Do *Prompt* ao Código de Qualidade: Democratizando a Construção de Aplicações Manuteníveis e Seguras (Baldoino Fonseca dos Santos Neto)

Os grandes modelos de linguagem (LLMs, do inglês Large Language Models) são modelos de aprendizado profundo treinado em grandes quantidades de dados para compreender e gerar linguagem natural natural de forma coerente e relevante [1]. Os LLMs, como o GPT (OpenAI) [2], Gemini (Google) [3], Llama (Meta) [4], Claude (Anthropic) [5] e Grok (xAI) [6], têm demonstrado impacto substancial em diversos domínios, com destaque particular para a engenharia de software [7, 8, 9].

Neste campo específico, os LLMs têm revolucionado processos fundamentais, incluindo a geração automatizada de código [10, 11, 12], identificação e correção de bugs ou vulnerabilidades [13, 14, 15], ou mesmo a criação de aplicações completas a partir de especificações em linguagem natural (também conhecido como prompts). Associado ao surgimento de LLMs, temos visto uma proliferação de ferramentas que facilitam a interação com estes modelos através de prompts. Em particular, estas ferramentas são capazes de processar prompts providos por indivíduos com pouca ou nenhuma experiência em programação e prover diversas funcionalidades, como a recomendação de código e implementação de algoritmos (através de plataformas como Cursor [16], ReplitAI [17], Windsurf [18], Bolt.new [19], v0 [20], ZedAI [21], Lovable [22], e GitHub Copilot [23]) até a criação integral de aplicações complexas (através de plataformas como Zoho [24], Bublle [25], and Marblism [26]). Como exemplo, utilizando a plataforma Marblism, indivíduos sem conhecimento técnico algum em programação podem construir aplicações funcionais completas a partir de um prompt como:

”TweetX é uma aplicação social onde os usuários podem:

- Postar tweets;
- Curtir tweets;
- Comentar tweets;
- Retransmitir tweets;
- Seguir outros usuários.

O uso de prompts nos processos de desenvolvimento de software têm apresentado potencial significativo para democratização da construção de aplicações uma vez que possibilita a contribuição de indivíduos com diferentes níveis de conhecimento técnico, ou mesmo sem conhecimento técnico algum em programação [27]. Além disso, o uso de prompts pode levar também a construção de aplicações que reflitam melhor as intenções dos indivíduos em relação

ao comportamento esperado das aplicações uma vez que estes podem expressar suas intenções e requisitos funcionais das aplicações através de linguagem natural. Entretanto, a formulação inadequada de prompts pode resultar na proliferação de aplicações com qualidade comprometida, caracterizadas por diversos problemas de manutenibilidade e segurança como a presença de code smells [28, 29, 30, 31, 32], bugs, ou vulnerabilidades [33, 34, 35, 36, 37, 38, 39, 40, 41]. Neste contexto, a engenharia de prompts, caracterizada pela formulação sistemática de instruções precisas para LLMs, exerce função fundamental na construção de aplicações com qualidade [42, 43, 44]. Embora alguns trabalhos [45, 46, 47, 48] tenham proposto técnicas para aprimoramento do prompt visando melhorar a qualidade do código gerado, ainda há uma lacuna significativa na literatura sobre a relação entre a engenharia de prompts e a qualidade do código gerado para construção de aplicações, especialmente quando se trata de prompts providos por indivíduos com pouca ou nenhuma experiência em programação [27].

[1] S. Raschka, Build a Large Language Model (From Scratch). Shelter Island, NY: Manning Publications, 2024. Includes free eBook in PDF and ePub formats with print book purchase.

[2] OpenAI, “Chatgpt (gpt-4),” 2023. Large language model.

[3] Google, “Gemini 1.5,” 2024. Multimodal generative AI model.

[4] M. AI, “Llama 3.1: Open and efficient foundation language models,” 2024. Large language model with multilingual support, extended context length, and advanced reasoning capabilities.

[5] Anthropic, “Claude 3.5: Advanced multimodal ai assistant,” 2024. Large language model with text and vision capabilities, optimized for reasoning, coding, and multilingual tasks.

[6] xAI, “Grok: Large language model by xai,” 2025. A general-purpose AI model with real-time access to X (formerly Twitter) data, designed for text generation, coding, and conversational tasks.

[7] I. Ozkaya, “Application of large language models to software engineering tasks: Opportunities, risks, and implications,” IEEE Software, vol. 40, no. 3, pp. 4–8, 2023.

[8] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. M. Zhang, “Large language models for software engineering: Survey and open problems,” in 2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE), pp. 31–53, 2023.

[9] e. a. Zhao, “Towards an understanding of large language models in software engineering tasks,” Empirical Software Engineering, vol. 30, no. 2, p. 50, 2024.

[10] X. Du, M. Liu, K. Wang, H. Wang, J. Liu, Y. Chen, J. Feng, C. Sha, X. Peng, and Y. Lou, “Evaluating large language models in class-level code generation,” in Proceedings of the IEEE/ACM 46th International

Conference on Software Engineering, ICSE '24, (New York, NY, USA), Association for Computing Machinery, 2024.

[11] X. Jiang, Y. Dong, L. Wang, Z. Fang, Q. Shang, G. Li, Z. Jin, and W. Jiao, "Self-planning code generation with large language models," ACM Trans. Softw. Eng. Methodol., vol. 33, Sept. 2024.

[12] J. Jiang, F. Wang, J. Shen, S. Kim, and S. Kim, "A survey on large language models for code generation," ArXiv, vol. abs/2406.00515, 2024.

[13] X. Yin, C. Ni, S. Wang, Z. Li, L. Zeng, and X. Yang, "Thinkrepair: Self-directed automated program repair," in Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2024, (New York, NY, USA), p. 1274–1286, Association for Computing Machinery, 2024.

[14] H. Li, Y. Hao, Y. Zhai, and Z. Qian, "Enhancing static analysis for practical bug detection: An llm-integrated approach," Proc. ACM Program. Lang., vol. 8, Apr. 2024.

[15] M. Jin, S. Shahriar, M. Tufano, X. Shi, S. Lu, N. Sundaresan, and A. Svyatkovskiy, "Inferfix: End-to-end program repair with llms," in Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, (New York, NY, USA), p. 1646–1656, Association for Computing Machinery, 2023.

[16] C. Team, "Cursor: Ai-powered code editor." https://www.cursor.so, 2025. Accessed: 2025-01-06.

[17] R. Team, "Replit ai: Ai-powered development platform." https://replit.com/ai, 2025. Accessed: 2025-01-06.

[18] C. Team, "Windsurf: Ai-powered code editor." https://codeium.com/windsurf, 2024. Accessed: 2025-01-06.

[19] S. Team, "Bolt.new: Ai-powered development tool." https://bolt.new, 2024. Accessed: 2025-01-06.

[20] V. Team, "v0: Generative ui by vercel." https://v0.dev, 2024. Accessed: 2025-01-06.

[21] Z. Industries, "Zed ai: Ai-powered code editor." https://zed.dev/ai, 2024. Accessed: 2025-01-06.

[22] L. Team, "Lovable: Ai-powered full-stack application builder." https://lovable.dev, 2024. Accessed: 2025-01-06.

[23] GitHub, “Github copilot: Ai-powered code completion tool.” https://github.com/features/copilot, 2024. Accessed: 2025-01-06.

[24] Z. Corporation, “Zoho creator: Low-code application development platform.” https://www.zoho.com/creator/, 2025. Accessed: 2025-01-06.

[25] B. Team, “Bubble: No-code development platform.” https://bubble.io, 2025. Accessed: 2025-01-06.

[26] M. Team, “Marblism: Ai-powered full-stack application builder.”https://www.marblism.com, 2024. Accessed: 2025-01-06.

[27] B. Binzer and T. Winkler, “Democratizing software development: A systematic multivocal literature review and research agenda on citizen development,” in Software Business: 13th International Conference, ICSOB 2022, Bolzano, Italy, November 8–11, 2022, Proceedings (N. Carroll, A. Nguyen-Duc, X. Wang, and V. Stray, eds.), Lecture Notes in Business Information Processing, (Germany), p. 244–259, Springer, Oct. 2022. 13th International Conference on Software Business.ICSOB 2022 : Software for Digital Transformation, ICSOB2022 ; Conference date: 08-11-2022 Through 11-11-2022.

[28] A. Aljohani and H. Do, “From fine-tuning to output: An empirical investigation of test smells in transformer-based test code generation,” in Proceedings of the 39th ACM/SIGAPP Symposium on Applied
Computing, SAC ’24, (New York, NY, USA), p. 1282–1291, Association for Computing Machinery, 2024.

[29] M. L. Siddiq, S. H. Majumder, M. R. Mim, S. Jajodia, and J. C. S. Santos, “An empirical study of code smells in transformer-based code generation techniques,” in 2022 IEEE 22nd International Working Conference on Source Code Analysis and Manipulation (SCAM), pp. 71–82, 2022.

[30] B. Zhang, P. Liang, Q. Feng, Y. Fu, and Z. Li, “Copilot-in-the-loop: Fixing code smells in copilot-generated python code using copilot,” in Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, ASE ’24, (New York, NY, USA), p. 2230–2234, Association for Computing Machinery, 2024.

[31] W. C. Ouédraogo, Y. Li, K. Kaboré, X. Tang, A. Koyuncu, J. Klein, D. Lo, and T. F. Bissyandé, “Test smells in llm-generated unit tests,” 2024.

[32] A. Velasco, D. Rodriguez-Cardenas, D. N. Palacio, L. R. Alif, and D. Poshyvanyk, “How propense are large language models at producing code smells? a benchmarking study,” 2024.

[33] J. Chen, Q. Zhong, Y. Wang, K. Ning, Y. Liu, Z. Xu, Z. Zhao, T. Chen, and Z. Zheng, “Rmcbench: Benchmarking large language models’ resistance to malicious code,” in Proceedings

of the 39th IEEE/ACM International Conference on Automated Software Engineering, ASE '24, (New York, NY, USA), p. 995–1006, Association for Computing Machinery, 2024.

[34] H. Hajipour, K. Hassler, T. Holz, L. Schönherr, and M. Fritz, "Codelmsec benchmark: Systematically evaluating and finding security vulnerabilities in black-box code language models," in 2024 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML), pp. 684–709, 2024.

[35] Y. Song, R. Liu, S. Chen, Q. Ren, Y. Zhang, and Y. Yu, "SecureSQL: Evaluating data leakage of large language models as natural language interfaces to databases," in Findings of the Association for Computational Linguistics: EMNLP 2024 (Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, eds.), (Miami, Florida, USA), pp. 5975–5990, Association for Computational Linguistics, Nov. 2024.

[36] W. Zhang, Y. Wang, and M. Fan, "Towards robustness ofnbsp;larglanguage models onnbsp;text-to-sql task: An adversarial andnbsp;crossdomain investigation," in Artificial Neural Networks and Machine Learning – ICANN 2023: 32nd International Conference on Artificial Neural Networks, Heraklion, Crete, Greece, September 26–29, 2023, Proceedings, Part V, (Berlin, Heidelberg), p. 181–192, Springer-Verlag, 2023.

[37] X. Peng, Y. Zhang, J. Yang, and M. Stevenson, "On the vulnerabilities of text-to-sql models," in 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE), pp. 1–12, 2023.

[38] J. Zhang, Y. Zhou, B. Hui, Y. Liu, Z. Li, and S. Hu, "TrojanSQL: SQL injection against natural language interface to database," in Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (H. Bouamor, J. Pino, and K. Bali, eds.), (Singapore), pp. 4344–4359, Association for Computational Linguistics, Dec. 2023.

[39] R. Pedro, M. E. Coimbra, D. Castro, P. Carreira, and N. Santos, " Prompt-to-SQL Injections in LLM-Integrated Web Applications: Risks and Defenses ," in 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), (Los Alamitos, CA, USA), pp. 76–88, IEEE Computer Society, May 2025.

[40] M. Nazzal, I. Khalil, A. Khreishah, and N. Phan, "Promsec: Prompt optimization for secure generation of functional source code with large language models (llms)," in Proceedings of the 2024 on ACM SIGSAC
Conference on Computer and Communications Security, CCS '24, (New York, NY, USA), p. 2266–2280, Association for Computing Machinery, 2024.

[41] I. D. Fagadau, L. Mariani, D. Micucci, and O. Riganelli, "Analyzing prompt influence on automated method generation: An empirical study with copilot," in Proceedings of the 32nd IEEE/ACM International

Conference on Program Comprehension, ICPC '24, (New York, NY, USA), p. 24–34, Association for Computing Machinery, 2024.

[42] Y. Sasaki, H. Washizaki, J. Li, D. Sander, N. Yoshioka, and Y. Fukazawa, "Systematic literature review of prompt engineering patterns in software engineering," in 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 670–675, 2024.

[43] M. Hamdi and L. D. Kim, "A prompt-based approach for software development," in 2023 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 1612–1614, 2023.

[44] S. Blog, "Prompts are programs." https://blog.sigplan.org/2024/10/22/prompts-are-programs/, 2024. Accessed: 2025-01-06.

[45] C. Liu, X. Bao, H. Zhang, N. Zhang, H. Hu, X. Zhang, and M. Yan, "Guiding chatgpt for better code generation: An empirical study," in 2024 IEEE International Conference on Software Analysis, Evolution
and Reengineering (SANER), pp. 102–113, 2024.

[46] J. Li, Y. Zhao, Y. Li, G. Li, and Z. Jin, "Acecoder: An effective prompting technique specialized in code generation," ACM Trans. Softw. Eng. Methodol., vol. 33, Nov. 2024.

[47] J. Li, G. Li, Y. Li, and Z. Jin, "Structured chain-of-thought prompting for code generation," ACM Trans. Softw. Eng. Methodol., Aug. 2024. Just Accepted.

[48] C. Tony, N. E. D. Ferreyra, M. Mutas, S. Dhiff, and R. Scandariato, "Prompting techniques for secure code generation: A systematic investigation," 2024.