Code Smells e Qualidade de Código: Uma Abordagem Experimental com Desenvolvedores Humanos e Inteligências Artificiais

Keren Apuque Cardoso de Morais
Instituto Federal do Paraná
Paranavaí, Paraná
cardosokerenapuque@gmail.com

Eduardo Albuquerque Ribeiro
Instituto Federal do Paraná
Paranavaí, Paraná
eduardoribeiro.advg@gmail.com

Evanise Araujo Caldas Ruiz
Instituto Federal do Paraná
Paranavaí, Paraná
evanise.ruiz@ifpr.edu.br

Frank Willian Cardoso de Oliveira
Instituto Federal do Paraná
Paranavaí, Paraná
frank.willian@ifpr.edu.br

Marcelo Figueiredo Terenciani
Instituto Federal do Paraná
Paranavaí, Paraná
marcelo.terenciani@ifpr.edu.br

***Abstract*—This study investigates the influence of knowledge about code smells on code quality in object-oriented projects, comparing human developers and artificial intelligence language models. The research includes a literature review to consolidate key concepts and an empirical experiment with four groups (humans and AIs, with and without instructions about code smells). The experimental phase involving human developer groups is currently underway, with no consolidated results so far. Code quality will be assessed based on the occurrence of code smells, enabling a comparative analysis among the profiles. Preliminary findings highlight the impact of knowledge or guidelines regarding code smells on software development and bring implications for the use of AI and the adoption of quality-oriented coding practices.**

***Keywords*—Code smells; Artificial Intelligence Language Models; Object-Oriented.**

***Resumo*—Este trabalho investiga a influência do conhecimento sobre *code smells* na qualidade do código em projetos orientados a objetos, comparando desenvolvedores humanos e modelos de linguagem de inteligência artificial. A pesquisa envolve uma revisão de literatura, para consolidar conceitos e um experimento empírico com quatro grupos (humanos e IAs, com e sem instruções sobre *code smells*). A etapa de execução do experimento com grupos de desenvolvedores humanos está em andamento, sem resultados consolidados até o momento. A qualidade do código será avaliada pela ocorrência de *code smells*, permitindo uma análise comparativa entre os perfis. Os resultados parciais destacam o impacto do conhecimento ou de diretrizes sobre *code smells* na produção de software e trazem implicações para o uso de IA e a adoção de práticas de codificação de qualidade.**

***Palavras-chave*—Code Smells; Modelos de Linguagem de Inteligência Artificial; Programação Orientada a Objetos**

I. Introdução

O desenvolvimento de *software* orientado a objetos (POO) é amplamente adotado na indústria devido à sua capacidade de promover modularidade, reutilização de código e manutenibilidade [1]. No entanto, mesmo em projetos bem estruturados, é comum a introdução de más práticas de programação, como *code smells*, que, embora não sejam erros de compilação, indicam possíveis problemas de qualidade, dificultando a legibilidade, evolução e a manutenção do *software* [2].

Code smells são sintomas de código de baixa qualidade. O conceito foi amplamente difundido por Fowler [3], que catalogou 22 *code smells*. Posteriormente, outros autores, como Mäntylä [4], Wake [5] e Lanza e Marinescu [6], expandiram essa classificação, originando taxonomias que fundamentam as atuais ferramentas de detecção de *code smells* [7]. Embora não afetem diretamente o funcionamento do sistema, os *code smells* dificultam futuras manutenções.

Com o avanço dos modelos de linguagem de inteligência artificial (IA) para geração automatizada de código [8], surge uma nova questão: como esses modelos se comparam a desenvolvedores humanos em relação à produção de *code smells*? Enquanto programadores humanos podem aplicar conhecimentos adquiridos em treinamentos, as IAs dependem de instruções explícitas para evitar más práticas. Essa diferença levanta discussões sobre a eficácia do ensino de boas práticas de programação e o papel da IA no desenvolvimento de *software*.

A. Justificativas

Diante disso, este trabalho propõe investigar de forma comparativa a influência do conhecimento sobre *code smells* na produção de código orientado a objetos, considerando dois perfis de desenvolvedores: humanos e modelos de linguagem de inteligência artificial. A pesquisa busca responder à seguinte questão: de que forma o conhecimento prévio em humanos e o fornecimento de diretrizes em modelos de linguagem de inteligência artificial afetam a ocorrência de *code smells* no código gerado durante o desenvolvimento de software orientado a objetos?

B. Metodologia

Para alcançar esse objetivo, esta pesquisa está sendo conduzida em duas etapas principais. Primeiramente, foi realizada uma revisão da literatura, conforme as diretrizes propostas por Kitchenham *et al.* [9], com o intuito de identificar, classificar e consolidar os principais *code smells* documentados na programação orientada a objetos.

Essa etapa consolidou o referencial teórico para embasar a segunda fase do estudo, que consiste em um experimento empírico controlado, conforme orientações metodológicas de Wohlin *et al.* [10]. O experimento envolverá quatro grupos: (i) desenvolvedores humanos com conhecimento prévio sobre *code smells*; (ii) desenvolvedores humanos sem conhecimento prévio sobre *code smells*; (iii) modelos de linguagem de IA com diretrizes explícitas sobre *code smells*; e (iv) modelos de linguagem de IA sem tais diretrizes.

O experimento com os grupos (iii) e (iv) já possuem resultados parciais, enquanto que a realização das atividades previstas para os grupos (i) e (ii) ainda estão em andamento devido aos trâmites necessários no Comitê de Ética em Pesquisa Envolvendo Seres Humanos (CEP).

Cada grupo será responsável por resolver a mesma tarefa de desenvolvimento utilizando a linguagem Java ou Python. A qualidade do código produzido será avaliada com base na frequência, variedade e criticidade dos *code smells* identificados, permitindo uma análise comparativa entre os diferentes perfis. Os dados coletados também incluirão métricas auxiliares, como número de linhas de código e complexidade do código, com o objetivo de oferecer uma visão mais abrangente da qualidade das soluções geradas.

Os resultados obtidos serão organizados e comparados entre os grupos, sendo apresentados por meio de gráficos e tabelas para facilitar a visualização. A interpretação desses dados será realizada com base nos objetivos e hipóteses da pesquisa, fornecendo subsídios para compreender o impacto do conhecimento prévio ou do fornecimento de instruções sobre a ocorrência de *code smells* em ambientes de desenvolvimento.

II. Trabalhos relacionados

A fim de buscar os trabalhos relacionados, foi realizado um Mapeamento Sistemático da Literatura, seguindo as propostas de Kitchenham, Chartes *et al.* [9]. A questão de pesquisa principal que norteia este trabalho é: *Agentes de Inteligência Artificial produzem código com maior ou menor qualidade em comparação aos desenvolvedores humanos, especialmente no que se refere à presença de code smells?*

Para a *string* de busca utilizou-se a seguinte: *("code smells" OR "bad smells") AND ("developers" OR "artificial intelligence") AND ("empirical study" OR "experiment")*. Os critérios de inclusão levaram em consideração publicações entre 2020 e 2025, disponíveis integralmente para leitura e cujo resumo tivesse relação com o tema.

Após esse refinamento, 5 artigos foram incluídos nesta pesquisa, sendo eles Albuquerque *et al.* [11]; Bezerra, Damasceno e Teixeira [12]; Castellano *et al.* [13]; Cordeiro, Noei e Zou [14]; Martinović e Rozić [15]. Quatro autores abordaram a análise de *code smells* em códigos orientados a objetos, mas focando apenas em um dos públicos (desenvolvedores humanos ou inteligência artificial), com exceção de [14]. De forma geral, os resultados indicam que a combinação entre ferramentas baseadas em IA e o conhecimento humano tende a gerar melhores resultados na prática da engenharia de software.

III. Desenvolvimento

O planejamento deste estudo está seguindo as etapas propostas por Wohlin *et al.* [10] na definição, planejamento e aplicação do experimento. Para a etapa de definição, foi elaborado um desafio inserido no contexto do desenvolvimento de uma funcionalidade de software para uma empresa fictícia do setor varejista. O sistema deve operar exclusivamente em terminal, sem interface gráfica e sem integração com banco de dados. A implementação deve funcionar localmente, utilizando apenas estruturas de dados em memória, e realizar o cadastro e a listagem de clientes (pessoas físicas e jurídicas).

Durante a aplicação do experimento com o uso de inteligências artificiais, uma ameaça à validade identificada foi o histórico salvo em cada conta de usuário, o que poderia influenciar os resultados. Para mitigar esse risco, foi criada uma conta nova especificamente para o experimento.

Inicialmente, o experimento foi conduzido com o grupo de IAs (*ChatGPT*, *Github Copilot* e *DeepSeek*) sem o uso do *prompt* orientador, de forma a observar os *code smells* mais recorrentes. A partir dessa análise, elaborou-se um *prompt* orientador com o objetivo de mitigar a ocorrência desses problemas de código. A etapa de execução do experimento com grupos de desenvolvedores humanos está em andamento, sem resultados consolidados até o momento.

IV. Resultados parciais e discussões

Os resultados parciais obtidos até o momento referem-se à análise manual dos *code smells*, bem como à comparação entre os *code smells* gerados por IAs a partir de *prompts* sem orientação e aqueles produzidos por IAs a partir de *prompts* com orientação.

A. Inteligências Artificiais sem o prompt orientador

As IAs que receberam apenas o enunciado do desafio produziram 45 ocorrências de *code smells*, distribuídas nos seguintes tipos: *Magic Numbers*, *Long Parameter List*, *Switch Statements*, *Duplicate Code*, *Long Method*, *Data Class*, *Comments*, *Global Data*, *Refused Bequest*.

Com base na Figura 1, observa-se que o *smell Magic Numbers/Strings* é o mais predominante (35,6%), seguido por *Long Parameter List* e *Switch Statements*, ambos com 13,3%. Os principais trechos que introduziram *code smells* foram relacionados à lógica do *loop* principal e aos parâmetros do construtor padrão das classes.

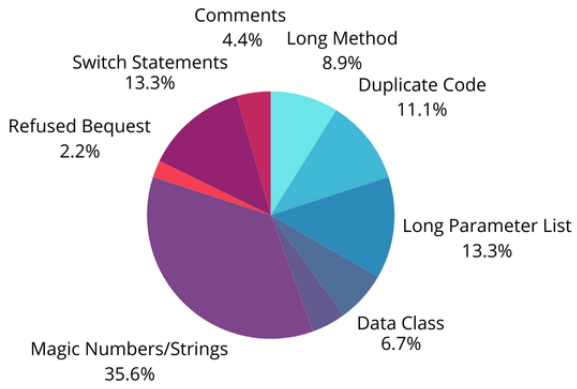

Fig. 1. Porcentagem total de *code smells* produzidos por IAs sem *prompt* orientador. **Fonte**: Autoria própria (2025)

A Figura 2 apresenta a ocorrência de *code smells* identificados nos modelos de linguagem de inteligência artificial sem *prompt* orientador. O *ChatGPT* gerou sete tipos, totalizando 13 ocorrências. Embora o código utilize orientação a objetos na hierarquia de classes, a classe pai deveria ser abstrata, já que serve apenas como uma estrutura compartilhada. Sua ausência caracteriza o *code smell Data Class*.

Além disso, o uso de *strings* nas comparações realizadas no *loop* do menu principal, juntamente com as condicionais aninhadas, ocasionou diversas ocorrências dos *Magic Numbers/Strings* e *Switch Statements*. Outro aspecto relevante refere-se à regra de negócio: o e-mail era opcional para pessoa física e o telefone obrigatório, entretanto o modelo de linguagem de IA considerou ambos como opcionais, gerando inconsistência em relação aos requisitos.

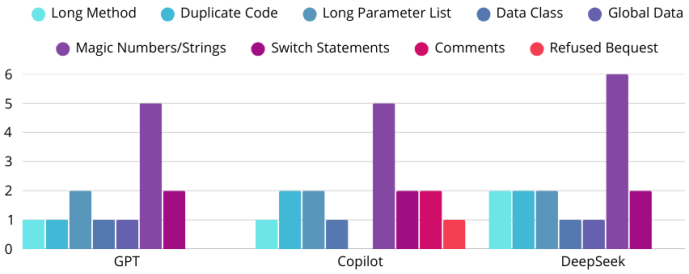

Fig. 2. Ocorrência de *code smells* por modelos de linguagem de inteligência artificial sem *prompt* orientador. **Fonte**: Autoria própria (2025).

O *GitHub Copilot*, por sua vez, produziu oito tipos de *code smells*, totalizando 16 ocorrências. Diferentemente do *ChatGPT*, não implementou a herança de forma adequada, resultando na repetição de dados nas classes derivadas e ocasionando *Refused Bequest*.

Além disso, como esperado, inseriu comentários no código, diferentemente das demais IAs. Em contrapartida, ele não produziu o *code smell Global Data*, observado nas outras inteligências artificiais que declararam variáveis globais.

Por fim, o *DeepSeek* produziu sete tipos de *code smells*, distribuídos em 16 ocorrências. Semelhante ao *ChatGPT*, os sete tipos de *smells* foram os mesmos; no entanto, em maior quantidade, aumentando, portanto, a complexidade do código em comparação ao *ChatGPT*.

B. Inteligências Artificiais que receberam o prompt orientador

Com relação às inteligências artificiais que receberam o enunciado do desafio juntamente com o *prompt* orientador, foram produzidas 34 ocorrências de *code smells*, distribuídas nos seguintes tipos: *Magic Numbers*, *Long Parameter List*, *Switch Statements*, *Duplicate Code*, *Long Method*, *Data Class*, *Comments* e *God Class*. A Figura 3 apresenta a porcentagem total desses *code smells*.

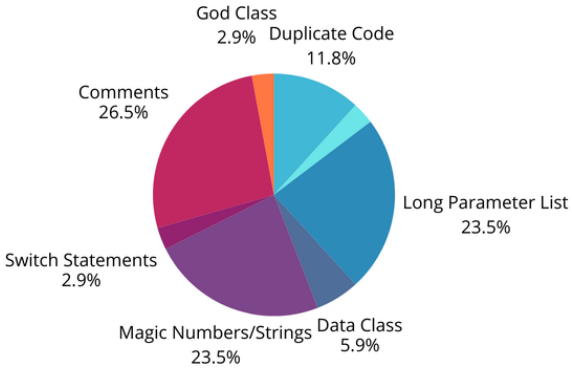

Fig. 3. Porcentagem total de *code smells* produzidos por IAs com *prompt* orientador. **Fonte**: Autoria própria (2025).

A análise da Figura 3 evidencia que o *smell Comments* é o mais recorrente (26,5%), seguido por *Magic Numbers/String* e *Long Parameter List*. Observa-se que os principais trechos responsáveis pela introdução de *code smells* estão relacionados aos métodos abstratos e ao *loop* principal de execução.

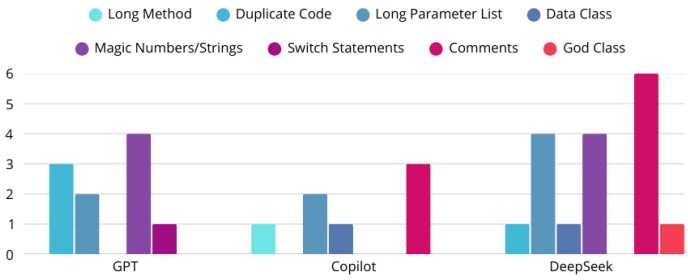

Fig. 4. Ocorrência de *code smells* por modelos de linguagem de inteligência artificial sem *prompt* orientador. **Fonte**: Autoria própria (2025).

No caso do *ChatGPT*, foram identificados quatro tipos de *code smells*, totalizando dez ocorrências. De maneira geral, a implementação seguiu corretamente o paradigma de orientação a objetos, utilizando herança de forma adequada e empregando uma superclasse abstrata de maneira otimizada.

O *Github Copilot* produziu igualmente quatro tipos, distribuídos em sete ocorrências. Embora a implementação também tenha seguido a orientação a objetos, não atendeu ao requisito de execução local via terminal, criando em seu lugar um método de entrada e saída diferente do especificado.

Por fim, o *DeepSeek* gerou seis tipos de *code smells*, totalizando 17 ocorrências. Diferentemente do esperado, a IA não implementou corretamente a orientação a objetos, introduzindo *God Class*, *Long Method* e *Long Parameter List*, principalmente em função da criação de uma única classe que concentrou toda a lógica do sistema.

V. Conclusão

A análise comparativa mostra que as IAs apresentaram comportamentos distintos na geração de *code smells*. O *ChatGPT* manteve-se equilibrado ao longo das implementações, apresentando inicialmente o número mínimo de ocorrências e mantendo a consistência nos resultados subsequentes.

O *Github Copilot* demonstrou uma curva de aprendizagem positiva, com redução significativa tanto na quantidade, quanto na diversidade de *smells*, o que indica uma evolução progressiva em sua capacidade de gerar código mais aderente às boas práticas.

Em contrapartida, o *DeepSeek* apresentou um desempenho menos favorável, caracterizado por confusão e retrocesso no processo de implementação. Essa ferramenta gerou *code smells* mais críticos, como *God Class* e *Long Method*, comprometendo a qualidade estrutural do código.

De modo geral, os resultados parciais apontam para o potencial de utilização de IAs no apoio ao desenvolvimento de software, mas indicam também a necessidade de mecanismos de orientação, como o uso de *prompts* direcionados, para mitigar riscos de más práticas e garantir maior consistência na qualidade do código.

Referências

[1] R. Troquete, "Programação orientada a objetos: uma visão conceitual dos elementos de modelagem," Trabalho de Conclusão de Curso, Pontifícia Universidade Católica de São Paulo, São Paulo, 2019.
[2] T. L. d. Silva, K. N. S. Vidotto, L. M. R. Tarouco, and P. F. d. Silva, "Inteligência artificial generativa no ensino de programação: um mapeamento sistemático da literatura," *RENOTE: Novas Tecnologias na Educação. Porto Alegre, RS. Vol. 22, n. 1 (jul. 2024), p. 262-272*, 2024.
[3] M. Fowler, *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 2018.
[4] M. Mantyla, "Bad smells in software-a taxonomy and an empirical study," Ph.D. dissertation, PhD thesis, Helsinki University of Technology, 2003.
[5] W. C. Wake, *Refactoring workbook*. Addison-Wesley Professional, 2004.
[6] M. Lanza and R. Marinescu, *Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems*. Springer Science & Business Media, 2007.
[7] A. Kaur, "A systematic literature review on empirical analysis of the relationship between code smells and software quality attributes: A. kaur," *Archives of Computational Methods in Engineering*, 2020.
[8] J. P. de Souza Rodrigues, L. A. Santos, and M. R. Almeida, "Uma análise sobre o uso de inteligência artificial no desenvolvimento de software," São Paulo, Brasil, pp. 120–131, 2024.
[9] S. Keele *et al.*, "Guidelines for performing systematic literature reviews in software engineering," Technical report, ver. 2.3 ebse technical report. ebse, Tech. Rep., 2007.
[10] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén *et al.*, *Experimentation in software engineering*. Springer, 2012, vol. 236.
[11] D. Albuquerque, E. Guimarães, M. Perkusich, T. Rique, F. Cunha, H. Almeida, and A. Perkusich, "On the assessment of interactive detection of code smells in practice: A controlled experiment," *IEEE Access*, vol. 11, pp. 84 589–84 606, 2023.
[12] C. Bezerra, H. Damasceno, and J. Teixeira, "Perceptions and difficulties of software engineering students in code smells refactoring," in *Anais do X Workshop de Visualização, Evolução e Manutenção de Software*. Porto Alegre, RS, Brasil: SBC, 2022, pp. 41–45. [Online]. Available: https://sol.sbc.org.br/index.php/vem/article/view/22328
[13] I. L. Castellano, G. F. C. Aguilar, N. Silega, T. Kamal, M. S. Al-Gaashani, N. A. Samee, and M. Alabdulhafith, "An ontology-based approach to reduce the negative impact of code smells in software development projects," *IEEE Access*, vol. 11, pp. 100 146–100 153, 2023.
[14] J. Cordeiro, S. Noei, and Y. Zou, "An empirical study on the code refactoring capability of large language models," *arXiv preprint arXiv:2411.02320*, 2024.
[15] B. Martinović and R. Rozić, "Perceived impact of ai-based tooling on software development code quality," *SN Computer Science*, vol. 6, no. 1, p. 63, 2025.