Impactos do Paradigma da Programação Orientada a Objetos no Desenvolvimento de Sistemas: Uma Análise de seus Pontos Positivos e Negativos

Esp. Deyvison Samuel Gomes do Nascimento[1], Renan Jucá da Silva[1], Esp. Marcos Ramon Paulino Resende [1]

[1]Instituto Federal do Piauí (IFPI)
Caixa Postal 64.260-000 – Piripiri – PI – Brasil

{deyvisonnascimento2025,renanjuca2004,marcosramon.prof10}@gmail.com

Abstract. *This article analyzes the impacts of the Object-Oriented Programming (OOP) paradigm on software development, highlighting its influence on code structuring, reuse, and maintenance. The approach demonstrates how OOP principles contribute to system modularity and readability, promoting more organized and scalable development practices. To support this analysis, data collected through surveys with professionals in the field are presented, along with a comparative table of programming paradigms. The study also encourages critical reflection on the advantages and limitations of this paradigm, considering the technical, operational, and contextual aspects involved in its adoption.*

Resumo. *Este artigo analisa os impactos do paradigma da Programação Orientada a Objetos (POO) no desenvolvimento de software, destacando sua influência na estruturação, reutilização e manutenção do código. A abordagem evidencia como os princípios da POO contribuem para a modularidade e legibilidade dos sistemas, favorecendo práticas de desenvolvimento mais organizadas e escaláveis. Para isso, são apresentados dados coletados por meio de pesquisa com profissionais formados na área, além de uma tabela comparativa entre paradigmas de programação. O trabalho também promove uma reflexão crítica sobre as vantagens e limitações desse paradigma, considerando aspectos técnicos, operacionais e contextuais envolvidos em sua adoção.*

1. Introdução

Com o crescimento contínuo da complexidade dos sistemas computacionais e a necessidade de soluções mais eficientes, reutilizáveis e fáceis de manter, a forma como o software é estruturado tem ganhado destaque entre profissionais e pesquisadores da área (Caputo 2006). Nesse contexto, a Programação Orientada a Objetos (POO) consolida-se como uma das abordagens mais adotadas no desenvolvimento de sistemas modernos, oferecendo um modelo que busca aproximar a lógica da programação das estruturas e relações do mundo real (Afonso 2013; DÁvila and Giraffa 2023).

Embora os fundamentos da POO, como encapsulamento, abstração, herança e polimorfismo, sejam amplamente ensinados e promovam vantagens como modularidade e reutilização de código, sua aplicação prática nem sempre é isenta de dificuldades.A dificuldade de muitos desenvolvedores em aplicar corretamente os conceitos da Programação Orientada a Objetos pode comprometer a eficiência dos projetos ou desviá-los dos princípios fundamentais do paradigma. Essa discrepância entre o que se aprende na teoria e o que se pratica no desenvolvimento real levanta reflexões relevantes sobre a eficácia da POO em diferentes contextos profissionais. (Cardoso 2023; Costa 2011).

Além disso, a POO é apenas uma entre diversas abordagens existentes no campo da programação. Paradigmas como o estruturado, funcional, lógico e imperativo apresentam diferentes formas de organizar o pensamento computacional e solucionar problemas por meio de código. Entender as características e aplicações desses modelos é essencial para escolher, de maneira crítica e contextualizada, a estratégia mais adequada para cada projeto (Santos 2013; Silva 2015).

Para compreender melhor a relevância e os impactos reais da Programação Orientada a Objetos na prática profissional, é necessário considerar não apenas os aspectos teóricos, mas também as percepções e experiências daqueles que atuam diretamente no desenvolvimento de software. Muitas vezes, a forma como a POO é utilizada em ambientes corporativos e acadêmicos revela adaptações, simplificações ou até mesmo distorções em relação ao modelo idealizado (Reis et al. 2015; Zanetti et al. 2023).

Em síntese, o presente artigo tem como objetivo apresentar uma análise sobre a importância do uso da Programação Orientada a Objetos (POO) no desenvolvimento de sistemas, evidenciando suas principais características, vantagens e limitações. A proposta é oferecer uma reflexão crítica sobre como esse paradigma contribui para a organização, manutenção e escalabilidade do código, considerando critérios como integração com novas tecnologias, facilidade de entendimento, suporte à colaboração e adaptação a diferentes plataformas. A partir dessa abordagem, busca-se fornecer subsídios teóricos e práticos que ajudem profissionais e estudantes a compreenderem os impactos positivos e os desafios da adoção da POO, promovendo escolhas mais conscientes e eficazes no processo de desenvolvimento de software.

O restante deste artigo está estruturado da seguinte forma: a Seção 2 apresenta o referencial teórico, abordando os principais paradigmas de programação, incluindo o Paradigma de Programação Imperativo, o Paradigma de Programação Orientado a Objetos, a Programação Funcional e a Programação Lógica; a Seção 3 descreve a metodologia adotada, bem como os materiais e métodos utilizados na análise; a Seção 4 discute os principais resultados obtidos com base na comparação entre as ferramentas estudadas; por fim, a Seção 5 apresenta a conclusão do estudo e as considerações finais acerca das contribuições e possíveis desdobramentos futuros.

2. Fundamentação Teórica

A fundamentação teórica apresenta uma breve revisão dos principais paradigmas de programação, com o intuito de construir uma base conceitual para a análise proposta. São abordados os paradigmas imperativo, orientado a objetos, funcional e lógica, destacando suas características, aplicações e contribuições para o desenvolvimento de sistemas.

2.1. Paradigma de Programação Imperativo

O paradigma de programação imperativo baseia-se na execução sequencial de instruções e no uso explícito de variáveis para representar e modificar o estado do programa ao longo do tempo. Caracteriza-se por comandos de atribuição, estruturas de controle de fluxo como condicionais e laços de repetição, e pelo detalhamento minucioso das etapas de processamento. A reutilização de código é frequentemente feita por meio de sub-rotinas e funções. (Ferreira 2014).

2.2. Paradigma de Programação Orientada a Objetos

Historicamente, o paradigma orientado a objetos surgiu na década de 1970, antecedendo até mesmo a criação da linguagem Java . No entanto, foi a partir da ampla adoção do Java que essa abordagem passou a ganhar maior visibilidade e reconhecimento no campo da programação (Mendes 2009).

O paradigma orientado a objetos foi desenvolvido com o intuito de aproximar as linguagens de programação da maneira como os seres humanos compreendem e interagem com o mundo. Nesse modelo, utiliza-se o conceito de "objetos"para representar entidades do mundo real, como uma mesa, um livro ou uma pessoa, permitindo uma organização do código mais intuitiva e baseada em abstrações (Oberleitner and Masiero 2021).

Atualmente, a orientação a objetos também é utilizada em contextos educacionais, especialmente no ensino de programação, sendo aplicada por meio de recursos didáticos e interativos, que contribui para tornar o aprendizado mais acessível e envolvente (Camargo et al. 2020; Zanetti and Borges 2021).

2.3. Paradigma de Programação Funcional

O paradigma de programação funcional destaca-se pela ênfase na avaliação de expressões e na aplicação de funções matemáticas puras, cujo resultado depende exclusivamente dos valores de entrada, sem causar efeitos colaterais, além de valorizar a imutabilidade, ou seja, a não modificação de variáveis externas. Essa abordagem é ideal para algoritmos de busca, ordenação e processamento de eventos. (Cardoso 2023).

2.4. Paradigma de Programação Lógica

A programação lógica é um paradigma declarativo que se fundamenta na lógica matemática para a construção de programas, sendo amplamente utilizada em sistemas de banco de dados. Nessa abordagem, definem-se fatos e regras que permitem ao sistema realizar inferências automáticas, chegando a conclusões a partir das informações fornecidas (Costa 2011).

3. Metodologia, Materiais e Métodos

Nesta seção, apresenta a metodologia utilizada na análise do uso do paradigma da POO, descreve os materiais e os métodos adotados para fundamentar e evidenciar o conteúdo proposto.

3.1. Metodologia

A pesquisa teve início com uma revisão bibliográfica centrada na Programação Orientada a Objetos (POO), considerada um dos paradigmas mais amplamente utilizados no desenvolvimento

de software. Essa etapa teve como objetivo compreender os fundamentos essenciais da POO e analisar como seus conceitos são abordados em livros técnicos, publicações científicas e materiais acadêmicos. A intenção foi construir uma base teórica sólida sobre o paradigma, ao mesmo tempo em que se buscou identificar possíveis lacunas entre o conteúdo ensinado em ambientes educacionais e sua aplicação prática no contexto profissional. Essa análise crítica permitiu observar não apenas os benefícios atribuídos à POO pela literatura, mas também as limitações e desafios enfrentados pelos desenvolvedores em sua adoção no cotidiano.

Com base nos conhecimentos obtidos na revisão teórica, foi elaborado um questionário voltado a profissionais atuantes no desenvolvimento de software, com diferentes níveis de experiência. O instrumento foi estruturado com perguntas objetivas que buscavam explorar desde a compreensão teórica dos conceitos da POO até relatos de experiências práticas, incluindo dificuldades enfrentadas na aplicação de princípios como herança, encapsulamento e polimorfismo em projetos reais. A escolha pelo uso de questionários teve como propósito permitir uma análise comparativa entre o aprendizado formal e a vivência prática, contribuindo para uma compreensão mais ampla do impacto da formação acadêmica no preparo dos profissionais. Os dados obtidos servirão como base para refletir sobre como o paradigma orientado a objetos é utilizado na prática e de que forma influencia aspectos como qualidade, manutenção e evolução dos sistemas desenvolvidos.

3.2. Materiais

A pesquisa contou com a utilização de dois computadores com acesso à internet, os quais permitiram a realização de buscas por referencial teórico em plataformas como o Google Acadêmico. Além disso, foi utilizado um formulário eletrônico desenvolvido no Google Forms, destinado à coleta de dados relacionados ao uso da Programação Orientada a Objetos (POO).

3.3. Métodos

O método adotado nesta pesquisa caracteriza-se como uma abordagem exploratória de natureza quali-quantitativa, envolvendo revisão bibliográfica e aplicação de questionário, com foco na análise comparativa entre a Programação Orientada a Objetos (POO) e outros paradigmas de programação. Para isso, foram seguidos os seguintes passos:

1. Foi realizada uma pesquisa bibliográfica sobre os fundamentos e aplicações da POO, com base em fontes do Google Acadêmico;
2. Em seguida, elaborou-se um questionário para programadores, com foco em suas percepções sobre a POO. Duas perguntas foram selecionadas para análise;
3. Os dados foram organizados e analisados qualitativamente, buscando padrões nas respostas;
4. Por fim, elaborou-se uma síntese crítica relacionando os resultados com a teoria, comparando a POO a outros paradigmas.

4. Resultados

Na próxima seção, serão apresentados os resultados da pesquisa, incluindo a análise dos paradigmas de programação e a interpretação das respostas obtidas nas entrevistas com profissionais da área. O objetivo é complementar a revisão teórica com dados práticos, ampliando a compreensão sobre a aplicação dos diferentes paradigmas no desenvolvimento de software.

4.1. Análise dos Paradigmas de Programação

Ao serem analisados os paradigmas de programação, é possível sintetizar algumas de suas principais características, observando que cada um possui modelos de execução, mecanismos de reutilização de código e usos específicos, conforme demonstrado na Tabela 1. Essa tabela atua como um recurso comparativo que organiza e evidencia as diferenças entre os paradigmas abordados na fundamentação teórica, facilitando a compreensão dos contextos mais apropriados para aplicação de cada abordagem. A comparação reforça a Programação Orientada a Objetos como uma alternativa robusta, especialmente em cenários que exigem manutenção, escalabilidade e estrutura modular, ao mesmo tempo em que destaca a importância de se considerar as necessidades específicas de cada projeto na escolha do paradigma mais adequado.

Comparativo dos Paradigmas de Programação

Característica	**Imperativo**	**Orientado a Objetos (POO)**	**Funcional**	**Lógico**
Conceito central	Comandos sequenciais e variáveis	Objetos com atributos e métodos	Funções puras e imutabilidade	Regras e inferência lógica
Modelo de execução	Passos sequenciais	Comunicação entre objetos	Avaliação de funções	Inferência a partir de fatos
Reutilização de código	Sub-rotinas, loops	Herança, polimorfismo	Composição de funções	Regras reutilizáveis
Exemplos de linguagens	C, Pascal, Fortran	Java, C#, Python (POO)	Haskell, Scheme, Elixir	Prolog
Uso comum	Sistemas operacionais, jogos	Grandes sistemas, softwares complexos	Aplicações matemáticas, paralelas	IA, sistemas especialistas

Tabela 1. Fonte: Próprio autor.

4.2. Análise de Entrevista

Os resultados da pesquisa, apresentados na Figura 1, revelam as situações em que os profissionais consideram a Programação Orientada a Objetos (POO) mais apropriada. Aproximadamente 42,86% dos participantes destacaram que a POO é a melhor escolha em sistemas complexos que exigem manutenção a longo prazo. Outros 28,57% indicaram que ela é particularmente útil em projetos com equipes grandes e código colaborativo, enquanto uma proporção equivalente (28,57%) mencionou sua aplicabilidade quando há a necessidade de uma modelagem mais próxima da realidade. Esses resultados evidenciam a importância e a versatilidade da POO nos sistemas modernos.

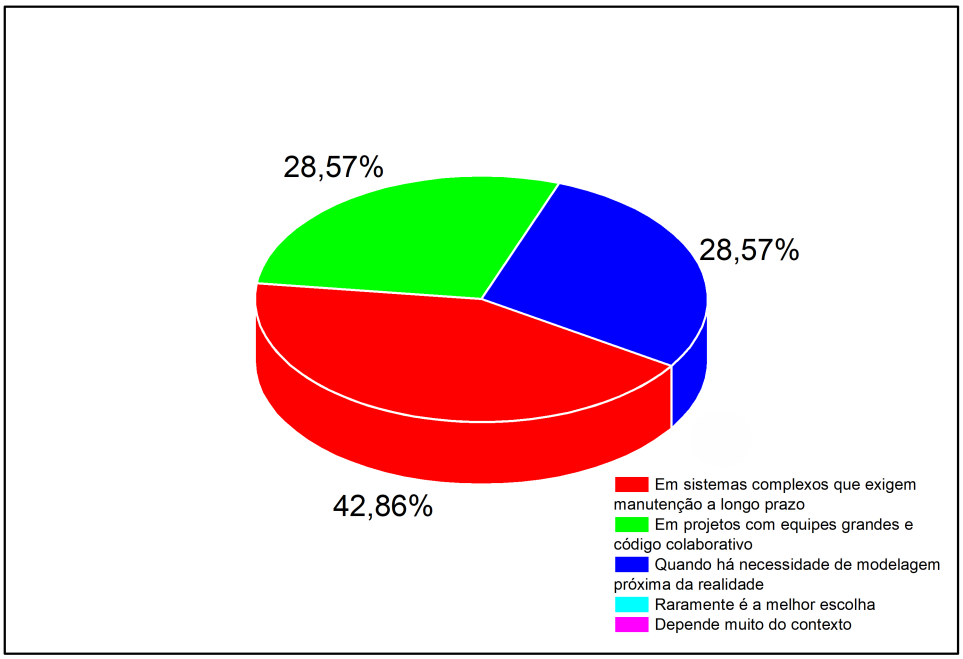

Figura 1. Fonte: Próprio autor.

Observa-se a Figura 2 que aproximadamente 57,14% dos entrevistados identificaram que a principal melhoria proporcionada pela utilização da Programação Orientada a Objetos ocorreu na organização e legibilidade do código. Esse dado sugere que, na prática, a orientação a objetos tem contribuído significativamente para uma maior clareza estrutural, o que facilita o entendimento, a manutenção e a evolução dos projetos desenvolvidos.

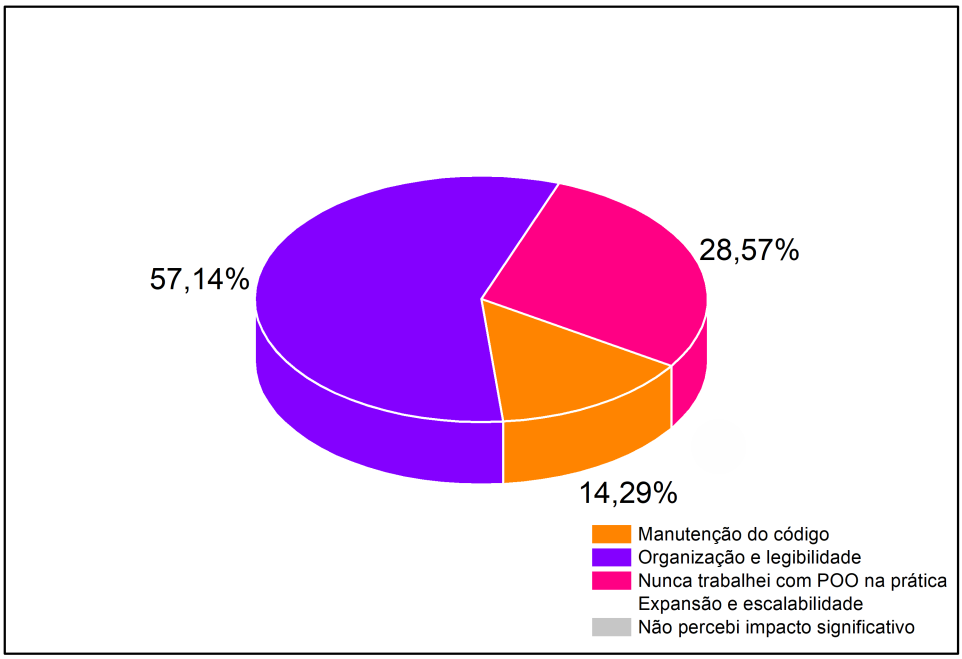

Figura 2. Fonte: Próprio autor.

Ademais, todos os entrevistados afirmaram já ter utilizado POO em diversos projetos, demonstrando segurança e familiaridade com o paradigma. Quando questionados sobre os principais benefícios percebidos, os aspectos mais mencionados foram: melhor organização do código, facilidade na manutenção e escalabilidade dos sistemas, além de uma modelagem mais clara e alinhada a problemas do mundo real. Esses resultados reforçam a importância da Programação Orientada a Objetos no desenvolvimento de soluções robustas, modulares e de fácil manutenção.

5. Conclusão e Considerações Finais

A Programação Orientada a Objetos (POO) mostrou-se eficaz para organizar sistemas de software de maneira clara e estruturada. Ao agrupar dados e comportamentos relacionados, o paradigma facilita a representação de sistemas complexos, aproximando o código dos conceitos do domínio aplicado. Essa abordagem melhora a legibilidade e a organização dos projetos, especialmente quando há necessidade de manutenção e crescimento do sistema.

Os resultados indicam que a POO contribui para a criação de sistemas mais flexíveis e fáceis de adaptar. A independência entre os componentes permite que modificações sejam feitas com menor impacto no restante do programa. Além disso, a reutilização de partes do código reduz o tempo de desenvolvimento e diminui a ocorrência de erros, o que é fundamental para garantir a qualidade do software.

Para futuras pesquisas, pretende-se ampliar o número de participantes, visando aumentar a diversidade de experiências analisadas. Também pretende-se investigar a aplicação da

POO em diferentes contextos, como sistemas móveis, distribuídos e em nuvem. Essas investigações têm o potencial de fornecer insights mais aprofundados sobre as vantagens e desafios do paradigma em ambientes variados.

Referências

[Afonso 2013] Afonso, N. M. M. (2013). Da tarefa ao projeto: uma visão construtivista do ensino da programação orientada a objetos. Master's thesis, Universidade do Minho (Portugal).

[Camargo et al. 2020] Camargo, R. G., Ribeiro, C. E., Sordi Junior, F., Anastácio, P. R., and Merlin, J. R. (2020). Utilização de pygame para ensino e aprendizado de orientação a objetos. *Revista Brasileira de Informática na Educação*, 28(1):227–252.

[Caputo 2006] Caputo, G. M. (2006). Sistema computacional para o processamento textual de patentes industriais. *Universidade Federal do Rio de Janeiro*, pages 1–142.

[Cardoso 2023] Cardoso, R. (2023). Programação funcional e poo: veja as diferenças dos paradigmas. Blog Locaweb.

[Costa 2011] Costa, H. (2011). Programação lógica.

[DÁvila and Giraffa 2023] DÁvila, W. and Giraffa, L. (2023). Ensino de programação orientada a objetos para iniciantes: Uma metodologia para programação criativa. In *Simpósio Brasileiro de Informática na Educação (SBIE)*, pages 335–344. SBC.

[Ferreira 2014] Ferreira, T. A. d. C. (2014). *Um estudo sobre a correspondência entre programação funcional com continuações e programação imperativa Single assignment*. PhD thesis.

[Mendes 2009] Mendes, D. R. (2009). *Programação Java com ênfase em orientação a objetos*. Novatec, São Paulo.

[Oberleitner and Masiero 2021] Oberleitner, A. and Masiero, A. A. (2021). *Programação orientada a objetos: da teoria à prática*. Editora Senac São Paulo, São Paulo.

[Reis et al. 2015] Reis, J. N., Vale, G., and Costa, H. (2015). Manutenibilidade de tecnologias para programação de linhas de produtos de software: um estudo comparativo. In *Simpósio Brasileiro de Qualidade de Software*, Salvador. SBC.

[Santos 2013] Santos, R. (2013). *Introdução à programação orientada a objetos usando Java*. Elsevier, Rio de Janeiro, 2 edition.

[Silva 2015] Silva, M. C. (2015). Programação orientada a objetos versus programação estruturada: comparativo de paradigmas.

[Zanetti and Borges 2021] Zanetti, H. A. and Borges, M. A. (2021). Por que estimular a aprendizagem significativa no ensino de programação orientada a objetos? In *Simpósio Brasileiro de Educação em Computação (EDUCOMP)*, pages 290–295. SBC.

[Zanetti et al. 2023] Zanetti, H. A. P., Borges, M. A. F., and Ricarte, I. L. M. (2023). Comfapoo: Método de ensino de programação orientada à objetos baseado em aprendizagem significativa e computação física. *Revista Brasileira de Informática na Educação*, 31:01–30.