Impacto das Métricas de Software no Índice de Manutenção - Uma Revisão Sistemática

Allana C. D. M. de Ávila[1], Camila Bastos[1]

[1]Universidade Prof. Edson Antônio Velano (Unifenas)
Alfenas – MG – Brazil

allana.avila@aluno.unifenas.br, camila.bastos@unifenas.br

Abstract. *This article describes the importance of software quality, highlighting its ability to meet users' needs and add value to the product. However, the lack of references compromises the investigation of analyzes in the quantitative quality management of object-oriented software. The central proposal involves the development of an automated tool to calculate and present revisions in real time, simplifying assessment and enabling developers to make proactive adjustments. This approach aims to contribute significantly to the continuous improvement of product quality throughout development.*

Resumo. *Este artigo descreve a importância da qualidade do software, ressaltando sua capacidade de atender às necessidades dos usuários e agregar valor ao produto. No entanto, a falta de referências compromete a investigação das análises no gerenciamento quantitativo da qualidade do software orientado a objetos. A proposta central envolve o desenvolvimento de uma ferramenta automatizada para calcular e apresentar em tempo real revisões, simplificando a avaliação e capacitando os desenvolvedores a realizar ajustes proativos. Essa abordagem visa contribuir significativamente para a melhoria contínua da qualidade do produto ao longo do desenvolvimento.*

1. Introdução

A qualidade do software é um fator crucial para o sucesso e para a aceitação de produtos e de serviços no mercado atual. Garantir que um software atenda às necessidades e às expectativas dos usuários, bem como dos demais stakeholders, é essencial para agregar valor ao produto e obter vantagem competitiva. Nesse contexto, a medição e a avaliação da qualidade interna do software, especialmente a partir do código-fonte, desempenham um papel fundamental para assegurar a qualidade geral do produto [1].

Já as métricas de software, têm se mostrado valiosas ferramentas na Engenharia de Software para avaliar a qualidade dos projetos em diferentes fases de desenvolvimento. Ao fornecerem medidas quantitativas e objetivas, essas métricas permitem compreender o nível de qualidade do software, avaliar a capacidade do processo de desenvolvimento e identificar áreas de melhoria. Dessa forma, as equipes de desenvolvimento podem realizar ajustes ao longo do ciclo de vida do software, visando melhorar a qualidade e garantir a entrega de um produto confiável e de acordo com as necessidades dos usuários [2].

Ao se tratar de softwares orientados a objetos, surge um desafio adicional em relação às métricas de software: a falta de valores de referência conhecidos para muitos índices. Isso pode limitar o uso efetivo dessas métricas na indústria de software,

tornando complexo o gerenciamento quantitativo da qualidade do software orientado a objetos [2][3]. Diante desse cenário, esta pesquisa buscou investigar e definir métodos para a obtenção de valores de referência de métricas de software orientado a objetos, a fim de identificar seu impacto no Índice de Manutenção.

O presente trabalho tem como objetivo principal explorar a importância das métricas de software para a avaliação da qualidade em projetos orientados a objetos. No artigo [A1][A2], foi conduzida uma revisão sistemática da literatura com o objetivo de identificar os principais indicadores de qualidade de software utilizados. Entretanto, o referido trabalho não apresentou a sequência entre essas métricas e o índice de manutenção, deixando uma lacuna na compreensão do impacto dessas métricas na qualidade do software em termos de sua manutenção. Para preencher essa lacuna de pesquisa, o presente estudo busca investigar e definir as relações entre as métricas de qualidade de software e o índice de manutenção em projetos orientados a objetos. Ao fazê-lo, visa-se contribuir para o avanço da área de Engenharia de Software, proporcionando uma visão mais abrangente e embasada para a medição e para a gestão eficaz da qualidade em projetos dessa natureza.

Nas próximas seções, serão apresentados os fundamentos teóricos relacionados às métricas de software, sua importância na Engenharia de Software e os desafios específicos no contexto de softwares orientados a objetos. Em seguida, será detalhada a metodologia utilizada para a pesquisa e para o alcance dos objetivos propostos. Por fim, serão apresentados os resultados obtidos, as análises e as discussões relevantes, bem como as contribuições desse estudo para a área de Engenharia de Software.

2. Revisão Sistemática da Literatura

A Revisão Sistemática da Literatura é uma abordagem metodológica que permite identificar, analisar e sintetizar os documentos disponíveis na literatura científica sobre um tema específico. Uma RLS pode ser realizada para explorar as pesquisas já existentes sobre o uso de métricas de software na avaliação da qualidade de projetos orientados a objetos. A questão de pesquisa definida para guiar a seleção de trabalhos relevantes foi:

Quais métricas de qualidade impactam no Índice de Manutenção de Software Orientado à Objeto? Como elas impactam? Como elas são calculadas?

Com base na questão, uma string de busca foi definida para auxiliar na padronização da pesquisa no repositório de trabalhos científicos. A string é composta por palavras-chave relevantes para o assunto associado ao artigo:

"Abstract": "software quality", "software quality assurance", "software metrics", "software measurements", "maintenance index".

O nome e o endereço eletrônico do repositório utilizado estão listados na Tabela 1. A opção de utilizar um único repositório de pesquisa é fundamentada em sua especialização na área de Engenharia de Software e de métricas de qualidade, oferecendo uma ampla gama de fontes relevantes, como revistas científicas e

conferências [4]. Essa escolha visa garantir foco e eficiência na busca por artigos de alta qualidade, otimizando os recursos disponíveis para a pesquisa.

Tabela 1. Repositórios de artigos científicos utilizados

Repositórios	Endereços Eletrônicos
IEEE Xplore Digital Library	http://ieeexplore.ieee.org

Para serem selecionados, os artigos deverão possuir acesso livre ao seu conteúdo e serem publicados a partir do ano 2000 (critérios de seleção). Não foram definidas restrições quanto ao idioma do artigo. Após a definição do protocolo, a seleção dos artigos relevantes foi realizada por meio da execução dos seguintes passos:

- Passo 1: Foi definida uma string de busca específica , utilizada no repositório selecionado para coletar artigos de conferências, de revistas publicadas no período de 2000 a 2023. Após a busca, os trabalhos foram analisados, retirando aqueles que não atendiam a esses critérios.
- Passo 2: Em seguida, foi realizada a seleção dos trabalhos relevantes por meio de leitura do título, do resumo e das palavras-chave dos trabalhos obtidos na busca.
- Passo 3: Os artigos resultantes na primeira seleção (Passo 2) foram reunidos em um único conjunto, facilitando o processo de análise para a verificação dos trabalhos duplicados.
- Passo 4: Foi realizada a seleção secundária. Nos artigos resultantes do Passo 3, foi realizada a releitura do resumo e a leitura das considerações finais elaboradas de cada artigo.

A quantidade de artigos obtidos com a execução desses passos é apresentada na Tabela 2. A segunda coluna contém a Quantidade Inicial (QI) de artigos obtidos após a realização do Passo 1. A terceira coluna contém a quantidade de artigos obtidos após a Seleção Primária (SP), em que foi realizada a leitura de títulos, dos resumos e das palavras-chave . Na quarta, na quinta e na sexta colunas, são apresentados os resultados obtidos com Seleção Secundária (SS), em que foi realizada a releitura do resumo e a leitura das conclusões. É apresentada a quantidade de artigos irrelevantes (IR), de artigos repetidos (RP), de artigos incompletos (IN) e de artigos selecionados em cada repositório (R).

Tabela 2. Quantidade de artigos selecionados

Repositórios	**QI**	**SP**	**SS**			
			IR	**RP**	**IN**	**R**
IEEE Xplore	2699	54	25	0	0	29

Os 29 artigos selecionados foram utilizados para realizar a análise qualitativa e a análise quantitativa descritas nas próximas seções.

3. Análise Qualitativa

O software orientado a objetos possui a proteção necessária para garantir um funcionamento eficiente, estabelecendo conexões entre várias subclasses. No entanto, é crucial ressaltar que a vedação entre classes é uma característica de um sistema de

software bem planejado. Dessa forma, a orientação, a exibição e a análise dos resultados das proporções das classes de software tornou-se cada vez mais importante no campo.

Com leitura dos artigos selecionados na RSL, foram encontradas métricas de qualidade que impactam no Índice de Manutenção do Software Orientado a Objeto (OMI).

- Coesão: a coesão é um grau em que os métodos dentro de uma classe estão relacionados entre si. Com isso, quanto maior a coesão, menor a probabilidade de que um erro de um elemento afete outro elemento do módulo. [12]
- Acoplamento: o acoplamento avalia o estado entre os módulos de um sistema. Com isso, quanto maior o acoplamento, maior a probabilidade de que uma mudança do módulo afete outros módulos. [12]
- Complexidade: a complexidade avalia a dificuldade de entendimento de um módulo. Com isso, quanto maior a dificuldade, maior a probabilidade de que uma mudança no módulo obtenha erros. [12]
- Abstração: a abstração é um ato de capacidade de um módulo que apresenta conceitos essenciais, ignorando detalhes irrelevantes. Com isso, quanto maior a abstração, menor a probabilidade de que uma mudança no módulo afete outros módulos. [2][12]
- Reusabilidade: a reusabilidade avalia a capacidade de um módulo que pode ser utilizado em diferentes partes de um sistema. Com isso, quanto mais pode ser reutilizado no módulo, menor a probabilidade de que uma mudança nesse módulo afete outros módulos. [2]
- Testabilidade: a testabilidade avalia a facilidade de testar um módulo. Com isso, é a métrica que mede a facilidade com que um módulo ou método pode ser testado para detectar erros ou bugs. [2]

A análise do Índice de Manutenção de Software Orientado a Objeto (OMI) é influenciada diretamente pelas métricas acima, sendo que a importância de cada métrica pode variar conforme o contexto do projeto. A Tabela 3 oferece uma análise detalhada dessas métricas, sendo que cada métrica é acompanhado de sua fórmula, uma descrição e o impacto estimado no Índice de Manutenção.

4. Discussão dos Resultados

Em relação às questões de pesquisa, foi possível extrair os seguintes resultados após a leitura e análise dos artigos da RSL.

Quais métricas de qualidade impactam no Índice de Manutenção de Software Orientado a Objeto?

Tabela 3. Métricas de Qualidade

Métrica	Fórmula	Descrição	Impacto no índice de manutenção	Referência
LCOM - Lack of Cohesion in Methods	LCOM = ((n/2) – (2* (n – 1))) + P	**N:** número total de métodos das classes **P:** é o número de conjuntos de métodos que acessam pelo menos uma variável de instância comum.	Quanto maior a coesão de uma classe, maior a clareza e o foco de seus métodos em uma única responsabilidade, facilitando a compreensão e a manutenção do código. Baixa coesão pode tornar o código menos classe e mais difícil de entender e de manter.	[A1] [A2] [A3] [A4] [A6] [A7] [A10] [A15] [A16] [A19] [A21] [A22] [A23] [A24] [A26] [A28] [A29]
TCC - Tight Class Cohesion	TCC = NDC / NP	**NDC:** é o número de conexões diretas entre métodos públicos na classe. **NP:** número máximo de pares que podem ser formados a partir de um conjunto com N elementos. Para ser calculado o número máximo de pares possíveis, é NP = [N * (N - 1)] / 2, n: métodos públicos.	Quanto maior a coerência melhor é a qualidade do código e menor é a probabilidade de que uma mudança em um módulo afete outros módulos.	[A4] [A23]
CBO - Coupling Between Objects	CBO = Ca + Ce	**Ca** (Acoplamentos aferentes): acoplamentos aferentes de uma classe que efetua a medida de outras classes que utilizam uma classe especifica . **Ce** (Acoplamento eferente): acoplamento eferente de uma classe que foi utilizado na medida de outras classes que utilizam uma classe especifica .	Quanto maior o número de pares, maior a sensibilidade a mudanças em outras partes do projeto e, portanto, a manutenção é mais difícil.	[A1] [A3] [A5] [A6] [A7] [A10] [A16] [A17] [A20] [A21] [A22] [A26] [A28] [A29]
DIT - Depth of Inheritance Tree	DIT = Comprimento máximo do caminho da classe até a raiz da árvore de herança	P e Q são as subclasses	Quanto mais profunda uma classe estiver na hierarquia, maior será o número de métodos que será herdado, tornando-a mais completa prevendo seu comportamento.	[A1] [A2] [A6] [A7] [A10] [A19] [A21] [A22] [A26] [A27] [A28] [A29]
RFC - Response for a Class	RS = {MI U all i {RI}}	**RS:** conjunto de todos os métodos em uma classe que podem ter respostas ou serem solicitados. **(all i {RI}):** conjunto de métodos chamados por todos os métodos da classe. **MI:** representa um método específico (Método i) dentro de uma classe, representa um único método na classe. **R:** conjunto de métodos chamados pelo método i **M**: conjunto de todos os métodos da classe	Quanto maior o número de métodos que uma classe pode responder, maior a complexidade dessa classe.	[A1] [A2] [A3] [A6] [A7] [A10] [A19] [A20] [A21] [A22] [A26] [A28] [A29]
WMC - Weighted Methods per Class	WMC = ∑Ci (para i = 1 a n)	**Ci:** representa a complexidade do método da classe; **n:** é o número total de métodos dentro da classe.	Quanto maior o número de métodos em uma classe, maior é o impacto potencial da herança dos métodos definidos nessa classe para as subclasses.	[A1] [A2] [A3] [A6] [A7] [A15] [A17] [A19] [A21] [A22] [A27] [A28] [A29]
CLOC - Count Lines of Code	Vn = Vn-1 + V0	**Vn:** número de linhas do código **Vn-1:** tamanho **V0:** tamanho inicial do software, primeira versão do projeto	O CLOC é um indicador do tamanho do código que pode ter um impacto positivo no índice de manutenção, fornecendo informações ao longo do tempo, garantindo que o software possa ser mantido de forma eficiente e sustentável.	[5] [A29]
CC - Complexidade Ciclomático	CC = E – N + P	**E:** é o número de arestas do grafo; **N:** é o número de nós; **P:** é o número de componentes conectados.	Quanto menor a complexidade, maior é a facilidade de manutenção e menor é a probabilidade de erros.	[A2] [6] [A8] [A10] [A16] [A17] [A18] [A19]
NOC - Número de Filhos	NOC (P + Q) = np + nQ - d	**Np:** é o número de subclasses imediatas da classe P; **nQ:** é o número de subclasses imediatas da classe Q; d: é número de filhos em comum entre P e Q.	Quanto maior o número de filhos, maior o reuso, pois a herança é uma forma de reuso. Quanto maior o número de filhos, maior a probabilidade de abstração indevida da classe pai.	[A1] [A2] [A3] [A6] [A7] [A10] [A19] [A21] [A22] [A26] [A27]

Na análise qualitativa dos artigos selecionados, ficou evidente que diversas análises de qualidade exercem influência direta sobre o Índice de Manutenção de Software Orientado a Objeto (OMI). Conforme listado na Tabela 3, destacam-se, entre os mais relevantes, a complexidade ciclomática, a coesão e o acoplamento entre as classes, assim como a taxa de defeitos e a duplicação de código. Essas conclusões contêm insights sobre a estrutura interna do código, sua manutenibilidade e a propensão a erros em futuras modificações.

Como elas impactam?

As métricas de qualidade desempenham papéis específicos no contexto do Índice de Manutenção de Software Orientado a Objeto (OMI). Por exemplo, uma complexidade ciclomática está intrinsecamente ligada à compreensão e à modificação de trechos de código. À medida que a complexidade aumenta, a manutenção torna-se mais desafiadora. Já a união e a proteção entre classes, afetam a interdependência das partes do sistema, influenciando a facilidade de realizar alterações sem impactar outras áreas do software. A taxa de defeitos e a duplicação de código indicam a qualidade global do código e sua propensão a erros, impactando diretamente a estabilidade e a confiabilidade do software. Em resumo, essas avaliações de qualidade exercem impactos diversos, mas todos relevantes no Índice de Manutenção de Software Orientado a Objeto.

Como elas são calculadas?

Conforme ilustrado na Tabela 3, o cálculo das métricas pode variar conforme a ferramenta utilizada, mas geralmente envolve uma contagem de elementos do código, como classes, métodos, linhas de código, superclasses, subclasses, entre outros. O objetivo é obter um valor numérico que indique a qualidade do código em relação à métrica em questão. Cada métrica possui uma fórmula específica para o cálculo, disponível nos documentos das ferramentas de análise de código-fonte.

Para calcular cada métrica, é necessário aplicar a fórmula específica associada a ela. Essas fórmulas dependem da métrica em questão e das ferramentas utilizadas para a análise de código-fonte. O propósito é obter um valor numérico que represente a qualidade do código em relação à métrica específica. Essas análises não são cruciais apenas para avaliar a qualidade do código ao longo do ciclo de vida do software, mas também para identificar áreas de possível aprimoramento na manutenção do software.

Para mensurar o impacto de cada métrica no Índice de Manutenção do software, foi definida uma escala de impacto que varia em Baixo, Médio, Alto e Sem Impacto. A classificação de cada métrica em uma dessas categorias varia de acordo com a quantidade de artigos que citaram tal métrica como sendo impactante no Índice de Manutenção:

- **Sem Impacto**. Indica que a métrica tem um impacto insignificante no índice de manutenção. Mudanças ou variações nessa métrica não terão um efeito significativo na manutenção do sistema.
- **Baixo Impacto (menos de 5 Artigos)**. Reflete um impacto relativamente pequeno no índice de manutenção. Um número limitado de artigos sugere que as mudanças podem ser gerenciadas com facilidade.

- **Médio Impacto (Entre 5 a 10 Artigos)**. Indica um impacto moderado no índice de manutenção. Com um número moderado de artigos, variações na métrica começam a ter um efeito prático na manutenção, exigindo alguma atenção.
- **Alto Impacto (Acima de 10 Artigos)**. Indica um impacto significativo no índice de manutenção. Um número significativo de artigos sugere que variações nesta métrica podem ter um impacto, exigindo atenção e esforço consideráveis para gerenciar e manter o sistema.

Considerando essa escala, na Tabela 4, são apresentados os resultados obtidos após análise de impacto de cada métrica no Índice de Manutenção.

Tabela 4. Escala no impacto ao índice de manutenção

	COESÃO	ACOPLAMENTO	TAMANHO DO CÓDIGO	COMPLEXIDADE	ABSTRAÇÃO
LCOM	◉	○	○	○	○
TCC	○	○	○	○	○
NOC	◉	○	◐	◐	●
CC	○	◐	◐	●	●
CLOC	○	○	○	○	●
WMC	◉	◐	◉	◐	○
RFC	◐	◉	◐	◉	◐
DIT	◐	◉	◉	●	○
CBO	◉	◉	◉	◉	◉

Sem Impacto: ● Baixo Impacto: ○ Médio Impacto: ◐ Alto Impacto: ◉

6. CONSIDERAÇÕES FINAIS

Neste artigo, foram apresentados os resultados obtidos com a execução de uma RSL para encontrar na literatura trabalhos que propuseram métricas de qualidade de software, suas fórmulas e qual seu impacto no Índice de Manutenção do Software. Após a seleção secundária, foram obtidos 29 artigos considerados relevantes para a questão de pesquisa.

Com este trabalho, foi possível consolidar um entendimento sobre as principais métricas de software, tais como coesão, acoplamento, tamanho do código, complexidade, abstração, reusabilidade e testabilidade e como elas impactam no Índice de Manutenção do Software. A combinação de análises quantitativas e qualitativas

proporcionou uma visão abrangente do tema, destacando a interação complexa entre essas métricas e o índice de manutenção.

Métricas como LCOM, TCC, NOC, CC, CLOC, WMC, RFC, DIT e CBO foram comprovadas em termos de seu impacto no desenvolvimento e na manutenção de software orientado a objetos. A compreensão obtida a partir desta pesquisa fornece insights para profissionais da área e para pesquisadores, orientando práticas de desenvolvimento e de manutenção mais eficientes. A continuidade deste trabalho poderá se concentrar no desenvolvimento de tecnologias que automatizam o cálculo dessas métricas e façam a integração com ambientes de desenvolvimento.

7. Referências

[1] DE SOUZA, Priscila Pereira et al. A utilidade dos valores referência de métricas na avaliação da qualidade de softwares orientados por objeto. 2016.

[2] FILÓ, Tarcisio Guerra Savino. Identificação de valores referência para métricas de softwares orientados por objetos. 2014.

[3] GAIA, Josiane Rosa de Oliveira. Manutenção de software e estudo de caso aplicado ao software Openbiblio. 2013.

[4] WILDE, Michelle. Biblioteca digital Ieee xplore. The Charleston Advisor , v. 17, n. 4, pág. 24-30, 2016.

[5] BAER, Nikolaus; ZEIDMAN, Roberto. Medindo a evolução do software com a mudança de linhas de código. Em: CATA . 2009. pág. 264-170.

[6] JURECZKO, Marian. Significado de diferentes métricas de software na previsão de defeitos. Engenharia de Software: An International Journal , v. 1, n. 1, pág. 86-95, 2011.

[7] MAXIM, Bruce R.; PRESSMAN, Roger S. Software Engineering: A Practitioner'S Approach. Britania Raya: McGraw-Hill Education, 2014.

[8] CHIDAMBER, Shyam R.; KEMERER, Chris F. Um conjunto de métricas para design orientado a objetos. Transações IEEE sobre engenharia de software , v. 6, pág. 476-493, 1994.

[9] FENTON, Norman E.; PFLEEGER, Shari Lawrence. Métricas de software: uma abordagem rigorosa e prática: Brooks. 1998.

[10] BASILI, Victor R.; BRIAND, Lionel C..; MELO, Walcélio L. Validação de métricas de design orientado a objetos como indicadores de qualidade. Transações IEEE sobre engenharia de software , v. 10, pág. 751-761, 1996.

[11] KANAKI, Kalliopi et al. Investigando a Associação entre Pensamento Algorítmico e Desempenho em Estudo Ambiental. Sustentabilidade , v. 14, n. 17, pág. 10672, 2022.

[12] PRESSMAN, Roger S.; MAXIM, Bruce R. Engenharia de software-9. McGraw Hill Brasil, 2021.

8. Apêndice - Artigos Selecionados

[A1]	S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," in IEEE Transactions on Software Engineering, vol. 20, no. 6, pp. 476-493, June 1994, doi: 10.1109/32.295895.
[A2]	O. P. Rotaru and M. Dobre, "Reusability metrics for software components," The 3rd ACS/IEEE International Conference onComputer Systems and Applications, 2005., Cairo, 2005, pp. 24-, doi: 10.1109/AICCSA.2005.1387023.
[A3]	M. Hitz and B. Montazeri, "Chidamber and Kemerer's metrics suite: a measurement theory perspective," in IEEE Transactions on Software Engineering, vol. 22, no. 4, pp. 267-271, April 1996, doi: 10.1109/32.491650.
[A4]	S. H. Mustafa and R. Alawneh, "The Impact of Transitive Class Relations on Measuring the Degree of Class Cohesion," 2019 International Arab Conference on Information Technology (ACIT), Al Ain, United Arab Emirates, 2019, pp. 63-69, doi: 10.1109/ACIT47987.2019.8991071.
[A5]	R. Harrison, S. Counsell and R. Nithi, "Coupling metrics for object-oriented design," Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No.98TB100262), Bethesda, MD, USA, 1998, pp. 150-157, doi: 10.1109/METRIC.1998.731240.
[A6]	S. Anwer, A. Adbellatif, M. Alshayeb and M. S. Anjum, "Effect of coupling on software faults: An empirical study," 2017 International Conference on Communication, Computing and Digital Systems (C-CODE), Islamabad, Pakistan, 2017, pp. 211-215, doi: 10.1109/C-CODE.2017.7918930.
[A7]	G. Alkadi and D. L. Carver, "Application of metrics to object-oriented designs," 1998 IEEE Aerospace Conference Proceedings (Cat. No.98TH8339), Snowmass, CO, USA, 1998, pp. 159-163 vol.4, doi: 10.1109/AERO.1998.682165.
[A8]	M. Alfadel, A. Kobilica e J. Hassine, "Evaluation of Halstead and Cyclomatic Complexity Metrics in Measuring Defect Density", *2017 9ª Conferência e Exposição IEEE-GCC (GCCCE)* , Manama, Bahrein, 2017, pp. doi: 10.1109/IEEEGCC.2017.8447959.
[A9]	X. Zhang, K. Ben e J. Zeng, "Cross-Entropy: A New Metric for Software Defect Prediction," *2018 IEEE International Conference on Software*

	Quality, Reliability and Security (QRS) , Lisboa, Portugal, 2018, pp. -122, doi: 10.1109/QRS.2018.00025.
[A10]	L. Kumar e A. Sureka, "Analisando a utilidade da previsão de falhas a partir da perspectiva de custo usando métricas de código-fonte", *2017 Décima Conferência Internacional sobre Computação Contemporânea (IC3)* , Noida, Índia, 2017, pp. .2017.8284297.
[A11]	Y. Zhao, Y. Hu e J. Gong, "Pesquisa sobre Padronização Internacional de Qualidade de Software e Teste de Software", *2021 IEEE/ACIS 20ª Conferência Internacional de Outono sobre Ciência da Computação e Informação (ICIS Fall)* , Xi'an, China, 2021 , pp.
[A12]	Kung, Gao, Hsia, Wen, Toyoshima e Chen, "Alterar identificação de impacto na manutenção de software orientada a objetos", *Proceedings 1994 International Conference on Software Maintenance* , Victoria, BC, Canadá, 1994, pp. .1994.336774.
[A13]	N. Wilde, P. Matthews e R. Huitt, "Manutenção de software orientado a objetos", em *IEEE Software* , vol. 10, não. 75-80, janeiro de 1993, doi: 10.1109/52.207232.
[A14]	M. Satpathy, NT Siebel e D. Rodriguez, "Asserções em manutenção de software orientada a objetos: análise e estudo de caso", *20ª Conferência Internacional IEEE sobre Manutenção de Software, 2004. Proceedings.* , Chicago, IL, EUA, 2004, pp.
[A15]	AK Malviya e VK Yadav, "Atividades de manutenção em sistemas de software orientados a objetos usando técnica de clustering K-means: Uma revisão," *2012 CSI Sexta Conferência Internacional sobre Engenharia de Software (CONSEG)* , Indore, Índia, 2012, pp. : 10.1109/CONSEG.2012.6349490.
[A16]	M. Dagpinar e JH Jahnke, "Prevendo a manutenibilidade com métricas orientadas a objetos - uma comparação empírica", *10ª Conferência de Trabalho sobre Engenharia Reversa, 2003. WCRE 2003. Proceedings.* , Victoria, BC, Canadá, 2003, pp. 155-164, doi: 10.1109/WCRE.2003.1287246.
[A17]	J. Pantos, A. Beszedes, P. Gyenizse e T. Gyimothy, "Experiences in Adapting a Source Code-Based Quality Assessment Technology," 2008 *12ª Conferência Europeia sobre Manutenção e Reengenharia de Software* , Atenas, Grécia, 2008, pp. 313, doi: 10.1109/CSMR.2008.4493335.
[A18]	GH Kencana, A. Saleh, HA Darwito, RR Rachmadi e EM Sari, "Comparação de medição do índice de manutenção do Microsoft CodeLens e linha de código", 2020 7ª Conferência *Internacional sobre Engenharia Elétrica, Ciências da Computação e Informática (EECSI)* , Yogyakarta, Indonésia , 2020, pp.
[A19]	KK Zaw, HW Hnin, KY Kyaw e N. Funabiki, "Software Quality Metrics Calculations for Java Programming Learning Assistant System", *2020 IEEE Conference on Computer Applications (ICCA)* , Yangon, Myanmar, 2020, pp. 10.1109/ICCA49400.2020.9022823.
[A20]	S. Prykhodko, N. Prykhodko and T. Smykodub, "A Joint Statistical Estimation of the RFC and CBO Metrics for Open-Source Applications

	Developed in Java," 2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, 2022, pp. 442-445, doi: 10.1109/CSIT56902.2022.10000457.
[A21]	T. Honglei, S. Wei e Z. Yanan, "The Research on Software Metrics and Software Complexity Metrics", *Fórum Internacional de Ciência da Computação, Tecnologia e Aplicações de 2009* , Chongqing, China, 2009, pp. /IFCSTA.2009.39.
[A22]	W. Li e S. Henry, "Métricas de manutenção para o paradigma orientado a objetos", *[1993] Proceedings First International Software Metrics Symposium* , Baltimore, MD, EUA, 1993, pp.
[A23]	H. Kabaili, RK Keller e F. Lustman, "Cohesion as changeability Indicator in Object-Oriented Systems," *Proceedings Fifth European Conference on Software Maintenance and Reengineering* , Lisboa, Portugal, 2001, pp. .2001.914966.
[A24]	ENH Kirğil e TE Ayyildiz, "Analysis of Lack of Cohesion in Methods (LCOM): A Case Study," 2021 *2ª Conferência Internacional de Informática e Engenharia de Software (IISEC)* , Ancara, Turquia, 2021, pp. /IISEC54230.2021.9672419.
[A25]	C. Catal, B. Diri and B. Ozumut, "An Artificial Immune System Approach for Fault Prediction in Object-Oriented Software," 2nd International Conference on Dependability of Computer Systems (DepCoS-RELCOMEX '07), Szklarska, Poland, 2007, pp. 238-245, doi: 10.1109/DEPCOS-RELCOMEX.2007.8.
[A26]	Y. Hassoun, R. Johnson e S. Counsell, "Uma métrica de acoplamento de tempo de execução dinâmico para arquiteturas de meta-nível", *Oitava Conferência Europeia sobre Manutenção e Reengenharia de Software, 2004. CSMR 2004. Procedimentos.* , Tampere, Finlândia, 2004, pp.
[A27]	S. Jain, V. Yadav and R. Singh, "A simplified formulation of predictive object points (POP) sizing metric for OO measurement," 2014 IEEE International Advance Computing Conference (IACC), Gurgaon, India, 2014, pp. 1367-1372, doi: 10.1109/IAdCC.2014.6779526.
[A28]	M. O. Elish and D. Rine, "Investigation of metrics for object-oriented design logical stability," Seventh European Conference onSoftware Maintenance and Reengineering, 2003. Proceedings., Benevento, Italy, 2003, pp. 193-200, doi: 10.1109/CSMR.2003.1192427.
[A29]	A. Kaur, K. Kaur and K. Pathak, "A proposed new model for maintainability index of open source software," Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization, Noida, India, 2014, pp. 1-6, doi: 10.1109/ICRITO.2014.7014758.