PRINCÍPIOS SOLID NO DESENVOLVIMENTO ORIENTADO A OBJETOS EM JAVA: REVISÃO DE LITERATURA

Arthur Moura Freitas[1]
Ubirany dos Reis Ribeiro[2]

Ubirany_ribeiro@hotmail.com

ÁREA DO CONHECIMENTO: Engenharias

PALAVRAS-CHAVE: SOLID; Engenharia de Software; Programação Orientada a Objetos; Java; Arquitetura de Software

1 INTRODUÇÃO

A crescente complexidade dos sistemas computacionais tem tornado a qualidade arquitetural um dos principais desafios da Engenharia de Software. A evolução contínua dos requisitos e a necessidade de manutenção frequente exigem que sistemas sejam projetados de forma a facilitar sua adaptação sem comprometer sua estrutura. Nesse contexto, a Programação Orientada a Objetos (POO) se destaca por fornecer mecanismos como encapsulamento, abstração, herança e polimorfismo, amplamente utilizados no desenvolvimento de software moderno. Entretanto, tais mecanismos não garantem, por si só, a qualidade arquitetural, sendo necessária a adoção de princípios de projeto que orientem decisões ao longo do ciclo de desenvolvimento (Sommerville, 2019). Entre esses princípios destacam-se os SOLID, propostos por Robert C. Martin, voltados à construção de sistemas mais coesos, desacoplados e de fácil manutenção (Martin, 2002). O conjunto inclui SRP, OCP, LSP, ISP e DIP, amplamente utilizados em arquiteturas modernas, especialmente no ecossistema Java, por favorecerem extensibilidade, testabilidade e reutilização de código. Estudos indicam que a aplicação desses princípios está associada à melhoria de atributos como manutenibilidade, coesão e redução de acoplamento (Singh; Hassan, 2015). Além disso, pesquisas recentes mostram sua relevância em contextos modernos, como aprendizado de máquina e refatoração de sistemas legados (Cabral *et al.,* 2024; Yanakiev; Lazar; Capiluppi, 2025). Assim, este trabalho analisa os fundamentos dos princípios SOLID e suas contribuições para o desenvolvimento orientado a objetos em Java.

2 METODOLOGIA

Este estudo caracteriza-se como uma revisão narrativa da literatura, de abordagem qualitativa, descritiva e analítica. O objetivo foi compreender as contribuições dos

[1] Acadêmico do 4º período do curso de Ciência da Computação do Centro Universitário Vértice - Univértix
[2] Professor(a) do curso de Ciência da Computação do Centro Universitário Vértice - Univértix

princípios SOLID para a qualidade do desenvolvimento orientado a objetos em Java (Prodanov; Freitas, 2013). A pesquisa foi realizada nas bases Google Scholar, IEEE Xplore e arXiv, além de livros clássicos da Engenharia de Software. Foram utilizados descritores relacionados a SOLID, arquitetura de software e qualidade de código. Foram incluídos estudos que abordassem os princípios SOLID no contexto de desenvolvimento orientado a objetos, refatoração ou qualidade de software. Foram excluídos materiais não científicos, como blogs e conteúdos sem revisão técnica. Os dados foram analisados de forma qualitativa, identificando convergências e padrões relacionados aos impactos dos princípios sobre coesão, acoplamento, testabilidade e manutenibilidade (Martin, 2002; Sommerville, 2019).

3 RESULTADOS E DISCUSSÃO

A literatura analisada indica consenso de que os princípios SOLID contribuem significativamente para a melhoria da qualidade estrutural de sistemas orientados a objetos. De forma integrada, eles reduzem acoplamento, aumentam coesão e facilitam manutenção e evolução de software. O SRP promove separação clara de responsabilidades, reduzindo impactos de alterações. O OCP permite extensão sem modificação do código existente, aumentando a estabilidade do sistema. O LSP garante consistência em hierarquias de herança, evitando comportamentos inesperados. O ISP reduz dependências desnecessárias ao promover interfaces mais específicas. Já o DIP reduz acoplamento ao direcionar dependências para abstrações. Em conjunto, esses princípios favorecem arquiteturas mais flexíveis e testáveis, especialmente em aplicações Java, onde mecanismos de orientação a objetos facilitam sua implementação (Martin, 2002; Fowler, 2019). Estudos recentes indicam que sua aplicação também é eficaz em domínios modernos, como aprendizado de máquina, melhorando a organização do código e sua manutenibilidade (Cabral *et al.,* 2024). Além disso, sua utilização em refatoração de sistemas legados contribui para redução de dívida técnica e melhoria estrutural (Yanakiev; Lazar; Capiluppi, 2025). Entretanto, sua aplicação deve ser contextualizada. Em sistemas pequenos, o uso excessivo pode aumentar a complexidade sem ganhos proporcionais. Assim, sua adoção deve considerar o contexto do projeto, evitando abstrações desnecessárias (Sommerville, 2019).

4 CONSIDERAÇÕES FINAIS

Os princípios SOLID representam um conjunto consolidado de diretrizes para o desenvolvimento orientado a objetos, contribuindo para a construção de sistemas mais organizados, flexíveis e sustentáveis. A revisão realizada demonstra que sua aplicação melhora atributos essenciais da qualidade de software, como coesão, acoplamento, testabilidade e manutenibilidade. Além disso, sua relevância se mantém em cenários contemporâneos, como aprendizado de máquina e modernização de sistemas legados, evidenciando sua aplicabilidade além do contexto tradicional da POO. Contudo, sua aplicação deve ser criteriosa, considerando o porte e a complexidade do sistema. Conclui-se que os princípios SOLID permanecem como

referência fundamental na Engenharia de Software, especialmente no desenvolvimento em Java, contribuindo para arquiteturas mais robustas e adaptáveis.

REFERÊNCIAS

CABRAL, Raphael; KALINOWSKI, Marcos; BALDASSARRE, Maria Teresa; VILLAMIZAR, Hugo; ESCOVEDO, Tatiana; LOPES, Hélio. Investigating the impact of SOLID design principles on machine learning code understanding. arXiv preprint, arXiv:2402.05337, 2024. DOI: 10.48550/arXiv.2402.05337. Disponível em: https://arxiv.org/abs/2402.05337. Acesso em: 01 jul. 2026.

FOWLER, Martin. Refatoração: aperfeiçoando o projeto de código existente. 2. ed. Porto Alegre: Bookman, 2019.

LISKOV, Barbara; WING, Jeannette M. A behavioral notion of subtyping. **ACM Transactions on Programming Languages and Systems**, New York, v. 16, n. 6, p. 1811-1841, nov. 1994. Disponível em: https://doi.org/10.1145/197320.197383. Acesso em: 01 jul. 2026.

MARTIN, Robert C. Design principles and design patterns. Object Mentor, 2000. Disponível em: http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf. Acesso em: 01 jul. 2026.

MARTIN, Robert C. Agile software development: principles, patterns, and practices. Upper Saddle River, NJ: Pearson Education, 2002.

MARTIN, Robert C. Arquitetura Limpa: o guia do artesão para estrutura e design de software. Tradução de Cássio de Trapani Gomes. Rio de Janeiro: Alta Books, 2019.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de. Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico. 2. ed. Novo Hamburgo: Feevale, 2013.

SINGH, Harmeet; HASSAN, Syed Imtiyaz. Effect of SOLID design principles on quality of software: an empirical assessment. **International Journal of Scientific & Engineering Research**, v. 6, n. 4, p. 1321-1324, abr. 2015.

SOMMERVILLE, Ian. Engenharia de Software. 10. ed. São Paulo: Pearson, 2019.

YANAKIEV, Ivan; LAZAR, Bogdan-Mihai; CAPILUPPI, Andrea. Applying SOLID principles for the refactoring of legacy code: an experience report. **Journal of Systems and Software**, v. 220, p. 112254, 2025. Disponível em: https://doi.org/10.1016/j.jss.2024.112254. Acesso em: 01 jul. 2026.