Análise da Relação Entre *Code Smells* e Métricas Qualitativas em Repositórios do *GitHub* de Sistemas de Direção Autônoma

Lorrayne Reis Silva [1]

[1]Bacharelado em Engenharia de Software ´

[2]Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)

[3]Edifício Fernanda, Rua Claudio Manoel, 1.162, Funcionários
30.140-100 — Belo Horizonte — MG — Brazil

lorrayne.silva.1220819@sga.pucminas.br

Abstract. *Autonomous driving systems have gained prominence in the recent industrial market, being pioneers for the performance of autonomous vehicles that preach about driving without human interference. These repositories for simulating are often used in the development and measurement of tests, but the provision of reliability in these simulations is directly related to qualitative factors. Given the scarcity of studies that define the subject, the present work seeks to understand the qualitative relationships presented by this segment, through code smells and quality attributes. From the evaluation carried out in 200 repositories of the Python language, the classification of the main code smells found and the correlation of these with qualitative metrics and associated realeses, it was observed the presence of three main code smells and the predominance of the Halstead metric of effort in the development of these repositories.*

Resumo. *Sistemas de direção autônoma tem ganhado destaque no mercado industrial recente, sendo percursores para atuação de veículos autônomos que fomentam uma condução sem interferência humana. Assim, repositórios para a simulação desses são frequentemente usados no desenvolvimento e aferição de testes, porém a provisão de confiabilidade nestas simulações esta diretamente relacionada a fatores qualitativos. Dado a escassez de estudos que delimitam sobre o assunto, o presente trabalho busca entender as relações qualitativas apresentadas por esse segmento, por meio de code smells e atributos de qualidade. A partir da avaliação realizada em 200 repositórios da linguagem Python, da classificação dos principais code smells encontrados e a correlação desses com métricas qualitativas e realeses associadas, observou-se a presença de três code smells principais e a predominância da métrica Halstead de esforço no desenvolvimento desses repositórios.*

Bacharelado em Engenharia de *Software* - PUC Minas
Trabalho de Conclusão de Curso (TCC)

Orientador de conteúdo (TCC I): Cleiton Tavares - cleitontavares@pucminas.br
Orientadora de conteúdo (TCC I): Simone de Assis - simone@pucminas.br
Orientador acadêmico (TCC I): Laerte Xavier - laertexavier@pucminas.br
Orientador do TCC II: Johnatan Alves De Oliveira - johnatan.alves@sga.pucminas.br

Belo Horizonte, 14 de Maio de 2023.

1. Introdução

A adoção de novas tecnologias aplicadas ao desenvolvimento de *softwares* para sistemas de condução automatizados (ADSs, do inglês *Automated Driving Systems*) proporcionou um novo patamar de direcionamento para a evolução de veículos autônomos (AV, do inglês *Autonomous Vehicles*) Garcia and Chen (2020). *Softwares* ADSs são frequentemente compostos de várias unidades funcionais, conhecidas como recursos *ADS*, que automatizam tarefas de direção específicas como frenagem de emergência, reconhecimento de sinais de trânsito e controle de cruzamento [Abdessalem et al. 2020]. Estes sistemas possuem aplicabilidade em vias públicas e são *softwares* complexos que devem atender a vários requisitos, como segurança, cumprimento das regras de trânsito e conforto [Luo et al. 2022], atreladamente a sua execução. Assim, como todo *software*, está suscetível a *bugs* e podem causar acidentes fatais [Garcia and Chen 2020].

Algumas abordagens foram propostas na literatura para resolver esta condição. Dentre elas, os resultados do artigo de Stocco et al. (2020) discorrem sobre a detecção de falhas atreladas a disponibilidade de um conjunto rotulado de erros. Por meio de um ambiente de simulação como fator de execução, empregaram a classificação de erros para a geração de previsão e autorrecuperação de ADSs [Stocco et al. 2020]. Não obstante, Sharma and Kessentini (2021) apresentam um *dataset* identificativo de *code smells* relacionados a erros decorrentes de métricas de qualidade, coletadas de repositórios do *GitHub*. Assim, repositórios simuladores são frequentemente utilizados durante o desenvolvimento de *AVs* para previsão de erros de atuação, porém como são fontes *open source* podem não apresentar certo nível qualitativo que confere confiabilidade as simulações. **O problema a ser resolvido pontua-se na escassez informacional da qualidade de repositórios de *softwares open source* simuladores na promoção de *code smells*, que podem ocasionar em simulações de baixa confiabilidade**.

Com intuito de realizar buscas efetivas por *bugs*, Garcia and Chen (2020) realizaram análises em sistemas AVs através da coleta de 499 *bugs* por meio de *commits* de sistemas de *software* de código aberto. Além disso, demonstram que componentes de planejamento têm um alto número de *bugs* e exibem sintomas que são importantes para uma condução segura e correta de veículos [J. Garcia and Chen 2020]. Não obstante, Tang et al. (2021) por meio da avaliação de *issues* do *software open source Openpilot*, categorizaram *bugs* em relação a seu local de ocorrência no *software*, como *model bugs*, *plan/control bugs* e *UI bugs*. Promoveram também uma caraterização de *bugs* que ocorrem recorrentemente e refletem na abertura de *issues* com características semelhantes [Tang et al. 2021]. Logo, resolver o problema deste trabalho é importante, pois outras abordagens da literatura não apontam a necessidade atencional da relação entre a *softwa-*

res ADSs *open source* e a qualidade do código na promoção de *code smells* que podem contribuir para simulações falhas no desenvolvimento de *AVs*.

Dessa forma, **o objetivo geral do trabalho visa identificar o relacionamento entre atributos de qualidade de *softwares* simuladores *open source* ADSs na ocorrência de *code smells***. Para atingir esse objetivo foram estabelecidos três objetivos específicos: 1) Classificação de *code smells* frequentes em repositórios simuladores de *ADSs*, 2) Relação entre métricas de qualidade e a ocorrência de *code smells* classificados 3) Comparação e correlação da tendência quantitativa de *code smells* e métricas de acordo com a presença ou não de *releases* nestes repositórios.

Como resultado, foram pontuados os *code smells* frequentemente atrelados a repositórios *ADSs*, com o *code smell Long Method* possuindo maior ocorrência, seguidamente pelo *smell Comments*. Também foram realizadas a exposição de quais classificações estão diretamente relacionadas a um maior prejuízo para atributos internos de qualidade desses *software*, com o *code smell Long Method* novamente se destacando no que concerne a maior ocorrência de métricas como *Comment Ratio* e Complexidade Ciclomática.

Ademais, foram avaliadas a existência de uma associação de aumento ou decremento qualitativo relacionada a atuação de *code smells* em métricas qualitativas, onde se observou que *code smells* como *Comments* podem gerar o aumento de Complexidade Ciclomática e *Fan-out* mas não interferir em métricas *Halstead*. Por fim, foram caracterizados a tendência numérica de ocorrência de *code smells* relacionados a repositórios que não possuem *releases* em comparação aqueles que possuem, onde foi verificado que em ambos tipos de repositórios *Long Method* apresenta-se com maior ocorrência, em aproximadamente 70% dos repositórios. E que o fato de um repositórios apresentar *releases* ou não, não se relaciona diretamente a valores efetivos de mudança de métricas.

As próximas seções do trabalho se definem em : Seção 2 sobre a Fundamentação Teórica, Seção 3 na qual são demonstrados artigos relacionados a temática trabalhada. Seção 4 referente a Materias e Métodos, Seção 5 que discorre sobre as Definições do Estudo, Seção 6 em referência aos Resultados, Seção 7 sobre Discussões e Ameaças a Validade, por fim Seção 8 que detalha sobre a Conclusão e Trabalhos Futuros.

2. Fundamentação Teórica

Esta seção apresenta os principais conceitos e técnicas diretamente envolvidos na solução da problemática apresentada. A Seção 2.1 apresenta a conceitualização de Qualidade de *Software* , abordando sua difusão no meio tecnológico. A Seção 2.2 apresenta a Análise Estática de *Software* no que concerne como pode ser utilizada e o que pode apontar sobre um código após avaliação. A Seção 2.3 discorre sobre *Code Smells* e como esses podem apontar problemas relativos à qualidade de um código. Por final, a Seção 2.4 pontua sobre Sistemas de Direção Autônoma em relação ao seu funcionamento e como atuam.

2.1. Qualidade de *Software*

Segundo Ian Sommerville (2011), a qualidade de *software* se estabelece principalmente em seu modo ocorrente de execução, estruturação e de organização. Tais características se refletem nos atributos de qualidade de um *software*, como: robustez, complexidade e modularidade. No qual, por meio utilização de padrões durante o desenvolvimento, promovem propriedades determinísticas para o facilitamento de testes e manutenção em

produtos gerados. Assim como promulga [Chen et al. 2019], a respeito do relacionamento direto entre a qualidade apresentada por um *software* na criação de relações de dependência que podem fomentar a difusão de *bugs*.

2.2. Análise Estática de *Software*

Análises estáticas em *software* referem-se a uma avaliação em código fonte estático com o intuito de checar a existência de possíveis padrões errôneos, violações de programação e a identificação de paradigmas de qualidade em código [Plosch et al. 2008]. Os resultados apresentados por essas podem demonstrar o relacionamento direto entre o código desenvolvido e a qualidade externa demonstrada pelo *software* [Plosch et al. 2008]. Outrossim, tais análises podem apontar características qualitativas relacionadas a maior ou menor consumo de recursos na execução de programas [Chen et al. 2021]. Por fim, são recomendadas pela ISO 26262/2011 para reduzir *bugs* em tempos de execução, pertencente ao seguimento de padrões de segurança automotiva no desenvolvimento de *software* [Imparato et al. 2017].

2.3. *Code Smells*

Code smells definem implementações ou escolhas de *design* realizados negativamente que impactam a evolução de *softwares*, contribuindo para reduzir a capacidade de programadores no entendimento e mantenimento de aplicações [Hamdi et al. 2021]. Os mesmos podem apontar problemas com a qualidade do desenvolvimento e concomitantemente contribuem para a identificação de possíveis erros, além da promoção de facilitamento da refatoração a ser realizada por uma equipe técnica [Sharma 2018]. A partir da identificação de *code smells* e seu tratamento, é possível promover o aumento da qualidade do *software*, torná-lo mais fácil de manter e reduzir as chances de falhas em um sistema [Dewangan et al. 2021].

2.4. Sistemas de Direção Autônoma

Sistemas de direção autônoma são sistemas complexos que possuem segurança crítica diretamente relacionada ao atendimento de requisitos pré-estabelecidos [Luo et al. 2022]. Essa ocorrência pressupõe o recebimento de dados de vários sensores, analisados em tempo real por meio de redes neurais profundas. As mesmas são responsáveis por determinar parâmetros para o acionamento de atuadores de execução [Stocco et al. 2020]. Logo, os *softwares* desses sistemas são de fundamental importância para operação e sua composição decorre de várias unidades de componentes, que automatizam tarefas de direções específicas. Consequentemente, estão envolvidos diretamente na atuação e execução total de funcionamento, com a solicitação de recursos sendo realizada continuamente, além da promoção de abertura e recebimento de dados de componentes advindos de outras camadas [Abdessalem et al. 2020].

3. Trabalhos Relacionados

Nesta seção, são apresentados os trabalhos relacionados que concatenam-se ao objeto de estudo do presente trabalho. Assim, para demonstrar a relação direta entre a co-ocorrência de *code smells* em valores qualitativos de um *software*, Martins et al. (2020) realizaram a avaliação de dois *softwares closed source* na decorrência de 11 *releases*, através da verificação de *code smells* independentes e do estabelecimento de relações

de co-ocorrência. Assim, avaliaram métricas qualitativas como coesão, acoplamento e complexidade em decorrência anterior e posterior a remoção de *code smells* seletos. Como resultados relevantes ao presente estudo, pontua-se a promoção de *refactoring* desses com tendência a redução significativa da complexidade de um projeto e o aumento de valores de métricas como coesão e acoplamento [Martins et al. 2020a]. Logo, o trabalho se relaciona a este, na questão metodológica de verificação da relação entre *code smells* e atributos de qualidade.

Martins et al. (2020) a partir da análise de projetos de Linhas de Produtos de *Software* (*SPL*, do inglês *Software Product Line*). Na observância de funcionalidades desses *softwares*, postula sobre a interconexão existente entre *inter-smells* e a manutenibilidade desses sistemas. Por meio da utilização de dois SPLs, classificaram cinco *code smells* e realizam a definição de suas inter-relações. Nas quais posteriormente efetuaram refatoração e compararam *releases* antes e depois para verificar o impacto em relação a manutenibilidade utilizando métricas como Número de filhos (NOC, do inglês *Number of Children*)[Martins et al. 2020b]. Além disso, os principais resultados do artigo, que servem como embasamento dessa discussão, se referem a ponderação de não ocorrência de impactos a qualidade e manutenibilidade com a presença de *inter-smells*. Por fim, a conexão com o atual trabalho concerne-se no mesmo teor avaliativo da decorrência de *smells* em relação a *releases*.

Em detrimento do esforço despendido na coleta de informações de códigos necessárias para análise investigativa por parte de pesquisadores, Sharma and Kessentini (2021) propõem um conjunto de dados denominado *QScore*. Mediante a coleta e avaliação de qualidade de códigos de 86 mil repositórios de linguagens C# e Java minerados do *GitHub*, com a aplicação de um índice de qualidade e de cálculos de classificação de *code smells*. Para o fomento de sete tipos de arquitetura de *architecture smells*, 20 tipos de *design smells* e 27 métricas de qualidade de código. As análises realizadas são importantes para este trabalho, na promoção de classificações relacionais entre *code smells* e qualidade de métricas [Sharma and Kessentini 2021]. Logo, o trabalho estabelece uma conexão na presente manifestação por meio da promoção de passos executivos no que tange a classificação de *code smells*.

Outro trabalho relacionado, com o intuito de entender *bugs* decorrentes de AVs Garcia and Chen(2020) por meio dos *softwares open source* Apollo e Autoware, realizaram a investigação de 16.851 *commits*, 499 *bugs* e posteriormente classificaram esses conforme ocorrência de operação em componentes diretos do *software*. Como resultado, obtém-se uma taxonomia de erros recorrentes que afetam diretamente a ação desses *softwares* em relação a controle, planejamento e localização. Onde obtiveram a identificação de 13 causas principais, 20 sintomas de *bugs* e 18 categorias de componentes que esses *bugs* geralmente influenciam [J. Garcia and Chen 2020]. Logo, a relação causal com o trabalho valida-se no estabelecimento da ideia principal no que tange a categorização de *code smells*.

Por fim, para fomentar estratégias para refatoração de códigos em *Python*, Chen et al.(2016) promovem a detecção de *code smells* e realizam a classificação dos tipos que possuem maior ocorrência em sistemas dessa linguagem, por meio da ferramenta de detecção de *code smells* denominada *Pysmell*. Os resultados apontam a identificação de 285 instâncias de *smells* e revela a predominância de maior ocorrência dos tipos *Large*

Class e *Large Method* [Chen et al. 2016]. Logo, o trabalho relaciona-se com mesmo intuito investigativo, através da verificação de *code smells* que ocorrem majoritariamente em *softwares Python* de *ADSs*.

Com tal embasamento, o trabalho atual se diferencia dos trabalhos relacionados apresentados no intuito de pontuar uma classificação dos principais *code smells* ocorrentes em repositórios utilizados para a simulação de sistemas autônomos de direção. Não obstante, busca-se por meio da avaliação de análise estática de código averiguar a existência de uma relação ou discrepância existente entre a presença de determinada classificação de *smell*. Conjuntamente a avaliação dos elementos citados em relação ao impacto de repositórios que possuem *releases* e aqueles que não possuem, formulando assim uma pontuação qualitativa do presente objeto de estudo.

4. Materias e Métodos

Este trabalho é classificado como uma pesquisa quantitativa cujo objetivo é descritivo. Tal embasamento pode ser justificado dado a intenção desse em compreender o relacionamento existente entre atributos qualitativos de *software ADSs* e a presença de *code smells* por meio da análise de dados obtidos através da mineração de repositórios no *GitHub*.

Esta seção descreve ações para a realização da coleta e avaliação de repositórios simuladores de direção autônoma por meio de um estudo empírico. A Seção 4.1 mostra os Objetivos e Questões de Pesquisa estabelecidos para avaliação proposta. A Seção 4.2 descreve Etapas do Experimento, a seção 4.3 discorre sobre a Coleta de Dados, a Seção 4.3 aborda sobre a metodologia de Identificação de Métricas e *Code Smells* e por fim a Seção 4.5 apresenta a Sumarização e Padronização do processo realizado.

4.1. Objetivos e Questões de Pesquisa

A partir do modelo *Goal Question Metric* (GQM) [Ya-hong et al. 2013] com base em formulações voltadas ao *software*, foram definidas questões objetivas sobre o estudo. Com o objetivo de (i) identificar uma classificação frequente ocorrente entre *code smells*, métricas e *releases*, (ii) entender o relacionamento entre as respostas coletadas. Adentradamente, com base na identificação de um *dataset* de *code smells* Sharma and Kessentini (2021), discorreram sobre o relacionamento recorrente existente entre esses e métricas de qualidade relacionados a repositórios no *GitHub*. Motivados por tal aplicação, foram formuladas as seguintes questões de pesquisa.

RQ1: Quais os *code smells* frequentemente encontrados em repositórios de ADS?

RQ2: Qual a relação entre métricas de qualidade e *code smells* em repositórios ADS?

RQ3: Qual a relação entre repositórios *ADS* que possuem *releases* entre valores de métricas e *smells* distintos em comparação a repositórios *ADS* que não possuem *releases*?

Assim, as questões de pesquisa demonstradas objetivam um entendimento qualitativo de tais repositórios. Observa-se na primeira questão o objetivo de criação de uma classificação de *code smells* recorrentes nesses sistemas. Enquanto nas duas ultimas questões, examina-se uma possível relação entre métricas de qualidade, *smells* e *releases*. Uma vez que ao realizar o entendimento das relações existentes com os objetos coletados e classificados é possível apontar valores numéricos relacionais qualitativos. Tais aspectos

podem servir como embasamento para (i) demonstrar padrões tendenciosos negativos que ocorrem em *softwares* relacionados a essa temática, (ii) apontar a necessidade de atenção a projetos atuais em desenvolvimento e (iii) servir como base averiguativa no que tange a erros que devem ser evitados para futuros projetos.

4.2. Etapas do Experimento

Esta seção discorre sobre etapas para realizar o aferimento comparativo e qualitativo dos repositórios propostos. Onde foram realizados um conjunto de métodos para a obtenção das proposições, com o intuito organizativo foram separados em três decorrências, listadas em : 1) *Coleta de Dados*, 2) *Identificação de Métricas e Code Smells* e 3) *Sumarização e Padronização*.

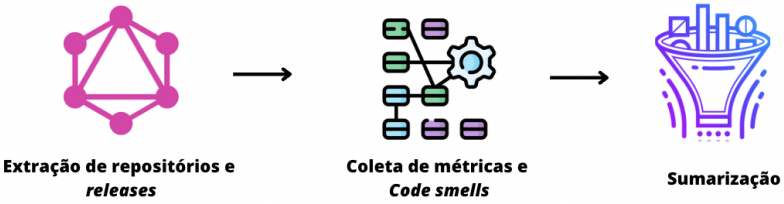
Com o objetivo de analisar repositórios relevantes escritos na linguagem *Python*, foram coletados 200 repositórios que contribuem com prerrogativas simulatórias para sistemas autônomos por meio do *GitHub*. A partir de *scripts* com a mesma linguagem, no ambiente de desenvolvimento *Pycharm*, foram desenvolvidos códigos para a coleta de repositórios, averiguação da presença de *releases*, geração e análise de métricas. Os dados gerados foram armazenados no formato de arquivos CSV (do inglês, *comma separated values*).

Tais conceitos foram realizados após averiguação manual em buscas no *GitHub* das principais linguagens utilizadas para no desenvolvimento desses, no qual Python se apresentou de modo impeto a mais utilizada. Conjuntamente, a coleta do número de *releases* se mostrou importante com embasamento na prerrogativa de que muitos repositórios não possuem *releases* ou possuem em grande quantidade, podendo ser utilizado como fator relacional comparativo.

Para a obtenção de resultados com maior embasamento analítico, foram realizados os seguintes critérios em relação à seleção e utilização de repositórios. 1) Estipulação de uma temática central de *topic* no inglês, com o intuito de obter dados qualitativos de um nicho padronizado linguístico mundial para a coleta dos repositórios. 2) Definição de análise da coleta do valor total de *releases* de um repositório, uma vez que seria dispendioso o tempo gasto de aferimento para análise no decorrer de todas *releases* que um repositório possui e levando em conta baixo valor de apontamento qualitativo para o projeto no que tange mudanças.

A figura abaixo demonstra os passos relacionado para a formulação de coletas e averiguações, onde o primeiro passo do fluxo se relaciona a extração dos repositórios e coleta de *releases*, seguidamente da averiguação das métricas e dos *smells* presentes nesses e por último a seleção e formalização de uma classificação das sentenças coletadas.

Figura 1. Procedimentos para realização do experimento



4.3. Coleta de dados

A coleção de repositórios ocorreu atráves da *API* (do inglês, *Application Programming Interface*) *GraphQL* do *Github*, devido a facilidade que essa promove o selecionamento de diversas informações, para a seleção de campos desejados. Para a estruturação do conjunto de dados foram selecionados via *query* os 200 repositórios da linguagem *Python* em ordem de atualização mais recente com a estipulação dos tópicos em *autonomous-driving*, *self-driving-cars* e *automous-vehicles*. Não obstante, foi também coletado o nome do repositório referencial, sua *url* (do inglês, *Uniform Resource Locator*), seu *owner* e o número de *releases* que esse possui.

A partir dos nomes de projetos coletados foi realizada em código uma verificação da presença de *releases* em cada repositório coletado, uma vez que nem todos possuem *releases*. Posteriormente foram gerados CSV's separados entre repositórios que possuem *releases* e aqueles que não possuem.

4.4. Identificação de Métricas e *Code Smells*

Para a realização do aferimento investigativo de *code smells*, no que tange a realização de análise estática do código foram coletadas diferentes métricas por meio das bibliotecas: *Radon* e *Multimetric*. A primeira foi escolhida por apresentar diversas informações relacionadas a computação numérica de códigos desenvolvidos, fornecendo informações como comentários e tamanho de código e servir como base para o apontamento de *code smells* e a segunda por fornecer uma variedade de métricas que podem apontar propriedades mensuráveis do *software*.

Na Tabela 1 são apresentadas informação relacionadas a biblioteca *Randon*, no qual métricas como *LOC*, *Comments* e suas subdivisões foram utilizadas para fomentar características relacionadas a definição de *code smells*. Adentralmente, na Tabela 2 são apresentados as métricas relacionadas a biblioteca *Multimetric*, onde se verificam as métricas *Halstead*, que medem a complexidade de módulos de um programa e fornecem vários indicadores de complexidade relacionados a dificuldade e esforço na manutenção de códigos por meio de métricas como *Halstead Effort*, *Halstead Difficulty* e *Halstead Timequired*, utilizadas para averiguar a existência de relações com os *smells* verificados.

Tabela 1. Métricas biblioteca *Radon*

Métrica	Especificação
LOC	Número total de linhas de código
Comments	Número total linhas de comentário
Single Comments	Número de linhas únicas de comentários
Blank	O número de linhas em branco (ou apenas com espaços em branco)
Multi	O número de linhas que representam strings de várias linhas

Tabela 2. Métricas biblioteca *Multimetric*

Métrica	Especificação
Comment Ratio	Porcentagem de comentários em código
Cyclomatic Complexity	Número de decisões que um bloco de código contém
Fan-in	Núnero de funções que chamam uma determinada função
Fan-out	Número de funções chamadas pela função
Halstead Effort	Esforço necessário para manter um programa
Halstead Difficulty	Dificuldade para manter um programa
Halstead Timequired	Tempo necessário de desenvolvimento
Halstead Volume	Volume de desenvolvimento
Halstead Bugprop	Número de *bugs* entregáveis de acordo com Halstead

4.5. Sumarização e Padronização

Após a computação das métricas com base em cada código-fonte, os arquivos gerados foram tradados de acordo a se obter a informação qualitativa desejada, como exemplo, no arquivo *Multimetric* gerado após a análise dos repositórios foram coletadas todas as métricas presentes. Porém, em contrapartida as métricas *Halstead* pertencentes a biblioteca *Radon* não foram utilizadas. Outro exemplo a ser citado é a utilização do arquivo referente a métrica Complexidade Ciclomática para a verificação de Funções, Métodos e Classes referentes a um repositório mas não necessariamente a coleta do valor dessa métrica, uma vez que a mesma foi coletada por meio da biblioteca *Multimetric*.

Posteriormente a coleta das métricas ocorreu a sumarização da decorrência de *code smells* mais frequentes encontrados em tais repositórios e a pontuação numérica de métricas qualitativas mais predominantes para posteriormente a realização de análises finais. No qual, para averiguar a relação entre *code smells* e as métricas pontuadas foram realizadas correlações *Pearson* por meio de biblotecas da linguagem *Python*.

A correlação *Pearson* foi escolhida, pois assim como apresentado por [Sch 2018] os valores avaliativos variam entre -1 a 1, onde um valor em sentido positivo pode indicar que as variáveis estão diretamente relacionadas, ou seja, correlações positivas completas e um valor em sentido negativo, correlações completas negativas, podendo apontar variáveis não relacionadas.Adentradamente, valores que se promulgam na variação desses como +0.5 ou -0.5 são referentes a correlações moderadas e valores que se aproximam das relações completas com +0.98 e -0,98 são considerados correlações fortes . Ademais, as estatísticas que foram analisadas são ambas quantitativas, o que pontua uma comparação de variáveis para demonstração de uma relação linear.

Ao final foram elaborados gráficos por meio do software *PoweBI* que facilita a demonstração de dados. Conjuntamente a gráficos comparativos em relação ao percentual de ocorrência numérica da classificação de *code smells* frequentemente encontrados em *ADSs* e a relação numérica das métricas coletadas. No intuito de formular uma padronização de dados para esses gráficos e facilitar a visualização dos rótulos de métricas, considerando também a quantidade de repetitividade de citamento dessas. Foram definidas siglas em acordância com o nome das métricas, assim como pode ser visto na Tabela 3, onde a métrica *Comment Ratio* foi definida como *CR*.

Tabela 3. Siglas definidas

Métricas	***Siglas***
Comment Ratio	*CR*
Cyclomatic Complexity	*CC*
Halstead Effort	*HE*
Halstead Difficulty	*HD*
Halstead Timequired	*HT*
Halstead Volume	*HV*
Halstead Bugprop	*HB*

5. Definições do Estudo

Para a resolução das questões RQ1, RQ2 e RQ3 foram estabelecidas prerrogativas que objetivam a promoção das respostas em observância a averiguações numéricas e classificatórias, no intuito de uma abordagem dinâmica. Assim, após execução das bibliotecas referenciadas na Seção 4 e posteriormente a coleta e armazenamento dos dados fornecidos, foram estabelecidos passos e critérios para a resolução das problemáticas, listados a seguir.

Para responder RQ1, a partir das informações fornecidas pelas bibliotecas utilizadas, foram estabelecidas duas prerrogativas de classificação. Foram realizadas buscas, com embasamento no tamanho de *LOC* de um código referenciado a tipos de blocos de código como: métodos e classes. Assim, por meio da estipulação referencial dos *code smells Large Class* e *Long Method* apontados por Chen et al.(2016), no qual o *smell Large Classe* é pontuado quando uma classe possui um valor definido em $LOC \geq$200 e o *smell Long Method* é contabilizado a partir da ocorrência de um método com $LOC \geq$100.

Segundamente a averiguação do total de comentários presentes em código é baseada na soma dos subgrupos de *Comments*: *Single*, *Multi* e *Blank*. Assim, por meio da definição de um *code smell* do tipo *Comment* caracterizado por [Rasool and Arshad 2016] a partir do cálculo: *LOCCmntd*/*LOCTotal* $\geq$ LOC%. Onde, *LOCCmntd* é referente a soma dos subgrupos de *Comments* . Esse *smell* é levado em consideração quando *LOCCmntd* dividido pelo *LOC* total do código, definido no artigo por *LOCTotal*, é maior ou igual a porcentagem de *LOC* encontrada, definida pelo estudo em *LOC%*.

Para responder RQ2 a partir da coleta das respostas das bibliotecas *Radon* e *Multimetric*, respectivamente sobre métricas *Raw* e *Halstead*, foram observadas as relações existentes entre a decorrência de cada tipo de *smell* e o valor numérico das métricas. Por meio do estabelecimento de prerrogativas de análises, no qual a separação ocorre a partir

dos tipos de *code smells* encontrados em repositórios que apresentam ou não um determinado tipo de *smell* e as métricas relacionadas a esses repositórios. Como exemplificação, a comparação entre repositórios que possuem o *smell Comment* e suas métricas e os repositórios que não possuem esse *smell* conjuntamente as métricas. Tal escolha foi realizada no intuito de averiguar relações existentes ou não entre *smells* e métricas, dado a suposição de que um *smell* pode ou não alterar valores referentes a determinada métrica como Complexidade Ciclomática ou *Halstead Difficulty*.

Para responder RQ3 com viés quantitativo, após a coleta das informações de cada biblioteca e sua sumarização foram comparados numericamente os valores obtidos de repositórios que possuem *releases* e daqueles que não possuem, conjuntamente a contabilização de grupos de *smells* presentes em ambos tipos de repositórios. Com o intuito de apontar a existência de um relacionamento entre o fato de um repositório possuir *releases* ou não, em acordança com os valores numéricos de métricas e percentualidade da presença de *smells*. Baseado na ponderação de que um repositório com *releases* podem possuir valores de métricas como *Halstead Effort* menores em comparação aqueles que não possuem.

6. Resultados

Nesta seção são demonstrados os resultados obtidos neste estudo. Na qual, são apresentados os valores e análises obtidos a partir da execução de etapas de cada método e posteriormente discorre-se sobre os os outputs obtidos.

6.1. RQ1: Quais os *code smells* frequentemente encontrados em repositórios de ADS?

Após a sumarização dos resultados foram obtidos três *code smells* frequentemente encontrados em sistemas *ADS*, como demonstrado na Tabela 4, que apresenta os principais *code smells* e suas totais ocorrências. Na mesma é possível verificar uma maior ocorrência de *smells* em repositórios *ADS* atrelados ao *smell Long Method*, conjutamente a pontuação de menor quantidade de *smells* ser referente a *Large Class*, com 737.476 ocorrências.

Tabela 4. *Code smells frequentemente encontrados em repositórios ADS*

Tipos de Smell	*Total de Ocorrências*
Long Method	5.550.865
Comments	1.541.942
Large Class	737.476

6.2. RQ2: Qual a relação entre métricas de qualidade e *code smells* em repositórios *ADS*?

Foram observados 95 repositórios que possuem *Comment smells* e 112 repositórios que não possuem esse *smell*. Assim, a partir da Figura 2 é possível observar os repositórios que possuem esse tipo de *smell*, os quais apresentam altos valores de Complexidade Ciclomática correspondente a um grande número de estruturas de decisões como também um alto número *Fan-out* que indica um grande número de módulos que possuem relações de uso. Enquanto na Figura 3, que referência os repositórios que não possuem esse *smell*,

é possível notar que existe uma alternância de valores dos repositórios que não possuem *Comments smells*, uma vez que o valor de *Comment Ratio* que se refere a taxa de comentários, diminui e o valor de Complexidade Ciclomática aumenta consideravelmente. É possível pontuar também a repeito da diferença numérica, na qual foram encontrados em menor quantidade repositórios que possuem esse tipo de *smell* porém com valores altos das métricas.

Figura 2. Métricas relacionadas a repositórios que apresentam Comment smells

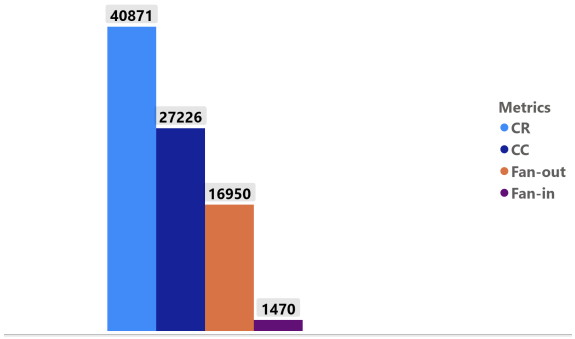
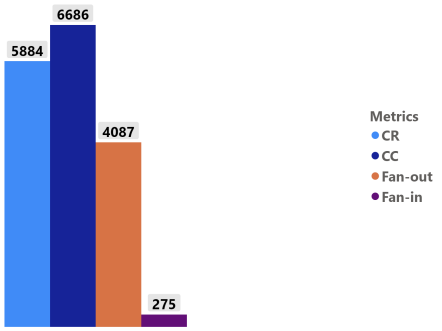


Figura 3. Métricas relacionadas a repositórios que não apresentam Comment smells



Em relação a métricas *Halstead* é possível observar por meio da Figuras 4 que apresenta as métricas *Halstead* relacionadas a repositórios que possuem *Comments smells* e da Figura 5 que apresenta as métricas *Halstead* relacionadas a repositórios que não possuem *Comments smells*, que não ocorre alternância de valores altos de métricas em repositórios que possuem e não possuem *Comment smells*. É possível também visualizar que a métrica de *Halstead Effort* se apresenta como principal em ambas averiguações, relacionada a um alto esforço para manter tais repositórios mesmo que possuam ou não esse tipo de *smell*.

Figura 4. Métricas Halstead relacionadas a repositórios com Comment smells

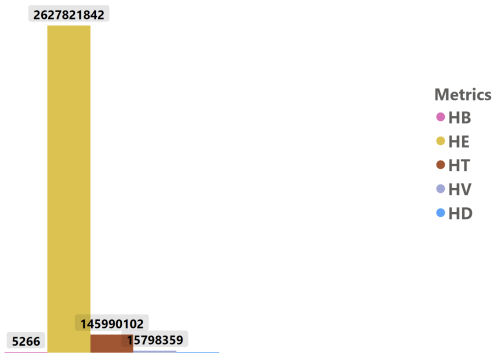
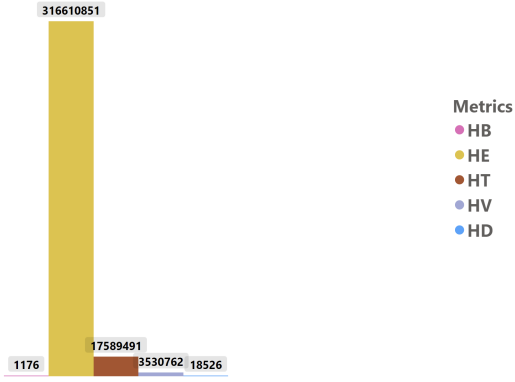


Figura 5. Métricas Halstead relacionadas a repositórios que não possuem Comment smells



A Figura 6 demonstra as métricas relacionadas a repositórios que possuem o *smell Large Class*, onde é possível observar que repositórios que possuem *Large Class* apresentam alta diferença entre os valores de Complexidade Ciclomática e *Fan-out*, diferentemente da Figura 7 que demonstra os repositórios que não possuem o *smell Large Class*. Sendo essas métricas em quase equidade, indicando um menor número de estruturas de decisões e menor número de módulos com relações. Não obstante, é possível verificar que os valores de *Fan-in* em relação a estrutura de repositórios que contém *Large Class* se apresenta um pouco maior em relação aos que não possuem, porém nos dois gráficos o valor é considerado baixo em relação a outras métricas demonstradas.

Figura 6. Métricas relacionadas a repositórios que possuem Large Class

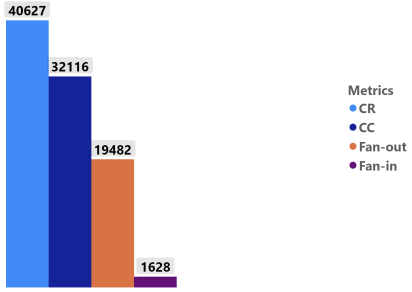
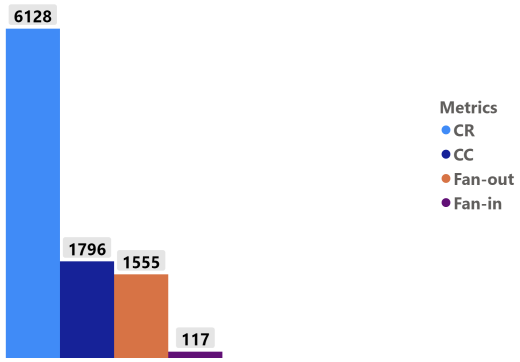


Figura 7. Métricas relacionadas a repositórios que não possuem Large Class



Novamente, por meio da Figura 8 é possível observar as métricas *Halstead* relacionadas aos repositórios que possuem *Large Class* e na Figura 9 é possível visualizar as métricas *Halstead* relacionadas a repositórios que não possuem *Large Class*. No qual se nota a predominância da métrica *Halsted* de esforço, além da reafirmação de que não existe disparidade desses valores em repositórios que apresentam ou não esse tipo de *code smell*. Observa-se também a presença da métrica *Halstead Volume* em maior presença nos dois gráficos se comparada a sua presença nos gráficos 4 e 5 a despeito do *smell Comments*.

Figura 8. Métricas Halstead relacionadas a repositórios que possuem Large Class smells

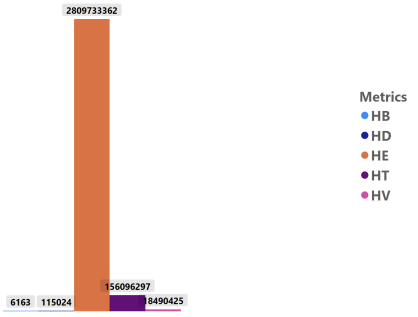
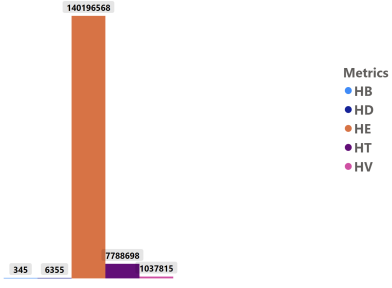


Figura 9. Métricas Halstead relacionadas a repositórios que não possuem Large Class smells



Por meio das Figuras 10 e 11 é possível observar que repositórios que possuem *code smells* do tipo *Long Method* e os repositórios que não apresentam esse tipo de *smell*, apresentando altos valores de *Comment Ratio*, conjuntamente é possível observar Complexidade Ciclomática apresenta considerável valor alto dado a presença desse *smell*. Sendo possível notar também tal disparidade envolvendo a estrutura de *Fan-out*, principalmente no que tange a estrutura comparativa entre Complexidade Ciclomática nos dois gráficos. Novamente, não possuem valores discrepantes em relação a comparação da métrica *Fan-in*, com ambos gráficos apresentando um fator relacional.

Ademais, é possível verificar estruturas gráficas semelhantes em relação aos gráficos destacados e aos gráficos 6 e 7 referentes respectivamente a ocorrência *Large Class* e não ocorrência, apontando uma possível relação significativa no que tange a ocorrência de ambos *smells*, uma vez que como as métricas fornecidas apresentam valores similares podem apresentar também que ambos *smells* estejam ocorrendo em conjunto.

Figura 10. Métricas relacionadas a repositórios que possuem Long Method smells

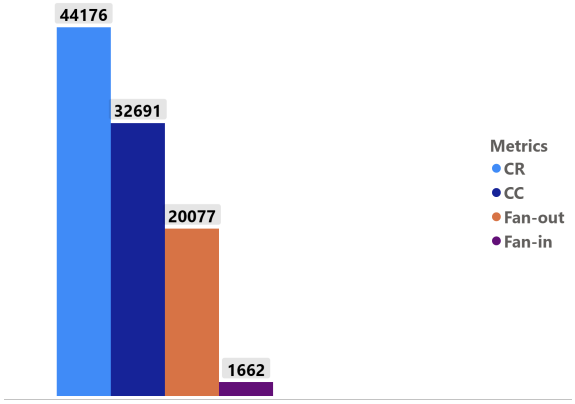
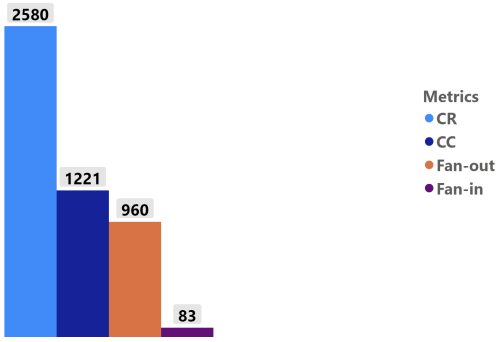


Figura 11. Métricas relacionadas a repositórios que não possuem Long Method smells



Em relação as métricas *Halstead* por meio da Figura 12 são observadas as métricas *Halstead* relacionadas a repositórios que possuem o *smell Long Method* e por meio da Figura 13 se vizualiza as métricas *Halstead* relacionadas a repositórios que não possuem o *smell Long Method*. Onde se vizualiza novamente *Halstead Effort* como principal métrica em repositórios que possuem ou não o *smell Long Method*.

Figura 12. Métricas Halstead relacionadas a repositórios que possuem Long Method smells

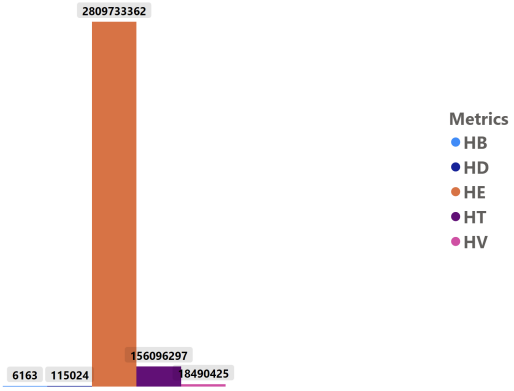
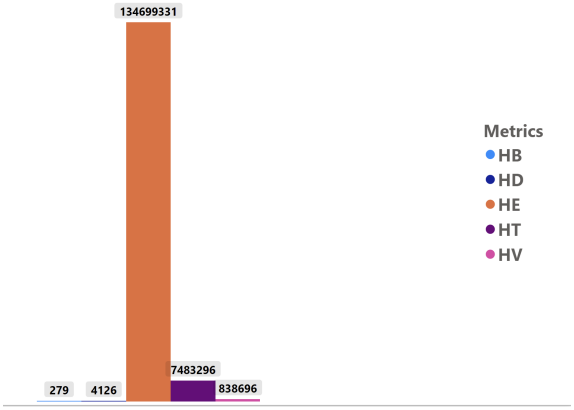


Figura 13. Métricas Halstead relacionadas a repositórios que não possuem Long Method smells



A partir das correlações *Pearson* realizadas, é possível notar por meio da Figura 14 como ocorre a relação linear entre as métricas verificadas por meio de repositórios que apresentaram os *smells* destacados nesse estudo. Observa-se que *CR* possui valores próximos e de correlação negativa completa com *CC*, *Fan-out* e *Fan-in*, respectivamente com os valores -0.96, -0.99 e -1. Porém nenhuma das outras métricas desse mesmo grupo apresenta correlações negativas e somente positivas com os mesmos elementos do grupo. Ademais, pode se pontuar que CR apresenta correlações próximas de positivas e positivas completas com métricas *Halstead* como *HB*, *HD*, *HE*, respectivamente 0.98, 0.99 e 1. Verifica-se também que não existem correlações moderadas, ou seja, ou a correlação se apresenta completa ou forte, em seu sentido positivo ou negativo.

Em observância as correlações *Pearson* realizadas, é possível notar por meio da Figura 15 como ocorre a relação linear entre as métricas verificadas por meio de repositórios que não apresentam um dos tipos *smells*. No qual é possível observar a presença de relacionamentos fortes e completas positivas por meio dos dois grupos de métricas avaliados, excetuando-se *CR* que apresente correlação moderadas negativa com *HB*, *HE*, *HT*, respectivamente 0.5, 0.47 e 0.47 . É possível também verificar que não ocorrem relações completas fortes ou completas negativas, diferentemente como visto no gráfico anterior.

Figura 14. Correlação *Pearson* entre métricas e ocorrência de *smells* em repositórios

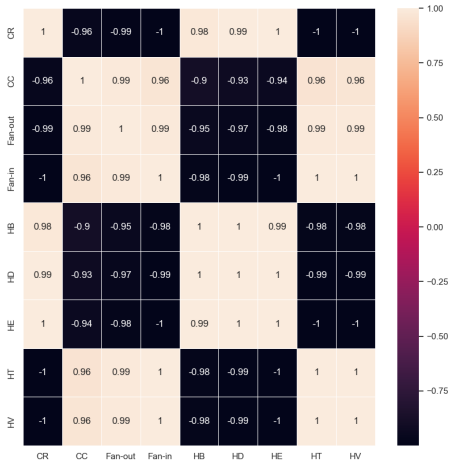
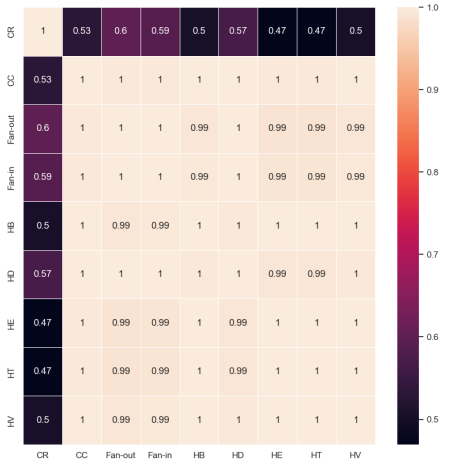


Figura 15. Correlação *Pearson* entre métricas e não ocorrência de *smells*



6.3. Qual a relação entre repositórios *ADS* que possuem *releases* entre valores de métricas e *smells* distintos em comparação a repositórios *ADS* que não possuem *releases*?

A partir da coletas e classificação de *smells* e *releases*, é possível verificar que em repositórios que possuem *releases* o número de *code smells* do tipo *Long Method* possui maior ocorrência, assim como pode ser visto na Figura 16 que referencia a porcentagem de *smells* relacionados a repositórios que possuem *releases*. Não obstante esse é o mesmo *smell* com maior ocorrência também em repositórios que não possuem *releases*, assim como visto na Figura 17. É possível observar uma pequena disparidade percentual entre esses tipos de repositórios onde em relação a segunda figura nota-se um maior percentual de *Comment smell*.

Figura 16. Porcentagem de smells relacionados a repositórios que possuem releases

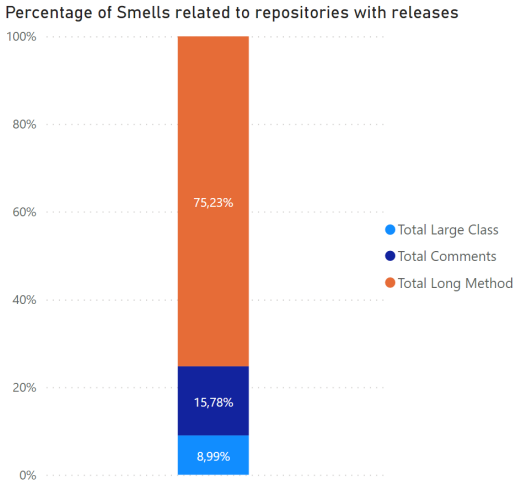
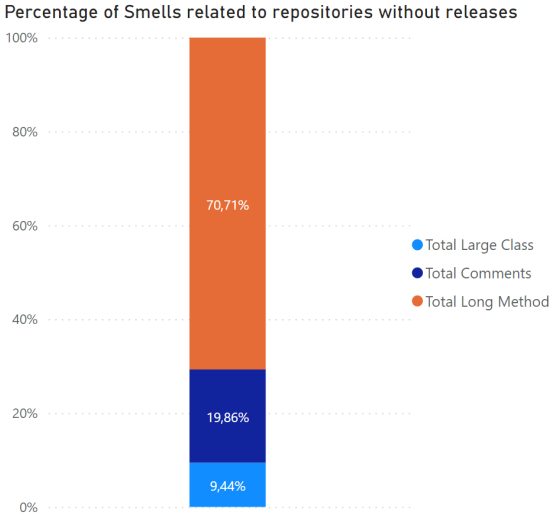


Figura 17. Porcentagem de smells relacionados a repositórios que não possuem releases



A despeito de métricas, em repositórios que possuem *releases* observa-se uma maior predominância de métricas relacionadas a taxa de comentários, seguidamente a um alto número de Complexidade Ciclomática e *Fan-out*, como visto na figura 18. Semelhantemente ao que pode ser visto na Figura 19 que representa a porcentagem de métricas relacionadas a repositórios que não possuem *releases*. Em que repositórios que não possuem *releases* apresentam como maior valor *Comment Ratio* seguidamente de Complexidade Ciclomática.

Figura 18. Porcentagem de métricas relacionadas a repositórios que possuem releases

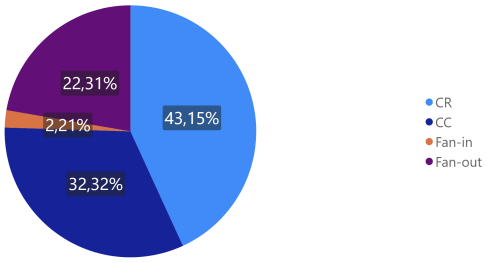
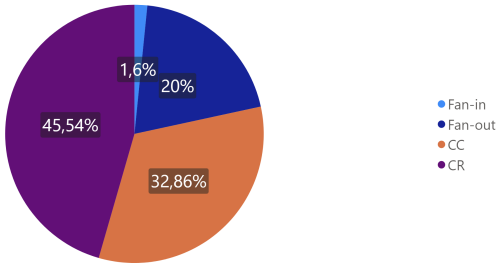


Figura 19. Porcentagem de métricas relacionadas a repositórios que não possuem releases



Novamente é possível verificar por meio da Figura 20 a porcentagem de métricas *Halstead* relacionadas a repositórios que possuem *releases*, enquanto na Figura 21 observa-se a porcentagem de métricas relacionadas a repositórios que não possuem. Nos quais ambos repositórios, possuem valores semelhantes de métricas *Halstead*, com a predominância da métrica *Halstead Effort*.

Figura 20. Porcentagem de métricas Halstead relacionadas a repositórios que possuem releases

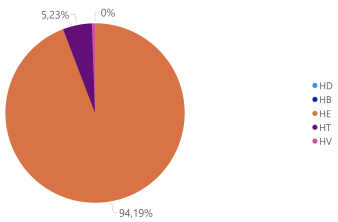
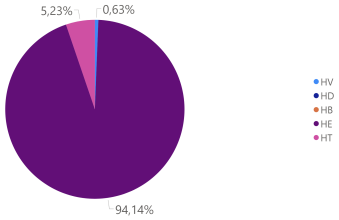
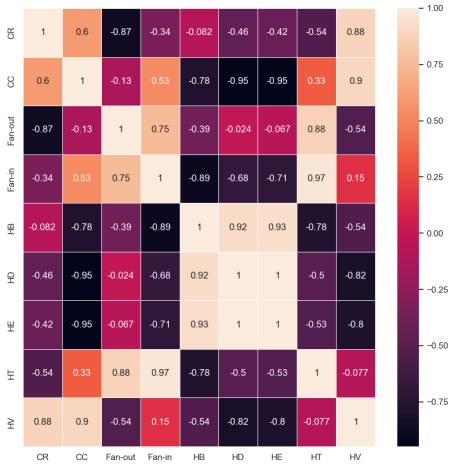


Figura 21. Porcentagem de métricas Halstead relacionadas a repositórios que não possuem releases



Por meio das correlações Pearson realizadas em repositórios que apresentam *smells* a respeito do relacionamento com as métricas coletadas, assim como pode ser visto na Figura 22. Verifica-se em primeira instância correlações fortes e completas entre as métricas do grupo *Halstead* como a relação de HB com HD e HE, respectivamente 0.92 e 0,93. Verifica-se também que métricas como CR, CC, Fan-out e Fan-in apresentam entre si correlações negativamente fortes, fracas positivas, moderadas positivas e nenhuma completa positiva. Logo, tais valores dispersos, sem o apontamento de correlações completas positivas e negativas, demonstram um quadro não direto de linealidade de relacionamento.

Figura 22. Correlação *Pearson* entre métricas e ocorrência de smells em repositórios que apresentam *releases*



A despeito das correlações realizadas por meio dos repositórios que não apresentam *releases*, como mostrado na Figura 23 verificam-se relacionamentos positivos fortes entre métricas como CC, Fan-out e Fan-in, respectivamente com valores relacionais de 0.98 e 0.72. Novamente excetuando CR, que apresenta valores positivos fracos e até negativos moderados, como visto em relação as métricas CC e Fan-in, respectivamente 0.094 e -0,62. Ademais, é possível visualizar resultados fortes e completos positivos em relação as métricas do grupo *Halstead*, excetuando-se HT e HV.

Figura 23. Correlação *Pearson* entre métricas e ocorrência de *smells* em repositórios que não apresentam *releases*

7. Discussões e Ameaças a Validade

Com base nos estudos realizados, pode-se inferir que em relação as análises coletados dos repositórios uma atenção no que concerne a grande quantidade de comentários pontuados, uma vez que essa associação pode estar relacionadas a formas explicativas de se tentar validar o funcionamento de um método ou classe, mascarando problemas relacionados a qualidade do código, dado que muitas vezes um pequeno comentário pode ser simplório e significativo para a explicação de determinado método. Atreladamente foram verificados altos valores de Complexidade Ciclomática, relacionados ou não a pontuação de *smells*, pre-supondo assim que tais repositórios possuem grandes números de estruturas de decisões e consequentemente estarem ligados a *softwares* que possuem alta complexidade.

Com relação as ameaças de validade do presente trabalho, como validade interna pode-se postular sobre falsos positivos na averiguação da disparidade de repositórios com *releases* e sem *releases*. Uma vez que existe um grande número de repositórios sem *releases* e um pequeno número de repositórios com *releases*, e que para a análise realizada foram selecionados o mesmo número de repositórios de ambas as partes de maneira não padronizada mas randômica. Consequentemente as ferramentas de análise estática de código, podem apresentar problemas em relação a qualidade avaliativa. Como validade externa, o presente estudo aponta especificamente sobre *code smells* relacionados ao nicho de repositórios *ADS* relacionados a linguagem *Python* e as conclusões apresentadas não devem ser estendidas a outro nicho mesmo que ponderado a partir da linguagem *Python* ou sobre o nicho de *ADS* relacionados a outros tipos de linguagem.

Adentradamente, a metodologia promulgada inicialmente nesse estudo passou por decorrências relacionadas a limitação de ferramentas utilizadas para análises e sobre os métodos de coleta dos repositórios. Assim, é fundamental que em pesquisas futuras leve-se em consideração tais fatores e sejam realizados ajustes não somente nas ferramentas utilizadas para análise mas também a melhoria dos fatores de sumarização observados e desenvolvidos. Não obstante, recomenda-se a exploração de novos nichos relacionais

para maior fomento informacional final, conjuntamente a diferentes buscas por projetos em diferentes fontes, promovendo assim diversidade qualitativa para o apontamento de resoluções final.

8. Conclusão e Trabalhos Futuros

Neste estudo, foram investigadas as relações existentes em repositórios *open source* de simulação para *ADS* por meio da análise de *code smells*, métricas qualitativas e a promoção de *releases*. Durante a pesquisa foram descobertos três *code smells* frequentemente relacionados a esses repositórios: *Large Class*, *Long Method* e *Comments*.

Ademais, é possível pontuar sobre a predominância dos *smells Long Methods* e *Comments*, na qual ocorre uma relação existente entre esses, uma vez que os *smells Long Method* podem estar relacionados métodos que possivelmente aumentaram em grandes proporções e se tornaram difíceis em serem trabalhados. Logo, tal método pode ter sido preenchido com comentários explicativos, onde esses se relacionam ao fato do autor do código perceber que seu código não é intuitivo ou óbvio. Nesses casos, os comentários são uma forma de mascar os *smells* do código que poderia ter sido melhorado.

A despeito dos relacionamentos existentes entre *code smells* e métricas foi possível inferir em primeira instância que repositórios que possuem o *code smell Comments* em comparativo aos repositórios que não possuem esse *smell*, apresentam em relação as suas métricas valores diferenciais de predominância. Como exemplo, as discrepâncias existentes entre a métrica Complexidade Ciclomática, com essa sendo predominante em repositórios que não possuem esse *smell* e aparecendo em segundo lugar em relação a repositórios que apresentam esse *smell*. Logo, é possível averiguar que em repositórios que não apresentam Comments verificam-se maior números de estruturas de decisão, logo não existem um grande número de comentários associados dado ao seguimento de um fluxo com grande quantidade de subdivisões.

Conjuntamente a respeito do *code smell Large Class* verifica-se que em repositórios que possuem esse *smell* apresentam altos valores de Complexidade Ciclomática e *Fan-out* em comparação a repositórios que não possuem esse *smell*. Assim, observou-se que repositórios que possuem esse *smell* estão diretamente relacionados a classes que possuem muitas estruturas de decisão e funções que são chamadas e consequentemente apontam para classes grandes que são responsáveis por diversas funcionalidades. Porém, é possível verificar que em ambos tipos de repositórios a métrica *Comment Ratio* aparece em destaque, promulgando que nos dois casos independentemente do *code smell Large Class*, existe uma taxa alta de comentários associadas a esses repositórios.

Sobre o *code smell Long Method* é possível verificar que para repositórios que apresentam e não apresentam esse *smell* a métrica Complexidade Ciclomática possuiu alta diferenciação quantitativa, assim é possível verificar que esses repositórios estão associados a métodos com altos números de linhas de código e consequentemente possuem um grande número de estruturas de decisão.

A respeito da análise de métricas *Halstead* é possível concluir que os *code smells* encontrados não estão relacionados a maior ou menor atuação dessas métricas, dado que independentemente dos repositórios possuírem ou não certo tipo de *smell* ocorreu sempre a predominância das métrica *Halstead Effort* e *Halstead Timequired*. Com a primeira

sendo apresentada em altos valores e promulgando que o desenvolvimento de tais repositórios exigem alto esforço para desenvolvimento. A despeito da segunda métrica, ela se apresenta em segunda predominância porém não em altos valores se comparado a primeira, porém é possível verificar que quando comparadas a outras métricas analisadas como *Halstead Difficulty* e *Halstead Volume* se destaca , postulando sobre a existência de um grande tempo despendido para o desenvolvimento de tais repositórios, porém não há uma grande taxa de volume ou dificuldade para esse desenvolvimento.

Sobre as correlações *Pearson* realizadas verifica-se que a presença de *smells* provoca uma relação direta no aumento ou decremento de métricas. Dado que quando esses ocorrem verificam-se relações completas fortes entre métricas dos mesmos grupos, porém relacionamentos fortes negativos de um grupo em comparação ao outro. Diferentemente do que é visto quando esses *smells* não ocorrem, com as correlações apresentando-se todas positivas completas ou fortes.

A partir da averiguação que existem mais repositórios com *releases* em comparação aos repositórios que não possuem *releases* e da averiguação de *code smells* e métricas, conclui-se que para repositórios que apresentam ou não *releases*, predominam a ocorrência de *code smells Long Method*, seguidamente de *Large Class* e *Comments*. Observa-se também que independentemente dos repositórios possuírem *releases* ou não métricas como *Comment Ratio*, *Complexidade Ciclomática* e *Fan-out* possuem altos valores. Igualmente como ocorrem em relação as métricas *Halstead Effort* e *Halstead Timequired*, que se apresentam proeminentes em ambos percentuais.

A respeito da averiguação das correlações sobre a presença ou não de *releases* nos repositórios, verificam-se que os relacionamentos não possuem uma relação predominante estabelecida,logo as *releases* se apresentam como não diretamente relacionadas no aumento ou decremento de métricas.Também, é possível discorrer que a partir dos repositórios analisados o percentual de *smells* e métricas não se distingue com a presença ou não de *releases*, apontando um aspecto de atenção de que tais repositórios não costumam possuir atualização e quando realizadas são feitas de maneira superficial, de modo a seus valores qualitativos não sofrem alteração.

Ao final, avalia-se por meio dos repositórios coletados, que embora esses apresentem certo nível qualitativo devido a não grande diferenciação de métricas durante as análises realizadas, esses necessitam de atenção quanto a códigos desenvolvidos e em desenvolvimento, dada a crescente atuação de *softwares ADSs*.Uma vez que após as observações, esses repositórios apresentam *smells* que podem ser facilmente refatorados, gerando mais qualidade ao código, não somente a nível de legibilidade mas também de entendimento.Como trabalhos futuros, sugere-se a busca por outras classificações de *smells* de modo a caracterizar melhor os *code smells* presentes em *ADSs*. Não obstante, a promoção de *refactoring* a partir dos *code smells* encontrados no intuito de promover uma análise qualitativa para verificar se esses *smells* empregam efeitos negativos em código. Ademais, a utilização e aplicação de mais métricas com intuito de prover maior embasamento analítico, conjuntamente a averiguação de aumento ou decremento dessas a partir da análise no decorrer de *releases* dos repositórios que possuem.

Pacote de Replicação

O pacote de replicação deste trabalho encontra-se disponível em:

`https://github.com/ICEI-PUC-Minas-PPLES-TI/plf-es-2022-2-tcci-5308100-pes-lorrayne-reis`

Referências

[Sch 2018] (2018). Correlation coefficients: Appropriate use and interpretation.

[Abdessalem et al. 2020] Abdessalem, R. B., Panichella, A., Nejati, S., Briand, L. C., and Stifter, T. (2020). Automated repair of feature interaction failures in automated driving systems. page 88–100.

[Chen et al. 2019] Chen, C., Shoga, M., and Boehm, B. (2019). Exploring the dependency relationships between software qualities. pages 105–108.

[Chen et al. 2021] Chen, L., Chen, T., Fan, G., and Yin, B. (2021). Static analysis of resource usage bounds for imperative programs. pages 580–581.

[Chen et al. 2016] Chen, Z., Chen, L., Ma, W., and Xu, B. (2016). Detecting code smells in python programs. pages 18–23.

[Dewangan et al. 2021] Dewangan, S., Rao, R. S., Mishra, A., and Gupta, M. (2021). A novel approach for code smell detection: An empirical study. *IEEE Access*, 9:162869–162883.

[Hamdi et al. 2021] Hamdi, O., Ouni, A., AlOmar, E. A., and Mkaouer, M. W. (2021). An empirical study on code smells co-occurrences in android applications. pages 26–33.

[Imparato et al. 2017] Imparato, A., Maietta, R. R., Scala, S., and Vacca, V. (2017). A comparative study of static analysis tools for autosar automotive software components development. pages 65–68.

[J. Garcia and Chen 2020] J. Garcia, Y. Feng, J. S. S. A. Y. X. and Chen, Q. A. (2020). A comprehensive study of autonomous vehicle bugs. *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 385–396.

[Luo et al. 2022] Luo, Y., Zhang, X.-Y., Arcaini, P., Jin, Z., Zhao, H., Ishikawa, F., Wu, R., and Xie, T. (2022). Targeting requirements violations of autonomous driving systems by dynamic evolutionary search (hop at gecco'22). page 33–34.

[Martins et al. 2020a] Martins, J., Bezerra, C., Uchôa, A., and Garcia, A. (2020a). Are code smell co-occurrences harmful to internal quality attributes? a mixed-method study. page 52–61.

[Martins et al. 2020b] Martins, J., Bezerra, C., Uchôa, A., and Garcia, A. (2020b). Are code smell co-occurrences harmful to internal quality attributes? a mixed-method study. page 52–61.

[Plosch et al. 2008] Plosch, R., Gruber, H., Hentschel, A., Pomberger, G., and Schiffer, S. (2008). On the relation between external software quality and static code analysis. pages 169–174.

[Rasool and Arshad 2016] Rasool, G. and Arshad, Z. (2016). A lightweight approach for detection of code smells. *Arabian Journal for Science and Engineering*, 42.

[Sharma 2018] Sharma, T. (2018). Detecting and managing code smells: Research and practice. pages 546–547.

[Sharma and Kessentini 2021] Sharma, T. and Kessentini, M. (2021). Qscored: A large dataset of code smells and quality metrics. pages 590–594.

[Stocco et al. 2020] Stocco, A., Weiss, M., Calzana, M., and Tonella, P. (2020). Misbehaviour prediction for autonomous driving systems. page 359–371.

[Tang et al. 2021] Tang, S., Zhang, Z., Tang, J., Ma, L., and Xue, Y. (2021). Issue categorization and analysis of an open-source driving assistant system. pages 148–153.

[Ya-hong et al. 2013] Ya-hong, L., Jian, L., and Ke-gang, H. (2013). The software project progress measurement frame based on gqm model. pages 133–138.