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
Instituto de Ciências Exatas e Informática - PUC Minas
Edifício Fernanda, Rua Claudio Manoel, 1.162, Funcionários
30.140-100 — Belo Horizonte — MG — Brasil

lorrayne.silva.1220819@sga.pucminas.br

Abstract. *Autonomous driving systems have gained prominence in the recent industrial market, serving as precursors to autonomous vehicles that promote driving without human interference. As a result, repositories for simulating these systems are frequently used in development and testing, and the reliability of the simulations may be directly related to qualitative factors. Due to the scarcity of studies addressing this subject, this research aims to understand the qualitative relationships presented by this segment through the analysis of code smells and quality attributes in 200 repositories written in the Python language. The presence of three main code smells and a higher occurrence of the Halstead metric, which measures development effort, was observed in the analyzed repositories.*
Keywords: *Autonomous Driven Systems, Code Smells, Python*

Resumo. *Sistemas de direção autônoma têm ganhado destaque no mercado industrial recente, sendo precursores para a atuação de veículos autônomos, que promovem uma condução sem interferência humana. Assim, repositórios para a simulação desses sistemas são frequentemente usados no desenvolvimento e na realização de testes, e a confiabilidade das simulações pode estar diretamente relacionada a fatores qualitativos. Devido à escassez de estudos que abordam esse assunto, o presente trabalho busca compreender as relações qualitativas apresentadas por esse segmento, por meio da análise de "code smells"e atributos de qualidade em 200 repositórios escritos na linguagem Python. Observou-se a presença de três principais "code smells"e uma maior ocorrência da métrica Halstead, que mensura o esforço de desenvolvimento, nos repositórios analisados.*
Palavras-chave: *Sistemas de Direção Autônoma, Code Smells, Python*

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@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 *ADSs*, 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 [J. 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 à 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 ADS [Stocco et al. 2020]. Sharma (2018) apresenta um *dataset* identificativo de *code smells* relacionados a erros decorrentes de métricas de qualidade, coletadas de repositórios do *GitHub* [Sharma 2018]. 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 e Chen (2020) realizaram análises em sistemas AVs através da coleta de 499 *bugs* por meio de *commits* de sistemas de *software* de código aberto [Chen et al. 2019]. Além disso, eles 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. Por outro lado, 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* [Tang et al. 2021]. Os autores também promoveram uma caracterização de *bugs* que ocorrem recorrentemente e refletem na abertura de *issues* com características semelhantes. Logo, resolver o problema deste trabalho é importante, pois outras abordagens da literatura não apontam a necessidade atencional da relação entre os *softwares* 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 dos *code smells* mais encontrados em repositórios simuladores de *ADSs*, 2) Relação entre métricas qualitativas e a ocorrência dos *code smells* classificados 3) Comparação e correlação da tendência quantitativa de *code smells* e métricas de acordo com a presença ou não de *releases* nesses repositórios. Para alcançar os objetivos deste

trabalho, foram propostas três questões de pesquisa. As questões são apresentadas a seguir. RQ1: Quais são os code smells frequentemente encontrados em repositórios de ADSs? RQ2: Qual é a relação entre métricas de qualidade e code smells em repositórios de ADSs? RQ3: Qual é a relação entre repositórios de ADSs que possuem releases e as diferenças nos valores das métricas e code smells em comparação com repositórios de ADSs que não possuem releases?

Como resultado, foram pontuados os *code smells* frequentemente atrelados a repositórios *ADSs*, com o *code smell Long Method*, 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 *softwares*, com o *code smell Long Method* novamente se destacando no que concerne a maior ocorrência de métricas como Complexidade Ciclomática. Ademais, foi caracterizada a tendência numérica de ocorrência de *code smells* relacionados a repositórios que não possuem *releases* em comparação àqueles que possuem, onde foi verificado que em ambos tipos de repositórios com *Long Method* apresenta-se com maior ocorrência, em aproximadamente 70% dos repositórios. E que o fato de um repositório apresentar *releases* ou não, não se relaciona diretamente a valores efetivos de mudança de métricas.

As próximas seções do trabalho são: Seção 2 apresenta a Fundamentação Teórica. Seção 3 apresenta os artigos relacionados. Seção 4 apresenta os Materias e Métodos utilizados no trabalho. Seção 5 apresenta as Definições do Estudo. Seção 6 apresenta os Resultados. Seção 7 apresenta as Discussões e Ameaças a Validade. Por fim a Seção 8 detalha a Conclusão e Trabalhos Futuros.

2. Fundamentação Teórica

2.1. Qualidade de *Software*

Segundo Ian Sommerville (2011), a qualidade de *software* se estabelece principalmente em seu modo ocorrente de execução, estruturação e organização. Tais características se refletem nos atributos de qualidade de um *software*, como robustez, complexidade e modularidade [Sommerville 2010]. Por meio da utilização de padrões durante o desenvolvimento, é possível promover propriedades determinísticas que facilitam testes e manutenção em produtos gerados. Além disso, os padrões podem estabelecer um relacionamento direto na criação de relações de dependência, o que pode indicar problemas profundos no código [Chen et al. 2019].

2.2. Análise Estática de *Software*

Análises estáticas em *software* referem-se à avaliação do código fonte estático com o intuito de verificar a existência de possíveis padrões errôneos, violações de programação e identificar paradigmas de qualidade no código [Plosch et al. 2008]. Os resultados dessas análises podem demonstrar o relacionamento direto entre o código desenvolvido e a qualidade externa demonstrada pelo *software* [Plosch et al. 2008]. Além disso, essas análises podem indicar características qualitativas relacionadas ao consumo de recursos durante a execução de programas [Chen et al. 2021]. Por fim, elas são recomendadas pela ISO 26262/2011 para reduzir a ocorrência de *bugs* em tempo de execução, fazendo parte dos padrões de segurança automotiva no desenvolvimento de *software* [Imparato et al. 2017].

2.3. *Code Smells*

Code smells são deficiências na implementação ou escolhas de *design* que têm um impacto negativo na evolução de *softwares*, dificultando a compreensão e manutenção das aplicações [Hamdi et al. 2021]. Eles podem indicar problemas na qualidade do desenvolvimento e, ao mesmo tempo, ajudar na identificação de possíveis erros, além de facilitar a refatoração por parte da equipe técnica [Sharma 2018]. Ao identificar e tratar os *code smells*, é possível aumentar a qualidade do *software*, torná-lo mais fácil de ser mantido e reduzir as chances de falhas no sistema [Dewangan et al. 2021].

2.4. Sistemas de Direção Autônoma

Sistemas de direção autônoma são sistemas complexos que possuem requisitos de segurança crítica a serem atendidos [Luo et al. 2022]. Esses sistemas recebem dados de vários sensores, que são analisados em tempo real por meio de redes neurais profundas. Essas redes neurais são responsáveis por determinar parâmetros para o acionamento dos atuadores [Stocco et al. 2020]. Os softwares desses sistemas desempenham um papel fundamental em sua operação e são compostos por várias unidades de componentes que automatizam tarefas de direção específicas. Eles estão diretamente envolvidos na execução e funcionamento do sistema, solicitando recursos continuamente e trocando dados com componentes de outras camadas [Abdessalem et al. 2020].

3. Trabalhos Relacionados

Nesta seção, apresentam-se os trabalhos relacionados que estão ligados ao objeto de estudo do presente trabalho. Para demonstrar a relação direta entre a ocorrência de code smells e os atributos de qualidade de um software, Martins et al. (2020) realizaram uma avaliação em dois softwares de código fechado ao longo de 11 releases. Eles verificaram a presença de code smells independentes e estabeleceram relações de co-ocorrência entre eles [Martins et al. 2020a]. Além disso, avaliaram métricas de qualidade, como coesão, acoplamento e complexidade, antes e depois da remoção seletiva de code smells. Os resultados relevantes desse estudo mostraram que o refactoring desses code smells resultou em uma redução significativa da complexidade do projeto e um aumento nos valores das métricas de coesão e acoplamento [Martins et al. 2020a]. Portanto, esse trabalho se relaciona ao presente estudo, principalmente em relação à metodologia utilizada para verificar a 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*) discorrem 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* para verificar o impacto em relação a manutenibilidade utilizando métricas como Número de filhos (NOC, do inglês *Number of Children*). 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* [Martins et al. 2020b].

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 revelam a predominância de maior ocorrência dos tipos *Large Class*, referente a uma classe que cresceu muito em relação ao número de linhas de código, e *Large Method*, sendo definido como um método muito longo [Chen et al. 2016]. Logo, o trabalho relaciona-se com o mesmo intuito investigativo, através da verificação de *code smells* que ocorrem majoritariamente em *softwares* Python de *ADSs*.

4. Materias e Métodos

4.1. Etapas do Experimento

Foram realizadas 3 etapas divididas em coleta de dados, identificação de métricas & *smells*, sumarização e padronização. Com o objetivo de analisar 200 repositórios escritos na linguagem *Python*, que contribuem com prerrogativas simulatórias para sistemas autônomos por meio do *GitHub*. A partir de *scripts*, 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.

4.2. Coleta de dados

A coleta dos repositórios ocorreu atráves da *API* (do inglês, *Application Programming Interface*) *GraphQL* do *Github*. Para a estruturação do conjunto de dados foram sele-

cionados 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*. Uma vez que as tecnológicas utilizadas no desenvolvimento desses programas sofrem constante mudança e tal ação permite a reutilização do código de coleta. 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. Posteriormente foi realizada em código a verificação da presença de *releases* em cada repositório coletado e gerados CSV's separados entre repositórios que possuem *releases* e aqueles que não possuem.

4.3. Identificação de Métricas e *Code Smells*

Para a realização da investigação de *code smells* e métricas, foram coletadas diferentes métricas geradas por meio das bibliotecas *Radon* e *Multimetric*. A primeira foi escolhida por fornecer informações como comentários e tamanho de código como visto na Tabela 1 a definição de *LOC*. A segunda por fornecer uma variedade de métricas que podem apontar propriedades mensuráveis do *software*, como visto na Tabela 2, onde se verificam as métricas *Halstead*, que medem a complexidade de módulos de um programa e fornecem indicadores de complexidade por meio de métricas como *Halstead Effort* e *Halstead Difficulty*.

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

Para responder das questões RQ1, RQ2 e RQ3 foram estabelecidas definições para verificação dos *smells* presentes em código. Para responder RQ1, os *smells Large Class* e *Long Method* são apontados por Chen et al.(2016), respectivamente 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. Em seguida é checado o total de comentários presentes em código sendo 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] como: *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*, foram observadas as relações existentes entre a decorrência de cada tipo de *smell* e o valor numérico das métricas. 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. Para responder RQ3 com viés quantitativo, 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.

4.4. Sumarização e Padronização

Após a coleta das métricas, ocorreu a sumarização dos *code smells* mais frequentes encontrados nos repositórios, bem como a atribuição de pontuações numéricas às métricas qualitativas predominantes. Essas etapas foram seguidas pela realização de análises finais. Para investigar a relação entre os *code smells* e as métricas pontuadas, foram realizadas correlações *Pearson* utilizando bibliotecas da linguagem Python. A correlação *Pearson* foi escolhida porque, como mostrado por Schober et al. (2019) [Sch 2018], os valores dessa correlação variam de -1 a 1. Valores positivos indicam uma relação direta entre as variáveis, ou seja, correlações positivas completas, enquanto valores negativos indicam correlações negativas completas, o que pode indicar que as variáveis não estão relacionadas.

Foram elaborados gráficos utilizando o software *PowerBI*, que facilita a visualização dos dados. Esses gráficos incluíram comparações percentuais da ocorrência de *code smells* comumente encontrados em *ADSs* e as métricas coletadas. Para facilitar a interpretação dos dados, foram utilizadas siglas correspondentes aos nomes das métricas, como *Comment Ratio* (*CR*), *Cyclomatic Complexity* (*CC*), *Halstead Effort* (*HE*), *Halstead Difficulty* (*HD*), *Halstead Time Required* (*HT*), *Halstead Volume* (*HV*) e *Halstead Bugprop* (*HB*). Esses gráficos proporcionaram uma visualização clara das relações numéricas entre os *code smells* e as métricas analisadas.

5. Resultados

5.1. RQ1: Quais os *code smells* frequentemente encontrados em repositórios de ADSs?

Após a sumarização dos resultados, identificamos três *code smells* frequentemente encontrados em sistemas *ADSs*. O *code smell* mais prevalente foi o *Long Method*, com 5.550.865 ocorrências. Em seguida, observamos o *code smell Comments*, com 1.541.942 ocorrências. Por fim, o *code smell Large Class* teve a menor quantidade de ocorrências, com 737.476 registros. Esses dados evidenciam os *code smells* mais comuns encontrados nos sistemas analisados.

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

Com base na análise realizada, foram identificados 95 repositórios que possuem o *code smell Comment smells*, enquanto 112 repositórios não apresentam esse *smell*. A Figura 1 ilustra a diferença entre esses dois grupos. Nos repositórios que possuem o *Comment smells* (Figura 1A), é possível observar valores elevados de Complexidade Ciclomática, indicando um grande número de estruturas de decisão no código. Além disso, esses repositórios apresentam um alto valor de *Fan-out*, indicando um grande número de módulos que são utilizados pelo código. Por outro lado, nos repositórios que não possuem o *Comment smells* (Figura 1B), é observada uma variação nos valores. Nesses casos, nota-se uma diminuição no valor de *Comment Ratio*, que representa a taxa de comentários no código, e um aumento considerável no valor de Complexidade Ciclomática. Essas observações sugerem que a presença do *code smell Comment smells* está associada a um maior número de estruturas de decisão no código. Por outro lado, a ausência desse *smell* pode levar a uma diminuição na taxa de comentários e um aumento na complexidade do código.

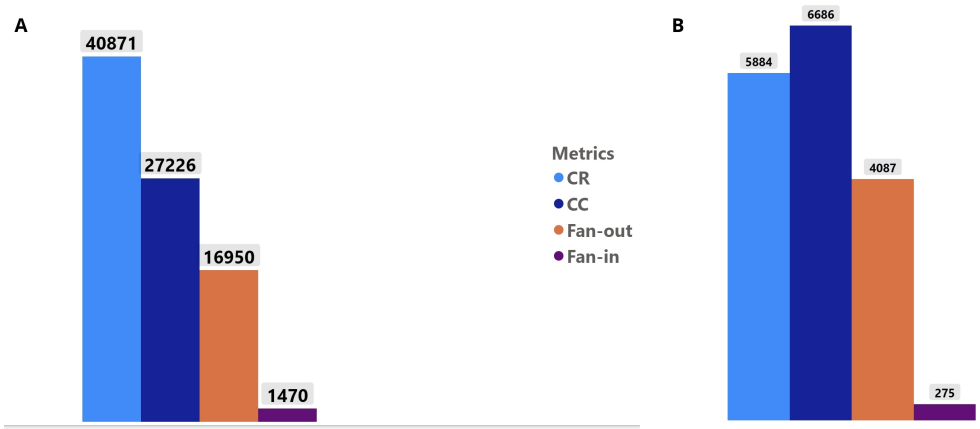

Figura 1. Métricas relacionadas a repositórios que apresentam *Comment smells* A e não apresentam esse *smell* B

Ao analisar as métricas *Halstead* relacionadas aos repositórios que possuem e não possuem o *code smell Comment smells*, conforme ilustrado na Figura 2, pode-se observar que não há uma alternância significativa nos valores das métricas entre esses dois grupos. Na Figura 2A, que representa os repositórios com *Comment smells*, é possível observar que as métricas *Halstead* apresentam valores variados, sem uma tendência clara de altos ou baixos valores. Por sua vez, na Figura 2B, que retrata os repositórios sem *Comment smells*, também não é observada uma padronização nos valores das métricas. Destaca-se que a métrica *Halstead Effort* se destaca em ambos os grupos, indicando um alto esforço necessário para manter esses repositórios, independentemente da presença ou ausência do *Comment smells*. Essa métrica está relacionada à complexidade do código e à quantidade de esforço requerida para entendê-lo e mantê-lo. Portanto, com base nessas observações, conclui-se que a presença ou ausência do *Comment smells* não parece influenciar diretamente os valores das métricas *Halstead* nos repositórios analisados. Outras métricas e fatores podem estar mais relacionados à variação desses valores.

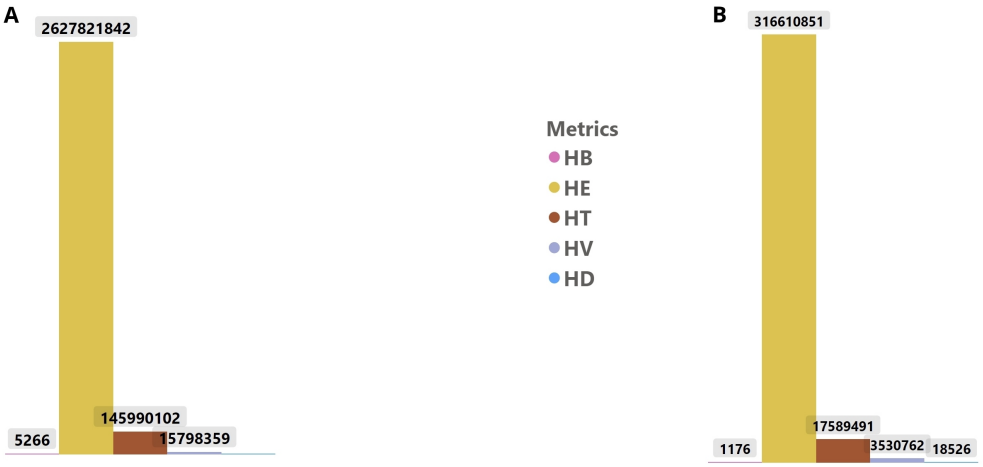

Figura 2. Métricas Halstead relacionadas a repositórios com *Comment smells* A e sem esse *smell* B

A Figura 3 demonstra as métricas relacionadas a repositórios que possuem o *smell Large Class* (Figura 3 A) , onde é possível observar que repositórios que possuem *Large Class* apresentam alta diferença entre os valores de Complexidade Ciclomática e *Fan-out*, diferentemente de (Figura 3 B) que demonstra os repositórios que não possuem o *smell Large Class*.

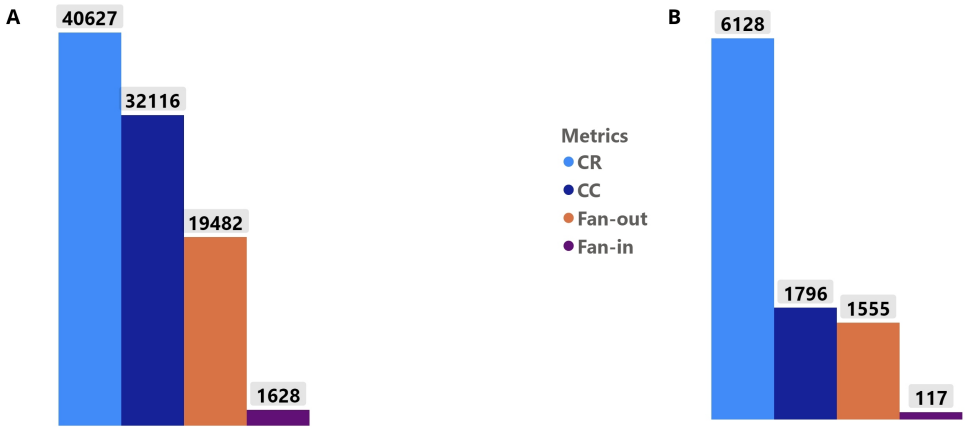

Figura 3. Métricas relacionadas a repositórios que possuem *Large Class* A e não possuem esse *smell* B

Figura 4 apresenta as métricas *Halstead* relacionadas aos repositórios que possuem *Large Class* (Figura 4 A) e em Figura 4 B 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*.

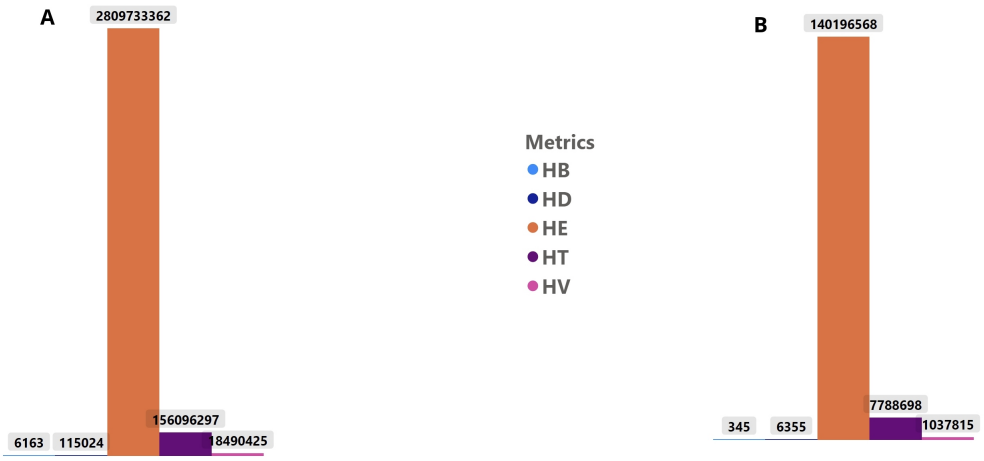

Figura 4. Métricas Halstead relacionadas a repositórios que possuem *Large Class smells* A e não possuem esse *smell* B

Figura 5 apresenta repositórios que possuem *code smells* do tipo *Long Method* (Figura 5 A) e os repositórios que não apresentam esse tipo de *smell* (Figura 5 B), apresentando altos valores de *Comment Ratio*, conjuntamente é possível observar Complexidade Ciclomática apresenta considerável valor alto. 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.

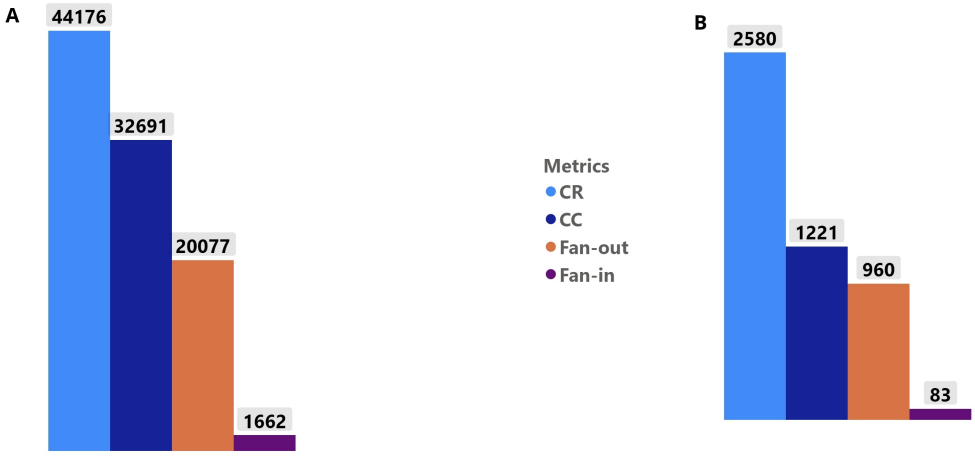

Figura 5. Métricas relacionadas a repositórios que possuem *Long Method smells* A e não possuem B

Em relação as métricas *Halstead* na Figura 6 são observadas as métricas *Halstead* relacionadas a repositórios que possuem o *smell Long Method* Figura 6 A e em Figura 6 B as métricas relacionadas a repositórios que não possuem, onde se nota novamente *Halstead Effort* como principal métrica em repositórios que possuem ou não *smell Long*

Method.

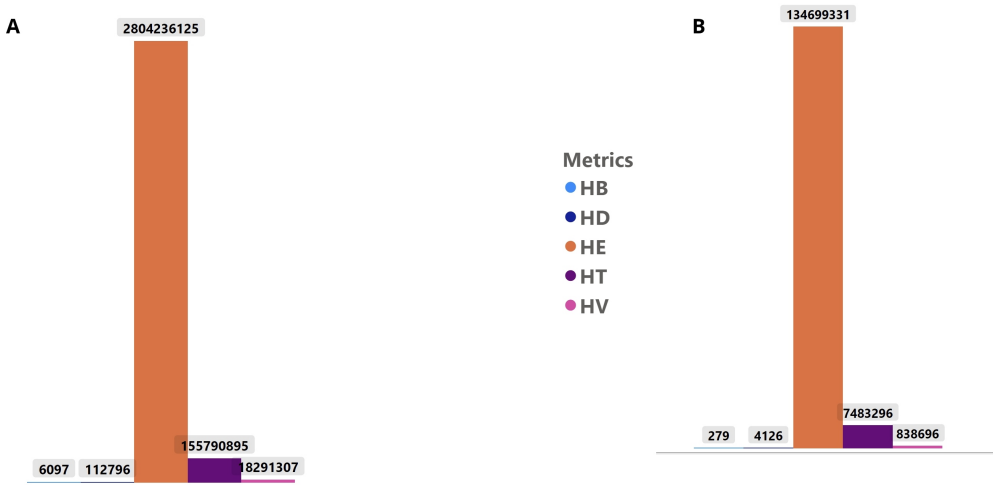

Figura 6. Métricas Halstead relacionadas a repositórios que possuem *Long Method smells* A e não possuem B

A partir das correlações *Pearson* realizadas, por meio do cálculo da ocorrência de smells e dos valores de métricas relacionadas a esses, é possível notar na Figura 7 que *CR* possui valores próximos 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.

Figura 8 mostra as métricas verificadas por meio de repositórios que não apresentam um dos tipos *smells*. No qual observa-se a presença de relacionamentos fortes e completos positivos 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.

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

A partir da coletas, classificação de *smells* e *releases*, é possível verificar que em ambos tipos de repositórios, o número de *code smells* do tipo *Long Method* possui maior ocorrência, assim como pode ser visto na Figura 9 que referencia a porcentagem de *smells* relacionados a repositórios que possuem *releases* Figura 9 A e que não possuem Figura 9 B.

A respeito 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 10 A. Semelhantemente ao que pode ser visto em Figura 10 B, que representa a porcentagem

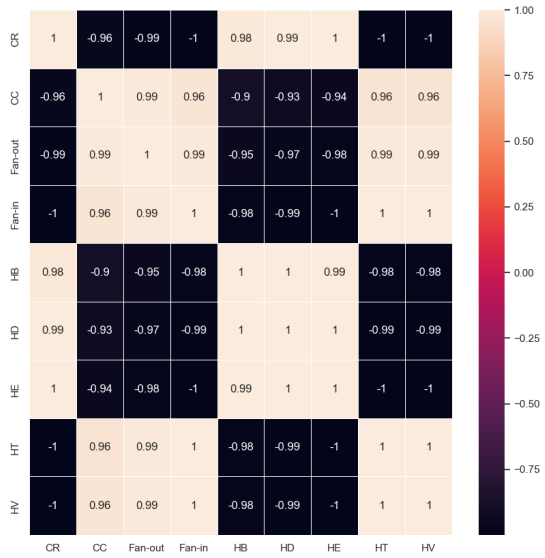

Figura 7. Correlação *Pearson* entre métricas e ocorrência de *smells* em repositórios

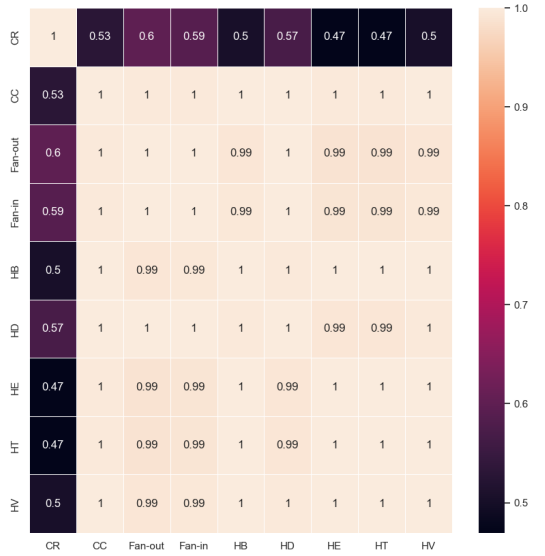

Figura 8. Correlação *Pearson* entre métricas e não ocorrência de *smells*

de métricas relacionadas a repositórios que não possuem *releases*, onde esses apresentam maior valor de *Comment Ratio* seguidamente de Complexidade Ciclomática.

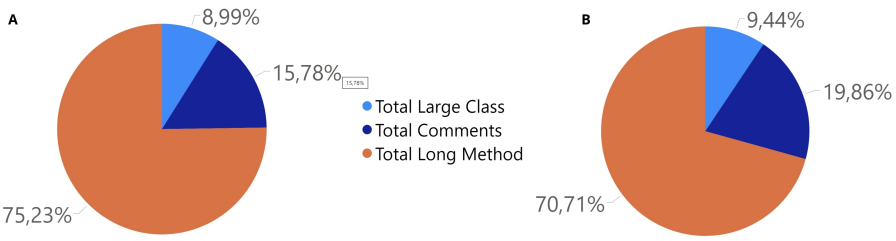

Figura 9. Porcentagem de smells relacionados a repositórios que possuem releases

Figura 10. Porcentagem de métricas relacionadas a repositórios que possuem releases

Figura 11 apresenta a porcentagem de métricas *Halstead* relacionadas a repositórios que possuem *releases* Figura 11 A e em Figura 11 B 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*.

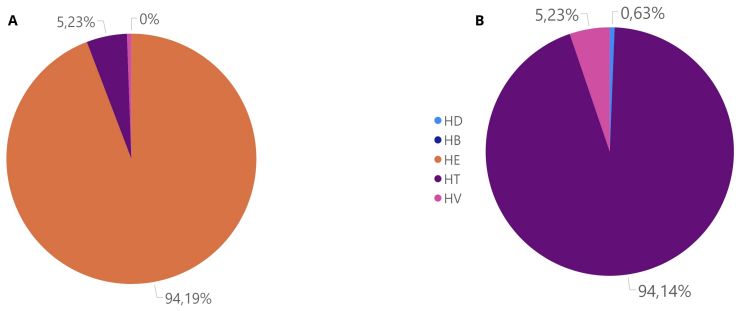

Figura 11. Porcentagem de métricas Halstead relacionadas a repositórios que possuem releases

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 12. 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 com-

pletas positivas e negativas, demonstram um quadro não direto de linealidade de relacionamento.

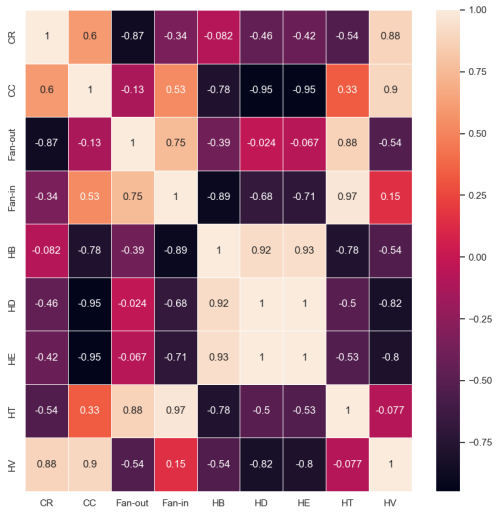

Figura 12. Correlação *Pearson* entre métricas e ocorrência de smells em repositórios que apresentam *releases*

Em repositórios que não apresentam *releases* como mostrado na Figura 13, verificam-se relacionamentos positivos fortes entre métricas como CC, Fan-out e Fan-in, respectivamente com valores relacionais de 0.98 e 0.72.

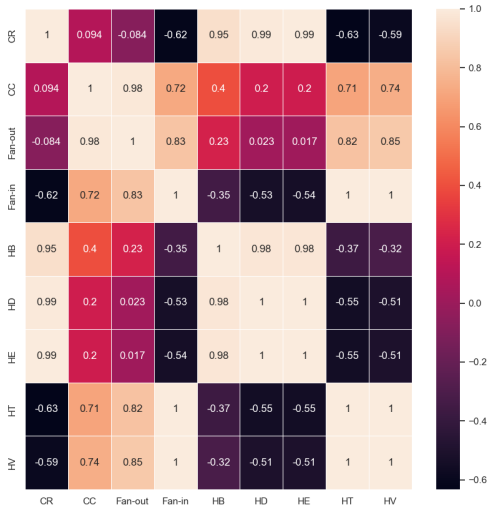

Figura 13. Correlação *Pearson* entre métricas e ocorrência de *smells* em repositórios que não apresentam *releases*

6. Discussões e Ameaças a Validade

Com base nos estudos realizados, pode-se inferir uma 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. 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. Como validade externa, o presente estudo aponta especificamente sobre *code smells* relacionados ao nicho de repositórios *ADSs* relacionados a linguagem *Python* e as conclusões apresentadas não devem ser estendidas a outros nichos.

Adentradamente, a metodologia promulgada inicialmente por esse 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.

7. Conclusão e Trabalhos Futuros

Foram descobertos três *code smells* frequentemente relacionados a esses repositórios: *Large Class*, *Long Method* e *Comments*. Onde é possível verificar que para repositórios que apresentam e não apresentam determinado *smell* a métrica Complexidade Ciclomática possuiu alta diferenciação quantitativa, assim é notório que a ocorrência e não ocorrência de *smells* estão associados a disparidades nos valores das métricas. 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 da métrica *Halstead Effort*.

A partir da averiguação de repositórios em relação a presença ou não de *releases*, observa-se que independentemente do tipo de repositório métricas como *Comment Ratio*, *Complexidade Ciclomática* e *Fan-out* possuem altos valores. Igualmente como ocorrem em relação a métrica *Halstead Effort*, que se apresenta proeminente em ambos percentuais. Logo as *releases* se apresentam como não diretamente relacionadas no aumento ou decremento do percentual métricas e *smells*.

Ao final, avalia-se por meio dos repositórios coletados que esses necessitam de atenção quanto a códigos desenvolvidos e em desenvolvimento, dada a crescente atuação de *softwares ADSs*. 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* no intuito de promover uma análise qualitativa para verificar se esses *smells* empregam efeitos negativos em código.

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.

[Sommerville 2010] Sommerville, I. (2010). *Software Engineering*. Addison-Wesley Publishing Company, USA, 9th edition.

[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.