Um Estudo Empírico de Padrões de Prompt usados durante a realização de tarefas de programação e compreensão de código

Bruno Alves de Oliveira
Universidade Tecnológica Federal do Paraná - Campus Campo Mourão
Brazil
bruno-alves90@hotmail.com

Jonan Richards
Radboud University
Netherlands
jonan.richards@ru.nl

Walter Takashi Nakamura
Universidade Tecnológica Federal do Paraná - Campus Campo Mourão
Brazil
waltertakashi@utfpr.edu.br

Ivanilton Polato
Universidade Tecnológica Federal do Paraná - Campus Campo Mourão
Brazil
ipolato@professores.utfpr.edu.br

Reginaldo Ré
Universidade Tecnológica Federal do Paraná - Campus Campo Mourão
Brazil
reginaldo@utfpr.edu.br

Mairieli Wessel
Radboud University
Netherlands
mairieli.wessel@ru.nl

Igor Wiese
Universidade Tecnológica Federal do Paraná - Campus Campo Mourão
Brazil
igor@utfpr.edu.br

Resumo

A popularização dos modelos de Linguagem de Grande Escala (LLMs), especialmente os assistentes inteligentes de programação, também conhecidos como copilotos, tem transformado a engenharia de software, permitindo que programadores de diversos níveis obtenham suporte em tarefas de compreensão e implementação de código. O objetivo principal desta pesquisa foi compreender como diferentes perfis de desenvolvedores interagem com um copiloto, investigando a influência da experiência e do estilo cognitivo na elaboração de prompts e nas estratégias de resolução de problemas. A metodologia consistiu em um estudo empírico com estudantes e profissionais de diferentes nacionalidades, totalizando 27 participantes. Ao todo, os participantes realizaram seis tarefas de desenvolvimento e de correção de defeitos utilizando o Copiloto livremente. Os resultados indicaram que o esforço linguístico na elaboração dos prompts é adaptativo: tarefas mais complexas resultam em maior verbosidade. Qualitativamente, a categoria de prompt predominante foi a "Pergunta Aberta", sugerindo que os usuários utilizam o copiloto principalmente como tutor para compreender o contexto do código, e não apenas como gerador de soluções. A análise comparativa revelou uma associação estatisticamente significativa entre o perfil do usuário e o tipo de interação. Conclui-se que a interação com assistentes de programação é influenciada pela combinação entre o estilo cognitivo e o nível de experiência dos usuários. Os resultados deste trabalho podem ajudar na definição de guias de uso de assistentes inteligentes de programação para diferentes perfis de desenvolvedores.

Palavras-chave

Assistentes Inteligentes de programação, Copiloto, Padrão de Prompt, Estilos Cognitivos de desenvolvedores

1 Introdução

A forma como programadores estão resolvendo tarefas no dia a dia está mudando rapidamente com a popularização dos grandes modelos de linguagem (LLMs). Programadores de diferentes níveis de conhecimento e experiência têm obtido bons resultados com a ajuda de LLMs em soluções para tarefas de compreensão de código, correção de defeitos, autocompletamento de código, depuração e melhoria da segurança do software. É amplamente reconhecido que programadores com maior conhecimento e experiência tendem a obter resultados superiores ao utilizarem essas ferramentas [12, 14].

Na engenharia de software, diversos trabalhos têm discutido os desafios relacionados ao uso de LLMs para apoiar tarefas de engenheiros de software, como o trabalho de [13], focado em testes, qualidade e legibilidade do código gerado; o de [3], focado em segurança e correção de bugs; e o de [10], entre outros.

Neste contexto, o desenvolvimento de "*prompts*" é considerado um elemento fundamental para a eficácia dos resultados obtidos ao utilizar modelos de linguagem, visto que falhas em seu design exigem múltiplas interações para a obtenção de soluções [11]. Além disso, a adoção de padrões reutilizáveis influencia diretamente a eficiência das respostas [20], sugerindo que a experiência do desenvolvedor molda a interação com LLMs. Assim, espera-se que desenvolvedores experientes e novatos interajam de maneira diferente com LLMs ao realizarem uma tarefa, e que os experientes encontrem mais facilidade no ajuste de prompts para alcançar soluções mais ágeis e precisas, graças ao seu maior entendimento do problema e do comportamento do modelo. [5, 21]. Apesar dos esforços de pesquisadores, há poucos estudos voltados a compreender o impacto do perfil cognitivo dos indivíduos e de sua experiência durante a interação com e o uso de inteligência artificial generativa na realização de tarefas de desenvolvimento de software. Nesse sentido, a análise do estilo cognitivo, aliada à experiência no uso de LLMs, pode fundamentar a personalização de futuras interfaces

de programação. Compreender essas facetas permite que as ferramentas auxiliem o desenvolvedor em uma interação mais eficaz e adaptada ao seu perfil, otimizando o fluxo de trabalho.

Assim, o objetivo deste trabalho é entender como diferentes perfis de pessoas desenvolvedoras (definidos pelo nível de experiência e pelo estilo cognitivo) utilizam modelos de linguagem de grande porte na realização de tarefas de programação e de compreensão de código. Busca-se identificar se e como tais características influenciam a elaboração de *prompts* na interação com assistentes de Inteligência Artificial (IA). Entender esta dinâmica pode ajudar na criação de recursos adaptativos nessas ferramentas, habilitando-as a reconhecer padrões comportamentais específicos, como a inserção em massa de trechos de código por programadores iniciantes, e a propor estratégias que mitiguem a carga cognitiva do usuário e otimizem a janela de contexto do modelo.

Para realizar esta análise, três questões de pesquisa são investigadas: **QP1** - Como varia o nível de sofisticação linguística e o esforço de elaboração dos *prompts* durante a interação com LLMs em tarefas de programação?; **QP2** - De que maneira os participantes estruturam suas solicitações à LLM: via questionamento aberto, verificação de hipóteses, instruções diretas ou por comandos implícitos?; **QP3** - Como se comparam os *prompts* de pessoas com diferentes perfis?.

Em resumo, a sofisticação linguística (**QP1**) mantém-se acessível e adaptada ao contexto técnico. Métricas como *Flesch-Kincaid* e *Gunning Fog* indicam estruturas de frases simples, mesmo em tarefas de programação. Embora a inclusão de códigos e jargões reduza a legibilidade formal, a maioria dos prompts permanece de fácil leitura, sem aumento real na complexidade linguística.

Quanto às categorias (**QP2**), a predominância de Perguntas Abertas em quase todas as atividades sugere o uso do assistente para a investigação e a compreensão da base de código, em vez de apenas a geração automática. Essa tendência reforça que, diante de códigos desconhecidos, a prioridade dos participantes é compreender o contexto.

Sobre a comparação entre perfis (**QP3**), a análise estatística confirmou uma associação significativa entre o grupo e o tipo de *prompt* ($p = 0,0309$), evidenciando a influência do estilo cognitivo e da experiência. Usuários TIM elaboraram *prompts* mais concisos (média de 13 palavras), enquanto os ABI foram mais verbosos (média de 20 palavras). A experiência atuou como moderadora: o grupo ABI-Experiente superou a aversão ao risco com instruções diretas, enquanto os novatos ABI adotaram estratégias compensatórias com prompts longos e complexos (agrupamento "Instrução + Implícito"). Já o perfil TIM-Experiente mostrou-se o mais versátil, alternando entre investigação e instrução conforme a demanda.

2 Trabalhos Relacionados

Na interação com LLMs, o design de *prompts* tem impacto crucial na eficácia dos resultados. Esta seção apresenta estudos que analisam a interação entre desenvolvedores e esses modelos.

Tarefas repetitivas ou com soluções aproximadas podem ser resolvidas com o mesmo *prompt*. White et al.[20] destacam a importância de adotar e documentar padrões de *prompts*. Os autores apresentam um catálogo de padrões de *prompt*, aplicados para resolver problemas comuns no domínio da interação conversacional com LLMs e da geração de resultados para a automação de tarefas de *software*. Esses padrões de *prompt* são análogos aos padrões de *software* e visam fornecer soluções reutilizáveis para problemas enfrentados pelos usuários ao interagirem com LLMs.

Fagadau et al., [6] , em estudo experimental controlado com 124.800 combinações de 200 métodos Java, mostraram que incluir um resumo do propósito do método e exemplos de entrada/saída melhora significativamente a corretude do código gerado pelo *GitHub Copilot*, enquanto o tempo verbal não exerce influência.

Mondal et al., [11] identificaram que *prompts* com falhas de design (especificações ausentes, respostas verbosas, contexto ausente) demandam múltiplas interações, e que corrigir essas lacunas permite alcançar o resultado em uma única interação. Análises do repositório DevGPT [22] mostraram que programadores frequentemente refinam seus pedidos iterativamente para melhorar as respostas do ChatGPT.

O trabalho de Liang et al. [9] introduz o conceito de *prompt programming*: em estudo qualitativo com 20 desenvolvedores, identificaram 14 comportamentos específicos, notando que os programadores desenvolvem modelos mentais do comportamento do modelo de linguagem (FM) em vez de modelos mentais de código — processo que permanece rápido, porém pouco sistemático mesmo após dezenas de *prompts*.

Em estudo empírico com 3.044 trechos de código do dataset DevGPT [22], Grewal et al.,[7] observaram que 54% das linhas geradas pelo ChatGPT aparecem em projetos do *GitHub*, permanecendo inalteradas por média de 89 dias; das alterações posteriores, 82% são modificações (36% de baixo impacto funcional, 33% reorganização, 13% renomeação de variáveis) e 18% exclusões, geralmente em menos de 24h após a submissão.

Shah et al., [17], ao analisarem estudantes avançados com *GitHub Copilot*, identificaram um padrão de *one-shot prompting* — solicitação da funcionalidade completa em um único *prompt*, seguida de depuração — com maior confiança relatada na compreensão do que na geração de código. Em perspectiva organizacional, Stray et al., [18] mostraram, via entrevistas e o framework HACAF, que desenvolvedores mais experientes são mais cautelosos na adoção de LLMs, enquanto os menos experientes as veem como necessárias.

Em resumo, os trabalhos citados discutem diversas abordagens e desafios relacionados à criação de *prompts* e à interação entre desenvolvedores e modelos de linguagem. Entretanto, ainda são necessários estudos mais aprofundados para compreender melhor como os desenvolvedores formulam seus *prompts* e como os perfis cognitivos existentes entre eles afetam essa interação na resolução de tarefas de codificação.

3 Metodologia de Pesquisa

Este estudo visa investigar a interação entre desenvolvedores e grandes modelos de linguagem (LLMs) na resolução de tarefas de programação. O foco central consiste em analisar se a experiência técnica e o estilo cognitivo — avaliados pelas facetas do GenderMag, conforme abordado por Santos et al. [16] — influenciam a elaboração de *prompts* em assistentes de codificação baseados em IA, como o *GitHub Copilot*.

Para tanto, a metodologia a seguir descreve o cenário do estudo, as tarefas de programação e de compreensão de código, os procedimentos de coleta de dados das interações com o *GitHub Copilot* e as técnicas de análise adotadas.

3.1 Recrutamento e Coleta de Dados do Perfil dos Desenvolvedores

No total, 27 participantes, que trabalham na indústria ou são estudantes de diferentes anos de graduação, participaram do estudo. A estratégia de recrutamento dos participantes foi realizada por meio de: (i) apresentações curtas em aulas; (ii) grupos de mensagens em programas de estudo; (iii) contatos pessoais e profissionais; e (iv) divulgação em redes sociais, como LinkedIn, Facebook e Twitter.

Todos os participantes foram voluntários e assinaram um termo de consentimento antes das sessões. O único pré-requisito era ter experiência básica em programação.

Antes da realização do experimento, cada participante respondeu a dois questionários. O primeiro instrumento consistiu em um questionário sociodemográfico para coleta de dados sobre a experiência do participante. Por exemplo, neste questionário, perguntamos aos participantes: *Qual sua Escolaridade? Sua graduação em programação? Como você avalia sua proficiência em programação geral:programação em JavaScript? Desenvolvimento web?; React? Visual Studio Code?*. O segundo questionário teve como objetivo mapear o estilo cognitivo dos participantes, utilizando a escala GenderMag, [4]. Esse método utiliza 5 características de diferença de gênero, chamadas facetas. Estes dados foram obtidos por meio de um questionário sobre estilos cognitivos.

Essas 5 facetas foram encapsuladas em personas [19]: Timothy / Timara, Patricia / Patrick e Abigail / Abishek, resumidas nesse trabalho para TIM e ABI:

- **ABI**: Prefere interações que forneçam explicações detalhadas e promovam um estilo exploratório, como "Explique a lógica deste código em detalhes."
- **TIM:** Prefere que as interações sejam mais rápidas e objetivas, como "Encontre o erro neste trecho de código."

Para determinar o perfil correspondente à persona (ABI ou TIM), cada participante respondeu ao questionário de aspectos do GenderMag, [8]. O questionário de aspectos do GenderMag, (Figura 1). O Gendermag é um instrumento baseado na escala *Likert* que coleta valores específicos de cada respondente para os cinco aspectos: *Autoeficácia*; *Motivação*; *Estilo de aprendizagem*; *Processamento de Informação* e *Atitude em relação ao risco* [8]. Optamos por classificar o estilo participante dos estudos com o GenderMag, uma vez que foi adaptado ao contexto da Ciência da Computação [8] e é amplamente utilizado em estudos de engenharia de software.

Após os participantes responderem ao questionário de aspectos, pontuamos as respostas de acordo com os passos a seguir:

- **Passo 1 (Complemento):** Converter as respostas em pontuações numéricas de 1 (Discordo completamente) a 9 (Concordo completamente). Para algumas perguntas, valores mais próximos de 9 indicam maior similaridade com TIM. Porém, o contrário é verdadeiro para as perguntas 2 e para as de 9 a 13. Para essas perguntas, inverta as respostas dos participantes para o complemento em base 10 (ou seja, converta “9” em “1”, “8” em “2” e assim por diante).

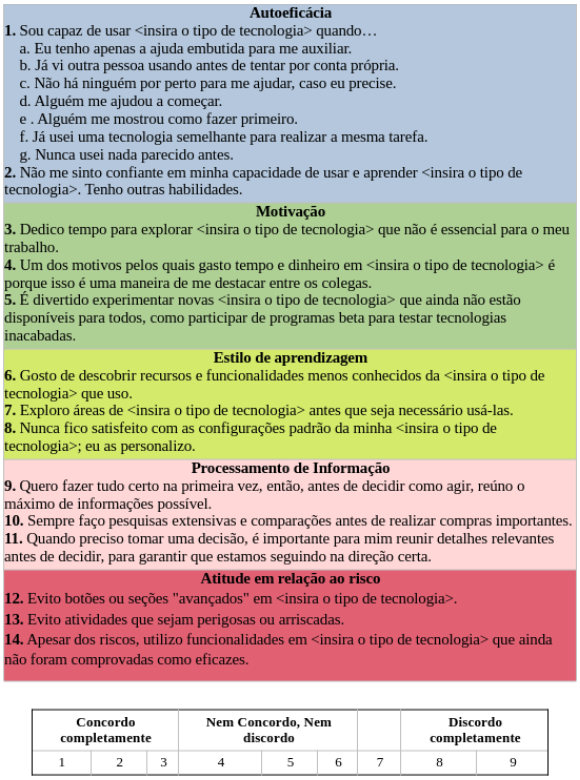

Figura 1: O questionário de aspectos. Todas as perguntas utilizam uma escala Likert de 9 pontos.

- **Passo 2 (Soma de cada aspecto):** Para cada participante, serão somados os resultados do Passo 1 de cada aspecto. Este passo gera 5 pontuações por participante, uma para cada aspecto.
- **Passo 3 (Calcular medianas dos aspectos):** Para identificar o valor mediano do grupo de pares (assumindo que o grupo de pares corresponde aos participantes recrutados para o estudo), calculamos a mediana da soma das pontuações de todos os participantes do mesmo grupo de pares, para cada aspecto.
- **Passo 4 (Marcar a pontuação de cada participante para cada aspecto):** À direita da mediana (acima) é considerado "TIM", caso contrário,"ABI".

Ao final, cada participante terá uma marcação de 5 elementos, cada um correspondente a um aspecto.

3.2 Tarefas

Foi utilizado o fork do projeto todoMvc[1] como base do código. Este projeto utiliza a linguagem de programação JavaScript, mais especificamente *React*. Modificamos o projeto e introduzimos alguns defeitos, como erros de importação nas classes e implementações incompletas de funcionalidades.

[1](http://todomvc.com)

Ao todo, cada participante deveria realizar 6 tarefas. A cada tarefa, o grau de complexidade aumenta. Abaixo estão detalhados as tarefas e os erros incluídos no projeto.

- Tarefa 1 - A - Alterar a cor. Quando não há afazeres, aparece uma caixa vermelha com a mensagem "Nenhum afazer aqui ainda". Altere a cor dessa caixa para azul (#007bff).
- Tarefa 1 - B - Alterar o placeholder. O texto do 'placeholder' no campo de entrada atualmente exibe a pergunta "O que precisa ser feito?". Altere o texto para "Digite um afazer aqui".
- Tarefa 1 - C - Ordenar afazeres. Atualmente, os afazeres são exibidos na ordem em que foram adicionados. Modifique o aplicativo para que os afazeres sejam ordenados alfabeticamente.

3.2.1 Tarefa 2 - A - Salvar dados. Atualmente, todos os afazeres desaparecem quando a página é recarregada. Modifique a aplicação para que os afazeres permaneçam mesmo após um recarregamento. Certifique-se de que o estado de cada afazer (se está concluído ou ativo) também seja mantido. Uma tentativa de implementar essa funcionalidade foi feita, mas, no momento, ela não está funcionando corretamente. Corrija ou reimplemente a funcionalidade.

3.2.2 Tarefa 2 - B - Salvar rascunhos. Se um usuário recarregar a página enquanto estiver digitando um novo afazer, o texto no campo de entrada será apagado. Modifique a aplicação para que o campo de entrada salve automaticamente os rascunhos a cada poucos segundos e os restaure ao recarregar a página.

3.2.3 Tarefa 2 - C - Paginação. Atualmente, a aplicação de Lista de Afazeres exibe todas as tarefas em uma única lista. Nesta tarefa, você implementará a paginação para limitar o número de afazeres exibidos por página, permitindo que os usuários naveguem facilmente entre as páginas.

3.2.4 Erros adicionados no projeto.

- No arquivo main.jsx adicionado o erro de importação na linha 6;
- No arquivo reducer.js adicionado erro de nome de atributo no *switch*, linha 44, causando o não carregamento das tarefas salvas;
- No arquivo paginationControls.jsx, linha 13, deletada a condição do if que trata da função de retroceder à página;
- No arquivo input.js foi adicionada uma verificação para que o afazer só seja adicionado na lista se o seu nome contiver mais de 2 caracteres, linha 18.

3.3 Ambiente do Experimento

Cada sessão do experimento ocorreu por meio de videochamada no Microsoft Teams, na qual foi disponibilizado um ambiente controlado para a resolução das tarefas. Antes de cada sessão, foi configurado um ambiente de desenvolvimento na plataforma *CodeSpace*, contendo o editor de código VSCode, com o modelo GPT-4o mini definido para o *GitHub Copilot*. Os passos de configuração do VSCode eram apresentados aos usuários para que todas as extensões e as configurações do ambiente fossem iguais entre os participantes.

Antes de iniciar, o participante era orientado a desligar sua câmera e um dos autores verificava se os questionários iniciais haviam sido preenchidos. O experimento iniciava após as orientações e a validação das configurações do VSCode. Durante a sessão, o participante compartilhava sua tela, e a gravação, incluindo a transcrição automática pelo Teams, era iniciada.

Ao final do experimento, solicitava-se que o participante enviasse a gravação do log de sua interação com o GitHub Copilot, e o experimento era encerrado se o participante concluísse todas as tarefas em até 40 minutos ou atingisse esse tempo limite.

Antes do experimento, realizaram-se três testes-piloto com graduados em Ciência da Computação com 2 a 5 anos de experiência. No primeiro teste, observou-se que o Copilot antecipava as tarefas no ambiente de desenvolvimento, comprometendo a interação entre o usuário e o sistema. Para mitigar isso, as especificações foram encaminhadas para links externos, forçando o participante a ler a tarefa e a elaborar o *prompt*.

3.4 Análise dos Dados

Para avaliar a escrita na interação com a LLM (**QP1 - Como variam o nível de sofisticação linguística e o esforço de elaboração dos *prompt* ao longo da interação com LLMs em tarefas de programação?**), utilizou-se a biblioteca *textstat* [1] para o cálculo das métricas detalhadas na Tabela 1. Estas métricas visam identificar variações na estrutura dos *prompts* e sua influência na execução das tarefas. A análise foi realizada por meio de estatística descritiva e de visualizações em *box plots*.

Para responder à segunda questão de pesquisa, **QP2 - De que maneira os participantes estruturam suas solicitações à LLM: via questionamento aberto, verificação de hipóteses, instruções diretas ou por comandos implícitos ?**, realizamos a análise de aspectos *qualitativos* dos prompts. Agrupamos os prompts de acordo com como as perguntas foram formuladas [15].

(1) **Pergunta Aberta (Open Question):** Solicitações que buscam explicações, localizações ou instruções sobre "como fazer". Normalmente iniciadas por pronomes interrogativos como "How", "Where" e "What", essas perguntas sugerem que o usuário considera o LLM como um consultor ou fonte de conhecimento. Ex: "How do I change the color of the message?".
(2) **Instrução (Instruction):** Comandos imperativos em que o usuário atribui uma ação direta ao modelo, aguardando um resultado imediato (como a geração ou a refatoração de código), sem necessidade de solicitar uma explicação do procedimento. Ex: "Sort this list." ou "Fix the code below.".
(3) **Hipótese (Hypothesis):** Interações em que o usuário apresenta uma solução ou compreensão prévia e pede a LLM que a valide ou corrija algo. Ex: "Is the error caused by the missing variable X?".
(4) **Implícito (Implicit):** Prompts que não apresentam uma pergunta ou instrução explícita, mas apenas trechos de código ou contexto, aguardando que a LLM deduza a necessidade de ajuda (normalmente "complete" ou "explique").

Para realizar esse agrupamento, dois pesquisadores conduziram uma análise manual e individual de cada prompt conjuntamente, discutindo e resolvendo eventuais divergências durante o processo.

Para a questão de pesquisa três (**QP3 - Como se comparam os prompts de pessoas com diferentes perfis?**), cruzamos os

Tabela 1: Métricas de Texto e Legibilidade

Métrica	Definição	Explicação e Interpretação
(i) Quantidade de palavras	Contagem total de *tokens* (palavras) no texto.	Medida básica. Base para quase todos os cálculos de legibilidade. Textos muito longos podem cansar, muito curtos podem ser superficiais.
(ii) Número de sentenças	Contagem total de frases gramaticais (delimitadas por pontuação como ., ?, !).	Indica a segmentação. Muitas sentenças, com poucas palavras, sugerem frases curtas. Poucas sentenças em um texto longo indicam complexidade sintática.
(iii) Flesch Reading Ease	Índice de 0 a 100 que estima a facilidade de leitura (com base em sílabas e no comprimento da frase).	Quanto maior, mais fácil. 90-100 (Muito fácil); 60-70 (Padrão); 0-30 (Muito difícil/Acadêmico).
(iv) Flesch-Kincaid Grade Level	Conversão do Flesch Reading Ease para o sistema de séries escolares dos EUA.	Indica anos de estudo necessários. Ex: 8.0 significa que um aluno da 8ª série conseguiria ler.
(v) Gunning Fog Index	Estima anos de educação formal necessários para entender o texto na primeira leitura.	Penaliza o uso de "palavras complexas" (3 ou mais sílabas).
(vi) SMOG Index	Estima anos de educação para 100% de compreensão (baseado em polissílabas).	Considerado o "padrão-ouro" para textos de saúde. Mais rigoroso que o Flesch-Kincaid.
(vii) Coleman-Liau Index	Índice de nível escolar baseado em caracteres, não em sílabas.	Usa a média de letras por 100 palavras. Computacionalmente eficiente e preciso.
(viii) Automated Readability Index (ARI)	Índice de nível escolar baseado em caracteres (similar ao índice de Coleman-Liau).	Ajusta os pesos de caracteres por palavra e palavras por frase para aproximar-se da idade de leitura.
(ix) Dale-Chall Readability Score	Métrica baseada na familiaridade do vocabulário (lista de 3.000 palavras comuns).	Calcula a proporção de palavras fora da lista de vocabulário comum. Avalia se o léxico é adequado ao público-alvo.

resultados de QP1 e QP2 com os perfis cognitivos (TIM e ABI) e os níveis de experiência (experientes e novatos). O objetivo foi verificar se há diferenças estatisticamente significativas no estilo de escrita dos *prompts* em função dessas variáveis.

Devido à natureza das métricas (contagem de palavras e índices de legibilidade), optou-se por testes não paramétricos. O teste U de *Mann-Whitney* foi aplicado para comparar as distribuições entre grupos independentes, como os perfis ABI e TIM, por ser adequado a dados sem distribuição normal. Além disso, calculou-se o tamanho do efeito (*d* de *Cohen*) para mensurar a magnitude das diferenças e a relevância prática dos resultados estatísticos.

O algoritmo de aprendizado de máquina não supervisionado *K-Means* foi empregado para investigar padrões de comportamento que poderiam não ser claros apenas com a separação de personas TIM e ABI. Por fim, para verificar a existência de associação entre as variáveis qualitativas — especificamente entre os grupos combinados (Estilo Cognitivo + Experiência) e a categoria de *prompt* utilizada (Pergunta Aberta, Instrução, Hipótese ou Implícito) — foi aplicado o teste *Chi-Square*.

3.5 Características dos Participantes

A amostra final compreende 27 indivíduos de diferentes nacionalidades. Quanto ao gênero, há predominância masculina (81, 5%; $n = 22$), seguida pelo feminino (14, 8%; $n = 4$) e por um participante (3, 7%) que optou por não informar. Geograficamente, o grupo é diversificado, com representantes de dez países: Brasil ($n = 13$), Holanda ($n = 5$), Romênia ($n = 2$) e participações unitárias da Rússia, do Reino Unido, da Índia, do Japão, da China, da Estônia e da Itália.

A situação profissional é heterogênea, composta por trabalhadores em tempo integral (40, 7%; $n = 11$), estudantes (33, 3%; $n = 9$), trabalhadores de meio período (14, 8%; $n = 4$), autônomos (7, 4%; $n = 2$) e um desempregado (3, 7%). Quanto ao idioma dos *prompts*, 55, 6% ($n = 15$) utilizaram o português e 44, 4% ($n = 12$) o inglês. Para garantir a uniformidade da análise, todas as interações em português foram traduzidas para o inglês.

4 Resultados

4.1 QP1 - Como Varia o Esforço de Elaboração de Prompts?

Para responder a esta questão de pesquisa, foram analisados 257 *prompts* coletados no estudo. Em geral, os *prompts* nas 6 tarefas apresentaram, em média, **19,80 palavras** por prompt. A tarefa 2-C apresentou a maior média, com 52,62 palavras, resultado esperado, já que pode ser considerada a mais complexa do estudo.

A maioria dos *prompts* é curta (entre 8 e 23 palavras), mas há também *prompts* muito extensos (até 163 palavras). Esta variação é o efeito de alguns participantes que utilizaram o enunciado da tarefa como *prompt*, por exemplo, o participante **P3**, na tarefa 2 - C.

A Figura 2 apresenta o boxplot da variação na extensão dos *prompts* (medida em número de palavras) submetidos pelos participantes em cada uma das seis tarefas analisadas (1-A a 2-C). O eixo horizontal representa as tarefas, enquanto o eixo vertical quantifica o número de palavras utilizadas nas interações. As tarefas introdutórias, especificamente as Tarefas 1-A e 1-B, apresentam as menores medianas e uma dispersão interquartil reduzida, indicando que a maioria dos participantes utilizou *prompts* curtos e diretos, com pouca variação entre si.

Ao abordar as tarefas subsequentes, especialmente nas séries 2-A, 2-B e 2-C, observa-se um aumento da mediana e maior dispersão dos dados, incluindo a presença de valores isolados (*outliers*), o que indica que, diante da maior complexidade das tarefas, os participantes tendem a adotar *prompts* com descrições mais longas.

Os resultados deste estudo indicam que a complexidade da tarefa sustenta a hipótese de que a interação com o LLM é adaptativa: a carga cognitiva e o esforço de formulação do *prompts*, considerando o número de palavras, aumentam proporcionalmente à complexidade percebida da tarefa.

Para aprofundar a análise, consideramos um conjunto de métricas que quantificam a complexidade dos prompts escritos. Para isso, utilizamos a métrica que determina o nível de escolaridade (*Grade Levels*) do *prompt*. Como estas métricas podem variar na sua interpretação, usamos 5 algoritmos distintos para verificar se há consenso quanto à complexidade do *prompt*. Como as métricas

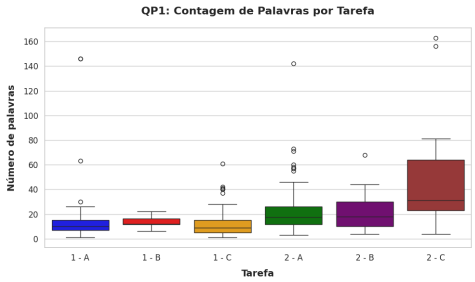

Figura 2: Quantidade de palavras por tarefa

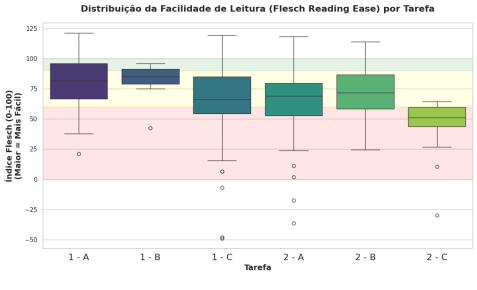

Figura 3: Distribuição da facilidade de leitura

`gunning_fog` e `smog_index` usam fórmulas diferentes (algumas atribuem mais peso às sílabas complexas, outras às frases longas), colocá-las lado a lado nos ajudou a verificar se o texto é consistentemente complexo.

A análise visual dos boxplots sugere que a linguagem empregada na interação humano-LLM durante as tarefas apresenta menor complexidade sintática, conforme Figura 3. Essa simplicidade na linguagem indica que os participantes priorizam a eficácia e a rapidez na comunicação, em detrimento da formalidade gramatical. Embora a tarefa seja técnica e introduza jargões específicos (capturados por métricas sensíveis a polissílabos, como Gunning Fog ou SMOG), a maioria das frases permanece direta e compreensível.

A distribuição dos dados mostra que, de forma consistente em todas as tarefas, a facilidade de leitura permanece majoritariamente alta, situando-se geralmente na faixa de "Leitura Fácil" a "Muito Fácil" (pontuações entre 70 e 90+). No entanto, nota-se uma leve queda no desempenho em tarefas que exigem a manipulação direta de trechos de código, como as Tarefas 2-A e 2-C. Em situações específicas como essas, a inclusão de trechos de código ou termos técnicos no *prompt* geralmente reduz temporariamente a facilidade de uso e pesa na diminuição do valor da métrica de facilidade de leitura, especialmente na tarefa 2-C.

4.2 QP2 - De Que Maneira os Participantes Estruturam Suas Solicitações à Llm?

A segunda pergunta de pesquisa (QP2) explora a natureza qualitativa das interações. Para abordar essa questão, 257 *prompts* foram classificados por dois pesquisadores (autor e orientador). Assim, cada *prompt* foi lido e classificado conjuntamente, com discussões até o consenso.

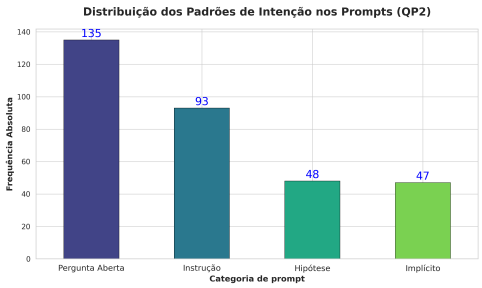

Figura 4: Distribuição dos padrões de prompts

A Figura 4 mostra a distribuição de frequência dos padrões ao longo das tarefas, indicando a incidência de cada tipo de *prompt* nas interações dos 27 participantes.

A análise dos dados revela uma grande desigualdade nas estratégias de interação adotadas pelos participantes. É possível perceber a predominância da categoria **"Pergunta Aberta"**, o que indica que, ao lidar com as tarefas propostas, os participantes, por não terem conhecimento da base de código, possivelmente recorrem a essa estratégia de *prompt* para explorar e compreender o código, em vez de se restringirem à geração automática de código. Os participantes empregaram o LLM principalmente como um instrumento de navegação e assistência.

A segunda estratégia mais usada foi a de fornecer instruções diretas, com comandos imperativos, empregados somente quando o usuário já tinha um entendimento claro do que deveria ser feito, com o objetivo de auxiliar na implementação e, possivelmente, impactar a produtividade da tarefa.

Por fim, é possível observar baixa utilização dos padrões de hipótese e implícitos, o que sugere que os participantes pouco recorreram ao Copiloto para confirmar uma possível solução que haviam considerado. Ao contrário disso, os participantes delegaram a construção da solução desde o início.

A Figura 5 apresenta um gráfico de barras empilhado que mostra o número de tipos de *prompts* por tarefa. O eixo vertical representa o total de *prompts*, permitindo comparar o volume total de interações e a proporção de cada tipo de *prompt* em relação à tarefa específica.

Os resultados indicam que a "Pergunta Aberta" (em azul) prevalece em quase todas as tarefas. Isso reforça a conclusão de que, independentemente do tipo específico do problema — seja a simples tarefa de trocar a cor de um elemento de uma página web (1-A) ou a correção de um defeito (2-A), a principal estratégia dos participantes foi questionar o Copilot para promover a compreensão, em vez de simplesmente ordenar ações. Nota-se que a categoria "Instrução" (verde-escuro) foi o segundo padrão mais utilizado pelos participantes; entretanto, foi o tipo de *prompt* mais utilizado na tarefa de maior complexidade **2-C**. Entre as tarefas simples e as mais complexas, o padrão **"Hipótese"** foi o menos utilizado.

Uma análise mais granular dos dados identificou casos com múltiplas classificações, apresentados na Tabela 2. Cerca de 64 dos 257 prompts (24,9%) possuem dois ou mais padrões. A existência de prompts híbridos sugere que os desenvolvedores adotam estratégias para garantir que o LLM compreenda o contexto da tarefa. A combinação mais frequente: **Pergunta aberta + Instrução**. Ao

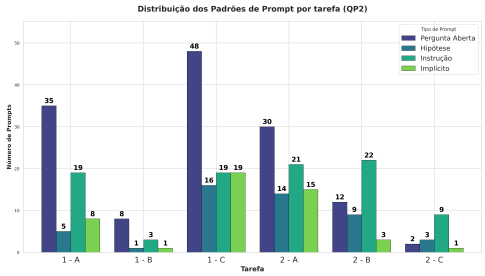

Figura 5: Distribuição dos Padrões de prompt por tarefa

criar um prompt que inclui uma pergunta ("Como faço X?") sobre um comando ("Gere o código para X"), o desenvolvedor mostra que seu objetivo é duplo: procura compreender o conceito (aprender) e, ao mesmo tempo, buscar uma solução prática (automatizar).

Tabela 2: Prompts multi padrões

Combinação	Total
Pergunta + Instrução	18
Pergunta + Implícito	12
Pergunta + Hipótese	11
Hipótese + Instrução	10
Hipótese + Implícito	7
Instrução + Implícito	3
Pergunta + Instrução + Implícito	2
Hipótese + Instrução + Implícito	1

Embora a categoria "Hipótese" tenha sido pouco usada isoladamente, encontramos casos em que os participantes realizaram combinações como Pergunta + Hipótese (n=11) e Hipótese + Instrução (n=10), o que sugere que os participantes geralmente empregam suas hipóteses não apenas para validação, mas também como contexto para direcionar a resposta da LLM. Outro fato é a recorrência da combinação **Pergunta + Implícito**. Este dado reflete o fluxo de trabalho típico de manutenção de código: o desenvolvedor fornece um fragmento de código ou um *log* de erro (Implícito) e acopla uma dúvida explícita (Pergunta).

4.3 QP3 - Como se comparam os prompts de pessoas com diferentes perfis?

A terceira pergunta de pesquisa (QP3) analisa, de forma qualitativa, os *prompts* dos participantes com estilos cognitivos distintos.

Para responder a esta questão de pesquisa, analisamos as métricas quantitativas de duas formas. Primeiro, comparamos as métricas de Facilidade de Leitura, o Nível de Escolaridade da estrutura textual estimada e o número de palavras, considerando apenas o estilo cognitivo (TIM e ABI), sem levar em conta a experiência dos participantes. Na segunda análise, comparamos o estilo cognitivo à experiência, formando quatro grupos: TIM - Experiente, ABI - Experiente, TIM - Não Experiente e ABI - Não Experiente; neste caso, a experiência é definida como 4 ou mais anos de atuação no desenvolvimento de software.

A Tabela 3 apresenta a distribuição do número de participantes e de *prompts* realizadas durante as seis tarefas executadas pelos participantes.

Tabela 3: Distribuição do número de participantes e prompts

Grupo	Particip.	IDs dos Particip.	Prompts
ABI (Exp.)	8	P04, P05, P06, P12, P16, P26, P27, P28	74
ABI (Não Exp.)	7	P07, P15, P17, P18, P19, P20, P22	76
TIM (Exp.)	7	P01, P03, P08, P10, P11, P13, P14	64
TIM (Não Exp.)	5	P02, P09, P21, P23, P24	43

A Figura 6a mostra que os participantes do TIM apresentam pontuação média maior (74,6) do que os do ABI (66,8) na métrica de facilidade de leitura. Contudo, o teste U de *Mann-Whitney* indica que a diferença não é significativa (*p-value* = 0.35). Essa ausência de poder estatístico provavelmente está relacionada ao número de participantes em cada grupo. No entanto, observa-se que a diferença na média entre os dois grupos é evidenciada pela presença de *outliers* no grupo ABI.

Da mesma forma, na Figura 6b, a persona ABI apresenta um nível de escolaridade (*Flesch-Kincaid Grade*) exigido ligeiramente maior (6,6) do que o TIM (5,8), sugerindo que ABI utiliza uma linguagem um pouco mais complexa ou técnica, conforme a Figura 6b, mas não há diferença estatística entre as personas nesta métrica (*p-value* = 0.64239).

Para aprofundar o entendimento do padrão de escrita dos *prompts* das personas, analisamos o tamanho dos *prompts*, expresso em número de palavras, na Figura 6c. A persona ABI tende a ser um pouco mais verbosa (média de 21,93 palavras) do que a TIM (média de 12 palavras), embora a diferença não seja estatisticamente significativa (*p-value* = 0.38958); o teste de tamanho de efeito indica uma pequena diferença estatisticamente significativa (*Cohen's d* = 0.303). De forma mais detalhada, observamos que a diferença ocorre entre os grupos de ABI - Experiente vs TIM - Não Experiente (*Cohen's d* = 0,26), conforme apresentado na Figura 7.

Como não observamos diferença estatisticamente significativa, realizamos uma análise para verificar se os participantes apresentaram comportamentos distintos em alguma das 6 tarefas. A média de palavras empregadas nos *prompts* por tarefa difere entre as personas ABI e TIM e é apresentada na Tabela 4.

Ao analisar a relação entre a realização da tarefa e os grupos e as médias de *prompts* e de palavras, observamos que, nas tarefas 1-A e 1-B, todos os grupos obtiveram 100% de sucesso ao completá-las. O grupo ABI - Experiente usou 94 palavras em média, enquanto os demais grupos usaram entre 14 e 35 palavras.

Na tarefa 1-C, observa-se que o grupo ABI utilizou mais *prompts* do que o grupo TIM. O Grupo ABI - Não Experiente precisou de média de 4,43 *prompts* (mais do que o dobro dos TIMs), consequentemente, também apresentou o maior número médio de palavras. Interessante observar que o grupo ABI - Não Experiente teve a menor taxa de conclusão dos participantes (71,43%), enquanto o TIM - Não Experiente teve (80%) e os grupos experientes de ABI e TIM concluíram com 100% de conclusão.

Nas tarefas da série 2, observa-se que o percentual de sucesso diminui em todos os grupos. Por exemplo, na tarefa 2-A, ABI - Experiente e TIM - Experiente têm as melhores taxas de conclusão, 87,5% e 71,4%. Os grupos não experientes obtiveram 42,85% (ABI), 60% (TIM). Interessante observar que o número de *prompts* da ABI - Não Experiente foi o maior entre os grupos e que a média de

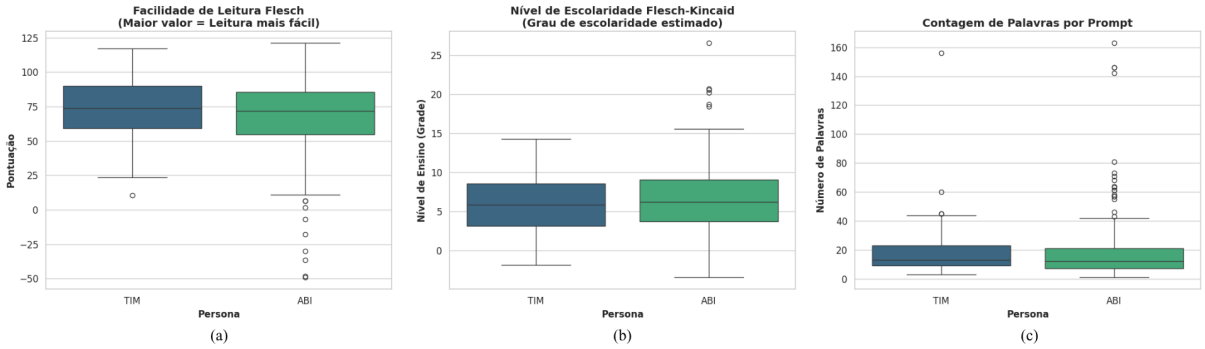

Figura 6: (a) Facilidade de leitura. (b) Flesch-Kincaid Grade. (c) Contagem de palavras por prompt.

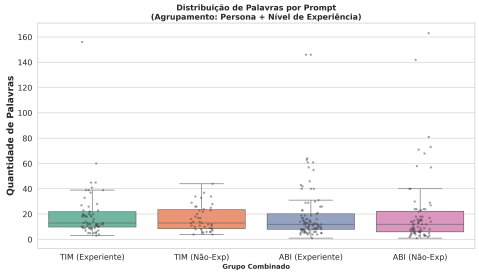

Figura 7: Distribuição de palavras por prompt (persona + experiência)

Tabela 4: Métricas de interação por tarefa

Grupos de Participantes	Tarefas	Média de prompts por tarefa	Média de palavras por tarefa	Concluintes
ABI (Exp.)	1-A	3.00	94.17	8
ABI (Exp.)	1-B	1.50	18.00	8
ABI (Exp.)	1-C	2.88	40.00	8
ABI (Exp.)	2-A	2.00	37.25	7
ABI (Exp.)	2-B	2.50	44.25	4
ABI (Exp.)	2-C	2.00	83.00	3
ABI (Não Exp.)	1-A	3.4	35.50	7
ABI (Não Exp.)	1-B	–	–	7
ABI (Não Exp.)	1-C	4.43	43.71	5
ABI (Não Exp.)	2-A	3.6	121.4	3
ABI (Não Exp.)	2-B	2.0	66.33	2
ABI (Não Exp.)	2-C	1.33	90.33	2
TIM (Exp.)	1-A	2.00	21.60	7
TIM (Exp.)	1-B	1.50	20.00	7
TIM (Exp.)	1-C	2.00	30.00	7
TIM (Exp.)	2-A	3.83	82.50	5
TIM (Exp.)	2-B	3.25	63.00	3
TIM (Exp.)	2-C	1.00	156.00	1
TIM (Não Exp.)	1-A	2.00	18.25	5
TIM (Não Exp.)	1-B	1.67	24.33	5
TIM (Não Exp.)	1-C	2.20	31.20	4
TIM (Não Exp.)	2-A	1.75	34.00	3
TIM (Não Exp.)	2-B	2.67	61.33	2
TIM (Não Exp.)	2-C	2.00	45.50	2

palavras também foi a maior. Ao contrário, os indivíduos do grupo ABI - Experiente apresentaram a menor quantidade de *prompts* em comparação com os do grupo TIM - Experiente.

Em relação à tarefa 2-B o grupo TIM - Experiente obteve a maior média de iterações (3,25 prompts) e palavras (63) para atingir apenas 42,9% de sucesso. Em contraste, o grupo ABI - Não Experiente apresentou a menor média de *prompts* (2,0) e o menor sucesso (25,57%).

A tarefa 2-C apresentou a maior discrepância entre os grupos. O grupo TIM - Experiente obteve apenas 14,3% utilizando a média de 1,00 *prompts* , com o maior volume textual (156 palavras). Já os grupos ABI (Não Experiente e Experiente) apresentaram desempenho entre 37,5% à 28,5% e idêntico entre si, realizando o dobro de interações (2,00 *prompts*) com textos longos (83-90 palavras), o que demonstra que, nesta tarefa complexa, a capacidade de iterar (conversar e ajustar) foi importante para melhorar a taxa de conclusão. O grupo TIM - Não Experiente obteve 40% de sucesso com a menor quantidade média de *prompts* (2,0) e de palavras (45,50).

A análise granular indicou variações na interação com o assistente, o que motivou a realização de um agrupamento para identificar padrões comuns. Para isso, utilizamos o algoritmo *K-Means* para agrupar os *prompts* com base nas métricas analisadas, o que resultou na identificação de quatro grupos (0 a 3). Embora os agrupamentos tenham reunido *prompts* de participantes com perfis cognitivos e experiências distintos, a nossa hipótese era que o algoritmo seria capaz de concentrar perfis semelhantes em grupos majoritários específicos.

Ao fazer a análise dos padrões de comportamento, a segmentação demonstrou a existência de 4 grandes grupos:

(1) **Pergunta Aberta**: O agrupamento 1, com 125 ocorrências, apresenta um comportamento em que os *prompts* são majoritariamente formulados como perguntas. Neste grupo, a legibilidade é elevada (82,1) e a concisão sugere que, na maioria das vezes, os participantes usam o Copiloto como ferramenta de consulta rápida e de baixo esforço cognitivo. Das 125 ocorrências, verificamos que 37 *prompts* pertencem ao grupo ABI - Experiente; 34 ABI - Não Experiente; 37 TIM - Experiente e 17 TIM - Não Experiente
(2) **Instrução**: O Agrupamento 2, com 77 ocorrências, representa a transição da incerteza à ação. A redução da facilidade de leitura para um nível médio (68,6) indica a necessidade de empregar terminologia técnica para fornecer instruções claras sobre como modificar o código. Das 77 ocorrências,

27 *prompts* estão no grupo ABI - Experiente; 12 ABI - Não Experiente; 18 TIM - Experiente e 20 TIM - Não Experiente

(3) **Implícito**: O Agrupamento 0, com 47 ocorrências, isola uma ação específica de"debug". A legibilidade diminui significativamente (40,1) não devido a uma má redação, mas à alta concentração de jargão técnico nos trechos de código colados. Este agrupamento mostra que, quando ocorrem erros, os usuários optam por manter a descrição em linguagem natural e por recorrer à evidência direta (o código). Das 47 ocorrências analisadas, observou-se que 7 *prompts* pertencem ao grupo ABI – Experiente, 26 ao grupo ABI – Não Experiente, 8 ao grupo TIM – Experiente e 6 ao grupo TIM – Não Experiente.

(4) **Instrução + implícito:** A característica que define o agrupamento 3, com 8 ocorrências, é a extensão significativa dos *prompts*. Em comparação com a média geral de $\tilde{2}0$ palavras, este grupo apresenta uma média de **118 palavras**, o que caracteriza textos mais longos. A classificação evidencia uma combinação de **Instrução** e **Implícito** (código colado), indicando que o participante busca, simultaneamente, fornecer diretrizes e inserir código. A baixa legibilidade (pontuação *Flesch* 28,4) indica um texto difícil, denso e com prováveis problemas na estrutura sintática. Das 8 ocorrências, 3 *prompts* pertencem ao grupo cognitivo ABI - Experiente; 4 ao grupo ABI - Não Experiente; e 1, ao grupo TIM - Experiente.

Por fim, para concluir a análise da QP3, verificou-se a associação entre o estilo cognitivo e o tipo de *prompt*, conforme a Tabela 5. Utilizou-se o teste *chi-square* (X^2), uma técnica não paramétrica adequada para verificar associações entre variáveis categóricas, ao comparar as frequências observadas às esperadas sob a hipótese de independência.

Tabela 5: Tipo de *prompts* por persona

Grupo	Pergunta	Hipótese	Instrução	Implícito
ABI (Exp.)	36	13	32	11
ABI (Não Exp.)	40	15	17	23
TIM (Exp.)	37	11	23	9
TIM (Não Exp.)	22	9	21	4

O teste revelou uma associação estatisticamente significativa entre o Grupo e o Tipo de *prompt* ($X^2 = 17,35$; GL = 9; p = 0.0380). Isso confirma que o padrão de escolha de interações depende do perfil do usuário, indicando que as personas adotam estratégias distintas de *prompts* para a resolução de tarefas, mesmo que as métricas globais sejam quantitativamente semelhantes.

O grupo ABI - Experiente utilizou mais instruções diretas do que o esperado estatisticamente (32 observadas vs. 26 esperadas). O comportamento alinha-se à literatura sobre usuários experientes, que tendem a formular comandos claros e orientados à ação, em vez de fazer perguntas exploratórias. Curiosamente, este resultado indica que os participantes do ABI foram menos avessos ao risco — um comportamento esperado de uma pessoa com este estilo cognitivo.

O grupo ABI - Não Experiente apresentou comportamento oposto. O uso de instruções foi abaixo do esperado (17 observados vs. 27,35 esperados), enquanto o de perguntas abertas foi acima do esperado (40 observados vs. 39,71 esperados), o que sugere uma possível falta de experiência e um estilo cognitivo da persona ABI.

O grupo TIM - Não Experiente não seguiu o padrão esperado para novatos. Eles utilizaram mais instruções diretas do que o esperado (21 observadas vs. 16,12 esperadas). Entretanto, este é um comportamento esperado entre pessoas com o estilo cognitivo TIM. Os participantes do grupo TIM - Experiente apresentaram um perfil equilibrado quanto aos tipos de *prompt*, ora investigativo, ora instrucional, ora baseado em hipótese, o que os torna mais equilibrados em comparação aos demais perfis cognitivos.

5 Discussões

Esforço linguístico e cognitivo na interação (QP1). Os resultados sugerem que a interação entre desenvolvedores e LLMs adapta-se à complexidade da tarefa, observando-se uma relação direta entre a complexidade da tarefa e a verbosidade dos *prompts*. Em tarefas iniciais (1-A, 1-B), os participantes foram objetivos; contudo, em tarefas complexas, como 2-A (Persistência) e 2-C (Paginação), houve aumento significativo na contagem de palavras e na dispersão das informações. Isso indica que desafios maiores exigem mais contexto para obter respostas precisas, o que aumenta o esforço de elaboração. No entanto, é evidente que, embora os *prompts* tenham crescido em tamanho, a sofisticação linguística permaneceu em nível escolar (8ª/9ª série), indicando preferência por uma comunicação prática, independentemente da experiência. Assim, a queda no índice Flesch em tarefas complexas deveu-se à inclusão de termos técnicos e de código, não a uma linguagem mais elaborada.

Participantes usaram o copiloto como tutor (QP2). A análise qualitativa revelou a prevalência de "Perguntas Abertas" em relação a "Instruções" ou "Hipóteses", sugerindo que o Copilot é usado como tutor para compreender uma base de código desconhecida (projeto *TodoMVC*) e não apenas como um gerador automático. Outro fato importante a ser observado é a pouca utilização isolada da categoria "Hipótese". Isso pode indicar que os desenvolvedores preferem confiar o raciocínio inicial ao copiloto, em vez de formular suas próprias suposições para validação. No entanto, a presença de *prompts* híbridos (por exemplo, Pergunta + Instrução ou Pergunta + Implícito) sugere uma tática de aprimoramento: o usuário procura, inicialmente, compreender o contexto ("Como faço X?") para, na mesma interação, solicitar a execução ("...e crie o código para isso"). A frequência da combinação "Pergunta + Implícito" indica um processo comum de manutenção e depuração, no qual o código implícito atua como uma validação, enquanto a pergunta natural orienta o copiloto na interpretação desse código.

A Influência do estilo cognitivo e da experiência (QP3). Os testes estatísticos confirmam que a interação é influenciada pelo estilo cognitivo e moderada pela experiência ($p = 0,0380$). O grupo ABI - Experiente superou a característica de aversão ao risco, utilizando mais instruções diretas do que o previsto. O grupo ABI - Não Experiente compensou a falta de vocabulário técnico com *prompts* longos e densos (Instrução + Implícito). Já o perfil TIM manteve-se objetivo. O TIM - Não Experiente priorizou instruções diretas, enquanto o TIM - Experiente foi mais equilibrado. Contudo, em tarefas complexas (2-C), a brevidade excessiva do TIM-Experiente resultou em menor sucesso (14,3%) em comparação à abordagem iterativa dos grupos ABI (37,5%), o que demonstra que a capacidade

de "conversar" com o modelo pode influenciar o sucesso em desafios mais complexos.

Implicações para os desenvolvedores. Os resultados sugerem que a eficiência no uso de IAs evolui com a experiência técnica. A ocorrência de *prompts* híbridos, como "Pergunta + Implícito" (12 ocorrências) e "Instrução + Implícito", indica que a inclusão de trechos de código é uma prática comum na depuração. No entanto, o Agrupamento 3 mostrou que iniciantes (especialmente ABI) tendem a criar *prompts* excessivamente longos (média de 118 palavras) e de baixa legibilidade ao tentarem fornecer contexto. Nesse sentido, os desenvolvedores devem considerar as LLMs como instrumentos para navegar e aprender a partir da base de código, e não apenas para gerar código. Essa abordagem foi a principal estratégia para lidar com a falta de conhecimento sobre o projeto.

Implicações para a pesquisa. A confirmação de que o estilo de escrita e a escolha dos *prompts* dependem do perfil ($X^2 = 17,35$) implica que pesquisas futuras não devem tratar os desenvolvedores de forma única nas análises. A distinção entre personas ABI (mais verbosas, com média de 20 palavras) e TIM (mais concisas, com média de 13 palavras), moderada pela experiência, é essencial para evitar vieses. O baixo sucesso do grupo TIM - Experiente em tarefas complexas (14,3%) devido a *prompts* curtos em comparação aos grupos ABI (42,85%) sugere que novos estudos devem investigar como a capacidade de iteração e o padrão de *prompt* impactam diretamente a solução de problemas

Implicações para o desenvolvimento de ferramentas de LLM. A predominância de "Perguntas Abertas" (125 ocorrências no Agrupamento 1) sugere que as ferramentas devem focar em interfaces de consulta e de esclarecimento. A interface não deve apenas simplificar a geração de código ("Instrução"), mas também oferecer suporte sólido ao fluxo de "consultor" ou "tutor" que os usuários tendem a seguir. A identificação do Agrupamento 3 (iniciantes com *prompts* longos e complexos) sinaliza a necessidade de recursos que ajudem a organizar o contexto. Ferramentas poderiam detectar colagens excessivas de código e sugerir métodos mais eficientes de referenciar funções ou arquivos, otimizando a interação com usuários menos experientes.

6 Ameaças à Validade

A primeira ameaça refere-se ao tamanho e à representatividade da amostra, dado que o estudo contou apenas com 27 participantes voluntários. Apesar do grupo ser composto por estudantes e profissionais da indústria de diferentes nacionalidades e graus de experiência, a amostra é estatisticamente reduzida. Isso significa que os padrões comportamentais identificados, como as diferenças entre os perfis ABI e TIM, podem não refletir a totalidade dos desenvolvedores de software. Outro aspecto a ser considerado é a especificidade da tarefa e da tecnologia empregada, já que as atividades foram limitadas a um projeto específico (*TodoMVC*) desenvolvido com a biblioteca *React* e a linguagem *JavaScript*. Em outros paradigmas de programação, como linguagens fortemente tipadas ou desenvolvimento *backend*, os padrões de interação e a verbosidade dos *prompts* podem variar, já que a sintaxe do código ou a necessidade de contexto podem exigir estratégias de *prompt* diferentes.

Destaca-se a ameaça à adequação das métricas de legibilidade empregadas para responder à primeira questão de pesquisa (QP1). Índices como *Flesch-Kincaid* e *Gunning Fog* foram originalmente desenvolvidos para a prosa convencional, e seu uso em *prompts* que mesclam linguagem natural a trechos de código (como nomes de variáveis e sintaxe de funções) pode gerar resultados distorcidos. Nesses casos, os algoritmos podem interpretar o código como "palavras complexas" ou "polissílabas", elevando artificialmente a estimativa do nível de escolaridade sem que isso reflita uma real sofisticação linguística do participante. Outro fator adicional à validade dos resultados refere-se ao processo de normalização dos dados linguísticos. Como a amostra foi composta por participantes de diversas nacionalidades, observou-se que 55,6% das interações (n=15) foram originalmente realizadas em português, enquanto 44,4% (n=12) ocorreram em inglês. Para garantir a uniformidade na análise e na aplicação das métricas de legibilidade, todos os *prompts* redigidos em português foram traduzidos para o inglês. Apesar do esforço para manter a fidelidade semântica, o processo de tradução pode ter alterado sutilmente a estrutura sintática original e o número de sílabas, o que constitui uma variável que pode influenciar os índices de legibilidade calculados.

7 Conclusão

Esta pesquisa teve como objetivo compreender como diferentes perfis de desenvolvedores interagem com LLMs, especificamente com o *GitHub Copilot*, em tarefas de programação. O estudo investigou como a experiência e o estilo cognitivo (por meio do método GenderMag) influenciam a engenharia de *prompts*, preenchendo uma lacuna quanto à influência de fatores humanos na interação humano-IA.

Por meio de um estudo empírico com 27 participantes em um projeto personalizado (*TodoMVC*), as três questões de pesquisa foram respondidas com base em dados quantitativos e qualitativos. Os resultados indicam que a interação é adaptativa, na qual o esforço de elaboração do *prompt* (contagem de palavras) cresce proporcionalmente à complexidade da tarefa. Contudo, a sofisticação linguística manteve-se acessível (nível escolar de 8ª/9ª série), sugerindo que os programadores priorizam a comunicação direta e a mescla entre linguagem natural e código, em detrimento de formalismos gramaticais.

Quanto à estruturação, predominou a categoria "Pergunta Aberta", revelando que o Copilot atuou prioritariamente como tutor para navegação e compreensão da base de código, e não apenas como gerador passivo. A análise estatística confirmou que o perfil do usuário correlaciona-se ao tipo de *prompt*, em que perfis TIM mantiveram objetividade e instruções diretas, enquanto perfis ABI inexperientes geraram *prompts* extensos devido à falta de vocabulário técnico. Em contrapartida, a experiência atuou como moderadora, permitindo que o grupo ABI - Experiente superasse a aversão ao risco e adotasse comandos mais diretos.

Disponibilidade de Artefatos

Material disponível em: DOI: 10.5281/zenodo.21337393[2].

Agradecimentos

Igor Wiese agradece ao CNPq (409359/2024-6, 444802/2024-0, 317015/2025-7) e à Fundação Araucária/Governo do Paraná (PRD2023361000043)

Referências

[1] Jens Albrecht, Sidharth Ramachandran, and Christian Winkler. 2020. *Blueprints for text analytics using Python.* " O'Reilly Media, Inc.".

[2] B. Alves de Oliveira. 2026. Material suplementar do artigo "Um Estudo Empírico de Padrões de Prompt usados durante a realização de tarefas de programação e compreensão de código". doi:10.5281/zenodo.21337393

[3] Markus Borg, Dave Hewett, Nadim Hagatulah, Noric Couderc, Emma Söderberg, Donald Graham, Uttam Kini, and Dave Farley. 2025. Echoes of AI: Investigating the Downstream Effects of AI Assistants on Software Maintainability. arXiv:2507.00788 [cs.SE] https://arxiv.org/abs/2507.00788

[4] Margaret Burnett, Scott D. Fleming, Shamsi Iqbal, Gina Venolia, Vidya Rajaram, Umer Farooq, Valentina Grigoreanu, and Mary Czerwinski. 2010. Gender differences and programming environments: across programming populations. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (Bolzano-Bozen, Italy) *(ESEM '10)*. Association for Computing Machinery, New York, NY, USA, Article 28, 10 pages. doi:10.1145/1852786.1852824

[5] Arifa I. Champa, Md Fazle Rabbi, Costain Nachuma, and Minhaz F. Zibran. 2024. ChatGPT in Action: Analyzing Its Use in Software Development. In *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*. 182–186.

[6] Ionut Daniel Fagadau, Leonardo Mariani, Daniela Micucci, and Oliviero Riganelli. 2024. Analyzing Prompt Influence on Automated Method Generation: An Empirical Study with Copilot. In *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension (ICPC '24)*. ACM, 24–34. doi:10.1145/3643916.3644409

[7] Balreet Grewal, Wentao Lu, Sarah Nadi, and Cor-Paul Bezemer. 2024. Analyzing Developer Use of ChatGPT Generated Code in Open Source GitHub Projects. In *Proceedings of the 21st International Conference on Mining Software Repositories* (Lisbon, Portugal) *(MSR '24)*. Association for Computing Machinery, New York, NY, USA, 157–161. doi:10.1145/3643991.3645072

[8] Md Hamid, Amreeta Chatterjee, Mariam Guizani, Andrew Anderson, Fatima Moussaoui, Sarah Yang, Isaac Escobar, Anita Sarma, and Margaret Burnett. 2024. *How to Measure Diversity Actionably in Technology.* 469–485. doi:10.1007/978-1-4842-9651-6_27

[9] Jenny T. Liang, Melissa Lin, Nikitha Rao, and Brad A. Myers. 2024. Prompts Are Programs Too! Understanding How Developers Build Software Containing Prompts. arXiv:2409.12447 [cs.SE] https://arxiv.org/abs/2409.12447

[10] Roman Macháček, Anastasiia Grishina, Max Hort, and Leon Moonen. 2025. The Impact of Fine-tuning Large Language Models on Automated Program Repair. arXiv:2507.19909 [cs.SE] https://arxiv.org/abs/2507.19909

[11] Saikat Mondal, Suborno Deb Bappon, and Chanchal K. Roy. 2024. Enhancing User Interaction in ChatGPT: Characterizing and Consolidating Multiple Prompts for Issue Resolution. arXiv:2402.04568 [cs.SE] https://arxiv.org/abs/2402.04568

[12] Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an LLM to Help With Code Understanding. arXiv:2307.08177 [cs.SE] https://arxiv.org/abs/2307.08177

[13] Caio Monteiro de Oliveira. 2024. Utilização de Chat Bots baseados em LLMs para automação de testes de software. (2024).

[14] Gustavo Pinto, Cleidson de Souza, Thayssa Rocha, Igor Steinmacher, Alberto de Souza, and Edward Monteiro. 2023. Developer Experiences with a Contextualized AI Coding Assistant: Usability, Expectations, and Outcomes. arXiv:2311.18452 [cs.SE] https://arxiv.org/abs/2311.18452

[15] Jonan Richards and Mairieli Wessel. 2024. What You Need is What You Get: Theory of Mind for an LLM-Based Code Understanding Assistant. arXiv:2408.04477 [cs.SE] https://arxiv.org/abs/2408.04477

[16] Italo Santos, João Felipe Pimentel, Igor Wiese, Igor Steinmacher, Anita Sarma, and Marco A. Gerosa. 2023. Designing for Cognitive Diversity: Improving the GitHub Experience for Newcomers. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 1–12. doi:10.1109/icse-seis58686.2023.00007

[17] Anshul Shah, Anya Chernova, Elena Tomson, Leo Porter, William G. Griswold, and Adalbert Gerald Soosai Raj. 2025. Students' Use of GitHub Copilot for Working with Large Code Bases. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1* (Pittsburgh, PA, USA) *(SIGCSETS 2025)*. Association for Computing Machinery, New York, NY, USA, 1050–1056. doi:10.1145/3641554.3701800

[18] Viktoria Stray, Astri Barbala, and Viggo Tellefsen Wivestad. 2025. Human-AI Collaboration in Software Development: A Mixed-Methods Study of Developers' Use of GitHub Copilot and ChatGPT. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering* (Clarion Hotel Trondheim, Trondheim, Norway) *(FSE Companion '25)*. Association for Computing Machinery, New York, NY, USA, 1325–1332. doi:10.1145/3696630.3730566

[19] Mihaela Vorvoreanu, Lingyi Zhang, Yun-Han Huang, Claudia Hilderbrand, Zoe Steine-Hanson, and Margaret Burnett. 2019. From Gender Biases to Gender-Inclusive Design: An Empirical Investigation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–14. doi:10.1145/3290605.3300283

[20] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. arXiv:2302.11382 [cs.SE] https://arxiv.org/abs/2302.11382

[21] Liangxuan Wu, Yanjie Zhao, Xinyi Hou, Tianming Liu, and Haoyu Wang. 2024. ChatGPT Chats Decoded: Uncovering Prompt Patterns for Superior Solutions in Software Development Lifecycle. In *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*. 142–146.

[22] Tao Xiao, Christoph Treude, Hideaki Hata, and Kenichi Matsumoto. 2024. DevGPT: Studying Developer-ChatGPT Conversations. In *Proceedings of the 21st International Conference on Mining Software Repositories (MSR '24)*. ACM, 227–230. doi:10.1145/3643991.3648400