Impactos da Valência Emocional na Qualidade de Código

Diogo Martins de Assis[1], **Renato Paganini Thürler Filho**[2]

[1]Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
Rua Cláudio Manoel, 1.162, Funcionários, Belo Horizonte – MG - Brasil

{diogo.assis, renato.filho.1094209}@sga.pucminas.br

Abstract. *Natural Language Processing (NLP) is an area of artificial intelligence that studies the interaction between computers and human language, enabling the automatic analysis, generation and understanding of texts. Sentiment analysis applies NLP techniques to classify emotional polarities in textual data, optimizing the extraction of subjective insights. However, the lack of understanding about the impact of emotional valence on code quality is a problem. This study seeks to analyze the impact of emotional valence on the code quality of open source repositories on GitHub. In addition, the study correlates code quality with emotional valence with specific statistical methods in order to provide valuable results for developers on the influence of emotions expressed in pull requests on the quality of the code produced. The results, obtained from the analysis of 94 repositories, indicated a predominance of neutral comments and weak correlations between variables. Statistical significance tests showed no robust evidence that emotional valence directly influences metrics such as bugs, code smells, or cognitive complexity in the analyzed projects.*
Keywords: *Natural Language Processing, Code Quality, Sentiment Analysis*

Resumo. *O Processamento de Linguagem Natural (PLN) é uma área da inteligência artificial que estuda a interação entre computadores e linguagem humana, permitindo a análise, geração e compreensão automática de textos. A análise de sentimentos aplica técnicas de PLN para classificar polaridades emocionais em dados textuais, otimizando a extração de insights subjetivos. Porém, a falta de entendimento sobre o impacto da valência emocional na qualidade do código é um problema. Este estudo busca analisar o impacto da valência emocional na qualidade do código de repositórios de código aberto no GitHub. Além disso, o estudo correlaciona a qualidade do código com a valência emocional com métodos estatísticos específicos, a fim de fornecer resultados valiosos para os desenvolvedores sobre a influência das emoções expressas em pull requests na qualidade do código produzido. Os resultados, obtidos a partir da análise de 94 repositórios, indicaram uma predominância de comentários neutros e correlações fracas entre as variáveis. Os testes de significância estatística não apresentaram evidências robustas de que a valência emocional influencie diretamente métricas como bugs, code smells ou complexidade cognitiva nos projetos analisados.*
Palavras-chave: *Processamento de Linguagem Natural, Qualidade de Código, Análise de Sentimentos*

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

Orientador de conteúdo (TCC I): Joana Souza - joanasouza@pucminas.br
Orientador de conteúdo (TCC I): João Pedro Batisteli - jp.batisteli@hotmail.com
Orientador de conteúdo (TCC I): Leonardo Vilela - leonardocardoso@pucminas.br
Orientador acadêmico (TCC I): Cleiton Tavares - cleitontavares@pucminas.br
Orientador do TCC II: Michelle Nery Nascimento - michellenery@pucminas.br
Belo Horizonte, 12 de dezembro de 2025.

1. Introdução

O reconhecimento precoce de emoções negativas, como estresse, frustração e raiva, permite que ações corretivas sejam tomadas por desenvolvedores ou gerentes para que a equipe ou projeto não sejam gravemente prejudicados [Girardi et al. 2020]. Emoções negativas podem ter efeitos prejudiciais na qualidade do código, no desempenho cognitivo e na produtividade geral. Elas podem levar ao afastamento do trabalho e a uma diminuição geral da eficiência. Embora emoções negativas, como a raiva induzida por conflitos, possam estimular a capacidade de resolução de problemas, há consenso na comunidade da Engenharia de Software que minimizar experiências emocionais negativas é fundamental para manter a saúde dos desenvolvedores e da equipe [Michels et al. 2024].

No contexto de desenvolvimento de código aberto, Miller et al. (2022) apontam que comentários negativos ou tóxicos sobre problemas podem prolongar o tempo de resolução. Além disso, a insatisfação e a falta de apoio entre colegas, de maneira geral, aumentam o risco de desengajamento e criam um ambiente hostil. Emoções positivas são benéficas para o bem-estar e a produtividade dos desenvolvedores, enquanto as negativas levam a um baixo desempenho no trabalho e são prejudiciais ao processo de desenvolvimento de software [Girardi et al. 2020]. Contudo, a literatura existente, embora robusta ao apontar os efeitos no ambiente e na produtividade, **ainda é incipiente em fornecer evidências empíricas que conectem diretamente a valência emocional das interações a atributos internos de qualidade de código**, como a incidência de *bugs*, *code smells* e complexidade cognitiva. Essa lacuna constitui o problema de pesquisa central deste estudo, pois, sem essa correlação quantitativa, o impacto das emoções na qualidade técnica do software permanece no campo da suposição, impedindo a criação de estratégias preventivas mais eficazes.

A toxicidade, entendida como o uso de linguagem rude, desrespeitosa ou irracional que tende a afastar e antagonizar, é um problema significativo no ambiente virtual, tem impactos negativos na saúde dos desenvolvedores e membros de comunidades virtuais [Miller et al. 2022]. Por isso, detectar a toxicidade nas interações de revisão de código é crucial não apenas para melhorar a qualidade do software, mas também para promover um ambiente mais inclusivo e sustentável para todos os desenvolvedores.

O objetivo geral deste trabalho é analisar empiricamente a influência das valências emocionais manifestadas nos comentários de revisão de código por meio de Processamento de Linguagem Natural e seus possíveis impactos na qualidade do código. Para o reconhecimento de emoções em texto, utiliza-se o XLM-T [Barbieri et al. 2022], um modelo multilíngue treinado para essa finalidade. Para avaliar a qualidade do código, são consideradas quatro métricas principais: *code smells*, com-

plexidade cognitiva, *bugs* e dívida técnica. Os objetivos específicos são: (i) identificar e categorizar as valências emocionais expressas nos comentários de *code review* por meio de técnicas de Processamento de Linguagem Natural; (ii) extrair e catalogar as métricas de qualidade do código obtidas pelo SonarQube, especificamente: *bugs*, *code smells*, dívida técnica e complexidade cognitiva; e (iii) analisar a correlação entre a intensidade das emoções nos comentários de revisão e as métricas de qualidade do código.

Ao responder a essas questões, este trabalho visa contribuir para a expansão do entendimento do assunto com a abordagem proposta. Para tanto, são analisados comentários de revisão em repositórios GitHub, utilizando modelos de Processamento de Linguagem Natural para identificar padrões emocionais e sua correlação com métricas técnicas de qualidade.

O restante deste trabalho está estruturado da seguinte maneira. A Seção 2 apresenta detalhes sobre os conceitos pertinentes ao estudo, abordando emoções no desenvolvimento de software, revisão de código e métricas de qualidade. A Seção 3 descreve os trabalhos relacionados ao tema, destacando pesquisas prévias sobre análise de sentimentos. A Seção 4 apresenta os materiais e métodos propostos, incluindo a coleta de dados, ferramentas de análise e técnicas estatísticas aplicadas. Os resultados e discussões são apresentados na Seção 5, evidenciando as relações entre emoções em revisões de código e métricas de qualidade. Por fim, as conclusões e ideias para trabalhos futuros são detalhadas na Seção 6, consolidando as contribuições desta pesquisa e sugerindo direções para investigações adicionais, além das principais limitações e ameaças à validade do estudo.

2. Fundamentação Teórica

Nesta seção, são apresentados os conceitos fundamentais para a compreensão do trabalho. Primeiramente, a Seção 2.1 trata do conceito de qualidade de código. Por fim, a Seção 2.2 discute o conceito de valência emocional.

2.1. Qualidade de Código

A qualidade do código tem um impacto significativo na usabilidade e efetividade do software. É importante que o código seja fácil de entender, manter e modificar. A ISO/IEC 25010 (2011) define a qualidade de software como "um conjunto de características e atributos do software que se relacionam com o grau em que o software satisfaz as necessidades explícitas e implícitas, quando usado em condições especificadas". Essa definição ressalta a importância de considerar diversos aspectos ao avaliar a qualidade de um software, incluindo a qualidade do código-fonte, pois este impacta de forma palpável tanto na manutenção e evolução quanto no sucesso geral do sistema.

No contexto específico do código-fonte, a qualidade pode ser definida como a capacidade do código em atender aos requisitos funcionais e não funcionais do software, bem como à sua facilidade de manutenção, extensibilidade e legibilidade. De acordo com Sommerville (2019), a qualidade do software está diretamente relacionada a características como modularidade, simplicidade e facilidade de compreensão, que são essenciais para garantir a manutenção eficiente e o desenvolvimento evolutivo do sistema.

A avaliação da qualidade do código pode ser realizada com base nas características de qualidade de software estabelecidas pela norma ISO/IEC 9126-1 (2001) propõe um conjunto de dimensões essenciais para esse fim. A funcionalidade refere-se à capacidade

do software em fornecer as funções esperadas, atendendo às exigências dos usuários. A confiabilidade diz respeito à habilidade do sistema em operar de forma estável e livre de falhas durante sua execução. A usabilidade considera o quão intuitiva e acessível é a interação do usuário com o sistema, valorizando a experiência de uso. A eficiência avalia o desempenho do software em relação ao uso de recursos como tempo de processamento e memória. Já a manutenibilidade aponta para a facilidade com que o software pode ser modificado, corrigido ou aperfeiçoado ao longo do tempo. Por fim, a portabilidade trata da aptidão do software em ser transferido e operado em diferentes ambientes tecnológicos sem comprometer sua funcionalidade ou desempenho. Essas dimensões fornecem uma base sólida para a análise e aprimoramento da qualidade do código ao longo do ciclo de vida do software.

Ao adotar as características de qualidade, os desenvolvedores têm um guia abrangente para avaliar e aprimorar a qualidade do código. Além disso, tais características podem ser desdobradas em métricas específicas (coesão, acoplamento, complexidade ciclomática, etc.), que podem ser utilizadas para avaliar o código-fonte. Ao se incorporar esses princípios ao projeto do código, os desenvolvedores ganham uma visão mais refinada das áreas que requerem atenção, permitindo a identificação de pontos específicos para otimização e melhoria da qualidade geral.

2.2. Processamento de Linguagem Natural (PLN)

O Processamento de Linguagem Natural (PLN), ou *Natural Language Processing* (NLP), é uma subárea da Inteligência Artificial dedicada a investigar a interação entre computadores e a linguagem humana. O objetivo central do PLN é desenvolver modelos computacionais capazes de compreender, interpretar e gerar linguagem natural de forma semelhante aos seres humanos [Manning and Schutze 1999].

No contexto da Engenharia de Software, técnicas de PLN têm sido amplamente adotadas para automatizar tarefas que envolvem dados não estruturados, como a análise de requisitos, documentação de software e, especificamente, a análise de sentimentos em repositórios de código. A aplicação dessas técnicas permite transformar grandes volumes de texto (como comentários de *code review*) em dados estruturados passíveis de análise estatística, viabilizando a extração de métricas subjetivas, como a valência emocional, que seriam inviáveis de serem coletadas manualmente em larga escala.

2.3. Valência Emocional

A valência emocional é uma das principais dimensões utilizadas para representar a carga afetiva de palavras no campo do Processamento de Linguagem Natural (PLN). Conceitualmente, a valência refere-se ao grau de agradabilidade ou desagradabilidade transmitido por uma palavra ou expressão, situando-se em um contínuo entre emoções positivas e negativas. De acordo com Recchia e Louwerse (2015), a valência pode ser entendida como "o grau de prazer associado a um estímulo linguístico", estando intimamente relacionada à forma como os indivíduos avaliam palavras em termos das emoções que evocam.

Esse conceito tem sido amplamente explorado em estudos que investigam os mecanismos cognitivos envolvidos no reconhecimento lexical, na atenção emocional e na categorização afetiva de conteúdos linguísticos. Essa abordagem é justificada pelo fato de que as palavras, além de transmitirem significados objetivos, também despertam respostas

emocionais subjetivas, influenciando diretamente a forma como são processadas e compreendidas. No âmbito do PLN, a consideração da valência emocional é especialmente relevante, uma vez que os textos carregam tanto informações factuais quanto afetivas, impactando tarefas como análise de sentimentos, geração de linguagem natural, sistemas de recomendação e classificação de opiniões.

Como observam Manning e Schutze (1999), os sistemas de linguagem natural precisam lidar com dinâmicas que vão além da denotação – abrangendo aspectos pragmáticos e emocionais da linguagem. Dessa forma, incorporar a valência emocional aos modelos computacionais não apenas enriquece a representação semântica, como também torna esses sistemas mais sensíveis ao contexto e à subjetividade humana.

Nesse contexto, a análise da valência emocional em ambientes de desenvolvimento colaborativo pode ser potencializada por modelos de linguagem treinados especificamente para captar nuances afetivas em textos informais, como o XLM-T. Proposto por Barbieri et al. (2022) , o XLM-T é um modelo multilíngue derivado do XLM-R, pré-treinado em milhões de mensagens do Twitter, o que o torna particularmente eficaz na detecção de sentimentos em textos curtos, marcados por gírias, abreviações e linguagem coloquial – características frequentemente presentes em comentários de revisão de código em plataformas como o GitHub. Sua capacidade de identificar emoções positivas, negativas ou neutras permite uma avaliação mais precisa e automatizada das interações entre desenvolvedores, enriquecendo a compreensão dos efeitos emocionais sobre a qualidade do software. Dessa forma, o XLM-T se apresenta como uma ferramenta adequada para operacionalizar a extração da valência emocional neste estudo, alinhando-se diretamente à proposta de investigar a relação entre estados afetivos expressos em linguagem natural e métricas técnicas de qualidade de código.

3. Trabalhos Relacionados

Nesta seção, são apresentados trabalhos que contextualizam este trabalho por fornecerem as bases teóricas, abordagens metodológicas, modelos e técnicas para lidar com o problema das emoções e suas implicações.

Qiu et al. (2022) investigam a detecção automática de conflitos interpessoais em discussões de desenvolvimento de software, focando em dois conceitos principais: toxicidade (linguagem hostil ou desrespeitosa) e *pushback* (conflito desnecessário durante revisões de código). O estudo compara duas abordagens: uma baseada em análise de texto (para detectar toxicidade em *issues* de código aberto) e outra baseada em métricas de *logs* (para identificar *pushback* em revisões de código corporativas). Os autores avaliam a generalização desses métodos em diferentes contextos (código aberto vs. corporativo) e tipos de discussão (*issues* vs. revisões de código), além de propor um classificador combinado. Os resultados mostram que a abordagem baseada em texto tem melhor desempenho para detectar toxicidade, enquanto a combinação de métodos melhora a detecção de *pushback* em ambientes corporativos. Esse trabalho se relaciona com o estudo proposto neste trabalho pela análise das emoções em conflitos interpessoais em *code reviews*. Este trabalho expande o escopo da análise para incluir métricas de qualidade de código.

Sarker et al. (2023) apresentam o *ToxiSpanSE*, uma ferramenta para detecção de toxicidade em comentários de revisão de código no domínio da Engenharia de Software (ES) em plataformas de código aberto. Interações tóxicas podem prejudicar os relaciona-

mentos entre desenvolvedores e afetar a qualidade do software. O problema identificado é que as ferramentas existentes classificam textos inteiros como tóxicos ou não tóxicos, sem explicar quais trechos específicos causam a toxicidade, o que dificulta a moderação. A solução proposta é um modelo de detecção de *spans* tóxicos (trechos específicos) que utiliza transformadores como RoBERTa, ajustados com um conjunto de dados anotados manualmente. Os resultados mostram que o modelo alcançou um F1-score de 0.88 para a classe tóxica, uma métrica utilizada para avaliar o desempenho de um modelo de classificação, destacando-se pela precisão e explicabilidade. Este trabalho se relaciona com o estudo proposto neste artigo, pois explora a detecção de toxicidade. Neste trabalho, adotou-se um processo semelhante para classificar a intensidade das emoções, utilizando, contudo, um modelo distinto, voltado para a extração da valência emocional.

Barbieri et al. (2021) apresentam o XLM-T, um modelo de linguagem multilíngue treinado com dados do Twitter (atualmente *X*) e derivado do XLM-R. O contexto do estudo é a crescente demanda por modelos de Processamento de Linguagem Natural que lidem com a diversidade linguística e as peculiaridades dos textos em redes sociais, que são marcados por uma comunicação informal, uso de abreviações, *emojis* e gírias. O problema abordado é a falta de modelos multilíngues treinados especificamente para esse domínio, capazes de capturar adequadamente essas características. A solução proposta é o modelo XLM-T pré-treinado em 198 milhões de *tweets* em mais de 30 idiomas e ajustado para tarefas como análise de sentimentos. Os resultados mostram que o XLM-T supera modelos gerais como o XLM-R em tarefas específicas do Twitter, especialmente em cenários de *zero-shot* e transferência cruzada entre idiomas. O artigo também introduz o *Benchmark* Unificado para Análise de Sentimentos (*BUAS*), do inglês *Unified Multilingual Sentiment Analysis Benchmark (UMSAB)*. O trabalho de Barbieri destaca-se pela eficácia do modelo em lidar com dados multilíngues e de domínio específico. Esse modelo é utilizado no presente trabalho para extrair a valência emocional dos comentários nas revisões de código.

Jin et al. (2023) propõem uma abordagem estatística para a medição da qualidade de código-fonte, com base na distribuição de métricas em larga escala. O estudo parte do desafio de avaliar de forma consistente a qualidade de software, considerando a natureza diversa das métricas disponíveis. Os autores distinguem entre métricas monotônicas — que apresentam relação direta e consistente com a qualidade, como o número de *code smells* — e métricas não-monotônicas, cuja relação depende do contexto, como a complexidade ciclomática. Para lidar com essas diferenças, os autores utilizam distribuições exponenciais para métricas monotônicas e distribuições gaussianas assimétricas para métricas não-monotônicas, atribuindo pontuações normalizadas de 0 a 100. Em seguida, aplicam um modelo de *boosting* para ponderar a relevância das métricas e gerar uma pontuação global de qualidade. A análise, realizada com base em 36.460 repositórios *open-source*, mostra uma forte correlação entre a pontuação de qualidade e a adoção de software, medida por estrelas no GitHub. Esse trabalho destaca-se por oferecer uma metodologia escalável e interpretável para avaliação automática da qualidade de código. Este trabalho se relaciona ao trabalho proposto no artigo ao fornecer métricas de qualidade de código que servem como base para as análises realizadas.

Miller et al. (2022) exploram como a toxicidade aparece em discussões de projetos de código aberto no GitHub e a diferença entre outras plataformas, como Reddit ou Wiki-

pedia. Os autores focam no ambiente colaborativo de desenvolvimento de software, onde interações tóxicas podem afastar potenciais contribuintes, prejudicar a diversidade e até levar os mantenedores ao esgotamento. Os autores exploram como a dinâmica da toxicidade funciona nesse ecossistema e a eficácia de ferramentas genéricas para detectar e lidar com isso. Para investigar o problema, foram coletadas 100 discussões tóxicas no GitHub usando diferentes estratégias e foi realizada uma análise qualitativa, classificando os tipos de toxicidade, identificando os autores, os gatilhos e as reações das pessoas. Os resultados mostraram que a toxicidade em projetos de código aberto tem características específicas: são comuns exigências exageradas (*"entitlement"*) e insultos técnicos. Destaca-se que, embora muitas vezes sejam iniciadas por usuários externos, essas interações também podem partir de membros mais experientes ou até de mantenedores. A toxicidade pode não apenas afastar novos contribuintes, mas também criar um ambiente hostil que prejudica a colaboração e a inovação. Esse trabalho se relaciona com o estudo proposto neste artigo ao fornecer uma base sólida sobre os efeitos negativos da toxicidade em comunidades de desenvolvimento de software.

Ao reunir esses estudos, obteve-se uma visão ampla sobre o impacto das emoções na qualidade do código durante revisões. Esse panorama não só amplia a compreensão teórica sobre os estados emocionais dos desenvolvedores, mas também serve como base para a investigação específica sobre como as valências emocionais nos comentários de revisão afetam a qualidade do código. Assim, esta pesquisa avança o conhecimento na área, contribuindo com análises de repositórios abertos por meio do Processamento de Linguagem Natural e focando na relação entre a qualidade do código-fonte e as emoções expressas nas revisões.

4. Materiais e Métodos

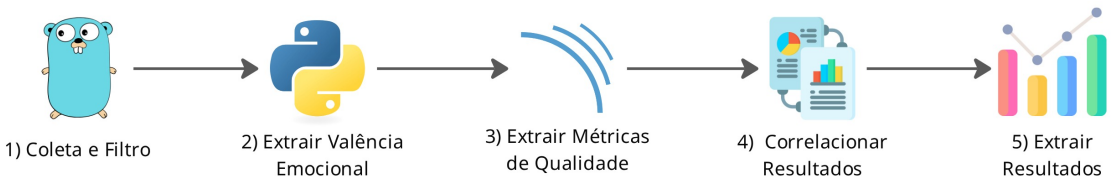

Figura 1. Metodologia de cinco etapas para o desenvolvimento do trabalho proposto.

Nesta seção, são apresentados os materiais e métodos adotados na condução deste estudo. O trabalho caracteriza-se como uma pesquisa quantitativa [Wohlin et al. 2012] ,onde é descrito o processo de obtenção dos dados por meio da mineração dos repositórios, bem como o método empregado para sua análise. Além disso, são apresentados os procedimentos utilizados para avaliar os resultados obtidos. A Figura 1 resume as etapas da metodologia, que são detalhadas a seguir nas seções.

4.1. Coleta e Filtro

Para coletar os dados, foi desenvolvido um *script* na linguagem Go — amplamente utilizada em aplicações que exigem alto desempenho — com o objetivo de extrair informações específicas de repositórios, como *pull requests*, comentários e a URL (*Uniform Resource*

Locator), utilizada para realizar o *clone* do repositório. O *script* realiza consultas à Interface de Programação de Aplicações (API, do inglês *Application Programming Interface*) GraphQL do GitHub. Como critério inicial, foram selecionados os repositórios públicos mais populares da plataforma, cuja principal língua é o inglês. O mesmo *script* também foi utilizado para identificar os repositórios com o número de *pull requests* necessárias para a análise.

A seleção dos repositórios foi realizada por meio de filtros específicos na consulta GraphQL, considerando apenas aqueles com, no mínimo, 500 *pull requests* nos estados *closed* ou *merged*. Para refinar a análise, foram incluídas apenas *pull requests* com pelo menos dez comentários. Esses critérios visam garantir uma base de dados estatisticamente significativa, uma vez que *pull requests* com baixo volume de comentários não oferecem contexto suficiente para análises quantitativas. Além disso, repositórios com maior atividade tendem a representar de forma mais precisa os padrões de colaboração e revisão de código. Os resultados foram armazenados em documentos no formato JSON (*JavaScript Object Notation*), um padrão leve de intercâmbio de dados.

Ao final do processo de coleta e filtragem, a base de dados consolidada foi composta por 94 repositórios de código aberto. A aplicação dos critérios de seleção resultou na extração de dados de 500 *pull requests* de cada um desses repositórios, totalizando um volume de 678.276 comentários.

4.2. Extração da Valência Emocional

Após a filtragem dos repositórios foi feita na extração da valência emocional de cada comentário presente nas *pull requests*, através de um *script* desenvolvido na linguagem Python — amplamente utilizada no campo da Inteligência Artificial — utilizando o modelo XLM-T para análise de sentimentos.

A escolha do XLM-T foi deliberada após a consideração de outros modelos de Processamento de Linguagem Natural disponíveis, como os da família BERT ou abordagens baseadas em léxico. Embora estes sejam eficazes em contextos gerais, a principal vantagem do XLM-T para este estudo reside em seu treinamento específico. Portanto, em comparação a modelos mais genéricos, o XLM-T oferece uma especialização que aumenta a precisão e se mostra mais adequada ao cenário proposto neste trabalho. Vale ressaltar, contudo, que, por ter sido treinado originalmente com dados do Twitter, o modelo não é nativo do domínio de Engenharia de Software (ES). Essa característica implica que, embora ele seja eficaz para captar informalidade, pode haver limitações na interpretação de jargões técnicos específicos ou nuances do contexto de desenvolvimento.

A aplicação do modelo permitiu a quantificação das valências emocionais em toda a base de dados coletada. Ao todo, foram analisados 678.276 comentários, e a análise da emoção dominante em cada um revelou uma predominância do sentimento neutro, presente em 60,54% (410.608) das interações. O sentimento negativo foi o segundo mais frequente, com 27,50% (186.527) dos comentários, seguido pelo Positivo, com 11,96% (81.141). Esse processo gerou um conjunto de dados estruturado para a etapa de correlação.

4.3. Extração das Métricas de Qualidade

Para a extração das métricas de qualidade foi desenvolvido um *script* em Python — linguagem amplamente adotada em tarefas de automação, ciência de dados e aplicações em inteligência artificial — para automatizar o processo de coleta e análise do código-fonte.

Como estratégia, optou-se por considerar apenas os arquivos do projeto a partir da data da *pull request* mais antiga disponível em cada repositório. Essa abordagem visa alinhar temporalmente os dados analisados, garantindo que as métricas extraídas reflitam com maior precisão o contexto do período em que as interações e discussões (avaliadas na etapa de análise de sentimentos) ocorreram. Dessa forma, evitam-se distorções que poderiam surgir caso fossem consideradas versões significativamente mais recentes ou desatualizadas do código, preservando a coerência entre o conteúdo técnico e os aspectos emocionais discutidos nas *pull requests*.

As métricas de qualidade apresentadas na Tabela 1 foram extraídas por meio da ferramenta SonarQube, amplamente adotada no setor de desenvolvimento de software. A escolha do SonarQube justifica-se pela sua capacidade de fornecer indicadores padronizados e comparáveis entre diferentes projetos e linguagens, o que favorece análises consistentes e reprodutíveis. As métricas selecionadas refletem aspectos essenciais da qualidade do código, como manutenibilidade, legibilidade, confiabilidade e esforço de refatoração. Ao considerar essas dimensões, é possível obter uma visão abrangente e objetiva da saúde do código-fonte, contribuindo para decisões mais informadas durante o processo de desenvolvimento e manutenção de software.

Métrica	Definição
Code smells	São sintomas ou indícios de problemas no código-fonte que podem indicar a presença de más práticas de programação ou áreas que precisam de melhoria. Esta métrica retorna a contagem total de *code smells* identificados no código.
Complexidade cognitiva	Refere-se a quão difícil é entender o fluxo de controle do código. Uma complexidade cognitiva de 1 indica que o código é simples e fácil de entender. Valores maiores que 1 indicam (progressivamente) a dificuldade de compreensão do código.
Bugs	Retorna a quantidade N de erros ou defeitos encontrados em um software. Quanto menor o número de *bugs*, melhor a qualidade do software.
Dívida técnica	É uma métrica de esforço que indica o tempo, em minutos, necessário para realizar a refatoração de todos os *code smells* detectados no projeto. Ao converter esse tempo em dias, considera-se que um dia de trabalho possui 8 horas.

Tabela 1. Métricas utilizadas para medir a qualidade dos códigos analisados pelo SonarQube [SonarQube 2025].

O processo de extração foi executado para todos os 94 repositórios selecionados. Em média, cada repositório apresentou 219,10 *bugs*, 3.329,65 *code smells*, e uma dívida técnica de 71,98 dias, refletindo o estado geral da qualidade do código no ecossistema analisado. Esses indicadores, juntamente com a complexidade cognitiva (média de 29.739,27), formam a base de indicadores técnicos para a análise de correlação.

4.4. Correlacionar Resultados

A etapa final do estudo concentra-se na análise das possíveis relações entre a valência emocional presente nos comentários das *pull requests* e as métricas de qualidade do código extraídas via SonarQube. O objetivo é verificar se interações positivas, neutras

ou negativas entre os desenvolvedores estão associadas a indicadores técnicos como manutenibilidade, complexidade, presença de *bugs* ou *code smells*.

Para operacionalizar a análise, primeiramente, foi necessário converter a valência emocional agregada de cada repositório em um valor numérico que pudesse ser utilizado em cálculos estatísticos. Para isso, foi criado um **escore de polaridade**, que quantifica o sentimento geral de um repositório. Este escore foi calculado para cada um dos 94 repositórios, utilizando a fórmula `(comentários positivos - comentários negativos) / (total de comentários)`. O resultado é um valor contínuo que varia de -1 (indicando um sentimento predominantemente negativo) a +1 (predominantemente positivo), que permite que a carga emocional seja tratada como uma variável quantitativa.

Uma vez que a valência emocional foi quantificada, a primeira abordagem adotada foi investigar a existência de uma tendência entre as variáveis. Para essa análise, foi utilizado o **coeficiente de correlação de Spearman (ρ)** [Statistics 2025b]. A escolha deste método se deve ao fato de ele ser não paramétrico, ou seja, apropriado para avaliar associações monotônicas (quando uma variável aumenta, a outra tende a aumentar ou diminuir, não necessariamente de forma linear) sem exigir que os dados sigam uma distribuição normal. Essa flexibilidade é ideal para o contexto deste estudo, pois não se pode presumir uma relação linear entre o sentimento expresso nos comentários e as métricas de qualidade do código.

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n(n^2 - 1)} \tag{1}$$

Na fórmula, ρ representa o coeficiente de correlação de Spearman, que varia de -1 a +1. O termo d_i refere-se à diferença entre os *ranks* de cada par de observações, ou seja, a diferença entre as posições das observações nas duas variáveis. A soma dos quadrados dessas diferenças, $\sum_{i=1}^{n} d_i^2$, é multiplicada por 6 para ajustar a escala da correlação. O denominador da fórmula, $n(n^2 - 1)$, normaliza a soma dos quadrados das diferenças, onde n é o número total de observações. Essa normalização garante que o coeficiente resultante esteja dentro do intervalo de -1 a +1, permitindo a interpretação da força e direção da correlação. Valores próximos a +1 indicam uma forte correlação positiva, valores próximos a -1 indicam uma forte correlação negativa, e valores próximos a 0 indicam pouca ou nenhuma correlação.

Desta forma, os pares de variáveis analisados com o coeficiente de Spearman foram: (i) escore de polaridade e a contagem de *bugs*; (ii) escore de polaridade e a contagem de *code smells*; (iii) escore de polaridade e a complexidade cognitiva total; e (iv) Escore de Polaridade e a dívida técnica em dias.

Para aprofundar e validar os achados da correlação de Spearman, foi realizada uma análise complementar com o objetivo de verificar se as diferenças observadas nas métricas de qualidade eram estatisticamente significativas quando os repositórios eram divididos em grupos com base no sentimento predominante (positivo, neutro e negativo). Para este fim, utilizou-se o **teste de Kruskal-Wallis** [Statistics 2025a], um método não paramétrico que funciona como alternativa à ANOVA de uma via e é apropriado para comparar dois ou mais grupos independentes sem exigir a normalidade dos dados.

O teste de Kruskal-Wallis avalia a hipótese nula de que as distribuições dos grupos apresentam *ranks* médios semelhantes — o que, sob determinadas condições, pode ser interpretado como igualdade de medianas. Caso o teste resulte em um valor de p estatisticamente significativo, isso sugere que pelo menos um dos grupos difere dos demais em relação à métrica analisada. Assim, enquanto a correlação de Spearman evidencia tendências gerais, o Kruskal-Wallis fornece uma avaliação mais robusta sobre diferenças entre categorias, fortalecendo as conclusões do estudo. Esse procedimento foi aplicado individualmente a cada uma das quatro métricas de qualidade (*bugs*, *code smells*, etc.) para verificar se as variações observadas eram sustentadas estatisticamente.

4.5. Extrair Resultados

A análise dos resultados obtidos visa compreender de que maneira a valência emocional expressa nas discussões em ambientes de desenvolvimento colaborativo influencia a qualidade do código produzido. Presume-se que interações com carga emocional predominantemente positiva tendem a estar associadas a melhores métricas de qualidade, refletindo contextos mais colaborativos e produtivos.

Após a execução dos testes estatísticos de correlação de Spearman e Kruskal-Wallis, obteve-se os primeiros resultados que permitem uma análise preliminar da relação entre a valência emocional e a qualidade do código. O coeficiente de correlação de Spearman indicou uma correlação positiva fraca a moderada entre o escore de polaridade e todas as métricas de qualidade analisadas (*bugs*, *code smells*, complexidade cognitiva e dívida técnica). Isso sugere que, à medida que a polaridade dos comentários se torna mais positiva, as métricas de qualidade tendem a aumentar, o que é um resultado inicial contraintuitivo. Contudo, a análise de significância estatística, realizada com o teste de Kruskal-Wallis, revelou que não há evidências suficientes para afirmar que as diferenças nas distribuições das métricas de qualidade entre os grupos de sentimento (positivo, neutro e negativo) sejam estatisticamente significativas, com todos os p-valores superiores a 0,05. Esses achados iniciais indicam que, embora exista uma aparente tendência de correlação, ela não é estatisticamente robusta o suficiente para descartar a hipótese nula. A análise completa dos resultados e suas implicações serão detalhadas na próxima seção.

5. Resultados

A análise foi conduzida sobre 94 repositórios de código aberto, totalizando 678.276 comentários em *pull requests*. A distribuição das valências emocionais revelou predominância do sentimento neutro (60,54%), seguido pelo negativo (27,50%) e positivo (11,96%).

As métricas de qualidade extraídas pelo SonarQube (*bugs*, *code smells*, complexidade cognitiva e dívida técnica) foram analisadas em função do sentimento predominante nos repositórios. Repositórios com sentimento neutro apresentaram maior variabilidade e valores extremos para todas as métricas.

A Figura 2 mostra a distribuição global das métricas de qualidade (*bugs*, *code smells*, complexidade cognitiva e dívida técnica) para todos os repositórios, permitindo observar a amplitude e dispersão dos valores. Com a predominância do sentimento neutro que pode ser observado com os valores convergindo para o centro na imagem.

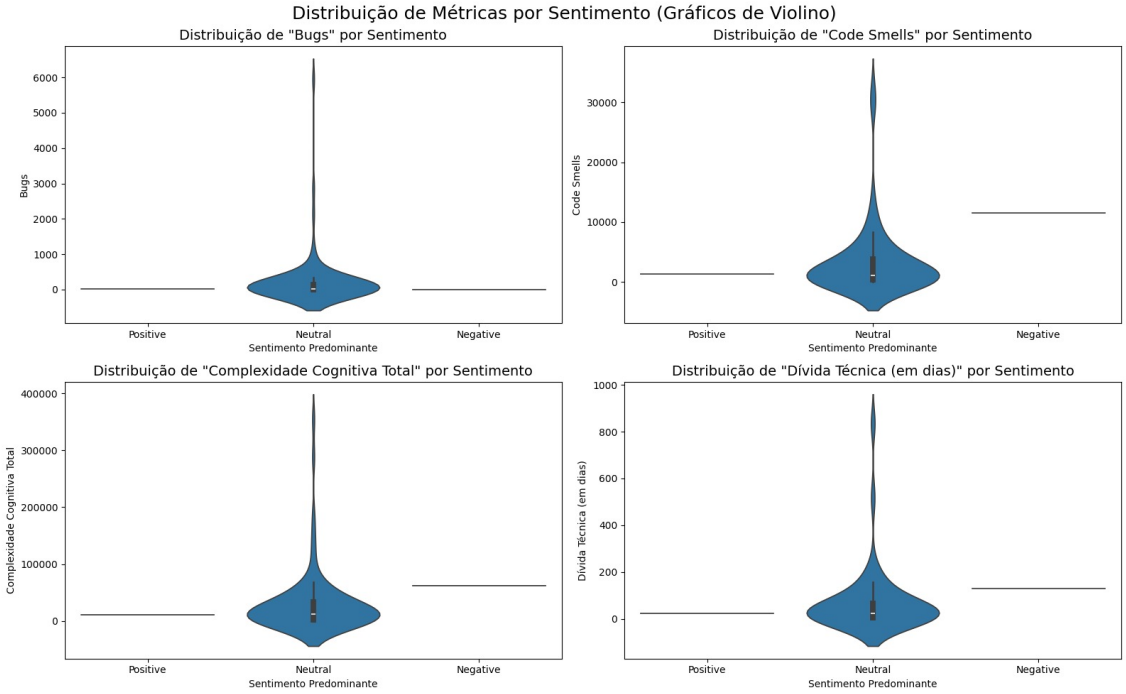

Figura 2. Resumo das métricas globais dos repositórios analisados

5.1. Resultados da correlação de Spearman

O coeficiente de correlação de Spearman foi calculado para avaliar a relação entre o escore de polaridade e as métricas de qualidade. Para verificar a significância estatística dessas associações, também foram calculados os *p-valores* correspondentes. Os resultados indicam associações positivas fracas entre o escore de polaridade e as métricas: *bugs* ($\rho = 0,220$; $p < 0,05$), *code smells* ($\rho = 0,331$; $p < 0,05$), complexidade cognitiva total ($\rho = 0,395$; $p < 0,05$) e dívida técnica em dias ($\rho = 0,315$; $p < 0,05$). Apesar de os *p-valores* indicarem significância estatística (rejeitando a hipótese nula de correlação zero), a magnitude dos coeficientes ($\rho < 0,40$) denota que a força dessa relação é baixa na prática.

A Figura 3 ilustra as correlações de Spearman entre o escore de polaridade e as métricas de qualidade. Todas as métricas apresentam correlação positiva, sendo a maior observada para complexidade cognitiva total.

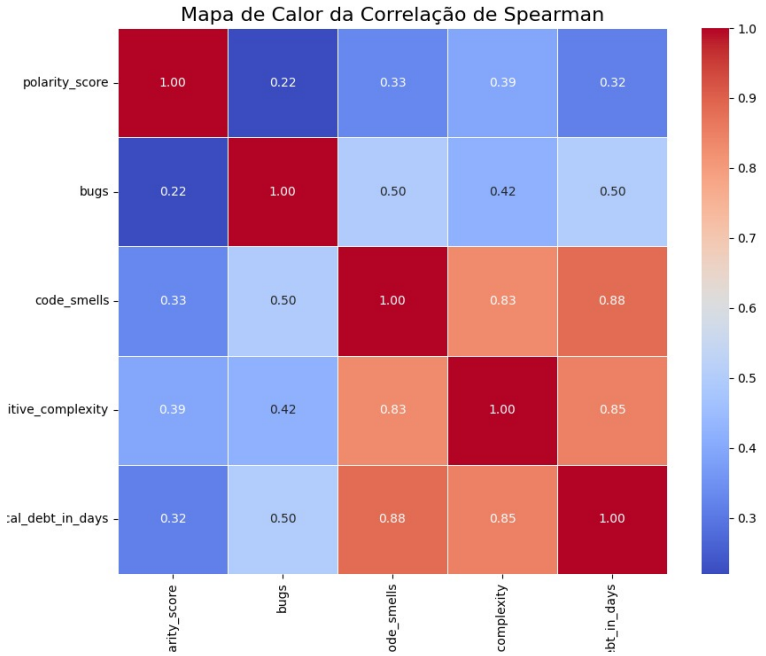

Figura 3. Mapa de calor das correlações de Spearman entre polaridade e métricas de qualidade

A Figura 4 apresenta a distribuição do escore de polaridade nos repositórios analisados. Observa-se concentração em valores próximos a zero, indicando predominância de comentários neutros, com poucos casos extremos positivos ou negativos.

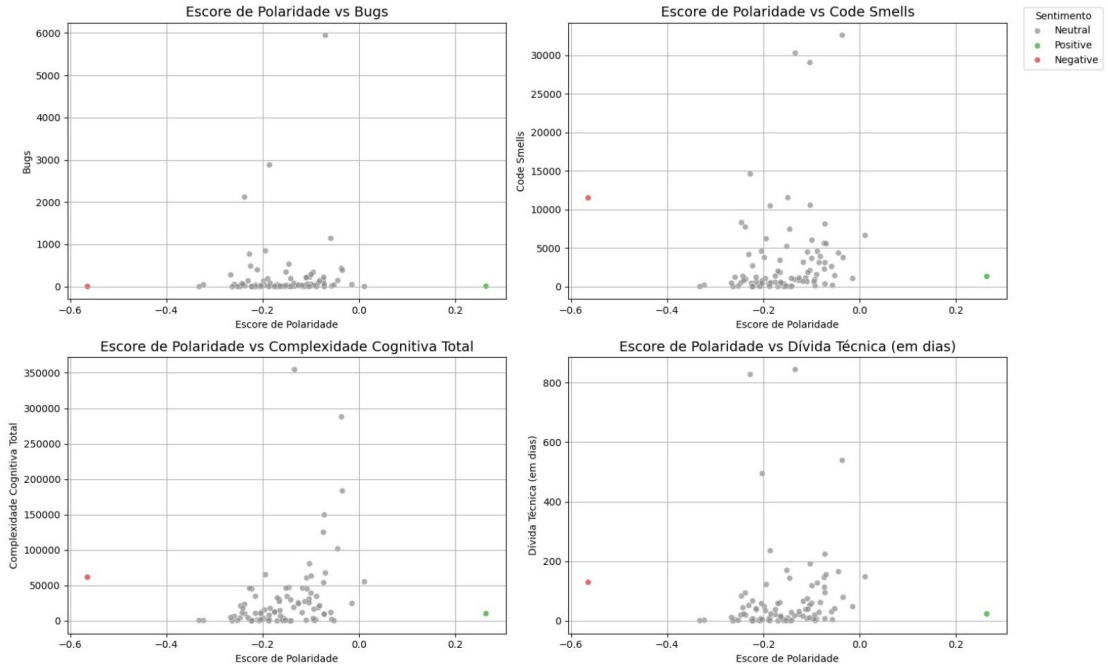

Figura 4. Distribuição do escore de polaridade nos repositórios

5.2. Resultados do teste de significância estatística (Kruskal-Wallis)

O teste de Kruskal-Wallis foi aplicado para verificar diferenças estatísticas entre as distribuições das métricas nos grupos de sentimento positivo, neutro e negativo. Esse teste não-paramétrico avalia se as medianas dos grupos são diferentes, e o *p-valor* indica a probabilidade de observar os dados caso a hipótese nula (de que não há diferença entre os grupos) seja verdadeira. Valores de p superiores a 0,05 indicam ausência de evidência estatística para rejeitar a hipótese nula.

Tabela 2. Resultados do teste de Kruskal-Wallis para cada métrica

Métrica	p-valor	Conclusão
Bugs	0.4911	Não há diferença significativa
Code Smells	0.3054	Não há diferença significativa
Complexidade Cognitiva Total	0.3975	Não há diferença significativa
Dívida Técnica (dias)	0.4666	Não há diferença significativa

Todos os *p-valores* obtidos são maiores que 0,05, indicando que não há diferença estatisticamente significativa entre os grupos de sentimento para nenhuma das métricas analisadas. Em outras palavras, as variações observadas nas distribuições podem ser atribuídas ao acaso, não havendo evidência robusta de que o sentimento predominante nos comentários esteja associado a mudanças nas métricas de qualidade.

5.3. Discussão dos Resultados

A predominância de comentários neutros (60,54%) observada na Figura 2 sugere que o ambiente de revisão de código nos projetos analisados é majoritariamente técnico e objetivo. Esse comportamento alinha-se ao esperado em comunidades de código aberto maduras, onde o foco das interações tende a ser a funcionalidade e a estrutura do código, em detrimento de expressões afetivas pessoais.

A ausência de uma correlação forte ou de diferenças significativas entre os grupos de sentimento (testada via Kruskal-Wallis) pode ser interpretada sob duas óticas. Primeiramente, a **profissionalização dos desenvolvedores**: engenheiros experientes tendem a manter a qualidade técnica do código independentemente de frustrações momentâneas ou elogios recebidos nos comentários. Em segundo lugar, a **limitação das métricas estáticas**: ferramentas como o SonarQube medem a estrutura do código (complexidade, duplicidade), mas não capturam nuances lógicas ou de design que poderiam ser mais sensíveis ao estado emocional do desenvolvedor. Portanto, embora a emoção possa afetar a produtividade ou o bem-estar [Girardi et al. 2020], os dados sugerem que ela não degrada imediatamente a estrutura sintática do código entregue.

A expectativa inicial deste estudo era encontrar uma correlação empírica direta entre a valência emocional nas discussões de *pull requests* e a qualidade técnica do código-fonte. Esperava-se que interações predominantemente negativas se correlacionassem com um aumento em métricas como *bugs* e *code smells*, enquanto um sentimento positivo se associaria a melhores indicadores de qualidade, alinhando-se à literatura que aponta os prejuízos de emoções negativas na produtividade. Os resultados, contudo, não forneceram evidências estatísticas que corroborem essa hipótese. Embora a correlação de

Spearman tenha apontado uma fraca tendência contraintuitiva (onde polaridade positiva se correlacionou levemente com piores métricas), a análise estatística robusta (teste de Kruskal-Wallis) demonstrou que não existe evidência estatística significativa que suporte essa relação. O principal achado do trabalho é, portanto, a ausência de uma ligação estatisticamente comprovada entre o sentimento expresso nos comentários e a qualidade estática do código nos 94 repositórios analisados.

6. Conclusão

O objetivo deste estudo foi investigar se a valência emocional presente em comentários de *pull requests* apresenta alguma relação com a qualidade do código dos repositórios analisados. Partiu-se da hipótese de que valências negativas poderiam estar associadas a piores indicadores de qualidade (maior número de *bugs*, *code smells*, dívida técnica e complexidade), enquanto valências positivas poderiam estar relacionadas a melhores indicadores. Para isso, foram utilizados dados provenientes de 94 repositórios do GitHub, totalizando 678.276 comentários analisados. O processamento envolveu a utilização do modelo XLM-T para identificação da valência emocional dos comentários e do SonarQube para coleta de métricas de qualidade estática do código. As análises estatísticas foram conduzidas com correlação de Spearman e teste de Kruskal-Wallis, visando identificar possíveis associações e diferenças significativas entre os grupos de valência.

Os resultados obtidos indicam correlações fracas e positivas entre a valência média dos comentários e as métricas de qualidade (*bugs* $\rho \approx 0{,}22$; *code smells* $\rho \approx 0{,}33$; complexidade $\rho \approx 0{,}395$; dívida técnica $\rho \approx 0{,}315$). Entretanto, os testes de significância não apontaram diferenças estatisticamente relevantes entre os grupos de sentimento (p-valores superiores a 0,05). Assim, os achados podem sugerir a existência de uma associação leve e contraintuitiva entre emoções positivas e piores métricas de qualidade, mas não fornecem evidências suficientes para afirmar a existência de uma relação direta ou causal.

Esse resultado pode estar relacionado a diversas limitações metodológicas. A predominância de comentários neutros reduziu a sensibilidade para detectar variações emocionais relevantes. Além disso, o modelo XLM-T, embora robusto, foi originalmente treinado em textos do Twitter, o que pode ter comprometido sua precisão ao lidar com linguagem técnica presente em revisões de código. As métricas fornecidas pelo SonarQube, por sua vez, medem aspectos estáticos e agregados da qualidade, não refletindo necessariamente o impacto direto das interações humanas no código em produção. Outro fator é o nível de agregação adotado (valência média por repositório), que pode ter mascarado efeitos locais ou pontuais. Também se reconhece a ausência de controle de variáveis externas, como linguagem de programação, tamanho dos projetos ou cultura de contribuição, que podem ter influenciado os resultados.

Diante dessas limitações, este estudo indica a necessidade de aprofundamentos futuros. Entre as possibilidades de continuidade, destacam-se: (i) realizar análises em nível de *commit* ou arquivo, reduzindo o efeito de agregação; (ii) ajustar o modelo de valência emocional com um *fine-tuning* em *corpus* de *code reviews*; (iii) combinar análises quantitativas com avaliações qualitativas para contextualizar o conteúdo dos comentários; (iv) utilizar modelos estatísticos multivariados para controlar variáveis de confusão; e (v) ampliar a amostra para projetos de diferentes domínios e comunidades de desenvolvimento, de forma a verificar a generalização dos resultados.

Como consideração final, este trabalho indica que, embora seja possível identificar tendências sutis entre emoções expressas em revisões de código e métricas de qualidade estática, não há evidências robustas de que a valência emocional influencie diretamente a qualidade técnica do software. O estudo, portanto, reforça a complexidade das interações humanas no contexto do desenvolvimento colaborativo e contribui metodologicamente ao demonstrar um caminho para integrar análise de sentimentos e métricas de engenharia de software em larga escala. Em síntese, os resultados sugerem que a relação entre emoções e qualidade de código é multifacetada e requer abordagens mais refinadas e contextualizadas para ser compreendida em profundidade.

Pacote de Replicação

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

`https://github.com/ICEI-PUC-Minas-PPLES-TI/plf-es-2025-1-tcci-0393100-pes-diogo-martins-e-renato-paganini`

Referências

[Barbieri et al. 2022] Barbieri, F., Anke, L. E., and Camacho-Collados, J. (2022). XLM-T: Multilingual language models in twitter for sentiment analysis and beyond.

[Girardi et al. 2020] Girardi, D., Novielli, N., Fucci, D., and Lanubile, F. (2020). Recognizing developers' emotions while programming. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 666–677.

[International Organization for Standardization 2001] International Organization for Standardization (2001). ISO/IEC 9126-1:2001 - Software engineering – Product quality – Part 1: Quality model. `https://www.iso.org/standard/22749.html`. Acesso em: 2025-06-11.

[International Organization for Standardization 2011] International Organization for Standardization (2011). ISO/IEC 25010:2011 - Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. `https://www.iso.org/standard/35733.html`. Acesso em: 2025-06-11.

[Jin et al. 2023] Jin, S., Li, Z., Chen, B., Zhu, B., and Xia, Y. (2023). Software code quality measurement: Implications from metric distributions. In *2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS)*, pages 488–496.

[Manning and Schutze 1999] Manning, C. and Schutze, H. (1999). *Foundations of statistical natural language processing*. MIT press. Acesso em: 2025-06-11.

[Michels et al. 2024] Michels, L.-M., Petkova, A., Richter, M., Farley, A., Graziotin, D., and Wagner, S. (2024). Overwhelmed software developers. *IEEE Software*, 41(4):51–59.

[Miller et al. 2022] Miller, C., Cohen, S., Klug, D., Vasilescu, B., and Kästner, C. (2022). "did you miss my comment or what?" understanding toxicity in open source discussions. In *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, pages 710–722.

[Qiu et al. 2022] Qiu, H. S., Vasilescu, B., Kästner, C., Egelman, C., Jaspan, C., and Murphy-Hill, E. (2022). Detecting interpersonal conflict in issues and code review: Cross pollinating open- and closed-source approaches. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, pages 41–55.

[Recchia and Louwerse 2015] Recchia, G. and Louwerse, M. M. (2015). Reproducing affective norms with lexical co-occurrence statistics: Predicting valence, arousal, and dominance. *Quarterly journal of experimental psychology*, 68(8):1584–1598.

[Sarker et al. 2023] Sarker, J., Sultana, S., Wilson, S. R., and Bosu, A. (2023). Toxispanse: An explainable toxicity detection in code review comments. In *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–12.

[Sommerville 2019] Sommerville, I. (2019). *Engenharia de software*. Pearson Education do Brasil, São Paulo, 10 edition. Tradução de Luiz Claudio Queiroz. Revisão técnica de Fávio Levy Siqueira.

[SonarQube 2025] SonarQube (2025). Sonarqube - metric definitions. `https://docs.sonarsource.com/sonarqube-server/latest/user-guide/code-metrics/metrics-definition/`. Acesso em: 2025-05-12.

[Statistics 2025a] Statistics, L. (2025a). Kruskal-Wallis H Test using SPSS Statistics. `https://statistics.laerd.com/spss-tutorials/kruskal-wallis-h-test-using-spss-statistics.php`. Acesso em: 2025-09-26.

[Statistics 2025b] Statistics, L. (2025b). Spearman's Correlation in Stata - Procedure, output and interpretation of the output using a relevant example. — statistics.laerd.com. `https://statistics.laerd.com/stata-tutorials/spearmans-correlation-using-stata.php`. Acesso em: 2025-05-18.

[Wohlin et al. 2012] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., and Wesslén, A. (2012). *Experimentation in Software Engineering*. Computer Science. Springer Berlin Heidelberg.