

Universidade Estadual de Mato Grosso do Sul

Unidade Universitária de Dourados

Projeto de Iniciação Científica Internacional

**Experimento para Análise Comparativa de Ferramentas de
Inteligência Artificial Generativa em Atividades de
Programação Introdutória**

Discentes: Igor Roberto Michalski de Souza,

Kauan Henrick Teixeira da Silva e

Victor Rech Vendruscolo

Orientador: Prof. Dr. Jorge Marques Prates

1: Protocolo do Experimento

Para garantir o rigor científico, o planejamento e a execução do estudo seguiram as diretrizes de experimentação em engenharia de software propostas por [Wohlin et al. \(2012\)](#). Em seguida, são descritas as fases de planejamento e execução da avaliação dos resultados.

1.1 Definição do Experimento

O objetivo deste estudo é investigar como um Modelo de Linguagem de Grande Escala (LLMs) influenciam o desempenho de estudantes do primeiro ano do curso de Sistema de Informação em tarefas de programação utilizando a metodologia *Test-Driven Development* (TDD), tendo como C a linguagem base. A motivação do experimento é avaliar a efetividade pedagógica do uso de IA generativa no apoio ao ensino de práticas modernas de desenvolvimento, uma vez que o TDD, apesar de seus benefícios comprovados na qualidade do código, apresenta uma baixa taxa de adoção na indústria por exigir a criação de mais código e desenvolvedores experientes. A utilização de *Generative artificial intelligence* (GenAI) pode reduzir o esforço adicional imposto pelo TDD ([Mock et al., 2025](#)).

A definição do objetivo segue o modelo GQM (*Goal-Question-Metric*), conforme proposto por [Basili et al. \(1994\)](#) e aplicado na metodologia de experimentação em engenharia de software descrita por [Wohlin et al. \(2012\)](#), conforme detalhado na tabela 1.1:

Tabela 1.1: Modelo GQM (Goal-Question-Metric) do Estudo.

Objetivo	Analisar o uso de LLMs como ferramenta de apoio para estudantes que aplicam a metodologia TDD, a fim de avaliar o impacto no desempenho, na qualidade do trabalho e na percepção dos alunos.
Perguntas	<ol style="list-style-type: none">1. O uso de IA como ferramenta de auxílio afeta a qualidade do código/testes em tarefas que seguem a metodologia TDD?2. Como a experiência com o uso da ferramenta de IA afeta a percepção de utilidade, dificuldade e confiança dos alunos ao programar com TDD?
Métricas	<ol style="list-style-type: none">1. Qualidade do código: Avaliada por meio de uma rubrica objetiva que considera clareza, estrutura e organização.2. Cobertura e Qualidade dos testes: Percentual de cobertura de código e análise da qualidade dos casos de teste (e.g., relevância, asserções).3. Log de Interação com a IA Generativa: Análise qualitativa e quantitativa dos prompts, respostas e uso efetivo no código/testes.4. Percepção de utilidade e confiança: Medida através de questionários com escala Likert e perguntas abertas, aplicados antes e depois do experimento.

1.1.1 Fatores e Tratamentos do Experimento

Para investigar as questões de pesquisa, o experimento foi simplificado e agora se estrutura em torno de um único fator principal, com dois tratamentos distintos. Como todos os participantes aplicarão a metodologia TDD, o fator de interesse do estudo é a ferramenta utilizada como auxílio.

- **Fator 1:** Ferramenta de Auxílio: Este fator avalia o impacto da assistência da Inteligência Artificial (IA) generativa no processo de desenvolvimento com TDD. A sua inclusão é justificada pela crescente adoção de LLMs como ferramentas de apoio na programação.

Os tratamentos, que representam as duas condições experimentais, são:

- Tradicional: Os participantes desenvolvem as soluções aplicando a metodologia TDD sem o auxílio de ferramentas de IA generativa
- Assistido por IA: Os participantes desenvolvem as soluções aplicando a metodologia TDD e são incentivados a utilizar o ChatGPT como ferramenta de apoio.

1.1.2 Seleção das Variáveis

Neste estudo, as variáveis são classificadas em dois tipos, conforme sua função na investigação: variáveis independentes e variáveis dependentes. As variáveis independentes correspondem aos elementos que podem ser controlados ou escolhidos com o intuito de analisar como influenciam outras variáveis. Já as variáveis dependentes representam os efeitos observados, ou seja, são aquelas que sofrem influência direta das mudanças nas variáveis independentes e servem como base para avaliar os resultados da pesquisa. A seguir, são apresentadas as variáveis independentes e dependentes consideradas na análise.

- **Variáveis Independentes** A variável independente deste estudo é a Ferramenta de Auxílio, que consiste nos dois tratamentos definidos na seção anterior: o desenvolvimento Tradicional (sem IA) e o desenvolvimento Assistido por IA seção [1.1.1](#).
- **Variáveis Dependentes**
 - **Qualidade do código:** Avaliada por uma rubrica objetiva. Para a medição desta variável, foi utilizada uma versão adaptada da rubrica proposta por [Stegeman et al. \(2016\)](#). O modelo original avalia programas de estudantes em quatro dimensões. No entanto, como a criação de comentários e documentação não foi um foco de ensino do curso nem um requisito nos exercícios, o critério de 'Documentação' foi removido. A rubrica final, pode ser consultada no Apêndice [A](#).
 - **Qualidade e Cobertura dos testes:** Percentual de cobertura do código e análise da qualidade dos casos de teste.
 - **Log da Interação com IA generativa:** Análise das interações entre os participantes e a ferramenta de IA generativa, incluindo tipo de perguntas, frequência de uso, aceitação ou rejeição das respostas sugeridas. Essa variável permitirá avaliar não apenas o produto final (código/testes), mas também o processo de interação.

- **Percepção de utilidade e confiança:** Medida através de questionários com escala Likert e perguntas abertas.

1.1.3 Seleção da amostra

A amostra utilizada no estudo é composta por 4 problemas de programação, caracterizados pela complexidade ciclomática. Esses problemas foram divididos em duas listas balanceadas, cada uma contendo 2 exercícios de complexidade semelhante. Cada rodada do experimento utilizará exclusivamente uma das listas, assegurando comparabilidade entre os cenários. A seleção desses problemas visa criar um conjunto de cenários de teste representativo, permitindo avaliar o desempenho das ferramentas de IA e das metodologias de desenvolvimento em diferentes contextos de dificuldade e escopo.

Cada problema foi escolhido para avaliar habilidades específicas, como manipulação de strings, lógica condicional, recursão e operações com arrays, refletindo desafios frequentemente encontrados por programadores. As Tabelas 1.2 e 1.3 apresentam a caracterização dos programas utilizados no experimento, utilizando a Complexidade Ciclomática como indicador da complexidade estrutural de cada problema. No decorrer do texto, os problemas serão referenciados por seu ID específico, conforme demonstrado na primeira coluna da tabela.

Tabela 1.2: Lista A de problemas (Rodada 1)

ID	Programa	Complexidade Ciclomática
P01	Calculadora de MDC	3
P02	Soma dos Dígitos de um Número	3

Tabela 1.3: Lista B de problemas (Rodada 2)

ID	Programa	Complexidade Ciclomática
P03	Contador de Vogais	3
P04	Conversor Celsius para Fahrenheit	3

A Complexidade Ciclomática é uma métrica de software que quantifica o número de caminhos independentes no fluxo de controle de um programa. Desenvolvida no estudo McCabe (1976), valores mais altos indicam um código com mais ramificações e decisões, o que o torna mais complexo de testar e manter. Nas Tabelas 1.2 e 1.3, todos os problemas apresentam complexidade ciclomática igual a 3, garantindo equivalência entre as rodadas. A média é 3, com desvio padrão igual a 0.

1.1.4 Seleção da Ferramenta de IA

Para este experimento, a abordagem para a escolha da ferramenta de Inteligência Artificial foi padronizar o uso de um único modelo para todos os participantes do grupo assistido por IA. Essa decisão metodológica visa remover a variabilidade entre diferentes LLMs como uma possível

variável de confusão, permitindo que a análise se concentre unicamente no impacto da assistência da IA no processo de desenvolvimento com TDD.

A escolha do ChatGPT como ferramenta padrão para o grupo experimental fundamenta-se em sua vasta popularidade, desempenho técnico documentado e eficácia comprovada em contextos educacionais, especialmente no ensino de programação para iniciantes.

A literatura acadêmica recente revela que estudantes e professores do ensino superior já estão cientes dos LLMs e, em sua maioria, já utilizaram o ChatGPT por curiosidade, para gerar conteúdo ou como ferramenta de tutoria (Ribeiras et al., 2025). Essa familiaridade prévia minimiza a curva de aprendizado da ferramenta em si, permitindo que os participantes se concentrem nos objetivos do experimento.

Estudos específicos sobre a aplicação de LLMs no ensino de programação para iniciantes demonstram o impacto positivo da ferramenta. O ChatGPT tem se mostrado eficaz ao oferecer suporte interativo e personalizado, permitindo que os alunos esclareçam dúvidas e acessem exemplos práticos e feedbacks em tempo real. Em um relato de experiência com um curso de Python para iniciantes, Maia e Sarkis (2025) reportaram que 100% dos alunos destacaram uma melhora no entendimento dos conceitos de programação, na facilidade de acesso à informação e na interatividade proporcionada pela ferramenta. Alunos relataram que a ferramenta "ajuda bastante a entender a sintaxe" e fornece um "aprendizado mais estruturado".

Além do seu valor pedagógico, o ChatGPT e os modelos GPT da OpenAI apresentam fortes capacidades na geração de código e testes. O estudo de Olmez e Gehringer (2024) mostra que o modelo é capaz de gerar testes que se assemelham aos escritos por humanos, sendo particularmente eficaz quando combinado com prompts bem estruturados. Em uma avaliação comparativa, Ferreira et al. (2025) concluíram que os modelos GPT (GPT-3.5 e GPT-4) tiveram um desempenho superior na geração de cenários de teste, com erros de sintaxe em apenas um dos 50 arquivos gerados.

1.1.4.1 Prompt

Os participantes dos grupos que utilizarão LLMs terão liberdade para interagir com as ferramentas por meio de prompts livres, ou seja, poderão escrever suas perguntas e comandos de maneira natural, sem um roteiro fixo. Essa escolha visa garantir que a experiência com a ferramenta seja o mais próxima possível do uso real e espontâneo que teriam em um ambiente de aprendizado assistido por IA.

Apesar dessa liberdade, todos os participantes receberão uma orientação prévia sobre boas práticas na formulação de prompts, incluindo exemplos claros de como solicitar explicações, gerar testes automatizados, identificar erros e revisar soluções. Essa base teórica mínima assegura que os alunos tenham condições de interagir com as ferramentas de maneira produtiva e alinhada com os objetivos do experimento.

Essa base teórica mínima garantirá que os alunos possam interagir com as ferramentas de maneira mais produtiva e alinhada aos objetivos do experimento. O treinamento incluirá exemplos claros de como fornecer contexto adequado, solicitar explicações, gerar testes automatizados, identificar erros e revisar soluções. A engenharia de prompt é um processo iterativo, e a experimentação é essencial para otimizar a precisão e a relevância das respostas da IA. Portanto, essa capacita-

ção inicial visa dar aos participantes o conhecimento necessário para refinar continuamente seus prompts e obter os melhores resultados possíveis das ferramentas ([Amazon Web Services, 2025](#)).

1.1.5 Desenho Experimental

O experimento é desenvolvido para avaliar o impacto do uso de Modelos de Linguagem (LLMs) como ferramenta de apoio no processo de desenvolvimento orientado a testes (TDD), assegurando que todos os participantes possuam o conhecimento técnico necessário e que os dados coletados sejam suficientes para análise. O estudo é conduzido pelo autor deste trabalho e por mais dois alunos do curso de Ciência da Computação, que são responsáveis por gerenciar o experimento, incluindo ministrar as aulas de formação dos alunos participantes, definir os problemas que serão resolvidos durante o experimento, preparar o ambiente de execução no período de realização do estudo e fiscalizar os participantes para que todos mantenham as práticas estipuladas.

O presente experimento será iniciado com uma etapa de formação, na qual todos os participantes receberão um treinamento padronizado sobre Test-Driven Development (TDD) e boas práticas de utilização de Modelos de Linguagem (LLMs). Esse treinamento terá como objetivo nivelar o conhecimento técnico dos alunos, apresentando os conceitos fundamentais do ciclo de TDD (escrever o teste, implementar e refatorar), além de orientações sobre o uso responsável e eficiente de LLMs, incluindo a formulação adequada de prompts, a validação das respostas geradas e a conscientização sobre aspectos éticos. Também será demonstrado o ambiente de desenvolvimento e a utilização da biblioteca "assert.h", garantindo que todos estejam familiarizados com as ferramentas antes da execução experimental.

Na fase de execução, os alunos serão divididos aleatoriamente em dois grupos distintos: um grupo controle, responsável por resolver os exercícios aplicando TDD sem apoio de LLM, e um grupo experimental, que aplicará TDD com o auxílio de uma LLM como ferramenta de apoio na geração dos testes. Cada participante terá 90 minutos para resolver os exercícios da rodada, seguindo todo o ciclo do TDD (escrita de testes, implementação e refatoração). Esse tempo foi definido para garantir condições uniformes entre os participantes. Para cada conjunto de exercícios, haverá sempre um aluno no grupo controle e outro no grupo experimental, permitindo comparação direta entre os tratamentos. Ao término da rodada, será aplicado um formulário de avaliação já previsto no protocolo. Esse instrumento tem como objetivo coletar dados sobre a percepção de utilidade, dificuldade e confiança dos alunos em relação à atividade realizada e ao uso (ou não) da IA generativa durante a resolução. Após a conclusão da primeira rodada de resoluções, será realizada uma segunda rodada com os mesmos participantes, porém com novos exercícios, preservando a mesma lógica de distribuição e os grupos serão invertidos. O objetivo dessa segunda aplicação é ampliar a quantidade de dados disponíveis para análise, fortalecendo os resultados obtidos. Durante ambas as rodadas, serão coletados os artefatos produzidos.

1.1.6 Instrumentação

Os artefatos centrais do experimento são os 4 problemas de programação descritos nas Tabelas [1.2](#) e [1.3](#). Esses problemas foram elaborados para avaliar habilidades específicas em lógica de programação, manipulação de estruturas de dados e algoritmos fundamentais em C. Cada conjunto de

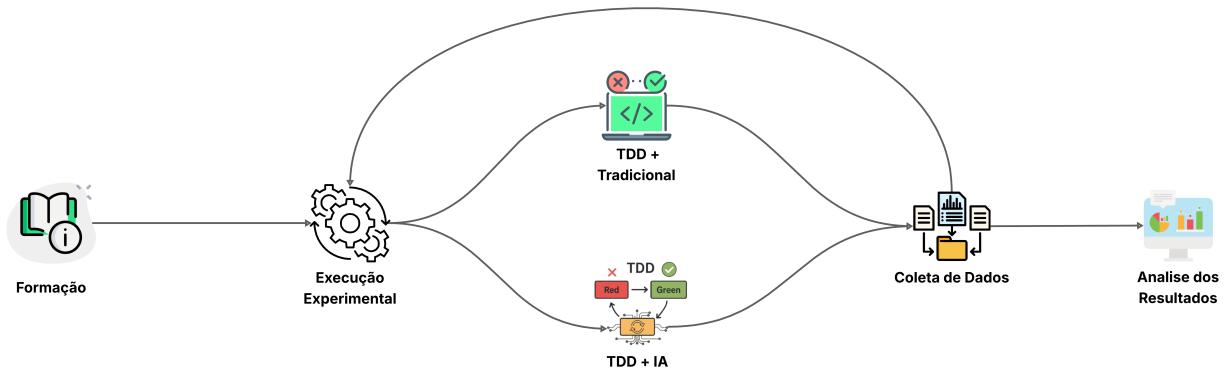


Figura 1.1: Visão Geral do Processo Experimental.

problemas serviu como tarefa prática para os participantes. Todos os exercícios foram balanceados com complexidade ciclomática igual a 3, assegurando equivalência entre as rodadas.

No ambiente de desenvolvimento, a biblioteca assert.h (padrão da linguagem C) foi utilizada como o principal instrumento para a escrita das asserções nos casos de teste, conforme demonstrado na etapa de formação.

Para os grupos de tratamento que utilizaram a abordagem assistida por Inteligência Artificial, foi empregado o ChatGpt, selecionada conforme o processo descrito na seção 1.1.4. Essa ferramenta serviu como o principal instrumento de auxílio para a geração de código e testes.

A coleta de dados foi realizada por meio de múltiplos instrumentos:

- Qualidade do Código: A avaliação da qualidade do código-fonte produzido foi realizada por meio de uma rubrica adaptada de Stegeman et al. (2016), apresentada no Apêndice A. Essa rubrica permitiu uma análise objetiva de critérios como correção, design, estrutura e legibilidade.
- Cobertura dos testes: Foi utilizada a ferramenta gcov, parte do conjunto de ferramentas do compilador GCC, para aferir o percentual de cobertura de código alcançado pelos testes (escritos com assert.h) desenvolvidos pelos alunos.
- Log de Interação com a IA Generativa: Foram coletados os registros das conversas dos participantes com a ferramenta de IA, permitindo análise qualitativa e quantitativa do processo de interação. Esse log será usado para investigar o tipo de perguntas realizadas, frequência de uso, aceitação ou rejeição das respostas sugeridas e como essas interações influenciaram a implementação dos exercícios.
- Percepção de Utilidade e Confiança: Foram aplicados questionários antes e depois da fase experimental. Esses questionários continham perguntas com escala Likert e questões abertas, destinadas a medir a percepção dos alunos sobre as ferramentas e metodologias utilizadas.

Para garantir a organização, a transparência na coleta e a futura análise dos dados (códigos-fonte, arquivos de teste e respostas dos questionários), foi utilizado um sistema de gerenciamento de arquivos estruturado. Essa abordagem assegura a integridade e a rastreabilidade dos artefatos coletados, permitindo a verificação e a replicação do estudo.

Referências Bibliográficas

AMAZON WEB SERVICES What is prompt engineering? [On-line], acesso em: 03 jul. 2025.

Disponível em <https://aws.amazon.com/what-is/prompt-engineering/>

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. The goal question metric approach. 1994.

Disponível em <https://api.semanticscholar.org/CorpusID:13884048>

FERREIRA, M.; VIEGAS, L.; FARIA, J. P.; LIMA, B. Acceptance test generation with large language models: An industrial case study. 2025.

Disponível em <https://arxiv.org/abs/2504.07244>

MAIA, S.; SARKIS, L. Utilização de llm como ferramenta de apoio no ensino-aprendizagem de programação python para iniciantes: Um relato de experiência. In: *Anais do XXXIII Workshop sobre Educação em Computação*, Porto Alegre, RS, Brasil: SBC, 2025, p. 385–396.

Disponível em <https://sol.sbc.org.br/index.php/wei/article/view/36182>

MCCABE, T. A complexity measure. *IEEE Transactions on Software Engineering*, v. SE-2, n. 4, p. 308–320, 1976.

MOCK, M.; MELEGATI, J.; RUSSO, B. *Generative ai for test driven development: Preliminary results* Springer Nature Switzerland, p. 24–32, 2025.

Disponível em http://dx.doi.org/10.1007/978-3-031-72781-8_3

OLMEZ, M. M.; GEHRINGER, E. Automation of test skeletons within test-driven development projects. In: *2024 36th International Conference on Software Engineering Education and Training (CSEET)*, 2024, p. 1–10.

RIBEIRAS, A. D. S.; RIBEIRAS, V. D. B.; DE SOUZA, G. B. O uso de modelos de linguagem na educação superior: Uma revisão sistemática da literatura. *Revista de Informática Aplicada*, v. 21, n. 1, p. 1–14, 2025.

STEGEMAN, M.; BARENSEN, E.; SMETSERS, S. Designing a rubric for feedback on code quality in programming courses. In: *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, Koli Calling '16, New York, NY, USA: Association for Computing Machinery, 2016, p. 160–164 (*Koli Calling '16*,).

Disponível em <https://doi.org/10.1145/2999541.2999555>

WOHLIN, C.; RUNESON, P.; HST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLN, A. *Experimentation in software engineering*. Springer Publishing Company, Incorporated, 2012.

A: Rubrica de Avaliação de Qualidade de Código

Tabela A.1: Rubrica de Avaliação de Qualidade de Código

Categoría	Critério de Avaliação	1 - Insuficiente	2 - Básico	3 - Bom/Excelente	Pontos
Correção	O programa funciona conforme o esperado?	O código não compila/executa ou falha em passar nos testes mais básicos. A lógica principal está incorreta.	O código passa na maioria dos testes principais, mas falha em casos de borda (ex: arrays vazios, entradas inválidas).	O código passa em todos os casos de teste propostos, incluindo os casos de borda, e a lógica é robusta.	/3
Design e Estrutura	O código está bem organizado?	A lógica está toda em um único bloco, sem uso de funções auxiliares. Há repetição excessiva de código.	O código utiliza funções, mas a separação de responsabilidades não é clara ou poderia ser melhor.	O código é modularizado em funções pequenas e com responsabilidades bem definidas. A estrutura é lógica e eficiente.	/3
Estilo e Legibilidade	O código é fácil de ler e entender?	Nomes de variáveis e funções são vagos (ex: a, b, func1). A indentação é inexistente ou inconsistente.	Nomes são comprehensíveis, mas não seguem um padrão claro. A formatação é aceitável, mas com inconsistências.	Nomes de variáveis e funções são descritivos e seguem um padrão consistente. O código é bem formatado e limpo.	/3
					Total /9