Revista ARACÊ
ISSN: 2358-2472

# TESTES DE SOFTWARE EM CAIXA DE SILÍCIO: UMA NOVA ABORDAGEM BASEADA EM INTELIGÊNCIA ARTIFICIAL

# SOFTWARE TESTING IN A SILICON BOX: A NEW APPROACH BASED ON ARTIFICIAL INTELLIGENCE

# PRUEBAS DE SOFTWARE EN UNA CAJA DE SILICIO: UN NUEVO ENFOQUE BASADO EN INTELIGENCIA ARTIFICIAL

https://doi.org/10.56238/arev7n6-343

**Data de submissão:** 30/05/2025 **Data de publicação:** 30/06/2025

**Lucas Moscatel de Andrade**
Graduando em Bacharelado em Engenharia de Software.
Universidade do Estado do Pará, UEPA
Lattes: http://lattes.cnpq.br/0823901215815720
E-mail:lucas.mandrade@aluno.uepa.br

**Hilbert Sobreira Fernandes**
Graduando em Bacharelado em Engenharia de Software.
Universidade do Estado do Pará, UEPA
Lattes: http://lattes.cnpq.br/8678669668435015
E-mail:lucas.mandrade@aluno.uepa.br

**Gustavo Cerpa Galvão**
Graduando em Bacharelado em Engenharia de Software.
Universidade do Estado do Pará, UEPA
Lattes: https://lattes.cnpq.br/0447079399979676
E-mail: gustavo.cgalvao@aluno.uepa.br

**Matheus Lima Sousa**
Graduando em Bacharelado em Engenharia de Software.
Universidade do Estado do Pará, UEPA
Lattes: https://lattes.cnpq.br/1950117890899539
E-mail: matheus.lsousa@aluno.uepa.br

**Marcus Mariano da Paz Pereira**
Graduando em Bacharelado em Engenharia de Software.
Universidade do Estado do Pará, UEPA
Lattes: http://lattes.cnpq.br/7492180200727733
E-mail: marcusmariano1806@gmail.com

**Maykon Lucas de Sousa Carneiro**
Graduando em Bacharelado em Engenharia de Software.
Universidade do Estado do Pará, UEPA
Lattes: https://lattes.cnpq.br/1254961025209532
E-mail: maykon.carneiro@aluno.uepa.br

Revista ARACÊ
ISSN: 2358-2472

**Milly Grahan Dias Melo**
Graduando em Bacharelado em Engenharia de Software.
Universidade do Estado do Pará, UEPA
E-mail:milly.gdmelo@aluno.uepa.br
Lattes: http://lattes.cnpq.br/6775314128117007

**Wilker José Caminha dos Santos**
Especialista em Engenharia de Sistemas.
Universidade do Estado do Pará, UEPA
Lattes: https://lattes.cnpq.br/3314938287386016
E-mail: wilkercaminha@uepa.br

**Thiago Nicolau Magalhães de Souza Conte**
Doutor em Engenharia Elétrica.
Universidade do Estado do Pará, UEPA
Lattes: http://lattes.cnpq.br/078337432552911
E-mail: thiagonconte@uepa.br

**RESUMO**
A crescente complexidade de sistemas de software e o volume massivo de dados impõem desafios significativos aos métodos de teste tradicionais, exigindo abordagens mais ágeis e eficientes. Este trabalho propõe a "Caixa de Silício", um novo sistema de teste de software baseado em inteligência artificial (IA), que visa superar as limitações das técnicas convencionais de teste (caixa-preta, branca e cinza) por meio de automação inteligente e adaptativa. A principal metodologia deste trabalho foi a revisão da literatura e fundamentação teórica, tendo como base estudos e métodos desenvolvidos anteriormente para fundamentar e fortalecer o método proposto por este artigo. O sistema funciona como um analisador de especificações de requisitos, gerador inteligente de casos de teste impulsionados pela IA para criação de cenários, um analisador de resultados com feedback em loop para aprendizado contínuo do sistema. As vantagens esperadas incluem a redução substancial do esforço manual,grande aumento da cobertura de teste, e a identificação antecipada de defeitos, promovendo assim melhoria contínua na alocação de recursos. A "Caixa de Silício" representa uma evolução prática nas estratégias de teste de software, unificando raciocínio computacional com execução autônoma.

**Palavras-chave:** Teste de software. Inteligência Artificial. Automação de testes. Caixa de Silício. Aprendizado de máquina.

**ABSTRACT**
The increasing complexity of software systems and the massive volume of data pose significant challenges to traditional testing methods, requiring more agile and efficient approaches. This paper proposes the "Silicon Box", a new software testing system based on artificial intelligence (AI), which aims to overcome the limitations of conventional testing techniques (black, white and gray box) through intelligent and adaptive automation. The main methodology of this work was the literature review and theoretical foundation, based on studies and methods previously developed to support and strengthen the method proposed by this paper. The system functions as a requirements specification analyzer, an intelligent AI-driven test case generator for scenario creation, and a results analyzer with looped feedback for continuous system learning. The expected advantages include a substantial reduction in manual effort, a large increase in test coverage, and early identification of defects, thus

Revista ARACÊ
ISSN: 2358-2472

promoting continuous improvement in resource allocation. The “Silicon Box” represents a practical evolution in software testing strategies, unifying computational reasoning with autonomous execution.

**Keywords:** Software testing. Artificial intelligence. Test automation. Silicon box. Machine learning.

**RESUMEN**
La creciente complejidad de los sistemas de software y el volumen masivo de datos plantean desafíos significativos a los métodos de prueba tradicionales, requiriendo enfoques más ágiles y eficientes. Este documento propone “Silicon Box”, un nuevo sistema de pruebas de software basado en inteligencia artificial (IA), que busca superar las limitaciones de las técnicas de prueba convencionales (caja negra, blanca y gris) mediante la automatización inteligente y adaptativa. La metodología principal de este trabajo fue la revisión de literatura y la fundamentación teórica, basada en estudios y métodos previamente desarrollados para respaldar y fortalecer el método propuesto en este documento. El sistema funciona como un analizador de especificaciones de requisitos, un generador inteligente de casos de prueba impulsado por IA para la creación de escenarios y un analizador de resultados con retroalimentación en bucle para el aprendizaje continuo del sistema. Las ventajas esperadas incluyen una reducción sustancial en el esfuerzo manual, un gran aumento en la cobertura de las pruebas y la identificación temprana de defectos, promoviendo así la mejora continua en la asignación de recursos. “Silicon Box” representa una evolución práctica en las estrategias de pruebas de software, unificando el razonamiento computacional con la ejecución autónoma.

**Palabras clave:** Pruebas de software. Inteligencia artificial. Automatización de pruebas. Silicon box. Aprendizaje automático.

Revista ARACÊ
ISSN: 2358-2472

## 1 INTRODUÇÃO

Os testes sempre estiveram presentes na vida dos cientistas de todas as áreas, com isso temos o exemplo na área elétrica que foi o inventor da lâmpada incandescente Thomas Edison que para prova um dos seus testes que utilizou a elefanta asiática chamada Topsy foi eletrocutada diante de 1.500 espectadores por Thomas Edison em Nova York em 4 de janeiro de 1903.

De acordo com Stallwood (2018) Foi com esse teste que marcou a história que Edison queria provar que a corrente contínua era mais segura do que a corrente alternada (a alternativa promovida por seu rival, George Westinghouse) e, assim, vencer a batalha pela eletrificação dos Estados Unidos. No entanto o Nikola Tesla cita um contraponto das formas de testes do Thomas Edison e conforme Wills (2019), no dia seguinte à morte de Thomas Edison, Nikola Tesla, que trabalhou para Edison entre 1882 e 1883, teria dito: "Seu método era extremamente ineficiente, pois era preciso percorrer um terreno imenso para se obter qualquer coisa, a menos que o acaso interviesse. A princípio, quase me senti uma testemunha triste de seus feitos, sabendo que um pouco de teoria e cálculo lhe teria poupado 90% do trabalho". O método que Tesla ridicularizou foi o de tentativa e erro, um método que se tornou tão intimamente associado a Edison que às vezes era chamado de método edisoniano.

Este método também é utilizado na área da tecnologia com ensino aos alunos na qual citado anteriormente que Nikola Tesla também presenciou esse mesmo modelo de método que segundo Edwards (2004), os alunos de ciência da computação introdutória dependem do método tentativa e erro, abordagem para corrigir erros e depurar por muito tempo. Mudar para uma estratégia de reflexão em ação pode ajudar os alunos a se tornarem mais bem-sucedidos.

Outro meio é o teste unitário que é uma das abordagens fundamentais para teste de software, com base em Neto e Claudio (2007), teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi desenvolvido. Nos termos de Papis et al. (2022), o teste unitário verifica a parte desejada da implementação diretamente, simplesmente executando-a. Entre muitas abordagens possíveis para teste unitário, duas técnicas principais de desenvolvimento são escolhidas para comparação: Desenvolvimento Orientado a Testes (TDD) e Desenvolvimento Teste-Último (TLD).

Outras formas de testes de software são manuais, automatizados, unidade, integração, funcional, aceitação, entre outros. Segundo o Linkages (2023), os métodos tradicionais utilizados para testar um software, apesar do esforço para se adaptarem e evoluírem de acordo com o crescente avanço da tecnologia atual e dos tempos modernos, estão tendo grandes dificuldades nesse processo de adaptação. Muito se deve ao constante crescimento de grandes quantidades e volume de dados que são gerados em uma base de dados.

Revista ARACÊ
ISSN: 2358-2472

Com base no relatório chamado “Data Age 2025”, cada pessoa gerou cerca de 1,3 GB de dados por dia (IDC, 2015). A luz de Linkages (2023), diariamente, o mundo gera cerca de 2,5 quintilhões de dados. Já a IBM (International Business Machines Corporation) afirma que 90% dos dados disponíveis hoje foram gerados nos últimos 3 anos. Por outro lado, a IDC ainda complementa: espera-se que a quantidade de dados gerados no mundo alcance 175 zettabytes até 2025, um aumento significativo em relação aos 33 zettabytes gerados em 2018, um desafio para manipular estruturas abstratas.

Dessa forma, segundo a reflexão de Camacho (2024), um dos desafios presente nos testes de softwares, é a complexibilidade de sistemas modernos, que exigem ferramentas e abordagens avançadas para cobrir todas as possíveis falhas. Aponta outro desafio, que é a pressão por prazos que significa como os ambientes de desenvolvimento rápido, como metodologias Ágil, exige uma constante pressão para o prazo de entrega, o que pode prejudicar a qualidade dos testes.

Mais um desafio mencionado por Camacho (2024), é o desafio da cobertura de testes, que consiste em assegurar que todas as características do software sejam testadas, principalmente em grandes sistemas integrados. Outro desafio é a manutenção de scripts de teste, que envolve o mantimento de scripts de teste modernos de acordo com a evolução do software, principalmente em contextos ágeis com alterações recorrentes.

Com esse cenário de avanço e agilidade conforme o parágrafo anterior veio para contribuir no teste de software e um dos auxilio para isso é a inteligência artificial (IA) de acordo com o exposto por Liu (2024), a IA se destaca no processamento de grandes conjuntos de dados e na execução de tarefas especializadas com velocidade e precisão, ela carece da compreensão detalhada, da inteligência emocional e do raciocínio adaptativo característicos da cognição humana.Diante desse cenário o avanço da inteligência artificial (IA), ela tem se tornado cada vez mais presente no dia a dia das pessoas, oferecendo várias possibilidades, como assistentes virtuais, plataformas de aprendizado feitas sob medida, tutoria personalizada, além de ferramentas de tradução e de verificação de plágio (Josué et al. 2023).

Os modelos de IA operam em arquiteturas estáticas, nas quais as conexões são pré-determinadas e imutáveis após o treinamento, sem a adaptabilidade dinâmica do cérebro. O aprendizado em IA é alcançado por meio de técnicas de otimização matemática, como a retropropagação, que adapta o modelo aos dados em vez de evoluir com novas experiências (Liu, 2024). Portanto no campo da engenharia de software, modelos de linguagem de última geração, como Claude Opus 4, da Anthropic, têm ampliado as possibilidades de automação inteligente, destacando se por alcançar 72,5% de acurácia no benchmark que avalia a capacidade de sistemas de IA em

Revista ARACÊ
ISSN: 2358-2472

compreender e corrigir erros reais em projetos de software hospedados no GitHub, com base em problemas documentados e patches validados por testes da SWE-bench Verified, demonstrando superioridade em comparação ao GPT-4.1 em tarefas de compreensão, correção e geração de códigos (ITPro, 2025).

Na maioria dos casos, de acordo com ITPro (2025), o Opus 4 foi capaz de fornecer soluções funcionais para os erros, sem necessidades de intervenção humana, demonstrando um nível de precisão e entendimento técnico comparável ao de desenvolvedores experientes. Com capacidade de manter operações de complexidade por até sete horas contínuas, o Opus 4 apresenta grande potencial para ser integrado a ambiente de teste automatizado baseado em nuvem, como as propostas por soluções tipo Cloud Opus 4.

Essas características tornam viável a automação inteligente de tarefas como geração e reparo de casos de teste, adaptação dinâmica de scripts e análise de falhas em tempo real. Complementando essa perspectiva, o mapeamento sistemático realizado por Dias (2023) demonstra que técnica como algoritmos genéticos, redes neurais e processamento de linguagem neural já vêm sendo amplamente aplicadas no contexto de testes de software, reforçando que modelos avançados de IA, como o Opus 4, estão alinhados às necessidades e tendências emergentes dessa área.

Com isto, este artigo tem como objetivo apresentar e discutir a proposta da "Caixa de Silício", um novo modelo de testes de software baseado em inteligência artificial. A abordagem visa superar as limitações das técnicas tradicionais, como os testes de caixa-preta, branca e cinza, por meio da automação inteligente, adaptativa e autônoma. A pesquisa também analisa o estado da arte das ferramentas de teste assistidas por IA destacando os avanços recentes e as lacunas ainda existentes. Além disso, são discutidos os principais desafios técnicos e éticos relacionados à transparência, segurança e confiabilidade dos sistemas inteligentes de teste, propondo diretrizes práticas para a adoção da Caixa de Silício em diferentes contextos de desenvolvimento (Garousi; Felderer; Mantyla, 2024).

Sendo a principal contribuição do modelo está na introdução de um sistema cognitivo de teste automatizado, capaz de gerar casos de teste inteligentes, monitorar resultados em tempo real e ajustar-se continuamente com base em dados anteriores. A Caixa de Silício combina aprendizado de máquina, análise semântica e priorização automática de testes críticos, promovendo maior cobertura, redução de custos e melhoria da qualidade. Diferente das abordagens convencionais, o modelo propõe autonomia com governança, aplicabilidade e rastreabilidade das decisões tomadas pela IA. Assim, a Caixa de Silício representa uma evolução conceitual e prática nas estratégias de teste de software,

Revista ARACÊ
ISSN: 2358-2472

unificando raciocínio computacional com execução automatizada em um ambiente seguro, transparente e eficiente (Wang; Guo; Tan, 2025).

## 2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

Devido à crescente relevância das metodologias de IA para sistemas e softwares vigentes e futuros, existe uma necessidade de desenvolver medidas apropriadas para a garantia de qualidade. Contudo estas medidas precisam vir acompanhadas de algumas garantias de que os produtos desenvolvidos atendem aos seus respectivos requisitos, como por exemplo, se existem precauções com a segurança e fornecem a funcionalidade requisitada (Wotawa, 2021).

A Inteligência Artificial (IA) tem se destacado e sendo cada vez mais adotada em vários setores, promovendo mudanças importantes nos programas utilizados pelas empresas. Segundo o estudo de Kokol (2024), a inteligência artificial moderna faz uso de técnicas avançadas, como o Machine Learning (ML), para criar novos conhecimentos, gerar conteúdo, formular hipóteses e até propor ideias originais. Isso é possível graças à sua capacidade de detectar padrões e extrair informações de grandes volumes de dados. Como resultado, a IA passou a ser aplicada em uma variedade crescente de áreas, incluindo o design e desenvolvimento de software.

Os modelos de IA que se encontram em destaque no cenário atual são ChatGPT, Gemini, DeepSeek e Cloud Opus, segundo a Anthropic (2025), o Cloud Opus 4 é o modelo mais poderoso, atualmente é o melhor modelo para codificação do mundo, liderando no SWE-bench (72,5%) e no Terminal-bench (43,2%).O Claude Opus 4 destaca-se na codificação e resolução de problemas complexos, impulsionando produtos de agentes de fronteira. A Cognition IA observa que a Opus 4 se destaca na resolução de desafios complexos no qual outros modelos não conseguem, lidando com êxito com ações críticas que modelos anteriores não conseguiram. Na qual foi feita uma análise comparativa dos modelos de IA incluindo Claude Opus 4, Claude Soneto 4, Claude Soneto 3.7, OpenAI o3, OpenAI GPT-4.1 e Gêmeos 2.5 Pro (Visualização de 05-06), revela desempenhos variados em diversas categorias e benchmarks.

No que diz respeito à Codificação Agêntica (Banco SWE Verificado 5), o OpenAI o3 se destacou significativamente, alcançando o melhor resultado com 88,10%, enquanto o OpenAI GPT-4.1 apresentou um desempenho consideravelmente inferior, com 54,80%, evidenciando uma grande diferença de 33,30 pontos percentuais entre os modelos da OpenAI nesta categoria. Para a Codificação de Terminal Agêntico (Banco-terminal-2.5), Claude Opus 4 emergiu como o líder, com 43,2%, superando o Gêmeos 2.5 Pro, que obteve o pior resultado com 25,30%, indicando uma vantagem de 17,9 pontos percentuais para o Claude Opus 4.

Revista ARACÊ
ISSN: 2358-2472

Na área de Raciocínio em Nível de Pós-graduação (GPQA - diamante 5), o OpenAI GPT-4.1 demonstrou um desempenho superior, com 86,30%, mantendo uma pequena vantagem de 3,30 pontos percentuais sobre o Gêmeos 2.5 Pro, que registrou 83,00%. Quando se trata do Uso de Ferramentas em Agências (TAU-banco), o Gêmeos 2.5 Pro liderou na categoria de varejo com 88,0%, mas é importante notar que as análises para varejo e linha aérea foram feitas separadamente, sendo que na categoria de linha aérea, o OpenAI GPT-4.1 teve o pior resultado, com 49,4%.

No campo das Perguntas e Respostas Multilingues (MMLU13), houve um desempenho muito similar e de alto nível entre Gêmeos 2.5 Pro e os modelos OpenAI (GPT-4.1 e o3), com os resultados variando entre 88,80% e 88,70%, mostrando uma competitividade acirrada conforme mostrado na Figura 1. Para o Raciocínio Visual (MMMU - validação), o OpenAI o3 obteve o melhor resultado com 82,90%, superando o Gêmeos 2.5 Pro que alcançou 79,60%, uma diferença de 3,30 pontos percentuais. Finalmente, na Competição de Matemática do Ensino Médio (AIME 2024.5), o OpenAI o3 novamente se destacou com o melhor resultado, 88,80%, enquanto o Gêmeos 2.5 Pro ficou um pouco atrás, com 83,00%, resultando em uma diferença de 5,80 pontos percentuais a favor do OpenAI o3.

**Figura 1 -** Estrutura comportamental dos modelos OpenAI e Claud

| | **Claude** Opus 4 | **Claude** Soneto 4 | **Claude** Soneto 3.7 | **OpenAI o3** | **OpenAI GPT-4.1** | **Gêmeos 2.5 Pro** Visualização (05-06) |
|---|---|---|---|---|---|---|
| **Codificação agêntica** *Banco SWE Verificado1, 5* | 72.5% / 79.4% | 72.7% / 80.2% | 62.3% / 70.3% | 69,10% | 54,60% | 63,20% |
| **Codificação de terminal agêntico** *Banco-terminal2, 5* | 43.2% / 50.0% | 35.5% / 41.3% | 35,20% | 30,20% | 30,30% | 25,30% |
| **Raciocínio em nível de pós-graduação** *GPQADiamante 5* | 79.6% / 83.3% | 75.4% / 83.8% | 78,20% | 83,30% | 66,30% | 83,00% |
| **Uso de ferramentas agenciais** *TAU-banco* | Varejo 81.4% Linha aérea 59.6% | Varejo 80.5% Linha aérea 60.0% | Varejo 81.2% Linha aérea 58.4% | Varejo 70.4% Linha aérea 52.0% | Varejo 68.0% Linha aérea 49.4% | – – |
| **Perguntas e respostas multilingues** *MMMLU3* | 88,80% | 86,50% | 85,90% | 88,80% | 83,70% | – |
| **Raciocínio visual** *MMMU(validação)* | 76,50% | 74,40% | 75,00% | 82,90% | 74,80% | 79,60% |
| **Competição de matemática do ensino médio** *AIME 20254, 5* | 75.5% / 90.0% | 70.5% / 85.0% | 54,80% | 88,90% | – | 83,00% |

**Fonte:** Anthropic, 2025.

Test.ai é destacado por Ramadan, Yasin e Pektas (2024), como uma ferramenta que utiliza inteligência artificial para automatizar testes funcionais. Os autores afirmam que a IA "automatiza geração de casos de teste, execução e análise de resultados" e destacam ferramentas como Test.ai entre

Revista ARACÊ
ISSN: 2358-2472

as que aplicam essas técnicas na prática. Essa abordagem assistiva facilita o reconhecimento de elementos de interfaces e a adaptação automática de scripts quando a interface muda, reduzindo o esforço de manutenção, embora ainda exige supervisão humana.

Diffblue Cover, por sua vez, utiliza aprendizado por reforço para gerar automaticamente testes de unidade em Java, funcionando de forma autônoma. Como evidenciado por Khaliq, Farooq e Khan (2022), “a comunidade de testes está adotando IA para preencher a lacuna, visto que pode verificar o código para bugs sem intervenção humana, de forma muito mais rápida”. Diferente de ferramentas assistivas, Diffblue Cover cria suítes completas, atualiza continuamente os testes conforme o código evolui e se integra a processos de CI/CD, mostrando ganhos significativos em cobertura e eficiência.

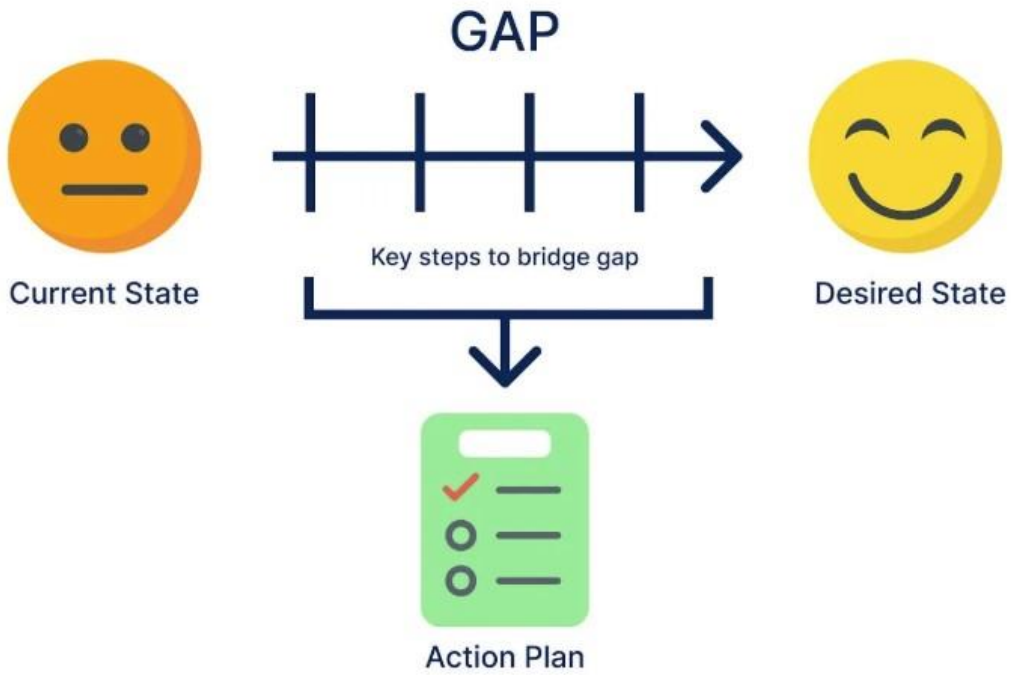
Um modelo de linguagem avançado da Anthropic que tem se destacado em benchmarks de engenharia de software é o Claude Opus 4, pois ele vem demonstrando capacidades superiores na execução de tarefas de codificação e teste. No trabalho desenvolvido por López Martínez, Cano e Ruiz-Martínez (2025), destacam que Claude Opus apresentou desempenho superior em testes de penetração assistida por IA superando concorrentes de peso no mercado, como o ChatGPT-4 e Copilot, fornecendo suporte substancial, e exibindo maior eficiência em tarefas específicas. Este resultado evidencia o potencial do Opus 4, não apenas como ferramenta de geração de código, mas também como agente inteligente apto a otimizar e automatizar processos críticos na qualidade de um software.

Por fim, comparando essas soluções com a proposta da Caixa de Silício, observamos que a Test.ai oferece abordagem assistiva focada em UI, enquanto Diffblue traz autonomia para testes unitários específicos em Java. Já opus 4 expande o domínio para compreensão e correção de código em larga escala. A Caixa de Silício propõe um modelo holístico que une autonomia, aprendizado contínuo, governança e aplicabilidade, integrando múltiplos tipos de teste — unidade, integração e funcional em uma estrutura adaptativa e rastreável, superando limitações pontuais das soluções atuais.

A maioria das empresas que atuam no desenvolvimento de softwares tem grandes preocupações em testar os produtos de softwares de forma correta e precisa, evitando assim problemas futuros que um lançamento com erros pode ocasionar (QOBO, 2023). Essa preocupação levou a essas mesmas empresas a utilizarem o Gap Analysis (análise de lacuna),na Figura 2 mostra essa que é uma abordagem metódica que as organizações utilizam para identificar as diferenças entre o estado atual de um processo de teste e o desempenho esperado. Essa análise ajuda a identificar áreas que precisam de melhorias e as etapas necessárias para preencher as lacunas (Andrade, 2024).

Revista ARACÊ
ISSN: 2358-2472

**Figura 2** - Estrutura do Gap Analysis (análise de lacuna)




**Fonte:** Price Pal, Gap Analysis, 2022.

Esse método traz consigo algumas vantagens quando utilizado nas empresas, por exemplo: permite identificar código crítico não testado, que segundo o estudo realizado por Eder et. al (2013), quase 80% dos bugs podem ser rastreados, até mesmo código não testado, a análise de lacunas ajuda a esclarecer seções do código que os testadores não cobriram em nenhum conjunto de testes. Esse conhecimento pode ajudar testadores e desenvolvedores a alinharem fluxos de trabalho para configurar casos de teste para partes de código não testadas (Avon Automation, 2025).

Outro aspecto relevante é a identificação de código obsoleto no decorrer do processo de teste de software. Em contextos em que o sistema passa por frequentes atualizações, é comum que segmentos do código se tornem desatualizados, especialmente funções que não são mais utilizadas. A análise de lacunas de teste é instrumental para identificar essas seções, facilitando sua exclusão e dos conjuntos de testes. Dessa maneira, promove-se uma melhor alocação de recursos para áreas estratégicas e críticas do negócio, aumentando a eficiência do processo de desenvolvimento e manutenção.

Adicionalmente, destaca-se a importância da alocação adequada de recursos e força de trabalho. Os dados obtidos por meio da análise de lacunas de teste subsidiam os gestores na otimização da utilização de recursos e das ferramentas disponíveis. A partir da identificação dessas lacunas, torna-se possível direcionar esforços especificamente para módulos de código que demandam testes mais rigorosos, ao invés de investir recursos em segmentos já suficientemente verificados ou de baixo

Revista ARACÊ
ISSN: 2358-2472

impacto para o sistema. Assim, os gestores conseguem planejar e monitorar as etapas de testes de forma mais assertiva, fortalecendo as iniciativas voltadas para a qualidade do software. A aplicação eficiente dessa abordagem envolve práticas e métodos específicos, que contribuem para a precisão e eficácia da análise de lacunas de teste.

## 2.1 IMPLEMENTAÇÃO DA INTEGRAÇÃO DO SOFTWARE

A análise de lacunas fundamenta-se na combinação de técnicas de análise estática e dinâmica, objetivando identificar as áreas do software que foram ou não abrangidas pelos testes. Esse processo demanda a coleta de informações detalhadas, inclusive a comparação das modificações recentes com o sistema de controle de versão. Além disso, torna-se indispensável o uso de ferramentas de perfilamento (profilers), que são amplamente empregadas em engenharia de desempenho para registrar informações sobre múltiplos parâmetros de execução, como a simulação de conjuntos de instruções. De forma geral, testes manuais são realizados em ambientes controlados de teste, acoplados a ferramentas de perfilamento, que geram relatórios de cobertura posteriormente integrados a sistemas de rastreamento. (Avon Automation, 2025).

## 2.2 INICIAÇÃO EM ESCALA REDUZIDA, COM POSTERIOR ESCALONAMENTO PARA CONTEXTOS DE MAIOR COMPLEXIDADE

Para a implementação da análise de lacunas de teste, recomenda-se iniciar com configurações mais simples durante as fases iniciais de integração, progredindo gradualmente para cenários de maior complexidade. Este processo incremental consolida a confiança dos usuários no sistema e facilita a preparação das equipes para procedimentos mais sofisticados. Nos testes manuais, é essencial delimitar os pontos de coleta de informação de cobertura – por exemplo, em aplicativos servidor, a coleta restringe-se ao servidor; já em aplicações fat client, é necessário abranger todos os clientes, garantindo representatividade nos dados. Wotawa (2021)

## 2.3 OTIMIZAÇÃO E SIMPLIFICAÇÃO DO PROCESSO DE TESTAGEM

Considerando que o propósito central da análise de lacunas no teste de software consiste em proporcionar uma visão precisa e abrangente do estado do sistema testado, a divisão do processo em etapas bem estruturadas favorece sua simplificação. Tal abordagem contribui para o planejamento, execução e monitoramento dos testes de forma sistematizada. Conforme mencionado, recomenda-se que as primeiras integrações ocorram exclusivamente em ambientes de teste controlados. Após validação da estratégia, torna-se viável sua extensão a diferentes ambientes de teste. Realizar a análise

Revista ARACÊ
ISSN: 2358-2472

de forma incremental, partindo de cenários simples até configurações mais elaboradas, minimiza riscos de falhas e potencializa o sucesso na cobertura do sistema. Ainda com base em Wotawa (2021), a validação e verificação de sistemas e softwares é a parte fundamental do ciclo de desenvolvimento para auxiliar aos requisitos fornecidos.

## 3 COMPARAÇÃO DO MODELO SILÍCIO COM MÉTODOS TRADICIONAIS

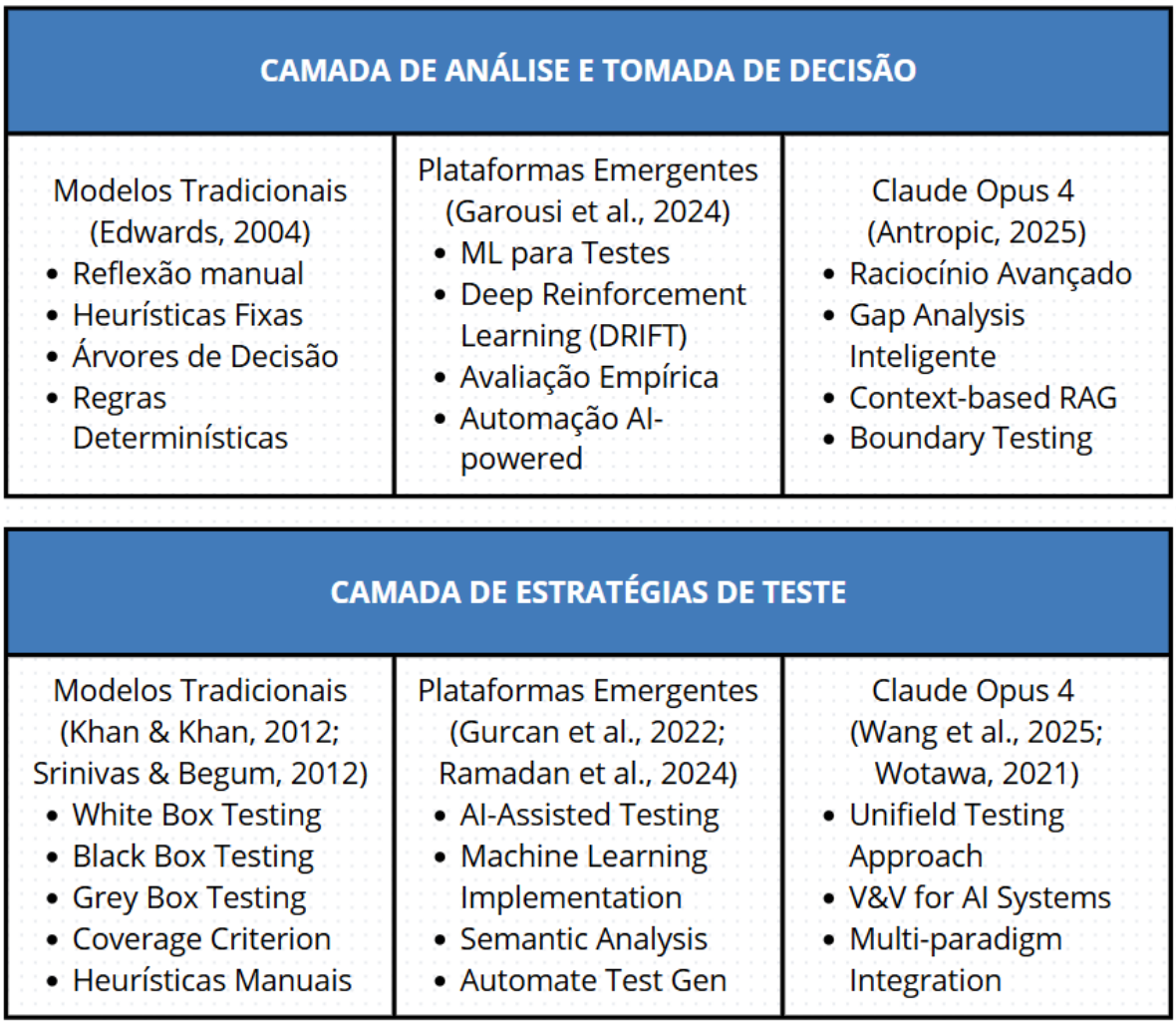
A Caixa de Silício é uma proposta que tem como objetivo superar as limitações das abordagens tradicionais, por meio de testes automatizados baseados em Inteligência Artificial, tendo como principais diferenças comparando aos testes anteriormente mencionados, a autonomia e a adaptabilidade. Como muito bem evidenciado por Bakar, Normi Sham Awang Abu (2025), em seu texto eles explicitam a dependência que os testes tradicionais possuem na análise humana, pois eles são estáticos e dependem de criação manual ou semi-manual, já a caixa de silício e proativa e generativa, utiliza a capacidade da IA de aprendizado e evoluir em cada execução, para assim estar otimizando os testes e adaptando suas estratégias a cada mudança no software.

Em contraste ao método anterior, o Teste de Caixa-Branca, exige conhecimento aprofundado da lógica interna, do código fonte e da arquitetura do software. É realizada uma análise da estrutura do código, os caminhos de execução, as condições lógicas e as operações para garantir que todas as partes do software funcionem corretamente, sendo este a parte mais crucial para identificar erros de implementação e falhas lógicas, segundo o trabalho de Ostrand Thomas (2002), Este método é muito utilizado em testes unitários de integração para verificar a cobertura do código e a integridade estrutural.

Teste de Caixa-Cinza representa uma abordagem híbrida, combinando elementos dos dois testes anteriores, Caixa-Preta e Caixa-Branca, onde o testador possui conhecimento parcial da estrutura interna do software, mas não possui o acesso total ao código fonte. Como descrito no trabalho de Khan, Mohd Ehmer; Khan, Farmeena (2012), essa ação intermediária permite a criação de casos de testes mais direcionados, otimizando a detecção de defeitos que afetam a performance ou a integração entre módulos, ao mesmo tempo que simula o comportamento do usuário de forma mais informal e na Figura 3 mostra a diferença da arquitetura da caixa de silício para Caixa-cinza.

Revista ARACÊ
ISSN: 2358-2472

**Figura 3** - Arquitetura Comparativa para aplicação da proposta do modelo Caixa de silício

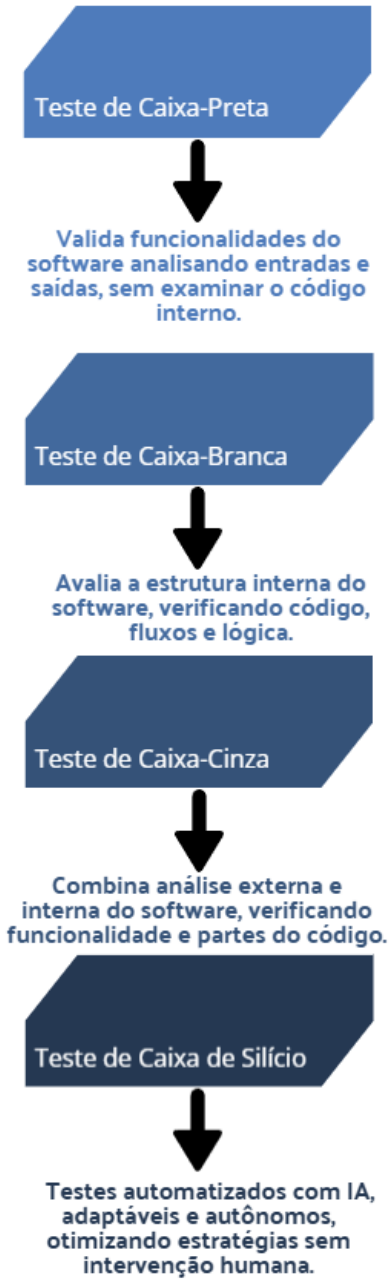



**Fonte:** Autores, 2025

O teste de caixa-preta concentra-se na validação das funcionalidades do software, analisando as entradas e saídas do sistema sem se aprofundar no código interno, com foco no comportamento observado pelo usuário. Por fluxos de programação para garantir o correto funcionamento do código. Já o teste de caixa-cinza adota uma abordagem híbrida, combinando a análise externa das funcionalidades com inspeção de partes do código, proporcionando uma validação mais ampla do sistema. Recentemente, o teste de caixa de silício tem se destacado por sua natureza automatizada e pelo uso de inteligência artificial, permitindo a realização de testes inteligentes, evolutivos e autônomos, que otimizam continuamente as estratégias e contribuem para maior qualidade, confiabilidade e eficiência no desenvolvimento de software que na Figura 4 mostra a cascata e a metodologia de teste de software e a diferença de cada Caixa.

Revista ARACÊ
ISSN: 2358-2472

**Figura 4** - Diagrama de cascata metodologia tradicionais teste de software




**Fonte:** Autores, 2025.

## 4 VISÃO GERAL DO MÉTODO PROPOSTO

O método de análise proposto por esse estudo, chamado de Caixa de Silício, representa uma nova abordagem de teste de software, com o objetivo de superar métodos tradicionais, como a caixa preta, branca e cinza. Sua premissa central é utilizar a IA para a automação inteligente e adaptativa essencial para sistemas modernos de alta complexidade, como descrito no trabalho de Camacho (2024), os sistemas atuais geram muito volume de dados que demandam agilidade de processamento. O objetivo principal é superar as restrições dos métodos tradicionais, que, segundo estudos, se

Revista ARACÊ
ISSN: 2358-2472

deparam com desafios cada vez maiores devido à complexidade dos sistemas, ao volume de dados e à necessidade de agilidade no ciclo de desenvolvimento (Wang, J., & Zhang, L., 2021).

## 4.1 PREMISSAS E CONTEXTO DE USO (SISTEMAS WEB, MICROSERVIÇOS)

O modelo da Caixa de Silício baseia-se na ideia de usar a Inteligência Artificial de forma estratégica para proporcionar adaptabilidade e autonomia aos testes de software de automação inteligente. Essa metodologia se distingue significativamente das práticas convencionais, que historicamente exigem uma intervenção humana considerável para o planejamento, execução e manutenção de testes. Como resultado, são criadas suítes estáticas que rapidamente se tornam inflexíveis e onerosas em contextos de desenvolvimento acelerado, um aspecto fundamental nos debates sobre os progressos e obstáculos do setor (Engler, B., et al., 2021).

Essa habilidade de aprender e evoluir, geralmente apoiada em métodos como aprendizado por reforço ou redes neurais generativas para a criação de artefatos de teste, é considerada essencial para superar as restrições dos testes baseados em scripts fixos e possibilitar que o sistema de teste se auto aperfeiçoe (Pan, Y., et al., 2022). Aqui estão alguns exemplos de aplicação em casos de uso dessa nova abordagem tal como os Sistemas Web com interfaces em constante evolução, fluxos de usuário complexos e a exigência de verificar a compatibilidade em diferentes navegadores e aparelhos, micro-serviços: arquiteturas distribuídas que elevam a complexidade da integração e demandam testes completos e eficazes das interações entre os serviços, ambientes com forte pressão por prazos: onde se utilizam metodologias ágeis e a rapidez na identificação e resolução de falhas é fundamental para o êxito das entregas. Outro fator que torna a automação e a adaptabilidade baseadas em IA não só vantajosas, mas indispensáveis, é o grande volume de dados produzido por sistemas modernos, conforme evidenciado anteriormente com as projeções da IDC e IBM.

## 4.2 ANALISADOR DE ESPECIFICAÇÕES

Este componente inicial funciona como o "cérebro" para entender os requisitos. Sua tarefa é absorver e analisar as definições do software (requisitos funcionais e não funcionais, casos de uso, documentação, etc.) para extrair dados fundamentais, identificar fluxos críticos e entender o comportamento previsto do sistema. É com base nessa análise que se estabelecem os fundamentos para a criação de testes.

Ao compreender de forma abrangente as funcionalidades e os comportamentos esperados do sistema, o Analisador de Especificações oferece o suporte necessário para a criação eficiente de casos de teste e para a verificação dos resultados (Peixoto, R. J. S., et al., 2018).

Revista ARACÊ
ISSN: 2358-2472

### 4.3 GERADOR INTELIGENTE DE CASOS DE TESTE

Com base em um modelo de IA sofisticado (como o Claude Opus 4, cujas habilidades de compreensão e geração de código foram analisadas anteriormente), este componente tem a função de gerar automaticamente cenários de teste e dados pertinentes. A inteligência está na habilidade de ir além da simples produção baseada em regras estabelecidas; o gerador é capaz de deduzir cenários de teste mais complexos e diversos, com o objetivo de diminuir significativamente a necessidade de elaboração manual de testes. O objetivo é otimizar a cobertura de maneira inteligente. Ele trabalha para diminuir significativamente a necessidade de intervenção humana na elaboração de testes, que geralmente é um processo longo e sujeito a erros (Bastos & Almeida, 2023).

### 4.4 PRIORIZADOR ADAPTATIVO

A criticidade e a probabilidade de falha não são iguais para todos os testes. Ele emprega algoritmos de classificação de risco (fundamentados, por exemplo, em registros de falhas, complexidade do código relacionado ou regularidade de uso de uma funcionalidade) para estabelecer a sequência na qual os testes serão realizados. O aspecto "adaptativo" indica que essa priorização se ajusta de forma dinâmica com base nos resultados de execuções passadas, assegurando que os testes mais relevantes ou com maior probabilidade de detectar defeitos sejam realizados primeiro.

Os critérios para essa priorização podem abranger o histórico de falhas em testes específicos, a complexidade do código relacionado aos testes, o impacto de alterações recentes no código ou a importância funcional das áreas do sistema abrangidas pelos testes. O objetivo é executar primeiro os testes mais eficazes na identificação de riscos (Mahajan, S., et al., 2021).

Este elemento atua como a conexão entre a inteligência da Caixa de Silício e os ambientes de desenvolvimento e teste. Ele se conecta de forma nativa a ferramentas de Integração Contínua/Entrega Contínua (CI/CD), como Jenkins, GitLab CI ou Azure DevOps, com o objetivo de automatizar a realização dos testes criados e priorizados. Administra a paralelização dos testes e o isolamento dos ambientes, assegurando eficácia e confiabilidade na execução em grande escala. A integração em pipelines de CI/CD é um aspecto fundamental na busca por entregas rápidas de alta qualidade (Rahman, A., *et al*., 2021).

### 4.5 ANALISADOR DE RESULTADOS E FEEDBACK LOOP

Depois de executado, o Analisador de Resultados analisa as saídas, compara-as com o que era esperado e aponta erros. Coletam-se métricas significativas, como taxa de identificação de falhas, abrangência de código e tempo médio para detecção. O essencial "Feedback Loop" (ciclo de

Revista ARACÊ
ISSN: 2358-2472

realimentação) emprega esses dados para reprocessar o modelo de IA possibilitando que ele aprenda de forma contínua, aperfeiçoe suas heurísticas de geração e priorização, e se adapte para execuções futuras. É esse ciclo que dá ao modelo sua habilidade de evolução.

A implementação do modelo de testes em Caixa de Silício oferece uma série de benefícios importantes que abordam diretamente os desafios citados na introdução e que os métodos convencionais têm dificuldade em superar:

Ao automatizar a criação, priorização e realização dos testes, o modelo reduz consideravelmente o esforço manual das equipes de teste e desenvolvimento. Isso permite que os profissionais se dediquem a tarefas mais importantes, como a análise exploratória, o desenvolvimento de testes complexos em áreas de alto risco ou o aprimoramento da estratégia geral de qualidade (Camacho, 2024).

A habilidade do gerador inteligente de investigar uma variedade maior e mais diversificada de cenários, aliada à priorização adaptativa, tende a expandir consideravelmente a cobertura de testes em relação a métodos manuais ou automatizados estáticos. A IA é capaz de reconhecer combinações e trajetórias de execução que seriam complexas ou demoradas para um testador humano antecipar. A revisão sistemática conduzida por Garousi, Felderer e Mäntylä (2024), sobre ferramentas de automação de teste baseadas em IA evidencia, sem dúvida, de que maneira essas ferramentas impactam as métricas de cobertura.

## 4.6 IDENTIFICAÇÃO PRECOCE E MAIS EFICIENTE DE DEFEITOS

A Caixa de Silício possibilita a identificação de falhas mais cedo no ciclo de desenvolvimento por meio de testes mais completos e uma execução automatizada e contínua (integrada ao CI/CD). Isso diminui o custo e o efeito da correção, que geralmente aumentam significativamente quanto mais tarde um bug é identificado. A rapidez na detecção de bugs, sem exigir intervenção humana contínua, é um dos benefícios da IA no teste, conforme discutido por (Khaliq, Farooq e Khan53, 2022).

O ciclo de feedback e aprendizado possibilita que o sistema se desenvolva ao longo do tempo, tornando-se mais eficiente na detecção de padrões de falha e na adaptação a alterações no software. Como resultado, temos uma base de código mais sólida e confiável. A própria progressão dos modelos de IA para arquiteturas mais "agênticas", que exibem maior independência e habilidade para raciocínio complexo em tarefas de código, conforme enfatizado no site da Keysight Technologies (2025), fortalece a possibilidade de sistemas de teste que aprendem e se desenvolvem.

A priorização baseada em risco orienta os esforços de teste para as áreas mais críticas, maximizando a utilização dos recursos humanos e computacionais. A Análise de Lacuna (Gap

Revista ARACÊ
ISSN: 2358-2472

Analysis), frequentemente impulsionada pela automação e análise de dados, já evidencia como a identificação de áreas não testadas ou a eliminação de testes obsoletos (Andrade, 2024; Avo Automation, 2025; Qobo, 2023) pode resultar em uma distribuição de esforços mais eficiente. Por meio do Priorizador Adaptativo, a Caixa de Silício incorpora essa inteligência de alocação em seu funcionamento central.

O Método Caixa de Silício é um sistema cognitivo de teste composto por elementos integrados, o Analisador de Especificações extrai requisitos, enquanto o Gerador Inteligente de Casos de Teste cria cenários automaticamente, com a intenção de reduzir bruscamente a intervenção humana. O Priorizador Adaptativo otimiza a ordem de execução dos testes com base em risco e histórico, e o Orquestrador de Execução integra-se aos ambientes CI/CD necessários. Junto a eles o Analisador de Resultados e Feedback Loop monitora e realimenta o modelo de IA, para que o ajuste e o aprendizado sejam contínuos. (Y wang; S guo; CW tan, 2025).

Ainda convém lembrar que a seguir a figura 5 mostra o diagrama que representa um processo contínuo e automatizado de testes de interações em IA. Esse fluxo envolve etapas de automação, coleta de feedbacks e integração, começando com a entrada dos requisitos e indo até o monitoramento dos resultados. Ele funciona de forma circular, aprendendo constantemente partindo dos feedbacks recebidos durante as execuções, aprimorando-se cada vez mais.

**Figura 5** - Diagrama sobre processo contínuo e automatizado de testes em IA


**Fonte:** Autores, 2025

Revista ARACÊ
ISSN: 2358-2472

## 5 DETALHAMENTO DAS ETAPAS METODOLÓGICAS

A efetividade do modelo "Caixa de Silício" está em sua metodologia claramente estruturada, que incorpora a inteligência artificial em pontos cruciais do ciclo de vida do teste de software. De acordo com o seu documento "Testes de Software em Caixa de Silício", o uso da IA vai além da simples execução; ele abrange o planejamento, a geração, a priorização e, principalmente, o aprendizado contínuo. Ao tratar das etapas de "Planejamento de Teste Automatizado" e "Ciclo de Feedback e Aprendizado", nota-se o quão profundamente a inteligência artificial está integrada à otimização e flexibilidade do processo. (Kkesgin e Amasyali., 2023).

O planejamento é a primeira fase e estratégica que estabelece a direção e o escopo dos esforços de teste. É um alicerce para a eficácia de qualquer metodologia de teste, especialmente em um contexto em que a automação e a IA têm papéis centrais. Um planejamento rigoroso garante a alocação otimizada de recursos e a definição clara e alcançável de objetivos de qualidade.

O primeiro passo para um planejamento eficiente é estabelecer objetivos claros. Em um sistema complexo como o sugerido pela "Caixa de Silício", é fundamental ir além da mera identificação de "bugs". Os objetivos devem abranger os diversos aspectos da qualidade do software. Cobertura: estabelecer a cobertura desejada vai além da cobertura de código (linhas, ramos, caminhos). Inclui também a cobertura funcional (casos de uso, requisitos), a cobertura de dados (ranges, formatos, cenários de borda) e até a cobertura de ambiente (sistemas operacionais, navegadores, dispositivos) diferentes. Em sistemas baseados em IA, a "cobertura" pode incluir a exploração do espaço de estados e comportamentos que o modelo de IA é capaz de produzir, com o objetivo de identificar vieses ou resultados inesperados (Kesgin e Amasyali, 2023).

Performance: a performance envolve métricas como tempo de resposta, taxa de transferência, latência e escalabilidade. É fundamental garantir que o software atenda aos requisitos de desempenho, dada a crescente complexidade dos sistemas modernos e ao volume massivo de dados mencionado anteriormente no documento na Introdução. A IA pode ser empregada no planejamento para simular cargas de trabalho complexas ou detectar possíveis gargalos antes da execução real (WU e XIE., 2023).

Segurança: a segurança é um objetivo fundamental que, em algumas situações, é deixado de lado no planejamento de testes. Estabelecer metas de segurança envolve a identificação de vulnerabilidades, a avaliação de riscos e o planejamento de testes de penetração, testes de fuzzing ou testes de conformidade com normas de segurança. A incorporação de IA no planejamento pode ajudar na detecção automática de padrões de vulnerabilidade em código ou arquitetura, orientando os testes de segurança de maneira mais eficiente (López Martínez, A., et al, 2025)

Revista ARACÊ
ISSN: 2358-2472

Depois de estabelecer os objetivos, a escolha dos módulos-alvo é a fase em que o escopo do teste é restringido a partes específicas do software. A escolha inteligente de módulos-alvo é fundamental para a eficácia. Essa seleção pode levar em consideração vários critérios, como: Módulos que executam funções de alto impacto para o negócio ou que manipulam dados sensíveis têm prioridade, histórico de defeitos: Módulos que tiveram o maior número de defeitos no passado ou que são regularmente modificados (e, consequentemente, mais suscetíveis a novos erros) devem ser o foco de testes mais rigorosos, efeito das alterações: ferramentas de análise de impacto de código podem detectar módulos afetados por mudanças recentes, orientando os testes para as áreas com maior probabilidade de regressão.A escolha de módulos-alvo orientada por IA possibilita que o sistema "Caixa de Silício" maximize a utilização de seus recursos, concentrando os esforços na geração e execução de testes nas áreas que proporcionam o maior retorno em relação à detecção de falhas e garantia de qualidade. A IA pode ajudar nesse processo de seleção ao examinar repositórios de código, registros de alterações e dados de execução para identificar automaticamente módulos de alto risco ou áreas que precisam de mais atenção, tornando o planejamento mais flexível e baseado em dados (Landauer e Wurzenberger., 2024).

## 6 CICLO DE FEEDBACK E APRENDIZADO

O "Ciclo de Feedback e Aprendizado" representa o núcleo da capacidade de adaptação e evolução do modelo "Caixa de Silício". Ele reflete a habilidade do sistema de aprender de forma contínua a partir de suas próprias operações e dos resultados dos testes, ajustando e melhorando suas estratégias ao longo do tempo. Esse é um ponto de diferença significativo em relação aos métodos de teste convencionais, que são estáticos e necessitam de intervenção manual para se ajustarem a alterações. (Rouhi e Zamani., 2017).

O processo de realimentação para o modelo é o mecanismo fundamental que promove o aprendizado contínuo, também conhecido como continual learning. Cada execução de teste, independentemente de seu sucesso ou fracasso, produz informações úteis: quais cenários foram abordados, quais caminhos foram percorridos, quais dados de entrada foram eficazes na identificação de falhas, duração da execução e gravidade de quaisquer defeitos detectados.

Esse processo de realimentação constante é semelhante ao aprendizado que acontece em sistemas biológicos ou em modelos de aprendizado por reforço, nos quais o sistema adapta seu comportamento com base em "recompensas" (sucesso na detecção de falhas) e "penalidades" (ineficiência, falsos positivos).

Revista ARACÊ
ISSN: 2358-2472

Essas informações são processadas e empregadas para reabastecer os algoritmos de Inteligência Artificial que constituem o Gerador Inteligente e o Priorizador Adaptativo. Por exemplo, se um determinado tipo de caso de teste (ou uma sequência de ações) demonstrou ser especialmente eficiente na identificação de falhas, o modelo de IA pode ser treinado para criar mais testes desse tipo no futuro. De maneira semelhante, caso um conjunto de testes se torne ineficaz ou redundante, o modelo pode aprender a diminuir sua importância ou a não o produzir (Rouhi e Zamani., 2017).

O ajuste de heurísticas e a redefinição de prioridades são consequências diretas da realimentação. No âmbito da "Caixa de Silício", as "heurísticas" referem-se às diretrizes ou algoritmos que orientam a criação e a ordenação de testes. Esses algoritmos são aprimorados com base no feedback do Analisador de Resultados, podemos citar caso um módulo de software, anteriormente considerado estável, comece a apresentar falhas recorrentes, o Priorizador Adaptativo ajustará sua prioridade para que os testes nesse módulo sejam realizados com maior frequência e antecedência. Ou caso a criação de testes para uma funcionalidade específica não esteja atingindo a cobertura esperada, o Gerador Inteligente pode modificar suas heurísticas para investigar mais cenários ou tipos de dados para essa funcionalidade. (Khaliq, Z., et al., 2022).

Essa habilidade de auto ajuste e adaptação é o que possibilita que a "Caixa de Silício" preserve sua eficácia, mesmo diante de um software em constante transformação, com alterações nos requisitos, adição de novas funcionalidades e correções de falhas. A IA aprimora constantemente o "como" e o "onde" realizar os testes, aumentando ao máximo a eficácia na detecção de falhas.

Para comprovar a eficácia do Ciclo de Feedback e Aprendizado e evidenciar a importância do modelo "Caixa de Silício", o seu documento propõe a comparação com abordagens manuais. As métricas são fundamentais para medir o efeito e a eficácia da abordagem baseada em IA: observando o esforço (horas/pessoa) versus métodos manuais: Está métrica avalia a eficiência em termos de recursos humanos. Espera-se que o modelo "Caixa de Silício" diminua consideravelmente o tempo que os engenheiros de teste e desenvolvimento gastam na criação, execução e manutenção de testes repetitivos, permitindo que eles se concentrem em atividades de maior valor, como análise exploratória ou estratégia de qualidade. A automação baseada em IA pode permitir que os profissionais se dediquem a tarefas mais cognitivas (Khaliq, Z., et al., 2022).

Contudo a cobertura de testes antes/depois: Avaliar a cobertura antes e após a implementação da "Caixa de Silício" evidenciará a habilidade do Gerador Inteligente em explorar um espaço de teste mais extenso e variado. Espera-se um aumento significativo na cobertura, o que sugere que o sistema de IA está alcançando partes do código ou funcionalidades que seriam desafiadoras ou demoradas para serem abordadas por métodos manuais ou automatizados estáticos (Kesgin e Amasyali., 2023).

Revista ARACÊ
ISSN: 2358-2472

Ainda convém lembrar que o número e gravidade de defeitos identificados: essa é a métrica mais direta para medir a eficácia na detecção de falhas. Um modelo de teste sólido, como a "Caixa de Silício", deve ser capaz não só de detectar um maior número de defeitos, mas também de identificar aqueles de maior gravidade (maior severidade) e, preferencialmente, localizá-los mais cedo no ciclo de desenvolvimento. A habilidade da IA em reconhecer padrões complexos e investigar cenários imprevistos pode resultar na identificação de falhas que métodos tradicionais não conseguiriam detectar. A análise comparativa dessas métricas oferece evidências quantitativas do valor que o modelo "Caixa de Silício" agrega, ressaltando sua importância para aprimorar a qualidade do software, otimizar recursos e acelerar o ciclo de entrega. (Yang, M., et al., 2020).

## 7 CONSIDERAÇÕES FINAIS

A proposta da Caixa de Silício surge como uma abordagem vantajosa no teste de Software, reformulando os métodos atuais por meio de uma integração mais aprofundada de Inteligência Artificial. Portanto este modelo, tem por objetivo superar as limitações que as metodologias convencionais enfrentam, gerando grandes impactos transformadores na indústria:

Automação e redução do esforço manual: O sistema automatiza a criação, priorizando a execução de testes, liberando a execução de tarefas repetitivas. Permitindo que profissionais foquem em atividades que demandam maior valor estratégico e de grande importância, influenciando numa maior influência do capital humano. Aumento na cobertura e eficiência na detecção de defeitos: Por possuir gerador de casos de teste e priorizar a adaptabilidade, a Caixa de Silício explora cenários complexos e otimiza a ordem de execução dos testes. Isso possibilita em uma cobertura maior e na identificação antecipada de falhas, reduzindo significativamente os custos na correção e contribuindo na melhora da qualidade do software desde o seu início. Melhoria contínua e adaptação dinâmica: Executando com um ciclo de feedback, o modelo aprende a se adaptar às mudanças, mudanças que ocorrem no software, tornando-se cada vez mais eficiente na detecção de padrões de falha. Essa capacidade de aprendizado contínuo, aliada à otimização da alocação de recursos que se baseiam em riscos, garante uma base de código mais robusta e confiável.

Para validar e expandir a aplicação da “Caixa de Silício”, um dos próximos passos envolvem sua validação em domínios distintos da engenharia, como sistemas de internet das coisas (IoT) E softwares embarcados, áreas essas que apresentam restrições de hardware, requisitos de tempo real e interações complexas com o ambiente físico. A adaptação e o desempenho do modelo em tais contextos testaram sua robustez e versatilidade, com o intuito de evidenciar a capacidade da

Revista ARACÊ
ISSN: 2358-2472

inteligência artificial de gerenciar situações complexas e a gerir diversidades inerentes a esses ecossistemas, que frequentemente demandam abordagens de teste altamente específicas.

A extensão do escopo do modelo proposto representa outro avanço a se pensar, enquanto o foco inicial do modelo reside na detecção funcional e estrutural de defeitos, a avaliação da experiência do usuário e da conformidade com padrões de acessibilidade, a inteligência artificial pode contribuir para simular interações de usuários com diferentes perfis e necessidades, identificar barreiras de uso e verificar automaticamente a aderência a diretrizes de acessibilidade, promovendo softwar mais inclusivo e intuitivo. Essa expansão solidifica a “Caixa de Silício” como uma ferramenta de avaliação de qualidade ainda mais abrangente e holística.

Revista ARACÊ
ISSN: 2358-2472

## REFERÊNCIAS

ALETI, A. Software Testing of Generative AI Systems: Challenges and Opportunities. In: IEEE/ACM INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING: FUTURE OF SOFTWARE ENGINEERING (ICSE-FoSE), 2023, Melbourne. Anais [...]. Melbourne: IEEE, 2023. p. 4-14.

AMALFITANO, D. et al. Artificial Intelligence applied to Software Testing: a Tertiary Study. **ACM Computing Surveys**, 17 ago. 2023.

ANDRADES, G. GAP Analysis in Testing: How AI Impact? **AccelQ**, [2024]. Disponível em: <https://www.accelq.com/blog/gap-analysis-in-testing/>. Acesso em: 12 jun. 2025.

ANTHROPIC. Claude Opus 4. **Anthropic**, [2025]. Disponível em: <https://www.anthropic.com/claude/opus>. Acesso em: 14 jun. 2025.

ANTHROPIC. Claude 4. **Anthropic News**, 2025. Disponível em: <https://www.anthropic.com/news/claude-4>. Acesso em: 14 jun. 2025.

ANTHROPIC. Anthropic's new AI model could be a game changer for developers: Claude Opus 4 'pushes the boundaries in coding'. **IT Pro**, [S.l.], 26 maio 2025.

ARRUDA CAVALCANTE, Josué et al. Explorando a inteligência artificial no ensino médio: introdução à ia com alunos do 1º utilizando a plataforma code. org. **Revista Ibero-Americana de Humanidades**, Ciências e Educação, v. 9, n. 8, p. 2364-2385, 2023.

ATLASSIAN. Os diferentes tipos de testes em software. **Atlassian**, [2024]. Disponível em: <https://www.atlassian.com/br/continuous-delivery/software-testing/types-of-software-testing>. Acesso em: 12 jun. 2025.

AVO AUTOMATION. What is Gap Analysis in Software Testing? **Avo Automation**, [2024]. Disponível em: < https://avoautomation.com/blog/gap-analysis-in-software-testing>. Acesso em: 12 jun. 2025.

BAKAR, Normi Sham Awang Abu. Machine Learning Implementation in Automated Software Testing: A Review. **Journal of Data Analytics and Artificial Intelligence Applications**, v. 1, n. 1, p. 110-122.

BELÉN, B. P. A era dos dados: quantos geramos e como isso impacta nossa vida. **Linkages**, 2023. Disponível em: <https://linkages.com.br/2023/03/29/dados-quantos-geramos-e-como-isso-impacta-nossa-vida/>.

BORTS, B. V.; TKACHENKO, I. V.; TKACHENKO, V. I. Cable Free Transmission of Electricity: from Nikola Tesla to Our Time. **East European Journal of Physics**, [S.l.], v. 3, n. 4, p. 51-59, 10 mar. 2017.

CAMACHO, Teste de Software: Benefícios, Desafios e Tendências Futuras. **Software Engineers Academy**, 2024. Disponível em: <https://softwareengineersacademy.wordpress.com/2024/08/20/teste-de-software-beneficios-desafios-e-tendencias-futuras/>. Acesso em: 12 jun. 2025.

Revista ARACÊ
ISSN: 2358-2472

CAROLINE, A.; EDSON. Geração Automática de Casos de Testes de Software: Uma Avaliação Empírica na Utilização das Ferramentas Evosuite e Randoop. **Revista do Encontro de Gestão e Tecnologia**, v. 1, n. 09, p. e373–e373, 2024.

COSTA DIAS, J. da. Teste de Software com IA: Um Mapeamento Sistemático da Literatura. 2024. Trabalho de Conclusão de Curso - Universidade Federal de Pernambuco, Recife, 2024. Disponível em: <https://repositorio.ufpe.br/jspui/bitstream/123456789/50401/1/TCC%20Jailson%20da%20Costa%20Dias.pdf>. Acesso em: 12 jun. 2025.

EDWARDS, S. H. Using software testing to move students from trial-and-error to reflection-in-action. **ACM Digital Library**, [2004]. Disponível em: <https://dl.acm.org/ft_gateway.cfm?id=971312&ftid=251805&dwn=1>. Acesso em: 12 jun. 2025.

GAROUSI, V.; FELDERER, M.; MÄNTYLÄ, M. V. AI-powered test automation tools: A systematic review and empirical evaluation. **arXiv**, 2024. Disponível em: https://arxiv.org/abs/2401.01830. Acesso em: 11 jun. 2025.

GUO, X.; OKAMURA, H.; DOHI, T. Automated Software Test Data Generation With Generative Adversarial Networks. **IEEE Access**, v. 10, p. 20690–20700, 2022.

Gurcan, Fatih, et al. "Evolution of software testing strategies and trends: Semantic content analysis of software research corpus of the last 40 years." **IEEE Access** 10 (2022): 106093-106109.

HARRIES, R. et al. DRIFT: Deep Reinforcement Learning for Functional Software Testing. arXiv, 2020. Disponível em: https://arxiv.org/abs/2007.10662. Acesso em: 11 jun. 2025.

HAUPTMANN, B. et al. Did We Test Our Changes? Assessing Alignment between Tests and Development in Practice. **Teamscale**, [2013]. Disponível em: <https://teamscale.com/hubfs/Publications/2013-did-we-test-our-changes-assessing-alignment-between-tests-and-development-in-practice.pdf>. Acesso em: 12 jun. 2025.

IBM - Brasil. **IBM Brasil**, [2025]. Disponível em: <https://www.ibm.com/br-pt?utm_content=SRCWW&p1=Search&p4=43700065491227438&p5=e&gclid=Cj0KCQjww4-hBhCtARIsAC9gR3bGYP-6anQYPhMug1XYI8IvbNVDdlrYap_QoI8XBZSHCn8WO3TaVykaAiAKEALw_wcB&gclsrc=aw.ds>. Acesso em: 11 jun. 2025.

JIANG, D. et al. An Integrated Bidirectional Multi-Channel Opto-Electro Arbitrary Waveform Stimulator for Treating Motor Neurone Disease. 2021 IEEE International Symposium on Circuits and Systems (ISCAS), v. 181, p. 1–4, maio 2021.

KESGIN, H. T.; AMASYALI, M. F. Iterative Mask Filling: An Effective Text Augmentation Method Using Masked Language Modeling. **Communications in Computer and Information Science**, p. 450–463, 23 dez. 2023.

KEYSIGHT TECHNOLOGIES. The evolution of AI in software testing: From machine learning to agentic AI. Santa Rosa, CA: Keysight, 2025. White paper. Disponível em: https://www.keysight.com/us/en/assets/7123-1103/white-papers. Acesso em: 11 jun. 2025.

Revista ARACÊ
ISSN: 2358-2472

KHALIQ, Zubair; FAROOQ, Sheikh Umar; KHAN, Dawood Ashraf. Artificial intelligence in software testing: Impact, problems, challenges and prospect. arXiv preprint **arXiv**:2201.05371, 2022. Disponível em: < https://arxiv.org/abs/2201.05371>. Acesso em: 12 jun. 2025.

KHAN, Mohd Ehmer; KHAN, Farmeena. A comparative study of white box, black box and grey box testing techniques. **International Journal of Advanced Computer Science and Applications**, v. 3, n. 6, 2012.

LANDAUER, M.; FLORIAN SKOPIK; WURZENBERGER, M. A Critical Review of Common Log Data Sets Used for Evaluation of Sequence-Based Anomaly Detection Techniques. v. 1, n. **FSE**, p. 1354–1375, 12 jul. 2024.

LÓPEZ MARTÍNEZ, A.; CANO, A.; RUIZ-MARTÍNEZ, A. Generative Artificial Intelligence-Supported Pentesting: A Comparison between Claude Opus, GPT-4, and Copilot. **arXiv**, [S.l.], 12 jan. 2025. Disponível em: <https://arxiv.org/abs/2501.06963>. Acesso em: 21 jun. 2025.

Neto, Arilo; Claudio, Dias. Introdução a teste de software. **Engenharia de Software Magazine**, v. 1, p. 22, 2007.

OLTEANU, A. Claude 4: Tests, Features, Access, Benchmarks, and More. **DataCamp**, [2025]. Disponível em: < https://www.datacamp.com/blog/claude-4>. Acesso em: 14 jun. 2025.

OpenAI. "Introducing SWE-bench Verified." **OpenAI**, 2024, openai.com/index/introducing-swe-bench-verified/. Acesso em: 24 jun. 2025.

OSTRAND, Thomas. White-Box Testing. **Encyclopedia of Software Engineering**, 2002.

PAL, P. Gap Analysis. **Think360 Studio**, [2024]. Disponível em: <https://think360studio.com/blog/gap-analysis>. Acesso em: 12 jun. 2025.

PORTAL INSIGHTS. Quais são as técnicas de testes de software? **Portal Insights**, 2025. Disponível em: <https://www.portalinsights.com.br/perguntas-frequentes/quais-sao-as-tecnicas-de-testes-de-software>. Acesso em: 12 jun. 2025.

QODO TEAM. Gap Analysis in Software Testing. **Qodo**, [2024]. Disponível em: <https://www.qodo.ai/blog/gap-analysis-in-software-testing/>. Acesso em: 12 jun. 2025.

RAMADAN, A.; YASIN, H.; PEKTAS, B. The Role of Artificial Intelligence and Machine Learning in Software Testing. **arXiv**, [S.l.], 4 set. 2024.

REDDIT. Discussões sobre IA e automação de testes em software. r/softwaredevelopment; r/QualityAssurance. **Reddit**, 2024-2025. Disponível em: https://www.reddit.com. Acesso em: 11 jun. 2025.

REINSEL, D.; GANTZ, J.; RYDNING, J. Data Age 2025: The Evolution of Data to Life-Critical Don't Focus on Big Data; Focus on the Data That's Big. **Seagate**, [2017]. Disponível em: < https://www.seagate.com/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf>. Acesso em: 12 jun. 2025.

Revista ARACÊ
ISSN: 2358-2472

RICCA, F.; MARCHETTO, A.; STOCCO, A. A multi-year grey literature review on AI-assisted test automation. arXiv, 2024. Disponível em: https://arxiv.org/abs/2403.02951. Acesso em: 11 jun. 2025.

ROUHI, A.; ZAMANI, B. A model-based framework for automatic generation of a pattern language verifier. **Software: Practice and Experience**, v. 47, n. 12, p. 1945–1980, 24 ago. 2017.

SRINIVAS, N.; NAGOOR BEGUM, T. A. Black Box and White Box Testing Techniques – A Literature Review. **International Journal of Embedded Systems and Applications (IJESA)**, v. 2, n. 2, p. 29–39, jun. 2012.

STALLWOOD, K. Topsy: The Elephant We Must Never Forget. **Palgrave studies in animals and literature**, n. 978-3-319-98287-8, p. 227–242, 1 jan. 2018.

TRYBE. Teste de software: o que é, importância e como fazer? [O GUIA], **Blog da Trybe**, 2024. [Online]. Available: https://blog.betrybe.com/tecnologia/teste-software-guia/. Acesso em: 14 jun. 2025.

WAGER, M. The Evolution of AI in Software Testing: From Machine Learning to Agentic AI. **CSRWire**, [2025]. Disponível em: < https://www.csrwire.com/press_releases/818026-evolution-ai-software-testing-machine-learning-agentic-ai>. Acesso em: 18 jun. 2025.

WANG, S.; GUO, M.; TAN, W. From code generation to software testing: AI Copilot with context-based RAG. **arXiv**, 2025. Disponível em: https://arxiv.org/abs/2504.00928. Acesso em: 11 jun. 2025.

WILLS, I. The Edisonian Method: Trial and Error. **Studies in History and Philosophy of Science**, v. 52, n. 978-3-030-29939-2, p. 203–222, 2019.

Wotawa, Franz. "On the Use of Available Testing Methods for Verification & Validation of AI-based Software and Systems." *SafeAI@ AAAI*. 2021.

WU, N.; XIE, Y. A Survey of Machine Learning for Computer Architecture and Systems. **ACM Computing Surveys**, v. 55, n. 3, p. 1–39, 30 abr. 2023.

YANG, M.; GUO, J.; BAI, L. A Data Privacy-preserving Method for Students' Physical Health Monitoring by Using Smart Wearable Devices. 1 ago. 2020.

YE, J. et al. Multi-Regularized Correlation Filter for UAV Tracking and Self-Localization. **IEEE Transactions on Industrial Electronics**, v. 69, n. 6, p. 6004–6014, 1 jun. 2022.