Fonte: Criado pelo Autor.

Figura 8: Sistema de Versionamento de Arquivos Fontes Distribuído.

Dentre as desvantagens, pode-se citar: (1) Clonagem inicial (cópia local) do projeto poderá ser mais demorada, devido a inexistência dos códigos-fontes na estação local de trabalho, (2) Os códigos-fontes devem ser sincronizados remotamente através dos comandos push e pull – o que poderá gerar conflitos de integração resultantes das alterações dos códigos-fontes e por fim, (3) Avaliação da integridade do projetos – isto ocorre devido as escolhas feitas para a resolução dos conflitos ou a partir da integração realizada com erros sintáticos, isto acontece devido à comparação meramente textual dos códigos-fontes sem considerar a estrutura hierárquica dos artefatos.

Esse Capítulo apresentou o contexto histórico sobre integração de arquivos fontes, fundamentando os principais conceitos sobre o tema, passando pelas principais abordagens existentes, e detalhou os principais aspectos relacionados aos tipos de análise para a integração dos códigos-fontes. No Capítulo 3 foi detalhada a metodologia de pesquisa adotada que avaliou uma amostra de setenta estudos publicados durante as últimas duas décadas utilizando uma metodologia sistemática muito bem estabelecida em (CARBONERA; FARIAS; BISCHOFF, 2020; BISCHOFF et al., 2019; AHMAD; BABAR, 2016; GONÇALES et al., 2015b). Por fim, desenvolvimento de novas pesquisas futuras na área de integração de arquivos fontes.

3 MAPEAMENTO SISTEMÁTICO DA LITERATURA

Este Capítulo tem como finalidade realizar um mapeamento sistemático da literatura na área de integração de arquivos fontes, respondendo a QP–1 conforme detalhada na Seção 1.3. Na literatura atual, foram realizadas muitas publicações dessa importante área da engenharia de software moderna nas últimas duas décadas. Portanto, ainda existem muitas questões em aberto para serem investigadas por essa importante área da engenharia de software moderna. Foi evidenciado que existem duas linhas pesquisas principais: (1) Desenvolvimento de novas abordagens ou o aprimoramento das existentes utilizadas para integração de arquivos fontes e, (2) Avaliação dos resultados obtidos a partir da execução de experimentos empíricos. Na segunda linha de pesquisa, ainda existem lacunas quanto a formalização de uma base de conhecimento ampla e detalhada sobre as características avaliadas pelas pesquisas e experimentos realizados. Assegurando que os indicadores obtidos possam ser utilizados para a avaliação do nível de acurácia de integrações realizadas ou ainda dos resultados obtidos através da avaliação da realização de um experimentos controlado por tempo determinado entre outras possibilidades. O mapeamento sistemático da literatura foi realizado considerando diretrizes e orientações amplamente utilizadas e validadas anteriormente em (CARBONERA; FARIAS; BISCHOFF, 2020; BISCHOFF et al., 2019; GONÇALES et al., 2015b; COSTA; MURTA, 2013; PETERSEN; VAKKALANKA; KUZNIARZ, 2015; KITCHENHAM et al., 2009).

A Seção 3.1 apresenta a metodologia aplicada por essa pesquisa para seleção dos estudos avaliados por essa pesquisa. A Seção 3.2 detalha o protocolo de filtragem e a seleção dos estudos primários, e refere-se aos estudos de pesquisa geralmente publicado em periódicos de relevância e revisado por pares, onde são detalhados os principais aspectos analisados, métodos aplicados e avaliados os resultados obtidos através de experimentos empíricos. A Seção 3.3 detalha os resultados obtidos e avalia os principais achados. A Seção 3.4 detalha algumas direções obtidas e os trabalhos futuros a partir desses apontamentos. A Seção 3.5 detalha os cuidados adotados que foram considerados durante o desenvolvimento de um SMS. A Seção 3.6 detalha um conjunto de oportunidades que possam vir a ser exploradas futuramente. Por fim, a Seção 3.7 apresenta consideração obtidas com a realização de um SMS direcionando os trabalhos futuros a partir das conclusões obtidas.

3.1 Seleção dos Estudos

Essa Seção tem como finalidade detalhar a metodologia de seleção e avaliação dos estudos analisados por essa pesquisa, e explorar algumas lacunas existentes na literatura atual. Fornecendo uma compreensão generalista das abordagens e experimentos realizados na área de integração de arquivos fontes sob um contexto abrangente. Para isso, vários estudos foram selecionados para serem discutidos após uma análise criteriosa e imparcial onde foram considerados aspectos como: importância dos critérios analisados, relevância dos assuntos abordados

e o tipo de análise comparativa dos resultados.

3.1.1 Estratégia de Pesquisa

Essa Seção apresenta a estratégia de pesquisa abordada por essa pesquisa, a qual possui dois objetivos principais: (1) Realizar um SMS considerando os principais aspectos abordados pelos estudos avaliados nas últimas duas décadas (2002 – 2023) sobre integração de arquivos fontes e por fim, (2) Apresentar algumas diretrizes de pesquisa para o desenvolvimento de novas pesquisas. Assim, nove questões de pesquisas (QP) foram definidas com a finalidade de investigar as diferentes facetas dos objetivos avaliados. A Tabela 1 detalha as QP, as motivações e as variáveis investigadas, além de fornecer uma visão geral de como serão abordadas as lacunas existentes e mencionadas anteriormente.

Questão de Pesquisa	**Motivação**	**Variável**
QP1: Quais os aspectos de merge de software avaliados?	Entender os diferentes aspectos de merge de software avaliados pelos estudos	Tipos de conflito
QP2: Quais os tipos de merge de artefatos avaliados?	Determinar quais os tipos de merge de artefatos avaliados pelos estudos	Tipos de merge
QP3: Quais os tipos de conflitos investigados?	Avaliar os principais tipos de conflitos investigados	Tipos de granularidade
QP4: Quais as principais abordagens aplicadas?	Determinar os tipos de abordagens mais investigadas e exploradas pelos estudos	abordagens aplicadas
QP5: Quais os métodos utilizados para avaliar as abordagens de integração?	Investigar os métodos utilizados na avaliação das abordagens de integração	Métodos de pesquisa
QP6: Qual o grau de automatização?	Investigar os principais graus de automatização (semiautomático e automático)	Grau de automatização
QP7: Quais as principais contribuições?	Apresentar as principais contribuições dos estudos avaliados	Principais Contribuições
QP8: Qual a duração dos experimentos até a extração de resultados?	Qual o tempo necessário para a execução dos experimentos	Duração dos experimentos
QP9: Qual o veículo e ano da publicação?	Identificar os principais veículos de publicação	Veículo de publicação

Tabela 1: Questões de Pesquisa Investigadas.

A literatura atual foi analisada sobre os seguintes aspectos: quais os tipos de integração que foram utilizados com maior frequência, quais os métodos de pesquisa aplicados na análise desses estudos, quais as questões de pesquisas abordadas, quem são os principais pesquisadores e as suas contribuições durante o período avaliado e por fim, a aplicação das abordagens avaliadas na indústria de software. Em (CARBONERA; FARIAS; BISCHOFF, 2020; BISCHOFF et al., 2019; GONÇALES et al., 2015b; KITCHENHAM et al., 2009) os autores detalham que as QP dos estudos de SMS devem ser genéricas para que as tendências, abordagens e os tópicos abor-

dados na literatura possam ser apresentados o mais detalhado possível, pois proporcionam uma visão geral da literatura atual. Portanto, foi apresentado os objetivos dessa pesquisa a partir do modelo GQM (CALDIERA; ROMBACH, 1994) da seguinte forma:

Analisar *a literatura*
para compreender *o seu estado atual*
no que diz respeito *aos métodos de pesquisa aplicados,*
aspectos técnicos, análise e avaliação dos experimentos
da perspectiva dos *pesquisadores e profissionais*
no contexto da integração *de códigos-fontes.*

As QP relacionadas na Tabela 1 estão abaixo detalhadas, com a justificativa da sua necessidade e relevância para essa pesquisa. As variáveis investigadas foram cuidadosamente elaboradas e revisadas para mitigar qualquer erro nos resultados. A fundamentação teórica e os termos técnicos apresentados na resolução de cada uma das QP foram obtidos a partir de uma análise criteriosa de todos os estudos primários avaliados por essa pesquisa e relacionados no Apêndice A.

QP1: Quais os aspectos de merge de software avaliados? Em [P04,P06],(MENS, 2002) os autores afirmam que existem dois tipos de aspectos: (1) Sintático: Esse tipo avalia a dependência entre os artefatos alterados, situação essa que será somente detectado quando existirem erros de compilação ou mensagens de aviso. Por fim, (2) Semântico: Avalia o tipo de conflito gerado durante a execução de um programa, como por exemplo um erro matemático resultante da realização de um processamento, esse tipo de conflito está diretamente condicionado ao valor da variável. Sendo na atualidade, um dos principais desafios da engenharia de software moderna, tanto para a indústria de software como para a comunidade acadêmica, que vêm investigando e tentando aprimorar às abordagens existente, mas sem conseguir apresentar melhorias significativas em comparação às abordagens existentes na atualidade.

QP2: Quais os tipos de merge de artefatos avaliados? Segundo [P03,P29,P38],(MENS, 2002), existem apenas dois tipos: (1) Two Way Merge: Historicamente foi a primeira abordagem de integração desenvolvida. Portanto, trata-se de uma abordagem de integração entre os artefatos analisados, realizando uma análise puramente textual dos arquivos de códigos-fontes, que são considerados como unidades indivisíveis. Consiste em integrar dois ou mais arquivos com a última versão avaliada, e desconsiderando a hierarquia dos objetos e realizada de forma automática. Por fim, (2) Three Way Merge: Essa abordagem trata-se da evolução direta da Two Way Merge, sendo a principal abordagem na atualidade. A diferença em relação à Two Way Merge está em considerar a árvore histórica dos objeto, analisando os antecessores dos artefatos e na ocorrência de conflitos por alterações realizadas em iguais trechos ou funcionalidades dos códigos-fontes terá a necessidade de interação humana para a resolução dos conflitos.

QP3: Quais os tipos de conflitos investigados? Segundo [P10,P19,P39,P40,P58,P61],(MENS, 2002) os autores afirmam quem existem quatro tipos de conflitos possíveis: (1) Linha de Có-

digo: Trata-se do principal tipo de análise na atualidade. As linhas de códigos-fonte são avaliadas com unidades indivisíveis (letra a letra), (2) Branch: abordagem muito pouco investigada na atualidade, os experimentos envolvendo esse tipo de abordagem realizam integração de branches, possuindo um escopo maior de abrangência para avaliação e análise dos resultado, (3) Arquivos: Avalia conflitos ocorridos entre arquivos físicos, que são considerados como unidades indivisíveis, como exemplo pode-se evidenciar a alteração da versão de um código-fonte, versão da IDE utilizada entre outras possibilidades. E por fim, (4) Bloco de Código: Experimentos que avaliaram os conflitos ocorridos entre blocos de códigos-fonte analisaram a totalidade do bloco e não com uma unidade única e indivisível (linha a linha e caractere a caractere), essa é outra muito pouco empregada também. A maioria dos estudos avaliados por essa pesquisa (Apêndice A) analisaram ao menos um dos tipos de conflitos acima listados.

QP4: Quais as principais abordagens aplicadas? Conforme apontado na QP2 existem duas abordagens, a Two Way Merge que trata-se de primeira abordagem de integração desenvolvida e que realiza a integração entre artefatos alterados. E a Three Way Merge, que é a evolução direta da primeira, e que considera a árvore histórica do objeto alterado. A finalidade dessa RQ está em investigar os benefícios resultantes da aplicação da técnica Three Way Merge, quais os motivos dessa abordagem ser a mais empregada atualmente, quais as investigações realizadas como parte de melhoria e da evolução dessa abordagem entre outras possibilidades. Outro aspectos importante a ser considerado, é que de acordo com os estudos primários listados no Apêndice A não foi possível identificar quais tipos de abordagens são aplicadas pelos algoritmos durante a integração dos arquivos de códigos-fontes.

QP5: Quais os métodos utilizados para avaliar as abordagens de integração? Foi detectado dois tipos de análises realizadas pelos estudos avaliados por essa pesquisa: (1) Análises Estatísticas: Refere-se aos estudos que avaliam estatisticamente os dados coletados, com a finalidade de obter indicadores de qualidade, produtividade e de melhoria de processo. Esse tipo de análise é realizado principalmente para validar a acurácia dos resultados obtidos em comparação aos resultados de outras abordagens ou experimentos, avaliando os indicadores obtidos. Portanto, trata-se de uma abordagem muito aplicada na avaliação de ferramentas ou de algoritmos de integração de arquivos fontes. Por fim, (2) Análises Empíricas: Estudos que avaliam os resultados obtidos através da execução de experimentos empíricos, considerando os aspectos comportamentais e processuais que influenciam nos indicadores obtidos, tais como: qualificação dos desenvolvedores, quantidade de participantes, número de projetos, quantidade de commits, quantidade de branches e arquivos fontes envolvidos entre muitos outros aspectos importantes. O resultado tenderá a ser a definição de um conjunto de boas práticas a ser adotadas pelas equipes de desenvolvimento, eliminando impurezas detectadas a partir da análise dos resultados. Portanto, a finalidade é a de estabelecer regras mitigando possíveis vieses para a geração de conflitos.

QP6: Qual o grau de automatização? Existem dois diferentes graus de automatização: (1) Automático: Refere-se ao grau onde não existe a necessidade de interação humana para a re-

solução dos conflitos, as alterações nos arquivos de códigos-fontes foram realizados em pontos distintos e não conflitantes, e a integração será realizada automaticamente. Assim, esse cenário é muito incomum de acontecer, principalmente quando uma equipe for composta de vários desenvolvedores em ambientes distribuídos ou local. Por fim, (2) Semi Automático: Ocorre quando existe a necessidade de iteração humana na resolução dos conflitos. As alterações foram realizadas nas mesmas funcionalidades ou em iguais trechos de código. O algoritmo de integração não possui a capacidade de solucionar conflitos, necessitando da interação dos desenvolvedores na definição da melhor resolução dos conflitos. Existindo uma grande probabilidade de ocorrer novos e diferentes tipos de conflitos após a integração, violando a integridade dos arquivos de códigos-fontes ou do projeto, existindo a necessidade de maior empenho para a solução dos conflitos gerados. Segundo [P20] a escolha dos desenvolvedores com maior nível de experiência e com maior conhecimento técnico quanto às funcionalidades envolvidas para a realização da integração dos códigos-fontes reduzirá a incidência desses tipos de conflitos após a integração dos arquivos de código-fonte alterados, reforçando que a adoção de boas práticas entre outras opções poderá aumentar os índices de qualidade e assertividade das equipes.

QP7: Quais as principais contribuições? Os estudos foram classificados em cinco tipos: (1) Processo: Estudos onde o foco principal na realização de avaliações empíricas. Os experimentos são executados por tempo finito até a obtenção de indicadores válidos ao contexto do experimento realizado. Caracterizam-se por avaliarem principalmente variáveis independentes e com resultados generalistas. Investigando a possível influência dessas variáveis no experimento, (2) Método: Avaliam especificamente as metodologias utilizadas, onde foi definido quais os parâmetros que apresentaram os melhores resultados, e disponibilizam informações para a realização de outros experimentos, (3) Métrica: Possuem a definição de alguma nova métrica ou avaliam alguma abordagem existente, com a finalidade de verificar a integridade dos resultados obtidos ou a formalização de novos indicadores. Na maioria das vezes, esses indicadores são resultantes de variações de outros indicadores já existentes, analisando os aspectos técnicos especificamente envolvidos, (4) Ferramenta: Apresentam novas abordagens ou novas funcionalidades. Detalhando novos conceitos e as vantagens da sua utilização em relação a outros métodos existentes. Trata-se de estudos desenvolvidos de forma experimental, e que contribuem na melhoria contínua desse importante segmento da engenharia de software moderna e por fim, (5) Modelos: Avaliam resultados de experimentos empíricos aplicando um modelo de análise, apresentam considerações amplas, podendo ser trabalhadas especificamente, fornecendo uma visão conceitual e tratando as questões avaliadas em nível mais amplo, com a finalidade de desenvolver novos conceitos ou métricas.

QP8: Qual a duração dos experimentos até a extração de resultados? Refere-se aos estudos que buscam a definição (o mais precisa possível) do intervalo de tempo de execução de um experimento até a obtenção de indicadores válidos de qualidade e assertividade. Logo, quanto maior a equipe e o período de execução do experimento, maior será a quantidade de variáveis dependentes e independentes envolvidas aumentando consideravelmente a complexi-

dade da integração dos arquivos fontes. A correta análise da dependência ou da relação direta e indireta entre essas variáveis poderá fornecer resultados importantes na composição de indicadores de qualidade, assertividade e produtividade, e ainda a quantidade de tempo aplicados para a resolução dos conflitos. A indústria de software vem buscando definir o melhor ponto de equilíbrio entre o período de desenvolvimento e a integração dos códigos-fontes. Para assim, reduzir o tempo das correções realizadas, aumentando a produtividade e assertividade dos resultados. Portanto, conclui-se que a execução desse tipo de experimento com um intervalo de tempo otimizado resultará para a indústria de software em excelentes indicadores assertividade e produtividade.

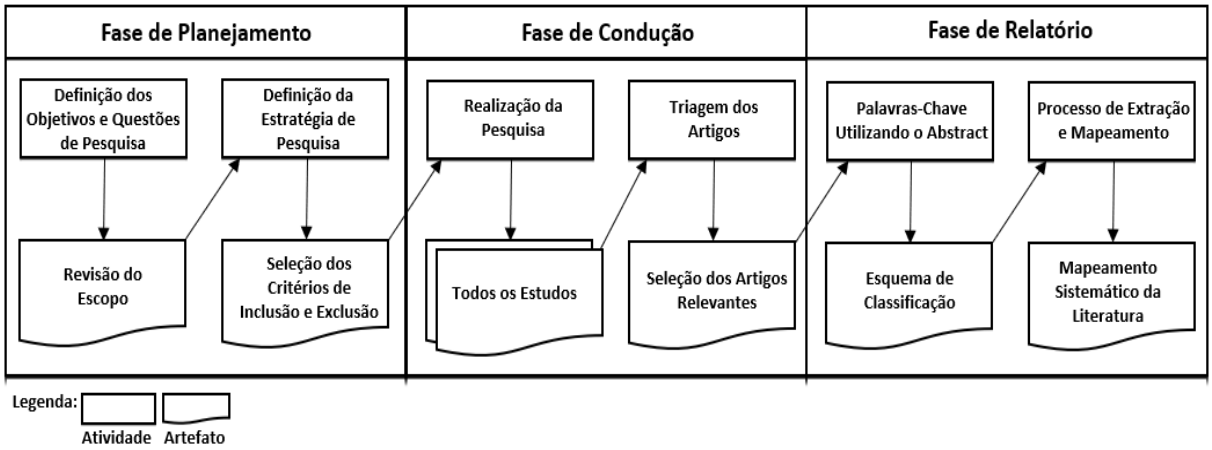
QP9: Qual o veículo e ano da publicação? As pesquisas realizadas geralmente são disponibilizadas em diferentes fontes de publicação, entre elas: sites específicos, conferências, periódicos ou capítulos de livros. Cada uma dessas fontes possui um nível de qualificação diferente, e conseguir compreender e analisar os aspectos dos estudos é uma preocupação fundamental para estabelecer um excelente nível de maturidade para a correta análise dos estudos avaliados. Assim, torna-se possível definir o ano aproximado do surgimento de uma determinada área de pesquisa, identificar qual ano em que as publicações atingiram os melhores e maiores índices de quantidade e qualidade ou ainda, identificar tendências futuras para o desenvolvimento de novas pesquisas. Essa questão, avaliou onde os estudos foram mais publicados e buscou compreender como essas publicações se desenvolveram ao longo dos anos e quais fatores às influenciaram.

Após a definição das QP, a próxima etapa foi definir a estratégia para recuperar os estudos potencialmente relevantes e disponíveis na literatura atual. Para isso, foram seguidas orientações bem definidas conforme definido em (CARBONERA; FARIAS; BISCHOFF, 2020; GONÇALES et al., 2015b; BISCHOFF et al., 2019; PETERSEN; VAKKALANKA; KUZNIARZ, 2015; KITCHENHAM et al., 2009; KITCHENHAM; CHARTERS, 2007; KEELE et al., 2007) quanto à construção das strings de buscas (SS) e do escopo abordado por essa pesquisa. Foi possível definir uma estratégia de busca iterativa imparcial para a seleção de estudos candidatos. e que apresentassem relação direta com às questões de pesquisas definidas em 1.3, indicando os SMS existentes, estudos empíricos e os surveys, bem como na avalição quantitativa e qualitativa das informações apresentadas. Por fim, foi definida e refinada a estratégia de busca para mitigar as ameaças à construção da estratégia de busca dessa pesquisa.

3.1.1.1 Estratégia de Pesquisa

Após a definição das questões de pesquisa, e etapa seguinte foi definida uma estratégia de busca. Seguindo as diretrizes empíricas já conhecidas e discutidas em (CARBONERA; FARIAS; BISCHOFF, 2020; GONÇALES et al., 2015b; BISCHOFF et al., 2019; PETERSEN; VAKKALANKA; KUZNIARZ, 2015). Após a formalização de uma estratégia de busca imparcial e iterativa, a próxima etapa foi definição das string de buscas, as quais foram fundamentais na filtragem e seleção de uma lista de estudos representativos da literatura atual. A Tabela 2

apresenta as palavras chaves e os sinônimos utilizadas nas string de buscas avaliadas. Esses termos e sinônimos foram definidos após uma cuidadosa revisão das palavras-chaves mais utilizadas nos estudos selecionados, que contribuíram de forma significativa para a formalização dos termos. Por fim, a Figura 9 detalha as fases conduzidas.



Fonte: Criado pelo Autor.
Figura 9: Processo de Mapeamento Sistemático da Literatura Adaptado de (CARBONERA; FARIAS; BISCHOFF, 2020; PETERSEN et al., 2008).

Principal	**Alternativos**
Merge	Integration, Fusion, Consolidation
Development	Maintenance, Evolution, Productivity
Software	Application, Product, Project
Effort	Cost, Pricing, Time

Tabela 2: Definição dos Termos Principais e Alternativos da Consulta.

Passos para definição da string de busca. Abaixo estão detalhados os aspectos que auxiliaram na definição da string de busca. Esse modelo foi amplamente validado e aplicado em (CARBONERA; FARIAS; BISCHOFF, 2020; BISCHOFF et al., 2019; GONÇALES et al., 2015b):

1. Definição dos termos principais;
2. Definição dos termos alternativos ou sinônimos relevantes aos termos principais;
3. Avaliação se os termos principais estão presentes nos estudos;
4. Associar os termos alternativos ou sinônimos com os termos principais com o operador "OR";
5. Associar os termos principais com o operador "AND".

As combinações que produziram os resultados mais significativos são apresentadas a seguir:

(Merge OR Integration OR Fusion) AND
(Development OR Maintenance OR Integration) AND
(Software OR Application OR Product OR Project) AND
(Effort OR Cost OR Pricing OR Evolution) AND
(Consolidation OR Productivity OR Time)

A próxima etapa foi definir de onde a literatura atual seria recuperada ou seja, quais as fontes de informação que seriam utilizadas. A Tabela 3 detalha os bancos de dados eletrônicos utilizados para recuperar os estudos, que foram selecionadas devido a sua importância para a comunidade acadêmica e pela considerável quantidade de estudos hospedadas nelas. Além disso, essas fontes de dados sendo utilizadas em [P07],(CARBONERA; FARIAS; BISCHOFF, 2020; BISCHOFF et al., 2019; GONÇALES et al., 2015b; PETERSEN; VAKKALANKA; KUZNIARZ, 2015).

Fonte	**Site**
ACM DL	`https://dl.acm.org/`
Elsevier	`https://www.elsevier.com`
IEEE Xplore	`https://ieeexplore.ieee.org/Xplore/home.jsp`
Scopus	`https://www.scopus.com/`
Springer	`https://www.springer.com/`

Tabela 3: Base de Dados.

3.1.2 Critérios de Inclusão e Exclusão

Essa Seção definiu os critérios de exclusão (CE) e de inclusão (CI) utilizados na filtragem dos estudos avaliados por essa pesquisa. A escolha desses critérios seguiu padrões e aspectos técnicos muito bem definidos em (CARBONERA; FARIAS; BISCHOFF, 2020; RODRÍGUEZ et al., 2017; FERNÁNDEZ-SÁEZ; GENERO; CHAUDRON, 2013; GONÇALES et al., 2015b; BISCHOFF et al., 2019). Porém, adaptados ao contexto dessa pesquisa, considerando principalmente a qualidade e a relevância dos trabalhos publicados.

- **CE1:** O título, resumo ou outro conteúdo não estava intimamente relacionado com o tema da pesquisa;
- **CE2:** Não publicado em inglês, patente, ou considerado como estágio inicial;
- **CE3:** Nenhum aspecto das questões de pesquisas foi encontrado no resumo;
- **CE4:** Estudos duplicados ou que não abordassem problemas relacionados a integração de trechos de códigos-fonte.

Escolha dos Critérios de Exclusão (CE). A definição desses critérios ocorreu por vários motivos. Primeiro, seria irrelevante avaliar estudos que não apresentassem nenhuma relação semântica com o assunto de integração de arquivos fontes devido a string de busca ter relação com o título ou resumo (CE1). Segundo, os estudos em estágio inicial não apresentam informações conclusivas ou opiniões definidas, e podem distorcer a formação de uma visão analítica da literatura, tendências ou lacunas encontradas (CE2). Terceiro, trabalhos que não apresentassem minimamente nenhuma relação com às questões analisadas por essa pesquisa, definitivamente não contribuiriam na resolução das questões de pesquisas (CE3). Por fim, é totalmente desnecessário avaliar estudos em duplicidade (CE4).

Aplicação dos Critérios de Exclusão Os critérios foram aplicados individualmente, tendo a necessidade da interpretação e compreensão do texto como um todo, sendo uma atividade executada de forma iterativa e incremental. Assim , com a finalidade de mitigar possíveis erros, foi executado um ciclo de três avaliações, para que não fossem excluídos estudos importantes ou de maneira equivocada da listagem final.

Escolha dos Critérios de Inclusão (CI). A definição desses critérios envolveu vários e importantes aspectos, e que estão abaixo detalhados:

- **CI1:** Artigos científicos publicados com o objetivo de apresentar novas abordagens de integração de arquivos de códigos-fontes;
- **CI2:** Publicados ou divulgados em inglês;
- **CI3:** Trabalhos encontrados em revistas científicas, conferências, grupos de pesquisa, página da web ou instituições de ensino; e
- **CI4:** Publicados de janeiro de 2002 até Março de 2023. O período de duas décadas para avaliação considerou os estudos publicados a partir do ano da publicação do estudo (MENS, 2002). Sendo uma publicação de referência na área de integração de códigos-fontes desde a sua publicação até a atualidade, possuindo atualmente 764 citações bibliográficas (acessado em janeiro de 2024).

Extração de Dados

Essa Seção detalha como os dados dos estudos foram extraídos, apresentados na Seção 3.2. O procedimento de extração consistiu em ler e interpretar cuidadosamente cada um dos estudos selecionados e catalogar os dados extraídos em uma planilha de controle formatada especificamente para essa atividade. Facilitando o trabalho de síntese das informações, pois permitiu obter detalhadamente os dados necessários, gerando indicadores qualitativos e quantitativos, apresentando evidências para as QP que estão detalhadas na Tabela 1). Portanto, foram gerados valores, indicadores, dados nominais ou ordinais que auxiliaram nas tentativas de criação de visões da literatura atual. No QP5, cada artigo foi analisado e classificado quanto a estratégia de avaliação aplicada, seja em análise estatística ou empírica. Portanto, foi possível elencar seis

diferentes tipos de estudos desenvolvidos na linha de merge de software, conforme demonstrado na Tabela 4 e onde cada um deles foi devidamente conceituado.

Definição	**Descrição**
Estudo de Experiência	Detalha a opinião pessoal dos autores e como foi realizada a experiencia prática
Estudo Filosófico	Nova visualização sobre uma técnica analisada, na forma de novo conceito ou taxonomia
Estudo de Opinião	Afirmação dos autores sobre uma técnica, detalha a execução e não relaciona outros experimentos ou metodologias
Pesquisa de Avaliação	Experimentos caracterizados e executados com diferentes parâmetros a cada iteração, compara os resultados com outros existentes
Pesquisa de Validação	Desenvolvimento de novas abordagens, que geralmente são experimentos técnicos, não existem dados publicados ou catalogados
Proposta de Solução	Nova solução para um problema existente. Os resultados são apresentados evidenciando os resultados melhores

Tabela 4: Tipos de Estudos.

Em QP1, os estudos que executaram experimentos técnicos e não empíricos foram cuidadosamente avaliados com a finalidade de identificar os tipos de análises realizadas, e que se pode identificar como: (1) Análise Sintática, que detecta erros de formatação da linguagem utilizada e, (2) Análise Semântica, que avalia a possibilidade de ocorrer problemas ou falhas durante a execução de um determinado processamento, sendo essa área um enorme desafio de pesquisa a ser investigada. Em QP3 aborda aspectos relacionados das abordagens de pesquisa avaliadas pelos estudos selecionados, incluindo resolução de conflitos, estimativas de tempo, avaliações abordagens por especialistas na área, estudos empíricos e estimativa de tempo para a resolução de conflitos.

O termo estimativa de tempo refere-se a utilização de abordagens baseadas na experiência dos desenvolvedores durante a execução das tarefas de integração, as informações obtidas podem ser qualificadas como irrelevantes devido a pouca ocorrência de dados de avaliação, sendo outra lacuna de pesquisa a ser investigada pela comunidade acadêmica e pela indústria de software. A Tabela 5 classifica os estudos de acordo com o seu tipo de contribuição, e que estão assim segmentados: Ferramentas, Métricas, Métodos, Modelos e Processos. E para cada tipo de classificação foi evidenciada uma quantidade considerável de publicações, o que demonstra individualmente a importância de cada um dos segmentos. Por fim, foi detectada uma tendência para o desenvolvimento de pesquisas que executam e avaliam processos por tempo finito e com parâmetros claramente definidos.

Quanto ao QP8, foi detectado a inexistência do detalhamento e de especificação dos aspectos avaliativos referentes à quantificação de tempo de execução (data de início e fim) ou a quantidade de horas duração dos experimentos. Os estudos avaliados executaram os experimentos durante o período que os pesquisadores determinaram como tempo ideal até a obtenção de resultados relevantes, não existindo um padrão de tempo definido e sequer uma investiga-

Tipo	Total	Percentual	Estudos
Ferramentas	9	12,85%	[P09],[P11],[P40],[P41],[P42],[P43],[P48],[P65],[P69]
Métodos	14	20%	[P01],[P02],[P05],[P07],[P08],[P21],[P22],[P57],[P60], [P64],[P66],[P67],[P68],[P70]
Métricas	12	17,14%	[P06],[P10],[P12],[P13],[P14],[P16],[P17],[P23],[P24], [P25],[P44],[P63]
Modelos	6	8,57%	[P26],[P27],[P28],[P45],[P49],[P52]
Processos	29	41,43%	[P03],[P04],[P15],[P18],[P19],[P20],[P29],[P30],[P31], [P32],[P33],[P34],[P35],[P36],[P37],[P38],[P39],[P46], [P47],[P50],[P51],[P53],[P54],[P55],[P56],[P58],[P59], [P61],[P62]

Tabela 5: Quantidade por Tipo de Estudo.

ção da influência exercida pela definição do tempo de duração de um experimento. Assim, foi detectada a inexistência de pesquisas que executem experimentos com iguais parâmetros de tempo e diferentes parâmetros relacionados a integração de códigos-fontes, como por exemplo: quantidade de códigos-fontes envolvidos, quantidade de commits realizados, quantidade de desenvolvedores em diferentes níveis de conhecimento técnico. Sendo esse um aspecto muito importante a ser considerado, pois fornecerá indicadores de qualidade e de produtividade para as equipes.

Avaliação de Qualidade. A avaliação de qualidade realizada nos estudos selecionado considerou os aspectos investigados durante essa etapa de seleção dos estudos, e que estão assim definidos: (1) Estudo de Risco de Avaliação de Viés – Avalia os tipos de experimentos executados pelos estudos, (2) Métodos de Síntese – Considera os aspectos e as características sensíveis ao contexto dessa pesquisa e por fim, (3) Avaliação de Certeza – Identifica a validade e a importância dos estudos dentro do contexto pesquisado, e que estão melhores detalhados a seguir:

Estudo de Risco de Avaliação de Viés. Foram utilizadas as informações referentes a classificação do tipo dos estudos coletados durantes as etapas de seleção, como avaliação de risco de viés pode-se considerar os estudos de casos, experimentos controlados (empíricos ou estatísticos), Review, Surveys e SMS. A aplicação da análise dos aspectos de qualidade e relevância dos estudos foi muito importante nesta etapa, pois possibilitou eliminar distorções nos resultados.

Métodos de Síntese. A inclusão da análise de sensibilidade exploratória foi realizada com a finalidade de validar a robustez das abordagens utilizadas pelos estudos filtrados. Foram definidos critérios avaliativos quanto aos estudos que abordassem experimentos empíricos ou técnicos. Foram considerados aspectos como o tempo de execução do experimento, quantidade de desenvolvedores e as abordagens de validações aplicadas.

3.2 Filtragem dos Estudos

Essa Seção detalha o processo de filtragem dos estudos candidatos, sendo composta de oito etapas, nas quais foram aplicados os CE descritos na Seção 3.1.2. O snowballing foi aplicado para se conseguir encontrar outros estudos relevantes a essa pesquisa, demandando muito esforço de leitura e avaliação durante essa etapa. Foram adicionados mais oito estudos seguindo as diretrizes muito bem definidas em (WOHLIN, 2014). A Figura 10 detalha a execução do processo de filtragem, e cada uma das etapas é descrita a seguir:

- **Passo 1: Pesquisa Inicial.** Apresentou os resultados iniciais da pesquisa após o envio da string de pesquisa para os bancos de dados eletrônicos (Tabela 3), o total de 17.800 estudos candidatos foram recuperados.

- **Passo 2: Remoção de Impurezas (CE1 & CE2).** Foi aplicado os dois primeiros critérios de exclusão (CE1 e CE2), com a finalidade de remover as impurezas encontradas. Após a aplicação desses critérios, foram eliminados 15.662 estudos representando 87,99%, onde 2.138 estudos foram selecionados para a próxima etapa. Assim, muitos desses trabalhos caracterizam-se por serem chamadas para trabalhos de conferências, edições especiais de periódicos, relatórios de pesquisa e materiais que foram publicados e não revisados por pares. Assim, foram eliminados os estudos que não estavam escritos em língua inglesa também.

- **Passo 3: Filtrar por Similaridade (CE3).** Essa etapa removeu todos os estudos que não tinham similaridade com a string de busca dessa pesquisa ou a total ausência de interação semântica do seu título, resumo ou o conteúdo em relação com o tema investigado por essa pesquisa. Ao final, 1.830 estudos foram eliminados representando 83,17%, restando apenas 308 estudos que continuaram no processo de filtragem.

- **Etapa 4: Filtrar por Resumo (CE3).** Essa etapa examinou os 308 estudos a partir do resumo. Foram excluídos onze estudos devido à sua irrelevância ao contexto dessa pesquisa, que representou 3,57%. Posteriormente, foram removidos os estudos que o conteúdo não encontrava-se intimamente relacionado às questões-chaves de pesquisa abordadas por essa pesquisa, restando apenas 297 estudos possivelmente candidatos.

- **Passo 5: Filtrar por Irrelevância (CE3).** Nessa etapa complementar foram removidos outros 142 (47,81%) estudos por apresentarem métricas avaliativas muito equivalentes a de outros estudos já avaliados – em muitos casos, apenas a ordem de execução dos fatores eram diferentes, ou ainda a quantidade de vezes que o experimento era iterado, restando ainda 155 estudos candidatos.

- **Passo 6: Filtrar Duplicados (CE4).** Normalmente, um determinado artigo poderá ser encontrado em outras bibliotecas digitais. Assim, foi aplicado o CE4 para remover todos os estudos duplicados. Assim, 22 estudos foram eliminados representando 14,19%,

garantindo assim a singularidade de cada estudo, resultando em apenas 133 estudos candidatos.

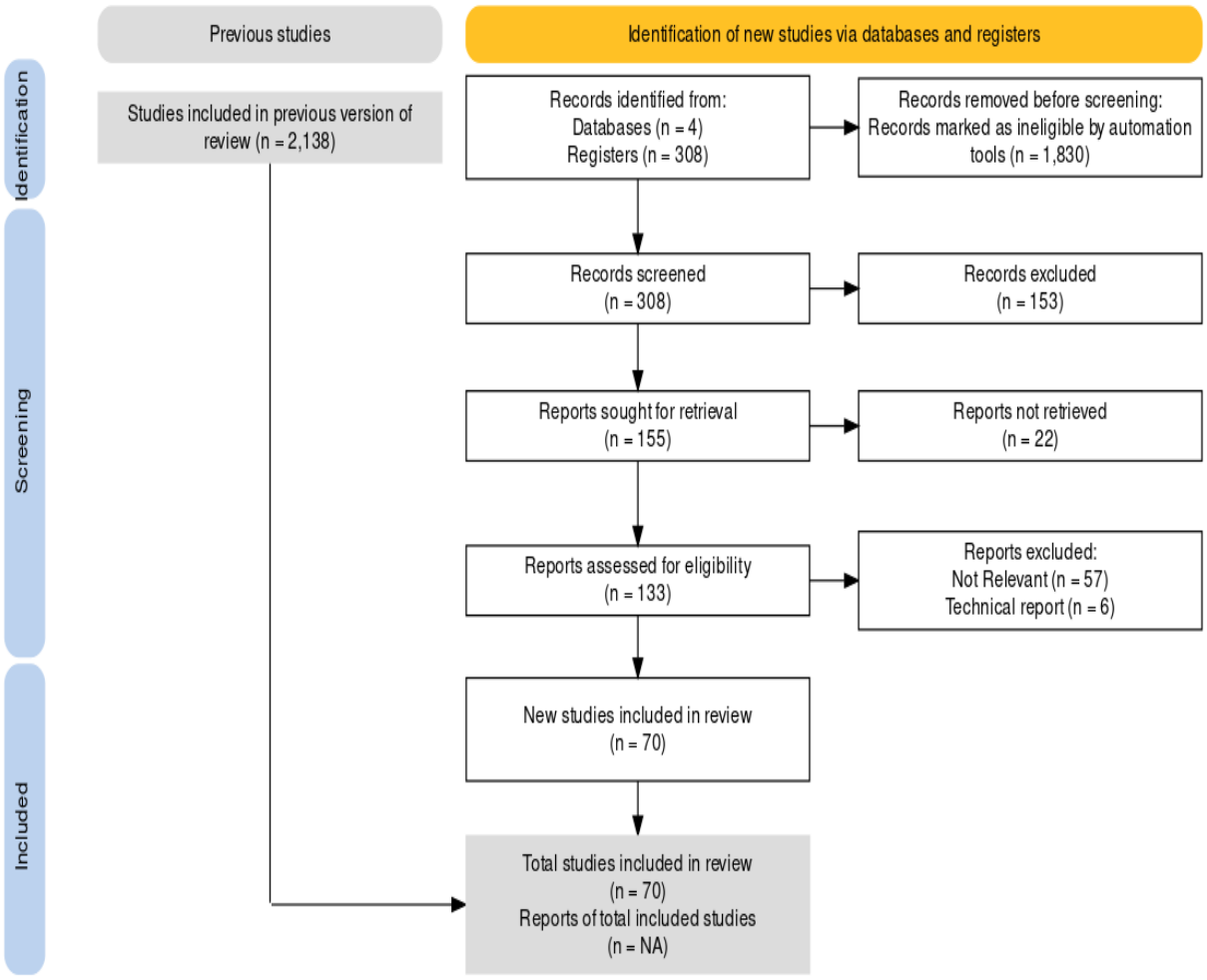
- **Etapa 7: Filtrar por Texto Completo (CE4).** Após a leitura do texto completo dos 133 estudos restantes, 57 estudos foram eliminados pela aplicação do CE4, excluindo estudos cujos conteúdos estavam longe das questões esperadas referentes estimativa de tempo, representando 42,86%. As seguintes regras foram aplicadas para dar suporte ao nosso processo de filtragem:

  - *Regra 1:* Estudos cujo conteúdo estava relacionado ao tema de estimativa de tempo, mas que não foi aplicado à área de engenharia de software. Embora tenham sido identificados pela string de busca, seu conteúdo não está dentro do escopo dessa pesquisa conforme detalhado na Seção 2.1.1.
  - *Regra 2:* Os estudos de revisão de literatura foram removidos e classificados como trabalhos relacionados.
  - *Regra 3:* Os estudos considerados pequenos (até 3 páginas) foram eliminados.
  - *Regra 4:* Estudos que estavam alinhados diretamente com os objetivos dessa pesquisa e detalhados na Seção 1.4. Foram filtrados os estudos que o conteúdo não estava intimamente relacionado aos objetivos dessa pesquisa.

- **Passo 8: Seleção dos Trabalhos Representativos.** Posteriormente mais seis estudos (7,89%) foram eliminados por estarem categorizados como relatórios técnicos; Mesmo que no título não apresentasse essa definição, chegando ao total de 70 estudos, agora denominados como estudos primários e apresentados no Apêndice A. As várias etapas de filtragem estão representadas na Figura 10.

Portanto, com a finalidade de mitigar quaisquer questões relacionadas à confiabilidade e qualidade dos resultados presentados pelo processo de filtragem. Foram realizados três ciclos de revisão para analisar detalhadamente os critérios ou as questões de qualidade estabelecidas, evitando assim falsos positivos que surgissem pela aplicação inadequada dos CI e CE. As etapas de filtragem e seleção dos estudos avaliados, desde a definição até a execução, duraram aproximadamente quatro meses para que os estudos fossem corretamente filtrados e classificados. Por fim, mais dois meses para se conseguir extrair dados relevantes ao contexto dessa pesquisa, incluindo as várias etapas de avaliação para cada um dos estudos candidatos filtrados.

Avaliação de Certeza. Como avaliação de certeza dessa pesquisa, foram considerados os seguintes aspectos: validade descritiva, teórica e interpretativa, além da capacidade de generalização das abordagens aplicadas durante os experimentos. Essa etapa foi muito importante para eliminar os estudos que se encontravam em nível inferior ao padrão de qualidade definido. Eliminando também os estudos com padrões de configuração similares de ambiente, parâmetros ou que apresentassem resultados de proporcionalidades semelhantes à de outros estudos.



Fonte: Criado pelo Autor.
Figura 10: Metodologia Prisma (PAGE et al., 2021).

3.3 Resultados

Essa Seção apresenta os resultados obtidos a partir da análise das QPs conforme detalhadas na Seção 1.4. As QPs foram respondidas individualmente e várias conclusões foram apresentadas e fundamentadas através dos resultados obtidos, evidenciando a importância das QPs para essa pesquisa.

QP1: Aspectos de Integração. A Seção 2.1.2 detalha os principais aspectos técnicos relacionados a integração de arquivos fontes. A Tabela 6 classifica os estudos de acordo com o tipo de análise realizada.

Tipo	Total	Percentual	Estudos
Sintática	50	71,42%	[P03],[P07],[P08],[P10],[P11],[P13],[P14],[P15],[P16], [P17],[P19],[P20],[P22],[P23],[P26],[P27],[P28],[P29], [P30],[P31],[P32],[P33],[P34],[P36],[P38],[P39],[P40], [P41],[P43],[P44],[P45],[P46],[P47],[P48],[P49],[P51], [P52],[P55],[P56],[P57],[P58],[P59],[P60],[P61],[P62], [P63],[P64],[P66],[P67],[P69]
Indefinido	14	20%	[P01],[P02],[P09],[P12],[P18],[P21],[P24],[P25],[P35], [P37],[P50],[P53],[P65],[P70]
Semântica	3	4,29%	[P06],[P42],[P54]
Sintática / Semântica	3	4,29%	[P04],[P05],[P68]

Tabela 6: Tipos de Conflitos.

Portanto, foi detectado que (1) A maioria dos estudos avaliaram aspectos sintáticos; (2) Em apenas 4,29% (3/70) dos estudos foram avaliados aspectos semânticos e sintático/semântico; e por fim (3) Em 20% (14/70) não foram realizadas nenhuma avaliação envolvendo aspectos sintáticos ou semânticos, mas experimentos empíricos sem considerar os dois aspectos. O desenvolvimento de abordagens mais precisas de avaliação da dependência entre os artefatos contribuirá de forma expressiva na redução de resultados indesejados, como por exemplo erros de cálculos matemáticos entre outros tipos possíveis. Esse tipo de problema somente será detectado durante a execução do programa. Assim, a validação contribuirá significativamente na redução de custos e dos recursos humanos empregados na correção desse tipo de problema. Assim, devido à complexidade da validação de tipo integração. Portanto, a partir da análise do artefato integrado e de suas dependências, o compilador da linguagem, quando detectar uma possibilidade de conflito, poderá emitir uma mensagem de atenção levando em consideração o histórico.

QP2: Tipos de Merge. A Seção 2.1.1 detalha os tipos de merges utilizados pelos estudos analisados, foi constatado que a totalidade dos estudos primários (Apêndice A) utilizou a abordagem Three Way Merge por se tratar da evolução direta da abordagem Two Way Merge. E comprovou ser tecnicamente mais confiável e com melhores resultados. Apenas em (MENS,

2002) foi definido e exemplificado à abordagem Two Way Merge. Porém, não há nenhum experimento em que foi aplicada essa abordagem, devido a expressiva utilização da Three Way Merge. Portanto, conclui-se que a sua constante melhoria e evolução, contribuirá expressivamente na melhoria de indicadores de desempenho e assertividade. Logo, não foi evidenciada nenhuma análise de desempenho e o detalhamento dos algoritmos utilizados por estas abordagens, seja pelo tipo de integração realizada, quantidade de ramificações ou branches. Tendo como característica analisar a possibilidade de escolha do algoritmo mais capacitado para a resolução dos conflitos. Por fim, a inexistência desse tipo de análise e de fundamentação histórica deixa mais uma questão em aberto para o desenvolvimento de novos estudos.

QP3: Tipos de Conflitos. A Tabela 7 quantifica e classifica os tipos de conflitos investigados por essa pesquisa, que estão assim detalhados: Linha de Código, Branch, Arquivos e Bloco de Código. Em 22,86% (16/70) dos estudos analisados não foi possível detectado nenhum tipo de análise de conflito, priorizando a avaliação dos resultados obtidos a partir da execução de experimentos empíricos em ambientes configurados.

Tipo	**Total**	**Percentual**	**Estudos**
Arquivos	2	2,86%	[P19],[P58]
Bloco de Código	1	1,42%	[P10]
Branch	3	4,28%	[P25],[P40],[P61]
Linhas de Código	48	68,58%	[P01],[P04],[P05],[P07],[P08],[P09],[P11],[P12],[P13], [P14],[P16],[P21],[P22],[P26],[P27],[P29],[P30],[P31], [P33],[P34],[P35],[P36],[P37],[P38],[P39],[P41],[P42], [P43],[P45],[P46],[P47],[P48],[P49],[P50],[P51],[P52], [P53],[P54],[P55],[P59],[P60],[P63],[P64],[P65],[P66], [P67],[P68],[P69]
Indefinido	16	22,86%	[P02],[P03],[P06],[P15],[P17],[P18],[P20],[P23],[P24], [P28],[P32],[P44],[P56],[P57],[P62],[P70]

Tabela 7: Tipos de Análise de Conflitos.

A maioria dos estudos avaliou a abordagem Linha de Código, onde as linhas de códigos-fonte são avaliadas individualmente e na forma puramente textual, apresentando os melhores resultados e demonstrou-se ser muito confiável por considerar cada caractere como uma unidade indivisível. Assim, devido a sua abordagem de avaliação confere a ela um excelente desempenho técnico e exatidão de resultados, em 4,28% (3/70) estudos analisaram conflitos relacionados a branches, 2,86% (2/70) conflitos relacionados a arquivos, 1,42% (1/70) conflitos relacionados a blocos de códigos.

QP4: Abordagens Aplicadas. As Seções 2.1.1.1 e 2.1.1.2 apresentam o detalhamento das duas principais abordagens de integração existentes nos estudos avaliados: Two Way e Three Way Merge. Exceto em (MENS, 2002), todos os demais estudos primários avaliados (Apêndice A) utilizaram exclusivamente a abordagem Three Way Merge, onde comprovadamente o seu desempenho técnico é muito superior e mais eficiente a abordagem Two Way Merge devido

a inclusão da versão original (VO') nas ramificações criadas (V1´e V2'). A análise da árvore sintática dos objetos contribui de forma significativa ao seu desempenho. As comparações ocorrem a partir das ramificações para com a sua ancestralidade, e isto facilita a identificação das alterações existentes, fornecendo maior suporte à resolução de conflitos. A abordagem Three Way foi abordada na totalidade dos estudos primários avaliados.

QP5: Métodos de Pesquisa. Em [P07] os autores apresentam informações estruturais dos artefatos através da gramática anotada. Assim, uma variedade de idiomas poderá ser suportada pelas abordagens de integração, auxiliando na resolução de conflitos durante a integração dos artefatos. Em [P25], os autores realizaram a mineração de repositórios de software para extrair e analisar os dados originais de vários projetos com a finalidade de compreender o histórico de desenvolvimento de software. Os autores apresentaram boas práticas que contribuem para a evolução dessas abordagens em tarefas futuras. Por fim, foi realizada uma comparação objetiva entre os algoritmos de integração existentes, os resultados foram validados a partir da seleção de vários projetos de códigos-fonte aberto e em desenvolvimento.

Em [P50], os autores executaram um experimento empírico sobre a utilização de ramificações para 2.923 projetos. Investigaram como os desenvolvedores utilizaram as ramificações e quais os efeitos na produtividade desses projetos. Concluíram que: (1) Poucas ramificações foram utilizadas durante o desenvolvimento, (2) Os projetos de grande escala utilizaram mais ramificações do que projetos menores, (3) As ramificações foram utilizadas na implementação de novos recursos, (4) A grande maioria dos commits realizados foram integrados na ramificação principal e por fim, (5) A quantidade de ramificações utilizadas em um projeto tem efeito positivo na produtividade. Esse efeito é pequeno e não há diferenças estatísticas significativas entre projetos dos mais variados tipos. A Tabela 8 apresenta as estratégias encontradas nos estudos avaliados, e os estudos que realizaram análises estatísticas representaram a maioria.

Tipo	**Total**	**Percentual**	**Estudos**
Análise Estatística	40	57,14%	[P02],[P03],[P05],[P06],[P08],[P11],[P17],[P19],[P21], [P22],[P23],[P25],[P26],[P28],[P30],[P34],[P35],[P36], [P37],[P38],[P39],[P41],[P44],[P46],[P47],[P49],[P50], [P52],[P53],[P54],[P55],[P56],[P59],[P60],[P61],[P64], [P65],[P68],[P69],[P70]
Análise Empírica	30	42,86%	[P01],[P04],[P07],[P09],[P10],[P12],[P13],[P14],[P15], [P16],[P18],[P20],[P24],[P27],[P29],[P31],[P32],[P33], [P40],[P42],[P43],[P45],[P48],[P51],[P57],[P58],[P62], [P63],[P66],[P67]

Tabela 8: Tipos de Análise de Experimentos.

Estudos que avaliaram aspectos estatísticos representaram a maioria, embora sendo segmentos distintos. Portanto, as áreas de pesquisas investigadas são muito relevantes para a evolução contínua das abordagens de integração de arquivos fontes mesmo que sejam totalmente independentes e coexistindo paralelamente. Os resultados obtidos contribuem direta ou indiretamente

na evolução das pesquisas realizadas de maneira mais generalista.

QP6: Grau de Automatização. A Tabela 9 apresenta os dois tipos de grau de automatização encontrados: (1) Automático – Executado sem a necessidade da iteração humana, o algoritmo responsável pela integração dos artefatos define qual o melhor resultado e realiza a integração e por fim, (2) Semiautomático – Existe a necessidade de iteração humana, o desenvolvedor deverá optar pela melhor maneira de resolver o conflito. Evidenciou-se que a abordagem semiautomática representa a maioria dos estudos analisados.

Grau	**Total**	**Percentual**	**Estudos**
Semiautomático	57	81,43%	[P03],[P05],[P06],[P07],[P08],[P09],[P10],[P11],[P12], [P13],[P14],[P15],[P16],[P18],[P19],[P20],[P21],[P22], [P23],[P25],[P26],[P27],[P28],[P31],[P32],[P33],[P34], [P35],[P36],[P37],[P38],[P39],[P41],[P42],[P44],[P45], [P46],[P47],[P48],[P49],[P50],[P51],[P52],[P53],[P55], [P58],[P59],[P60],[P61],[P62],[P63],[P65],[P66],[P67], [P68],[P69],[P70]
Automático	13	16,31%	[P01],[P02],[P04],[P17],[P24],[P29],[P30],[P40],[P43], [P54],[P56],[P57],[P64]

Tabela 9: Grau de Automatização.

Segundo [P04,P06,P45(MENS, 2002)], a presença de conflitos é detectada pelos softwares utilizados para a integração de arquivos fontes e na maioria dos casos a resolução desses conflitos requerem a iteração humana. O grau de automatização referente ao processo de integração de arquivos fontes, é um assunto muito importante a ser investigado. Conforme as decisões tomadas durante as tarefas de integração poderão surgir problemas resultante das escolhas feitas, e que poderá resultar em problemas de análise da sintaxe ou da semântica, demandando muito tempo na sua resolução. Assim, devido à sua importância no contexto técnico, muitos estudos foram realizados principalmente nos últimos anos, buscando melhorar o desempenho da avaliação da granularidade como forma de melhoria constante. O desenvolvimento de novas técnicas de granularidade com melhores índices de desempenho de avaliação em relação às abordagens e comprovadamente eficazes, contribuirá significativamente para a melhoria dos resultados e maior rapidez no processo de integração de código-fonte.

QP7: Principais Contribuições. A Tabela 5 classifica os estudos em termos quanto à sua contribuição, incluindo: Ferramentas, Métodos, Métricas, Modelos e Processos. Assim, foi observado uma forte tendência para estudos que realizem a execução e avaliação de processos por tempo finito e com parâmetros de execução claramente definidos, representando a maioria dos estudos avaliados. Esse tipo de estudo caracteriza-se por apresentarem um ambiente definido, com regras para a execução e de configuração, além de uma determinada quantidade de desenvolvedores envolvidos. O período de execução, e as etapas de avaliação dos resultados estão diretamente relacionadas aos parâmetros de configuração dos experimentos, resultando em indicadores mais precisos e detalhados.

Assim, outra situação encontrada é que muitos processos modificaram parâmetros de execução e de avaliação ao longo do experimento. A finalidade está em detectar e estabelecer indicadores de mudança de estado, para avaliar os resultados em diferentes contextos, como por exemplo: menor espaço de tempo para término das atividades relacionadas ao integrados de artefatos ou o tamanho da alteração do artefato. Por fim, os experimentos comprovaram que os processos realizados com menores períodos de execução e realizam integrações particionadas ou em blocos menores de códigos-fontes, tendem a apresentar melhores indicadores de desempenho e menores índices de retrabalhos ou erro. Assim, devido a capacidade de entendimento técnico do código-fonte pelos desenvolvedores.

QP8: Duração dos Experimentos. Após a avaliação de todos os estudos primários listados no Apêndice A, não foi possível quantificar ou determinar qual a média de tempo utilizada na avaliação dos experimentos (a partir dos estudos primários selecionados). Em [P25,P13] detectou-se a existência de indicadores quanto à execução dos experimentos com definição de tempo de execução. Contudo, inexistem informações relacionadas ao estabelecimento de um período de execução do experimento. Os estudos primários avaliados eram direcionados para a mensuração ou na geração de indicadores a partir da quantidade de commits, linhas alteradas ou branchs modificadas. A configuração dos parâmetros dos experimentos determinava a quantidade de projetos ou desenvolvedores avaliados, especialmente quanto ao cargo ocupado ou na quantidade de anos de experiencia profissional.

Por fim, a utilização de indicadores referente ao tempo de execução de um experimento, com a definição de uma data de inicial e final, poder contribuir na formação de métricas qualitativas e quantitativas que possam contribuir na evolução do processo. Pois, quanto maior o tempo aplicado e a complexidade das variáveis envolvidas, como por exemplo, quantidade de desenvolvedores e projetos avaliados, ou quanto maior a quantidade de commits realizados, maior será a possibilidade de geração de conflitos durante a integração dos artefatos.

QP9: Veículo de Publicação. Evidenciou-se que entre os anos de 2017 e 2021 foram publicados a maioria dos estudos primários avaliados e conforme detalhados na Tabela 10. Assim, nota-se o grande interesse da comunidade acadêmica e da indústria de software na melhoria contínua dos processos relacionados ao integração de arquivos fontes.

A Tabela 11 detalha as quantidades e os percentuais dos estudos publicados por fonte de dados, conferência ou evento. Os sites utilizados para buscar os estudos avaliados por essa pesquisa estão relacionados na Tabela 3. Sendo selecionados devido a elevada quantidade e importância dos estudos armazenados. Segundo (RODRÍGUEZ et al., 2017; FERNÁNDEZ-SÁEZ; GENERO; CHAUDRON, 2013; CARBONERA; FARIAS; BISCHOFF, 2020) os sites utilizados na busca dos estudos avaliados por essa pesquisa são amplamente utilizados pela comunidade acadêmicaa assegurando a qualidade das publicações neles realizadas.

A Figura 11 detalha as quantidades agrupados por ano, facetas de pesquisa e principais contribuições dos estudos primários avaliados por essa pesquisa. Entre os anos entre 2022 e 2023 não foram selecionadas quantidades representativas de estudo. Contudo, entre os anos entre

Ano	Total	Percentual
2007	3	4,29%
2008	1	1,43%
2010	2	2,86%
2011	3	4,29%
2012	4	5,71%
2013	2	2,86%
2014	2	2,86%
2015	5	7,14%
2016	4	5,71%
2017	6	8,57%
2018	7	10%
2019	9	12,86%
2020	8	11,43%
2021	9	12,86%
2022	4	5,71%
2023	1	1,43%

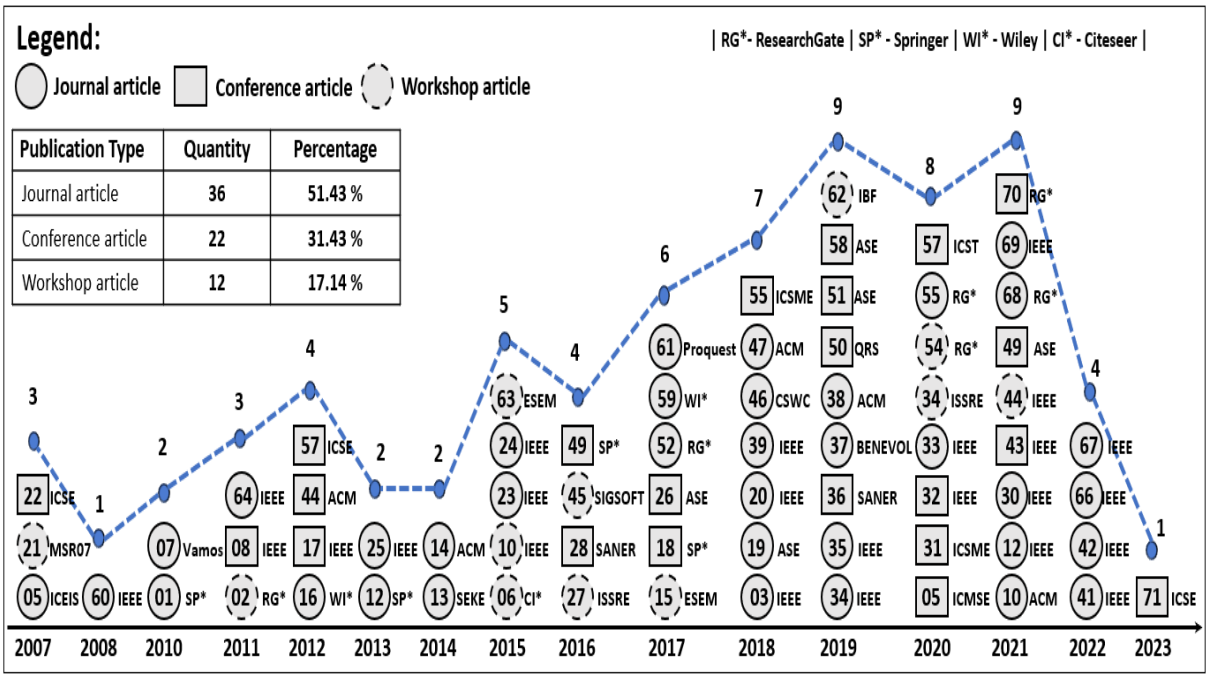
Tabela 10: Quantidade e Percentual de Estudos por Ano.

Site	Total	Percentual
IEEE Xplore	35	50%
ACM DL	10	14,29%
Springer	7	10%
ResearchGate	7	10%
Citeseer	2	2,86%
Elsevier	2	2,86%
Wiley Library	2	2,86%
Informatica.si	1	1,43%
PROQUEST	1	1,43%
Semantic Scholar	1	1,43%
Toronto.Edu	1	1,43%
World Scientific	1	1,43%

Tabela 11: Quantidade de Estudos Analisados por Site.

2017 e 2021, foi identificado como os anos que mais publicações foram realizadas. Os estudos foram filtrados considerando a importância e relevância em relação aos aspectos avaliados por essa pesquisa, embora considerando até o primeiro trimestre (Março) de 2023 o período limite para avaliação. Journal Article representa a maioria das publicações realizadas, seguido de Conference Article e por fim, WorkShop Article. Outro aspecto importante, é que todos os tipos de relatórios técnicos foram desconsiderados em todas as etapas do processo de filtragem. Essas publicações não representam a categorização de um experimento empírico e não detalham nenhuma informação relevante aos aspectos avaliativos dessa pesquisa, sendo especificamente condicionados aos parâmetros estabelecidos pelos autores. Detectou-se que entre os anos de 2022 até o mês de março de 2023 a quantidade de estudos avaliados reduziu de forma signifi-

cativa. Assim, conclui-se que o ano de 2022 pode ser definido como um ano atípico em relação as quantidades de publicações dos anos anteriores.



Fonte: Criado pelo Autor.
Figura 11: Distribuição das Publicações por Ano.

3.3.1 Estudos Comparativos

Essa Seção avaliou vários estudos representativos, todos esses estudos desenvolveram SMS, SOA, Reviews ou Surveys. Sendo foram selecionados devido à sua importância e pela relevância dos critérios analisados em relação aos demais estudos primários relacionados no Apêndice A. O foco principal foi a produção de conhecimento baseado em evidências e indicadores encontrados por esses estudos. Foram constatados poucos esforços da comunidade acadêmica para a produção de SMS ou revisões da literatura nas últimas duas décadas. Assim, a seleção dos estudos avaliados por essa pesquisa abrangeu o período analisado de duas décadas, considerando a relevância, contribuição, conteúdo e robustez dos aspectos avaliados.

Murta et al. (COSTA; MURTA, 2013). Realizou uma avaliação detalhada de vinte e nove SMS selecionados abordando versionamento e gerenciamento de arquivos de código-fonte entre o período 2002 até 2012. Os autores apontaram que: (1) A utilização de ferramentas de controle e versionamento no gerenciamento de projetos dá-se a partir das etapas seguintes ao planejamento e no início do desenvolvimento de software, (2) Desenvolvimento, testes, entrega e implantação são realizadas a partir das configurações estabelecidas para cada projeto e, (3) Em ambientes que os desenvolvedores trabalham de forma distribuída, o tempo aplicado é muito maior para se conseguir o controle efetivo do versionamento dos códigos-fontes – na maioria

dos casos as situações culturais do trabalho distribuído ainda precisa ser melhor discutida entre os desenvolvedores.

Segundo os autores, a principal questão de pesquisa está em como pode ser realizado o versionamento dos códigos-fontes em desenvolvimento distribuído de software. Logo, um dos aspectos complicadores encontrado nas tentativas de respostas foi o limitado número de estudos existentes. A realização de experimentos controlados na indústria de software, com projetos distribuídos ajudou a identificar como o controle e versionamento dos códigos-fontes foram realizados, e os resultados foram apresentadas conclusões claras auxiliando na melhoria do ciclo produtivo. Por fim, os autores conseguiram realizar um detalhamento robusto dos principais desafios existentes e das novas propostas de soluções no apoio ao controle de versão no cenário de desenvolvimento distribuído, e comprovaram através dos experimentos que a adoção de menores espaços de tempos entre os commits, e escolha dos desenvolvedores mais experientes na resolução de conflitos complexos ajudaram na melhoria dos índices de assertividade.

Mens (MENS, 2002). Realiza uma análise detalhada das principais abordagens de integração existentes até o ano de 2002. Detalha os aspectos técnicos dos algoritmos, onde inicialmente as abordagens de integração eram puramente baseadas na modelo textual, e com a evolução da abordagem Three Way Merge a análise sintática e semântica passou a ser considerada. Descreve os conceitos e a aplicação dos diferentes níveis de granularidade para a integração de arquivos fontes, e o nível do detalhamento para a detecção de conflitos. As ferramentas de integração podem apresentar um grau de automatização manual e demorado, passando para um processo semiautomatizado, que ainda necessita da interação humana, até a abordagem totalmente automatizada que não necessita da iteração humana, as vantagens e desvantagens que cada uma delas possui. Por fim, o autor apresenta vários aspectos e novos direcionamentos para a melhoria do processo de integração de arquivos fontes, foi evidenciado que na atualidade os desafios propostos pelo autor ainda continuam vigentes, como a evolução das abordagens de integração baseadas em operações que atuem em modificações mais complexas. Esse estudo, configura-se como referencial técnico e conceitual para novas pesquisas, pois conseguiu alinhar a definição de conceitos e exemplificação prática.

Apel et al. [P09]. Os autores investigaram quais são os fatores que tornam a resolução dos conflitos lenta. Analisaram 66 projetos contendo 81 mil cenários de integração, tendo 2 milhões de arquivos e mais de 10 milhões de blocos de códigos-fonte avaliados, aplicaram a correlação de classificação com a análise de componentes, modelos de regressão múltipla e a análise de efeito, para investigar quais variáveis independentes influenciam diretamente as variáveis dependentes. Portanto, concluíram que o aumento de tempo para resolução de conflitos é diretamente proporcional à quantidade dos artefatos alterados, e conseguiram estabeleceram uma taxonomia com quatro tipos de desafios para a resolução de conflitos.

Concluíram a pesquisa apresentando seis direções de novas pesquisas: (1) Como determinados cenários específicos impactam na contribuição para o projeto, (2) Entrevistar desenvolvedores para o desenvolvimento de novas taxonomias que auxiliam na resolução de conflitos, (3)

Investigação do impacto do tempo aplicado para a resolução de conflitos comparando cenários, onde os artefatos já foram alterados com cenários onde os artefatos ainda não foram alterados e, (4) Avaliação do tempo aplicado na resolução dos mesmos conflitos em diferentes tipos de cenários, e quais as contribuições possíveis para esses outros cenários; (5) Avaliação do tempo aplicado para a resolução de diferentes tipos de conflitos e por fim; (6) Avaliação do tempo para resolução de diferentes tipos de conflitos utilizando abordagens como Redes Neurais ou aprendizado de máquina (machine learning).

Apel et al. [P19]. Analisou o grau preditivo de vários indicadores obtidos a partir da integração de várias ramificações, tais como: tamanho ou grau de dispersão dos commits obtidos de software de versionamento ou do código-fonte diretamente. Foram avaliados 41 desenvolvedores, tendo como base de experimento 163 projetos de códigos-fonte aberto, com 21.488 cenários de integração e 49.449.773 linhas de código-fonte. Por fim, conseguiram identificar 07 potenciais indicadores para predição do número de conflitos de integração, sendo: (1) Variáveis dependentes, (2) Variáveis independentes, (3) Quantidade de commits, (4) Complexidade dos commits, (5) Quantidade de nós alterados, (6) Quantidade de blocos de códigos-fonte alterados, (7) Quantidade de mudanças dentro de declarações de classe e por fim, (8) Quantidade de alterações acima das declarações de classe. Concluíram que nenhum desses oito potenciais indicadores conseguiu prever a quantidade de conflitos ocorridos de maneira confiável e segura.

Zolkifli et al. [P20]. Investigaram a aplicação e a utilização continuada de softwares de versionamento como ferramentas de suporte ao desenvolvimento colaborativo, concluíram que a adoção de boas práticas contribuirá de maneira efetiva no controle do ciclo de versionamento, evitando a criação de conflitos e na diminuição do tempo para a resolução dos conflitos. Os principais apontamentos feitos pelos autores são: (1) Criação de uma branch por desenvolvedor, (2) Separação de recursos em ramificações nomeadas e, (3) Trabalhar com bases de dados estáveis e configuradas. Assim, com a avaliação desses cinco estudos, foram definidos também cinco critérios comparativos (CC) para identificar as similaridades e as diferenças entre vários estudos conforme listados na Seção 3.3.1. Essa comparação procura identificar oportunidades de novas pesquisas utilizando critérios objetivos e que estão assim definidos:

- **Estudos Experimentais (CC1):** Avaliação das abordagens experimentais e que identificam aspectos de causa e efeito.
- **Avalição de Software (CC2):** Análise da assertividade dos softwares de controle de versionamento existentes na atualidade.
- **Controle de Processo (CC3):** Investigação das abordagens e metodologias aplicadas no controle e versionamento de códigos-fontes em equipes distribuídas ou em paralelo.
- **Resolução de Conflitos (CC4):** Compreensão das abordagens aplicadas para a resolução de conflitos. Avaliação das boas práticas que podem ser adotadas mitigando a possíveis causas de geração de conflitos.

- **Avalição de Qualidade (CC5):** Análise e avaliação da qualidade após a realização da integração dos artefatos, principalmente a partir da ocorrência de conflitos em equipes distribuídas.

A Tabela 12 apresenta a abrangência dos CCs quantos aos estudos avaliados na Seção 3.3.1, em [P09],(MENS, 2002) é possível constatar que os CCs são atendidos na totalidade e em [P19] apenas CC4 e CC5 são atendidos parcialmente e os demais critérios são contemplados na totalidade, sendo muito divergentes dos demais estudos analisados. A principal contribuição foi o detalhamento dos desafios existentes e as novas soluções apresentadas pela *ProMerge* para dar suporte ao controle de versão no cenário de desenvolvimento distribuído, contribuindo para a formação de uma base histórica, com apontamentos para trabalhos futuros.

Trabalhos relacionados	**CC1**	**CC2**	**CC3**	**CC4**	**CC5**
Apel et al. [P09]	●	●	●	●	●
Apel et al. [P19]	●	●	●	◗	◗
Zolkifli et al. [P20]	●	❍	◗	❍	❍
Mens (MENS, 2002)	●	●	●	●	●
Murta (COSTA; MURTA, 2013)	●	◗	◗	❍	◗

Tabela 12: Análise dos Estudos Relacionados.

3.4 Discussão e Direções Futuras

Esta seção apresenta algumas discussões adicionais e destaca algumas direções futuras. Assim, foram classificados e organizados os estudos primários com base em três dimensões. A figura 12 mostra um gráfico de bolhas referente ao relacionamento entre três facetas. O eixo x mostra o método de pesquisa utilizado e a faceta de contribuição. O eixo y representa granularidade (Tabela 7). O tamanho dos círculos (ou bolhas) consiste no número de estudos primários classificados com base nos critérios ao longo dos eixos x e y.

(1) Falta de proposta de solução. Existe um padrão de distribuição no uso do tipo de pesquisa. Embora estudos cruciais propondo soluções (10%, 7/70) tenham sido uma das pesquisas menos comuns. Estudos empíricos (62,86%, 44/70) apresentaram o maior número. Os demais métodos juntos registraram um número pequeno (27,14%, 19/70), ou seja, pesquisas de validação (4,29%, 3/70), artigos filosóficos (1,43%, 1/70), experiência pessoal (7,14% , 5/70) e artigos de opinião (14,29%, 10/70). Ou seja, o número de estudos realizando estudos empíricos foi quase quatro vezes maior. Isto pode significar a necessidade de estudos empíricos que triangulem os resultados empíricos recolhidos até agora. Além disso, isto pode significar que são necessárias propostas de soluções inovadoras para gerar novas técnicas de integração.

(2) Estratégias de resolução de conflitos em integração de software. Um grande desafio na pesquisa de integração de código-fonte é o desenvolvimento de estratégias eficazes de resolução

de conflitos [P48]. À medida que os códigos-fonte desenvolvidos em paralelo ou por equipes distribuídas são integrados, podem surgir conflitos onde diferentes alterações se sobrepõem ou se contradizem. Investigar e desenvolver técnicas para resolver tais conflitos de forma automática e inteligente é crucial para melhorar a eficiência e a precisão do processo de integração.

(3) Compreendendo a semântica das alterações de código. Compreender a semântica e a intenção por trás das alterações de códigos-fonte é um desafio significativo na integração de código-fonte. A integração puramente baseada em semelhanças textuais ou sintáticas pode levar a resultados incorretos. Investigar métodos para capturar o significado semântico das alterações de códigos-fonte e aproveitar esse entendimento para orientar o processo de integração pode melhorar a precisão e a qualidade do códigos-fonte integrados. Estudos recentes já apontam investigações nessa direção (OCHOA et al., 2022; HANAM; MESBAH; HOLMES, 2019; FANG et al., 2020; SUNG et al., 2020).

(4) Escalabilidade e desempenho em casos de integração de software. A integração de código-fonte torna-se cada vez mais desafiadora à medida que o tamanho e a complexidade dos projetos de software e o número de filiais aumentam. A escalabilidade e o desempenho das técnicas de integração tornam-se fatores críticos a serem considerados [P26]. A investigação de abordagens que possam lidar com grandes quantidades de códigos-fonte de forma eficiente e eficaz, mantendo ao mesmo tempo a integridade e a correção de códigos-fonte integrados, caracterizando-se como uma área importante de pesquisa da engenharia de software moderna.

(5) Manipulação de artefatos não textuais em cenários de integração. A integração de trechos de códigos-fonte geralmente envolvem não apenas a análise textual, mas também à de artefatos não textuais como imagens, arquivos de configurações e de dados. A incorporação desses artefatos não textuais no processo de integração apresenta desafios únicos. Investigar técnicas para lidar com esses artefatos, resolver conflitos e garantir consistência entre diferentes tipos de artefatos é uma área de interesse na pesquisa de integração de código-fonte.

(6) integração em Ambientes Dinâmicos e em Evolução. Projetos de software não são estáticos e passam por mudanças e evoluções contínuas. A integração de códigos-fonte em ambientes dinâmicos e em evolução, onde novos recursos são acomodados, as dependências também serão atualizadas, apresenta desafios. Investigar estratégias para lidar com a integração em tais ambientes, onde os trechos de códigos-fonte passam por modificações frequentes poderá contribuir para novas técnicas de integração mais robustas e eficientes.

Esses desafios destacam áreas importantes de investigação na pesquisa de integração de código-fonte, com foco na resolução de conflitos, compreensão semântica, escalabilidade, manipulação de artefatos não textuais e integração em ambientes dinâmicos. Enfrentar esses desafios pode levar a avanços nas técnicas de integração e melhorar a eficiência e eficácia do processo de desenvolvimento de software. Explorar esses desafios é fundamental para produzir insights e abrir novos caminhos de pesquisa até então inexplorados. Além disso, diretrizes adequadas de integração e suporte a ferramentas de versionamento voltadas aos usuário impulsionando o trabalho colaborativo em larga escala.

3.5 Ameaças à Validade

Essa Seção tem como finalidade investigar como podem ser mitigadas as ameaças construtivas, internas e de conclusão, apresentando algumas sugestões de melhorias para o aprimoramento de desenvolvimento em pesquisas futuras. Excluímos a validade externa do SMS pois não foi possível ser generalizada para a indústria de software. Pois, a dificuldade em estabelecer uma relação precisa entre os diferentes tipos de abordagens de pesquisa (enquanto algumas são baseadas em estudos empíricos, outras em estudos estatísticos) torna o trabalho de eliminação de ameaças complexos e propenso a falhas. Segundo (CARBONERA; FARIAS; BISCHOFF, 2020), identificar o escopo dos recursos (detalhado ou genérico) torna-se necessário para assegurar a completude e a precisão das bases de dados de domínio público e dos motores de busca, e auxilia também a identificar vários tipos de ameaças a construção de validade do experimento.

Validação Interna. Foi identificada duas grandes ameaças. Primeira, a dificuldade em se conseguir estabelecer uma relação entre as abordagens existentes devido aos muitos conceitos existentes (dentre eles, empírico e estatístico). Caracterizando-se como aspectos heterogêneos e divergentes entre as avaliadas. Foi realizada uma análise muito criteriosa com a finalidade de identificar as características semelhantes ou em comum. Em seguida, a dificuldade de identificar e classificar o escopo de cada um dos estudos primários. Para mitigar essas ameaças, foi compreendida cada uma das abordagens pesquisadas, para em seguida classificar cada um dos estudos selecionados.

Quanto ao processo de filtragem, foram realizados três ciclos de avaliação completos evitando assim qualquer falha. Todos os estudos primários selecionados e avaliados nesta pesquisa estão listados no Apêndice A, assegurando que o processo de seleção dos estudos primários foi realizado o mais imparcial possível, sendo estabelecido que a seleção de estudos primários foi uma atividade realizada em várias etapas, documentando quais os motivos da inclusão ou da exclusão e os critérios de seleção, conforme apresentados na Seção 3.1.2.

Validação de Construção. A possível classificação incorreta e a exclusão de estudos relevantes foram duas ameaças iminentes no SMS realizado. Na tentativa de eliminar esse problema foi estabelecido um protocolo de revisão robusto, com critérios de inclusão e exclusão bem definidos e auditáveis durante todas os ciclos.

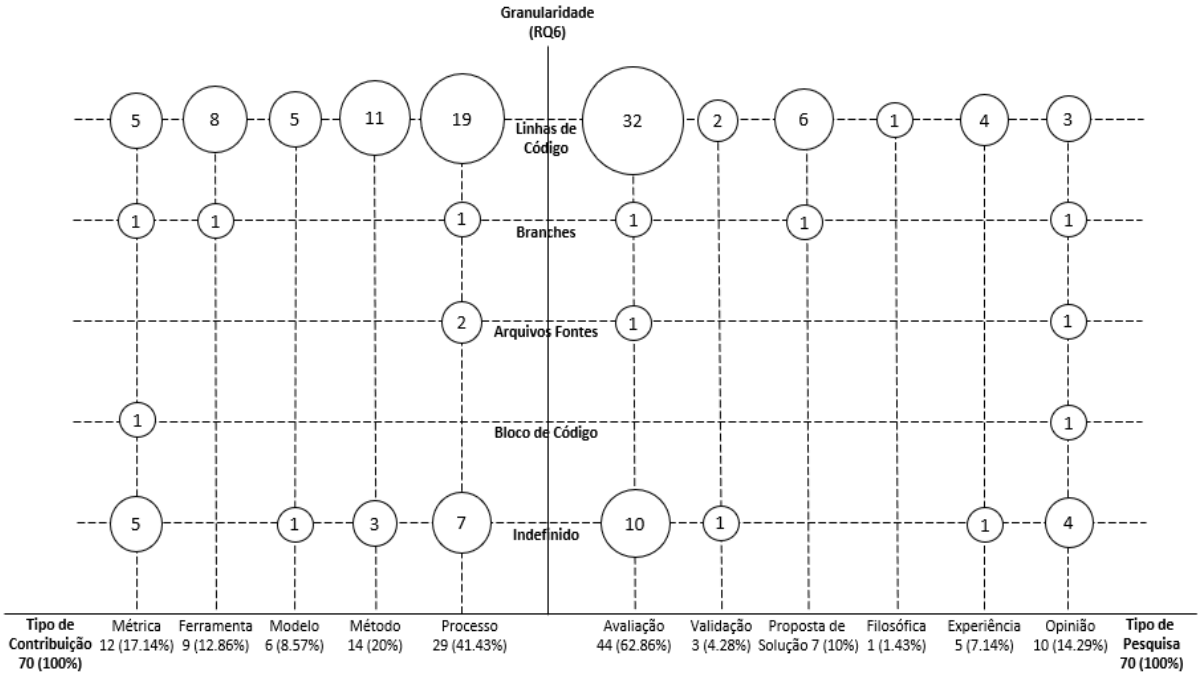
Validação de Conclusão. Essa ameaça está relacionada aos problemas que possam afetar a confiabilidade das conclusões obtidas. Foi observado que o processo para a seleção dos estudos pode ter sido influenciado por aspectos que foram interpretados indevidamente, seja por dúvida ou conhecimento. Para mitigar esse problema, foram estabelecidos critérios de inclusão e exclusão minimizando o risco durante o processo de seleção e filtragem, conforme apresentados na Seção 3.1.2.

Outra possível ameaça está relacionada à classificação incorreta dos estudos. Quando classificações duvidosas ou conflitantes foram detectadas, foi conduzida uma avaliação robusta para obter-se uma definição objetivo do tipo do estudo. A formulação de strings de busca é

um processo na qual algumas ameaças podem ocorrer, pois não será possível evitar que algum termo específico e importante tenha sido desconsiderado, que pode ter sido devido ao desconhecimento de algum sinônimo importante. Por fim, todas as conclusões dessa pesquisa foram obtidas após a coleta e avaliação dos resultados, mitigando qualquer tipo de problema.

3.6 Oportunidades de Pesquisa

A Tabela 12 detalha a comparação entre os cinco estudos selecionados, apresentando as diferenças e as similaridades. Assim, é possível relacionar os seguintes aspectos: (1) Ausência de estudos que reportem evidências empíricas quanto ao tempo de resolução de conflitos em equipes distribuídas, (2) Desenvolvimento de estudos que avaliem a influência de variáveis dependentes e independentes na integração dos artefatos e por fim, (3) Ausência de estudos empíricos que avaliem a qualidade dos resultados dos softwares de controle e versionamento existentes na atualidade. A Figura 12 demonstra a partir das evidências quantitativas encontradas, quais são os anos que mais estudos foram publicados e os assuntos que a comunidade acadêmica tem demonstrado maior ou menor interesse em desenvolver pesquisas. A Figura 12 demonstra algumas oportunidades de pesquisa, classificadas de acordo com as quantidades encontradas, dentre elas: (1) Desenvolvimento de novas propostas ou abordagens de resolução de conflitos. E por fim, (2) Técnicas de validação de resultados obtidos a partir da execução de experimentos empíricos.



Fonte: Criado pelo Autor.
Figura 12: Distribuição dos Estudos Primários por Granularidade.

Evidenciou-se que a área de integração de arquivos fontes vem aprimorando as abordagens

existentes e atuando no desenvolvimento de novas linhas de pesquisas. Os resultados apresentados na Figura 12 comprovam que os estudos que utilizaram abordagens de avaliação de processos ou que realizaram experimentos empíricos representaram a maioria dos estudos desenvolvidos. Detectou-se também, que os estudos que investigaram ou apresentaram novas metodologias de avaliação representaram 20% (14/70). Estudos que apresentaram a opinião dos autores quanto a avaliação de alguma abordagem investigada ou ferramenta existente representou 14,29% (10/70).

No entanto, apenas 12.86% (09/70) dos estudos analisados detalharam e apresentaram o desenvolvimento de novas ferramentas ou plugins para auxiliar na detecção e resolução de conflitos. Por fim, ficou comprovado que o principal tipo de granularidade avaliada nos experimentos ainda é o tipo linha de código. Portanto, por todas essas características apresentadas, comprovou-se que nenhuma abordagem inovadora que apresentassem diferentes tipos de análise e resolução de conflitos inovadores aos existentes na atualidade foi apresentada, comprovando a existências de várias lacunas e dificuldade de evolução dessa importante área para a comunidade acadêmica e para a indústria.

3.7 Conclusões e Trabalhos Futuros

O objetivo principal desse SMS foi fornecer uma classificação abrangente e uma análise temática sobre integração de trechos de códigos-fonte nas últimas duas décadas. Assim, com a finalidade de atingir os objetivos estabelecidos, foi realizado um meticuloso e detalhado estudo de mapeamento sistemático dos principais características e aspectos técnicos sobre integração de trechos de códigos-fonte existente na atualidade, seguindo rigorosamente às diretrizes muito bem definidas pela metodologia PRISMA. As análises realizadas abordaram nove questões de pesquisa cuidadosamente elaboradas, contribuindo para a correta compreensão dessa importante área da engenharia de software moderna.

Após rigoroso processo de filtragem, foram selecionados setenta estudos primários. Conseguiu-se identificar algumas importantes lacunas contribuindo para o desenvolvimento de trabalhos futuros, dentre os quais: (1) Aplicação e avaliação de algoritmos de evolução genética para análise de árvores de elementos sintáticos; (2) Avaliação de abordagens de integração utilizando técnicas semiestruturadas; (3) Desenvolvimento de técnicas cognitivas para avaliar o estresse em tarefas de integração; e por fim, (4) Base de dados histórica gerada por técnicas de integração que poderão ser aplicadas em experimentos futuros. Portanto, foi possível considerar que os resultados e descobertas elucidados serviram como fonte de inspiração para outros pesquisadores, instigando novas investigações e explorações. Assim, esse estudo representou um passo inaugural no sentido de melhorar a abrangência e a sofisticação das pesquisas relativas às metodologias atuais empregadas na integração de trechos de códigos-fonte.

4 ABORDAGEM PROMERGE

Este Capítulo apresenta a *ProMerge* que trata-se de uma abordagem para auxiliar desenvolvedores na detecção e resolução proativa de conflitos gerados a partir da integração de trechos de códigos-fontes utilizando histórico de contexto. As informações geradas a partir da compilação e das integrações dos trechos de códigos-fonte alterados junto ao repositório foram armazenadas. Na ocorrência de conflitos em branches diferentes (ou não) foi avaliado o grau de severidade das funcionalidades relacionadas às ocorrências dos conflitos. Outro aspecto importante a ser mencionado, refere-se ao committing time, que caracteriza-se pela avaliação do intervalo de tempo entre as sequência dos commits realizados, contribuindo para a geração de indicadores de desempenho e produtividade. Portanto, esse capítulo responde portanto à QP-2 conforme detalhada na Seção 1.3.

Essa pesquisa é baseada no trabalho inicialmente desenvolvido por (ARMINO, 2016). Todas as funcionalidades inicialmente desenvolvida por essa pesquisa e que estão relacionadas à detecção proativa e a resolução colaborativa de conflitos foram mantidas. Foram implementados os conceitos de nível de complexidade das funcionalidades, avaliação da taxa de erro e de corretude durante a integração dos trechos de códigos-fontes alterados ou acomodados. O referencial teórico foi atualizado para as funcionalidades existentes, e apresentada a fundamentação teórica necessária ao desenvolvimento das novas funcionalidades propostas pela *ProMerge* conforme detalhados na Seção 4.1.

Esse Capítulo está organizado da seguinte forma: A Seção 4.1 detalha os requisitos identificados na literatura e as abordagens que serão implementadas pela *ProMerge* fornecendo suporte a detecção proativa e na resolução colaborativa de conflitos. A Seção 4.3 apresenta o detalhamento da arquitetura baseadas em componentes adotada pela *ProMerge*. A Seção 4.4 descreve as funcionalidades dos principais algoritmos desenvolvidos. Por fim, a Seção 4.5 apresenta informações relevantes sobre os componentes, aspectos técnicos de implementação, linguagem de programação e o framework escolhido para a implementação da *ProMerge*.

4.1 Principais Requisitos

Essa Seção detalha os requisitos implementados pela *ProMerge*, e fornecem suporte e qualificam a percepção e o entendimento pelo desenvolvedor sobre os diferentes tipos de conflitos existentes, e inclusive a correta compreensão dos conflitos detectados. Esses requisitos são derivados de uma revisão abrangente e imparcial da literatura sobre o tema pesquisado e dos trabalhos e experimentos apresentados em [P42],(DULLEMOND; GAMEREN; SOLINGEN, 2014; ALE EBRAHIM; AHMED; TAHA, 2009; ARMINO, 2016) e que foi realizada no capítulo 3. Segundo os autores, o processo de detecção e resolução de conflitos devem ser realizados, sempre que possível, em modo colaborativo e proativa com a finalidade de mitigar possíveis problemas de conflitos relacionados a integração de arquivos de código-fonte.

Portanto, nove requisitos foram identificados a partir da análise das lacunas em aberto encontrada pela revisão sistemática da literatura. Os requisitos R1, R2, R3, R4 e R5 foram inicialmente propostos e implementados em (ARMINO, 2016). Todos os aspectos desses requisitos foram avaliados, testados e melhorados por essa pesquisa, principalmente no requisito R5 que a partir da sua ocorrência de um conflito. O resultado proveniente da compilação dos arquivos fontes existentes no projeto avaliado será armazenado como informações de histórico de contexto e disponibilizadas aos desenvolvedores para compreensão e resolução dos conflitos pelos desenvolvedores. Os requisitos R6, R7 e R8 foram implementados por essa pesquisa não existindo nenhum registro de implementação nos trabalhos avaliados e listado no apêndice A, todos os requisito encontram=se detalhados abaixo:

- **Detecção e propagação antecipada de conflitos (R1).** Em um ambiente de desenvolvimento colaborativo, os desenvolvedores podem trabalhar em paralelo ou de forma distribuída, e as alterações realizadas podem gerar conflitos sintáticos ou semânticos. Esses conflitos geralmente são detectados somente quando for realizada a integração dos códigos-fontes alterados junto ao repositório do projeto. Após a resolução dos conflitos, a propagação da solução geralmente acontece de maneira tardia gerando problemas nos códigos-fontes, e que estão relacionados a algum tipo de anomalia ou inconsistência do projeto, como por exemplo: erros de compilação, inexistência de variáveis ou funcionalidades, erros sintáticos ou semânticos entre outras possibilidades.

  Esse requisito busca realizar a detecção bem como a propagação antecipada dos conflitos gerados a partir da alteração dos códigos-fontes pelos desenvolvedores nas respectivas estações de trabalho. Os históricos das modificações serão armazenados identificados pelo usuário do desenvolvedor que realizou a alteração. Após a integração dos códigos-fontes, os históricos das compilações serão armazenados e as mensagens resultantes da compilação serão de fácil interpretação pelos desenvolvedores, o que irá auxiliar na resolução dos conflitos e contribuindo para a composição de indicadores de qualidade e produtividade. A literatura atual, explora parcialmente os seguintes requisitos:

  1. Códigos-fontes alterados em paralelo podem ser analisados em tempo real não sendo possível a detecção e propagação dos conflitos encontrados [P64],(BIEHL et al., 2007),(DEWAN; HEGDE, 2007).
  2. Inexistência da análise de histórico de contexto quando os conflitos forem detectados e propagados (ARMINO, 2016).

  Portanto, esse requisito será melhor investigado e avaliado por essa pesquisa, com a finalidade de analisar os resultados obtidos pelo experimento controlado, fornecendo indicadores mais assertivos e objetivos.

- **Apoio colaborativo na resolução de conflitos (R2).** Quando os desenvolvedores exercem suas atividades em um ambiente colaborativo, a comunicação e a troca de infor-

mações entre os desenvolvedores possibilita a resolução de forma rápida e precisa dos conflitos gerados a partir da integração de arquivos fontes. Quanto aos tipos de troca de mensagens, pode-se citar: áudio, vídeo, mensagens de texto ou ainda View Sharing e a atualização dos códigos-fontes entre os diferentes espaços de trabalhos (Workspaces) dos desenvolvedores, independentes da integração com o repositório dos projetos.

A literatura atual explora parcialmente esse requisito, e aponta que a melhoria contínua da consciência da comunicação entre os desenvolvedores auxilia na resolução de conflitos de maneira muito mais assertiva (ANDRES, 2002; SIEBDRAT; HOEGL; ERNST, 2009; ALE EBRAHIM; AHMED; TAHA, 2009; DULLEMOND; GAMEREN; SOLINGEN, 2014; PERRY; SIY; VOTTA, 2001; DOURISH; BELLOTTI, 1992; CATALDO et al., 2006). Portanto, verifica-se a importância dessa funcionalidade em ambientes de desenvolvimento colaborativo. Porém, nenhum dos trabalhos avaliados conseguiu auxílio na resolução de conflitos a partir das informações contidas no histórico de contexto, sendo um requisito parcialmente atendido e que a *ProMerge* contempla amplamente.

- **Identificação de conflitos sintáticos e semânticos (R3).** Integração de códigos-fontes conflitantes junto ao repositório, poderá ocorrer erros resultantes da compilação dos códigos-fontes do projeto. A detecção antecipada desses conflitos reduzirá a quantidade de tempo aplicado nas correções, assegurando a integridade do projeto e dos códigos-fontes. Esse requisito busca detectar conflitos sintáticos ou semânticos, onde os conflitos de ordem sintática trata-se de um processo de análise de uma sequência de símbolos léxicos determinando a sua estrutura gramatical conforme os padrões estabelecidos por uma determinada gramática formal (linguagem de programação).

  Os conflitos de ordem semântica estão diretamente relacionados aos significados das instruções, é quando ocorre a validação de uma série de regras que não podem ser verificadas na etapa de análise sintática. A análise semântica percorre a árvore sintática, relaciona os identificadores com seus dependentes de acordo com a estrutura hierárquica a fim de garantir que o programa fonte esteja coerente e que possa ser convertido para linguagem de máquina. A literatura atual aborda parcialmente esse requisito, principalmente quanto aos seguintes aspectos:

  1. Conflitos relacionados aos agrupamentos e suas respectivas ordens [P48].
  2. Aspectos técnicos relacionados a um determinado conflito a partir da classificação topológica [P43,P44].
  3. Estratégias de resolução de conflitos fundamentada em outros conflitos semelhantes e já resolvidos [P10,P12,P41].
  4. Preservação da formatação em maior grau que as demais ferramentas existentes, reduzindo o tempo de execução da avaliação dos códigos-fontes [P41].

A *ProMerge* disponibilizará de forma amigável e intuitiva aos desenvolvedores, mensagens informativas das ocorrências de conflitos e os arquivos fontes que relacionados. Sendo possível a avaliação de severidade dos conflitos, indicando quais ações que devem ser realizadas para assegurar a integridade dos arquivos fontes. Por fim, por todos esses aspectos esse requisito será mais bem investigado por essa pesquisa.

- **Resolução automática de conflitos (R4):** A ferramenta de desenvolvimento sempre que possível, deverá sugerir aos desenvolvedores a melhor abordagem para a resolução dos conflitos identificados. Para a integração de forma automática, o desenvolvedor poderá utilizar as funcionalidades de resolução dos conflitos existente no próprio controlador de versão escolhido, as alterações de dois arquivos envolvidos serão concatenadas em um terceiro arquivo para análise posterior, assegurando que não serem excluídas as alterações realizadas por mais de um desenvolvedor.

  Esse requisito foi atendido parcialmente em (DULLEMOND; GAMEREN; SOLINGEN, 2014; ALE EBRAHIM; AHMED; TAHA, 2009; ANDRES, 2002), onde os autores investigaram a necessidade de que as ferramentas em ambiente de desenvolvimento colaborativo possibilitassem a combinação de dados de vários códigos-fontes, pois trata-se de uma tarefa complexa e muito propensa a erros. A *ProMerge* apresentara essa opção de integração de códigos-fontes dentro do próprio ambiente de desenvolvimento, sendo um processo integrado auxiliando diretamente na resolução dos conflitos existentes.

- **Detecção e notificação de conflitos pós-commit em diferentes workspaces (R5):** Quando um ou mais códigos-fontes forem submetidos ao repositório, o sistema realizará uma checagem em outros espaços de trabalhos (Workspaces) buscando alterações nos arquivos que conflitem diretamente com os arquivos comitados, na ocorrência de outros conflitos os desenvolvedores devem ser avisados para sincronizarem os códigos-fontes imediatamente.

  A literatura explora parcialmente esse requisito também, em (WLOKA et al., 2009) os autores apresentam uma análise que identifica as mudanças pendentes que podem ser liberadas antecipadamente e que não causem falhas nos testes existentes, ainda que na presença de falhas no espaço de trabalho (Workspaces) de um desenvolvedor. Sendo também identificada em [P38], onde as avaliações dos conflitos são realizadas nos códigos-fontes que estão disponibilizados no repositório.

  Assim, verifica-se a importância desse requisito em ambientes de desenvolvimento colaborativo. Pois, a partir dos fontes integrados de um determinado ambiente de desenvolvimento junto no repositório, será executada a avaliação dos demais ambientes disponíveis, atuando como um agente ativo e analisando as alterações com a finalidade de detectar conflitos nos demais ambientes de desenvolvimento.

- **Histórico de contexto e resolução de conflitos (R6):** Quando forem comitados no repo-

sitório os trechos de códigos-fonte alterados, o *ProMerge Server* irá compilar os arquivos fontes do projeto e armazenará os históricos contendo as mensagens resultantes da avaliação dos arquivos fontes do projeto, detalhando informações dos conflitos gerados (caso existirem) e data e a hora das ocorrências. Essas informações serão importantes para calcular o committing time, que é o tempo de resolução dos conflitos ou auxiliando os desenvolvedores na compreensão e resolução dos conflitos detectados. Portanto, as informações de histórico de contexto, caracterizam-se por serem informações geradas após o commit das alterações e caracterizam-se por apresentar informações importantes para a resolução de conflitos, interpretação do grau de severidade ou por fim, nos cuidados e boas práticas necessárias assegurando a integridade do projeto.

Na literatura, existem uma quantidade muito limitada de estudos que abordam esse tema, em (KUDRJAVETS et al., 2022), os autores avaliaram o tema e concluíram que a quantidade de tempo aplicado na resolução de conflitos entre as integrações fornecerá indicadores de qualidade e produtividade das equipes e de seus desenvolvedores individualmente. Esse requisito foi proposto e implementado pela *ProMerge*, e irá auxiliar na criação de métricas de avaliação do tempo e quantidade de tempo aplicados para a resolução de conflitos e, no desenvolvimento de boas práticas com a finalidade de tentar assegurar a integridade do projeto e dos códigos-fontes.

- **Análise de impacto de conflitos indiretos (R7):** Foi criado um sistema de armazenamento da relação dos métodos influenciados por um conflito indireto. Quando for alterada alguma funcionalidade que é invocada em outras partes do sistema e for detectado tipo de conflito, a alteração deverá ser efetuada nos demais trechos onde a funcionalidade é requisita, garantindo a integridade dos códigos-fontes. Esse controle irá auxiliar na definição da severidade de uma determina funcionalidade e quais os cuidados necessários que necessitam ser realizados para assegurar a integridade do processo.

  Após a análise da literatura atual, foi detectado que esse requisito é explorado de forma parcial. Em [P40,P42,P49,P66] são abordados aspectos das funcionalidades afetadas na ocorrência de conflitos indiretos, mas sem o registro de informações de histórico de contexto. O registro dessas informações irá auxiliar na definição e na definição do grau de severidade dos conflitos gerados. Assim, devido a sua importância esse requisito será implementado pela *ProMerge*.

- **Avaliação do grau de severidade dos conflitos (R8):** Na ocorrência de conflitos, deverá ser avaliado qual a sua complexidade dentro dos códigos-fontes, ou seja, deverá ser identificada a sua importância dentro do contexto de um projeto. Foi desenvolvido na *ProMerge* uma escala de avaliação da severidade dos conflitos, onde foi considerada a quantidade de vezes que uma determinada funcionalidade é requisitada e multiplicada pela sua valoração. A valoração de uma determinada funcionalidade será estabelecida através da avaliação da experiência técnica dos desenvolvedores envolvidos.

Esse requisito não foi desenvolvido ou investigado por nenhum dos estudos avaliado por essa pesquisa, e não foi encontrada na literatura nenhuma pesquisa que abordasse esse tema. Devida sua importância, esse requisito será implementado dentro da *ProMerge* disponibilizando informações aos desenvolvedores da avaliação da severidade dos métodos conflitantes. A pontuação atribuída inicialmente ao grau de severidade de uma determinada funcionalidade é de 50 pontos, e para cada referência a uma funcionalidade é acrescido mais cinquenta pontos, assim se uma funcionalidade possuir duas referências nos arquivos fontes de um projeto, a sua pontuação final será e centro e cinquenta pontos, onde cinquenta pontos refere-se a sua declaração e mais cem pontos devido às duas referências encontradas.

A definição de um modelo de pontuação foi necessário para poderem ser estabelecidos os limites (inicial e final) de cada um dos quatro níveis de severidade adotados pela *ProMerge*. Assim, atribuição de cinquenta pontos para cada referência de uma determinada funcionalidade existente nos trechos de códigos-fontes foi definida. Esse valor não obedeceu à nenhum padrão existente ou evidência técnica de algum dos estudos avaliados. Essa categorização foi definida em caráter primário de avaliação podendo ser adaptada em diferentes faixas de valores. Por fim, foi necessária a definição de uma escala de avaliação, e quatro níveis foram estabelecidos: (1) Baixa: Valor até cinquenta pontos, (2) Média: Valor maior que cinquenta e menor ou igual à cem pontos, (3) Alta: Valor superior à cem e menor ou igual à trezentos pontos e por fim, (4) Séria: Pontuações superiores à trezentos pontos. A Tabela 13 apresenta os diferentes graus de severidade aplicados por essa pesquisa:

Grau	**Intervalo**
Baixa	Até 50 pontos
Média	$> 50 \leq 100$ pontos
Alta	$> 100 \leq 300$ pontos
Séria	> 300 pontos

Tabela 13: Grau de Severidade.

- **Análise e resolução de conflitos (R9):** Quanto aos aspectos relacionados à resolução de conflitos, é muito importante avaliar o tempo transcorrido e a quantidade de tentativas de integração realizadas até a correta efetivação (isentos de conflitos) dos códigos-fontes alterados junto ao repositório. Os registros históricos dos intervalos de tempos e das quantidades de tentativas realizadas serão armazenados com a finalidade de auxiliará na formação de indicadores de produtividade e de qualidade dos desenvolvedores. E na classificação e na quantidade de conflitos ocorridos podendo contribuir para a geração de boas práticas. Na literatura atual existem muito poucos estudos e pesquisas desenvolvidas que abordam esse importante aspecto, em (KUDRJAVETS et al., 2022) os autores avaliam

através de experimentos controlados aos seguintes aspectos:

1. Intervalo de tempo entre a primeira e a última submissão de códigos-fontes ao repositório com a integridade dos artefatos assegurada.
2. Espaço de tempo entre os envios e os retornos das entregas.
3. Tempo de espera entre a aceitação e a integração.

Assim, verifica-se a importância de melhor investigar esse requisito, pois através da avaliação dessas informações será é possível estabelecer indicadores de desempenho e de produtividade das equipes e dos desenvolvedores, como por exemplo: quantidade de conflitos ocorridos durante um determinado intervalo, tempo médio para resolução dos conflitos, quantidade de conflitos entre outros, qual o grau de severidade que mais tem ocorrido entre outros indicadores.

4.2 Análise do Estado da Prática

Essa Seção apresenta um resumo das principais ferramentas de desenvolvimento colaborativo disponíveis na atualidade, apresentando algum tipo de relação com as funcionalidades desenvolvidas pela *ProMerge*. Sendo selecionadas a partir das evidências encontradas nos estudos avaliados em observancia às tendências quanto a popularidade e a usabilidade pela indústria de software. Dentre as quais: Visual Studio Live Share (VSCODE, 2024), Saros (SAROS, 2024), FirePad (FIREPAD, 2024), Remote Collab (REMOTECOLLAB, 2024), Floobits (FLOOBITS, 2024), SoManyConflicts (SOMANY, 2024) e PACCS (ARMINO, 2016).

A ferramenta Visual Studio Live Share (VSCODE, 2024) é uma das mais populares e mais conhecidas na atualidade, permite a instalação de extensões para o desenvolvimento em linguagem Java, conectando nativamente com o gerenciador Git (GITHUB, 2024), mas é possível utilizar outros gerenciadores também, é um software de código-fonte aberto e possui o recurso *Live Share* onde é possível o desenvolvimento em pair programming e acompanhamento em tempo real das alterações realizadas nos códigos-fontes.

O Saros (SAROS, 2024) trata-se de um plugin para o Eclipse (ECLIPSE, 2024), muito utilizada por equipes de desenvolvedores da linguagem Java. Entre seus principais recursos pode-se citar: comunicação via chat, revisão conjunta, programação distribuída e programação em pares distribuída suportando até cinco desenvolvedores simultaneamente. Eessa ferramenta tem como finalidade auxiliar os desenvolvedor na resolução dos conflitos ocorridos após a integração dos arquivos fontes alterados, não possuindo nenhum recurso de resolução proativa de conflitos, não disponibilizando aos usuários nenhuma informação dos histórico das integrações realizadas e por fim, nenhuma informação referente ao committing time das equipes e seus desenvolvedores.

O FirePad (FIREPAD, 2024) é outro plugin para a ferramenta Eclipse (ECLIPSE, 2024), possibilitando apenas suporte ao desenvolvimento colaborativo, possuindo funcionalidades para

o acompanhamento em tempo real das alterações realizadas, visualização da lista de usuários conectados, sincronização da posição do cursor e por fim, destaque de texto e verificação de versão.

O Remote Collab (REMOTECOLLAB, 2024) é outro plugin de desenvolvimento colaborativo, permite a criação de sessões de comunicação com outro usuário para a realização de pair programing e acompanhamento em tempo real das alterações, possui conexão com o GitHub(GITHUB, 2024) onde é possível realizar as alterações e a resolução dos conflitos nos arquivos de código-fonte. Por fim, não possui suporte a compilação e avalição de conflitos.

O Floobits (FLOOBITS, 2024) também é um plugin de desenvolvimento colaborativo, permite o acompanhamento de desenvolvimento em tempo real, adição de desenvolvedores para atuar em pair programming, como no RemoteCollab (REMOTECOLLAB, 2024), possui conexão com o GitHub (GITHUB, 2024) onde é possível realizar as alterações e resolução de conflitos nos arquivos de código-fonte, e também não possui suporte a compilação e avalição de conflitos.

O SoManyConflicts (SOMANY, 2024) é um plugin de desenvolvimento colaborativo para a ferramenta Visual Studio Live Share (VSCODE, 2024), possui conexão com o Git (GITHUB, 2024) onde é possível agrupar conflitos de integração relacionados (dependente, semelhante ou próximo) e ordenar topologicamente, a ferramenta consegue sugerir interativamente qual o próximo conflito relacionado a ser resolvido a partir de outros conflitos relevantes e já resolvidos.

E por fim, o PACCS (ARMINO, 2016) que serviu como base para o desenvolvimento dessa pesquisa, possui a detecção proativa de conflitos diretos e indireto e checagem pós-commit. Novos requisitos técnicos foram implementados e que não foram detectados em nenhum dos estudos listados no Apêndice A, principalmente quanto ao committing time (intervalo de tempo entre os commits) e avaliação de severidade dos conflitos. A Tabela 14 detalha a análise comparativa das ferramentas avaliadas, todos os critérios avaliados estão definidos na Seção 4.1, os critérios R8 e R9 não possuem suporte através de nenhuma das ferramentas avaliadas.

Ferramenta	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**
FirePad	○	●	●	○	○	○	○	○	○
Floobits	○	●	○	○	○	○	○	○	○
Visual Studio Live Share	●	●	●	◗	○	○	○	○	○
Remote Collab	○	●	●	○	○	○	○	○	○
Saros	◗	●	○	○	○	○	○	○	○
SoManyConflicts	○	●	●	○	○	○	○	○	○
PACCS	●	●	●	●	●	○	○	○	○
ProMerge	●	●	●	●	●	●	●	●	●

Tabela 14: Análise Comparativa das Ferramentas e Publicações Relacionadas.

A ProMerge trata-se de uma evolução direta da ferramenta PACCS (ARMINO, 2016) uma vez que as funcionalidades R1, R2, R3, R4 e R5 que inicialmente desenvolvidas no PACCS (ARMINO, 2016) foram mantidas e novas e importantes funcionalidades foram acomodadas. As

funcionalidades R6, R7, R8 e R9 não foram contempladas por nenhuma das ferramentas avaliadas conforme detalhado na Tabela 14 e foram obtidas a partir do SMS realizado por essa pesquisa e conforme detalhado no Capítulo 3. Todas as funcionalidades foram obtidas a partir de uma cuidadosa e meticulosa análise dos PMs detalhados no apêndice A e foi uma atividade realizada em 03 ciclos para poder assegurar a validade dos resultados. Evidenciou-se que várias das ferramentas analisadas não apresentavam nenhuma evolução ou melhoria técnica desde a sua concepção e publicação dos estudos, o que contribuiu de forma assertiva na definição dos novos requisitos técnicos apontados.

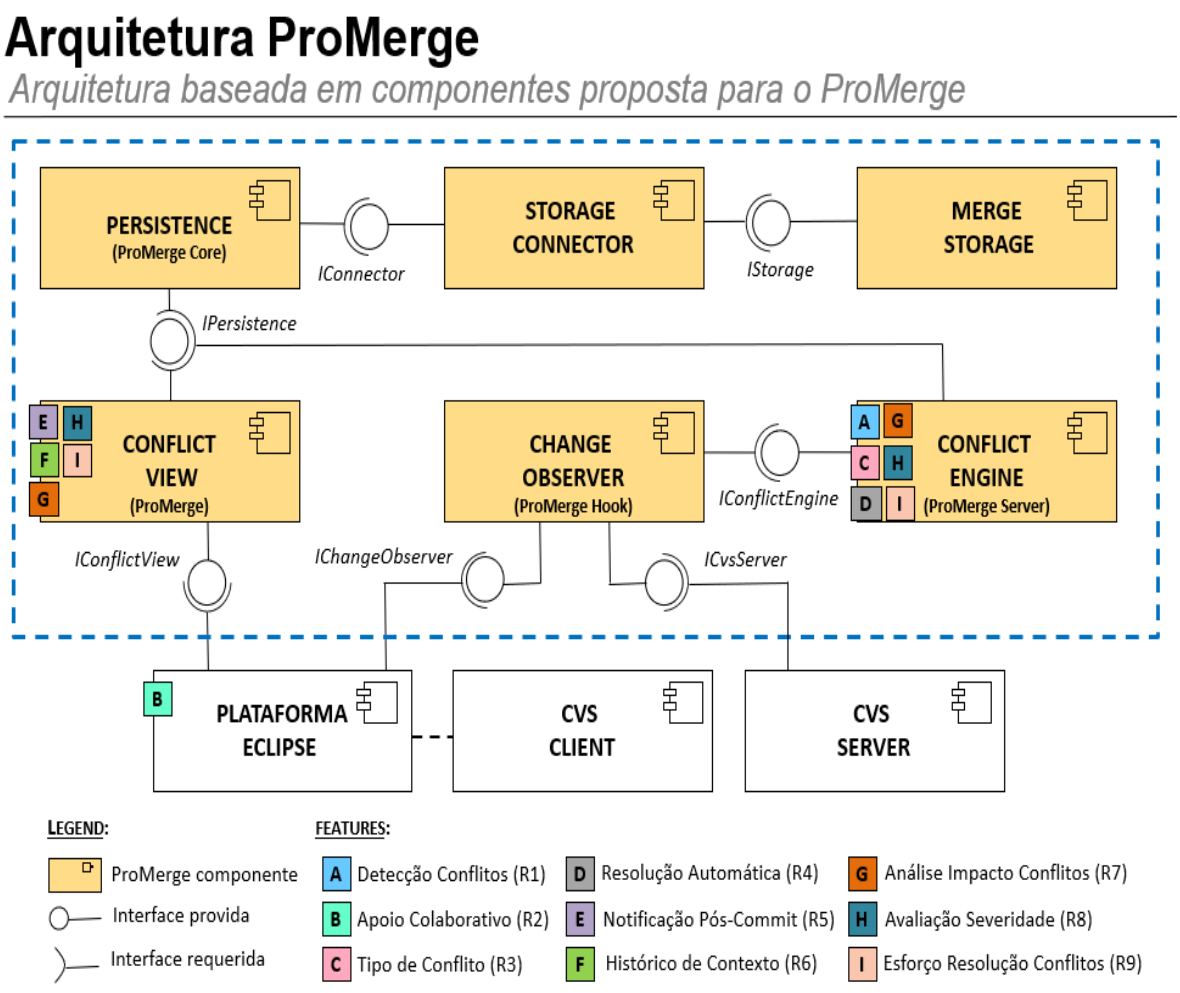
Portanto, o atendimento dos critérios R8 e R9 será o foco principal dessa pesquisa, não ignorando a importância das demais funcionalidades relacionadas anteriormente. A partir desses requisitos, novas funcionalidades serão agregadas na resolução proativa de conflitos da *ProMerge*. Fornecendo mais informações de desempenho e produtividade das equipes e dos desenvolvedores.

4.3 Arquitetura

A *ProMerge* tem a sua arquitetura totalmente desenvolvida em componentes. Esse padrão foi selecionado devido a facilidade de isolamento e desacoplamento entre os componentes, atuando no gerenciamento da complexidade e dos arquivos fontes, segregando as responsabilidades distintamente. Outro importante aspecto está na facilidade de reutilização dos artefatos desenvolvidos. A comunicação entre os componentes é realizada através de contratos, representados pelas figuras do contratante, que é o componente que envia as mensagens, e do contratado que é o componente que recepciona as mensagens do contratante realizando o seu processamento. Por fim, o contratado disponibiliza o retorno das mensagens processadas. Portanto, cada um dos componentes da *ProMerge* possui um ou mais componentes, e cada um desses componentes possui uma ou mais funcionalidades (artefatos) que atuam de forma independente podendo se inter-relacionar ou não, e que estão representados na Figura 13. Os principais artefatos existentes nos componentes da arquitetura da *ProMerge* estão abaixo detalhados:

Os componentes atuam de forma independente, com funcionalidades e responsabilidades distintas podendo ou não se inter-relacionar. A comunicação entre os componentes é realizada através de contratos, onde o contratante encaminhará uma mensagem ao contratado, e esse disponibilizará as informações solicitadas ao contratante como retorno do processamento, conforme detalhadas abaixo:

- **ProMerge View:** Esse componente disponibiliza aos desenvolvedores de forma amigável, mensagens intuitivas referentes às ocorrências dos conflitos e informações dos históricos de contexto relacionadas às integrações de códigos-fontes realizadas. Através dele será possível realizar algumas operações, como por exemplo, consultar a complexidade e a hierarquia dos métodos, consultar informações dos commits realizados, integrar arquivos e por fim, sobrescrever os arquivos localmente com a versão existente no servidor.



Fonte: Criado pelo Autor.

Figura 13: Arquitetura da *ProMerge*.

- **ProMerge Core:** É composto por todas as funcionalidades e protocolos relacionadas a manutenção de informações no sistema gerenciador de banco de dados, utilizando o Java Database Connectivity (JDBC) para acessar o SGDB PostgeSQL (POSTGRESQL, 2024). Foram utilizados os seguintes padrões de projeto:

  - **Value Object (VO):** Os VO são definidos como objetos sem nenhum tipo de identidade conceitual e por caracterizar outro objeto. A sua utilização foi devido a vários fatores: (1) Proporcionam o uso de orientação a objetos, encapsulando comportamento em um objeto ao invés da utilização de tipos de dados primitivos; (2) Centralização das regras de validação em um único ponto, podendo ser acessadas através de métodos de extensão; (3) Imutabilidade - vários objetos podem compartilhar o mesmo VO sem restrições; e por fim (4) Tornam explícitos conceitos do domínio que foram anteriormente definidos em atributos relacionados.
  - **Data Access Object (DAO):** É um padrão de projeto onde existe a segregação das regras de negócio e de acesso ao banco de dados. Implementada a partir de uma linguagem de programação orientadas a objetos utilizando a padrão arquitetural MVC. Todas as funcionalidades para manipulação do sistema gerenciado de banco de dados (SGDB), tais como: conexão, mapeamento de objetos para SQL ou DML devem obrigatoriamente serem feitas por classes. A sua utilização é devido a dois importantes aspecto: (1) Descentralização: Eliminam a necessidade de hierarquia, e (2) Abstração: Permitem a utilização de classes abstratas e interfaces.
  - **Model View Controller (MVC):** Trata-se de um padrão arquitetura de software onde o foco principal é a reutilização de código-fonte e na segregação dos conceitos em três camadas interconectadas: modelo, visão e controlador. A apresentação dos dados e a interação direta com os usuários são separados dos métodos e classes que manipulam informações junto ao SGDB. A utilização desse padrão foi devido a importantes características: (1) Reutilização do código-fonte; (2) Possibilidade de escalabilidade da aplicação; e por fim (3) Arquitetura modular.

- **ProMerge Hook:** Após ser realizado o commit das alterações junto ao repositório, o CVS executa o evento pós-commit. O ProMerge Hook é notificado da execução do evento pós-commit, recepciona as informações e os parâmetros contido por esse evento e envia automaticamente uma mensagem contendo as alterações realizadas ao *ProMerge Server* que inicia o processamento da mensagem recebida. Por tanto, após a análise das alterações, o processamento das informações é realizado e os históricos de contextos serão armazenados no SGDB, e disponibilizadas aos desenvolvedores através do *ProMerge View*.

- **ProMerge Server:** Componente onde a detecção proativa de conflitos é processada, e de todos as demais etapas necessários ao gerenciamento dos históricos de contextos e tratamento de logs de processamento. Após a efetivação dos códigos-fontes junto ao servidor

SVN (SVNSERVER, 2024), uma mensagem será enviada ao *PromergeHook* que por sua vez encaminhará ao *ProMergeServer* para realizar o processamento dos arquivos fontes comitados, gerando as informações de histórico de contexto. Como exemplos dessas informações podemos citar: Histórico dos commits contendo a data e hora da efetivação, resultado da compilação dos arquivos fontes integrantes dos projeto, atualização do nível de complexidade e da hierarquia dos métodos, onde são identificadas as classes que realizam a invocação dos métodos alterados e por fim, na ocorrência de conflitos avalia o nível do grau de severidade dos conflitos conforme detalhado na Tabela 13.

- **Conflict View (Promerge):** Esse componente é o responsável pela visualização das informações de contexto, conflitos diretos e indiretos e erros de compilação, foi desenvolvido especificamente para a *ProMerge* na forma de um plugin, ou seja uma recurso que pode ser adicionado ao ambiente de desenvolvimento do Eclipse (ECLIPSE, 2024). Possuindo uma interface amigável que disponibiliza ao usuário a visualização das informações referentes aos conflitos gerados pela integração de códigos-fontes e das demais funcionalidades desenvolvidas. Sendo composto também por mais duas importantes funcionalidades: (1) Sobrescrita no workspace do desenvolvedor dos arquivos locais com versão existente no repositório e, (2) Integração dos arquivos fontes no workspace do desenvolvedor com a versão existente no repositório.

- **Persistence (ProMerge Core):** Realiza a validação e formatação necessária das informações recebidas em alto nível que devem armazenadas junto ao SGDB, permitindo o acesso aos dados do histórico de contexto e ocorrência de conflitos a partir das solicitações e parâmetros recebidos.

- **Storage Conector:** Estabelece a comunicação de forma agnóstica entre o componente de persistência e o repositório de dados através de um conector JDBC.

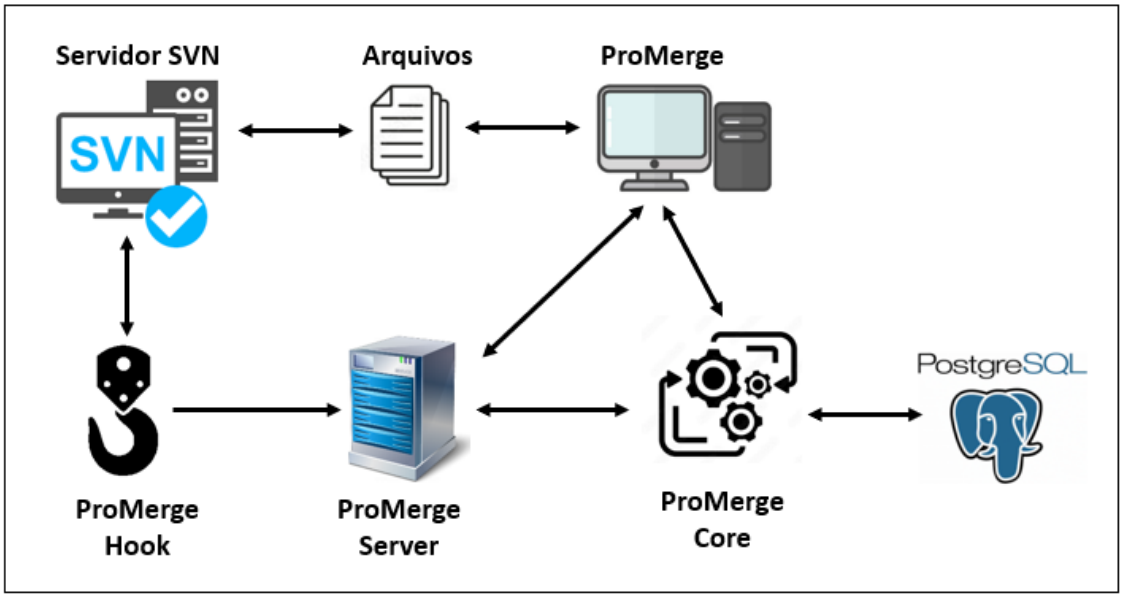
- **Merge Storage:** Armazena as informações dos históricos de contexto geradas a partir das integrações de códigos-fontes comitadas junto ao servidor SVN (SVNSERVER, 2024) e representado pelo SGDB PostgreSQL (POSTGRESQL, 2024).

- **Change Observer (ProMerge Hook):** Esse componente tem com a finalidade recepcionar e delegar às requisições em baixo nível às demais componentes da *ProMerge*, interagindo com a plataforma Eclipse (ECLIPSE, 2024) e com o componente *Conflict Engine*.

- **Conflict Engine (ProMerge Server):** Realiza a integração com o repositório dos códigos-fontes SVN (SVNSERVER, 2024), executando as rotinas responsáveis pela comunicação e processamento das informações com o SGDB PostgreSQL (POSTGRESQL, 2024).

O desenvolvedor terá disponibilizada a *ProMerge* através de um plugin no Eclipse (ECLIPSE, 2024), que permitirá consultar às informações de contexto e dos conflitos ocorridos. Quando

da integração de trechos de códigos-fontes alterados junto ao componente *Servidor SVN*, um evento pós-commit é executado, o componente *Promerge Hook* recebe essa informação e inicializa uma solicitação de processamento junto ao componente *ProMerge Server* que executa o processamento principal e que contêm toda a regra de negócio da *ProMerge*. Assim, quando uma mensagem é recepcionada pelo *ProMerge Server*, o processamento é realizado e serão armezenadas informações no repositório. Disponibilizando aos desenvolvedores o resultado das requisições enviadas através do componente *ProMerge*. Por fim, o componente *ProMerge Core* trata-se de um recurso que centraliza todas as funcionalidades necessárias e comuns a todos os demais componentes da *ProMerge*, tanto para manutenção e a consulta de dados junto ao respositório *PostgreSQL*. Isso, além das funcionalidades para resolução proativa de conflitos e das informações de contexto. O Eclipse (ECLIPSE, 2024) foi a ferramenta selecionada como plataforma de desenvolvimento devido a sua popularidade na indústria de software e na comunidade acadêmica, e também pela praticidade dos recursos disponibilizados aos desenvolvedores e fácil utilização. A Figura 14 fornece o suporte necessário ao entendimento do fluxo das informaçoes dos requisitos já previamente detalhados.

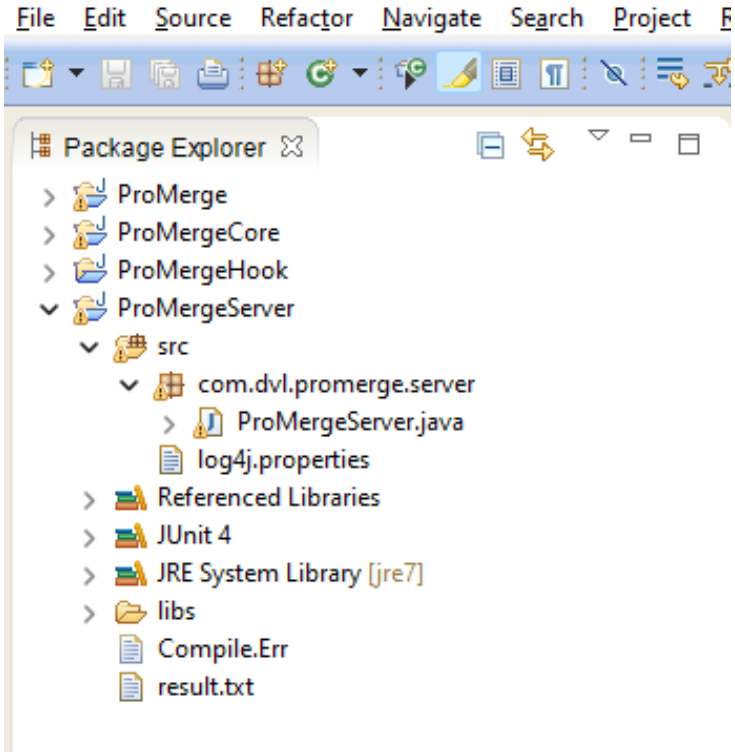


Fonte: Criado pelo Autor.
Figura 14: Componentes da Arquitetura da *ProMerge*.

A estrutura da arquitetura da *ProMerge* está dividida em quatro componentes principais atuando de maneira independentes mas interelacionados. A Figura 15 representa a organização das pastas e dos arquivos de códigos-fontes na atualidade.

4.4 Algoritmos

Essa Seção apresenta os algoritmos que implementam as principais funcionalidades existentes na *ProMerge*, conforme detalhados na Seção 4.1. Contudo, apenas algumas funcionalidades



Fonte: Criado pelo Autor.
Figura 15: Estuturação dos Arquivos Fontes da *ProMerge*.

envolvem o desenvolvimento de algoritmos, por exemplo, para o requisito de apoio colaborativo na resolução de conflitos (R2) será utilizada a ferramenta Saros (SAROS, 2024) para a troca de informações e mensagens entre os desenvolvedores.

A ferramenta Saros (SAROS, 2024) foi selecionada principalmente pelas funcionalidades de comunicação via chat e revisão conjunta, e por tratar-se de um plugin hospedado dentro do próprio Eclipse (ECLIPSE, 2024) facilitando a operacionalidade dos desenvolvedores mantendo o foco e a concentração nas atividades. Na atualidade, existem outras ferramentas que contemplam as necessidades do requisito de apoio colaborativo na resolução de conflitos (R2). Assim, essas ferramentas caracterizam-se por serem ferramentas externas ao Eclipse (ECLIPSE, 2024), podendo criar distrações e perda de foco nos desenvolvedores durante o desenvolvimento das atividades.

O Algoritmo 1 detalha o processo principal da *ProMerge*, envolvendo desde a alteração dos artefatos na área de trabalho (Workspaces) dos desenvolvedores até a integração com o repositório. Algumas etapas de validação foram necessárias, como por exemplo, se o usuário adicionou um ou mais arquivos na sua área de trabalho (Workspaces) e no projeto, e se por sua vez o projeto compilou corretamente ou não, contemplando o requisito de histórico de contexto e resolução de conflitos (R6). Interagindo diretamente com todos os componentes da arquitetura da *ProMerge*, conforme detalhado na Figura 13.

O componente Change Observer por sua vez encontra-se receptivo às mensagens enviadas pelo Eclipse (ECLIPSE, 2024) operada pelo desenvolvedor participante do experimento. As

Algoritmo 1: Processo Principal da *ProMerge*

Entrada: arquivosAlterados
Saída: status

```
1  statusAlteracoes ← AtualizarHistoricosAlteracoes(arquivosAlterados);
2  statusComplexidade ← CalcularComplexidade();
3  statusConflitos ← CompilarProjetoWorkspaceLocal();
4  se statusConflitos ≠ OK;
5   então
6      gravarReg ← GravarConflito(statusMerge, listaArquivos, usuario);
7      se gravarReg ← IsValid então
8          AtualizarListaConflitos();
9      fim
10 senão
11     listaArquivos ← BuscarArquivosOutrosWorkspaces(arquivosAlterados);
12     para i ← 0 até i < Quantidade(listaArquivos) faça
13         statusMerge ← IntegrarArquivo(listaArquivos[i], arquivoAlterado);
14         se statusMerge ≠ IsValid então
15             gravarReg ←
                GravarConflito(statusMerge, listaArquivos[i], usuario);
16             se gravarReg ← IsValid então
17                 AtualizarListaConflitos();
18             fim
19         fim
20     fim
21     compilar ← CompilarProjetoWorkspacerepositorio();
22     se compilar ← ErroCompilacao então
23         gravarReg ← GravarConflito(statusMerge, listaArquivos[i], usuario);
24         se gravarReg ← IsValid então
25             AtualizarListaConflitos();
26         fim
27     senão
28         statusConflito ← AvaliarTipoConflito(listaArquivos);
29         se statusConflito ≠ OK então
30         fim
31         gravarReg ← GravarConflito(statusMerge, listaArquivos[i], usuario);
32         se gravarReg ← IsValid então
33             AtualizarListaConflitos();
34         fim
35     fim
36 fim
```

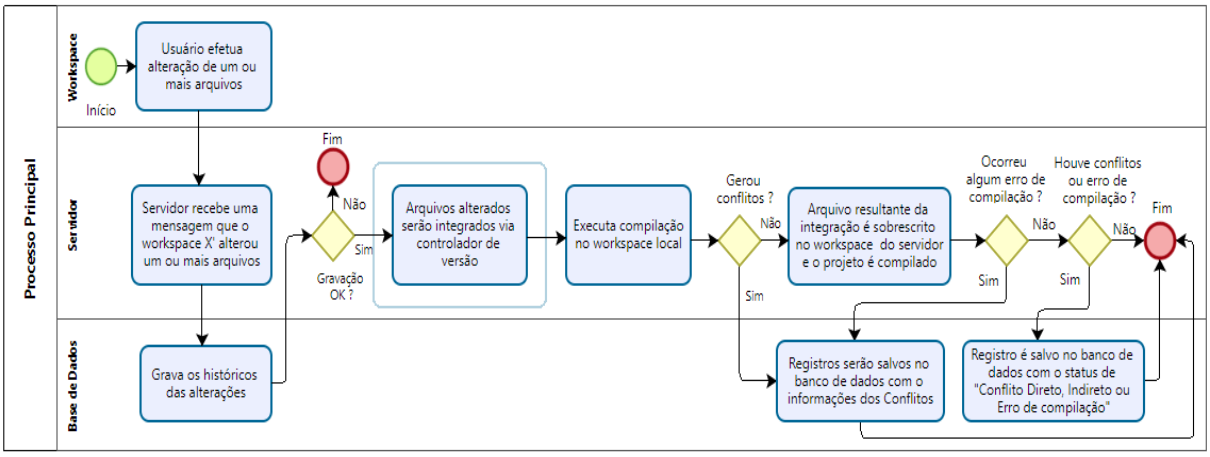
mensagens recepcionadas no componente Change Observer somente serão enviadas via protocolo TCP Socket quando forem salvas as alterações ou realizado o commit junto ao repositório. O componente Conflict Engine estará receptivo as mensagens enviadas pelo componente Change Observer, avaliando a atualização do status do componente à cada meio (0,5 s) segundo. Quando alguma mensagem for recepcionada pelo componente Change Observer será imediatamente processada. Em contrário o componente aguardará o intervalo de meio (0,5 s) segundo e avaliará novamente se alguma mensagem for enviada novamente, executando esse ciclo até que esse componente seja indisponibilizado.

Quado uma mensagem for recepcionada pelo componente Conflict Engine enviada pelo componente Change Observer, que irá realizar o processamento da mensagem que irá comunicar-se via protocolo TCP Socket com o componente Persistence, que por sua vez irá se comunicar com o componente Storage Connector. Por fim, irá se comunicar com o componente Merge Storage realizando a persistência dos dados resultantes do processamento da mensagem inicialmente enviada da plataforma Eclipse (ECLIPSE, 2024). O componente Conflict View deverá ser hospedado pela plataforma Eclipse (ECLIPSE, 2024) como um plugin, e enviará uma mensagem a cada cinco segundos ao componente Persistence via protocolo TCP Socket que irá se comunicar com o componente Storage Connector e por fim irá se comunicar com o componente Merge Storage realizando a consulta das informações persistidas referentes aos conflitos gerados a partir das integrações ou pelas demais funcionalidades implementadas junto ao componente Conflict View.

Por fim, o componente Change Observer poderá receber também mensagens diretamente do componente CVS Server que é o componente responsável pelo gerenciamento e pelas integrações dos códigos-fontes alterados junto ao repositório. Essa situação poderá acontecer quando o desenvolvedor participante do experimento comitar as alterações realizadas a partir do diretório de localização dos códigos-fontes ou pelo próprio Eclipse (ECLIPSE, 2024) através do evento pós-commit. Após o recepcionamento da mensagem pelo componente Conflict Engine, será realizado o processamento principal para avaliação e verificação da existência dos conflitos persistindo os dados resultantes da integração dos códigos-fontes alterados representados através da Figura 16.

O Algoritmo 2 irá verificar a ocorrência de conflitos à cada cinco segundos, os novos conflitos serão listados através do componente Conflict View. O intervalo de cinco segundos foi puramente arbitrado e não avaliou nenhum critério técnico para a sua respectiva definição, podendo ser totalmente auditável e configurável aos critérios de avaliação definidos pelos experimentos da pesquisa. As informações resultantes das ocorrências de conflitos serão atualizadas no Pro-Merge View com os resultados do processamento das mensagens recepcionadas inicialmente pelo componente Conflict Engine.

A partir da visualização dos conflitos através do componente Conflict View, o desenvolvedor terá disponibilizada duas opções para a resolução desses conflitos: (1) Sobrescrever – A versão local do código-fonte será sobrescrita com a versão existente no repositório. Esse processo será



Fonte: Criado pelo Autor.
Figura 16: Processamento Principal da *ProMerge*.

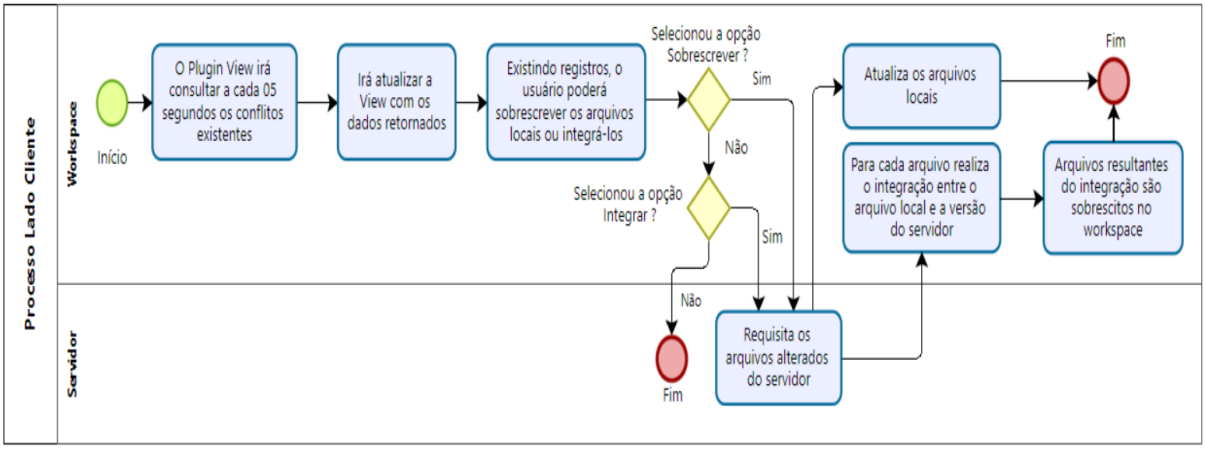
executado de forma automática, e não será solicitada a confirmação do aceite por parte do desenvolvedor e por fim, (2) Integrar – Realizara a integração dos arquivos fontes da versão existe no workspace do desenvolvedor com a versão existe no repositório. O processo de atualização será executado e não será solicitada a confirmação de aceite por parte do desenvolvedor. A Figura 17 representa o processo de resolução automática de conflitos, conforme detalhado no Algoritmo 2.

Algoritmo 2: Algoritmo de Resolução Automática de Conflitos.

Entrada: arquivoConflitante
Saída: Atualização arquivos conflitantes

```
1 listaConflitos ← BuscarListaConflitos();
2 se arquivoConflitante ← Existe então
3     arquivoRepositorio ← BuscarArquivoRepositorio();
4     atualizarArquivo ←
        MostrarDiferencasArquivos(arquivoRepositorio, arquivoConflitante);
5     se atualizarArquivo ← IsValid então
6         /*Atualizar o arquivo no workspace do usuário*/
          ee SobrescreverArquivoWorkspace(atualizarArquivo, usuario);
7     fim
8 fim
```

O Algoritmo 3 irá verificar a ocorrência de conflitos após a integração de códigos-fontes junto ao repositório, após a execução do evento pos-commit pelo componente CVS Server. O desenvolvedor poderá submeter as alterações realizadas nos códigos-fontes junto ao repositório, o Eclipse (ECLIPSE, 2024) enviará as alterações submetidas ao componente CVS Cliente, que por sua vez enviará ao componente CVS Server que realizará o processamento principal.



Fonte: Criado pelo Autor.
Figura 17: Resolução Automática de Conflitos.

O componente CVS Server após finalizar o processamento enviará uma mensagem ao componente Change Observer através do evento pós-commit. Por fim, o componente Conflict Engine irá recepcionar a mensagem recebida pelo componente Change Observer e irá realizar o processamento e as validações necessárias tentando assegurando a integridade do projeto. A Figura 18 representa o processo de validação da integridade do projeto após a integração dos códigos-fontes alterados junto ao repositório.

Por fim, as informações de contexto serão geradas e armazenadas a partir da interação dos desenvolvedores com o ambiente de desenvolvimento auxiliados pela *ProMerge*. Por exemplo, a definição do grau de severidade de um conflito será fundamentada pela pontuação atribuída a uma determinada funcionalidade e que apresentar a ocorrência de um ou mais conflitos. O committing time será obtido pela diferença de tempo entre a efetivação de um commit e o próximo commit, onde muitas informações relevantes serão armazenadas e disponibilizadas para avaliação. A geração de informações de contexto é necessária aos processos complementares da *ProMerge*, além dos recursos de detecção e resolução proativa de conflitos.

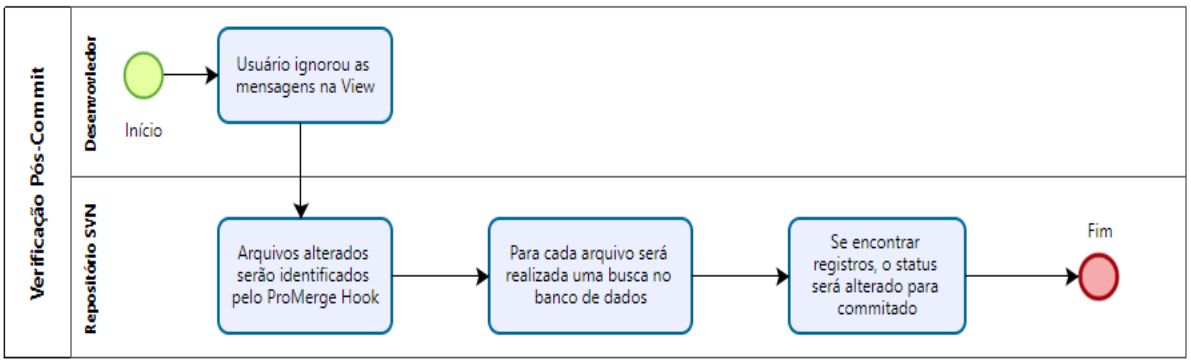
4.5 Aspectos de Implementação

Essa Seção apresenta os principais aspectos da implementação, descreve como os componentes arquiteturais dos algoritmos propostos foram desenvolvidos. No componente Storage Connector foi utilizado o adaptador JDBC para conexão com o SGDB PostgreSQL (POSTGRESQL, 2024) existente no componente Merge Storage, a finalidade é armazenar as informações de histórico de contexto. O Persistence (ProMerge Core) apresenta a regra responsável pela formatação dos dados a serem armazenados, foram adotados os seguintes padrões: Value Object (VO), Data Acess Object (DAO) e Model-View-Controller (MVC) segmentando às funcionalidades em camadas isoladas e independente, conforme detalhados na Seção 4.3. O

Algoritmo 3: Algoritmo de Checagem Pós–Commit.

Entrada: arquivosEnviados
Saída: status

```
1  listaMetodosAlterados ← VerificarMetodosAlterados(arquivosEnviados);
2  se listaMetodosAlterados ≠ Vazia então
3      /*Integrar os arquivos no repositório*/
       statusIntegracao ← IntegrarArquivosrepositrio(arquivosEnviados);
4      se statusIntegracao ← IsValid então
5          /*Informações dos arquivos, classes e métodos afetados pela alteração*/
           montarArqvoreHierarquia ←
           ArvoreHierarquiaMetodos(listaMetodosAlterados);
6          se montarArqvoreHierarquia ≠ IsEmpty então
7              /*Lista de arquivos e os workspaces afetados pelas alterações comitadas*/
               arquivosWorkspaces ←
               ConsultarWorkspaces(montarArqvoreHierarquia);
8              se arquivosWorkspaces ≠ IsEmpty então
9                  para i ← 0 até i < Quantidade(arquivosWorkspaces) faça
10                     se Conteudo(NomeArquivo(arquivosWorkspaces[i])) ≠
                        Conteudo(metodosAlterados) então
11                         AtualizarListaConflitos();
12                     fim
13                 fim
14             fim
15         fim
16     fim
17 fim
```



Fonte: Criado pelo Autor.
Figura 18: Checagem Pós–Commit.

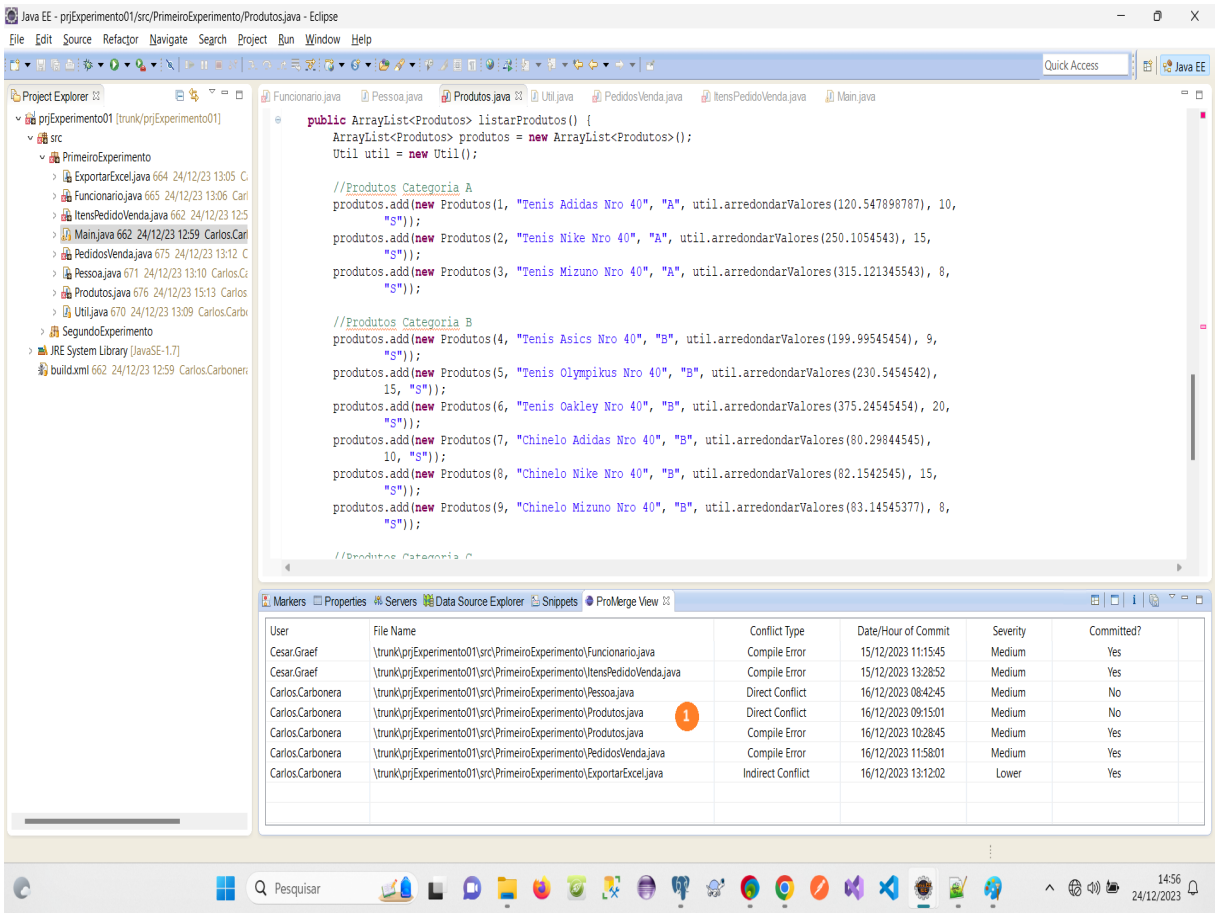
Conflict View (ProMerge) irá disponibilizar de forma intuitiva informações dos conflitos ocorridos, e informações do histórico de contexto auxiliando os desenvolvedores na resolução dos conflitos.

O componente Change Observer (ProMerge Hook) será o responsável por enviar e receber requisições entre os componentes Conflict Engine (ProMerge Server) e Conflict View (ProMerge), tendo como funcionalidade principal a intercomunicação entre componentes. O Conflict Engine (ProMerge Server) conterá toda a regra de processamento, após a submissão e efetivação das alterações nos códigos-fontes junto ao repositório, o componente Change Observer (ProMerge Hook) irá enviar uma requisição solicitando o processamento. Todas as informações referentes aos conflitos e históricos de contexto serão processadas e armazenadas no SGDB PostgreSQL (POSTGRESQL, 2024).

O componente Conflict View (ProMerge) será implementado para poder disponibilizar de forma intuitiva informações dos conflitos ocorridos, informações do histórico de contexto aos usuário e também auxiliando na resolução dos conflitos. O Change Observer (ProMerge Hook) será o responsável por enviar e receber requisições entre os componentes Conflict Engine (ProMerge Server) e Conflict View (ProMerge), tendo como funcionalidade principal a intercomunicação entre componentes. O Conflict Engine (ProMerge Server) conterá toda a regra de processamento, após a submissão e efetivação das alterações nos códigos-fontes junto ao repositório, o componente Change Observer (ProMerge Hook) irá enviar uma requisição solicitando o processamento. Todas as informações referentes aos conflitos e históricos de contexto serão processadas e armazenadas no SGDB PostgreSQL (POSTGRESQL, 2024).

O Eclipse (ECLIPSE, 2024) foi selecionado como o ambiente utilizado para o desenvolvimento com a linguagem de programação Java (JAVA, 2024). Os desenvolvedores do experimento realizarão as tarefas que envolvem alterações nos arquivos fontes e terão interatividade com os componentes CVS Client (TORTOISESVN, 2024) e CVS Server (SVNSERVER, 2024). O CVS Client (TORTOISESVN, 2024) será disponibilizado em cada uma das estações de desenvolvimento e será responsável pelo gerenciamento das alterações nos arquivos fontes que devem ser submetidas ao repositório representado através do componente CVS Server (SVNSERVER, 2024).

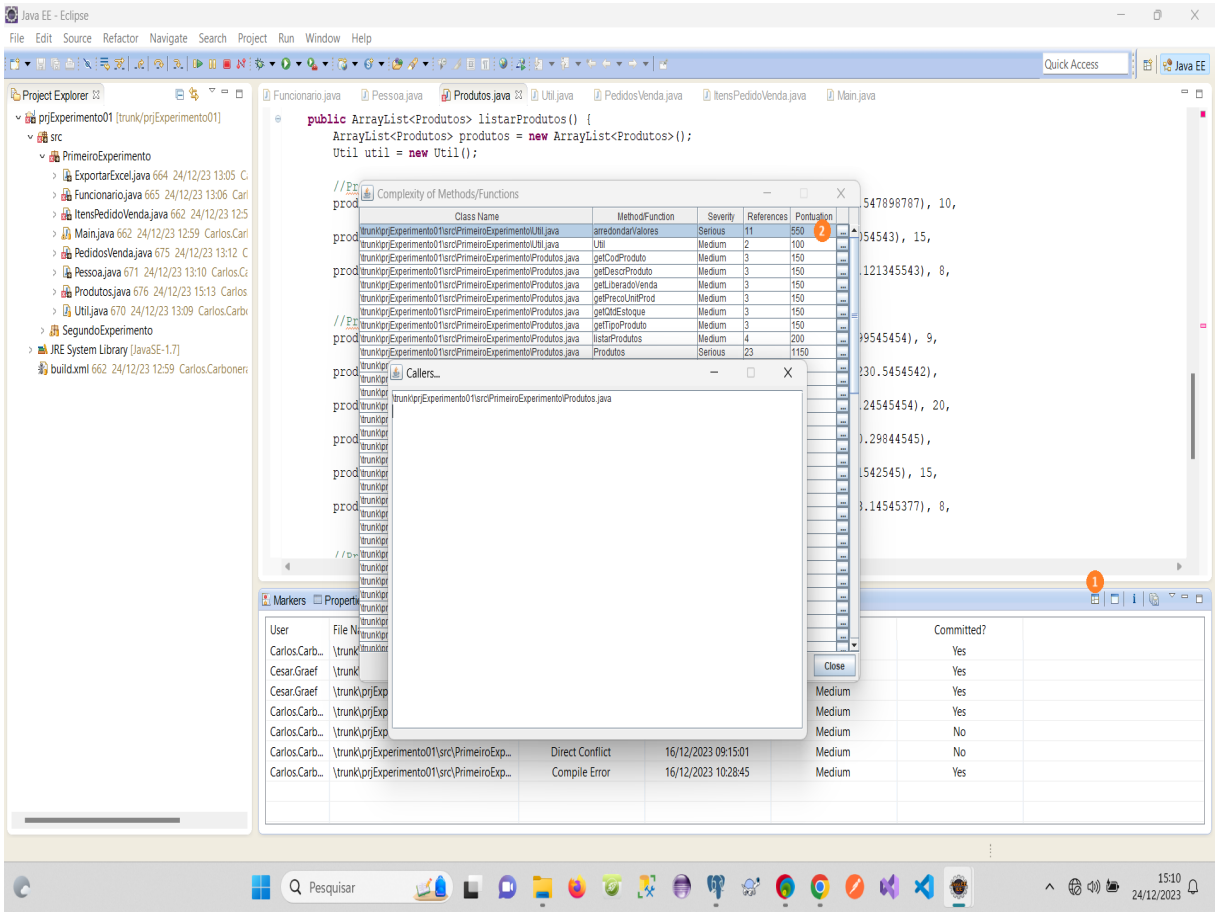
As interfaces gráficas existentes no componente *ProMerge View* estão representadas através das Figuras 19, 20 e 21. A Figura 19 detalha a listagem dos conflitos diretos ou indiretos existentes e detectados de forma proativa, ou ainda os erros de compilação detectados pela verificação pós-commit. Sendo possível identificar o usuário que realizou as alterações, bem como o caminho completo dos arquivos fontes, o grau de severidade e se os arquivos fontes foram comitados ou não no repositório. Existem ainda duas opções de ações para os usuários, onde ao clicar com o botão direito em qualquer das linhas existentes no componente *ProMerge View*: (1) Integrar fontes, onde o arquivo fonte selecionado poderá ser integrado com o arquivo fonte do repositório; e por fim (2) Sincronizar fonte, onde o arquivo fonte do workspace atual seja sobrescrito com a versão do arquivo fonte existente no repositório.



Fonte: Criado pelo Autor.

Figura 19: Componente *ProMerge View*.

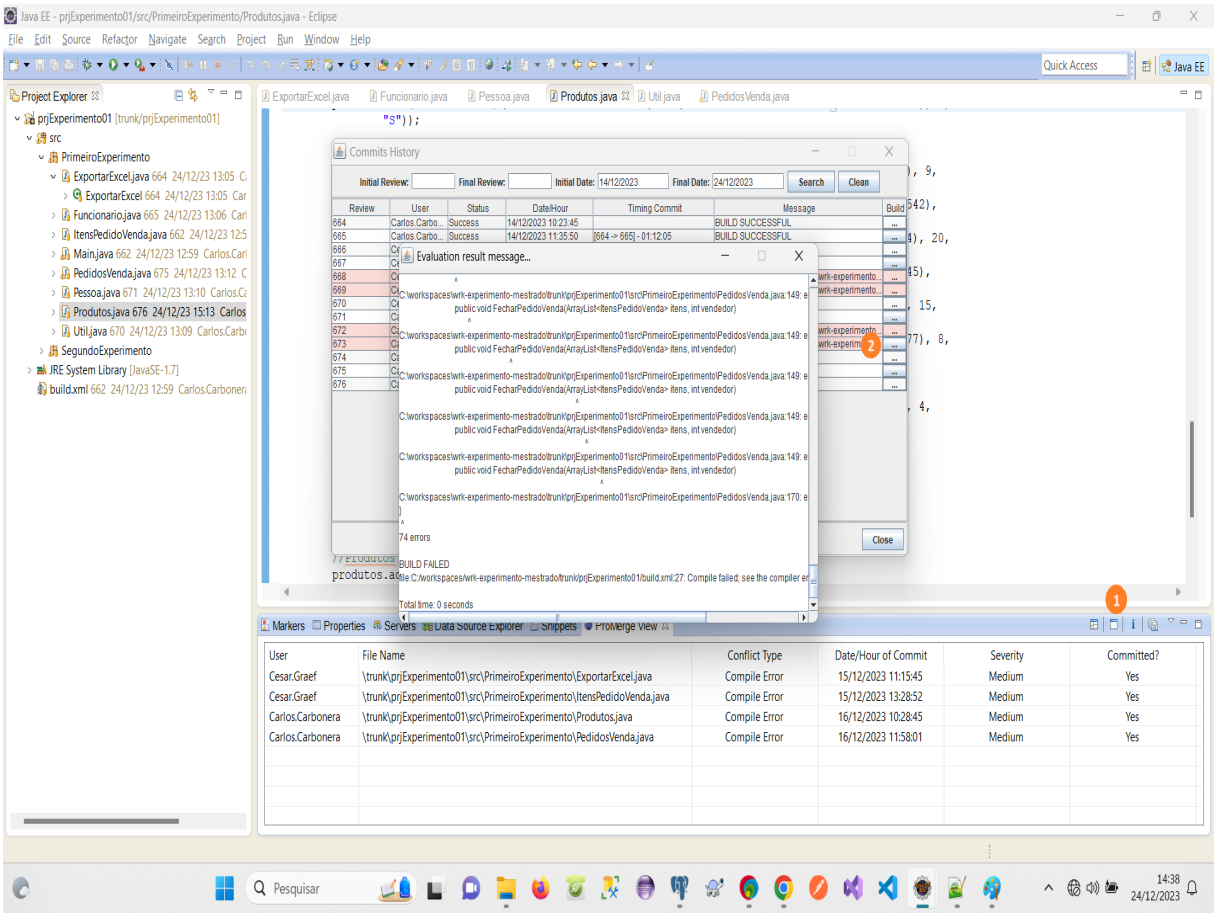
A Figura 20 detalha a lista de todas as classes que compõe o projeto, e de todas as funcionalidades ou métodos que compõem cada uma das classes, e classificando individualmente para cada funcionalidade ou método o respectivo grau de severidade detalhado na Tabela 13 e a(s) classe(s) chamadora(s) de cada uma das funcionalidades. Essas informações são cadastradas ou atualizadas através da verificação pós-commit dos arquivos fontes comitados junto ao repositório.



Fonte: Criado pelo Autor.
Figura 20: Referências dos Métodos ou Funcionalidades.

A Figura 21 apresenta outras informações de contexto também geradas durante a verificação pós-commit dos arquivos fontes alterados pelos usuários e comitados junto ao repositório. Após a realização de cada um dos commits, os arquivos fontes são compilados e a integridade do projeto é avaliada, caso existir algum erro de compilação a linha de ocorrência será destacada e a mensagem do erro resultante da compilação poderá ser visualizada para verificação e correção dos problemas. Através dessa interface gráfica é possível acompanhar a quantidade de commits com e sem erros de compilação durante um determinado intervalo de tempo. Assim, a partir dessas informações torna-se possível evidenciar indicadores de desempenho e de qualidade dos commits efetuados, por exemplo: Caso durante um intervalo de tempo forem realizados quinze commits, dos quais três commits apresentaram erros de compilação, resultando em uma taxa

de erro de 20% (3/15), o que caracteriza-se como uma taxa elevada de commits com erros em relação a quantidade de commits realizados. Portanto, a partir da análise dessas informações torna-se possível quantificar o esforço (tempo) empenhado na correção dos commits efetuados com conflitos ou erros de sintaxe.



Fonte: Criado pelo Autor.
Figura 21: Históricos de Commits por Período.

Assim, outro importante indicador de produtividade e qualidade das integrações realizadas está relacionado ao tempo de commit (Committing Time). Logo, para cada um dos commits realizados é identificado o usuário, a data e a hora e o número da revisão. A complexidade e a severidade das tarefas podem ser avaliadas e classificadas, e de acordo com o nível técnico ou a experiência profissional do desenvolvedor que realizou o(s) commit(s). Assim, pode-se definir que as tarefas foram executadas com um índice de produtividade excelente ou não. Disponibilizando ao gestor da equipe importantes informações de contexto e auxiliando na atribuição das tarefas de acordo com o melhor índice de desempenho aumentando a produtividade da equipe e dos desenvolvedores. As informações de contexto serão geradas pelo resultado do processamento realizados pelo componente Conflict Engine, serão armazenados os resultados das compilações dos códigos-fontes integrados e da complexidade dos conflitos ocorridos com o grau de severidade conforme detalhado na Tabela 13. Por fim, a definição das hierarquias

das funcionalidades contidas nos códigos-fontes e que detalham a pontuação da avaliação da severidade é necessária para definir o grau de severidade a partir da ocorrência de conflitos.

5 AVALIAÇÃO

Este Capítulo tem como objetivo apresentar os estudos experimentais realizados para avaliar a abordagem *ProMerge*. Sendo respondida a questão de pesquisa a QP-3 conforme detalhada na Seção 1.3. A geração de conhecimento empírico sobre avaliação de esforço (tempo), taxa de corretude e erro, grau de severidade dos conflitos gerados a partir da integração de trechos de códigos-fontes e do conceito de committing time no que refere-se ao intervalo de tempo entre as sequências de commits realizados, foram seguidas as recomendações muito bem estabelecidas em (CARBONERA; FARIAS; BISCHOFF, 2020; BISCHOFF et al., 2019; WOHLIN et al., 2012; FARIAS; GARCIA; LUCENA, 2014; FARIAS et al., 2015). Assim, vários cuidados foram necessários para mitigar qualquer tipo de anomalia nos resultados. Primeiramente, a seleção cuidadosa dos participantes do experimento para assegurar resultados o mais homogêneos possíveis.

Assim, para avaliação da *ProMerge*, foram definidos dois ambientes distintos para a realização dos experimentos, cada qual composto por cinco atividades semelhantes e com igual nível de complexidade. Envolvendo a implementação de alterações nos arquivos de códigos-fonte para mensurar os aspectos técnicos da *ProMerge*. No primeiro ambiente, o participante teve o auxílio da proposta de resolução proativa de conflitos. Porém, no segundo ambiente o participante utilizou a abordagem tradicional, sem nenhum tipo de apoio técnico para a resolução dos conflitos exceto os recursos disponibilizados pelo próprio Eclipse (ECLIPSE, 2024). A Seção 5.1 detalha o desenho do experimento controlado realizado. A Seção 5.2 detalha as hipóteses nulas e Alternativa investigadas. A Seção 5.3 detalha as variáveis dependentes e independentes analisadas por essa pesquisa. A Seção 5.4 detalha a seleção dos participantes dos experimentos controlados realizados. A Seção 5.5 apresenta o *ProMerge* aos participantes do experimento e detalha as suas funcionalidades. Portanto, foi realizado um treinamento de utilização da *ProMerge* com todos os participantes individualmente. A Seção 5.6 detalha as atividades de implementação que deve ser realizadas pelos participantes. A Seção 5.7 avaliou os resultados obtidos com a execução dos experimentos. E por fim, a Seção 5.8 discute as limitações e as ameaças à validade dos resultados encontrados.

5.1 Desenho Experimental

Essa Seção detalha a execução do experimento controlado realizado por essa pesquisa e investiga como os objetivos, hipóteses e as variáveis foram utilizadas na avaliação dos resultados obtidos. A finalidade da execução dos experimentos controlados é avaliar as vantagens e desvantagens obtidas com a utilização da *ProMerge*. Foram analisados os efeitos da utilização da *ProMerge* sobre os esforço (tempo) empenhados para a resolução de conflitos. As demais seções estão assim segmentadas: (1) A Seção 1.4 apresenta os objetivos relacionados aos experimentos controlados aplicando o formato muito bem definido do GQM (CALDIERA;

ROMBACH, 1994). (2) A Seção 5.2 detalha as hipóteses com a finalidade de alcançar os objetivos estabelecidos. Por fim, (3) A Seção 5.3 detalha as variáveis dependentes e independentes envolvidas diretamente na execução do experimentos controlado.

Analisar *abordagens de integração de códigos-fontes*
com o objetivo *de investigar seus efeitos*
com relação *a esforço (tempo), corretude e taxa de erro*
do ponto de vista *dos desenvolvedores de software*
no contexto *de evolução de códigos-fontes*

Os experimentos propostos avaliaram os efeitos da compreensão e resolução dos conflitos em arquivos de arquivos de códigos-fonte de um experimento controlado. A análise dos resultados do experimento é dividido em dois grupos principais: avalição abordagem e qualitativa. Na primeira etapa foram aplicadas cinco atividades de implementação de código-fonte com diferentes níveis de complexidade. E ao final foi aplicado um questionário qualitativo (apêndice C) composto por dezesseis perguntas para avaliar a impressão dos participantes quanto a utilização da *ProMerge*. O grau de conhecimento, e a experiência técnica dos participantes do experimento foi considerado quanto aos resultados obtidos.

5.2 Hipóteses Formuladas

Foram avaliadas as abordagens propostas que auxiliaram na detecção e resolução proativa de conflitos gerados a partir da integração de arquivos de códigos-fonte no experimento controlado proposto. Assim, a partir dos objetivos apresentados na Seção 1.4, três hipóteses foram formuladas para avaliação do modelo *ProMerge* conforme apresentado no Capítulo 4. A Hipótese 1 analisa a detecção antecipada de conflitos para avaliar o esforço (tempo) empenhado na resolução dos conflitos com ou sem o apoio da *ProMerge*. A Hipótese 2 analisa a Taxa de Corretude dos conflitos detectados e corrigidos pelos desenvolvedores participantes do experimento. Por fim, a Hipótese 3 avalia a Taxa de Erro após a execução das tarefas propostas nos experimentos, analisando os resultados obtidos da *ProMerge* com os resultados da abordagem tradicional.

Hipótese 1: Esforço (Tempo) Foi considerado o intervalo de tempo entre as integrações de arquivos de códigos-fonte conflitantes realizadas junto ao repositório e o esforço (tempo) das correções assegurando a integridade do projeto. O intervalo de tempo será medido em minutos (min). Essa medida de tempo foi definida de maneira técnica, não sendo fundamentada em nenhuma pesquisa ou publicação em que foi utilizada essa escala de tempo ou qualquer outro tipo de experimento empírico. Podendo ser totalmente auditável e adaptada de acordo com os critérios avaliativos considerados durante a execução dos experimentos facilitando a análise dos indicadores. No entanto, não é de forma alguma óbvia que esta hipótese seja válida, pois na existência de repetidas ocorrências dessa hipótese e multiplicada pela quantidade de

desenvolvedores envolvidos, o esforço (tempo) final poderá ser considerado como muito mais elevado que para os padrões de desenvolvimento em equipe. Portanto, a Hipótese 1 avaliou se a *ProMerge* apresenta reduções significativas do esforço (tempo) em comparação às abordagens baseadas em heurísticas. Assim, foi detalhada a hipótese nula e a alternativa da seguinte forma:

Hipótese Nula 1, $H_1{-}_0$**:** A utilização da *ProMerge* não apresenta reduções significativas de esforço (tempo) na resolução de conflitos.

$$H_1{-}_0\textbf{:}\ Esforco(Tempo)_{ProMerge} \geq Esforco(Tempo)_{Traidicional}$$

Hipótese Alternativa 1, $H_1{-}_1$**:** Verifica-se uma redução significativa de esforço (tempo) na resolução de conflitos com a utilização da *ProMerge*.

$$H_1{-}_1\textbf{:}\ Esforco(Tempo)_{ProMerge} < Esforco(Tempo)_{Traidicional}$$

Hipótese 2: Taxa de Corretude. Avaliou a utilização da técnica de programação em pares (pair programming) para a resolução de conflitos. Historicamente essa abordagem demonstrou-se ser uma alternativa robusta e que aumenta sensivelmente o nível de assertividade nas correções. As partes interessadas podem não apresentar o conhecimento técnico necessário (individual ou coletivamente) para a correta avaliação do grau de impacto das alterações realizadas. Isso é devido em partes por não ser possível identificar todos os cenários de testes necessários para as devidas correções nos arquivos de códigos-fontes, mas comprovadamente a análise realizada em pares comprovou ser mais eficaz em relação a individual. Portanto, o grau de corretude analisará a assertividade das integrações dos arquivos de códigos-fonte realizadas. E como critério avaliativo será considerado a quantidade de ocorrências de problemas técnicos ocorridos multiplicado pelo esforço (tempo) empenhado nas resoluções desses problemas.

Embora as heurísticas possam compor indicadores com elevado grau de acurácia, não está claro se de fato existe uma abordagem objetiva que reduzirá o tempo para correção dos conflitos pelos desenvolvedores. A maioria desses problemas somente é possível de serem detectados durante a sua execução ou na avaliação dos resultados obtidos a partir das funcionalidades alteradas, pois nenhuma ocorrência de conflito foi detectada durante a integração dos arquivos de códigos-fonte alterados. Portanto, foram considerados os intervalos de tempos em minutos transcorridos entre a detecção dos problemas e a sua resolução. Podendo ser totalmente auditáveis e possíveis de adaptação conforme os critérios avaliativos considerados durante a execução dos experimentos e multiplicadas pela quantidades de desenvolvedores envolvidos. O esforço (tempo) final poderá ser elevado para os padrões de desenvolvimento em equipes, característica já definida anteriormente. Portanto, a Hipótese 2 avaliará se o grau de corretude da *ProMerge* apresentará reduções significativas em comparação à não utilização de abordagens baseadas em heurísticas. Por fim, foi detalhada a hipótese nula e a alternativa da seguinte forma:

Hipótese Nula 2, $H_2{-}_0$**:** A utilização da *ProMerge* não apresenta reduções significativas da taxa de corretude na resolução de conflitos.

$$H_2{-}_0\textbf{:}\ Corretude_{ProMerge} \geq Corretude_{Traidicional}$$

Hipótese Alternativa 2, $H_2{-}_1$**:** Verifica-se uma redução significativa da taxa de corretude na resolução de conflitos com a utilização da *ProMerge*.

$$H_2{-}_1\textbf{:}\ Corretude_{ProMerge} < Corretude_{Traidicional}$$

Hipótese 3: Taxa de Erro. A utilização das abordagens avaliativas propostas pelo *ProMerge* forneceu as taxas de erros para cada um dos desenvolvedores participantes do experimento controlado quanto a realização das tarefas. A finalidade das abordagens propostas é facilitar a análise e o entendimento da complexidade dos arquivos de códigos-fonte. Obviamente essa hipótese não poderá ser considerada como válida imediatamente. Embora as de avaliação da taxas de erro baseadas na análise da complexidade forneçam uma possibilidade mais sistemática de compor esse tipo de índice. Podendo oferecer algumas vantagens para avaliar as funcionalidades em cenários de evolução complexos.

Contudo, a sua utilização não é perceptível o suficiente para reduzir as taxas de erro mais rápida e precisamente. pois, foi considerado os resultados da subtração de um menos a Taxa de Corretude encontrada e conforme detalhada na Tabela 16. Portanto, a hipótese 3 avaliará se as taxas de erros produzidas pela da *ProMerge* apresentará índices de redução significativos em relação a abordagem tradicional. A hipótese nula e a alternativa estão detalhadas da seguinte forma:

Hipótese Nula 3, $H_3{-}_0$**:** A utilização da *ProMerge* não apresenta reduções significativas da taxa de erro na resolução de conflitos.

$$H_3{-}_0\textbf{:}\ TaxaErro_{ProMerge} \geq TaxaErro_{Traidicional}$$

Hipótese Alternativa 3, $H_3{-}_1$**:** Verifica-se uma redução significativa da taxa de erro na resolução de conflitos com a utilização da *ProMerge*

$$H_3{-}_1\textbf{:}\ TaxaErro_{ProMerge} < TaxaErro_{Traidicional}$$

Portanto, a hipótese H1 avaliou se a abordagem baseada em tempo implica (ou não) em uma taxa menor de esforço (tempo) em relação ao uso de abordagens baseadas em heurísticas. A hipótese H2 produziu conhecimento empírico sobre o grau de corretude das alterações realizadas de arquivos de códigos-fonte principalmente quanto ao nível de inconsistência. Por fim, a hipótese H3 produziu conhecimento empírico relacionado a Taxa de Erro na execução das tarefas com o auxílio da *ProMerge* em relação à abordagem tradicional, também em relação ao uso de abordagens baseadas em heurísticas. Portanto, identificando a abordagem mais propensa a erros, os desenvolvedores podem optar por qual é a mais adequada para a sua necessidade de resposta. A Tabela 15 lista todas as hipóteses investigadas por essa pesquisa.

Essa Seção apresentou uma visão generalista das três hipóteses nulas e alternativas propostas para os dois cenários dos experimento controlado, fornecendo o suporte adequado para resolução de todas às três questões de pesquisas detalhadas na Seção 1.3. As três hipóteses avaliaram

Hipóteses Nulas	Hipóteses Alternativas
$H_1{-}_0$: $Esforco(T)_{Pr} \geq Esforco(T)_{Trd}$	$H_1{-}_1$: $Esforco(T)_{Pr} < Esforco(T)_{Trd}$
$H_2{-}_0$: $Corretude_{Pr} \geq Corretude_{Trd}$	$H_2{-}_1$: $Corretude_{Pr} < Corretude_{Trd}$
$H_3{-}_0$: $TaxaErro_{Pr} \geq TaxaErro_{Trd}$	$H_3{-}_1$: $TaxaErro_{Pr} < TaxaErro_{Trd}$

Legenda: *(Pr)* ProMerge, *(T)* Tempo, *(Trd)* Tradicional

Tabela 15: Hipóteses Nulas e Alternativas Testadas.

as abordagens proposta pelo modelo *ProMerge* de detecção e resolução proativa de conflitos com a utilização de informação de contexto, sendo esse o principal objetivo investigado por essa pesquisa.

5.3 Variáveis de Estudo

Essa Seção discute as variáveis dependentes e independentes envolvidas na execução dos experimentos controlados propostos por essa pesquisa. Identificando às variáveis que representam níveis crescentes de complexidade no que refere-se à conflitos gerados a partir da integração de arquivos fontes. A análise dessas variáveis busca mensurar o nível de esforço (tempo) para a integração dos arquivos de códigos-fonte alterados e ao mesmo tempo compreender os conflitos gerados pelas integrações. A eficácia e a relevância do conhecimento prévio dessas variáveis poderá elevar os níveis de compreensão dos desenvolvedores envolvidos com a realização dos experimentos.

As variáveis independentes é representada através da técnica aplicada, sendo representada pela *ProMerge* e a abordagem tradicional. Assim, para cada uma das tarefas proposta em um determinado cenário haverá uma tarefa correspondente no cenário oposto, e será semelhante em estrutura e complexidade. Assim, em um dos cenários será utilizada a *ProMerge* e no outro cenário apenas a tradicional. Os impactos resultantes da realização dos experimentos controlados foram investigados exclusivamente através das variáveis dependentes, conforme detalhadas na Tabela 16.

Os experimentos controlados são compostos de três variáveis dependentes: Esforço (Tempo) (ET), Corretude (CO) e Taxa de Erro (TE). (1) O ET avalia o tempo de resolução das tarefas em minutos propostas como parte da execução ou dos conflitos gerados após a integração dos arquivos fontes. (2) A CO avalia a correta execução das atividades pelos desenvolvedores nos experimentos controlados. Foram analisada as vantagens e desvantagens da utilização da *ProMerge*. A CO é o resultado da quantidade de acertos pela quantidade total de tarefas executadas corretamente, sendo representada através da fórmula: *CO = Total Acertos / Total Respostas*. (3) a TE avalia o resultado da diferença de um menos a CO encontrada, representada através da fórmula: *TE = 1 - CO*. Por fim, quanto as variáveis independentes, será investigada a *ProMerge*

em relação a abordagem tradicional, todas as demais variáveis independentes serão desconsideradas por essa pesquisa. A Tabela 16 detalha todas as variáveis envolvidas e analisadas com o objetivo de validar a *ProMerge*.

Tipo	**Variável**	**Escala**
Dependentes	Esforço (Tempo) (ET) Taxa de Corretude (CO) Taxa de Erro (TE)	Intervalo [0..n] Intervalo [0..1] Intervalo [0..1]
Independentes	Técnica de Integração	Nominal: *ProMerge* Tradicional

Legenda: *(n)* Tempo em minutos

Tabela 16: Variáveis Dependentes e Independentes.

5.4 Seleção dos Participantes

Inicialmente foram selecionados trinta e dois participantes de uma amostra de cinquenta e sete possíveis candidatos para a execução dos experimentos controlados propostos pelo *ProMerge*. A quantidade inicial de candidatos foi disponibilizada em partes pelo próprio PPGCA – Programa de Pós–Graduação em Computação Aplicada da Unisinos – Universidade do Vale dos Sinos e pela aproximação profissional a partir da rede de contato profissionais Linkedin (`https://www.linkedin.com/`). Logo, todos os participantes são alunos(as) ou ex–alunos(as) de instituições de ensino superior como: UFRGS – Universidade Federal do Rio Grande do Sul, Unisinos – Universidade do Vale dos Sinos, UCS – Universidade de Caxias do Sul, Feevale – Federação dos Estabelecimentos de Ensino Superior em Novo Hamburgo e PUCRS – Pontifícia Universidade Católica do Rio Grande do Sul. Para que um candidato(a) estivesse apto(a) à participar dos experimentos foram considerados aspectos como a experiência mínima de um ano com desenvolvimento de software (preferencialmente com a linguagem Java ou afins). E familiaridade com alguma ferramenta de versionamento de arquivos fontes existentes na atualidade. O nível de experiência abordagem foi um dos principais requisitos considerados durante a seleção dos participantes, pois em havendo dois ou mais candidatos(as) com qualificações muito semelhantes, o(a) candidato(a) com maior nível de experiência abordagem era o(a) selecionado(a) para participar do experimento.

Para ambientação dos participantes quanto aos experimentos, foi realizada uma demonstração inicial quanto à utilização das funcionalidades propostas pelo *ProMerge*. Após um prévio treinamento com cada um dos participante do experimento antes da execução das atividades, com a finalidade de ambientação e contextualização das funcionalidades existentes. Por fim, todos os experimentos foram realizados individualmente, gravados com autorização do participantes e os arquivos fonte resultantes foram separados assegurando a validade da análise dos

resultados obtidos.

5.5 Processo Experimental

Para a condução do experimento foi estabelecido um processo muito bem definido e dividido em três etapas, cada uma das etapas foi definida para não interferir na realização das demais. Os participantes foram instruídos sobre o processo do experimento para assegurar que estivessem aptos e adquirido o entendimento necessário para a execução do experimento. A Figura 23 apresenta o processo que foi elaborado para realizar o experimento controlado. Todo o processo experimental é dividido em três etapas principais. A Etapa 1 busca extrair características por meio de um questionário aplicado aos participantes, conforme detalhado no apêndice B. Esse questionário detalha as informações coletadas dos participantes antes do experimento. A Etapa 2 consistiu em explicar individualmente aos participantes a *ProMerge* para em seguida iniciar a execução e gravação das tarefas dos experimentos, bem como coletar as informações de cada uma delas nos dois cenários propostos. Por fim, a Etapa 3 teve como objetivo capturar a percepção técnica dos participantes sobre o experimento executado através de um questionário detalhado no apêndice C, realizado após a finalização das tarefas dos experimentos para isso foi aplicado um questionário TAM (Modelo de aceitação da tecnologia). As três etapas principais são detalhadas abaixo:

Etapa 1: Treinamento. Na etapa de treinamento, os participantes selecionados para os experimentos (Seção 5.4) foram instruídos quanto a utilização da *ProMerge* assegurando o pleno entendimento do processo e das atividades propostas. Sendo feitas demonstrações da utilização da *ProMerge* e realizada a contextualização do ambiente dos experimentos com os participantes envolvidos. Após responder o questionário de características (Apêndice B), o ambiente de execução foi disponibilizado aos participantes de forma individual e as instruções necessárias foram detalhadas bem como a apresentação das atividades que deveriam ser executadas.

Etapa 2: Execução das tarefas experimentais. As atividades nos dois cenários propostos pelos experimentos são semelhantes, com iguais níveis de complexidade, contudo não se repetem. Torna-se importante ressaltar que os critérios técnicos avaliados foram os mesmos nos dois cenários, mitigando assim a criação de qualquer problema nos participantes. No primeiro cenário, as atividades foram realizadas tendo o auxílio da *ProMerge* para a detecção proativa de conflitos e das demais funcionalidades desenvolvidas. O segundo cenário utilizou a abordagem tradicional para a detecção de conflitos, ou seja as funcionalidades existentes no Eclipse (ECLIPSE, 2024).

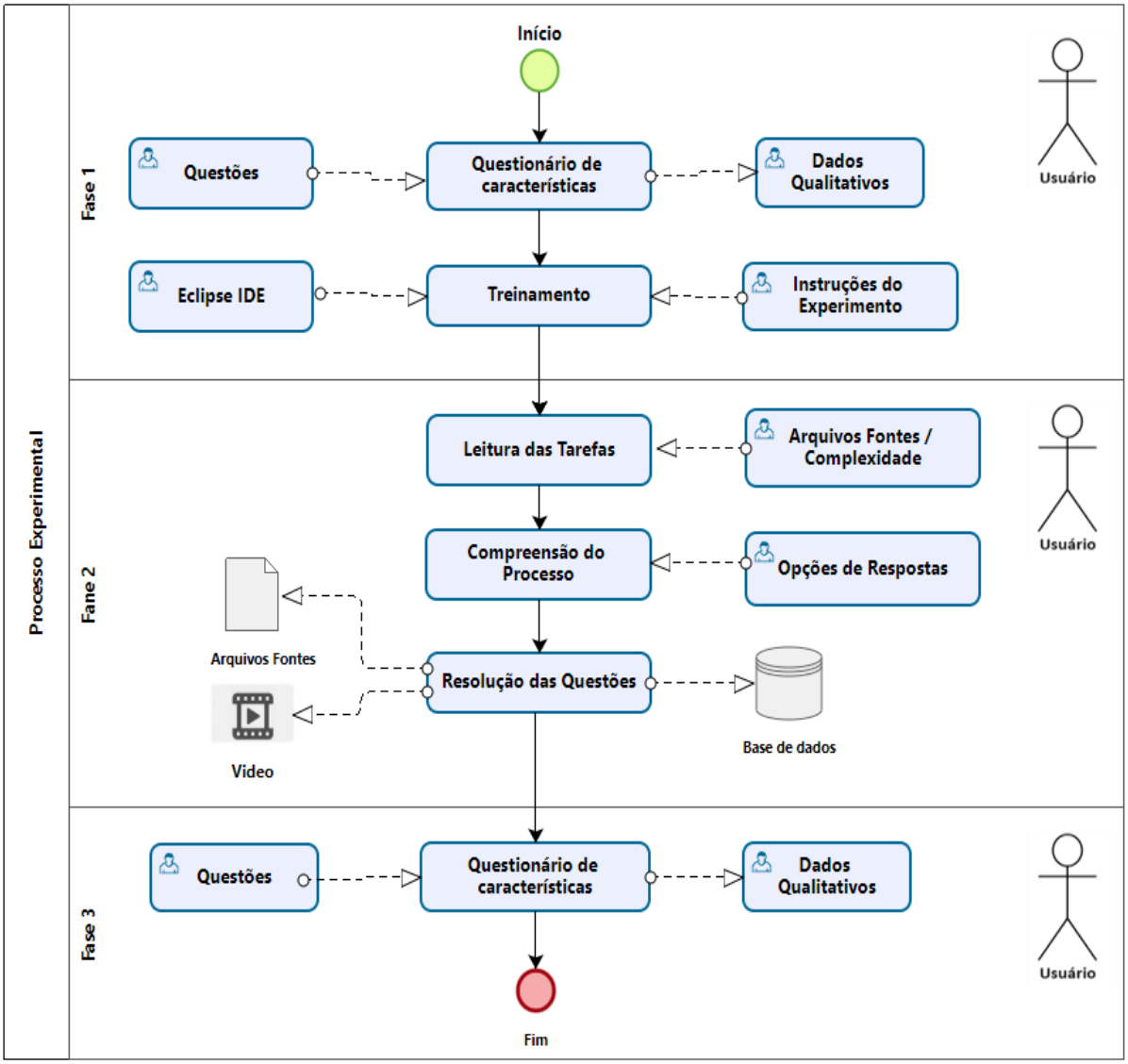
Os arquivos fontes disponibilizados nos experimento foram divididos em dois pacotes, com a finalidade de separação dos cenários em: (1) Experimentos realizados com apoio da *ProMerge* e, (2) Realizados com o apoio da abordagem tradicional. Sendo assim segmentados para não permitir a criação de qualquer problema entre os desenvolvedores. Outro aspecto importante a ser considerado, é que se um desenvolvedor iniciou a execução dos experimentos utilizando o

primeiro cenário o desenvolvedor seguinte iniciou as atividades utilizando o segundo cenário que utiliza a abordagem tradicional para a detecção de conflitos. Por fim, somente na próxima etapa do experimento (primeiro cenário) o desenvolvedor utilizará o apoio proativo da *ProMerge*.

Etapa 3: Análise e avaliação dos resultados. Os participantes responderam à dois questionários. O primeiro foi realizado antes da execução dos experimentos e o segundo após à conclusão dos experimentos. O primeiro deles, foi o questionário de características, que encontra-se detalhado no Apêndice B é comporto por questões que buscam informações do perfil dos participantes, dentre elas: experiência profissional, formação acadêmica, idade, cargo ocupado atualmente, tempo de experiência entre outras. O segundo, o questionário de impressões é detalhado no Apêndice C, e apresenta dezesseis questões que buscaram analisar às percepções dos participantes quanto aos experimentos executados e da aceitação da *ProMerge*. Por fim, a descrição do processo executado está detalhado na Figura 22.

Assim, cada uma das cinco atividades do experimento era composta uma ou mais tarefas utilizando a ferramenta Eclipse (ECLIPSE, 2024) conjuntamente com o Tortoise SVN (TORTOISESVN, 2024) realizando a integração dos trechos de códigos-fontes alterados. Todas as tarefas apresentaram diferentes níveis de complexidade. Os participantes envolvidos executaram todas as cinco atividades nos dois cenários propostos sem nenhum tipo de interrupção durante a realização das tarefas. O formato utilizado para avaliação das atividades consistiu na pontuação obtida por cada participante. Assim, cada uma das cinco atividades apresentava uma quantidade definida de respostas corretas. Logo, foi possível estabelecer um indicador de erros e acertos para cada questão, por exemplo: Uma determinada atividade tinha como respostas corretas cinco itens e se um dos desenvolvedores acertou apenas quatro de cinco itens possíveis, o resultando do índice de corretude será de 75% para a questão desenvolvida. Esse índice foi obtido a partir da divisão da quantidade de acertos pela quantidade de respostas corretas possíveis, como por exemplo: 04 / 05 = 0,75, resultando em 75% de corretude.

As atividades foram definidas com a finalidade de avaliar a performance e a qualidade do desenvolvimento de cada um dos participantes envolvidos. Envolveram o desenvolvimento de tarefas com conflitos diretos, indiretos, sintáticos e semânticos. Os Apêndices D e E descrevem as atividades e os cenários utilizados diretamente no experimento controlado por essa pesquisa. Após a finalização dos experimentos, todos os arquivos fontes resultantes das execuções das tarefas propostas foram identificados e catalogados (por participante). Sendo posteriormente comparados com o modelo de respostas padrão definido como ideal para a definição do grau de erro e acerto das questões. Os participantes realizaram todas as atividades de forma individual e sem nenhuma interrupção externa, para evitar que qualquer problema. Cada participante teve o ambiente configurado com os arquivos fontes inicialmente propostos para os experimentos evitando qualquer situação inesperada durante a execução dos experimentos, assegurando a qualidade dos resultados.



Fonte: Criado pelo Autor.
Figura 22: Definição do Processo Experimental.

5.6 Tarefas Experimentais

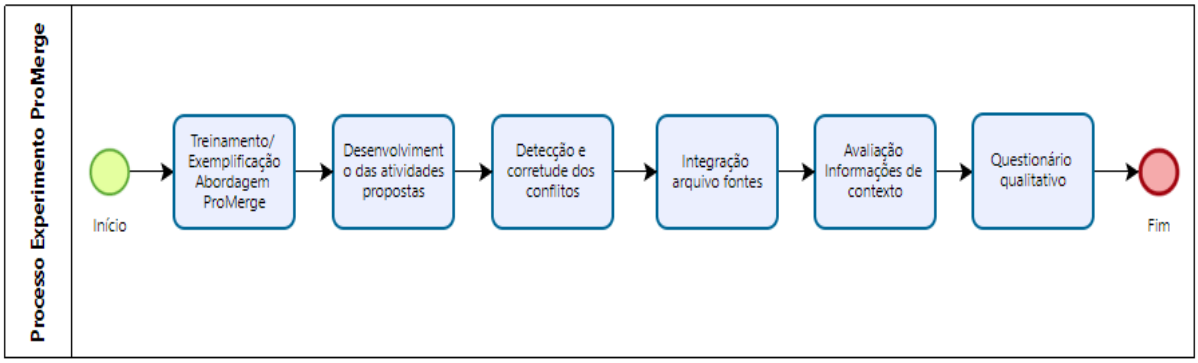
Foram definidas quatro funcionalidades que seriam avaliadas: (1) avaliação do grau de severidade dos conflitos(Seção 4.1), (2) avaliação do tempo de corretude, (3) avaliação da hierarquia dos métodos dentro do contexto e por fim, (4) conflitos diretos e indiretos. Dentre elas, pode-se afirmar que a funcionalidade (4) é a que tem a maior probabilidade de ocorrer na prática. As funcionalidades (1, 2 e 3) foram implementadas no modelo proposto pelo *ProMerge* e não foram detectados estudos publicados que abordassem essas funcionalidades ou correlatas para comparação de resultados ou avaliação de experimentos.

O projeto contendo os arquivos fontes utilizado para a execução do experimento de validação da *ProMerge* foi especificamente criado. Consistindo em um sistema muito simples e composto por nove classes em cada pacote de arquivos fontes e com algumas funcionalidades inicialmente definidas. A maioria das funcionalidades propostas serão implementadas pelos participantes em atendimento as atividade do experimento controlado. Assim, a definição de uma projeto simples utilizando a linguagem Java e com o Eclipse (ECLIPSE, 2024) foi com a finalidade de facilitar o desenvolvimento das tarefas em um curto espaço de tempo. Segundo (WOHLIN et al., 2012; FARIAS et al., 2015), o tamanho de um artefato de software utilizado em um experimento controlado poderá influenciar diretamente nos resultados obtidos.

As atividades que compõem os experimentos dos dois cenários de avaliação dessa pesquisa estão detalhadas nos Apêndices D e E. Quando um participante realizava o experimento com o apoio da *ProMerge*, o participantes seguinte iniciava o experimento utilizando a abordagem tradicional. Essa abordagem de execução dos experimento foi necessária para mitigar qualquer possibilidade de problema nos resultados. Todas as atividades propostas nos dois experimentos foram inicialmente implementadas para obter os arquivos fontes definidos como resultados ótimos. Sendo o modelos de resultados e comparando com os resultados das implementações realizadas pelos participantes. Fornecendo uma avaliação precisa e contribuindo com a obtenção de indicadores e taxas de erros ou acertos, conforme definidos na Seção 5.4.

Por fim, nenhum limite de tempo foi definido para a execução das tarefa dos experimentos, os participantes foram orientados a anotar o horário de início e de conclusão das atividades, e uma vez iniciadas as tarefas as mesmas teriam que ser concluídas sem interrupções, com a finalidade de mitigar qualquer problema nos resultados. Após a conclusão de todas as tarefas por todos os trinta e dois participantes, os resultados das variáveis dependentes Taxa de Corretude e esforço (tempo) e informações de contexto foram avaliadas, iniciando o processo de análise dos dados obtidos, conforme detalhado na Figura 23.

Por fim, torna-se importante ressaltar que o questionário de características e impressão conforme detalhados nos Apêndices B e C disponibilizou a qualificação e a percepção dos participantes do experimentos quanto a utilização da *ProMerge* em relação a abordagem tradicional, e não terá relevância alguma quanto a análise e avaliação dos resultados obtidos pela execução das tarefas dos experimentos.



Fonte: Criado pelo Autor.
Figura 23: Modelo de Avaliação dos Experimentos Controlados da *ProMerge*.

5.7 Resultados

Esta seção apresenta os resultados obtidos a partir da execução de um experimento controlado em dois cenários e das principais características do perfil da amostra pesquisada. Para a definição do desenho da avaliação dos experimentos, foram escolhidos cenários que exploraram diferentes aspectos durante a execução das atividades que envolveram principalmente à alteração de código-fonte. O avaliação dos dados foi feita para as variáveis dependentes: ET, CO e por fim TE conforme detalhados na Tabela 16. Os resultados alcançaram os objetivos especificados na Seção 1.4 dessa pesquisa, sendo avaliados pelas ferramentas RStudio1[1] e Winks SDA[2], utilizadas para a geração dos gráficos e tabelas comparativas, disponibilizando os dados para a parametrização dos testes e avaliação dos resultados obtidos.

5.7.1 Procedimentos de Análises

Essa Seção descreve os principais procedimentos de análise dos resultados adotados por essa pesquisa. O teste de Wilcoxon caracteriza-se por ser um teste de hipóteses que analisa a diferença existente entre duas amostras pareadas. Sendo muito utilizado quando existem duas medidas de uma mesma amostra. Assim, os participantes de um experimento controlado são avaliados em duas condições distintas, sendo uma das principais técnicas para a avaliação de amostras pareadas. Portanto, o teste de Wilcoxon converte os resultados obtidos em classificações comparando os resultados de dois intervalos de tempo, sendo aplicado nessa pesquisa para as hipóteses H1 e H3. O t-test pareado investiga se as médias de duas medidas de resultados relacionadas são classificadas como estatisticamente diferentes. A hipótese de nulidade de um t-test em amostras pareadas caracteriza-se pela média das diferenças entre as medidas relacionadas seja igual a zero, portanto não existe diferença alguma entre as medidas.

[1] `https://www.rstudio.com/`
[2] `https://www.texasoft.com/download3.htm`

O modelo de avaliação de McNemar foi selecionado para determinar se as proporções pareadas dos resultados são diferentes, é realizado em experimentos onde são comparadas duas abordagens diferentes executando os mesmos experimentos. Esse teste foi realizado para avaliar os resultados obtidos entre a utilização da *ProMerge* em relação a tradicional, sendo também aplicado nessa pesquisa para as hipóteses H1 e H3. Foram utilizadas técnicas não paramétricas e de análise descritiva para avaliação dos resultados, as técnicas não paramétricas são utilizadas para analisar dados correspondentes (pareados) ou medidas repetidas em escalas nominais (categóricas) e ordinais (classificadas), principalmente quando existirem amostras pequenas de dados não podendo serem avaliados por técnicas paramétricas.

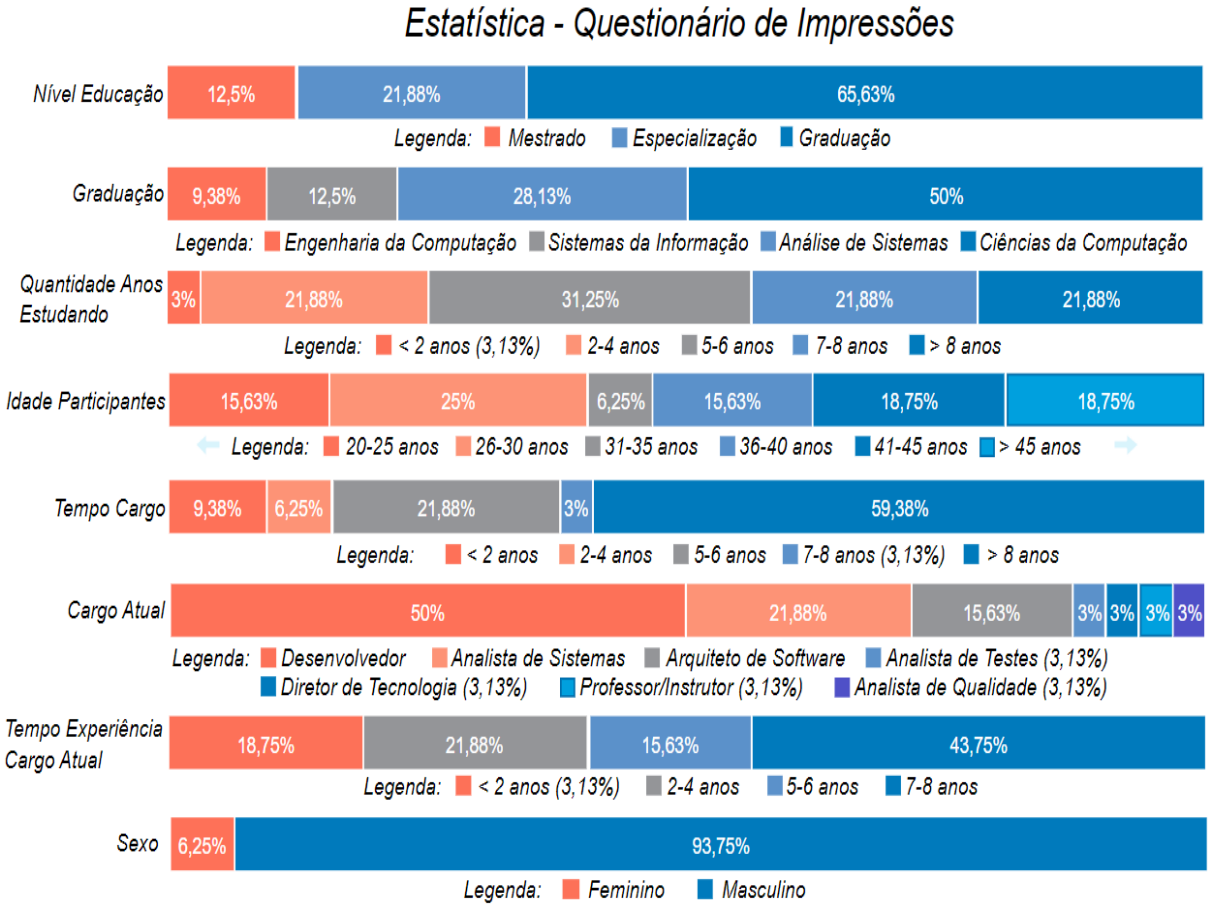
Assim, o desvio padrão é uma medida que expressa o grau de dispersão de um conjunto de dados. Ou seja, o desvio padrão indica o quanto um conjunto de dados é uniforme. Quanto mais próximo de 0 for o desvio padrão, mais homogêneo são os dados. Ela indica qual é o valor que está exatamente no meio de um conjunto de dados, quando eles estão ordenados. A Mediana indica que a metade (50%) dos valores presentes em um conjunto de valores encontra-se abaixo ou acima dela, sendo uma medida de tendência central. A média é definida como o valor que demonstra a concentração dos dados de uma distribuição, sendo o ponto de equilíbrio das frequências dos resultados em um histograma, podendo ser interpretada como um valor significativo de uma sequência de resultados. Portanto, a média pode ser definida através da combinação dos valores de maneira específica com a geração de um valor significativo.

5.7.2 Análise do Perfil dos Participantes

Após o término da execução das tarefas do experimento controlado da *ProMerge* (apêndices D e E), foi realizado a aplicação do modelo de aceitação da tecnologia – TAM (DAVIS, 1989) mensurando a facilidade de utilização percebida em relação ao uso e a intenção de uso da *ProMerge*. O TAM postula que as crenças de um indivíduo sobre a facilidade e a utilização afetam sua atitude em relação ao uso de uma tecnologia e podem influenciar na sua intenção e no uso real, definindo os motivos de aceitação ou rejeição (DAVIS, 1989). Os participantes do experimento deveriam obrigatoriedade responder à dois questionários: impressão e características, detalhados nos apêndices B e C, sendo parte complementar do experimento. Assim, a partir das informações coletadas, conclui-se que a maioria dos participantes 65,63% (21/32) dos participantes possuem a graduação como o maior nível de escolaridade, outro aspecto importante é que muitos dos participantes ainda cursavam alguma especialização relacionada com a área de engenharia de software, no entanto a avaliação considerou somente o maior nível de graduação concluído.

A maioria dos participantes (16/32) possuem o curso de ciências da computação como graduação. Outro aspecto, é que a maioria dos participantes possuem mais de oito anos de estudos, comprovando que a graduação foi concluída em um período mais longo ou adicionado o período em anos de alguma pós–graduação ainda não concluída. A idade da maioria dos participantes

encontram-se na faixa etária compreendida entre vinte e seis e quarenta e cinco anos. A grande maioria dos participantes possuem mais de 8 anos no cargo que exercem atualmente nas instituições onde exercem suas ocupações. O cargo de desenvolvedor/programador corresponde à 50% (16/32), 21,88% (7/32) de analistas de sistemas e 15,63% (5/32) de arquitetos de software. Em relação aos aspectos que envolvem a experiência dos participantes com desenvolvimento de software, constatou-se que a maioria dos participantes possuem mais de cinco anos ou mais de experiência. Por fim, a grande maioria dos participantes foram indivíduos do sexo masculino representando, conforme detalhado na Tabela 17 e na Figura 24.



Fonte: Criado pelo Autor.
Figura 24: Estatísticas do Questionário de Impressões.

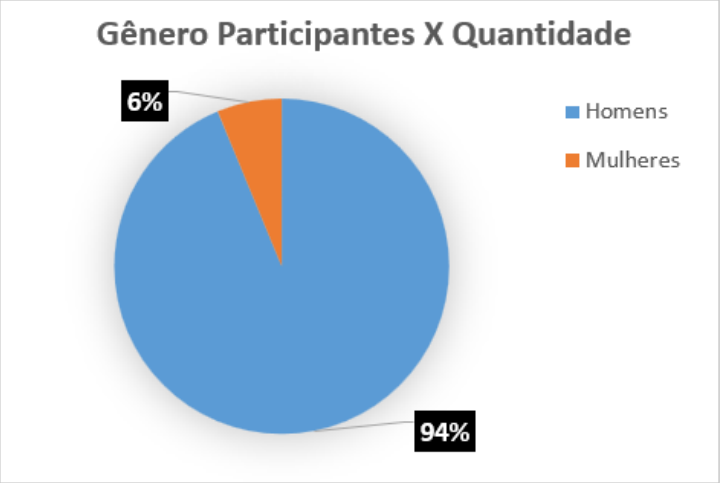
A Figura 25 detalha a quantidade de participantes por gênero, a maioria dos participantes (30/32) são do sexo masculino e apenas 02 participantes foram do sexo feminino. A Tabela 18 e a Figura 26 detalham informações quanto tempo(em anos) que os participantes ocupam os seus cargos. A idade dos participantes está representada na Tabela 20. A maioria dos participantes (19/32) possuem mais de cinco anos de experiência na função exercida, aspecto que assegura a qualidade dos resultados obtidos dos experimentos.

A Tabela 19 e a Figura 27 detalham informações quanto às profissões que os participantes exerciam durante a realização dos experimentos. A maioria dos participantes (16/32) atuam

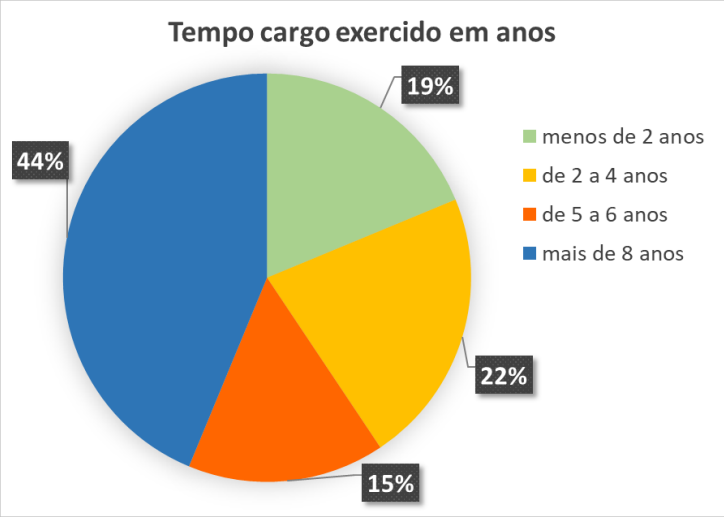
Características (n=32)	Resposta	Quantidade	Percentual
Nível Educação	Graduação	21	**65,63%**
	Especialização	7	21,88%
	Mestrado	4	12,5%
Graduação	Análise de Sistemas	9	28,13%
	Sistemas da Informação	4	12,5%
	Ciência da Computação	16	**50%**
	Engenharia da Computação	3	9,38%
Quantidade Anos Estudando	< 2 anos	1	3,13%
	2–4 anos	7	21,88%
	5–6 anos	10	31,25%
	7–8 anos	7	21,88%
	> 8 anos	7	21,88%
Idade Participantes	20–25 anos	5	15,63%
	26–30 anos	8	25%
	31–35 anos	2	6,25%
	36–40 anos	5	15,63%
	41–45 anos	6	18,75%
	> 45 anos	6	18,75%
Tempo Cargo	< 2 anos	3	9,38%
	2–4 anos	2	6,25%
	5–6 anos	7	21,88%
	7–8 anos	1	3,13%
	> 8 anos	19	**59,38%**
Cargo Atual	Desenvolvedor	16	**50%**
	Analista de Sistemas	7	21,88%
	Arquiteto de Software	5	15,63%
	Analista de Testes	1	3,13%
	Diretor de Tecnologia	1	3,13%
	Professor/Instrutor	1	3,13%
	Analista de Qualidade	1	3,13%
Tempo Experiência Cargo Atual	< 2 anos	6	18,75%
	2–4 anos	7	21,88%
	5–6 anos	5	15,63%
	> 8 anos	14	43,75%
Sexo	Masculino	30	**93,75%**
	Feminino	2	6,25%

(n) Número de participantes

Tabela 17: Resultado Questionário de Características (Apêndice B).



Fonte: Criado pelo Autor.
Figura 25: Gênero dos Participantes.



Fonte: Criado pelo Autor.
Figura 26: Tempo no Cargo dos Participantes.

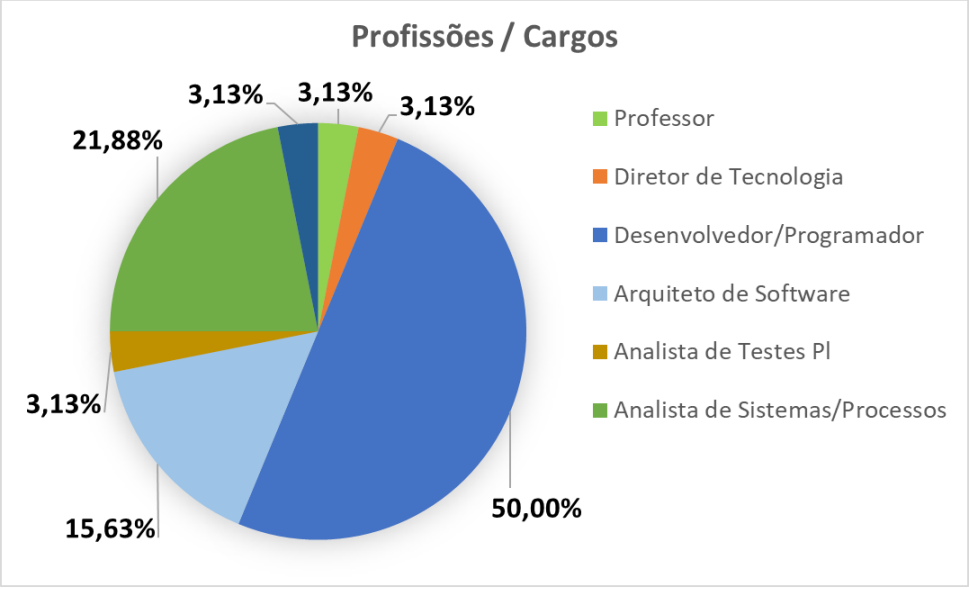
Tempo cargo participantes	Quantidade
Menos de 2 anos	6
De 2 a 4 anos	7
De 5 a 6 anos	5
Mais de 8 anos	14

Tabela 18: Dados dos Cargos dos Participantes.

como desenvolvedores de software. Esse é outro aspecto muito importante, pois o experimento realizado por trinta e dois participantes buscou avaliar a utilização da *ProMerge* que trata-se de um abordagem de resolução proativa de conflito utilizando informações de contexto. Assim, ficou assegurado que o público destinado foi prospectado corretamente, contribuindo com a robustez e qualidade dos resultados obtidos.

Profissão/Cargo participantes	Quantidade
Analista de Qualidade de Software	1
Analista de Sistemas/Processos	7
Analista de Testes	1
Arquiteto de Software	5
Desenvolvedor/Programador	16
Diretor de Tecnologia	1
Professor	1

Tabela 19: Cargos dos Participantes.

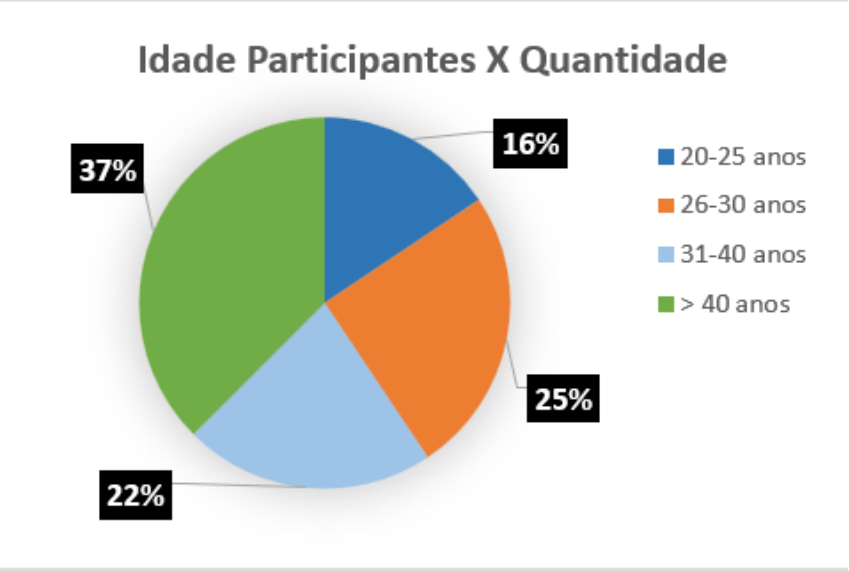


Fonte: Criado pelo Autor.
Figura 27: Definição dos Cargos dos Participantes.

A análise do perfil dos participantes apresentados indicou que a amostra de trinta e dois experimentos executados é aceitável e válida. Logo, produziu informações e dados quantitativos e

Idade participantes	Quantidade
20 – 25 anos	5
26 – 30 anos	8
31 – 40 anos	7
> 40 anos	12

Tabela 20: Idade dos Participantes.



Fonte: Criado pelo Autor.
Figura 28: Avaliação da Idade dos Participantes.

qualitativos que expressam a realidade quanto a utilização da *ProMerge* dentro de um ambiente de desenvolvimento de software. Tendo relação direta com todas as áreas e atividades de desenvolvimento de software. Os resultados estatísticos a partir dos experimentos representaram evidências claras para a rejeição da hipótese 1. As hipóteses nulas 2 e 3 não poderão ser rejeitadas. A análise dos dados de todas as hipóteses está apresentada na Seção 5.2. As avaliações dos dados de todas as questões do experimento controlado foram resumidas na Tabela 21 sendo agrupadas por variável dependente.

A Tabela 21 indica que o grupo de variáveis dependentes da *ProMerge* apresentaram melhores resultados em relação à tradicional. A Tabela 22 detalha as informações segmentadas por cenário e tarefa. Exibindo os valores médios de cada uma das variáveis dependentes, detalhando as informações apresentadas na Tabela 21. Portanto, com os resultados coletados, pode-se observar que houve diferença significativa entre os dois cenários do experimento controlado para cada uma das tarefas da Tabela 22. Sendo sustentada pelas estatísticas nas seções seguintes. A variável esforço (tempo) (ET) utilizando a *ProMerge* reduziu em média 5,49% o tempo utilizado na resolução das tarefas em comparação a abordagem tradicional. A variável Taxa de Corretude (CO) utilizando a *ProMerge* também apresentou uma redução média de 12,18% em relação à abordagem tradicional. Por fim, a Taxa de Erro (TE) utilizando a *ProMerge* apresentou uma

	ProMerge			Tradicional		
	ET	CO	TE	ET	CO	TE
Min	194	0,25	0,00	210	0,25	0,00
25th	245,50	1,00	0,00	289	0,75	0,00
Med	298	1,00	0,00	323,50	1,00	0,00
75th	375,50	1,00	0,00	395,75	1,00	0,25
Max	470	1,00	0,75	461	1,00	0,75
Mdn	311,65	0,90	0,09	338,75	0,87	0,13
DsP	72,67	0,22	0,28	65,64	0,22	0,21

Legenda: *(Min)* Mínimo, *(Med)* Média, *(Mdn)* Mediana, *(Max)* Máximo, *(DsP)* Desvio Padrão, *(ET)* esforço (tempo) em segundos, *(CO)* Taxa de Corretude, *(TE)* Taxa de Erro, *(n)* Participantes

Tabela 21: Estatística Descritiva dos Resultados (n=32).

redução significativa média de 16,42% em relação à abordagem tradicional.

Abordagem	Variável	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Tarefa 5	Média
ProMerge	Esforço (Tempo) (ET)	312s	245s	223s	134s	93s	201s
	Taxa de Corretude (CO)	0,91	0,91	0,86	0,92	0,94	0,91
	Taxa de Erro (TE)	0,09	0,09	0,14	0,08	0,06	0,09
Tradicional	Esforço (Tempo) (ET)	339s	254s	233s	138s	95s	212s
	Taxa de Corretude (CO)	0,88	0,89	0,80	0,88	0,90	0,87
	Taxa de Erro (TE)	0,13	0,11	0,20	0,12	0,10	0,13

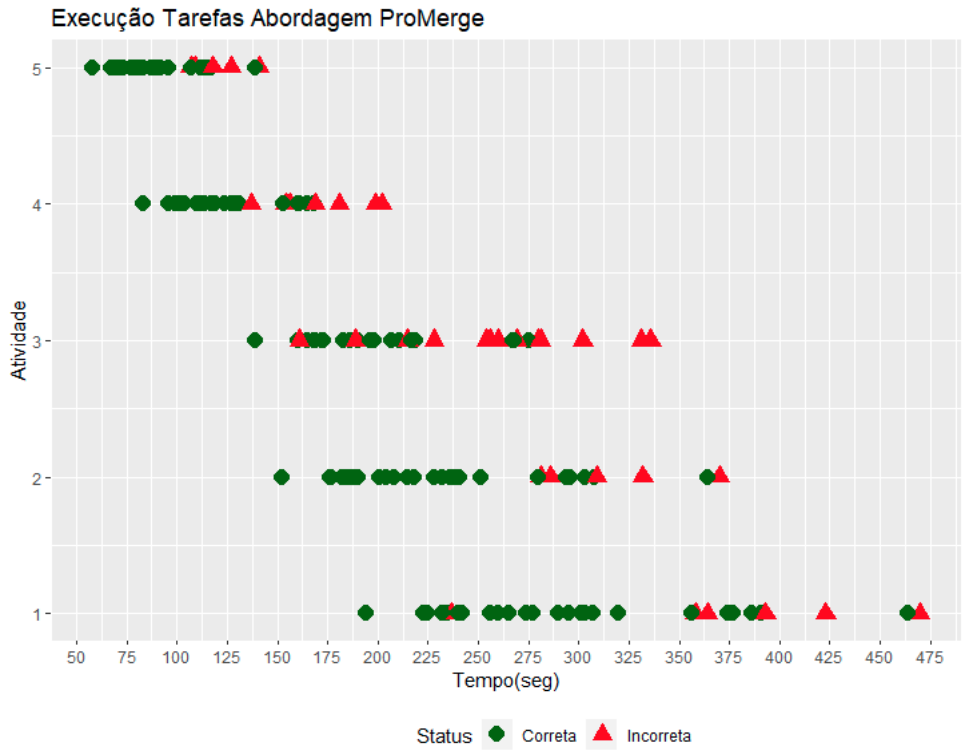
Legenda: (s) - Tempo em segundos.

Tabela 22: Valores Médios por Tarefa.

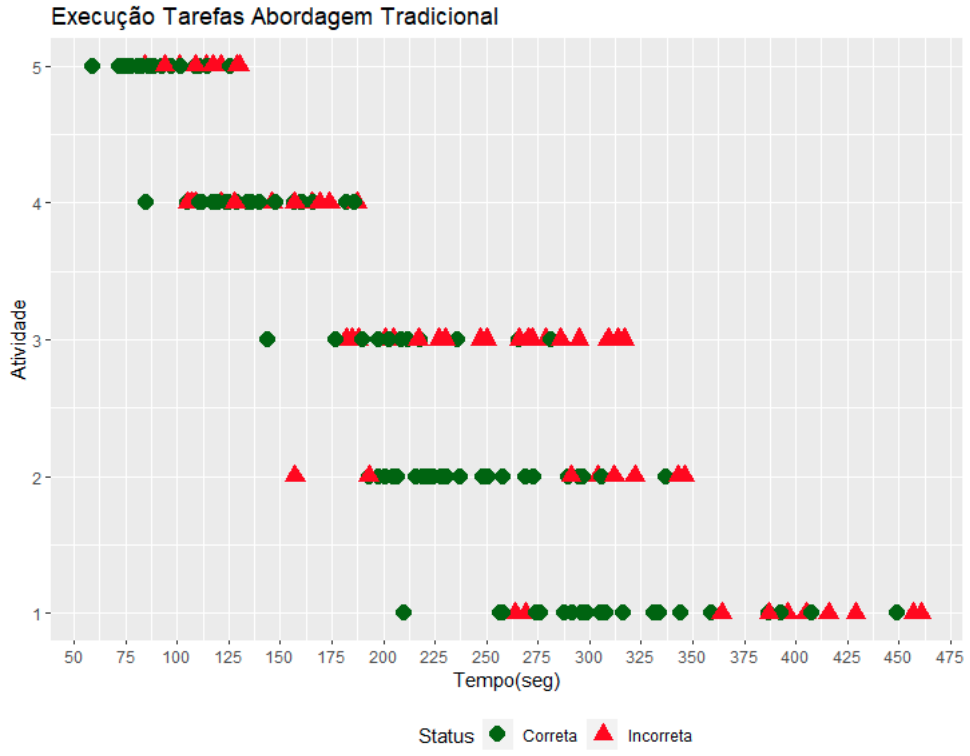
5.7.3 Hipótese 1: Esforço (Tempo)

A Hipótese H1 investigou o esforço (tempo) aplicado na resolução das tarefas propostas para os dois cenários dos experimentos. A Tabela 23 apresenta os resultados da estatística descritiva a partir das informações obtidas. O principal achado indica que o esforço (tempo) aplicado pelos participantes na resolução das tarefas utilizando a *ProMerge* foi inferior em relação utilização da abordagem tradicional, conforme representado pelas Figuras 29 e 30. Os dois cenários apresentaram iguais níveis de pontuação das tarefas considerados durante as avaliações. Todas as questões apresentaram características técnicas de implmentação semelhantes entre si e com iguais níveis de complexidade.

Para a compreensão dos resultados, as Figuras 31 e 32 detalham as respostas classificadas em correta ou incorreta. Caso uma a atividade fosse definida como correta, todos os itens avali-

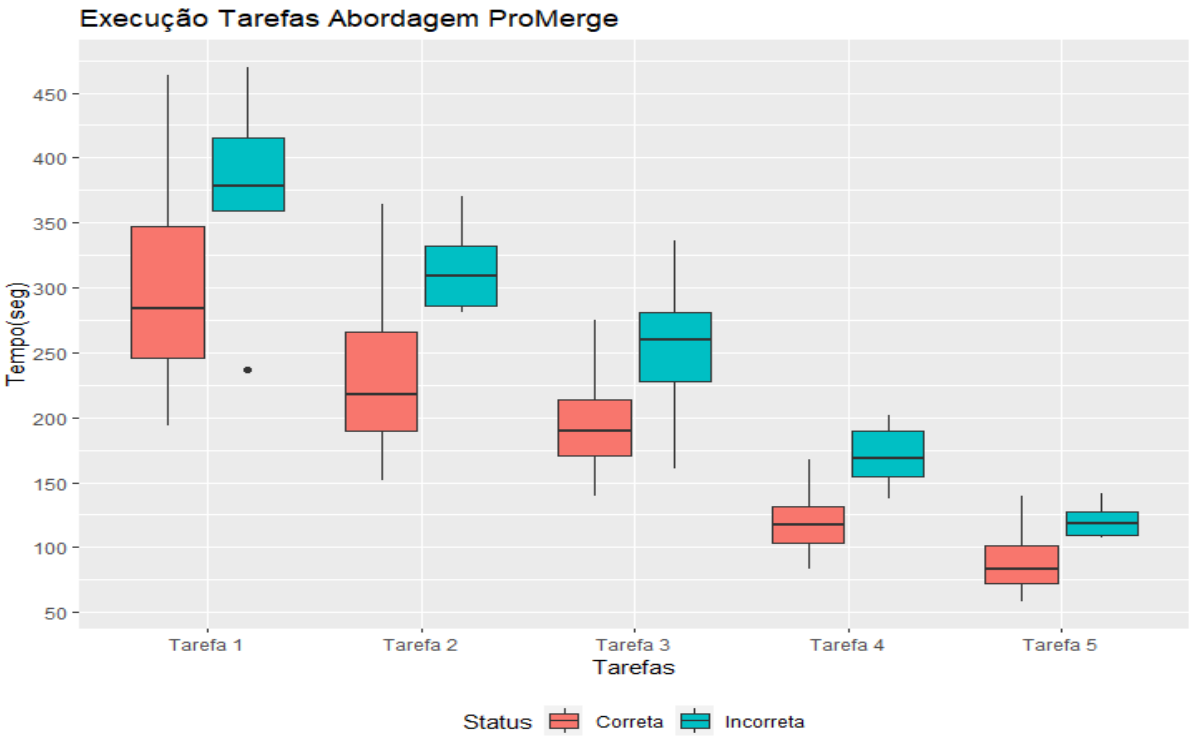


Fonte: Criado pelo Autor.
Figura 29: Avaliação do Esforço (Tempo) – *ProMerge*.



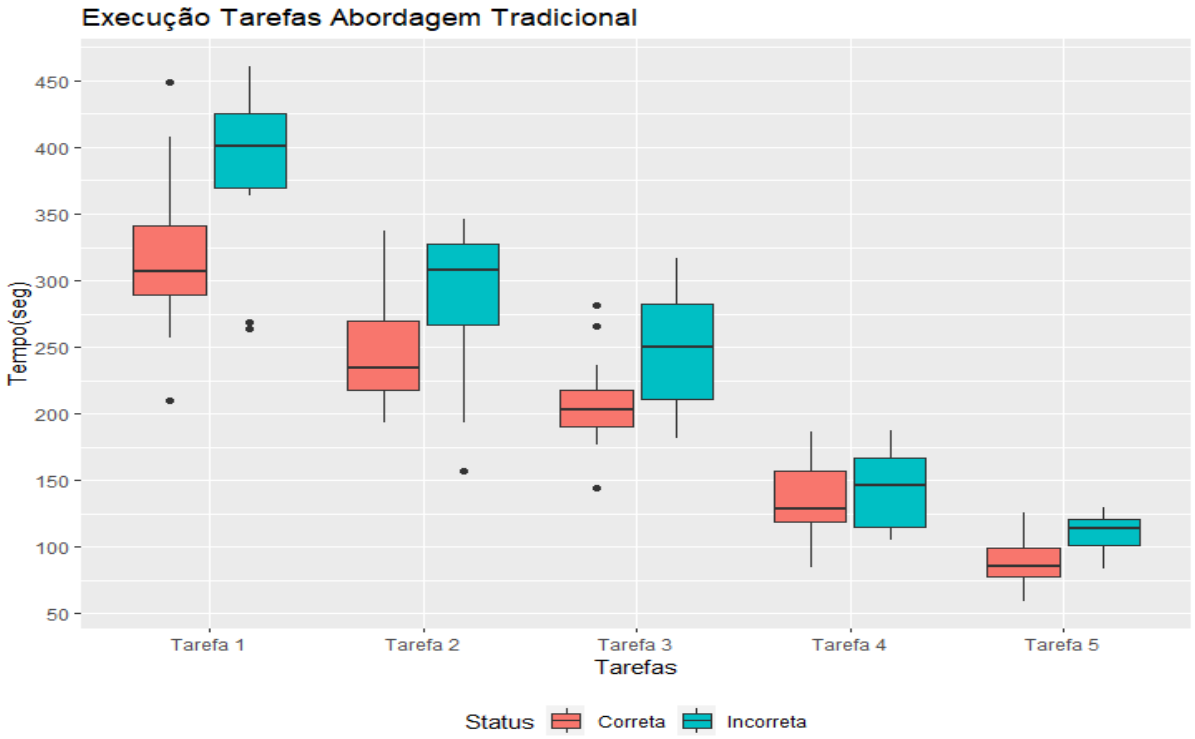
Fonte: Criado pelo Autor.
Figura 30: Avaliação do Esforço (Tempo) – Tradicional.

ados deveriam ser contemplados. Portanto, para cada uma das tarefas do experimento nos dois cenários possíveis, os dados foram comparados de acordo com a resposta apresentada. Facilitando a avaliação dos dados através de um gráfico do tipo box plot. Em todas as respostas corretas, detectou-se que o esforço (tempo) aplicado para resolução da tarefa com o auxílio da *ProMerge* foi menor em relação às tarefas executadas com a abordagem tradicional. Assim, os resultados obtidos com auxílio da *ProMerge* apresentaram índices melhores em relação a abordagem tradicional. O esforço aplicado na correta interpretação da utilização da *ProMerge* foi muito importante para a obtenção dos níveis de assertividade superiores em relação à abordagem tradicional.



Fonte: Criado pelo Autor.
Figura 31: Comparativo da Avaliação do Esforço (Tempo) – *ProMerge*.

A Tabela 23 detalha os resultados onde a utilização da *ProMerge* apresenta maior nível de correção quando comparadas com a abordagem tradicional. As descobertas revelam a estatística e os valores de p para as comparações aos pares. Na maioria das circunstâncias o valor p foi menor ($p \leq 0,05$), portanto a hipótese nula de H2 pode ser rejeitada. O teste de Wilcoxon foi utilizado para avaliar a distribuição não normalizada das informações, usado como uma hipótese estática não paramétrica alternativa aplicada para combinar as médias de duas amostras relacionadas, como testes de diferenças pareadas. O t-test pareado utiliza a variação média entre as medidas dos dados do experimento dentro dos diferentes níveis de complexidade e detalha os valores de p para a correlação pareada. Portanto, os resultados comprovaram que os participantes do experimento não necessitaram de um esforço (tempo) extra considerável para



Fonte: Criado pelo Autor.
Figura 32: Comparativo da Avaliação do Esforço (Tempo) – Tradicional.

a compreensão dos arquivos dos código-fonte utilizando a abordagem da *ProMerge* em relação a abordagem tradicional.

Tarefas	Wilcoxon		t-test pareado		
	p-value	W	p-value	GL	t-value
Todas	**0,001**	278,65	**0,001**	159	7,799
Tarefa 1	**0,001**	512,5	**0,001**	31	7,376
Tarefa 2	0,201	406	**0,004**	31	3,087
Tarefa 3	0,107	410,5	**0.001**	31	3,804
Tarefa 4	**0,042**	341	**0,019**	31	2,457
Tarefa 5	0,110	349	0,116	31	1,614

Legenda: *(W)* Soma dos Ranks Sinalizados, *(GL)* Grau de Liberdade.

Tabela 23: Análise Estatística da Resolução das Tarefas dos Experimentos.

Os resultados demonstram a existência de uma variação significativa entre as proporções de respostas corretas produzidas no código-fonte utilizando a *ProMerge* se comparadas com a abordagem tradicional e descreve os valores de p aos pares para cada medida. Os valores de

p destacados apontam os resultados estatisticamente significativos e rejeitam a hipótese nula. Assim, nota-se que a soma dos postos assinados (W) determina a direção em que o resultado é significativo. Assim, na tarefa 4, o W tem o valor de 341 e o valor p é inferior a 0,05 (p = 0,042) no teste de Wilcoxon para a medição entre as duas abordagens. O nível de inconsistência é significativamente menor quando comparada com o grupo com a abordagem tradicional. Por fim, para a H2 desse experimento há evidências estatísticas que concluem que a *ProMerge* impacta de maneira positiva na eficácia em diversos cenários. A variável esforço (tempo) (ET) apresentou uma redução de 5,49% no valor médio em relação a abordagem tradicional, conforme detalhado na Tabela 22.

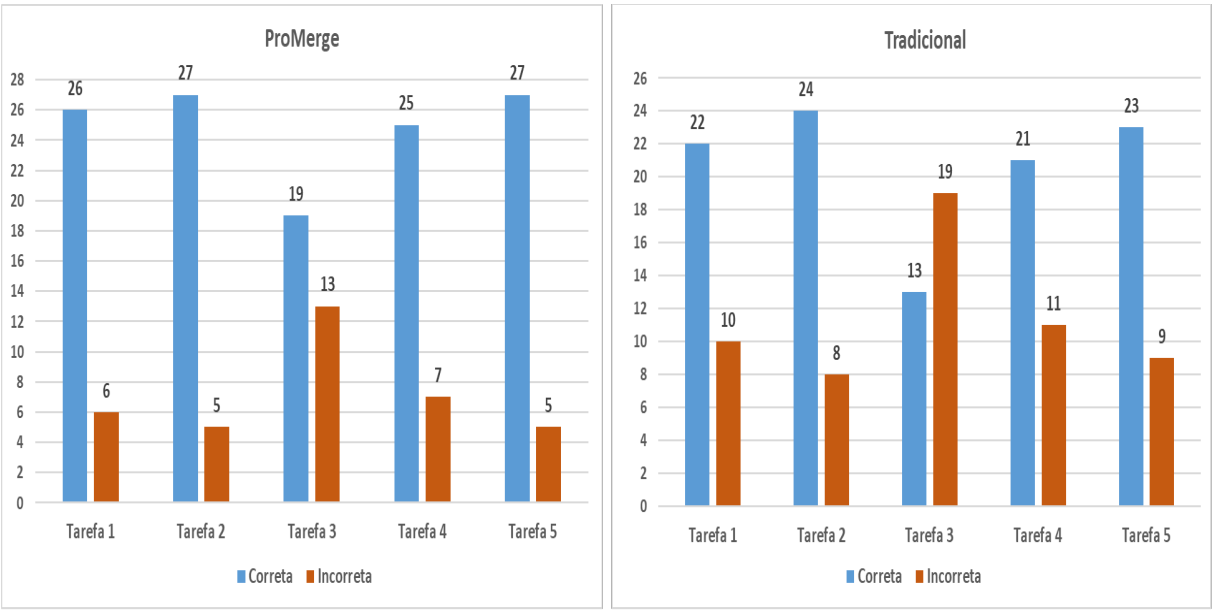
Conclusão Hipótese 1: O principal achado foi de que o esforço (tempo) da maioria das tarefas do experimento controlado utilizando a *ProMerge* foi inferior quando comparados a abordagem tradicional, negando assim a hipótese nula de H1.

5.7.4 Hipótese 2: Taxa de Corretude

A hipótese H2 analisou os dados coletados relativos ao impacto da Taxa de Corretude das respostas das atividades do experimento utilizando a *ProMerge* e a tradicional. A Tabela 24 e a Figura 33 analisaram a estatística descritiva dos dados coletados e apresentaram os resultados obtidos para cada uma das tarefas. O eixo y representa as quantidades de acertos e erros para cada uma das tarefa. O eixo x representa todas as tarefas executadas nos dois cenários que compõe o experimento. O histograma mostra os resultados por tarefas avaliando os graus de complexidades de cada uma das tarefas. Assim, evidenciou-se que a *ProMerge* apresentou melhores níveis de CO por tarefa executada em relação a abordagem tradicional. Portanto, auxiliou os desenvolvedores na compreensão e resolução das tarefas propostas pelos experimentos.

Portanto, a sua utilização não assegura a superioridade da *ProMerge* em relação a qualquer outro tipo de abordagem, às variações apresentadas foram categorizadas por tarefa executada e apresentaram alterações principalmente em decorrência da experiência e conhecimento técnico de cada um dos participantes do experimento. Assim, aplicou-se o método estatístico McNemar para avaliação da hipótese 2. A Tabela 24 apresenta os valores dos chi-quadrados e dos p-values produzidos. Por fim, um aspecto importante a ser considerado, é que na avaliação dos dados de todas as tarefas foi obtido o valor de p-value igual a 0,001 que é inferior a 0,05. Portanto, sendo um resultado significativo e sugere que existe uma diferença a ser melhor investigada entre as taxas de inconsistência usando a *ProMerge* e a tradicional.

Os resultados indicam que há uma variação significativa entre as proporções de respostas corretas produzidas pela utilização da *ProMerge* quando comparados a abordagem tradicional. Portanto, há evidências estatísticas suficientes para concluir que a *ProMerge* impacta positiva-



Fonte: Criado pelo Autor.
Figura 33: Taxa de Corretude das Tarefas dos Experimentos *ProMerge*.

Tarefas	ProMerge		Tradicional		Resultados	
	Correta	Incorreta	Correta	Incorreta	Chi-Square	p-value
Todas	124	36	103	57	12,90	**0,001**
Tarefa 1	26	6	22	10	1,50	0,221
Tarefa 2	27	5	24	8	0,80	0,372
Tarefa 3	19	13	13	19	3,13	0,078
Tarefa 4	25	7	21	11	1,13	0,289
Tarefa 5	27	5	23	9	2,25	0,134

Tabela 24: Teste McNemar – *ProMerge* Vs Tradicional.

mente na eficácia no cenário de avaliação dos dados de todas as tarefas. É importante mencionar que todos os resultados alcançados por tarefa quanto ao índice de CO foram 12,18% superiores utilizando como critério de avaliação os valores médios quando comparados com a abordagem tradicional. Por fim, a execução e a avaliação de novos experimentos segmentados e categorizados de acordo com a experiencia dos participantes, poderá apresentar resultados diferentes dos apresentados pelos experimentos e detalhados na Tabela 24.

Conclusão Hipótese 2. Os p-values de todas as tarefas encontrados individualmente foram superiores à 0,05 **(p > 0,05)**, portanto não podemos rejeitar a hipótese H2 nula. O principal achado foi que para a avaliação dos resultados de todas as tarefas, o p-value foi inferior à 0,05 **(p < 0,05)** podendo-se negar a hipótese H2 nula para esse contexto.

5.7.5 Hipótese 3: Taxa de Erro

A hipótese H3 analisou as taxas de erros encontradas individualmente em todas as tarefas realizadas pelos experimento controlado utilizando os dados coletados relativos à *ProMerge* e a tradicional e conforme detalhado na Seção 5.2. Cada uma das tarefas foi avaliada individualmente e os arquivos de código-fonte resultantes dos experimentos foram comparado com os arquivos fontes definidos como padrão de resposta, desenvolvidos separadamente e utilizados como modelo de análise. Essa avaliação foi importante e necessária para evitar a criação de qualquer problema e mitigar possíveis falhas decorrentes da uma análise parcial, conforme detalhado na Etapa 3 da Seção 5.5. A Tabela 25 analisou a estatística descritiva dos dados coletados de forma ampla e imparcial de todas as tarefas, individualmente e por fim agrupando todas as tarefas para avaliação em um contexto único.

Tarefas	Wilcoxon		t-test pareado		
	p-value	W	p-value	GL	t-value
Todas	**0.001**	49,15	**0,022**	31	2,395
Tarefa 1	0.218	17,5	0,103	31	1,678
Tarefa 2	0.312	12	0,183	31	1,359
Tarefa 3	**0.040**	40,5	**0,018**	31	2,489
Tarefa 4	0.164	35	0,096	31	1,716
Tarefa 5	0.062	15	**0,022**	31	2,395

Legenda: *(W)* Soma dos Ranks Sinalizados, *(GL)* Grau de Liberdade.

Tabela 25: Análise Estatística das Taxas de Erro dos Experimentos.

Detectou-se que apenas a tarefa 3 apresentou o p-value inferior a 0,05 ($p \leq 0,05$). Porém, quando da avaliação dos dados de todas as tarefas, o p-value foi menor que 0,05 ($p \leq 0,05$) o que é um resultado significativo. Esse resultado sugere a existência de uma diferença entre a taxas de inconsistência usando a *ProMerge* e a tradicional. No entanto, considerando os cenários de cada um dos experimentos, vários resultados obtidos assumiram significância válida ($p \leq 0,05$). Assim, é relevante notar que o grau de liberdade (DF) mostra a direção em que o

resultado é significativo. Portanto, o teste de Wilcoxon foi usado como uma hipótese estática não paramétrica alternativa, aplicada para combinar as médias de duas amostras relacionadas e também para os testes de diferenças pareadas. Por fim, os testes estatísticos foram aplicados para avaliar às hipóteses e fornecer suporte estatístico aos resultados encontrados. Assegurando a validade dos resultados, tanto que variável Taxa de Erro (TE) apresentou uma redução significativa de 30,76% no valor médio em relação a abordagem tradicional, conforme dados detalhado na Tabela 22.

Conclusão Hipótese 3. Os p-values encontrados da maioria das tarefas foram superiores à 0,05 **(p > 0,05)**, portanto não podemos rejeitar a hipótese H3 nula. O principal achado foi que para a avaliação única de todas as tarefas, o p-value encontrado foi inferior à 0,05 **(p < 0,05)**, podendo-se negar a hipótese H3 nula para esse contexto.

Os dados qualitativos foram coletados através de dois questionários: Características(B) e Impressões (C), registros de áudio/vídeo e transcrições, o que ajudou a obter potencialmente vários aspectos e características complementares para explicar os resultados quantitativos obtidos e posteriormente, realizando conclusões a partir de uma cadeia de evidências e do alinhamento sistemático dos dados quantitativos e qualitativos.

5.8 Ameaças de Validação

Em qualquer tipo de pesquisa que envolve execução de experimentos, as ameaças à validade dos resultados podem potencialmente influenciar no desenhos experimentais utilizados e nos resultados alcançados (FARIAS; GARCIA; LUCENA, 2014; FARIAS et al., 2015). Asssim, essa pesquisa não foi uma exceção, foram discutidas as novas abordagens desenvolvidas com a finalidade mitigar as situações que possam criar qualquer problema nos resultados alcançados. A Seção 5.8.1 discute as ameaças de construção detectadas por essa pesquisa e compara com seus objetivos definidos na Seção 1.4. A Seção 5.8.2, detalha as ameaças estatísticas encontradas para a correta validação dos resultados e como mitigá-las. A Seção 5.8.3, discute as ameaças internas das conclusões derivadas dos resultados obtidos conforme detalhados na Seção 5. Por fim, a Seção 5.8.4 discute as ameaças para a generalização dos resultados obtidos as ameaças que possam influenciá-los e a melhor maneira de mitigá-las.

5.8.1 Construção

A validade de construção refere-se ao grau de inferências a partir da avaliação dos resultados incorporados pela pesquisa (JORGENSEN, 2005). Particularmente, foram avaliados: (1) Informações coletadas a partir dos experimentos controlados e, (2) se os métodos qualificadores

e quantificadores das variáveis foram apropriados ao contexto do experimento, (3) se os processos do experimento foram executados corretamente e por fim,(4) se as alterações nos arquivos de códigos-fonte poderiam criar algum tipo de problema nos resultados

Métodos de Quantificação. Quanto à quantificação da variável esforço (tempo), foi levado em consideração a unidade de minutos utilizados por cada desenvolvedor na finalização das tarefas do experimento controlado. Assim, devido ao conhecimento técnico dos participantes e a complexidade envolvida nas atividades não foi necessário aumentar o tempo em horas. A variável corretude foi mensurada de forma manual realizando-se uma análise comparativo entre a alteração de um código-fonte previamente analisado e otimizado e o código-fonte resultante de cada tarefa de cada participante, e na quantidade dos aspectos esperados em cada resposta.

Corretude. Foram realizadas avaliações de resultados para que os dados coletados encontrassem alinhados com os objetivos e as hipóteses desse experimento controlado. Os procedimentos adotados para quantificação foram cuidadosamente elaborados e seguiram práticas muito bem conhecidas de quantificação conforme demonstrados em (CARBONERA; FARIAS; BISCHOFF, 2020; GONÇALES et al., 2015b; BISCHOFF et al., 2019; KITCHENHAM et al., 2009; KITCHENHAM; CHARTERS, 2007).

Composição dos Cenários. Outra ameaça gerenciada estava relacionada à definição das tarefas do experimento controlado, que poderia ter criado algum problena mesmo involuntariamente nos resultados. Foram realizados trinta e dois experimentos com desenvolvedores com diferentes níveis técnicos, experiência e formações acadêmicas, como alternativa de mitigação à essa ameaça, seria a segmentação dos participantes por níveis de experiência ou formação acadêmica. Os desenvolvedores que apresentavam maior tempo de experiência na área conseguiram de maneira comprovada compreender e resolver as tarefas mais rapidamente e com melhor desempenho. Por fim, observou-se que o padrão dos resultados obtidos não estava relacionado a nenhum dos achados empíricos nos estudos avaliados por essa pesquisa assegurando a qualidade das informações.

5.8.2 Estatísticas

Essa Seção detalha as ameaças ocorridas sobre os testes estatísticos e da sua mitigação. Foi validado também, a força covariável bem como da causa e efeito (HYMAN, 1982). Nas etapas iniciais, analisou-se a possibilidade dos testes serem definidos incorretamente, bem como o índice da covariação e o nível de confiança que a avaliação possibilita, o que não se concretizou. Adicionalmente, a segunda inferência poderia ter assumido indevidamente uma relação causal com as variáveis selecionadas (WEISS, 2000), o que não foi detectado também. Para mitigar possíveis ameaças quanto aos resultados, foram realizadas abordagens e avaliações estatísticas adequadas e a partir das variáveis dependente conforme detalhadas na Tabela]16. Os riscos da covariância de causa e efeito entre as variáveis dependentes foram reduzidos examinando a distribuição normal dos resultados. Portanto, foi necessário avaliar se os dados poderiam ser

aplicados em métodos estatísticos paramétricos ou não paramétricos. Assim, foi adotado o teste de t-test pareado para verificar a distribuição normal dos dados (PETRUCCELLI; NANDRAM; CHEN, 1999).

Os resultados dos testes estatísticos aplicados disponibilizaram informações suficientes para a realização de uma análise detalhada e criteriosa. Comprovando que a correta aplicação das atividades, sem interrupções durante a execução das atividades pelos participantes assegurou uma maior qualidade e confiabilidade dos resultados. O experimento controlado objetivou o maior número de amostras possíveis para melhorar a eficácia estatística. A variação estatística admitida para todas as hipóteses foi de 5,00% ($p \leq 0,05$) e seguiu diretrizes bem estabelecidas conforme definidas em (ARMINO, 2016; SEGALOTTO, 2018; TROCHIM; DONNELLY, 2001) aumentar as validades das conclusões. Outros aspectos importantes, foi que tentou-se obter o maior número possível de participantes, as atividades foram planejadas considerando os tipos de alteração mais comumente executadas pelos participantes. A adoção dessas boas práticas mitigou possíveis erros que poderiam criar qualquer situação quanto às verdadeiras causas das variáveis estudadas por essa pesquisa.

5.8.3 Internas

Essa Seção avalia os principais resultados obtidos na execução de um experimento controlado realizado. A variedade de elementos distintos pode impactar diretamente no desempenho do indivíduo e influenciar circunstâncias adicionais (WOHLIN et al., 2012). Assim, a validade representa o potencial para realizar o julgamento preciso a partir dos resultados do experimento controlado. Vários testes-pilotos foram executados para confirmar que a validade interna foi assegurada, pois: (1) a precedência temporal é conhecida. Portanto, a sequência de execução e avaliação estão definidas, as alterações dos arquivos fonte precedem a identificação dos conflitos e o tempo para resolução desse conflito auxiliados pelas informações de contexto histórico armazenadas; (2) a covariância foi avaliada, com a utilização da *ProMerge* foi detectado uma variação do esforço (tempo) aplicados pelos participantes na resolução de conflitos; (3) as informações de contexto obtidas auxiliaram na criação de indicadores de desempenho e produtividade. Por fim, (4) foi identificado o grau de severidade dos conflitos, através da relevância das funcionalidades dentro do contexto do projeto avaliado pelo experimento controlado. Essa pesquisa atende plenamente esses quatro requisitos para a validade interna do processo.

5.8.4 Externas

Essa Seção diz respeito às ameaças externas e à validade dos resultados coletados em outros contextos conforme conceitos amplamente difundidos em (MITCHELL; JOLLEY, 1988). As abordagens utilizadas por esse experimento controlado são equivalentes às utilizados na indústria de software. A generalização dos resultados dos experimentos é reconhecida pela

perspectiva acadêmica e pela indústria, agregando valor aos domínios realistas transformando conhecimento teórico em prático. Assim, é necessário avaliar qual é o nível da generalização possível, permitindo que os resultados possam ser abstraídos em outros ambientes ou configurações, como por exemplo: diferentes tipos de projetos, diferentes níveis de experiência entre os participantes ou na separação de atividades de acordo com a experiência de cada participante.

Assim, torna-se necessário destacar que os cenários desenvolvidos e empregados por esse experimento são menores se comparados aos utilizados pela indústria de software, foi analisada também se a relação causal poderia ser realizada por participantes em diferentes configurações. Portanto, como técnica de avaliação foi considerada a distribuição normal dos resultados, o que torna possível definir o nível da generalização a ser adotado. Foram identificados também vários critérios que podem ser aplicados em contextos simulares. Por fim, os resultados comprovam que é possível generalizar a execução do experimento controlado dessa pesquisa em contextos semelhantes. Desde que os participantes estejam devidamente contextualizados com a utilização de ferramentas de versionamento de arquivos de códigos-fonte centralizados e possuir entendimento técnico da *ProMerge*. Portanto, a partir da execução de novos experimentos as atividades deverão ser executadas nas mesmas sequências conforme estabelecidas nos Apêndices D e E fundamentados pelas informações de contexto coletadas durante a realização dos experimentos, que fornecem subsídios técnicos para a resolução dos conflitos detectados.

6 CONCLUSÕES E TRABALHOS FUTUROS

Esse trabalho apresentou a *ProMerge* que trata-se de uma abordagem para auxiliar desenvolvedores na detecção e resolução proativa de conflitos utilizando informação de contexto. Sendo composta por uma arquitetura cliente/servidor baseada em componentes que podem ser estendidos e em componentes e algoritmos atuando de maneira independente contudo interrelacionados. A *ProMerge* foi avaliada através da realização de experimento controlado executado por trinta e dois participantes cada qual realizando dez atividades experimentais divididas em dois cenários, totalizando trezentos e vinte cenários de avaliação. Sendo avaliadas as seguintes variáveis dependentes: esforço (tempo), Taxa de Corretude e de Erro, conforme detalhadas na Tabela 16.

Assim, para cada um dos cenários do experimento controlado dessa pesquisa foram definidas cinco atividades semelhantes, com iguais níveis de complexidade e critérios de avaliação. Os dados foram coletados e cuidadosamente analisados para produzir importantes métricas de avaliação e resultados conclusivos. Sendo suportados pelos testes estatísticos confirmando que a utilização da *ProMerge* em uma das etapas dos experimentos auxiliou os desenvolvedores na correta compreensão e resolução dos conflitos durante o desenvolvimento das atividades, reduzindo o esforço (tempo) aplicado em relação a abordagem tradicional. Os resultados obtidos após a aplicação dos questionários de características e de impressões (Apêndices B e C) corroboraram pela usabilidade da *ProMerge* apresentando uma interface amigável e de fácil interpretação pelos desenvolvedores participantes do experimento.

O conceito do commiting time foi uma das lacunas encontradas a partir da realização da SMS e que foi implementada na *ProMerge*. Esse conceito é representado pelos registros dos históricos dos intervalos de tempo entre os vários commits realizados durante a execução das tarefas. Podendo ser totalmente auditável e tem como finalidade ser um indicador de produtividade e qualidade quanto à execução das tarefas, pois são registrados os históricos resultantes da compilação. A regularidade dos commits realizados era de no mínimo um por tarefa, podendo chegar a três. A realização do commits junto ao repositório foi para atualização da base de histórico de contexto auxiliando os desenvolvedores e na avaliação da qualidade das tarefas executadas. Outro aspecto importante foi a avaliação do grau de severidade, sendo outra lacuna encontrada e implementada na *ProMerge*. Sendo a definição do nível de importância que uma funcionalidade tem dentro do contexto avaliado, sendo classificada a partir do padrão adotado na Tabela 13, e para cada referência encontrada da funcionalidade avaliada foi adicionada a pontuação de cinquenta pontos. Assim, definindo o nível de importância da funcionalidade avaliada e quais os cuidados necessários a serem adotados evitando qualquer tipo de problema.

O restante deste Capítulo está organizado em quatro seções: (1) A Seção 6.1 apresenta um resumo das contribuições obtidas a partir das QP definidas na Seção 1.3. (2) A Seção 6.2 destaca as limitações encontradas e os cuidados tomados por essa pesquisa. Por fim, (3) a Seção 6.3 apresenta algumas oportunidades de pesquisa contribuindo para o desenvolvimento de novos

trabalhos por essa importante área da engenharia de software moderna.

6.1 Contribuições

Essa Seção detalha as contribuições obtidas durante o desenvolvimento dessa pesquisa conforme as QP detalhados na Seção 1.3. Portanto, três aspectos contribuem significativamente: (1) Esse estudo realizou um SMS no área de integração de arquivos fontes; (2) A abordagem para resolução proativa de conflitos proposta pela *ProMerge* foi desenvolvido para avaliar aspectos relativos a conflitos diretos e indiretos. Por fim, (3) Realização de um experimento controlado com trinta e dois desenvolvedores para validar a *ProMerge* gerando conhecimento empírico e resultados estatísticos. Os itens abaixo descrevem individualmente as suas contribuições e contribuindo para a área pesquisada.

O estado da arte em integração de arquivos fontes (RQ–1): O SMS desenvolvido estabeleceu um protocolo muito bem definido para a busca de estudos relacionados à integração de código-fonte. Forneceu uma visão generalista sobre o tema na literatura atual. O conhecimento empírico foi obtido da organização dos dados dos SMS filtrados. Os estudos foram cuidadosamente classificados e categorizados quanto a metodologia empregada, grau de maturidade, local e ano de publicação, identificando-se assim várias lacunas a serem pesquisadas. A partir dos resultados do SMS, conseguiu-se definir uma direção para os desafios futuros na área de integração de código-fonte e os resultados apontaram quais áreas forma em que mais pesquisadas desenvolvidas e as potenciais áreas para o desenvolvimento de novas investigações.

Informações de contexto coletadas através da utilização da *ProMerge* (RQ–2 e RQ–3): A realização de um SMS a partir da avaliação das lacunas encontradas e a implementação dos conceitos relativos ao commiting time e ao grau de severidade caracterizam-se pelas principais contribuições dessa pesquisa e da *ProMerge*. Portanto, foi analisada a relação do tempo com a quantidade de commits realizados, podendo ser conflitantes ou não ou ainda com erros de integridade no código-fonte. Assim, foram apresentados indicadores robustos de produtividade e de qualidade a partir das informações coletadas. Por fim, o grau de severidade avaliou a importância que uma determinada funcionalidade ou método possui dentro de um contexto de um projeto. Sendo investigada a relevância e o relacionamento com aa demais funcionalidades existentes. Permitindo assim definir os cuidados e boas práticas necessários a partir da detecção dos diferentes tipos de conflitos existentes, além das análises estatísticas que investigaram as taxas de corretude e erro conforme detalhado na Seção 5.7.

6.2 Limitações da Pesquisa

Quanto aos estudos relacionados à integração de códigos-fontes, torna-se necessário reconhecer que existiram limitações durante o processo de análise e avaliação, nomeadamente nas áreas que envolvem a execução de experimentos e no desenvolvimento das novas abordagens.

Esse estudo, abordou durante o seu percurso, essencialmente questões a respeito do tempo aplicadas à resolução de conflitos pós-commit e das abordagens para detecção e resolução proativa de conflitos. Por fim, a partir das informações de contexto coletadas dos experimentos realizados, foi possível classificar as principais fragilidades desse estudo, que estão abaixo detalhadas e junto da melhor maneira de como mitigá-las.

Abordagem Proposta. A *ProMerge* foi desenvolvida a partir de um ambiente de avaliação robusto e confiável. Embora que tenha sido projetada para avaliar situações muito semelhantes às encontradas em ambientes da indústria de software, algumas ameaças e limitações potenciais podem ser evidenciadas principalmente quanto a entrada de dados. A arquitetura é totalmente componentizada e independente, eliminando possíveis ameaças quanto a validade do modelo proposto e dos dados coletados e avaliados. Assim, foi necessário a realização de testes em cenários conflituosos para assim conseguir realizar as correções necessárias, o que se caracteriza como um cenário atípico, mesmo após todos esses cuidados podemos afirmar a abordagem não está isenta de cenários não avaliados ou considerados;

Experimento Controlado. As variáveis de contexto existentes podem influenciar qualquer experimento envolvendo integração de código-fonte. Primeiro, há a necessidade de expandir e aumentar o número de participantes dos experimentos, segmentados por faixa de conhecimento e experiência técnica. Assim, foram selecionados trinta e dois desenvolvedores para o experimento controlado com a finalidade de mitigar parte desse problema e para não criar qualquer tipo de tendência na replicação desse experimento para futuras avaliações. Outro aspecto importante, envolve a avaliação dos resultados coletados dos experimentos realizados com a *ProMerge*. Portanto, para mitigar esse problema foram executados análises estatísticas, dentre elas o teste de Wilcoxon, McNemar e o t-Test pareado assegurando a integridade e validade dos dados. Ampliando a abrangência dos indicadores produzidos e na definição de novos modelos de avaliação, que poderá contribuir para a obtenção de resultado mais assertivos e objetivos.

6.3 Trabalhos Futuros

Portanto, com a finalização dessa pesquisa foi possível evidenciar quatro novas oportunidades de pesquisa para trabalhos futuros, as quais estão detalhadas abaixo:

- **Revisão Sistemática da Literatura.** A revisão analisou setenta estudos primários cuidadosamente selecionados (Apêndice A) após várias e cuidadosas etapas de seleção e utilizando critérios de avaliação muito bem definidos que são amplamente aplicados pela comunidade acadêmica, conforme detalhado no Capítulo 3. Assim, o tema pesquisa poderá sem ampliado considerando outras linha de pesquisas avaliados por outros estudos, dentre elas: inteligência artificial e dados psicofisiológicos (biométricos). A primeira linha de pesquisa através da investigação e avaliação dos algoritmos que realizam a integração dos trechos de códigos-fonte avaliados ou através das técnicas automáticas de integração. E por fim, a segunda técnica através da utilização de equipamentos de EEG

(EletroEncefaloGrama) que trata-se de um poderoso amplificador de corrente elétrica que é capaz de aumentar em milhares de vezes os sinais elétricos gerados pelo usuário. Tendo como finalidade avaliar o estado emocional e o nível de ansiedade e stress dos desenvolvedores durante a realização das atividades do experimento.

- **Abordagem Proposta.** A tecnologia cliente/servidor inicialmente adotada por essa pesquisa, permite que os componentes desenvolvidos sejam estendidos e melhorados com novas e importantes funcionalidades. Possibilitando a migração para uma arquitetura mais robusta e atual integrando com microserviços ou APIs externas. Outro aspecto importante é habilitar a integração com outros repositórios de arquivos fontes, dentre eles: Github (GITHUB, 2024) e TFS (TFS, 2024), sendo esses dois os mais utilizados pela indústria de software e pela comunidade acadêmica na atualidade.

- **Experimento Controlado.** Evidenciou-se a necessidade da realização de experimentos com uma quantidade de desenvolvedores superiores aos trinta e dois inicialmente avaliados. Assim, para uma avaliação mais completa o total de participantes avaliados poderá ser expandida para um total de cinquenta segmentados por faixa de experiência profissional e nível de conhecimento técnico. Inicialmente foram propostas dez atividades com o nível de complexidade entre médio e baixo. As dez atividades propostas divididas entre os dos cenários existentes sendo cinco atividades para cada um deles, a primeira atividade do primeiro cenário apresentava uma atividade correlata no segundo cenário, semelhante e com igual nível de complexidade d critérios de avaliação. O tempo médio de execução dos experimentos foi de cinquenta e dois minutos por participante. Portanto, como melhoria futura propõe-se realizar avaliações com novas atividades e em maior quantidade, tendo atividades com elevado nível de complexidade além das já existente, conseguindo assim realizar validações mais complexas, robustas e detalhadas.

- **Dados Psicofisiológicos (Biométricos).** Verificou-se a escassez de estudos que avaliam dados coletados com a utilização de equipamentos biométricos. Os EEG (EletroEncefaloGrama) são utilizados na coleta de informações das atividades elétricas do cérebro possibilitando medir o nível de stress e de ansiedade dos desenvolvedores durante a integração de trechos de códigos-fontes alterados. A realização de novas pesquisas e a execução de experimentos controlados poderá evidenciar os principais motivos e as causas que levam a incorreta integração, e consequentemente a geração de conflitos ou outros tipos de problemas. Portanto, com a coleta dessas informações torna-se possível mitigá-los, aumentando o nível de produtividade e assertividades dos desenvolvedores.

REFERÊNCIAS

AHMAD, A.; BABAR, M. A. Software architectures for robotic systems: a systematic mapping study. **Journal of Systems and Software**, [S.l.], v. 122, p. 16–39, 2016.

ALE EBRAHIM, N.; AHMED, S.; TAHA, Z. Virtual teams: a literature review. **Australian journal of basic and applied sciences**, [S.l.], v. 3, n. 3, p. 2653–2669, 2009.

ANDRES, H. P. A comparison of face-to-face and virtual software development teams. **Team Performance Management: An International Journal**, [S.l.], 2002.

ARMINO, D. B. **PACCS uma ferramenta para detecção proativa e resolução colaborativa de conflitos de código fonte**. 2016. 94 p. Mestrado em Computação Aplicada — Universidade do Vale do Rio dos Sinos, São Leopoldo, 2016.

BIEHL, J. T.; CZERWINSKI, M.; SMITH, G.; ROBERTSON, G. G. FASTDash: a visual dashboard for fostering awareness in software teams. In: SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2007. **Proceedings...** [S.l.: s.n.], 2007. p. 1313–1322.

BISCHOFF, V.; FARIAS, K.; GONÇALES, L. J.; BARBOSA, J. L. V. Integration of feature models: a systematic mapping study. **Information and Software Technology**, [S.l.], v. 105, p. 209–225, 2019.

CALDIERA, V. R. B.-G.; ROMBACH, H. D. Goal question metric paradigm. **Encyclopedia of software engineering**, [S.l.], v. 1, n. 528-532, p. 6, 1994.

CARBONERA, C. E.; FARIAS, K.; BISCHOFF, V. Software development effort estimation: a systematic mapping study. **IET Software**, [S.l.], v. 14, n. 4, p. 328–344, 2020.

CARBONERA, C.; FARIAS, K.; GRAEFF, C.; SILVA, R. Software Merge: a two-decade systematic mapping study. In: XXXVII BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING, 2023. **Proceedings...** [S.l.: s.n.], 2023. p. 99–108.

CATALDO, M.; WAGSTROM, P. A.; HERBSLEB, J. D.; CARLEY, K. M. Identification of coordination requirements: implications for the design of collaboration and awareness tools. In: COMPUTER SUPPORTED COOPERATIVE WORK, 2006., 2006. **Proceedings...** [S.l.: s.n.], 2006. p. 353–362.

COSTA, C.; MURTA, L. Version control in distributed software development: a systematic mapping study. In: IEEE 8TH INTERNATIONAL CONFERENCE ON GLOBAL SOFTWARE ENGINEERING, 2013., 2013. **Anais...** [S.l.: s.n.], 2013. p. 90–99.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. **MIS quarterly**, [S.l.], p. 319–340, 1989.

DEWAN, P.; HEGDE, R. Semi-synchronous conflict detection and resolution in asynchronous software development. In: **ECSCW 2007**. [S.l.]: Springer, 2007. p. 159–178.

DOURISH, P.; BELLOTTI, V. Awareness and coordination in shared workspaces. In: ACM CONFERENCE ON COMPUTER-SUPPORTED COOPERATIVE WORK, 1992., 1992. **Proceedings...** [S.l.: s.n.], 1992. p. 107–114.

DULLEMOND, K.; GAMEREN, B. van; SOLINGEN, R. van. Collaboration spaces for virtual software teams. **IEEE Software**, [S.l.], v. 31, n. 6, p. 47–53, 2014.

ECLIPSE. **Eclipse , `https://www.eclipse.org/`**, accessed: 01.2024. 2024.

FANG, C.; LIU, Z.; SHI, Y.; HUANG, J.; SHI, Q. Functional code clone detection with syntax and semantics fusion learning. In: ACM SIGSOFT INTERNATIONAL SYMPOSIUM ON SOFTWARE TESTING AND ANALYSIS, 29., 2020. **Proceedings...** [S.l.: s.n.], 2020. p. 516–527.

FARIAS, K.; GARCIA, A.; LUCENA, C. Effects of stability on model composition effort: an exploratory study. **Software & Systems Modeling**, [S.l.], v. 13, p. 1473–1494, 2014.

FARIAS, K.; GARCIA, A.; WHITTLE, J.; FLACH GARCIA CHAVEZ, C. von; LUCENA, C. Evaluating the effort of composing design models: a controlled experiment. **Software & Systems Modeling**, [S.l.], v. 14, p. 1349–1365, 2015.

FERNÁNDEZ-SÁEZ, A. M.; GENERO, M.; CHAUDRON, M. R. Empirical studies concerning the maintenance of UML diagrams and their use in the maintenance of code: a systematic mapping study. **Information and Software Technology**, [S.l.], v. 55, n. 7, p. 1119–1142, 2013.

FIREPAD. **Firepad, `https://firepad.io/`**, accessed: 01.2024. 2024.

FLOOBITS. **Floobits, `https://github.com/floobits/floobits-sublime`**, accessed: 01.2024. 2024.

GITHUB. **GitHub, `https://github.com/`**, accessed: 01.2024. 2024.

GONÇALES, L.; FARIAS, K.; SCHOLL, M.; OLIVEIRA, T. C.; VERONEZ, M. Model Comparison: a systematic mapping study. In: SEKE, 2015. **Anais...** [S.l.: s.n.], 2015. p. 546–551.

GONÇALES, L. J.; FARIAS, K.; SCHOLL, M.; ROBERTO VERONEZ, M.; OLIVEIRA, T. C. de. Comparison of design models: a systematic mapping study. **International Journal of Software Engineering and Knowledge Engineering**, [S.l.], v. 25, n. 09n10, p. 1765–1769, 2015.

HANAM, Q.; MESBAH, A.; HOLMES, R. Aiding code change understanding with semantic change impact analysis. In: IEEE INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE AND EVOLUTION (ICSME), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p. 202–212.

HYMAN, R. Quasi-experimentation: design and analysis issues for field settings (book). **Journal of Personality Assessment**, [S.l.], v. 46, n. 1, p. 96–97, 1982.

JAVA. **Java, `https://www.java.com/`**, accessed: 01.2024. 2024.

JORGENSEN, M. Practical guidelines for expert-judgment-based software effort estimation. **IEEE software**, [S.l.], v. 22, n. 3, p. 57–63, 2005.

KEELE, S. et al. **Guidelines for performing systematic literature reviews in software engineering**. [S.l.]: Technical report, ver. 2.3 ebse technical report. ebse, 2007.

KITCHENHAM, B.; BRERETON, O. P.; BUDGEN, D.; TURNER, M.; BAILEY, J.; LINKMAN, S. Systematic literature reviews in software engineering–a systematic literature review. **Information and software technology**, [S.l.], v. 51, n. 1, p. 7–15, 2009.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. , [S.l.], 2007.

KUDRJAVETS, G.; KUMAR, A.; NAGAPPAN, N.; RASTOGI, A. Mining Code Review Data to Understand Waiting Times Between Acceptance and Merging: an empirical analysis. **arXiv preprint arXiv:2203.05048**, [S.l.], 2022.

MENS, T. A state-of-the-art survey on software merging. **IEEE transactions on software engineering**, [S.l.], v. 28, n. 5, p. 449–462, 2002.

MERCURIALSCM. **MercurialSCM, `https://www.mercurial-scm.org/`**, accessadd: 01.2024. 2024.

MITCHELL, M.; JOLLEY, J. **Research design explained.** [S.l.]: Holt, Rinehart & Winston Inc, 1988.

OCHOA, L.; DEGUEULE, T.; FALLERI, J.-R.; VINJU, J. Breaking bad? Semantic versioning and impact of breaking changes in Maven Central: an external and differentiated replication study. **Empirical Software Engineering**, [S.l.], v. 27, n. 3, p. 61, 2022.

PAGE, M.; MCKENZIE, J.; BOSSUYT, P.; BOUTRON, I.; HOFFMANN, T.; MULROW, C. et al. Preferred reporting items for systematic reviews and meta-analyses: the prisma statement. **Syst Rev**, [S.l.], v. 10, 2021.

PERRY, D. E.; SIY, H. P.; VOTTA, L. G. Parallel changes in large-scale software development: an observational case study. **ACM Transactions on Software Engineering and Methodology (TOSEM)**, [S.l.], v. 10, n. 3, p. 308–337, 2001.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. In: INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING (EASE) 12, 12., 2008. **Anais...** [S.l.: s.n.], 2008. p. 1–10.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: an update. **Information and software technology**, [S.l.], v. 64, p. 1–18, 2015.

PETRUCCELLI, J. D.; NANDRAM, B.; CHEN, M. **Doing it with SAS**: a supplement to applied statistics for engineers and scientists. [S.l.]: Prentice Hall, 1999.

POSTGRESQL. **PostgreSQL, `https://www.postgresql.org/`**, accessed: 01.2024. 2024.

REMOTECOLLAB. **RemoteCollab, `https://teamremote.github.io/remote-sublime/`**, accessed: 01.2024. 2024.

RODRÍGUEZ, P.; HAGHIGHATKHAH, A.; LWAKATARE, L. E.; TEPPOLA, S.; SUOMALAINEN, T.; ESKELI, J.; KARVONEN, T.; KUVAJA, P.; VERNER, J. M.; OIVO, M. Continuous deployment of software intensive products and services: a systematic mapping study. **Journal of Systems and Software**, [S.l.], v. 123, p. 263–291, 2017.

SAROS. **Saros, `https://www.saros-project.org/`**, accessed: 01.2024. 2024.

SEGALOTTO, M. **ARNI**: an eeg-based model to measure program comprehension. 2018. 167 p. Mestrado em Computação Aplicada — Universidade do Vale do Rio dos Sinos, São Leopoldo, 2018.

SIEBDRAT, F.; HOEGL, M.; ERNST, H. How to manage virtual teams. **MIT Sloan Management Review**, [S.l.], v. 50, n. 4, p. 63, 2009.

SMIL PRUTCHI, E.; SOUZA CAMPOS JUNIOR, H. de; GRESTA PAULINO MURTA, L. How the adoption of feature toggles correlates with branch merges and defects in open-source projects? **arXiv e-prints**, [S.l.], p. arXiv–2007, 2020.

SOMANY. **SoManyConflicts, `https://marketplace.visualstudio.com/items?itemname=symbolk.somanyconflicts`**, accessed: 01.2024. 2024.

SUBVERSION. **Subversion, `https://subversion.apache.org/download.cgi`**, accessed: 01.2024. 2024.

SUNG, C.; LAHIRI, S. K.; KAUFMAN, M.; CHOUDHURY, P.; WANG, C. Towards understanding and fixing upstream merge induced conflicts in divergent forks: an industrial case study. In: ACM/IEEE 42ND INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING: SOFTWARE ENGINEERING IN PRACTICE, 2020. **Proceedings...** [S.l.: s.n.], 2020. p. 172–181.

SVNSERVER. **SVNServer, `https://www.visualsvn.com/server/download/`**, accessed: 01.2024. 2024.

TFS. **Team Foundation Server, `https://visualstudio.microsoft.com/downloads/`**, accessed: 01.2024. 2024.

TORTOISESVN. **TortoiseSVN, `https://tortoisesvn.net/downloads.html`**, accessed: 01.2024. 2024.

TROCHIM, W. M.; DONNELLY, J. P. **Research methods knowledge base**. [S.l.]: Atomic Dog Pub. Macmillan Publishing Company, New York, 2001. v. 2.

VSCODE. **Visual Studio Live Share, `https://visualstudio.microsoft.com/pt-br/services/live-share/`** accessed: 01.2024. 2024.

WEISS, C. H. The Experimenting Society. **Validity and Social Experimentation: Donald Campbell s Legacy**, [S.l.], v. 1, p. 283, 2000.

WLOKA, J.; RYDER, B.; TIP, F.; REN, X. Safe-commit analysis to facilitate team software development. In: IEEE 31ST INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 2009., 2009. **Anais...** [S.l.: s.n.], 2009. p. 507–517.

WOHLIN, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: OF THE 18TH INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING, 2014. **Proceedings...** [S.l.: s.n.], 2014. p. 1–10.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. **Experimentation in software engineering**. [S.l.]: Springer Science & Business Media, 2012.

APÊNDICE A – LISTA DE ESTUDOS PRIMÁRIOS

Abaixo estão relacionados os estudos primários utilizados por essa pesquisa.

[P01] Brun, Y., Reid Holmes, M. D. E., Notkin, D. (2010). Speculative identification of merge conflicts and non-conflicts. University of Washington, Seattle.

[P02] Phillips, S., Sillito, J., Walker, R. (2011, May). Branching and merging: an investigation into current version control practices. In Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering (pp. 9-15).

[P03] Ghiotto, G., Murta, L., Barros, M., Van Der Hoek, A. (2018). On the nature of merge conflicts: a study of 2,731 open source java projects hosted by github. IEEE Transactions on Software Engineering, 46(8), 892-915.

[P04] Da Silva, L., Borba, P., Mahmood, W., Berger, T., Moisakis, J. (2020, September). Detecting semantic conflicts via automated behavior change detection. In 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 174-184). IEEE.

[P05] Reiter, T., Altmanninger, K., Bergmayr, A., Schwinger, W., Kotsis, G. (2007). Models in conflict-detection of semantic conflicts in model-based development. MDEIS@ ICEIS, 7, 29-40.

[P06] Dias, M., Polito, G., Cassou, D., Ducasse, S. (2015, July). Deltaimpactfinder: Assessing semantic merge conflicts with dependency analysis. In Proceedings of the International Workshop on Smalltalk Technologies (pp. 1-6).

[P07] Apel, S., Liebig, J., Lengauer, C., Kästner, C., Cook, W. R. (2010). Semistructured Merge in Revision Control Systems. In VaMoS (pp. 13-19).

[P08] Apel, S., Liebig, J., Brandl, B., Lengauer, C., Kästner, C. (2011, September). Semistructured merge: rethinking merge in revision control systems. In Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering (pp. 190-200).

[P09] Vale, G., Hunsen, C., Figueiredo, E., Apel, S. (2021). Challenges of resolving merge conflicts: A mining and survey study. IEEE Transactions on Software Engineering, 48(12), 4964-4985.

[P10] Yuzuki, R., Hata, H., Matsumoto, K. (2015, March). How we resolve conflict: an empirical study of method-level conflict resolution. In 2015 IEEE 1st International Workshop on Software Analytics (SWAN) (pp. 21-24). IEEE.

[P11] Menezes, J. W., Trindade, B., Pimentel, J. F., Plastino, A., Murta, L., Costa, C. (2021). Attributes that may raise the occurrence of merge conflicts. Journal of Software Engineering Research and Development, 9, 14-1.

[P12] Brun, Y., Holmes, R., Ernst, M. D., Notkin, D. (2013). Early detection of collaboration conflicts and risks. IEEE Transactions on Software Engineering, 39(10), 1358-1375.

[P13] Costa, C., Figueirêdo, J. J., Murta, L. (2014). Collaborative Merge in Distributed Software Development: Who Should Participate?. In SEKE (pp. 268-273).

[P14] Costa, C., Figueiredo, J. J., Ghiotto, G., Murta, L. (2014). Characterizing the Problem of Developers' Assignment for Merging Branches. International Journal of Software Engineering and Knowledge Engineering, 24(10), 1489-1508.

[P15] Ahmed, I., Brindescu, C., Mannan, U. A., Jensen, C., Sarma, A. (2017, November). An empirical examination of the relationship between code smells and merge conflicts. In 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (pp. 58-67). IEEE.

[P16] Mashiyat, A. S. Is Our Merging Right? Towards More Reliable Merging Decision.

[P17] Guimarães, M. L., Silva, A. R. (2012, June). Improving early detection of software merge conflicts. In 2012 34th International Conference on Software Engineering (ICSE) (pp. 342-352). IEEE.

[P18] McKee, S., Nelson, N., Sarma, A., Dig, D. (2017, September). Software practitioner perspectives on merge conflicts and resolutions. In 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 467-478). IEEE.

[P19] Leßenich, O., Siegmund, J., Apel, S., Kästner, C., Hunsen, C. (2018). Indicators for merge conflicts in the wild: survey and empirical study. Automated Software Engineering, 25, 279-313.

[P20] Zolkifli, N. N., Ngah, A., Deraman, A. (2018). Version control system: A review. Procedia Computer Science, 135, 408-415.

[P21] Zimmermann, T. (2007, May). Mining workspace updates in CVS. In Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007) (pp. 11-11). IEEE.

[P22] Dig, D., Manzoor, K., Johnson, R., Nguyen, T. N. (2007, May). Refactoring-aware configuration management for object-oriented programs. In 29th International Conference on Software Engineering (ICSE'07) (pp. 427-436). IEEE.

[P23] Leßenich, O., Apel, S., Lengauer, C. (2015). Balancing precision and performance in structured merge. Automated Software Engineering, 22(3), 367-397.

[P24] Candrlic, S., Pavlic, M., Asenbrener, M. Code merging analysis for different procedure types.

[P25] Goeminne, M., Mens, T. (2013). A comparison of identity merge algorithms for software repositories. Science of Computer Programming, 78(8), 971-986.

[P26] Leßenich, O., Apel, S., Kästner, C., Seibt, G., Siegmund, J. (2017, October). Renaming and shifted code in structured merging: Looking ahead for precision and performance. In 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 543-553). IEEE.

[P27] Just, S., Herzig, K., Czerwonka, J., Murphy, B. (2016, October). Switching to Git: the good, the bad, and the ugly. In 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE) (pp. 400-411). IEEE.

[P28] Nishimura, Y., Maruyama, K. (2016, March). Supporting merge conflict resolution by using fine-grained code change history. In 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER) (Vol. 1, pp. 661-664). IEEE.

[P29] Seibt, G., Heck, F., Cavalcanti, G., Borba, P., Apel, S. (2021). Leveraging structure in software merge: An empirical study. IEEE Transactions on Software Engineering, 48(11), 4590-4610.

[P30] Brindescu, C., Ramirez, Y., Sarma, A., Jensen, C. (2020, September). Lifting the curtain on merge conflict resolution: A sensemaking perspective. In 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 534-545). IEEE.

[P31] Mahmood, W., Chagama, M., Berger, T., Hebig, R. (2020, February). Causes of merge conflicts: A case study of elasticsearch. In Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems (pp. 1-9).

[P32] Alnor Adam Khleel, N., Nehéz, K. (2020). Merging problems in modern version control systems. MULTIDISZCIPLINÁRIS TUDOMÁNYOK: A MISKOLCI EGYETEM KÖZLEMÉNYE, 10(3), 365-376.

[P33] Ji, T., Chen, L., Yi, X., Mao, X. (2020, October). Understanding merge conflicts and resolutions in git rebases. In 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE) (pp. 70-80). IEEE.

[P34] Costa, C., Figueirêdo, J., Pimentel, J. F., Sarma, A., Murta, L. (2019). Recommending participants for collaborative merge sessions. IEEE Transactions on Software Engineering, 47(6), 1198-1210.

[P35] Nelson, N., Brindescu, C., McKee, S., Sarma, A., Dig, D. (2019). The life-cycle of merge conflicts: processes, barriers, and strategies. Empirical Software Engineering, 24, 2863-2906.

[P36] Mahmoudi, M., Nadi, S., Tsantalis, N. (2019, February). Are refactorings to blame? an empirical study of refactorings in merge conflicts. In 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 151-162). IEEE.

[P37] Golzadeh, M., Decan, A., Mens, T. (2019, November). On the Effect of Discussions on Pull Request Decisions. In BENEVOL.

[P38] Shen, B., Zhang, W., Zhao, H., Liang, G., Jin, Z., Wang, Q. (2019). IntelliMerge: a refactoring-aware software merging technique. Proceedings of the ACM on Programming Languages, 3(OOPSLA), 1-28.

[P39] Accioly, P., Borba, P., Cavalcanti, G. (2018). Understanding semi-structured merge conflict characteristics in open-source java projects. Empirical Software Engineering, 23, 2051-2085.

[P40] Da Silva, L., Borba, P., Pires, A. (2022). Build conflicts in the wild. Journal of Software: Evolution and Process, 34(4), e2441.

[P41] Larsén, S., Falleri, J. R., Baudry, B., Monperrus, M. (2022). Spork: Structured Merge for Java with Formatting Preservation. IEEE Transactions on Software Engineering, 49(1), 64-83.

[P42] Pan, R., Le, V., Nagappan, N., Gulwani, S., Lahiri, S., Kaufman, M. (2021, May). Can program synthesis be used to learn merge conflict resolutions? an empirical analysis. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (pp. 785-796). IEEE.

[P43] Costa, C., Menezes, J., Trindade, B., Santos, R. (2021, September). Factors that affect merge conflicts: A software developers' perspective. In Proceedings of the XXXV Brazilian Symposium on Software Engineering (pp. 233-242).

[P44] Apel, S., Leßenich, O., Lengauer, C. (2012, September). Structured merge with auto-tuning: balancing precision and performance. In Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (pp. 120-129).

[P45] Costa, C., Figueiredo, J., Sarma, A., Murta, L. (2016, November). TIPMerge: recommending developers for merging branches. In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (pp. 998-1002).

[P46] Nguyen, H. L., Ignat, C. L. (2018). An analysis of merge conflicts and resolutions in git-based open source projects. Computer Supported Cooperative Work (CSCW), 27, 741-765.

[P47] Sousa, M., Dillig, I., Lahiri, S. K. (2018). Verified three-way program merge. Proceedings of the ACM on Programming Languages, 2(OOPSLA), 1-29.

[P48] Shen, B., Zhang, W., Yu, A., Shi, Y., Zhao, H., Jin, Z. (2021, November). SoManyConflicts: resolve many merge conflicts interactively and systematically. In 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 1291-1295). IEEE.

[P49] Saha, D., Dhoolia, P., Garg, M., Vaibhav, V. (2016, February). Delta refactoring for merge conflict avoidance. In Proceedings of the 9th India Software Engineering Conference (pp. 26-36).

[P50] Zou, W., Zhang, W., Xia, X., Holmes, R., Chen, Z. (2019, July). Branch use in practice: A large-scale empirical study of 2,923 projects on github. In 2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS) (pp. 306-317). IEEE.

[P51] Tavares, A. T., Borba, P., Cavalcanti, G., Soares, S. (2019, November). Semistructured merge in JavaScript systems. In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 1014-1025). IEEE.

[P52] Cavalcanti, G., Borba, P., Accioly, P. (2017). Evaluating and improving semistructured merge. Proceedings of the ACM on Programming Languages, 1(OOPSLA), 1-27.

[P53] Menezes, J. W., Trindade, B., Pimentel, J. F., Moura, T., Plastino, A., Murta, L., Costa, C. (2020, October). What causes merge conflicts?. In Proceedings of the XXXIV Brazilian Symposium on Software Engineering (pp. 203-212).

[P54] Brindescu, C., Ahmed, I., Jensen, C., Sarma, A. (2020). An empirical investigation into merge conflicts and their effect on software quality. Empirical Software Engineering, 25, 562-590.

[P55] Zhang, X., Chen, Y., Gu, Y., Zou, W., Xie, X., Jia, X., Xuan, J. (2018, September). How do multiple pull requests change the same code: A study of competing pull requests in github. In 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 228-239). IEEE.

[P56] Wuensche, T., Andrzejak, A., Schwedes, S. (2020, October). Detecting higher-order merge conflicts in large software projects. In 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST) (pp. 353-363). IEEE.

[P57] Hattori, L., Lanza, M., D'Ambros, M. (2012, August). A qualitative user study on preemptive conflict detection. In 2012 IEEE Seventh International Conference on Global Software Engineering (pp. 159-163). IEEE.

[P58] Cavalcanti, G., Borba, P., Seibt, G., Apel, S. (2019, November). The impact of structure on software merging: semistructured versus structured merge. In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 1002-1013). IEEE.

[P59] Bouras, Z. E., Maouche, M. (2017). Software architectures evolution based merging. Informatica, 41(1).

[P60] Dig, D., Manzoor, K., Johnson, R. E., Nguyen, T. N. (2008). Effective software merging in the presence of object-oriented refactorings. IEEE Transactions on Software Engineering, 34(3), 321-335.

[P61] Ma, P., Xu, D., Zhang, X., Xuan, J. (2019). Changes are similar: Measuring similarity of pull requests that change the same code in github. In Software Engineering and Methodology for Emerging Domains: 16th National Conference, NASAC 2017, Harbin, China, November 4–5, 2017, and 17th National Conference, NASAC 2018, Shenzhen, China, November 23–25, 2018, Revised Selected Papers 16 (pp. 115-128). Springer Singapore.

[P62] Xing, X., Maruyama, K. (2019, February). Automatic software merging using automated program repair. In 2019 IEEE 1st International Workshop on Intelligent Bug Fixing (IBF) (pp. 11-16). IEEE.

[P63] Cavalcanti, G., Accioly, P., Borba, P. (2015, October). Assessing semistructured merge in version control systems: A replicated experiment. In 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (pp. 1-10). IEEE.

[P64] Sarma, A., Redmiles, D. F., Van Der Hoek, A. (2011). Palantir: Early detection of development conflicts arising from parallel code changes. IEEE Transactions on Software Engineering, 38(4), 889-908.

[P65] Dinella, E., Mytkowicz, T., Svyatkovskiy, A., Bird, C., Naik, M., Lahiri, S. (2022). Deepmerge: Learning to merge programs. IEEE Transactions on Software Engineering.

[P66] Shen, B., Gulzar, M. A., He, F., Meng, N. (2023). A Characterization Study of Merge Conflicts in Java Projects. ACM Transactions on Software Engineering and Methodology, 32(2), 1-28.

[P67] Shen, Bowen Xiao, Cihan Meng, Na He, Fei. (2021). Automatic Detection and Resolution of Software Merge Conflicts: Are We There Yet?.

[P68] Svyatkovskiy, A., Fakhoury, S., Ghorbani, N., Mytkowicz, T., Dinella, E., Bird, C., Lahiri, S. K. (2022, November). Program merge conflict resolution via neural transformers. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 822-833).

[P69] Zhang, J., Mytkowicz, T., Kaufman, M., Piskac, R., Lahiri, S. K. (2022, July). Using pre-trained language models to resolve textual and semantic merge conflicts (experience paper). In Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 77-88).

[P70] Oliveira, Andre Neves, Vânia Plastino, Alexandre Bibiano, Ana Garcia, Alessandro Murta, Leonardo. (2023). Do code refactorings influence the merge effort?.

APÊNDICE B – QUESTIONÁRIO DE CARACTERÍSTICAS

Questionário de Características

Data: ___/___/_____	Hora Início: ___:___	Hora Fim: ___:___

Qual o seu nome (opcional):	
Qual a sua idade (opcional):	(anos)
Sexo:	( ) Masculino ( ) Feminino

Em qual empresa você trabalha atualmente ?	
Qual é a sua Formação Acadêmica ?	( ) Ciência da Computação ( ) Engenharia da Computação ( ) Sistema de Informação ( ) Análise de Sistemas ( ) Outro (descreva abaixo):
Qual é o seu maior grau de escolaridade?	( ) Técnico ( ) Graduação ( ) Especialização ( ) Mestrado ( ) Doutorado
Por quanto tempo você estudou (ou tem estudado) em alguma universidade ?	( ) menos de 2 anos ( ) de 2 a 4 anos ( ) de 5 a 6 anos ( ) de 7 a 8 anos ( ) Mais de 8 anos
Qual é o seu cargo/posição atual?	( ) Desenvolvedor ( ) Analista de Sistemas/Processos ( ) Arquiteto de Software ( ) Gerente de Projetos ( ) Product Owner (PO) ( ) Outro (descreva abaixo):
Por quanto tempo você tem exercido esse cargo/posição?	( ) menos de 2 anos ( ) de 2 a 4 anos ( ) de 5 a 6 anos ( ) de 7 a 8 anos ( ) Mais de 8 anos
Quanto tempo de experiência com desenvolvimento de software?	( ) menos de 2 anos ( ) de 2 a 4 anos ( ) de 5 a 6 anos ( ) de 7 a 8 anos ( ) Mais de 8 anos
Qual é a sua modalidade de trabalho?	( ) 100% Presencial ( ) 50% presencial e 50% remoto ( ) 30% presencial e 70% remoto ( ) Totalmente Remoto ( ) Outro (descreva abaixo):

Fonte: Criado pelo Autor.
Figura 34: Questionário de Características.

APÊNDICE C – QUESTIONÁRIO DE IMPRESSÃO

Questionário de Impressões

(Após a execução do experimento)

Qual o seu nome (opcional):	
Qual o seu e-mail (opcional):	

1) A utilização de ferramentas para versionamento de arquivos fontes em empresas de software (centralizado ou distribuído) é universal ?

( ) Concordo Totalmente ( ) Concordo Parcialmente ( )Neutro ( )Discordo Parcialmente ( ) Discordo Totalmente

2) A utilização de ferramentas para versionamento ajuda na resolução de conflitos durante as discussões sobre como integrar arquivos fontes corretamente ?

( ) Concordo Totalmente ( ) Concordo Parcialmente ( )Neutro ( )Discordo Parcialmente ( ) Discordo Totalmente

3) A detecção e a propagação **antecipada de conflitos** na view ProMerge **auxiliou na resolução de conflitos diretos e indiretos** ?

( ) Concordo Totalmente ( ) Concordo Parcialmente ( )Neutro ( )Discordo Parcialmente ( ) Discordo Totalmente

4) A detecção e a propagação antecipada de conflitos **ajudou a identificar os conflitos antes de submeter os arquivos fontes alterados ao repositório ?**

( ) Concordo Totalmente ( ) Concordo Parcialmente ( )Neutro ( )Discordo Parcialmente ( ) Discordo Totalmente

5) A opção de **resolução automática dos conflitos ajudou a não depender dos desenvolvedores** para o entendimento do contexto dos conflitos ?

( ) Concordo Totalmente ( ) Concordo Parcialmente ( )Neutro ( )Discordo Parcialmente ( ) Discordo Totalmente

6) A checagem e a validação **pós-commit auxiliou na detecção** dos conflitos diretos e indiretos ?

( ) Concordo Totalmente ( ) Concordo Parcialmente ( )Neutro ( )Discordo Parcialmente ( ) Discordo Totalmente

7) A utilização da abordagem proposta pelo ProMerge **foi intuitiva e de fácil** interpretação ?

( ) Concordo Totalmente ( ) Concordo Parcialmente ( )Neutro ( )Discordo Parcialmente ( ) Discordo Totalmente

8) A disponibilização das informações **facilitou o entendimento e a utilização** da abordagem proposta pelo ProMerge ?

( ) Concordo Totalmente ( ) Concordo Parcialmente ( )Neutro ( )Discordo Parcialmente ( ) Discordo Totalmente

9) A utilização da abordagem proposta pelo ProMerge em **relação a não utilização de nenhuma abordagem proativa auxiliou** na resolução dos conflitos mais facilmente ?

( ) Concordo Totalmente ( ) Concordo Parcialmente ( )Neutro ( )Discordo Parcialmente ( ) Discordo Totalmente

10) A utilização da abordagem proposta pelo ProMerge **atende plenamente as necessidades** de validação assegurando a integridade do projeto pós-commit ?

( ) Concordo Totalmente ( ) Concordo Parcialmente ( )Neutro ( )Discordo Parcialmente ( ) Discordo Totalmente

Fonte: Criado pelo Autor.

Figura 35: Questionário de Impressão – Página 1/2.

11) A detecção proativa de conflitos **tende a ser mais utilizada** nas etapas do pós-commit?

( ) Concordo Totalmente ( ) Concordo Parcialmente ( )Neutro ( )Discordo Parcialmente ( ) Discordo Totalmente

12) O **custo de promover o correto entendimento** dos conceitos sobre conflitos entre diferentes pessoas com diferentes níveis de formação/experiência é elevado?

( ) Concordo Totalmente ( ) Concordo Parcialmente ( ) Neutro ( ) Discordo Parcialmente ( ) Discordo Totalmente

13) Existe um ganho de **produtividade** entre as equipes com a detecção proativa de conflitos?

( ) Concordo Totalmente ( ) Concordo Parcialmente ( ) Neutro ( ) Discordo Parcialmente ( ) Discordo Totalmente

14) A deteção **proativa de conflitos será um recurso futuro muito importante** em ferramentas para versionamento de arquivos fontes?

( ) Concordo Totalmente ( ) Concordo Parcialmente ( ) Neutro ( ) Discordo Parcialmente ( ) Discordo Totalmente

15) O ganho de produtividade com a **correta integração** de arquivos fontes tende a ser mais **vantajoso** pois se trabalha de maneira proativa e não reativa?

( ) Concordo Totalmente ( ) Concordo Parcialmente ( ) Neutro ( ) Discordo Parcialmente ( ) Discordo Totalmente

16) É importante a **detecção proativa de conflitos** durante a integração dos arquivos fontes com o repositório?

( ) Concordo Totalmente ( ) Concordo Parcialmente ( ) Neutro ( ) Discordo Parcialmente ( ) Discordo Totalmente

Fonte: Criado pelo Autor.

Figura 36: Questionário de Impressão – Página 2/2.

APÊNDICE D – EXPERIMENTO CONTROLADO – CENÁRIO 1

Esse Apêndice detalha as atividades executadas pelo experimento controlado dessa pesquisa. Seguindo os padrões do próprio Eclipse (ECLIPSE, 2024), as atividades estão divididas em dois pacotes: **experimento01** e **experimento02**. Os dois pacotes de arquivos fontes são compostos por classes e funcionalidades distintas, porém com atividades semelhantes e com igual nível de complexidade e que podem ser executadas independentemente do apoio de qualquer tipo de ferramenta externa.

- **Atividade 1.** Supondo que você é um dos desenvolvedores seniores responsáveis pelo sistema BestSell, que é um software destinado para venda de artigos esportivos. Atualmente, a listagem de produtos disponíveis para venda está totalmente liberada. Todos os níveis de usuários acessam todas as categorias de produtos, sem restrição alguma. Os níveis de usuários são: **Vendedor, Gerente, Repositor e Conferencista** e as categorias de produtos são: **A, B, C e D**. Após uma cuidadosa avaliação técnica, ficou estabelecido que os produtos do tipo **B e D** devem ter o acesso mais restrito. Após a realização dessa melhoria, apenas os usuários do tipo **Vendedor e Gerente** podem ter acesso aos produtos dos tipos **B e D**. Para isso, vá no pacote **PrimeiroExperimento**, na classe **Produtos**, no método **listarProdutos** e adicione o parâmetro **nivel** do tipo **string** que deverá permitir o acesso ou não aos produtos do tipo **B e D**. Por fim, salve e commit as alterações realizadas junto ao repositório.

- **Atividade 2.** Supondo que você é um dos desenvolvedores seniores responsáveis pelo sistema BestSell. Por padrão do sistema, os funcionários dos tipos **Vendedor e Gerente** possuem além do salário mensal, um percentual de acréscimo à título de bonificação. Atualmente, o valor final do salário dos funcionários ainda não é calculado automaticamente e isso deverá ser corrigido. Após a realização dessa melhoria, o valor final do salário dos funcionários deverá ser calculado automaticamente já acrescido do percentual de bonificação. Para isso, vá no pacote **PrimeiroExperimento** e crie a classe **FuncionarioSalario** contendo um método construtor SEM parâmetros e o método **CalcularSalario** que deverá calcular o valor dos salários dos funcionários. O método **CalcularSalario** deverá receber como parâmetros o **valor do salário e o percentual de bonificação (ambos double) e deverá retornar o valor do salário já acrescido do percentual de bonificação**. Após, **salve e commit** as alterações realizadas junto ao repositório. Em seguida, vá na classe **Funcionario** no método construtor que faz a inicialização das propriedades e insira uma chamada para o método **CalcularSalario** passando os parâmetros necessários. Por fim, **salve e commit** novamente as alterações realizadas junto ao repositório.

- **Atividade 3.** Supondo que você é um dos desenvolvedores seniores responsáveis pelo sistema BestSell. Por padrão do sistema, **qualquer tipo de listagem deverá mostrar**

os campos de valores com APENAS duas casas decimais. Os analistas responsáveis pelo sistema perceberam que os produtos habilitados para venda eram consultados, o valor desses produtos eram visualizados com mais de duas casas decimais e esse problema deverá ser corrigido. Após a realização dessa melhoria, o preço final dos produtos deverá ser apresentado com apenas duas casas decimais. Para isso, vá no pacote **PrimeiroExperimento**, na classe **Util**, no método **arredondarValores** e insira o parâmetro **valorArredondar** do tipo de dados **double**. O método **arredondarValores** deverá retornar o valor recebido como parâmetro devidamente arredondado com duas casas decimais. Em seguida vá na classe **Produtos** no método **listarProdutos** e aplique o método **arredondarValores** três vezes para realizar o arredondamento dos valores dos produtos para duas casas decimais. Por fim, **salve e commit** as alterações realizadas junto ao repositório.

- **Atividade 4.** Supondo que você é um dos desenvolvedores trainee da equipe do sistema BestSell, é de seu conhecimento que a listagem dos pedidos de venda poderá ser consultada através de arquivos em formato **".csv"**. Os analistas responsáveis pelo sistema verificaram através dos testes realizados que os dados estão incompletos, faltando vários campos. Após uma avaliação, verificou-se que os campos **DataPedido e ValorPedido, ValorFrete, ValorICMS e ValorComissao** não constavam nos arquivos exportados, e esse problema deverá ser corrigido imediatamente. Para isso, vá no pacote **PrimeiroExperimento**, na classe **ExportarExcel**, no método **writeCsv** e insira os campos campos **DataPedido e ValorPedido, ValorFrete, ValorICMS e ValorComissao** para serem exportados junto com os demais campos. Por fim, **apenas salve as alterações realizadas, SEM realizar o commit** nos arquivos fontes alterados.

- **Atividade 5.** Supondo que você é um dos desenvolvedores trainee da equipe do sistema BestSell. Por padrão do sistema, deverá constar obrigatoriamente na declaração do(s) método(s) construtor(es) de todas as classes a mensagem **Inicializando o método construtor da classe** a título de log de monitoramento. Os analistas responsáveis pelo sistema perceberam que quando as telas de cadastros de **PedidosVenda e ItensPedidosVenda** eram acessadas as mensagens de log não eram visualizadas. Foi constatado pelos analistas que os métodos construtores dessas duas classes não existia as mensagens informativas declaradas e esse problema deverá ser corrigido imediatamente. Para isso, vá no pacote **PrimeiroExperimento**, em todos os métodos construtores das classes **PedidosVenda e ItensPedidosVenda** e insira a mensagem informativa **System.out.print("Inicializando o método construtor da classe");** logo no início da declaração dos métodos construtores. Por fim, **apenas salve as alterações realizadas, SEM a realizar o commit** dos arquivos fontes alterados.

APÊNDICE E – EXPERIMENTO CONTROLADO – CENÁRIO 2

- **Atividade 1.** Supondo que você é um dos desenvolvedores seniores responsáveis pelo sistema BestSell, que é um software destinado para venda de artigos esportivos. Atualmente, a listagem de produtos disponíveis para os pedidos de compra está totalmente liberada. Todos os níveis de usuários acessam todas as categorias de produtos sem restrição alguma. Os níveis de usuários são: **Comprador, Gerente, Supervisor e Repositor** e as categorias de produtos são: **A, B, C e D**. Após uma cuidadosa avaliação técnica, ficou estabelecido que os produtos do tipo **A e C** devem ter o acesso mais restrito. Após a realização dessa melhoria, apenas os usuários do tipo **Comprador e Gerente** poderão ter acesso aos produtos do tipo **A e C**. Para isso, vá no pacote **SegundoExperimento**, na classe **ProdutosCompra** e adicione o parâmetro **nivel** do tipo **string** na declaração do método **listarProdutosCompra** que deverá permitir o acesso ou não aos produtos do tipo **A e C**. Por fim, **salve e commit** as alterações realizadas junto ao repositório.

- **Atividade 2.** Supondo que você é um dos desenvolvedores seniores responsáveis pelo sistema BestSell. Por padrão do sistema, os funcionários dos tipos **Comprador e Gerente** possuem além do salário mensal, um percentual de acréscimo a título de bonificação. Atualmente, o valor final do salário ainda não é calculado automaticamente e isso deverá ser corrigido. Após a realização dessa melhoria, o valor final do salário deverá ser calculado automaticamente já acrescido do percentual de bonificação. Para isso, vá no pacote **SegundoExperimento**, e crie a classe **FuncionarioSalario** contendo um método construtor SEM parâmetros e o método **CalcularSalario** que deverá calcular o valor dos salários dos funcionários. O método **CalcularSalario** deverá receber como parâmetros o **valor do salário e o percentual de bonificação (ambos double) e deverá retornar o valor do salário já acrescido do percentual de bonificação**. Após, **salve e commit** as alterações realizadas junto ao repositório. Em seguida, vá na classe **Funcionario** no método construtor que faz a inicialização das propriedades e insira uma chamada para o método **CalcularSalario** passando os parâmetros necessários. Por fim, **salve e commit** novamente as alterações realizadas junto ao repositório.

- **Atividade 3.** Supondo que você é um dos desenvolvedores seniores responsáveis pelo sistema BestSell. Por padrão do sistema, **qualquer tipo de listagem deverá mostrar os campos de valores com APENAS duas casas decimais**. Os analistas responsáveis pelo sistema perceberam que quando vários produtos habilitados para venda eram consultados, o valor final era visualizado com mais de duas casas decimais e esse problema deverá ser corrigido. Após a realização dessa melhoria, o preço final dos produtos deverá ser apresentado com apenas duas casas decimais. Para isso, vá no pacote **SegundoExperimento**, na classe **Util**, no método **arredondarValores** e insira o parâmetro **valorArredondar** do tipo de dados **double**. O método **arredondarValores** deverá retornar o valor recebido

como parâmetro devidamente arredondado com duas casas decimais. Em seguida vá na classe **ProdutosCompra** no método **listarProdutos** e aplique o método **arredondarValores** três vezes para realizar o arredondamento dos valores dos produtos para duas casas decimais. Por fim, **salve e commit** as alterações realizadas junto ao repositório.

- **Atividade 4.** Supondo que você é um dos desenvolvedores trainee da equipe do sistema BestSell, é de seu conhecimento que a listagem dos pedidos de venda poderá ser consultada através de arquivos em formato **".csv"**. Os analistas responsáveis pelo sistema verificaram através dos testes realizados que os dados estão incompletos, faltando vários campos. Após uma avaliação, verificou-se que os campos **DataPedido e ValorPedido, ValorFrete, ValorICMS e ValorComissao** não constavam nos arquivos exportados, e esse problema deverá ser corrigido imediatamente. Para isso, vá no pacote **SegundoExperimento**, na classe **ExportarExcel**, no método **writeCsv** e insira os campos **DataPedido e ValorPedido, ValorFrete e ValorICMS** para serem exportados junto com os demais campos. Por fim, **apenas salve as alterações realizadas, SEM realizar o commit** nos arquivos fontes alterados.

- **Atividade 5.** Supondo que você é um dos desenvolvedores trainee da equipe do sistema BestSell. Por padrão do sistema, deverá constar obrigatoriamente na declaração do(s) método(s) construtor(es) de todas as classes a mensagem **Inicializando o método construtor da classe** a título de log de monitoramento. Os analistas responsáveis pelo sistema perceberam que quando as telas de cadastros de **PedidoCompra e ItensPedidosCompra** eram acessadas as mensagens de log não eram visualizadas. Foi constatado pelos analistas que os métodos construtores dessas duas classes não existia as mensagens informativas declaradas e esse problema deverá ser corrigido imediatamente. Para isso, vá no pacote **SegundoExperimento**, em todos os métodos construtores das classes **PedidoCompra e ItensPedidosCompra** e insira a mensagem informativa **System.out.print("Inicializando o método construtor da classe");** logo no início da declaração dos métodos construtores. Por fim, **apenas salve as alterações realizadas, SEM a realizar o commit** dos arquivos fontes alterados.

APÊNDICE F – ARTIGOS PUBLICADOS

CARBONERA, C.; FARIAS, K.; GRAEFF, C.,; SILVA, R. (2023, September). **Software Merge: A Two–Decade Systematic Mapping Study.** In Proceedings of the XXXVII Brazilian Symposium on Software Engineering (pp. 99–108).

GRAEFF, C. A.; FARIAS, K.; CARBONERA, C. E. (2023, May). **On the Prediction of Software Merge Conflicts: A Systematic Review and Meta-analysis.** In Proceedings of the XIX Brazilian Symposium on Information Systems (pp. 404–411).