UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

MARIANA OLIVEIRA FELIPE

UMA INVESTIGAÇÃO EMPÍRICA DA ENGENHARIA DE *PROMPTS* EM LLMS COMO ESTRATÉGIA PARA A GERAÇÃO DE HISTÓRIAS DE USUÁRIO

QUIXADÁ

2026

MARIANA OLIVEIRA FELIPE

UMA INVESTIGAÇÃO EMPÍRICA DA ENGENHARIA DE *PROMPTS* EM LLMS COMO ESTRATÉGIA PARA A GERAÇÃO DE HISTÓRIAS DE USUÁRIO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Orientadora: Profa. Dra. Carla Ilane Moreira Bezerra.

QUIXADÁ

2026

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

F353i Felipe, Mariana Oliveira.
Uma investigação empírica da Engenharia de Prompts em LLMs como estratégia para a geração de Histórias de Usuário / Mariana Oliveira Felipe. – 2026.
64 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2026.
Orientação: Profa. Carla Ilane Moreira Bezerra.

1. Histórias de Usuário. 2. Engenharia de Prompts. 3. Modelos de Linguagem. 4. Engenharia de Requisitos. 5. Desenvolvimento Ágil. I. Título.

CDD 005

MARIANA OLIVEIRA FELIPE

UMA INVESTIGAÇÃO EMPÍRICA DA ENGENHARIA DE *PROMPTS* EM LLMS COMO ESTRATÉGIA PARA A GERAÇÃO DE HISTÓRIAS DE USUÁRIO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Aprovada em: 22/01/2026.

BANCA EXAMINADORA

Profa. Dra. Carla Ilane Moreira Bezerra (Orientadora)
Universidade Federal do Ceará (UFC)

Carla Taciana Lima Lourenço Silva Schuenemann
Universidade Federal de Pernambuco (UFPE)

Rainara Maia Carvalho
Universidade Federal do Ceará (UFC)

Aos meus pais, por saberem de mim antes que eu soubesse de mim mesma, e por enxergarem caminhos que ainda aprendo a ver.

AGRADECIMENTOS

À minha mãe, por sempre me oferecer o abraço mais seguro, os gestos e palavras de carinho mais reconfortantes, e por me amparar e me defender com amor incondicional em todos os momentos.

Ao meu pai, por sua dedicação constante e sacrifícios, que inspiram em mim a índole que busco seguir e a pessoa e profissional que almejo ser, à altura do exemplo que ele representa.

Aos meus irmãos, por, de diferentes formas, terem me proporcionado suporte ao longo dos anos difíceis e cansativos desta jornada acadêmica.

À Profa. Dra. Carla Ilane Moreira Bezerra, por depositar em mim sua confiança, paciência e direcionamentos fundamentais durante a orientação deste estudo.

Às professoras que integraram a banca avaliadora, Carla Taciana Lima Lourenço Silva Schuenemann e Rainara Maia Carvalho, pelas valiosas sugestões que agragaram em muito minha percepção sobre este trabalho.

Aos amigos que carrego comigo ao longo de tantos anos, por ressignificarem e tornarem esta jornada mais leve, compartilhada e possível, obrigada por tanto.

"Arte significa felicidade e coisas bonitas que nos tocam... Bem, continuar vivo também é arte. Suponho que seja a arte mais sublime, não pensa assim?" (KIAROSTAMI, Abbas; *E a Vida Continua*, 1992.)

RESUMO

Os *Large Languages Models* (LLMs) têm sido aplicados para apoiar atividades de Engenharia de Requisitos, como a redação de histórias de usuário. No entanto, ainda há incerteza sobre a estabilidade dessas gerações quando o mesmo processo é repetido e sobre como essa estabilidade se relaciona com a qualidade percebida por especialistas. Este trabalho investigou a consistência semântica de histórias de usuário geradas por três LLMs (*ChatGPT 5 Instant, DeepSeek* e *Gemini 2.5 Pro*) em quatro domínios, repetindo um protocolo baseado nos 3Cs. A consistência foi estimada por *embeddings* e similaridade do cosseno, considerando apenas histórias comparáveis entre repetições, o mesmo indicador para a elaboração dos artefatos. Em paralelo, uma amostra controlada foi avaliada por dois especialistas sobre dois domínios, com critérios inspirados no framework *Quality User Story* (QUS). Os resultados indicaram consistência de moderada a alta, variando por modelo e domínio, com *ChatGPT* e *DeepSeek* mais previsíveis e o *Gemini* mais exploratório, gerando mais histórias, porém menos uniformes entre repetições. Na avaliação humana, as histórias foram consideradas úteis como ponto de partida, mas com fragilidades recorrentes em atomicidade e testabilidade. Por fim, observou-se que consistência semântica e qualidade percebida capturam aspectos distintos e devem ser tratadas como medidas complementares.

Palavras-chave: histórias de usuário; engenharia de prompts; modelos de linguagem; engenharia de requisitos; desenvolvimento ágil.

ABSTRACT

Large language models (LLMss) have been applied to support Requirements Engineering activities, such as drafting user stories. However, uncertainty remains regarding the stability of these generations when the same process is repeated and how such stability relates to the quality perceived by specialists. This study investigated the semantic consistency of user stories generated by three LLMss (*ChatGPT 5 Instant*, *DeepSeek*, and *Gemini 2.5 Pro*) across four domains by repeating a protocol based on the 3Cs. Consistency was estimated using text *embeddings* and cosine similarity, considering only stories that were comparable across repetitions (i.e., produced under the same user story identifier). In parallel, a controlled sample was assessed by two specialists in two domains using criteria inspired by the *Quality User Story* framework (QUS). The results indicated moderate to high consistency, varying by model and domain: *ChatGPT* and *DeepSeek* were more predictable, whereas *Gemini* behaved more exploratorily, generating a larger number of stories but showing less uniform repeatability across runs. In the human evaluation, the stories were deemed useful as a starting point, yet recurring weaknesses were observed in atomicity and testability. Overall, semantic consistency and perceived quality capture different properties and should be treated as complementary measures.

Keywords: user stories; prompt engineering; language models; requirements engineering; agile development.

LISTA DE ILUSTRAÇÕES

Figura 1 – Arquitetura *Transformer* utilizada pelos LLMs 24
Figura 2 – Visão geral do fluxo metodológico 37
Figura 3 – Exemplo base do prompt de inicialização do cenário de entrevista 40
Figura 4 – Exemplo base do prompt para a geração das Histórias de Usuário 41
Figura 5 – Quantidade de histórias de usuário geradas por interação (I01–I03), por domínio e modelo 45
Figura 6 – Heatmap do Domínio 01 51
Figura 7 – Heatmap do Domínio 02 52

LISTA DE QUADROS

Quadro 1 – Exemplo de estrutura de História de Usuário 17
Quadro 2 – Estrutura dos 3C's das Histórias de Usuário 18
Quadro 3 – Exemplo de critério de aceitação baseado em BDD 19
Quadro 4 – Os 13 Critérios de Qualidade do QUS 21
Quadro 5 – Comparativo entre os trabalhos relacionados e o trabalho proposto 36
Quadro 6 – Resumo do arranjo experimental 38
Quadro 7 – Categorias de erros e sua tradução em diretrizes do *Prompt A* 38
Quadro 8 – Exemplo de fluxo para construção de um *prompt* eficaz na geração de HU . 40
Quadro 9 – Questões de pesquisa 44

LISTA DE ABREVIATURAS E SIGLAS

BDD	*Behavior-driven development*
CA	Critérios de Aceitação
ER	Engenharia de Requisitos
HU	Histórias de Usuário
IA	Inteligência Artificial
LLMs	*Large Language Models*
NLP	*Natural Language Processing*
QUS	*Quality User Story*

SUMÁRIO

1 INTRODUÇÃO 13
1.1 Objetivos 15
1.1.1 Objetivo Geral 15
1.1.2 Objetivos Específicos 15
2 FUNDAMENTAÇÃO TEÓRICA 16
2.1 Engenharia de Requisitos no Contexto Ágil 16
2.1.1 Histórias de Usuário 17
2.1.1.1 Critérios de Aceitação *18*
2.1.2 Critérios de Qualidade para Artefatos Ágeis 19
2.1.2.1 Quality User Story (QUS) *20*
2.1.2.2 Similaridade semântica baseada em embeddings de sentença *22*
2.2 *Large Language Models* (LLMs) 23
2.2.1 ChatGPT 5 25
2.2.2 DeepSeek 25
2.2.3 Gemini 2.5 Pro 25
2.3 Engenharia de *Prompts* 26
2.3.1 Técnicas e Estratégias de Formulação de Prompts 27
2.4 Integração de LLMs na Engenharia de Requisitos Ágil 27
3 TRABALHOS RELACIONADOS 30
3.1 *Take Loads Off Your Developers: Automated User Story Generation Using Large Language Model* 30
3.2 *Transforming Software Requirements into User Stories with GPT-3.5: An AI-Powered Approach* 31
3.3 *Leveraging LLMs for User Stories in AI Systems: UStAI Dataset* 32
3.4 *Evaluating user story quality with LLMs: a comparative study* 33
3.5 *LLMREI: Automating Requirements Elicitation Interviews with LLMs* . . 34
3.6 Análise Comparativa 35
4 METODOLOGIA 37
4.1 Planejamento experimental 37
4.2 Estruturação das entrevistas de elicitação com LLMs 38

4.3 **Geração de histórias de usuário a partir das observações** 40
4.4 **Avaliação: métricas automáticas e por especialistas** 41
4.4.1 ***Avaliação automática*** 42
4.4.2 ***Avaliação por Especialistas*** 42
4.5 **Análise comparativa entre modelos** 43
5 **RESULTADOS** 44
5.1 **Consistência textual entre repetições** 44
5.1.1 ***QP1: Em que medida as histórias de usuário geradas permanecem semanticamente consistentes quando o mesmo protocolo é repetido sob condições semelhantes?*** 45
5.2 **Comparação da consistência textual observada entre modelos** 47
5.2.1 ***QP2: De que forma a consistência textual observada se diferencia entre modelos de LLMs e entre os domínios avaliados no estudo?*** 48
5.3 **Avaliação de qualidade por especialistas** 50
5.3.1 ***QP3: Como especialistas em Engenharia de Software avaliam a qualidade das histórias geradas e quais critérios de qualidade tendem a apresentar mais fragilidades?*** 50
5.4 **Relação entre consistência textual observada e avaliação humana** 53
5.4.1 ***QP4: Em que medida a consistência textual entre repetições se relaciona com a qualidade percebida por avaliadores humanos ao julgar as histórias geradas?*** 53
5.5 **Discussões e Implicações** 54
5.6 **Ameaças à validade** 56
6 **CONCLUSÕES E TRABALHOS FUTUROS** 58
6.1 **Trabalhos Futuros** 58
REFERÊNCIAS 59

1 INTRODUÇÃO

Os *Large Language Models* (LLMs) despontaram em uma crescente vertiginosa como ferramentas que potencializaram a área de desenvolvimento de software (Brockenbrough *et al.*, 2025), tornando-se promissores em domínios onde humanos e modelos de Inteligência Artificial (IA) pudessem em conjunto, colaborar para processos de engenharia mais ágeis (Carleton *et al.*, 2022). Ao impulsionarem a automação na elicitação, geração de especificações, garantia de qualidade e outros aspectos de requisitos (Hemmat *et al.*, 2025), evidencia-se o potêncial exploratório de novas demandas quanto à forma como esses modelos são utilizados.

Apesar dos avanços, os LLMs ainda enfrentam limitações, como a dificuldade em lidar com os nuances linguísticos específicos de determinados domínios e a falta de compreensão sobre dependências complexas que surgem em diferentes cenários da Engenharia de Requisitos (ER) (Alhanahnah *et al.*, 2025; Rahman *et al.*, 2024; Sarsa *et al.*, 2022). Para mitigar essas limitações e aprimorar o desempenho dos modelos nesse contexto, têm sido empregados diversos métodos de ajuste e estratégias de Engenharia de *Prompts* (Hemmat *et al.*, 2025).

Um *prompt* pode influenciar uma cadeia de interações e as respostas geradas, ao fornecer aos LLMs regras e especificações para a estrutura da conversação, estabelecendo um conjunto base de instruções e delimitando o viés contextual (White *et al.*, 2023), refinando assim a relevância das informações processadas, bem como a precisão das respostas esperadas. Em contextos ágeis, as histórias de usuário, devido à sua natureza semi-estruturada, porém flexível (Lucassen *et al.*, 2016), surgiram como um artefato enxuto e documentável para comunicar de forma eficaz os requisitos dos usuários finais (Lucassen *et al.*, 2015). Com isso, o uso estratégico de *prompts* pode viabilizar a automação da geração dessas histórias, otimizando tempo e consistência no processo de elicitação.

Contudo, garantir uma validação contextual eficaz ainda é um dos grandes desafios, já que abordagens automatizadas podem tanto interpretar incorretamente a intenção expressa na história de usuário, comprometendo sua qualidade semântica, quanto não reconhecer se ela é viável na prática de desenvolvimento, afetando sua qualidade pragmática (Sharma; Tripathi, 2025). Para reforçar a clareza e a testabilidade dos requisitos, é comum que essas histórias sejam acompanhadas por artefatos complementares, como os critérios de aceitação, frequentemente associados a essas histórias, funcionam como condições objetivas para validar sua implementação, oferecendo clareza e alinhamento sobre o que deve ser entregue (Ferreira *et al.*, 2022).

Nesse contexto, diferentes abordagens têm sido adotadas para avaliar artefatos

textuais produzidos por LLMs. Em cenários supervisionados, métricas tradicionais como precisão e *recall* são amplamente empregadas para mensurar desempenho quando há rótulos ou referências bem definidas (Hemmat *et al.*, 2025). Entretanto, em tarefas abertas de Engenharia de Requisitos, como a elicitação e a redação de histórias de usuário, frequentemente não existe um *gold standard* textual único, uma vez que diferentes formulações podem ser igualmente válidas. Assim, a literatura de NLP tem explorado medidas de similaridade semântica baseadas em representações distribuídas (*embeddings*), que permitem comparar textos no espaço vetorial mesmo diante de variações linguísticas (Devlin *et al.*, 2019). Em particular, embeddings em nível de sentença/documento combinados com similaridade do cosseno constituem uma alternativa recorrente para estimar proximidade de conteúdo, sendo úteis para analisar estabilidade e convergência semântica em textos gerados automaticamente.

Este trabalho propõe investigar como os LLMs de última geração — *ChatGPT 5 Istant*, *Gemini 2.5 Pro* e *DeepSeek* — se comportam na tarefa de elicitação e geração de histórias de usuário a partir de entrevistas estruturadas com um *stakeholder*, conduzidas por meio de *prompts* cuidadosamente elaborados. Ao realizar uma análise comparativa entre os modelos, esta pesquisa busca avaliar a qualidade dos artefatos gerados sob múltiplas perspectivas: semântica, pragmática e estrutural, dado a adoção do framework *Quality User Story* (QUS), proposta por Lucassen *et al.* (2015).

Com isso, a avaliação dos artefatos gerados será conduzida sob duas perspectivas complementares. A primeira consiste em uma análise automática de consistência textual entre múltiplas repetições do mesmo protocolo de elicitação, empregando *embeddings* de sentença/documento e similaridade do cosseno. Essa perspectiva permite quantificar o grau de repetibilidade das saídas geradas sob condições controladas, sem pressupor a existência de uma referência textual única. A segunda análise corresponde à avaliação por especialistas da área de Engenharia de Requisitos, que examinarão a adequação das histórias às boas práticas de documentação ágil, contemplando atributos estruturais e pragmáticos, em alinhamento com critérios consolidados na literatura, como os do *Quality User Story* (QUS) (Lucassen *et al.*, 2015).

Portanto, ao preencher lacunas metodológicas quanto à confiabilidade das histórias de usuário geradas e à eficácia das técnicas de *prompts* utilizadas, este estudo pretende não apenas contribuir para o entendimento das limitações e potencialidades desses modelos, mas também fornecer insumos práticos para profissionais da área, promovendo um uso mais consciente e efetivo de LLMs no desenvolvimento de software ágil.

1.1 Objetivos

1.1.1 Objetivo Geral

Investigar empiricamente o impacto de estratégias de engenharia de *prompts* na geração automática de histórias de usuários por modelos de linguagem natural.

1.1.2 Objetivos Específicos

a) Analisar a condução de entrevistas de elicitação por diferentes LLMs, considerando coerência e tipos de erros de elicitação.
b) Caracterizar a qualidade das histórias de usuário geradas a partir do *Quality User Story* (QUS) como critérios de análise (estrutura, semântica e pragmática).
c) Comparar os modelos de LLMs (*ChatGPT 5 Instant*, *Gemini 2.5 Pro* e *DeepSeek*) quanto à qualidade e consistência dos artefatos gerados em repetições experimentais.
d) Mensurar a consistência textual entre repetições por meio de embeddings de sentença/documento e similaridade do cosseno, relacionando-a à avaliação de especialistas.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os conceitos fundamentais para o desenvolvimento deste trabalho, organizados em quatro eixos. A Seção 2.1 trata da Engenharia de Requisitos no contexto ágil, abordando princípios, práticas e artefatos como histórias de usuário, critérios de aceitação e critérios de qualidade aplicáveis. A Seção 2.2 apresenta os LLMs utilizados — *GPT-4o*, *Gemini 2.5 Pro* e *DeepSeek* — com foco em suas principais características. A Seção 2.3 discute a Engenharia de Prompts, incluindo técnicas de formulação, boas práticas e seus impactos na performance dos modelos. Por fim, a Seção 2.4 aborda a aplicação dos LLMs na Engenharia de Requisitos Ágil.

2.1 Engenharia de Requisitos no Contexto Ágil

O gerenciamento ágil de projetos é uma metodologia em destaque no desenvolvimento de software, por sua versatilidade e flexibilidade de resposta a mudanças contínuas (Lei *et al.*, 2024). Algumas das metodologias ágeis mais conhecidas incluem *Extreme Programming* (XP), *Scrum* e *Kanban* (Mnkandla; Dwolatzky, 2004). Cada uma delas incorpora práticas e princípios específicos que contribuem para um processo de desenvolvimento mais dinâmico, interativo e orientado a feedbacks contínuos.

Por sua vez, o Manifesto Ágil propõe uma abordagem menos convencional para o desenvolvimento de software, priorizando indivíduos e suas interações em detrimento de processos e ferramentas, a entrega de software funcional em vez de uma documentação extensa, a colaboração com o cliente em vez da negociação de contratos formais e a adaptação constante às mudanças em vez da adesão rígida a um plano pré-definido (Beck *et al.*, 2001). Nesse contexto, as práticas se tornam abertas a mudanças em áreas que podem ser identificadas a qualquer momento, onde os métodos tornam-se uma resposta eficaz às mudanças constantes, promovendo ciclos curtos de entrega e melhoria contínua.

Estudos apontam que a adoção de métodos ágeis em projetos está associada a um aumento da produtividade, à redução de retrabalho e a uma maior eficiência na correção de defeitos (Eberlein; Cesar, 2002; Bin *et al.*, 2004; Lagerberg; Skude, 2013). Outras contribuições relevantes da Engenharia de Requisitos, como o envolvimento ativo do cliente, a priorização contínua de requisitos (Ramesh *et al.*, 2010), a modelagem de requisitos (Boness; Harrison, 2007) e a documentação de requisitos (Emmerich, 2011), também foram sugeridas para complementar

os métodos ágeis. Entretanto, a comunidade de desenvolvimento de software ainda compreende de forma limitada o papel dessas práticas em contextos tão flexíveis e dinâmicos. Nesse cenário, alternativas que contribuam para agilizar a definição e gestão de requisitos vêm sendo exploradas, a fim de mitigar desafios frequentemente enfrentados em abordagens tradicionais de ER.

2.1.1 Histórias de Usuário

As Histórias de Usuário (HU) têm se consolidado como um recurso cada vez mais presente no processo de desenvolvimento de software, especialmente em metodologias ágeis. Consideradas o artefato mais amplamente adotado nesse contexto (Schön *et al.*, 2017), elas representam os requisitos sob a perspectiva do usuário.

Atualmente, as HU seguem um modelo semiestruturado e compacto, proposto por Cohn (2004), conforme ilustrado no Quadro 1. Esse formato busca representar quem (papel) precisa do quê (recurso) e por quê (benefício) (Kassab, 2014; Lucassen *et al.*, 2016). Em Lucassen *et al.* (2016), a segunda e a terceira parte são chamadas de "meios"e "fins".

Quadro 1 – Exemplo de estrutura de História de Usuário

Parte	**Exemplo**
Quem	Como um usuário autenticado
O quê (meios)	Eu quero salvar um formulário preenchido
Por quê (fins)	Para que eu possa consultar os dados posteriormente

Fonte: Adaptado de Brockenbrough *et al.* (2025).

Segundo Cohn (2004), as HU são compostas por três elementos principais: (i) uma breve descrição da funcionalidade desejada, (ii) discussões entre os membros da equipe com o objetivo de esclarecer e refinar os detalhes da história, e (iii) critérios de aceitação ou testes que documentam os requisitos necessários para que a história seja considerada concluída. Esses elementos garantem que a funcionalidade seja compreendida de forma colaborativa e validada com clareza antes de sua implementação.

Para deixar mais evidente esse propósito, Jeffries (2001) propôs o modelo conhecido como 3C's. Cada "C" representa um aspecto essencial das histórias de usuário, conforme detalhado no Quadro 2: *Card* (Cartão), que contém a descrição concisa da funcionalidade; *Conversation* (Conversação), que corresponde ao diálogo entre cliente e equipe para esclarecer e evoluir a história; e *Confirmation* (Confirmação), que refere-se aos critérios de aceitação que determinam se a história está completa. Assim, mesmo que o cartão não detalhe tudo, ele serve como ponto de partida para a comunicação contínua e validação das necessidades do usuário.

Quadro 2 – Estrutura dos 3C's das Histórias de Usuário

Componente	Descrição
Cartão	O cartão não contém todas as informações do requisito, mas serve como lembrete para discussão posterior. Pode também incluir informações como prioridade e custo estimado.
Conversa	O requisito é comunicado através de conversas entre o cliente e o time de desenvolvimento. Essas discussões podem ser complementadas por documentos, mas o foco está na comunicação contínua.
Confirmação	A confirmação da história ocorre por meio de exemplos que geram critérios de aceitação. O cliente define como cada história será aceita, e esses critérios devem ser preferencialmente automatizados em testes. A história é considerada concluída quando todos os testes de aceitação são bem-sucedidos.

Fonte: Adaptado de Wildt *et al.* (2015).

Além disso, o livro eXtreme Programming: Práticas para o dia a dia no desenvolvimento ágil de software de Wildt *et al.* (2015), explora também que as HU seguem um ciclo de vida que envolve refinamento progressivo. Em etapas iniciais, elas podem surgir a partir de épicos, que representam funcionalidades amplas demais para serem desenvolvidas em uma única iteração e, por isso, são posteriormente decompostas em histórias menores e mais específicas. Esses épicos, por sua vez, são geralmente agrupados em temas, que refletem grandes capacidades ou objetivos do produto.

Ainda assim, apesar de serem o artefato mais amplamente utilizado no desenvolvimento ágil de software, as HU ainda apresentam desafios significativos de análise. Por serem escritas em linguagem natural, tornam-se acessíveis e fáceis de entender para as partes interessadas. No entanto, esse mesmo aspecto pode gerar desvantagens, como ambiguidade, inconsistências e lacunas de informação nos requisitos descritos (Meth *et al.*, 2013; Silva, 2017).

2.1.1.1 Critérios de Aceitação

Os Critérios de Aceitação (CA) compõem a estrutura das HU como condições de satisfação que delimitam quando uma funcionalidade pode ser considerada completa (North, 2006). Eles estabelecem limites claros para a implementação, auxiliando na validação e no alinhamento com as expectativas do usuário. Esses critérios podem abranger comportamentos funcionais, regras de negócio e aspectos de qualidade a serem testados (Ferreira *et al.*, 2022). Portanto, o objetivo é compreender melhor as HU, além de permitir a geração de testes de aceitação para que o código seja verificado e validado a cada iteração da *sprint*.

Diferentemente das HU, os CA não seguem uma estrutura padronizada amplamente consolidada. No entanto, como exemplificado no Quadro 3, North (2006) propõe uma forma estruturada baseada em três componentes: *Given* (Dado que), *When* (Quando) e *Then* (então)

(then). Nessa abordagem, o bloco **Dado que** define o contexto inicial; **Quando** descreve uma ação ou evento que ocorre; e **Então** especifica o resultado esperado diante da ação executada (Martinelli *et al.*, 2022). Baseado no exemplo de Brockenbrough *et al.* (2025), o modelo abaixo segue um estilo comum em *Behavior-driven development* (BDD):

Quadro 3 – Exemplo de critério de aceitação baseado em BDD

Componente	Descrição
Dado que	O usuário está logado
Quando	Ele clica no botão “Salvar”
Então	Os dados devem ser salvos com sucesso

Fonte: Adaptado de Brockenbrough *et al.* (2025).

Ao seguir essa estrutura, todos os *stakeholders* conseguem compreender claramente quais condições precisam ser satisfeitas para que uma HU seja considerada concluída. Cada critério deve ser redigido de forma objetiva e direta; quando muito extenso ou complexo, é recomendável dividi-lo em critérios menores e sequenciais, facilitando a validação em etapas. É essencial que a escrita desses critérios se tornem significativamente mais simples quando a HU é bem definida e independente. Como ressaltado anteriormente, os critérios de aceitação funcionam como elementos-chave na formulação de casos de teste, contribuindo diretamente para a verificação do comportamento esperado da funcionalidade (Ferreira *et al.*, 2022).

2.1.2 Critérios de Qualidade para Artefatos Ágeis

A qualidade dos artefatos de requisitos é um fator determinante para o sucesso de projetos de software, pois influencia diretamente na clareza, compreensão e viabilidade de implementação das funcionalidades. Especificações mal definidas ou ambíguas tendem a gerar retrabalho, desalinhamento entre as partes interessadas e dificuldades na validação do produto.

No contexto de desenvolvimento tradicional, a prática recomendada pelo IEEE para especificações de requisitos de software define oito características desejáveis para avaliar a qualidade desses artefatos: corretas, não ambíguas, completas, consistentes, classificadas por importância ou estabilidade, verificáveis, modificáveis e rastreáveis (IEEE Computer Society, 1994). Nesse cenário, o framework *Quality User Story* (QUS), proposto por Lucassen *et al.* (2016), se destaca por oferecer uma estrutura de avaliação mais abrangente, com critérios sintáticos, semânticos e pragmáticos.

Embora o framework QUS ofereça critérios relevantes para avaliar histórias de usuário em dimensões sintáticas, semânticas e pragmáticas, sua aplicação costuma depender

de inspeção manual ou de validações estruturais que não capturam plenamente a proximidade de sentido entre textos redigidos de formas distintas. Em cenários envolvendo geração automática por LLMs, essa limitação torna-se mais evidente, pois diferentes formulações podem expressar a mesma intenção funcional, variando apenas em estilo, granularidade ou escolha lexical. Assim, além de critérios qualitativos como os do QUS, a literatura explora métricas automáticas de similaridade semântica capazes de comparar textos com base em representações distribuídas, mitigando a dependência de correspondência literal entre palavras e permitindo avaliar convergência de conteúdo em linguagem natural.

2.1.2.1 Quality User Story (QUS)

Muitas vezes, as histórias de usuário precisam ser lapidadas, pois podem ser complicadas de entender, grandes demais para serem desenvolvidas, dependentes demais para serem entregues, ou podem não ser verificadas (Wildt *et al.*, 2015). Para esse refinamento, Bill Wake propôs o framework *INVEST*, acrônimo para Independente (autocontida), Negociável (não é um contrato rígido), Valiosa (gera valor para o usuário final), Estimável (possui tamanho previsível), Pequena (cabe em uma iteração delimitada no tempo) e Testável (permite definição de critérios de aceitação) (Wake, 2003).

No entanto, o *INVEST* não aborda a qualidade da escrita dos atributos das histórias de usuário, como a consistência textual entre elas. Esses aspectos, apesar de menos tangíveis, impactam de maneira considerável a comunicação entre os membros da equipe e a compreensão dos requisitos por todas as partes envolvidas.

Para lidar com essa limitação, o *Agile Requiments Verification Framework* de Heck e Zaidman (2014), estabelece três critérios de verificação para requisitos no desenvolvimento ágil: completude, uniformidade e consistência e correção. Entretanto, muitos desses critérios exigem informações suplementares e não estruturadas que não estão presentes no texto principal da HU, os tornando inadequados para ferramentas de ER e para a própria perspectiva tanto do desenvolvedor quanto do usuário (Lucassen *et al.*, 2015). Em razão disso, Lucassen *et al.* (2016) propôs o framework *Quality User Story* (QUS), uma coleção de 13 critérios que determinam a qualidade das histórias de usuário, sendo organizados em níveis do campo sintático, semântico e pragmático, segundo as categorias de Lindland *et al.* (1994):

– **Qualidade sintática**: diz respeito à forma como a história de usuário é organizada textualmente, levando em conta aspectos estruturais e gramaticais, mas não seu conteúdo

ou sentido.

- **Qualidade semântica**: se refere ao significado e às conexões lógicas entre os elementos que compõem o texto da HU de expressão mais apropriadas para transmitir claramente os requisitos desejados.
- **Qualidade pragmática**: envolve a seleção das formas de expressão mais apropriadas para transmitir claramente os requisitos desejados.

O Quadro 4 classifica os critérios conforme se relacionam a uma HU individual ou a um conjunto de histórias. Com isso, é possível distinguir os critérios que dizem respeito à estrutura e clareza de cada história individual (como ser atômica ou não ambígua) daqueles que só podem ser verificados em relação a outras histórias do sistema (como ausência de conflitos ou uniformidade textual entre elas).

Quadro 4 – Os 13 Critérios de Qualidade do QUS

Critério	**Descrição**	**Individual/Conjunto**
Sintáticos		
Bem formada	Uma história de usuário inclui pelo menos um papel e um meio	Individual
Atômica	Expressa um requisito para exatamente uma funcionalidade	Individual
Mínima	Contém apenas papel, meio e finalidade	Individual
Semânticos		
Conceitualmente correta	O "meio"expressa uma funcionalidade e o "fim"expressa uma justificativa	Individual
Orientada ao problema	Especifica apenas o problema, não a solução	Individual
Não ambígua	Evita termos ou abstrações com múltiplas interpretações	Individual
Livre de conflitos	Não deve ser inconsistente com outras histórias de usuário	Conjunto
Pragmáticos		
Estruturada	É uma sentença bem formada	Individual
Estimável	Não expressa requisitos genéricos difíceis de planejar ou priorizar	Individual
Única	Não há duplicatas entre as histórias de usuário	Conjunto
Uniforme	Segue o mesmo modelo/template das demais histórias	Conjunto
Independente	É autocontida, sem dependências diretas	Conjunto
Completa	O conjunto de histórias permite desenvolver uma aplicação completa, sem etapas faltando	Conjunto

Fonte: Adaptado de Lucassen *et al.* (2016).

No aspecto **sintático**, elas devem ser bem estruturadas, conter apenas um requisito por funcionalidade e manter uma forma mínima e clara. Do ponto de vista **semântico**, devem evitar conflitos entre si, ser coerentes, focar no problema e não na solução, e usar linguagem não ambígua. Já na dimensão **pragmática**, é esperado que as histórias sejam escritas de forma consistente, independentes, sem duplicações e, em conjunto, permitam o desenvolvimento completo da aplicação. Embora abrangente, o QUS pode ser considerado mais abstrato que

frameworks como o INVEST (Lucassen *et al.*, 2015), abordado anteriormente.

2.1.2.2 Similaridade semântica baseada em embeddings de sentença

A avaliação automática de textos em NLP apresenta dificuldades pois um mesmo significado pode ser expresso de diversas formas, especialmente em tarefas de geração textual. Métricas tradicionais baseadas na sobreposição de n-grams, como *BLEU* (Papineni *et al.*, 2002) e *ROUGE* (Lin, 2004), são amplamente adotadas devido à sua simplicidade, no entanto, tendem a penalizar paráfrases e variações lexicais, o que limita sua capacidade de representar adequadamente a similaridade semântica entre textos.

No contexto da Engenharia de Requisitos (ER), esse problema é ainda mais relevante, pois a elicitação e a documentação de requisitos são conduzidas majoritariamente em linguagem natural, demandando comunicação clara com stakeholders e posterior consolidação em artefatos formais (Das *et al.*, 2021). Nesse cenário, o uso de técnicas de NLP tem sido associado ao aumento da eficiência no desenvolvimento de software, ao automatizar ou apoiar atividades que tradicionalmente demandam grande esforço humano (Dalpiaz *et al.*, 2018). Essas técnicas permitem que sistemas computacionais analisem textos e identifiquem padrões por meio de abordagens como similaridade semântica textual, recuperação de informação, classificação de documentos e reconhecimento de entidades, entre outras (Das *et al.*, 2021).

Dessa forma, muitas abordagens recentes dependem de mecanismos eficazes de *sentence embeddings*, pois eles possibilitam a comparação semântica entre sentenças ou requisitos (Das *et al.*, 2021). Conceitualmente, *embeddings* representam informações linguísticas como vetores densos em um espaço de menor dimensionalidade, no qual elementos semanticamente semelhantes tendem a estar próximos entre si (Jatnika *et al.*, 2019). Nesse tipo de representação, a similaridade semântica pode ser estimada por medidas geométricas, sendo a similaridade do cosseno uma das mais utilizadas. Essa medida avalia o grau de alinhamento entre vetores, assumindo valores mais próximos de 1 quando há maior proximidade semântica (Jatnika *et al.*, 2019). Formalmente, para dois vetores u e v, a similaridade do cosseno é definida como:

$$\text{cos\_sim}(u,v) = \frac{u \cdot v}{\|u\|\|v\|} \tag{2.1}$$

Em tarefas de comparação textual, essa medida é interpretada como um indicador de proximidade semântica no espaço de *embeddings*. Ela é especialmente útil em cenários nos quais

diferentes formulações são esperadas, mas o objetivo é identificar convergência de conteúdo. Além disso, por considerar apenas o ângulo entre os vetores, e não sua magnitude, a similaridade do cosseno é menos sensível a diferenças relacionadas ao tamanho do texto ou à frequência de termos, o que favorece a comparação entre sentenças de comprimentos distintos.

2.2 *Large Language Models* (LLMs)

Segundo Zhang *et al.* (2024), os LLMs, devido às suas avançadas capacidades de interpretação e raciocínio em linguagem natural, são sistemas de IA altamente potentes, treinados em grandes volumes de dados textuais para gerar linguagem com traços humanos. Eles oferecem uma alternativa ou um recurso complementar à produção textual feita por humanos, ajudando a superar essas limitações (Long *et al.*, 2024).

Conforme apontado por Yao *et al.* (2024), os LLMs devem apresentar quatro atributos essenciais: (i) habilidade para interpretar com profundidade o contexto presente na linguagem natural; (ii) competência para produzir textos com características próximas à escrita humana; (iii) sensibilidade ao contexto, especialmente em áreas que demandam alto nível de conhecimento; e (iv) aptidão para seguir instruções, o que os torna úteis na resolução de problemas e em processos de tomada de decisão. Esses modelos são amplamente utilizados em áreas como tradução automática, geração de conteúdo, assistentes virtuais (*chatbots*) e síntese de informações em diversos contextos (Hemmat *et al.*, 2025).

A Figura 1 apresenta a arquitetura geral *Encoder-Decoder* dos LLMs, conforme proposta por Vaswani *et al.* (2017) e denominada *"The Transformer Model"*. Essa arquitetura é amplamente adotada em modelos de processamento de linguagem natural por sua eficiência na compreensão e geração de texto. A estrutura é composta por dois blocos principais: o *Encoder*, responsável por processar a entrada textual e gerar uma representação contextualizada dos *tokens*; e o *Decoder*, que utiliza essa representação para gerar a saída de forma autoregressiva.

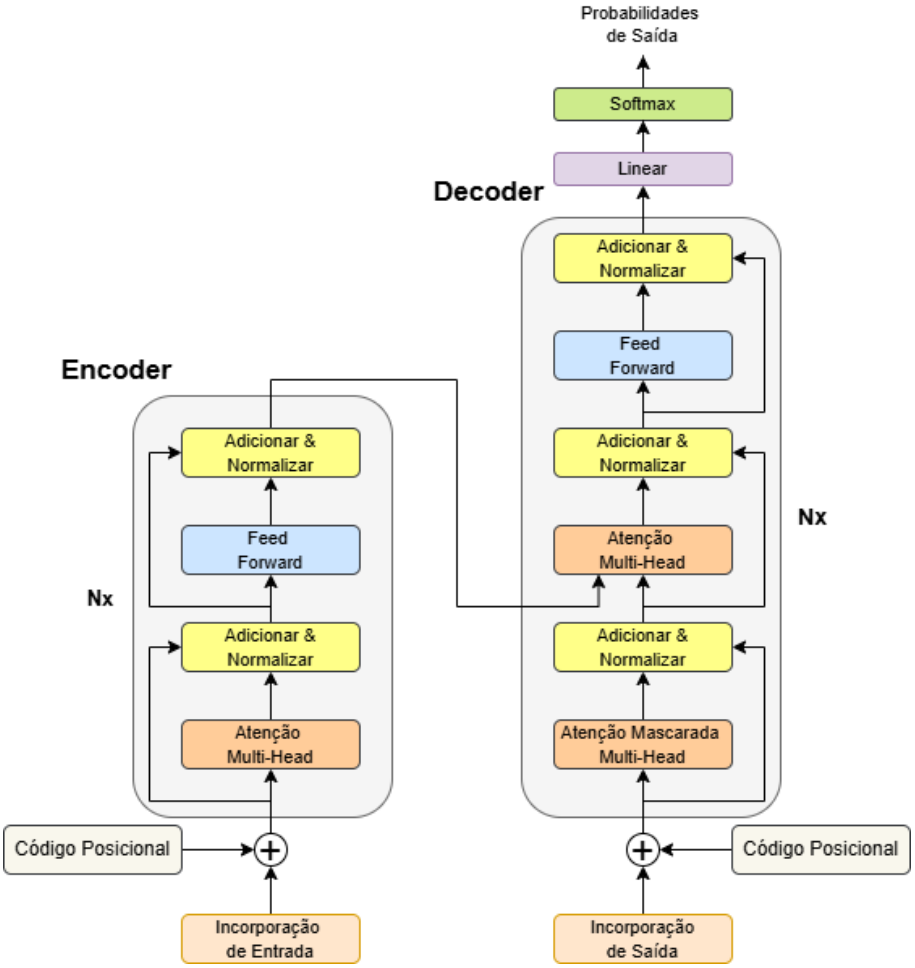
O *Encoder* é formado por múltiplas camadas idênticas que aplicam mecanismos de *self-attention* e *redes feed-forward*. Isso permite que o modelo capture relações complexas entre palavras, independentemente de sua posição na sequência, considerando o contexto de forma global. Já o *Decoder*, além do *self-attention*, incorpora um mecanismo de *encoder-decoder attention*, que permite consultar diretamente as saídas do Encoder. Esse processo é fundamental para a geração de textos coerentes, especialmente em tarefas como tradução automática e resposta a perguntas. Tanto o *Encoder* quanto o *Decoder* utilizam *positional encodings* para manter o

senso de ordem dos *tokens*, já que a arquitetura *Transformer* não é sequencial por padrão. Esse diferencial estrutural viabiliza o paralelismo no treinamento e contribui para a escalabilidade dos modelos.

Modelos como o BERT, que utiliza apenas o *Encoder*, e o GPT, baseado apenas no Decoder, são exemplos da flexibilidade dessa arquitetura. Já modelos mais completos utilizam os dois blocos combinados para tarefas mais complexas, como sumarização e tradução neural.

Considerando a diversidade de modelos de linguagem atualmente disponíveis, este estudo concentra-se em três LLMs de destaque, selecionados por sua relevância e ampla aplicação em tarefas que exigem compreensão contextual, raciocínio avançado e geração de texto com alta precisão. São eles: o GPT-4o, o Gemini 2.5 Pro e o DeepSeek. As subseções 2.2.1, 2.2.2 e 2.2.3 apresentam uma descrição detalhada das principais funcionalidades, características técnicas e capacidades de cada um desses modelos.

Figura 1 – Arquitetura *Transformer* utilizada pelos LLMs



Fonte: Adaptado de Vaswani *et al.* (2017).

2.2.1 ChatGPT 5

O *Chat Generative Pre-Trained Transformer* (*ChatGPT*) é um modelo de linguagem desenvolvido pela OpenAI para compreensão e geração de texto de forma contextualizada, sendo amplamente utilizado em assistentes virtuais, produção de conteúdo e sistemas interativos. A versão mais recente, baseada no *GPT-5*, introduz avanços relevantes em desempenho, eficiência e qualidade das respostas, com destaque para o modo *Instant*, que prioriza baixa latência e alta responsividade, permitindo interações mais fluidas e próximas de uma conversa humana em tempo real. Além disso, o *ChatGPT 5* mantém o suporte nativo a múltiplas modalidades, como texto, imagem e áudio, aprimora o raciocínio em tarefas complexas e amplia o suporte a diferentes idiomas, consolidando-se como uma ferramenta versátil tanto para aplicações cotidianas quanto para cenários mais avançados (OpenAI, 2024).

2.2.2 DeepSeek

O *DeepSeek* é uma família de modelos de linguagem desenvolvida com foco em alto desempenho em raciocínio lógico, análise matemática e aplicações técnicas especializadas. Seus modelos mais recentes, como o *DeepSeek-V3* e o *DeepSeek-R1*, se destacam por oferecer respostas precisas em diversas áreas, graças à incorporação de grafos de conhecimento específicos. Com dados de treinamento atualizados até o final de 2023, esses modelos também se mostram mais alinhados com as atualizações tecnológicas recentes. Embora ainda não ofereçam suporte multimodal completo, como geração de imagens ou interações por voz, os modelos do *DeepSeek* têm ganhado reconhecimento por sua capacidade de lidar com contextos extensos e tarefas complexas (Liao, 2025).

2.2.3 Gemini 2.5 Pro

O *Gemini 2.5 Pro*, desenvolvido pelo Google DeepMind, representa um avanço significativo na tecnologia de LLMs. Lançado em março de 2025, o modelo é projetado para lidar com tarefas complexas, se destacando por suas capacidades de raciocínio e codificação. Uma das inovações notáveis é o modo "*Deep Think*", que utiliza técnicas de pensamento paralelo para considerar múltiplas hipóteses antes de gerar respostas, resultando em um desempenho considerável em benchmarks avançados, como o *USAMO* e o *LiveCodeBench*. Além disso, o *Gemini 2.5 Pro* possui capacidades multimodais nativas e também ao gerar saídas em áudio

nativo que capturam nuances da fala humana. Essas características tornam o modelo eficaz em aplicações que exigem interação natural, se consolidando como uma ferramenta avançada na área de IA (Google DeepMind, 2024).

2.3 Engenharia de *Prompts*

Um *prompt* é um conjunto de instruções de entrada, fornecidas aos LLMs para orientar a geração de suas respostas (Liu *et al.*, 2023). Trata-se de uma ferramenta básica para otimizar o desempenho do modelo em tarefas específicas, onde se têm explorado diversos padrões de *prompts* para maximizar esses benefícios (Zhang *et al.*, 2024).

Por sua vez, a Engenharia de *Prompts* é definida como um conjunto de técnicas voltadas ao aprimoramento das instruções fornecidas a um modelo de IA (Nguyen-Duc *et al.*, 2023). A depender do contexto de uso do modelo, projetar *prompts* adequados torna-se fundamental para garantir respostas mais precisas e alinhadas aos objetivos desejados (Rahman *et al.*, 2024).

Avançando nesse cenário, *prompts* bem estruturados podem ir além da simples orientação de saída textual, apresentando capacidades mais sofisticadas, como para adaptação automática, sugerindo outros *prompts* para coletar informações adicionais ou gerar artefatos relacionados. Essas funcionalidades mais avançadas reforçam a importância de projetá-los cuidadosamente, buscando extrair valor que vá além da geração básica de texto ou código (White *et al.*, 2023).

Diante disso, é válido considerar que a qualidade das saídas geradas por LLMs conversacionais está diretamente relacionada à qualidade dos *prompts* fornecidos pelo usuário (White *et al.*, 2023). Nesse contexto, a discussão sobre a reutilização de conhecimento na forma de padrões ganha destaque, vertente essa estudada na literatura de engenharia de software em diversos níveis como design, arquitetura e análise (Schmidt *et al.*, 2013; Gamma *et al.*, 1995). Considerando que *prompts* são, em essência, uma forma de programação, documentá-los como padrões é uma abordagem natural e estratégica para promover sua reutilização e adaptação em diferentes domínios.

2.3.1 Técnicas e Estratégias de Formulação de Prompts

Estudos indicam que mudanças na formulação dos *prompts* em linguagem natural, como o uso de paráfrases, afetam a qualidade das respostas geradas por ferramentas como o *Copilot* (Mastropaolo *et al.*, 2023). Há uma variedade de técnicas aplicadas na Engenharia de *Prompts*, como a definição explícita de papéis *(role-prompting)*, o uso de aspas triplas para delimitar conteúdos, a realização de múltiplas tentativas de geração e outras abordagens que visam aprimorar a qualidade das respostas obtidas (Chen *et al.*, 2025).

Também destacam-se estratégias como *chain-of-thought*, que orienta o modelo a explicitar passos intermediários no raciocínio (Wei *et al.*, 2022). Já o *k-shot prompting* consiste em fornecer exemplos concretos no prompt, com o objetivo de contextualizar a tarefa e orientar a geração de respostas mais alinhadas ao esperado (Brown *et al.*, 2020). Outra abordagem é o padrão *fact checklist*, que garante a verificação explícita de critérios específicos na resposta do modelo, aumentando sua completude e precisão (White *et al.*, 2023).

Entretanto, segundo o livro *Prompt Engineering for LLMs: The Art and Science of Building Large Language Model–Based Applications* de Berryman e Ziegler (2024), a formulação de prompts não se limita apenas à escolha isolada de técnicas, mas também ao desafio de combiná-las de forma eficiente. Esse processo pode ser entendido como um problema de otimização, em que o pesquisador precisa decidir quais elementos incluir, em que ordem apresentá-los e como equilibrar requisitos e incompatibilidades entre eles.

Além disso, há a limitação do tamanho do contexto (*context window*), que impõe um orçamento de *tokens* a ser considerado, exigindo que apenas informações relevantes sejam priorizadas. Nesse sentido, ainda que não exista uma ferramenta padronizada que resolva automaticamente esse tipo de formulação, essa etapa se mostra essencial para adequar o *prompt* às necessidades de cada aplicação.

Assim, a engenharia de *prompts* deve ser entendida como um processo iterativo e criativo, que permite ao pesquisador construir soluções personalizadas, capazes de alinhar qualidade, eficiência e relevância no uso de LLMs.

2.4 Integração de LLMs na Engenharia de Requisitos Ágil

Ao contrário das abordagens tradicionais baseadas em *Natural Language Processing* (NLP), os LLMs demandam pouca adaptação a conjuntos de dados específicos e conseguem

se ajustar dinamicamente a diferentes formas de expressão linguística. Eles são capazes de analisar histórias de usuário e critérios de aceitação de maneira integrada, levando em conta tanto a correção gramatical quanto a consistência do conteúdo. Além disso, esses modelos conseguem reproduzir padrões presentes em dados reais e utilizar o extenso conhecimento acumulado durante a fase de pré-treinamento. A integração dos LLMs no processo de avaliação tem o potencial de economizar tempo, aumentar a consistência das análises e melhorar a precisão, reduzindo, assim, a dependência de esforços manuais extensivos (Sharma; Tripathi, 2025).

Segundo Zhang *et al.* (2024), em ambientes ágeis de desenvolvimento de software, os requisitos são definidos e priorizados de forma iterativa, geralmente na forma de histórias de usuário e critérios de aceitação. Essa abordagem facilita a adaptação contínua às mudanças nas necessidades dos usuários e assegura a entrega de valor por meio de ciclos incrementais. A qualidade dessas histórias de usuário (Ferreira *et al.*, 2022; Lucassen *et al.*, 2016) tem impacto direto tanto na agilidade do desenvolvimento quanto na satisfação do cliente. Diante da ênfase das metodologias ágeis na rápida iteração e flexibilidade, os modelos de LLMs vêm ganhando destaque como ferramentas promissoras para apoiar a análise desses artefatos, visto que alcançar critérios de qualidade nesses artefatos ainda representa um desafio.

A transição para o ágil, embora promissora, também impõe novos desafios à condução da ER, o que motivou estudos que buscam mapear as práticas adotadas e os obstáculos enfrentados por equipes ágeis. Paralelamente, tem-se observado um crescente interesse no uso de LLMs em tarefas de ER, como elicitação, análise e classificação de requisitos. No entanto, apesar do potencial, a pesquisa sobre sua aplicação prática na indústria e avaliação sistemática de desempenho ainda é limitada (Šmite *et al.*, 2024).

Contudo, mais recentemente, vêm surgido iniciativas que buscam explorar não apenas a geração de requisitos por LLMs, mas também sua validação automática, a detecção de ambiguidades e a avaliação da qualidade de histórias de usuário a partir de métricas estruturais e semânticas (Sharma; Tripathi, 2025; Zhang *et al.*, 2024). Essas abordagens abrem caminho para o uso de modelos como ferramentas de apoio direto ao processo decisório, integrando critérios de qualidade desde as fases iniciais de especificação, apesar das as soluções atuais ainda se encontram em estágio inicial e enfrentarem restrições, já que, em grande parte, dependem de documentos de requisitos previamente disponíveis, o que pode implicar em trabalho adicional no desenvolvimento de software (Pereira *et al.*, 2025).

Nesse sentido, os estudos mais recentes convergem para a visão de que os LLMs

não substituem o trabalho humano, mas funcionam como ferramentas colaborativas capazes de potencializar a produtividade e a consistência da análise de requisitos, já que uma das etapas mais críticas e trabalhosas do processo, ou seja, a captação de requisitos de diferentes fontes (documentos, entrevistas, conversas, pesquisas etc.), ainda depende exclusivamente de esforço humano, sem apoio automatizado (Pereira *et al.*, 2025). Esse caráter híbrido, que combina expertise humana e apoio automatizado, constitui uma das principais tendências atuais do desenvolvimento ágil (Cabrero-Daniel *et al.*, 2024).

3 TRABALHOS RELACIONADOS

A partir da análise da literatura disponível, foram encontrados estudos significativos que exploram diferentes perspectivas e abordagens por meio de LLMs, sobre a geração e a avaliação da qualidade de histórias de usuário e seus artefatos. Esta seção apresenta esses trabalhos, ressaltando suas principais contribuições e suas conexões com a pesquisa em desenvolvimento.

3.1 *Take Loads Off Your Developers: Automated User Story Generation Using Large Language Model*

Rahman *et al.* (2024) aplicaram um estudo que envolveu a Engenharia de *Prompts* para gerar histórias de usuário a partir de especificações de requisitos de alto nível e semi-detalhadas. Os pesquisadores tiveram como objetivo responder três questões de pesquisa: (Q1) se os LLMs poderiam gerar especificações de histórias de usuário aprimoradas ao aplicar a técnica de *prompt Refinement and Thought* (RaT); (Q2) com que precisão seria possível gerar histórias de usuário a partir de documentos de requisitos; e (Q3) se a saída gerada seria aceita por uma ampla variedade de profissionais da engenharia de software.

A metodologia da pesquisa seguiu uma abordagem de validação em três etapas: (i) aplicação de um questionário RUST; (ii) a utilização do *BERTScore* para comparar as histórias de usuário geradas com o conjunto de referência obtido a partir das respostas dos participantes humanos; e (iii) o *AlignScore* para validar as histórias de usuário geradas. Além disso, ainda propuseram uma nova técnica de *prompt*, *Refinement and Thought* (RaT), uma versão especial do *Chain-of-Thought* (CoT), projetada para otimizar o desempenho dos LLMs no processamento de entradas com informações redundantes e *tokens* não reconhecíveis.

Os resultados partem do desafio do estudo em lidar com as alucinações dos LLMs, que geram informações incorretas ou inexistentes, especialmente em documentos longos e complexos. A técnica RaT ajudou a reduzir essas alucinações, melhorando a precisão e consistência na geração das histórias de usuário, respondendo assim à Q1. Quanto à Q2, os resultados do *BERTScore* e *AlignScore* mostraram que o *Geneus* (ferramenta proposta que utiliza *GPT-4-turbo* para criar automaticamente histórias de usuários a partir de documentos de requisitos de software) tem desempenho melhor que o método *single-shot* dos LLMs pré-treinados, embora em alguns casos específicos o desempenho tenha sido um pouco inferior. Por fim, na Q3, as histórias geradas foram avaliadas como boas pela maioria dos especialistas, mas ainda com espaço para

melhorias, especialmente na especificidade e nos aspectos técnicos.

Enquanto Rahman *et al.* (2024) focam na redução de alucinações e na melhoria da coerência textual por meio da técnica de *prompt* Refinement and Thought (RaT), este trabalho investiga um protocolo estruturado de elicitação e geração baseado no formato 3Cs, aplicado com estratégias padronizadas de *prompting* e múltiplas repetições experimentais, avaliando a estabilidade do método por meio da consistência semântica entre execuções.

3.2 *Transforming Software Requirements into User Stories with GPT-3.5: An AI-Powered Approach*

Oswal *et al.* (2024) tiveram como objetivo para este trabalho, demonstrar a integração dos LLMs no processo de desenvolvimento ágil de software, por meio da criação de uma aplicação que converte textos de requisitos em histórias de usuário estruturadas, facilitando assim o processo de Engenharia de Requisitos Ágil. Diversas técnicas de *prompt* foram exploradas e testadas para obter a saída desejada em formato JSON, onde depois foi exibida em uma página web.

Em seus procedimentos metodológicos, o artigo propõe a implementação de uma aplicação simples que recebe textos de requisitos de software e, por meio da API do modelo *GPT-3.5*, gera histórias de usuário em formato *JSON*, exibidas na interface web. A conexão com a API da *OpenAI* utilizou parâmetros básicos, como o número máximo de *tokens* e o modelo *gpt-3.5-turbo-0613*. Após o recebimento da resposta em *JSON*, os dados são tratados e renderizados na interface. Inicialmente, os pesquisadores testaram *prompts* do tipo *Zero-Shot*, mas o resultado apresentou textos adicionais fora do padrão desejado. Assim, foi optado por utilizar a abordagem *Few-Shot*, fornecendo exemplos e a estrutura exata do *JSON* esperado, o que resultou em respostas mais consistentes e utilizáveis. Por fim, o objetivo do frontend é permitir ao usuário inserir diretamente o texto do requisito ou fazer *upload* de um arquivo .txt.

Como resultado, a aplicação proposta foi capaz de converter automaticamente um texto de requisito em histórias de usuário estruturadas. No exemplo testado, o sistema identificou corretamente os atores (como o cliente) e suas respectivas ações (navegar por produtos, adicionar ao carrinho, finalizar compras), gerando a saída no formato *JSON* conforme especificado pelo *prompt Few-Shot*. A ferramenta demonstrou ser eficaz na redução do tempo e esforço necessários para gerar histórias de usuário, além de minimizar erros humanos e inconsistências.

Apesar do estudo de Oswal *et al.* (2024) demonstrarem a aplicação prática de LLMs

para converter requisitos textuais em histórias de usuário estruturadas por meio de *prompts Few-Shot*, este trabalho diferencia-se por adotar uma abordagem voltada à confiabilidade dos resultados, utilizando um protocolo estruturado de elicitação, múltiplas repetições experimentais e a análise da consistência semântica das saídas, complementada por avaliação qualitativa de especialistas.

3.3 *Leveraging LLMs for User Stories in AI Systems: UStAI Dataset*

Yamani *et al.* (2025), em busca de gerar histórias de usuário sobre um domínio específico, de soluções de Inteligência Artificial (IA), que sejam de alta qualidade, o estudo pretende responder a três perguntas: (Q1) se LLMs podem ser utilizados para gerar histórias de usuário a partir da perspectiva de múltiplos *stakeholders*, com base na descrição de um resumo acadêmico sobre o componente de uma IA; (Q2) quais dos LLMs geram histórias de usuário com maior qualidade, segundo o framework *Quality User Story* (QUS); e (Q3) em que medida as histórias de usuário geradas por LLMs capturam considerações éticas, princípios e requisitos não funcionais relevantes esses sistemas de IA.

Para isso geração e avaliação de histórias de usuário a partir de resumos acadêmicos descrevendo componentes de IA consistiu nos seguindo procedimentos: foram geradas 3000 histórias com base em 100 resumos, dos quais 1260 histórias provenientes de 42 resumos foram selecionadas aleatoriamente para análise detalhada, abrangendo 26 domínios temáticos; três LLMs foram utilizados (*Gemini 1.5 Flash*, *ChatGPT o1-mini* e *LLaMA 3.1 70b*); também foi utilizado um *prompt* estruturado, instruindo o modelo a assumir o papel de engenheiro de requisitos e gerar dez histórias a partir de cada resumo; já a avaliação das histórias foi realizada com base no framework QUS; ao passo que foram anotadas implicações éticas e requisitos não funcionais. A avaliação inicial foi feita por dois avaliadores, e uma amostra de 100 histórias foi reavaliada por três engenheiros, utilizando o coeficiente *Kappa de Cohen* para verificar a concordância entre os avaliadores.

O artigo apresentou como resultado a geração de 3000 histórias de usuário, das quais 1260 foram selecionadas para análise. Sobre o desempenhos dos três modelos de LLMs, mostraram diversidade na representação de papéis dos *stakeholders*, com variações em número e especificidade dos papéis identificados. No fator de qualidade, todas as histórias estavam bem formadas sintaticamente, mas apresentaram problemas como ambiguidade, dependência e falta de estimabilidade, sendo o Gemini o modelo com melhor desempenho geral nos critérios

sintáticos, semânticos e pragmáticos, que também se destacou em termos éticos, e a validação por especialistas mostrou alta concordância entre avaliadores, com coeficientes de *Kappa* acima de 0,88 para qualidade e ética, e 0,76 para os requisitos não funcionais.

Em contraste, Yamani *et al.* (2025) concentram-se na geração e avaliação da qualidade de histórias de usuário a partir de resumos acadêmicos sobre componentes de IA, com ênfase em aspectos éticos e requisitos não funcionais, já este trabalho diferencia-se por adotar um fluxo alinhado à Engenharia de Requisitos Ágil, incorporando a etapa de elicitação por meio de entrevistas simuladas e avaliando a estabilidade das histórias geradas por meio da consistência semântica entre múltiplas repetições.

3.4 *Evaluating user story quality with LLMs: a comparative study*

A pesquisa de Sharma e Tripathi (2025) investigou a eficácia do uso de LLMs para a avaliação da qualidade de histórias de usuário como um aspecto crucial para o sucesso no desenvolvimento ágil de software e um passo fundamental em direção à avaliação automatizada da qualidade na engenharia de requisitos. O estudo se propôs a abordar três questões de pesquisa principais: (Q1) o quão eficaz é uma abordagem baseada em LLMs na avaliação da qualidade de histórias de usuários individuais; (Q2) qual categoria de critérios de qualidade (sintático, semântico ou pragmático) é melhor avaliada por LLMs; e (Q3) como diferentes LLMs se comparam em sua capacidade de avaliar a qualidade da história de usuário.

Para responder a essas questões, os pesquisadores utilizaram o *framework Quality User Story*, que organiza os critérios de qualidade em dimensões sintáticas, semânticas e pragmáticas, concentrando-se em oito critérios de qualidade aplicáveis. A metodologia empregou três LLMs: *GPT-4o*, *GPT-4-Turbo* e *GPT-3.5-Turbo*, sob duas estratégias de prompting, *context minimal* (usando descrições curtas dos critérios) e *context rich* (fornecendo descrições detalhadas e exemplos). A validação do desempenho se baseiou em um dataset de 960 histórias de usuário gerado por LLMs alternativos (*Gemini* e *LLaMA3*) e posteriormente avaliado por 69 estudantes de pós-graduação. Os resultados dessas avaliações foram rigorosamente verificados por uma equipe de especialistas, incluindo um professor, estabelecendo um conjunto de dados confiável. As métricas de desempenho utilizadas foram *Accuracy, Precision, Recall e F1 Score*.

Os resultados obtidos revelaram perspectivas úteis sobre os pontos fortes e fracos relativos dos modelos. Foi demonstrado que os modelos *GPT-4o* e *GPT-4-Turbo* exibiram desempenho superior na avaliação da qualidade das histórias de usuário individuais em compa-

ração com o *GPT-3.5-Turbo*. Os modelos avançados da série *GPT-4* mostraram alta eficácia, particularmente nas dimensões sintática e pragmática.

De forma complementar, Sharma e Tripathi (2025) concentram-se na avaliação da qualidade de histórias de usuário já geradas, utilizando LLMs e o framework QUS, enquanto este trabalho adota um fluxo mais amplo, que integra a elicitação e a geração das histórias por meio de entrevistas simuladas e estratégias estruturadas de *prompting*, além de avaliar a estabilidade dos resultados a partir da consistência semântica entre múltiplas repetições.

3.5 *LLMREI: Automating Requirements Elicitation Interviews with LLMs*

O estudo de Korn *et al.* (2025) propõe e avalia o *LLMREI*, um *chatbot* baseado em LLMs destinado a conduzir entrevistas de levantamento de requisitos com intervenção humana mínima, de forma autônoma, visando reduzir erros comuns de entrevistadores humanos e melhorar a escalabilidade do processo de levantamento. A avaliação da eficácia do *LLMREI* concentrou-se em três perguntas de pesquisa principais:(Q1) em que medida o *LLMREI* pode minimizar erros comuns que tipicamente ocorrem em entrevistas de requisitos tradicionais; (Q2) qual é a eficácia do *LLMREI* no levantamento de informações relevantes dos entrevistados e (Q3) em que medida o *LLMREI* consegue adaptar suas perguntas com base no conteúdo e contexto fornecidos pelo entrevistado.

A metodologia do estudo explorou três abordagens para otimizar o *LLMREI*: *zero-shot prompting* (chamada *LLMREI-short*) e *least-to-most prompting* (chamada *LLMREI-long*). A avaliação principal envolveu a comparação do desempenho do *LLMREI-long e LLMREI-short* em 33 entrevistas simuladas com *stakeholders*, utilizando o modelo *GPT-4o*. Os participantes, principalmente estudantes de Ciência da Computação, atuaram como *stakeholders* em dois cenários predefinidos: um salão de beleza/unhas ou um resort de esqui. A avaliação incluiu a verificação manual de erros comuns de entrevista, a medição do número de requisitos relevantes levantados em comparação com uma verdade fundamental predeterminada, e a categorização das perguntas geradas para avaliar a adaptabilidade ao contexto.

Os resultados demonstram que o *LLMREI* apresentou um número de erros comuns semelhante ao de entrevistadores humanos, com a versão de *prompt* longo (*LLMREI-long*) mostrando um desempenho um pouco superior na redução de erros, especialmente quando instruções explícitas eram fornecidas. Em termos de eficácia de levantamento, o *chatbot* foi capaz de extrair uma parcela significativa dos requisitos, sendo que o *prompt* curto (*LLMREI-*

short) foi o mais eficaz, levantando completamente até 60.94% dos requisitos e parcialmente até 12.76%. No que diz respeito à adaptabilidade, aproximadamente metade das perguntas geradas demonstrou forte dependência das respostas anteriores do usuário, comprovando a notável capacidade do modelo de adaptar suas perguntas ao contexto.

Embora o estudo de Korn *et al.* (2025) concentre-se na automação de entrevistas de levantamento de requisitos e avaliam o desempenho do modelo principalmente pela redução de erros e pela cobertura de requisitos em relação a uma verdade fundamental, enquanto este trabalho enfatiza a confiabilidade dos artefatos gerados, integrando a elicitação estruturada (3Cs) à geração de histórias de usuário e avaliando a estabilidade dos resultados por meio da consistência semântica entre múltiplas repetições.

3.6 Análise Comparativa

O Quadro 5 apresenta uma análise comparativa entre o presente trabalho e os cinco estudos relacionados, considerando os seguintes critérios: uso de LLMs na geração de histórias de usuário (com especificação dos modelos), integração do LLMs no processo ágil ou em entrevistas de elicitação, avaliação manual das histórias, aplicação de critérios formais de qualidade (framework QUS) e avaliação automática ou semi-automática com métricas semânticas (por exemplo, medidas baseadas em *embeddings* e similaridade entre textos).

Com isso, é possível observar que o estudo de Rahman *et al.* (2024) aplica a técnica *Refinement and Thought* (RaT) e avalia as saídas com métricas automáticas ancoradas em referência; Oswal *et al.* (2024) desenvolve uma aplicação de conversão automática de requisitos via *prompt few-shot*, com foco na viabilidade e na conformidade estrutural da saída; Yamani *et al.* (2025) compara diferentes modelos de LLMs para gerar histórias sobre soluções de IA, com avaliação via QUS e atenção a aspectos éticos e requisitos não funcionais; Sharma e Tripathi (2025) concentra-se na avaliação da qualidade de histórias já existentes segundo critérios do QUS, formulando o problema como tarefa de classificação supervisionada; e Korn *et al.* (2025) explora LLMs em entrevistas de levantamento de requisitos, avaliando principalmente redução de erros e cobertura de requisitos frente a uma verdade fundamental predeterminada.

O presente trabalho integra essas perspectivas ao empregar LLMs como agentes de apoio à elicitação em entrevistas simuladas, nas quais a autora assume o papel de *stakeholder* de domínios previamente definidos, conduzindo um processo de elicitação estruturado (3Cs). Em seguida, são geradas histórias de usuário com três modelos (*ChatGPT 5 Instant*, *Gemini*

2.5 Pro e *DeepSeek*) sob múltiplas estratégias de *prompting*. A avaliação combina uma análise qualitativa por especialistas orientada pelo QUS e uma medida automática de consistência semântica entre repetições, estimada por embeddings de sentença/documento e similaridade do cosseno. Dessa forma, o estudo discute simultaneamente qualidade e estabilidade dos artefatos gerados, aproximando a análise do uso de LLMs das práticas da Engenharia de Requisitos em contextos ágeis.

Quadro 5 – Comparativo entre os trabalhos relacionados e o trabalho proposto

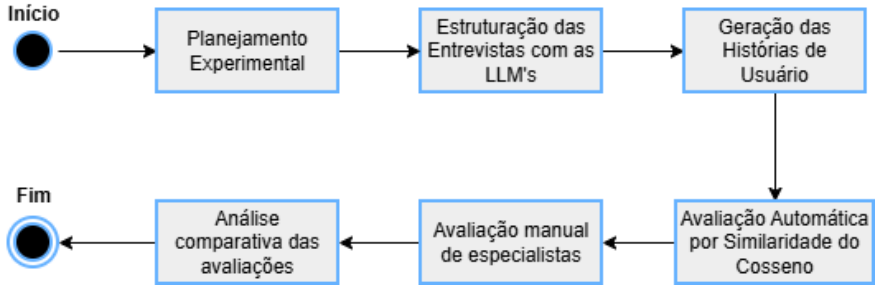
Trabalhos	**Geração com LLMs**	**Avaliação das histórias**	**Critérios formais de qualidade**	**Métricas semânticas automatizadas**
Este trabalho	Sim (*ChatGPT 5 Instant*, *Gemini 2.5 Pro*, *DeepSeek*)	Híbrida (especialistas + automática)	Sim (QUS)	Sim (embeddings + similaridade do cosseno)
Rahman *et al.* (2024)	Sim (GPT-4 Turbo)	Automatizada	Não	Sim (BERTScore, AlignScore)
Oswal *et al.* (2024)	Sim (GPT-3.5 Turbo)	Manual	Não	Não
Yamani *et al.* (2025)	Sim (Gemini, ChatGPT o1-mini, LLaMA 3.1)	Manual	Sim (QUS)	Não
Sharma e Tripathi (2025)	Não	Híbrida (manual + automatizada)	Sim (QUS)	Sim (Accuracy, Precision, Recall, F1)
Korn *et al.* (2025)	Sim (GPT-4o)	Manual	Não	Não

Fonte: Elaborado pela autora.

4 METODOLOGIA

Nesta seção, serão apresentados os procedimentos metodológicos adotados neste estudo, organizados em cinco etapas principais: (i) desenho e planejamento experimental; (ii) condução de entrevistas de elicitação com LLMs atuando como analista de requisitos; (iii) geração de histórias de usuário a partir das observações coletadas; (iv) avaliação automática (métricas semânticas) e avaliação por especialistas, visando capturar tanto similaridade semântica quanto qualidade de engenharia; e (v) análise comparativa entre modelos. Cada processo está detalhado nas subseções deste capítulo, onde a Figura 2 fornece uma visão geral da sequência das etapas e dos procedimentos deste estudo empírico.

Figura 2 – Visão geral do fluxo metodológico



Fonte: Elaborado pelo autor.

4.1 Planejamento experimental

O objetivo desse estudo é comparar três modelos de LLMs, *ChatGPT 5 Instant*, *Gemini 2.5 Pro* e *DeepSeek*, quanto a: (a) capacidade de conduzir entrevistas de elicitação com um mesmo *stakeholder*; e (b) qualidade das histórias de usuário geradas a partir das observações dessas entrevistas. Para garantir diversidade sem inviabilizar a análise, adotam-se quatro domínios (Agendamento Clínico, Doação Sangue, Projetos Extensão, Reposição Estoque)[1] e repetições independentes por combinação domínio X modelo, exemplificado melhor no Quadro 6.

Neste estudo, as **variáveis independentes** são os fatores que definimos e controlamos para observar seus efeitos: *modelo* (qual LLM é usado), *domínio* (o cenário do problema), *repetição* (execuções alternativas para captar variações ao acaso) e *ordem* (sequência em que

[1] https://docs.google.com/document/d/1_zxCATOl1VRd6xa75c94b85gb93JM_1nOT3B8NA5Sps/edit?usp=sharing

Quadro 6 – Resumo do arranjo experimental

Fator	Configuração
Modelos	3 (*ChatGPT 5 Instant*, *Gemini 2.5 Pro*, *DeepSeek*)
Domínios	4
Repetições por combinação	3
Total de entrevistas	$3 \times 4 \times 3 = 36$

Fonte: Elaborado pela autora.

os domínios são entrevistados). Já a escolha de atribuir ao procedimento as repetições se dá porque permitem estimar consistência entre os testes em um mesmo modelo e reduzir o efeito de variações ao acaso entre execuções, ou seja, testar os ruído ou variações aleatórias nas respostas.

4.2 Estruturação das entrevistas de elicitação com LLMs

No processo de estruturação e condução das entrevistas, o objetivo era que a partir de um prompt-base (*Prompt A*), o LLMs assumisse o papel de analista de requisitos, ao conduzir um diálogo em sequência lógica e transparente: *contexto e escopo* $\rightarrow$
textitatores $\rightarrow$ *processos* $\rightarrow$ *regras e exceções* $\rightarrow$ *dados e integrações* $\rightarrow$ *prioridades*. Além disso, a proposta era que o *Prompt A*, dentro de sua estrutura, conseguisse induzir que o modelo realiza-se uma auto-verificação contínua, ou seja, que evitasse desvios comuns em entrevistas de elicitação,efetuando checagens rápidas de cobertura, abordagem baseada no trabalho de Bano *et al.* (2019), onde categorizaram erros cometidos no processo de elicitação por um grupo de estudantes, sendo referenciado mais tarde pelo estudo de Korn *et al.* (2025), reestruturando e filtrando as categorias que mais se encaixavam na percepção que um modelo de LLMs é capaz de replicar, exemplificadas no Quadro 7.

Quadro 7 – Categorias de erros e sua tradução em diretrizes do *Prompt A*

Categoria	O que observar/evitar	Diretriz no prompt (síntese)
Question Formulation	Perguntas ambíguas, vagas ou incompletas que não direcionam a respostas úteis.	Formular perguntas claras e completas; solicitar exemplos ou critérios para desambiguar.
Question Omission	Omissão de tópicos essenciais (atores, regras, exceções, dados, integrações, restrições, prioridades).	Usar checklist por tópico e realizar *gap-check* ao final de cada bloco.
Order of Interview	Quebra da sequência lógica (p. ex., pular contexto/atores antes de mapear processos).	Seguir a ordem definida; se houver desvio, retomar explicitamente o ponto correto.
Communication Skills	Falhas de clareza, confirmação, escuta ativa e espelhamento do entendimento.	Parafrasear e confirmar entendimento; sinalizar mudanças de tópico ao entrevistado.
Customer Interaction	Baixo engajamento, pouca validação e negociação de escopo/prioridades.	Fazer perguntas abertas, validar decisões, negociar prioridades e registrar premissas e riscos.

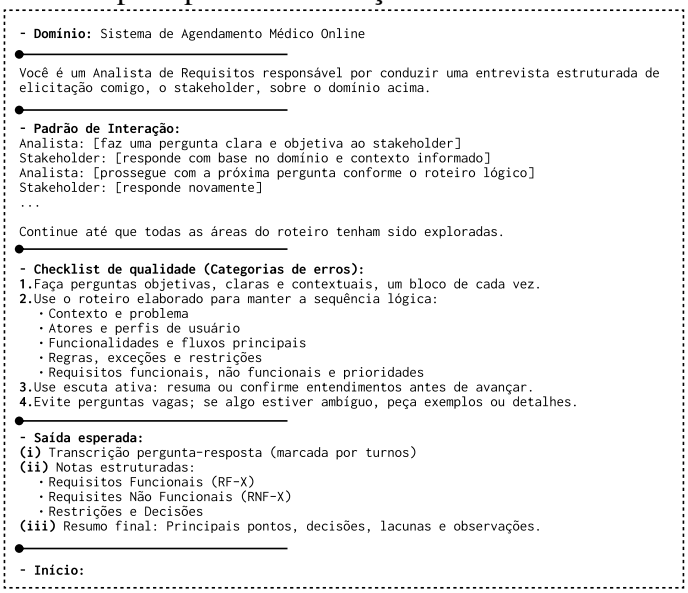
Fonte: Adaptado de Bano *et al.* (2019), Korn *et al.* (2025).

Para assegurar consistência das respostas do entrevistado, adota-se uma **persona única** (o autor como *stakeholder*), formalizada com objetivos, restrições e glossário por domínio. Em cada combinação *modelo* × *domínio*, realizam-se **três entrevistas independentes**, respeitando o contrabalanceamento da ordem dos domínios por modelo. O procedimento segue três passos: (a) execução do *Prompt A*, com o autor respondendo como *stakeholder*; (b) garantia do protocolo, com retomada do ponto correto sempre que necessário; e (c) coleta dos produtos padronizados (transcrição, notas e resumo). As categorias de erro são registradas ao longo do processo para análise posterior e correlação com a qualidade das histórias de usuário geradas na etapa seguinte.

Em seguida, os artefatos consolidados ao fim de cada entrevista (isto é, o resumo estruturado do domínio e a lista de requisitos extraídos ao longo do diálogo) foram exportados para um documento de apoio, de modo que separasse explicitamente a fase de elicitação da fase de geração de artefatos. A partir desse material consolidado, foi realizado uma filtragem manual para isolar apenas os **requisitos funcionais**, não incluindo como insumos os requisitos não funcionais. Essa filtragem foi necessária para garantir comparabilidade, pois o *Prompt B* opera sobre entradas que representam funcionalidades a serem implementadas, evitando que o modelo gere histórias a partir de informações vagas ou de caráter apenas informativo. Com isso, na etapa seguinte, cada requisito funcional selecionado foi utilizado como uma entrada independente do *Prompt B*, mantendo o mesmo formato de solicitação e evitando que requisitos distintos fossem combinados no mesmo comando. Essa decisão reduziu o risco de alucinações em uma demanda grande de entrada e saída e facilitou a rastreabilidade, pois estabelece uma correspondência direta requisito e história de usuário.

Ao final do processo de entrevista e elicitação, foi necessário que cada um dos LLMs buscasse um entendimento coerente do domínio, compondo um mapa inicial de requisitos priorizados e registrando um checklist de cobertura dos tópicos essenciais. Esse material serviu como base direta para a etapa seguinte de geração de histórias de usuário, garantindo rastreabilidade entre o que foi discutido na entrevista e o que foi produzido como artefato (QUS). Além disso, o encerramento da entrevista incluiu uma confirmação final com o *stakeholder* sobre pontos sensíveis, decisões tomadas e lacunas em aberto, de modo a reduzir ambiguidades e orientar futuras iterações. A Figura 3 ilustra o modelo de prompt utilizado para iniciar o cenário de entrevista.

Figura 3 – Exemplo base do prompt de inicialização do cenário de entrevista



```
- Domínio: Sistema de Agendamento Médico Online

Você é um Analista de Requisitos responsável por conduzir uma entrevista estruturada de
elicitação comigo, o stakeholder, sobre o domínio acima.

- Padrão de Interação:
Analista: [faz uma pergunta clara e objetiva ao stakeholder]
Stakeholder: [responde com base no domínio e contexto informado]
Analista: [prossegue com a próxima pergunta conforme o roteiro lógico]
Stakeholder: [responde novamente]
...

Continue até que todas as áreas do roteiro tenham sido exploradas.

- Checklist de qualidade (Categorias de erros):
1.Faça perguntas objetivas, claras e contextuais, um bloco de cada vez.
2.Use o roteiro elaborado para manter a sequência lógica:
   • Contexto e problema
   • Atores e perfis de usuário
   • Funcionalidades e fluxos principais
   • Regras, exceções e restrições
   • Requisitos funcionais, não funcionais e prioridades
3.Use escuta ativa: resuma ou confirme entendimentos antes de avançar.
4.Evite perguntas vagas; se algo estiver ambíguo, peça exemplos ou detalhes.

- Saída esperada:
(i) Transcrição pergunta-resposta (marcada por turnos)
(ii) Notas estruturadas:
   • Requisitos Funcionais (RF-X)
   • Requisites Não Funcionais (RNF-X)
   • Restrições e Decisões
(iii) Resumo final: Principais pontos, decisões, lacunas e observações.

- Início:
```

Fonte: Elaborado pela autora.

4.3 Geração de histórias de usuário a partir das observações

Nesta etapa, foi realizada a formulação dos *prompts* que serviram como entrada para os LLMs, com base nas notas e no resumo de cada entrevista, aplica-se um segundo **prompt** (*Prompt B*), voltado somente para a **geração de histórias de usuário**. Partindo disso, para garantir a qualidade e a uniformidade das entradas, os *prompts* foram alinhados aos princípios do framework *Quality User Story* (QUS), além de estruturar os templates seguindo as técnicas de *prompts* como: *Role Prompting*, *Chain-of-Thought* e *Fact Checklist*, abordadas na revisão bibliográfica anteriormente, onde no Quadro 8 foi buscado sintetizar esse encadeamento lógico.

Quadro 8 – Exemplo de fluxo para construção de um *prompt* eficaz na geração de HU

Etapa	**Técnica**	**Objetivo**
Contextualização	*Role Prompting*	Definir explicitamente o papel esperado do modelo (ex.: engenheiro de requisitos) para estabelecer o contexto e alinhar o tom da resposta.
Preparação	*k-shot prompting*	Fornecer exemplos de histórias de usuário para orientar o formato, a linguagem e o nível de detalhe esperado.
Raciocínio Guiado	*Chain-of-Thought (CoT)*	Estruturar o raciocínio do modelo em passos claros, decompondo requisitos complexos em tarefas menores e coerentes.
Verificação	*Fact Checklist*	Checar se cada história gerada contém todos os elementos essenciais (ator, ação, justificativa, critérios de aceitação), garantindo completude e precisão.

Fonte: Elaborado pela autora.

Além disso, a partir das entrevistas conduzidas com o *Prompt A*, foi obtido um conjunto de requisitos que serviu como o insumo de entrada para a geração das histórias. No total, foram gerados 738 requisitos, entre funcionais e não funcionais, a partir das saídas consolidadas das entrevistas, dos quais 468 foram classificados como requisitos funcionais e utilizados como entradas do *Prompt B*. Esses requisitos funcionais foram distribuídos entre os quatro domínios (DOM1–DOM4) e repetidos ao longo das três interações (I01–I03) para cada modelo, de forma que o volume de entradas variou conforme a cobertura obtida em cada entrevista e a forma como o modelo estruturou os resultados. Esse detalhamento é relevante porque o número de requisitos funcionais determina diretamente o volume de histórias geradas e também influencia a etapa posterior de comparabilidade entre repetições.

A Figura 4 ilustra o exemplo de *prompt* criado para servir como base para a geração das Histórias de Usuário:

Figura 4 – Exemplo base do prompt para a geração das Histórias de Usuário

```
Você é um analista de requisitos. Sua tarefa é criar uma História de Usuário no formato
3Cs (Cartão, Conversa, Confirmação), com base no contexto descrito abaixo.

- Contexto: [inserir aqui o requisito/funcionalidade desejada]

- Raciocínio (Chain-of-Thought):
1.Quem é o usuário principal dessa funcionalidade?
2.Qual é a ação principal que ele deseja realizar?
3.Qual é o benefício esperado dessa ação?
4.Quais critérios de aceitação tornam essa história validável e testável?

- Resultado (formato 3Cs):
Card: Como [persona], quero [ação] para [benefício].
Conversation: [descrição da motivação, contexto adicional ou diálogo com stakeholders]
Confirmation:
- [Critério de aceitação 1]
- [Critério de aceitação 2]
- [Critério de aceitação 3]

- Checklist de qualidade (Framework QUS):
 [ ?/3 ] Sintática: bem formada / atômica / mínima
 [ ?/4 ] Semântica: correta / orientada ao problema / não ambígua/ livre de conflito
 [ ?/6 ] Pragmática: estruturada / estimável / única / uniforme / independente / completa

- Gere apenas uma história por vez
```

Fonte: Elaborado pela autora.

4.4 Avaliação: métricas automáticas e por especialistas

Esta seção descreve a avaliação em duas frentes complementares: (i) métrica automática, baseada em *embeddings* de sentença/documento e similaridade do cosseno para comparar a consistência semântica/textual entre repetições do protocolo de geração; e (ii) avaliação por especialistas, em que dois avaliadores aplicaram uma avaliação qualitativa baseada em critérios adaptados do QUS. Também foi proposto examinar a convergência entre as duas perspectivas (correlação entre a similaridade do cosseno e as notas humanas) e analisar casos de divergência,

por exemplo, alta consistência entre os textos acompanhada de baixa testabilidade, a fim de explicitar os limites das medidas automáticas frente a valor de negócio, clareza e completude exigidos em requisitos.

4.4.1 Avaliação automática

A avaliação automática foi realizada por meio de **similaridade semântica baseada em embeddings** e **similaridade do cosseno**. Diferentemente de métricas supervisionadas que dependem de um texto de referência (*gold standard*) ou de sobreposição lexical, essa abordagem mapeia cada história de usuário para um vetor denso (*embedding*) e estima a proximidade semântica comparando os vetores no espaço vetorial. Como a similaridade do cosseno mede o ângulo entre vetores, ela é adequada para comparar textos com diferentes redações e extensões, priorizando alinhamento de conteúdo em vez de coincidência literal de tokens.

Na prática, para cada combinação *modelo* × *domínio*, foram geradas três repetições (I01, I02 e I03). Para estimar a **consistência semântica intra-modelo**, calculou-se a similaridade do cosseno entre os *embeddings* das histórias produzidas em repetições distintas do mesmo protocolo. Em seguida, as similaridades foram agregadas por média, produzindo um indicador único de consistência para cada *modelo* × *domínio*. Por fim, os resultados foram comparados entre modelos e domínios, permitindo observar variações de estabilidade semântica sob condições experimentais controladas. Todo o fluxo foi automatizado por um script em Python que leu os textos, gerou *embeddings*, computou as similaridades e produziu as visualizações necessárias para a análise.

4.4.2 Avaliação por Especialistas

A avaliação qualitativa das histórias foi conduzida por dois especialistas, que utilizaram um questionário adaptado dos princípios do QUS e de atributos fundamentais de qualidade, como clareza, atomicidade, valor, completude, testabilidade e consistência. Cada história foi avaliada em uma escala Likert de 1 a 5 para cada um dos critérios definidos. Para garantir maior imparcialidade, os avaliadores não tiveram acesso à identificação do modelo gerador nem sabiam se as histórias avaliadas pertenciam a repetições de um mesmo experimento.

4.5 Análise comparativa entre modelos

A comparação entre modelos foi feita a partir de dois eixos principais: (i) **resultados automáticos**, usando indicadores de **consistência textual** obtidos por *embeddings* de sentença/-documento e similaridade do cosseno; e (ii) **avaliação humana**, com notas por critério e um escore composto da avaliação qualitativa aplicada pelos especialistas. Observou-se, ainda, a consistência interna de cada modelo (variação entre as três repetições em um mesmo domínio) e possíveis efeitos da ordem de aplicação dos domínios. Por fim, relacionou-se a qualidade do processo de entrevista (menos ocorrências de erros) com a qualidade do produto (histórias geradas), buscando indícios de associação entre ambos.

5 RESULTADOS

Neste capítulo, são apresentados os resultados deste estudo, organizados de modo a responder às quatro questões de pesquisa definidas anteriormente. Ao longo da análise, cada um dos modelos de LLMs serão referenciados por sua denominação original.

Para garantir transparência e reprodutibilidade, foi disponibilizado um repositório[1] contendo os registros e documentos utilizados na aplicação prática do estudo, bem como os artefatos produzidos e os registros utilizados na análise. Os resultados das histórias de usuário geradas foram organizados em planilhas, estruturadas por domínio e por modelo de LLMs, de forma a facilitar a conferência dos valores reportados e a replicação das etapas de tratamento e comparação dos dados, além do recorte de artefatos avaliados pelos avaliadores externos. O Quadro 9 resume as questões de pesquisa e os respectivos objetivos que orientam a apresentação dos resultados ao longo deste capítulo.

Quadro 9 – Questões de pesquisa

QP	Pergunta	Objetivo
QP1	Em que medida as histórias de usuário geradas permanecem semanticamente consistentes quando o mesmo protocolo é repetido sob condições semelhantes?	Medir a estabilidade semântica das histórias geradas quando o protocolo é repetido (I01–I03), caracterizando o nível geral de repetibilidade do processo.
QP2	De que forma a consistência textual observada se diferencia entre modelos de LLMs e entre os domínios avaliados no estudo?	Comparar a consistência entre modelos e domínios, identificando diferenças de desempenho e padrões de sensibilidade ao contexto.
QP3	Como especialistas em Engenharia de Software avaliam a qualidade das histórias geradas e quais critérios de qualidade tendem a apresentar mais fragilidades?	Analisar a qualidade percebida por especialistas e apontar os critérios com falhas recorrentes (e.g., atomicidade, testabilidade, consistência).
QP4	Em que medida a consistência textual entre repetições se relaciona com a qualidade percebida por avaliadores humanos ao julgar as histórias geradas?	Examinar o alinhamento (ou desalinhamento) entre consistência automática (cosseno) e avaliação humana, discutindo convergências e divergências.

Fonte: Elaborado pela autora.

5.1 Consistência textual entre repetições

Esta seção apresenta os resultados relacionados no que diz respeito a repetibilidade do protocolo, ou seja, o quanto as histórias de usuário permanecem semanticamente semelhantes quando o mesmo procedimento é repetido sob condições semelhantes. Para isso, considerou-

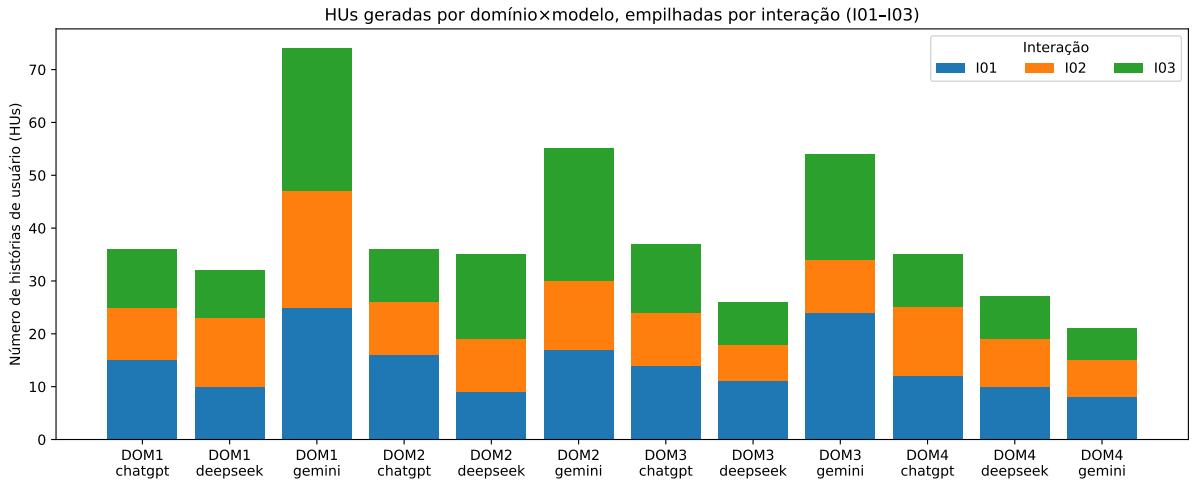
[1] https://drive.google.com/drive/folders/1UWtLKx28-xdzP_iN4dRLMsbKmeUn2ZRF?usp=sharing

se a comparação entre as três interações (I01, I02 e I03), observando tanto a viabilidade de comparação (quantidade de histórias comparáveis) quanto os escores de similaridade obtidos.

5.1.1 QP1: Em que medida as histórias de usuário geradas permanecem semanticamente consistentes quando o mesmo protocolo é repetido sob condições semelhantes?

Para avaliar a consistência semântica do texto gerado, buscou-se estimar o quanto as histórias de usuário permanecem estáveis quando o mesmo protocolo de elicitação e geração é repetido três vezes (I01, I02 e I03) sob condições semelhantes, com o propósito de testar a consistência de abordagem. A Figura 5 apresenta, para cada combinação de domínio e modelo, quantas histórias foram geradas em cada interação (468 histórias no total foram geradas no estudo). Essa visão é importante porque variações na quantidade de histórias por rodada afetam quais saídas podem ser comparadas diretamente entre repetições.

Figura 5 – Quantidade de histórias de usuário geradas por interação (I01–I03), por domínio e modelo



Fonte: Elaborada pela autora.

Como o número de histórias geradas pode variar entre repetições, a comparação exige definir quais saídas são comparáveis entre si. Assim, dentro de cada combinação domínio e modelo, as histórias foram organizadas por identificador (HU) e consideradas comparáveis apenas quando havia pelo menos duas ocorrências do mesmo contexto de histórias de usuário em interações diferentes, permitindo formar pares válidos. Na prática, cada história comparável foi representada por um *embedding* e a similaridade do cosseno foi calculada para os pares de interações disponíveis (I01↔I02, I01↔I03 e I02↔I03). Em seguida, os escores foram agregados por domínio e modelo. Valores médios mais altos indicam menor variação semântica entre

repetições (maior repetibilidade do conteúdo), enquanto valores mais baixos indicam maior divergência entre as saídas do processo.

A Tabela 1 apresenta os escores de consistência textual dos insumos por par de interações, detalhando, em cada par, a média e o desvio padrão da similaridade do cosseno, além do número de HUs comparáveis que se encontram entre parênteses. Esses valores são obtidos a partir de *embeddings* e refletem o grau de proximidade semântica entre as versões geradas em duas repetições específicas.

Tabela 1 – Consistência por par de interações

Domínio	**Modelo**	**I01↔I02**	**I01↔I03**	**I02↔I03**
DOM1	*ChatGPT 5 Instant*	0,630±0,130 (10)	0,656±0,111 (11)	0,619±0,207 (10)
DOM1	*DeepSeek*	0,586±0,095 (10)	0,791±0,096 (9)	0,604±0,070 (9)
DOM1	*Gemini 2.5 Pro*	0,607±0,121 (22)	0,684±0,141 (25)	0,578±0,096 (22)
DOM2	*ChatGPT 5 Instant*	0,654±0,203 (10)	0,592±0,170 (10)	0,610±0,142 (10)
DOM2	*DeepSeek*	0,656±0,131 (9)	0,651±0,088 (9)	0,557±0,137 (10)
DOM2	*Gemini 2.5 Pro*	0,616±0,136 (13)	0,596±0,114 (17)	0,599±0,175 (13)
DOM3	*ChatGPT 5 Instant*	0,678±0,148 (10)	0,784±0,091 (13)	0,730±0,106 (10)
DOM3	*DeepSeek*	0,741±0,141 (7)	0,714±0,143 (8)	0,829±0,126 (7)
DOM3	*Gemini 2.5 Pro*	0,620±0,095 (10)	0,583±0,104 (20)	0,640±0,123 (10)
DOM4	*ChatGPT 5 Instant*	0,728±0,128 (12)	0,702±0,095 (10)	0,773±0,156 (10)
DOM4	*DeepSeek*	0,700±0,142 (9)	0,642±0,091 (8)	0,590±0,092 (8)
DOM4	*Gemini 2.5 Pro*	0,590±0,185 (7)	0,575±0,162 (6)	0,604±0,173 (6)

Fonte: Elaborada pela autora.

Ademais, a Tabela 2 sintetiza esses resultados em nível mais agregado, resumindo a consistência intra-modelo por domínio. Para cada combinação de domínio e modelo, considerou-se apenas o conjunto de histórias de usuário comparáveis (ocorrência em pelo menos duas interações), permitindo formar ao menos um par válido. A coluna Média (cos) representa a média dos escores de similaridade obtidos nos pares disponíveis, enquanto a coluna DP (cos) indica a dispersão desses escores entre as histórias e comparações válidas no mesmo cenário. A coluna N (HUs comparáveis) reporta quantas histórias contribuíram para o cálculo e Pares/HU (média) informa quantas comparações foram possíveis por história.

Considerando os quatro domínios avaliados, no **DOM1** os três modelos apresentaram médias próximas (0,62 a 0,64), indicando estabilidade moderada e semelhante, com maior variabilidade no *ChatGPT*. No **DOM2**, houve redução de consistência, mais acentuada no *Gemini*, enquanto *ChatGPT* e *DeepSeek* permaneceram em níveis semelhantes. No **DOM3**, *ChatGPT* e *DeepSeek* apresentaram médias superiores, indicando maior repetibilidade, enquanto o *Gemini* permaneceu em patamar inferior, sugerindo maior sensibilidade ao contexto. No

Tabela 2 – Consistência textual intra-modelo por domínio

Domínio	Modelo	Média (cos)	DP (cos)	N (HUs comparáveis)	Pares/HU (média)
DOM1	*ChatGPT 5 Instant*	0.642	0.150	11	2.6
DOM1	*DeepSeek*	0.653	0.121	10	3.0
DOM1	*Gemini 2.5 Pro*	0.622	0.129	25	2.2
DOM2	*ChatGPT 5 Instant*	0.602	0.177	10	3.0
DOM2	*DeepSeek*	0.621	0.143	10	2.8
DOM2	*Gemini 2.5 Pro*	0.608	0.147	17	2.5
DOM3	*ChatGPT 5 Instant*	0.725	0.133	13	2.5
DOM3	*DeepSeek*	0.775	0.143	8	2.5
DOM3	*Gemini 2.5 Pro*	0.603	0.112	20	2.2
DOM4	*ChatGPT 5 Instant*	0.733	0.142	12	2.2
DOM4	*DeepSeek*	0.668	0.130	9	2.8
DOM4	*Gemini 2.5 Pro*	0.571	0.140	7	3.0

Fonte: Elaborada pela autora.

DOM4, o *ChatGPT* manteve alta consistência, o *DeepSeek* apresentou valor intermediário e o *Gemini* obteve a menor média e maior dispersão, indicando menor previsibilidade entre repetições.

Além das médias, a dispersão dos escores pode ajudar a interpretar o quão uniforme é a repetibilidade do protocolo. Em termos práticos, um modelo pode apresentar média moderada de similaridade, mas com alto desvio padrão, indicando que a estabilidade varia entre histórias e repetições. Valores menores de desvio padrão sugerem estabilidade mais homogênea, enquanto valores elevados apontam maior sensibilidade ao contexto e ao processo de geração, limitando a previsibilidade do *pipeline* quando aplicado de forma recorrente.

Como síntese, os resultados indicam que a geração de histórias de usuário é moderadamente a altamente consistente quando o protocolo é repetido, mas a estabilidade varia por modelo e domínio. No DOM1, as médias foram próximas (0,62–0,64), sugerindo preservação do conteúdo central, embora o *ChatGPT* apresente maior variabilidade. No DOM2, há redução geral da consistência, mais evidente no *Gemini*. As diferenças tornam-se mais nítidas no DOM3 e DOM4: no DOM3, *ChatGPT* e *DeepSeek* alcançam os maiores níveis de repetibilidade (médias em torno de 0,73), enquanto o *Gemini* permanece abaixo (cerca de 0,60). No DOM4, o *ChatGPT* mantém alta consistência, o *DeepSeek* fica em posição intermediária e o *Gemini* apresenta a menor média, com maior dispersão.

5.2 Comparação da consistência textual observada entre modelos

Nesta seção, os resultados de consistência são contrastados entre os modelos e entre os domínios avaliados. O objetivo foi identificar se existem diferenças sistemáticas de

repetibilidade atribuíveis ao modelo, buscando entender questões de previsibilidade ou ao domínio, visando a maior sensibilidade ao contexto, além de discutir como a dispersão dos escores afeta a interpretação dos resultados.

5.2.1 QP2: De que forma a consistência textual observada se diferencia entre modelos de LLMs e entre os domínios avaliados no estudo?

Os modelos foram comparados quanto à consistência semântica/textual entre repetições do protocolo. Para cada domínio e modelo, foram calculados *embeddings* das histórias geradas e, em seguida, a similaridade do cosseno entre pares de execuções (I01↔I02, I01↔I03 e I02↔I03). Os resultados são sintetizados por média e desvio padrão desses escores, permitindo comparar o nível e a uniformidade da repetibilidade entre modelos e domínios.

A Tabela 3 evidencia que a diferença entre modelos é fortemente dependente do domínio. Em **DOM1** e **DOM2**, a coluna Amplitude (max–min) permanece baixa ($\Delta = 0{,}017$ e $\Delta = 0{,}032$), indicando um cenário próximo de empate técnico: os três modelos produzem níveis semelhantes de repetibilidade para o subconjunto de histórias de usuário comparáveis. Ainda assim, o *ChatGPT* apresenta o desvio padrão mais alto que os concorrentes em ambos os domínios, sugerindo que sua estabilidade, apesar de competitiva na média, é menos uniforme entre os casos.

Tabela 3 – Comparação de consistência entre modelos por domínio

Domínio	ChatGPT 5 Instant	DeepSeek	Gemini 2.5 Pro	Δ (max–min)
DOM1	0,623±0,133 (11)	0,640±0,096 (10)	0,631±0,086 (25)	0,017
DOM2	0,619±0,154 (10)	0,608±0,103 (10)	0,587±0,109 (17)	0,032
DOM3	0,734±0,088 (13)	0,736±0,151 (8)	0,601±0,096 (20)	0,135
DOM4	0,732±0,103 (12)	0,639±0,067 (9)	0,585±0,152 (7)	0,147

Fonte: Elaborada pela autora.

O quadro muda em **DOM3** e **DOM4**, onde as amplitudes crescem consideravelmente ($\Delta = 0{,}135$ e $\Delta = 0{,}147$), evidenciando maior distanciamento entre os modelos. Em **DOM3**, *ChatGPT* e *DeepSeek* obtêm as maiores médias (0,734 e 0,736), enquanto o *Gemini* permanece em patamar inferior (0,601), indicando maior variação semântica entre repetições nesse domínio. Além disso, o desvio padrão do *DeepSeek* é o mais elevado (0,151), apontando que sua alta média convive com maior irregularidade entre as histórias, ao mesmo tempo em que o *ChatGPT* combina média alta com desvio padrão menor (0,088), sugerindo estabilidade mais consistente entre casos.

Em **DOM4**, o padrão se mantém, com contraste ainda mais acentuado: o *ChatGPT* preserva a maior média (0,732), o *DeepSeek* ocupa posição intermediária (0,639) com desvio padrão baixo (0,067), e o *Gemini* apresenta a menor média (0,585) e o maior desvio padrão (0,152). Esse padrão indica não apenas menor repetibilidade média no *Gemini*, mas também menor previsibilidade: parte das histórias mantém proximidade entre repetições e outra parte diverge de forma relevante, mesmo sob condições semelhantes.

Por fim, o *n* (HUs) varia entre modelos e domínios porque a análise inclui apenas o subconjunto comparável, com ocorrência suficiente para formar pares. Portanto, as médias e o desvio padrão devem ser interpretados como repetibilidade condicionada ao conjunto comparável; diferenças em *n* também podem sinalizar variação na cobertura do *pipeline* entre repetições, isto é, quantas histórias de usuário conseguiram ser reproduzidas com identificador correspondente.

Os resultados sugerem dois perfis de comportamento entre os LLMss avaliados. O *ChatGPT 5 Instant* e o *DeepSeek* apresentam maior previsibilidade estrutural nas gerações, com variações menores entre repetições sob o mesmo protocolo. Isso indica uma leitura mais uniforme do processo de elicitação: ao repetir as interações, esses modelos tendem a preservar com maior regularidade o escopo e a organização semântica do conjunto de histórias de usuário comparáveis. Por sua vez, o *Gemini 2.5 Pro* mostrou um padrão mais exploratório e abrangente, tendendo a produzir um número maior de histórias e a desdobrar o problema em mais casos e recortes funcionais; contudo, esse maior detalhamento foi acompanhado por menor uniformidade entre repetições, resultando em menor previsibilidade do *pipeline*. Essa diferença não deve ser interpretada como evidência de alucinação ou de menor qualidade textual, pois o que se mede aqui é estabilidade/repetibilidade sob repetição do protocolo e não a adequação individual de cada história produzida.

Portanto, a consistência varia tanto por domínio quanto por modelo. No DOM1 e DOM2, as médias ficaram próximas (aprox. 0,59–0,64), indicando diferenças pequenas entre modelos e um nível de repetibilidade moderado. Já em DOM3 e DOM4 a separação entre modelos se destacou mais: *ChatGPT* e *DeepSeek* atingiram os maiores escores (DOM3: 0,734–0,736; DOM4: *ChatGPT* 0,732), enquanto o *Gemini* permaneceu abaixo (DOM3: 0,601; DOM4: 0,585), caracterizando maior variação semântica entre repetições nesses domínios. O desvio padrão complementa a leitura ao indicar uniformidade: valores menores sugerem estabilidade mais regular entre as histórias de usuário, e valores maiores apontam instabilidade mais irregular; na prática, o *Gemini* tende a combinar médias menores com maior dispersão (como em DOM4),

enquanto *ChatGPT* e *DeepSeek* apresentam maior repetibilidade, com o *DeepSeek* exibindo maior oscilação em alguns cenários (como o desvio padrão mais alto no DOM3).

5.3 Avaliação de qualidade por especialistas

Esta seção reporta os resultados da avaliação humana das histórias de usuário, conduzida por especialistas em Engenharia de Software a partir de critérios de qualidade definidos no estudo. Considerando o volume total de artefatos, adotou-se uma amostragem controlada para viabilizar uma análise detalhada, mantendo o viés de comparação entre modelos e domínios selecionados. Os resultados são apresentados por critério e por avaliador, destacando padrões de fragilidade recorrentes.

5.3.1 QP3: Como especialistas em Engenharia de Software avaliam a qualidade das histórias geradas e quais critérios de qualidade tendem a apresentar mais fragilidades?

Diante do volume de dados gerados, tornou-se inviável submeter todo o conjunto à avaliação manual com profundidade e consistência. Por isso, esta etapa adotou uma amostragem limitada e controlada, buscando representar os resultados de cada modelo por meio da seleção de uma geração de repetições dos artefatos produzidos, em vez de avaliar apenas um recorte pontual. Essa estratégia permitiu comparar os modelos de forma mais equilibrada, mantendo a viabilidade operacional da avaliação humana.

Com isso, a avaliação foi conduzida em dois domínios definidos no estudo (Domínio 1: Agendamento Clínico e Domínio 2: Doação Sangue), escolhidos para viabilizar uma análise detalhada sem perder diversidade de contexto. Em questões de rigor qualitativo, os selecionados são especialistas da área de Engenharia de Software, com experiência prática em artefatos ágeis, especialmente em requisitos e histórias de usuário, o que os qualifica para julgar critérios sob uma perspectiva alinhada ao desenvolvimento de software.

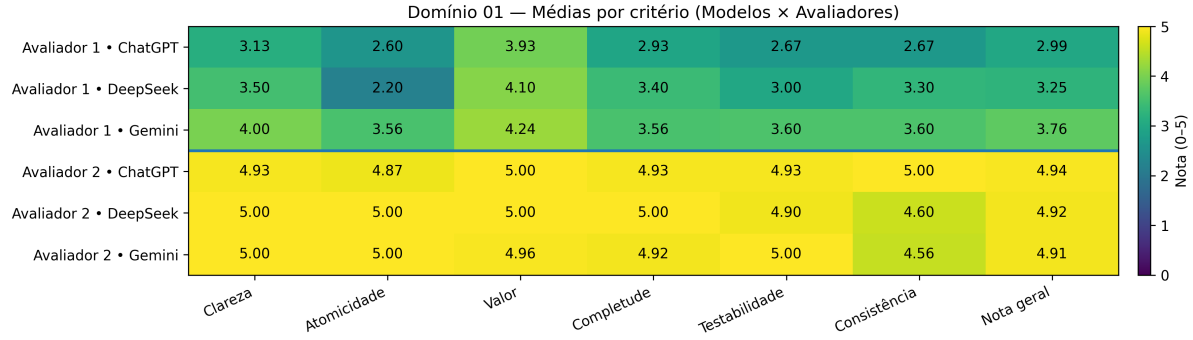
Para analisar a qualidade das histórias de usuário geradas, foram considerados critérios de avaliação adaptados do framework QUS, além de um escore composto, definido como a média dos critérios:

- **Clareza:** texto direto e compreensível, sem ambiguidades.
- **Atomicidade:** descreve um único objetivo/funcionalidade.
- **Valor:** deixa explícito o benefício para usuário/negócio.

– **Completude:** contém o essencial (ator, ação e objetivo/contexto).
– **Testabilidade:** permite definir critérios de aceitação verificáveis.
– **Consistência:** mantém coerência com regras e vocabulário do domínio.

A Figura 6 apresenta, para o Domínio 01, um *heatmap* das médias por critério atribuídas pelos dois avaliadores aos artefatos gerados pelos três modelos (*ChatGPT*, *DeepSeek* e *Gemini*). As colunas correspondem aos critérios de qualidade, e a média geral dos critérios, enquanto as linhas agrupam cada combinação Avaliador × Modelo. A escala cromática (0–5) facilita a leitura comparativa: tons mais claros indicam médias mais altas e tons mais escuros indicam médias mais baixas.

Figura 6 – Heatmap do Domínio 01



Fonte: Elaborada pela autora.

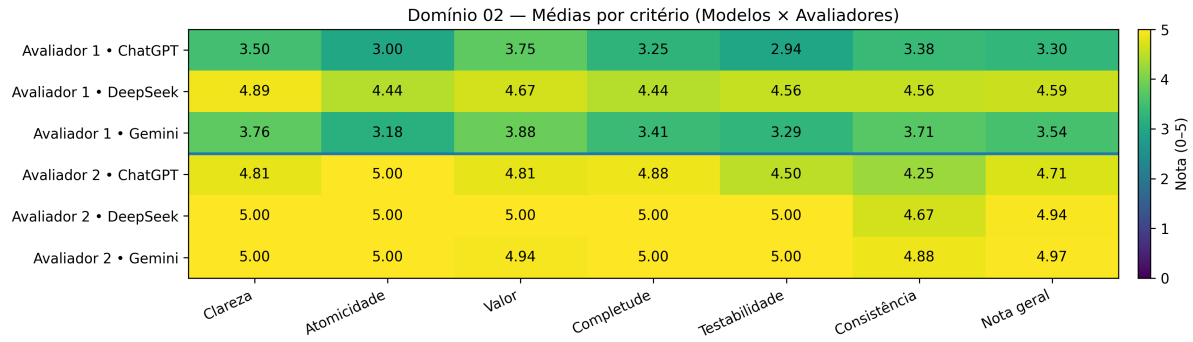
Observa-se uma diferença consideral entre os avaliadores, já que no que diz respeito ao Avaliador 1, suas médias se mostraram bem moderadas e apresentam maior variação entre modelos e critérios. Nesse recorte, o *Gemini* obteve a maior nota geral (3,76), seguido por *DeepSeek* (3,25) e *ChatGPT* (2,99). Como análise geral, para o Avaliador 1, o critério Valor foi consistentemente o mais alto nos três modelos (3,93 no *ChatGPT*; 4,10 no *DeepSeek*; 4,24 no *Gemini*), sugerindo que as histórias tendem a manter relevância funcional. Em comparação, o critério de Atomicidade aparece como o principal ponto fraco, especialmente para *ChatGPT* (2,60) e *DeepSeek* (2,20), indicando maior ocorrência de histórias com múltiplas intenções ou escopo agregado. Além disso, no *ChatGPT*, o critério de Testabilidade e Consistência se estabelecem entre as menores médias (2,67 em ambos), sugerindo maior presença de ambiguidades e/ou critérios de aceitação menos verificáveis.

O Avaliador 2, por sua vez, atribui notas altas em praticamente todos os critérios e modelos, resultando em notas gerais muito altas e com baixa dispersão (*ChatGPT* 4,94; *DeepSeek* 4,92; *Gemini* 4,91). Esse padrão indica uma avaliação globalmente mais positiva e

menos discriminativa entre os modelos. Ainda assim, o critério de Consistência é o critério que mais reduz as médias para o *DeepSeek* (4,60) e *Gemini* (4,56), enquanto o *ChatGPT* atinge 5,00 nesse critério; nos demais critérios, as médias variam pouco, aproximadamente entre 4,87 e 5,00.

No Domínio 02, o heatmap da Figura 7 mostra um padrão mais “separador"no Avaliador 1 e novamente notas positivas vindas do Avaliador 2.

Figura 7 – Heatmap do Domínio 02



Fonte: Elaborada pela autora.

Nesse sentido, o Avaliador 1 diferenciou bem os modelos ao destacar o *Deepseek* com uma nota geral positiva (4,59), em comparação às do *Gemini* (3,54) e *ChatGPT* (3,30). Como exemplificado, o *ChatGPT* apresentou as menores médias do avaliador, principalmente em critérios como Testabilidade (2,94) e Atomicidade (3,00), indicando histórias mais difíceis de validar por critérios objetivos e com maior chance de agrupar ações. O *Gemini* aparece em posição intermediária, com melhor desempenho em Valor (3,88) e Consistência (3,71).

Contudo, o Avaliador 2 atribuiu notas próximas a todos os modelos, ainda estabelecendo notas gerais altas (*ChatGPT* 4,71; *DeepSeek* 4,94; *Gemini* 4,97). O ponto que mais varia é o critério de Consistência: *ChatGPT* ficando com 4,25, *DeepSeek* com 4,67 e *Gemini* com 4,88, sugerindo que esse critério foi o que mais diversificou os modelos sob a ótica do Avaliador 2.

A partir do comportamento das notas atribuídas pelos especialistas, observa-se que parte das histórias apresenta observações recorrentes, principalmente nos critérios de atomicidade, testabilidade e consistência. Um padrão frequente é a presença de histórias que agregam mais de um objetivo na mesma formulação, misturando ações e responsabilidades, por exemplo, cadastrar e notificar, consultar e atualizar. Esse tipo de construção reduz a atomicidade e consequentemente dificulta o planejamento e a estimativa, além de tornar menos objetivos os critérios de aceitação.

Ademais, outro indício recorrente é a baixa testabilidade associada a descrições genéricas ou subjetivas, que não fornecem condições claras de verificação. Mesmo quando

a intenção do requisito é compreensível, a falta de regras mínimas como validações, estados, limites ou condições de sucesso reduz a possibilidade de extrair daí os testes ou critérios de aceitação.

Por fim, também se observam sinais de inconsistência com o domínio, como variações terminológicas e uso pouco estável de papéis e entidades. Esse problema enfraquece a consistência e aumenta a chance de interpretações divergentes entre avaliadores, especialmente quando a história é excessivamente genérica e pouco ancorada no vocabulário do contexto. Em conjunto, esses padrões sugerem que as histórias geradas podem ser úteis como ponto de partida, mas com frequência demandam revisão para se tornarem menores, verificáveis e alinhadas ao domínio antes de serem incorporadas ao *backlog*.

Dessa forma, os especialistas atribuíram notas de 1 a 5 por critério e uma nota geral (média dos critérios), avaliando os três modelos nos dois domínios. Em termos gerais, as histórias foram consideradas adequadas como ponto de partida, mas com necessidade frequente de refinamento antes de uso em *backlog*. Os critérios que mais apresentaram fragilidades recorrentes foram Atomicidade e Testabilidade, seguidos de Consistência. Em contraste, os critérios de Valor e Clareza tenderam a apresentar as maiores médias, indicando que o benefício pretendido costuma estar explícito, embora nem sempre de forma suficientemente específica para garantir verificação e decomposição correta da história.

5.4 Relação entre consistência textual observada e avaliação humana

Esta seção integra os dois eixos analisados no estudo: a consistência textual observada entre repetições do protocolo e a qualidade percebida por avaliadores humanos. O objetivo é examinar se modelos mais estáveis na geração também tendem a produzir histórias melhor avaliadas, ou se as dimensões capturam aspectos essencialmente distintos do fenômeno. A discussão enfatiza convergências e divergências entre as medidas.

5.4.1 QP4: Em que medida a consistência textual entre repetições se relaciona com a qualidade percebida por avaliadores humanos ao julgar as histórias geradas?

Dadas as duas vertentes de avaliação, por um lado, a mensuração automatizada da estabilidade semântica via similaridade do cosseno entre *embeddings*, e, por outro, a avaliação qualitativa de especialistas em Engenharia de Software, a ideia foi investigar a possibilidade

de estabelecer um paralelo entre consistência das repetições e a qualidade mensurada das histórias geradas. A motivação foi verificar se modelos mais estáveis, ao repetir o protocolo tendem, também, a produzir histórias melhor avaliadas, ou se as duas dimensões são amplamente independentes.

A análise conjunta dos resultados indicou que estabilidade e qualidade não são, necessariamente, dimensões convergentes. Em alguns cenários, modelos com maior consistência textual dos artefatos obtidos entre repetições não obtiveram as melhores avaliações humanas, enquanto modelos melhor avaliados não apresentaram maior repetibilidade. Isso ocorre porque as métricas capturam aspectos distintos: a consistência textual defendida mede o grau de padronização da interpretação do protocolo ao longo das execuções, enquanto a avaliação humana julga a adequação da história como requisito a partir de critérios como clareza, atomicidade, testabilidade e consistência com o domínio, conforme a adaptação de critérios do *framework* QUS.

Esse contraste evidencia um possível conflito prático, onde um modelo pode ser altamente previsível entre execuções e, ainda assim, produzir histórias com fragilidades recorrentes, como formulações genéricas, critérios de aceitação pouco verificáveis ou escopo excessivamente agregado. Nesses casos, a estabilidade observada reflete uniformidade do padrão de geração, e não necessariamente qualidade do artefato. Em sentido oposto, maior variabilidade entre repetições pode resultar de um comportamento mais exploratório, como a inclusão de alternativas de formulação, casos adicionais ou reorganização do escopo, o que pode enriquecer o conjunto de histórias produzidas, embora reduza a previsibilidade do processo.

Além disso, quando as avaliações humanas atribuem notas muito altas e semelhantes a todos os modelos, o poder discriminatório da comparação diminui. Mesmo havendo variações na consistência textual, a qualidade avaliada pode permanecer praticamente inalterada, dificultando a identificação de alinhamento entre as medidas. Em síntese, consistência semântica do texto e qualidade percebida devem ser tratadas como dimensões diferentes e complementares: a primeira avalia a regularidade do processo de geração sob repetição do protocolo, enquanto a segunda avalia a adequação do artefato como requisito sob critérios qualitativos.

5.5 Discussões e Implicações

Este estudo investigou consistência textual de histórias de usuário geradas por diferentes modelos de LLMs ao repetir um mesmo protocolo com base na coerência entre

repetições e contou com a percepção de qualidade por agentes externos e familiarizados na aplicação ágil na Engenharia de Software. Em conjunto, os resultados mostraram que estabilidade de geração e qualidade do artefato não são dimensões necessariamente convergentes, e que a repetibilidade observada é bastante dependente do domínio e do estilo de geração de cada modelo.

Do ponto de vista metodológico, a estratégia de representar histórias por *embeddings* e comparar versões por similaridade do cosseno aproxima este trabalho de abordagens consolidadas em Engenharia de Requisitos que utilizam proximidade semântica para comparação de sentenças. Em estudos como os de Das *et al.* (2021), é proposto modelos de *sentence embeddings* para requisitos e discutem também o uso do cosseno como medida de proximidade semântica entre sentenças de requisitos, relatando ganhos em tarefas de similaridade ao treinar *embeddings* no domínio de ER. Entretanto, há uma diferença importante no objetivo: enquanto Das *et al.* (2021) exploram similaridade para apoiar tarefas como comparação e recuperação de requisitos, neste estudo a similaridade foi empregada para estimar repetibilidade do processo de gerar as histórias de usuário. Isso traz implicações diretas para a interpretação dos resultados: a consistência não é influenciada apenas pela estruturação escrita, mas também por questões de nível de detalhe e precisão, recorte funcional visando o uso estável em um ambiente real de desenvolvimento.

Ademais, os resultados de QP2–QP4 reforçam que não há um desempenho totalmente superior: tanto na literatura quanto neste estudo, o comportamento varia por domínio e objetivo, e diferentes abordagens podem se destacar dependendo do contexto (Kabootari *et al.*, 2025). E também os achados mostram que consistência semântica/textual e qualidade percebida medem coisas diferentes: a primeira indica quão previsível e reprodutível é a geração ao repetir o protocolo, enquanto a segunda avalia se as histórias são bons requisitos. Por isso, um modelo pode ser mais estável sem necessariamente produzir histórias melhores, e um modelo melhor avaliado pode variar mais entre repetições; na prática, a consistência deve ser interpretada como evidência de reprodutibilidade operacional, não como sinônimo de qualidade intrínseca.

Além disso, partindo do que é de necessidade da indústria, os resultados sugerem que a escolha do modelo deve ser orientada pelo objetivo de uso, separando os cenários em que a previsibilidade é central, por exemplo quando a automação é recorrente, em casos de auditoria e padronização de *backlog*. Mesmo quando a consistência entre repetições é alta, a avaliação humana indica fragilidades recorrentes, o que reforça a necessidade de uma etapa explícita de

refino antes de incorporar histórias ao *backlog*. Nesse sentido, também é recomendável adotar mecanismos automáticos de pós-processamento para filtrar problemas, inspirados em tarefas já padrões de Engenharia de Requisitos, como classificação e detecção de ambiguidades, onde a viabilidade com diferentes estratégias de *embeddings* e modelos é discutida por Kabootari *et al.* (2025).

A cerca dessa lógica, para a academia, o estudo reforça a necessidade de tratar estabilidade como um eixo experimental próprio ao investigar as LLMs em Engenharia de Requisitos, indo além de avaliações pontuais de qualidade textual. Uma linha direta de avanço metodológico é avaliar o uso de *embeddings* especializados em requisitos e adotar isso para histórias de usuário, conforme sugerido por Das *et al.* (2021), para reduzir ruído e aumentar fidelidade semântica nas comparações. Além disso, os achados indicam que pesquisas futuras devem modelar separadamente as métricas de qualidade do processo (reprodutibilidade, cobertura de comparabilidade, variância entre execuções) e métricas de qualidade do artefato, investigando quando essas dimensões convergem ou divergem conforme domínio, estratégia de *prompting* e configuração do modelo.

Em síntese, ao dialogar com estudos que utilizam *embeddings* para similaridade em requisitos e com evidências de que desempenho varia por tarefa/dado em ER, este trabalho sugere que a adoção de LLMs em histórias de usuário deve ser orientada por dois objetivos distintos: reprodutibilidade do processo e adequação do artefato. A consistência textual observada oferece evidência útil sobre previsibilidade operacional, enquanto a avaliação humana evidencia onde o refinamento é mais necessário para transformar histórias geradas em requisitos efetivamente utilizáveis na prática.

5.6 Ameaças à validade

- **Validade da conclusão:** os resultados dependem do tamanho do conjunto analisado e de como os valores foram resumidos. Como foi usado médias por modelo e domínio, casos muito altos/baixos podem "sumir" na média, e diferenças pequenas entre modelos podem não ser tão sólidas sem testes estatísticos. Além disso, consistência automática e nota humana não vão necessariamente coincidir: um modelo pode ser bem consistente ao repetir o mesmo problema.
- **Validade interna:** há variações próprias das plataformas de LLMs (aleatoriedade, atualizações e estado de sessão) que não são totalmente controláveis. Embora o estudo tenha

tomado cuidados para separar o contexto entre repetições, não é possível garantir que todas as execuções foram 100% independentes. Também houve uma limitação importante: as entrevistas simuladas foram respondidas pela própria autora como *stakeholder*, o que reduz o realismo por não envolver demandas reais de projetos e stakeholders. Além disso, como as repetições abrangeram o *pipeline* completo, incluindo a elicitação de requisitos, variações nessa etapa podem ter alterado o conjunto de histórias produzidas e, consequentemente, reduzido o número de itens diretamente comparáveis entre interações. Além disso, diferentes perfis de *stakeholders* podem levar a requisitos com graus distintos de detalhamento, vocabulário e priorização; essa variação na elicitação tende a se propagar para a geração das histórias, afetando comparabilidade e consistência entre repetições.

– **Validade de construto:** *embeddings* e similaridade do cosseno capturam proximidade de sentido, mas não medem diretamente qualidade ou aspectos práticos, como testabilidade e viabilidade. Além disso, a avaliação por especialistas não cobriu todos os artefatos: como foram geradas 468 histórias, apenas uma amostra pôde ser avaliada. A própria avaliação humana também foi limitada pelo número de avaliadores, dois especialistas no caso, o que reduz a robustez da comparação e aumenta a sensibilidade a diferenças individuais de rigor e interpretação dos critérios. Portanto, os resultados qualitativos refletem esse recorte e os critérios adotados.
– **Validade externa:** os achados podem não se generalizar para outros contextos, pois dependem dos domínios escolhidos, do protocolo (3Cs), dos modelos avaliados e do ambiente controlado. Em projetos reais, há negociação, refinamento e restrições organizacionais que podem alterar o desempenho observado. Por fim, como as gerações foram feitas pela interface (e não por API), houve menos automação e controle de parâmetros, o que pode dificultar reproduzir o experimento e escalar o procedimento.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho investigou o uso de LLMs na geração de histórias de usuário sob um protocolo que buscou na repetição um parâmetro de reprodutibilidade, combinando duas leituras complementares: (i) consistência semântica entre repetições via *embeddings* e similaridade do cosseno e (ii) qualidade percebida por especialistas com critérios inspirados no QUS. Os resultados reforçam o potencial dessas tecnologias como suporte à Engenharia de Requisitos em contextos ágeis, sobretudo como mecanismo de geração e triagem inicial de artefatos (Sharma; Tripathi, 2025).

No eixo de consistência (QP1–QP2), observou-se repetibilidade moderada a alta, porém sensível a modelo e domínio. *ChatGPT* e *DeepSeek* apresentaram um comportamento mais previsível, enquanto o *Gemini* foi mais exploratório, gerando número bem mais consideráveis de histórias de usuário, mas menos uniformes entre repetições. Como a repetição abrangeu o *pipeline* completo, incluindo elicitação, parte da variabilidade decorreu também de mudanças no conjunto de artefatos gerados, reduzindo a quantidade de itens diretamente comparáveis.

Na avaliação humana (QP3), as histórias foram consideradas úteis como ponto de partida, mas com necessidade frequente de refinamento. Já na relação entre consistência e qualidade percebida (QP4), não se identificou alinhamento direto: consistência mede regularidade do processo, enquanto especialistas avaliam adequação do produto, de modo que as medidas devem ser interpretadas como complementares.

6.1 Trabalhos Futuros

- Repetir o experimento com base fixa de requisitos por domínio, e múltiplas gerações por modelo, isolando o efeito da etapa de geração de histórias, a fim de não gerar uma dispersão não controlável de quantidade de artefatos a serem postos em uma análise comparativa.
- Executar via API e registrar parâmetros e contexto para maior controle e reprodutibilidade.
- Ampliar a avaliação humana (mais avaliadores, mais domínios e amostra maior) e reportar concordância entre avaliadores.
- Realizar análise por Histórias de Usuário (item a item), alinhando escores humanos e de consistência, evitando apenas agregação por médias.
- Testar diferentes variações de técnicas de *prompts* de maneira independente, ou seja, analisar que protocolo soa mais adequada ao objetivo de elicitação..

REFERÊNCIAS

ALHANAHNAH, M.; HASAN, M. R.; XU, L.; BAGHERI, H. An empirical evaluation of pre-trained large language models for repairing declarative formal specifications. **Empirical Software Engineering**, Springer, v. 30, n. 5, p. 149, 2025.

BANO, M.; ZOWGHI, D.; FERRARI, A.; SPOLETINI, P.; DONATI, B. Teaching requirements elicitation interviews: an empirical study of learning from mistakes. **Requirements Engineering**, Springer, v. 24, n. 3, p. 259–289, 2019.

BECK, K.; BEEDLE, M.; BENNEKUM, A. van; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, D.; JEFFRIES, R.; KERN, J.; MARICK, B.; MARTIN, R. C.; MELLOR, S.; SCHWABER, K.; SUTHERLAND, J.; THOMAS, D. **Manifesto for Agile Software Development**. 2001. Disponível em: https://agilemanifesto.org/. Acesso em: 10 maio 2025.

BERRYMAN, J.; ZIEGLER, A. **Prompt Engineering for LLMs**: The art and science of building large language model–based applications. [*S. l.*]: O'Reilly Media, 2024. ISBN 9781098156152.

BIN, X.; XIAOHU, Y.; ZHIJUN, H.; MADDINENI, S. R. Extreme programming in reducing the rework of requirements change. In: **Canadian Conference on Electrical and Computer Engineering**. [*S. l.*: *s. n.*], 2004. p. 1567–1570.

BONESS, K.; HARRISON, R. Goal sketching: Towards agile requirements engineering. In: **International Conference on Software Engineering Advances**. [*S. l.*: *s. n.*], 2007. p. 1–6.

BROCKENBROUGH, A.; FEILD, H.; SALINAS, D. Exploring llms impact on student-created user stories and acceptance testing in software development. In: **Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 2**. [*S. l.*: *s. n.*], 2025. p. 1401–1402.

BROWN, T. B.; MANN, B.; RYDER, N.; SUBBIAH, M.; KAPLAN, J.; DHARIWAL, P.; NEELAKANTAN, A.; SHYAM, P.; SASTRY, G.; ASKELL, A.; AGARWAL, S.; HERBERT-VOSS, A.; KRUEGER, G.; HENIGHAN, T.; CHILD, R.; RAMESH, A.; ZIEGLER, D. M.; WU, J.; WINTER, C.; HESSE, C.; CHEN, M.; SIGLER, E.; LITWIN, M.; GRAY, S.; CHESS, B.; CLARK, J.; BERNER, C.; MCCANDLISH, S.; RADFORD, A.; SUTSKEVER, I.; AMODEI, D. Language models are few-shot learners. In: **Proceedings of the 34th International Conference on Neural Information Processing Systems**. Red Hook, NY, USA: Curran Associates Inc., 2020. (NIPS '20). ISBN 9781713829546.

CABRERO-DANIEL, B.; HERDA, T.; PICHLER, V.; EDER, M. **Exploring Human-AI Collaboration in Agile**: Customised llm meeting assistants. 2024. Disponível em: https://arxiv.org/abs/2404.14871.

CARLETON, A.; KLEIN, M. H.; ROBERT, J. E.; HARPER, E.; CUNNINGHAM, R. K.; NIZ, D. de; FOREMAN, J. T.; GOODENOUGH, J. B.; HERBSLEB, J. D.; OZKAYA, I.; SCHMIDT, D. C. Architecting the future of software engineering. **Computer**, v. 55, n. 9, p. 89–93, 2022.

CHEN, B.; ZHANG, Z.; LANGRENé, N.; ZHU, S. Unleashing the potential of prompt engineering for large language models. **Patterns**, Elsevier BV, v. 6, n. 6, p. 101260, jun. 2025. ISSN 2666-3899. Disponível em: http://dx.doi.org/10.1016/j.patter.2025.101260.

COHN, M. **User Stories Applied**: For agile software development. USA: Addison Wesley Longman Publishing Co., Inc., 2004.

DALPIAZ, F.; FERRARI, A.; FRANCH, X.; PALOMARES, C. Natural language processing for requirements engineering: The best is yet to come. **IEEE Software**, IEEE, v. 35, p. 115–119, 2018.

DAS, S.; DEB, N.; CORTESI, A.; CHAKI, N. Sentence embedding models for similarity detection of software requirements. **SN Computer Science**, Springer, v. 2, n. 2, p. 69, 2021.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In: **Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)**. [*S. l.*: *s. n.*], 2019. p. 4171–4186.

EBERLEIN, A.; CESAR, S. d. P. L. J. Agile requirements definition: A view from requirements engineering. In: **Proceedings of the International Workshop on Time Constrained Requirements Engineering**. [*S. l.*: *s. n.*], 2002.

EMMERICH, W. Working with user stories. In: **Agile Requirements Engineering Workshop**. [*S. l.*: *s. n.*], 2011.

FERREIRA, A. M. S.; SILVA, A. R. da; PAIVA, A. C. R. Towards the art of writing agile requirements with user stories, acceptance criteria, and related constructs. In: **Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)**. [*S. l.*]: SciTePress, 2022. p. 477–484.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns**: Elements of reusable object-oriented software. [*S. l.*]: Pearson Deutschland GmbH, 1995.

Google DeepMind. **Gemini 1.5 Pro**. 2024. Disponível em: https://deepmind.google/models/gemini/pro/. Acesso em: 01 jun. 2025.

HECK, P.; ZAIDMAN, A. A quality framework for agile requirements: A practitioner's perspective. **ArXiv**, abs/1406.4692, 2014. Disponível em: https://api.semanticscholar.org/CorpusID:15996575.

HEMMAT, A. *et al.* Research directions for using llm in software requirement engineering: a systematic review. **Frontiers in Computer Science**, v. 7, p. 1519437, 2025.

IEEE Computer Society. **IEEE Recommended Practice for Software Requirements Specifications**. 1994. IEEE Std 830-1993. DOI: 10.1109/IEEESTD.1994.121431.

JATNIKA, D.; BIJAKSANA, M. A.; SURYANI, A. A. Word2vec model analysis for semantic similarities in english words. **Procedia Comput. Sci.**, Elsevier Science Publishers B. V., NLD, v. 157, n. C, p. 160–167, jan. 2019. ISSN 1877-0509. Disponível em: https://doi.org/10.1016/j.procs.2019.08.153.

JEFFRIES, R. **Card, Conversation, Confirmation**. 2001. Disponível em: https://ronjeffries.com/xprog/articles/expcardconversationconfirmation/. Acesso em: 01 jun. 2025.

KABOOTARI, M.; ABDEAHAD, Y.; KHEIRKHAH, Y.; KHEIRKHAH, E. Enhancing software requirements classification: a comparative study of deep learning model integration with embedding techniques. **Computing**, Springer, v. 107, n. 10, p. 191, 2025.

KASSAB, M. An empirical study on the requirements engineering practices for agile software development. In: **40th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)**. [*S. l.*]: IEEE, 2014. p. 254–261.

KORN, A.; GORSCH, S.; VOGELSANG, A. Llmrei: Automating requirements elicitation interviews with llms. **arXiv preprint arXiv:2507.02564**, 2025.

LAGERBERG, L.; SKUDE, T. **The Impact of Agile Principles and Practices on Large-Scale Software Development Projects**. [*S. l.*], 2013.

LEI, H.; LAI, W.; FEASTER, W.; CHANG, A. C. Artificial intelligence and agile project management. In: **Intelligence-Based Cardiology and Cardiac Surgery**. [*S. l.*]: Elsevier, 2024. p. 401–405.

LIAO, H. Deepseek large-scale model: Technical analysis and development prospect. **Journal of Computer Science and Electrical Engineering**, v. 7, n. 1, p. 33–37, 2025.

LIN, C.-Y. Rouge: A package for automatic evaluation of summaries. In: **Text summarization branches out**. [*S. l.*: *s. n.*], 2004. p. 74–81.

LINDLAND, O.; SINDRE, G.; SOLVBERG, A. Understanding quality in conceptual modeling. **IEEE Software**, v. 11, n. 2, p. 42–49, 1994.

LIU, P.; YUAN, W.; FU, J.; JIANG, Z.; HAYASHI, H.; NEUBIG, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 55, n. 9, jan. 2023. ISSN 0360-0300. Disponível em: https://doi.org/10.1145/3560815.

LONG, L.; WANG, R.; XIAO, R.; ZHAO, J.; DING, X.; CHEN, G.; WANG, H. On llms-driven synthetic data generation, curation, and evaluation: A survey. In: KU, L.-W.; MARTINS, A.; SRIKUMAR, V. (Ed.). **Findings of the Association for Computational Linguistics: ACL 2024**. Bangkok, Thailand and virtual meeting: Association for Computational Linguistics, 2024. p. 11065–11082. Disponível em: https://aclanthology.org/2024.findings-acl.658.

LUCASSEN, G.; DALPIAZ, F.; WERF, J. M.; BRINKKEMPER, S. The use and effectiveness of user stories in practice. In: **Proceedings of the 22nd International Working Conference on Requirements Engineering: Foundation for Software Quality**. [*S. n.*], 2016. p. 205–222. Disponível em: https://doi.org/10.1007/978-3-319-30282-9_14.

LUCASSEN, G.; DALPIAZ, F.; WERF, J. M. E. M. van der; BRINKKEMPER, S. Forging high-quality user stories: Towards a discipline for agile requirements. In: **2015 IEEE 23rd International Requirements Engineering Conference (RE)**. [*S. l.*]: IEEE, 2015. p. 126–135.

LUCASSEN, G.; DALPIAZ, F.; WERF, J. M. Van der; BRINKKEMPER, S. Improving agile requirements: The quality user story framework and tool. **Requirements Engineering**, Springer, v. 21, p. 383–403, 2016.

MARTINELLI, S.; ARANHA, M.; ZAINA, L.; MEIRELLES, P.; SILVA, F. UX Requirements Matters: Guidelines to Support Software Teams on the Writing of Acceptance Criteria. In: **Proceedings of the XXXVI Brazilian Symposium on Software Engineering (SBES)**. [*S. l.*: *s. n.*], 2022. p. 398–408.

MASTROPAOLO, A.; PASCARELLA, L.; GUGLIELMI, E.; CINISELLI, M.; SCALABRINO, S.; OLIVETO, R.; BAVOTA, G. On the robustness of code generation techniques: An empirical study on github copilot. In: IEEE. **2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)**. [*S. l.*], 2023. p. 2149–2160.

METH, H.; BRHEL, M.; MAEDCHE, A. The state of the art in automated requirements elicitation. **Information and Software Technology**, v. 55, n. 10, p. 1695–1709, 2013.

MNKANDLA, E.; DWOLATZKY, B. A survey of agile methodologies. **Transactions of the South African Institute of Electrical Engineers**, v. 95, n. 4, p. 236–247, 2004.

NGUYEN-DUC, A.; CABRERO-DANIEL, B.; PRZYBYLEK, A.; ARORA, C.; KHANNA, D.; HERDA, T.; RAFIQ, U.; MELEGATI, J.; GUERRA, E.; KEMELL, K.-K.; SAARI, M.; ZHANG, Z.; LE, H.; QUAN, T.; ABRAHAMSSON, P. **Generative Artificial Intelligence for Software Engineering – A Research Agenda**. 2023. Disponível em: https://arxiv.org/abs/2310.18648.

NORTH, D. Introducing behaviour driven development. **Better Software Magazine**, 2006. Disponível em: https://dannorth.net/introducing-bdd/. Acesso em: 22 maio 2025.

OpenAI. **Introducing GPT-5**. 2024. Disponível em: https://openai.com/pt-BR/index/introducing-gpt-5/. Acesso em: 01 out. 2025.

OSWAL, J. U.; KANAKIA, H. T.; SUKTEL, D. Transforming software requirements into user stories with gpt-3.5 -: An ai-powered approach. In: **2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)**. [*S. l.*: *s. n.*], 2024. p. 913–920.

PAPINENI, K.; ROUKOS, S.; WARD, T.; ZHU, W.-J. Bleu: a method for automatic evaluation of machine translation. In: . USA: Association for Computational Linguistics, 2002. (ACL '02), p. 311–318. Disponível em: https://doi.org/10.3115/1073083.1073135.

PEREIRA, R. M.; MALUCELLI, A.; REINEHR, S. From real-time conversation to user story: Leveraging agile requirements through llm. In: SBC. **Simpósio Brasileiro de Engenharia de Software (SBES)**. [*S. l.*], 2025. p. 657–663.

RAHMAN, T.; ZHU, Y.; MAHA, L.; ROY, C.; ROY, B.; SCHNEIDER, K. Take loads off your developers: Automated user story generation using large language model. In: **2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)**. Flagstaff, AZ, USA: IEEE, 2024. p. 791–801.

RAMESH, B.; BASKERVILLE, R.; CAO, L. Agile requirements engineering practices and challenges: An empirical study. **Information Systems Journal**, v. 20, n. 5, p. 449–480, 2010.

SARSA, S.; DENNY, P.; HELLAS, A.; LEINONEN, J. Automatic generation of programming exercises and code explanations using large language models. In: **Proceedings of the 2022 ACM Conference on International Computing Education Research**. New York: ACM, 2022. p. 27–43.

SCHMIDT, D. C.; STAL, M.; ROHNERT, H.; BUSCHMANN, F. **Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects**. [*S. l.*]: John Wiley & Sons, 2013.

SCHöN, E. M.; THOMASCHEWSKI, J.; ESCALONA, M. J. Agile requirements engineering: A systematic literature review. **Computer Standards & Interfaces**, v. 49, p. 79–91, 2017.

SHARMA, A.; TRIPATHI, A. K. Evaluating user story quality with llms: a comparative study. **Journal of Intelligent Information Systems**, v. 63, p. 1423–1451, 2025. Disponível em: https://doi.org/10.1007/s10844-025-00939-3.

SILVA, A. R. D. Linguistic patterns and linguistic styles for requirements specification (i): An application case with the rigorous rsl/business-level language. In: **Proceedings of the ACM International Conference**. [*S. l.*: *s. n.*], 2017. (ACM Int. Conf. Proceeding Ser., I), p. 1–27.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. In: **Advances in Neural Information Processing Systems (NeurIPS)**. Curran Associates, Inc., 2017. v. 30. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

WAKE, B. **INVEST in Good Stories, and SMART Tasks**. 2003. Disponível em: http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/. Acesso em: 11 jun. 2025.

WEI, J.; WANG, X.; SCHUURMANS, D.; BOSMA, M.; ICHTER, b.; XIA, F.; CHI, E.; LE, Q. V.; ZHOU, D. Chain-of-thought prompting elicits reasoning in large language models. In: KOYEJO, S.; MOHAMED, S.; AGARWAL, A.; BELGRAVE, D.; CHO, K.; OH, A. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2022. v. 35, p. 24824–24837. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

WHITE, J.; FU, Q.; HAYS, S.; SANDBORN, M.; OLEA, C.; GILBERT, H.; ELNASHAR, A.; SPENCER-SMITH, J.; SCHMIDT, D. C. A prompt pattern catalog to enhance prompt engineering with chatgpt. In: **Proceedings of the 30th Conference on Pattern Languages of Programs (PLoP '23)**. Monticello, IL, USA: The Hillside Group, 2023. (PLoP '23), p. 5:1–5:31. ISBN 9781941652190.

WILDT, D.; MOURA, D.; LACERDA, G.; HELM, R. **eXtreme Programming**: Práticas para o dia a dia no desenvolvimento ágil de software. São Paulo: Casa do Código, 2015. 162 páginas; disponível em PDF e e-book. ISBN 978-8555191077.

YAMANI, A.; BASLYMAN, M.; AHMED, M. Leveraging llms for user stories in ai systems: Ustai dataset. In: **Proceedings of the 21st International Conference on Predictive Models and Data Analytics in Software Engineering**. New York, NY, USA: Association for Computing Machinery, 2025. (PROMISE '25), p. 21–30. ISBN 9798400715945. Disponível em: https://doi.org/10.1145/3727582.3728689.

YAO, Y.; DUAN, J.; XU, K.; CAI, Y.; SUN, Z.; ZHANG, Y. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. **High-Confidence Computing**, Elsevier, p. 100211, 2024.

ZHANG, Z.; MIRHASSANI, S.; AHMAD, A.; ZIMMERMANN, T.; TRAON, Y. L. Llm-based agents for automating the enhancement of user story quality: An early report. In: **International Conference on Agile Software Development (XP 2024)**. Cham: Springer Nature Switzerland, 2024. p. 117–126.

Proceedings of the 25th international conference on agile software development (xp 2024). In: ŠMITE, D.; GUERRA, E.; WANG, X.; MARCHESI, M.; GREGORY, P. (Ed.). **Agile Processes in Software Engineering and Extreme Programming**. Bozen-Bolzano, Italy: Springer Nature, 2024. p. 117–125. Open access book, 197 pages in total. Disponível em: https://link.springer.com/book/10.1007/978-3-031-60650-9.