APLICAÇÃO DE MODELOS DE LINGUAGEM DE GRANDE ESCALA NO APOIO À GARANTIA DE QUALIDADE DE SOFTWARE

Artur Dias Ribeiro[1], Luiz Ricardo Fiera[2]

Resumo: A garantia de qualidade de software é essencial para reduzir retrabalho e aumentar a confiabilidade dos sistemas. Nesse contexto, modelos de linguagem de grande escala surgem como ferramentas promissoras para apoiar a análise de requisitos e a geração de casos de teste. Este trabalho avalia o uso do ChatGPT *Thinking* e do Gemini *Flash* como apoio ao analista de qualidade de software. Foi conduzido um experimento com dez histórias de usuário do sistema OrangeHRM, de código aberto. As histórias foram submetidas aos modelos por meio de comandos padronizados, sem fornecimento de exemplos prévios, solicitando análise de requisitos, identificação de ambiguidades, geração de casos de teste, identificação de riscos, cenários de exceção, priorização e avaliação de prontidão. As respostas foram avaliadas por uma rubrica estruturada baseada em cinco critérios: cobertura funcional, relevância dos testes, clareza textual, cenários alternativos e aplicabilidade prática. Os resultados demonstram desempenho superior do ChatGPT *Thinking*. Destacando-se pela profundidade técnica e maior exploração de cenários alternativos, enquanto o Gemini *Flash* apresentou respostas mais concisas e claras. Conclui-se que os modelos de linguagem de grande escala constituem ferramentas auxiliares eficazes para as etapas iniciais da qualidade de software. Contudo, os achados reforçam a necessidade de supervisão e validação humana, especialmente em contextos que envolvem regras de negócio, segurança e requisitos ambíguos.

Palavras-chave: Garantia de Qualidade de Software. Testes de Software. Modelos de Linguagem de Grande Escala. Análise de Requisitos. Geração de Casos de Teste.

ABSTRACT: Software quality assurance is essential for reducing rework and increasing system reliability. In this context, large language models emerge as promising tools to support requirements analysis and test case generation. This study evaluates the use of ChatGPT *Thinking* and Gemini *Flash* as support tools for software quality analysts. An

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), Criciúma - Santa Catarina - Brasil. arturdias1010@unesc.net

[2]Professor Orientador, Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), Criciúma - Santa Catarina - Brasil. ricardo.fiera@gmail.com

experiment was conducted using ten user stories from the open-source OrangeHRM system. The stories were submitted to the models through standardized commands, without providing previous examples, requesting requirements analysis, ambiguity identification, test case generation, risk identification, exception scenarios, prioritization, and readiness assessment. The responses were evaluated using a structured rubric based on five criteria: functional coverage, test relevance, textual clarity, alternative scenarios, and practical applicability. The results demonstrate superior performance by ChatGPT *Thinking*. The former stood out for its technical depth and broader exploration of alternative scenarios, while Gemini *Flash* presented more concise and clear responses. It is concluded that large language models are effective auxiliary tools for the initial stages of software quality assurance. However, the findings reinforce the need for human supervision and validation, especially in contexts involving business rules, security, and ambiguous requirements.

Keywords: Software Quality Assurance. Software Testing. Large Language Models. Requirements Analysis. Test Case Generation.

1 INTRODUÇÃO

A garantia de qualidade de software constitui um aspecto crítico na indústria de Tecnologia da Informação (TI). De acordo com Khan e Khan (2014), os testes representam uma etapa fundamental no ciclo de vida do desenvolvimento de sistemas, sendo essenciais para garantir sua qualidade e confiabilidade. A busca por métodos e ferramentas mais eficazes para assegurar a excelência dos produtos de software tem sido uma preocupação constante das organizações que atuam nesse setor (GAROUSI; MäNTYLä, 2016). Nesse contexto, os modelos de linguagem de grande escala, do inglês *Large Language Models* (LLM), como o ChatGPT *Thinking*, da OpenAI, e o Gemini *Flash*, da Google, surgem como ferramentas inovadoras, impulsionadas por avanços em Processamento de Linguagem Natural (PLN), com potencial para apoiar e aprimorar atividades relacionadas à Garantia de Qualidade de Software.

A rápida evolução dos *LLM*, uma categoria de sistemas baseados em Inteligência Artificial (IA), tem gerado grande expectativa quanto ao seu uso em diferentes atividades da Engenharia de Software, especialmente por sua capacidade de interpretar instruções em linguagem natural, analisar descrições textuais, identificar inconsistências e gerar artefatos técnicos. No campo da qualidade de software, essas ferramentas podem auxiliar profissionais de Garantia de Qualidade, do inglês *Quality Assurance* (QA), na análise de requisitos, identificação de ambiguidades, criação de cenários de teste, geração de casos de teste, levantamento de riscos e priorização de validações. Tais ativi-

dades são relevantes porque requisitos ambíguos, incompletos ou inconsistentes podem comprometer a qualidade dos testes produzidos, gerar retrabalho e reduzir a eficácia da validação do sistema.

A literatura recente já apresenta estudos que investigam o uso de *LLM* em tarefas relacionadas a testes de software, geração de casos de teste e Engenharia de Requisitos. Guilherme e Vincenzi (2023), por exemplo, realizaram uma investigação inicial sobre a capacidade do ChatGPT na geração de testes unitários, observando seu potencial em comparação com ferramentas tradicionais de automação. Schäfer et al. (2023) também avaliaram empiricamente o uso de *LLM* para geração automatizada de testes unitários, identificando bons resultados em termos de cobertura, embora tenham destacado limitações na exploração de cenários mais complexos. De forma semelhante, Tang et al. (2023) compararam testes gerados pelo ChatGPT com testes produzidos por técnicas baseadas em busca, apontando diferenças relevantes em aspectos como corretude, legibilidade, cobertura e capacidade de detecção de defeitos.

Além dos estudos voltados à geração de testes unitários, revisões recentes têm ampliado a compreensão sobre o papel dos *LLM* em atividades de teste e qualidade de software. Wang et al. (2024) apresentaram um panorama sobre o uso de *LLM* em testes de software, destacando aplicações como geração de casos de teste, reparo de programas e identificação de defeitos. Celik e Mahmoud (2025), por sua vez, discutiram possibilidades e desafios da geração automatizada de casos de teste com *LLM*, ressaltando que esses modelos podem reduzir o esforço inicial de elaboração de artefatos, mas ainda exigem validação humana quanto à precisão, cobertura e adequação técnica das respostas produzidas.

No campo da Engenharia de Requisitos, também há pesquisas que demonstram o potencial dos *LLM* como ferramentas de apoio. Ronanki, Berger e Horkoff (2023) investigaram o uso do ChatGPT em processos de elicitação de requisitos, indicando que o modelo pode contribuir para a descoberta, compreensão e documentação de necessidades dos usuários. Marques, Silva e Bernardino (2024) apresentaram uma revisão sobre o uso do ChatGPT na Engenharia de Requisitos, destacando sua utilidade na identificação de ambiguidades, inconsistências e lacunas em especificações. De forma complementar, Almeida et al. (2025) avaliaram o desempenho de *LLM* na geração de requisitos a partir de entrevistas de elicitação, evidenciando diferenças entre modelos em tarefas relacionadas à produção e refinamento de requisitos.

Apesar desses avanços, observa-se que grande parte dos estudos existentes concentra-se na geração de testes unitários, na análise a partir de código-fonte ou em revisões teóricas sobre o tema. Ainda há menor ênfase em avaliações empíricas com-

parativas que investiguem o uso de diferentes *LLM* no apoio integrado às atividades do Analista de QA, considerando simultaneamente análise de histórias de usuário, identificação de ambiguidades, geração de cenários e casos de teste, levantamento de riscos, priorização e avaliação da prontidão dos requisitos para desenvolvimento. Além disso, poucos trabalhos realizam comparações diretas entre modelos comerciais de alto desempenho, como o ChatGPT *Thinking* e o Gemini *Flash*, utilizando histórias de usuário baseadas em funcionalidades reais de um sistema *open source*.

Diante dessa lacuna, o presente estudo busca contribuir para a área ao realizar uma análise experimental comparativa entre o ChatGPT *Thinking* e o Gemini *Flash*, utilizando histórias de usuário baseadas em funcionalidades reais do sistema *open source* OrangeHRM. A proposta é investigar de que forma esses modelos podem apoiar o analista de qualidade de software em atividades relacionadas à análise de requisitos e à geração de artefatos de teste, considerando seu potencial para identificar ambiguidades, sugerir cenários de validação, levantar riscos e apoiar a tomada de decisão no processo de Garantia de Qualidade de Software.

Desse modo, este trabalho tem como objetivo produzir evidências empíricas sobre o potencial, as limitações e a aplicabilidade prática desses modelos como ferramentas auxiliares no processo de Garantia de Qualidade de Software, comparando sistematicamente o comportamento das duas ferramentas em tarefas cotidianas do analista de testes.

2 MATERIAIS E MÉTODOS

Este capítulo descreve os procedimentos metodológicos adotados para a realização do estudo comparativo entre os modelos de linguagem de grande escala. Apresentam-se os detalhes do ambiente experimental, a estrutura dos comandos utilizados, bem como a rubrica técnica empregada para a avaliação qualitativa e quantitativa das respostas geradas pelas ferramentas.

2.1 PROCEDIMENTOS EXPERIMENTAIS

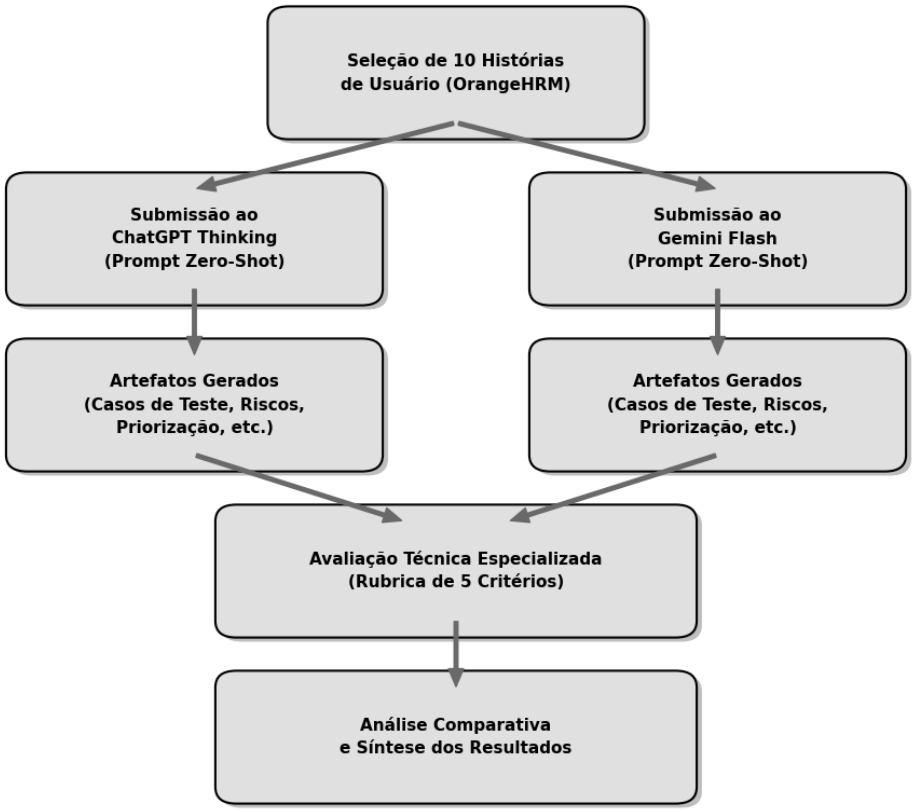
O experimento foi conduzido utilizando 10 histórias de usuário baseadas em funcionalidades críticas do sistema *open source* OrangeHRM, com foco em processos de gestão de pessoal e recrutamento.

Para cada história de usuário, foram submetidos *prompts* padronizados (Apêndice A) ao ChatGPT *Thinking* e Gemini *Flash*, solicitando: análise crítica de requisitos e identificação de ambiguidades; geração de cenários e casos de teste detalhados; identificação de riscos técnicos e cenários de exceção; priorização; e avaliação final (*readiness*,

informando se a história de usuário está pronta para desenvolvimento).

Para compreender o processo de avaliação das *LLM* na engenharia de requisitos, os procedimentos metodológicos deste experimento foram estruturados em etapas sequenciais, conforme detalhado na Figura 1

Figura 1 - Fluxo de submissão de histórias de usuário do OrangeHRM para geração de artefatos de testes via *LLM*.



Fonte: Elaborado pelo autor.

As histórias de usuário (Apêndice B) utilizadas neste experimento foram derivadas de cenários de teste reais, previamente elaborados de forma manual pelo pesquisador, que atua na área de Garantia de Qualidade de Software. A conversão desses cenários técnicos para o formato de histórias de usuário contou com o auxílio da ferramenta GitHub Copilot, utilizada para padronização textual e organização estrutural, buscando assegurar consistência no conjunto de dados utilizado e aderência às regras de negócio do sistema estudado.

Visando garantir a isenção do experimento e evitar o enviesamento das res-

postas (*bias*), as histórias de usuário não foram fornecidas previamente aos modelos ChatGPT *Thinking* e Gemini *Flash*. Cada requisito foi submetido individualmente no momento da avaliação, caracterizando um cenário de teste *zero-shot*. Neste trabalho, optou-se pela utilização de *prompts* padronizados nessa abordagem, com o objetivo de avaliar o comportamento nativo dos modelos e simular a experiência de um novo usuário, sem histórico prévio de interações. Dessa forma, busca-se estabelecer uma linha de base para a análise da capacidade bruta das ferramentas avaliadas.

2.2 AVALIAÇÃO E CRITÉRIOS DE ANÁLISE

As respostas geradas pelas *LLM* foram submetidas a uma análise técnica baseada em uma rubrica estruturada, apresentada no Apêndice C. Cada critério foi pontuado em uma escala de 0 a 2, sendo 0 correspondente a desempenho insatisfatório, 1 a desempenho parcial e 2 a desempenho satisfatório. A pontuação final foi obtida pela soma das notas atribuídas aos cinco critérios, resultando em uma nota máxima de 10 pontos por resposta avaliada.

A rubrica contempla cinco dimensões de avaliação:

- **Cobertura Funcional:** capacidade do modelo em contemplar os requisitos e fluxos principais da história de usuário;
- **Relevância dos Testes:** qualidade, assertividade e utilidade dos casos de teste gerados para o contexto de QA;
- **Clareza Textual:** nível de compreensibilidade, coesão e organização da resposta fornecida;
- **Cenários Alternativos:** capacidade de identificação de caminhos de exceção, erros e *edge cases* não óbvios;
- **Aplicabilidade Prática:** facilidade de transposição das sugestões para a rotina real de teste e documentação de software.

Os critérios estabelecidos na rubrica de avaliação foram definidos e fundamentados com base nas boas práticas consolidadas da Engenharia de Software e em métricas tradicionais de cobertura de testes da literatura. A aplicação da rubrica e a respectiva atribuição de notas foram conduzidas individualmente pelo próprio autor, aplicando seu critério técnico e experiência profissional na área de Garantia de Qualidade de Software para garantir uma avaliação estritamente alinhada aos padrões de aceitação industriais.

A partir dessa estrutura, tais critérios foram elaborados no intuito de verificar a capacidade dos modelos de interpretar histórias de usuário, identificar problemas de entendimento, gerar artefatos de teste e organizar tecnicamente a resposta conforme o formato solicitado.

3 RESULTADOS E DISCUSSÃO

Nesta seção são apresentados os resultados obtidos a partir da avaliação das respostas geradas pelo ChatGPT *Thinking* e pelo Gemini *Flash* para as dez histórias de usuário utilizadas no experimento. Cada resposta foi avaliada com base na rubrica definida na metodologia, considerando cinco critérios: Cobertura Funcional, Relevância dos Testes, Clareza Textual, Cenários Alternativos e Aplicabilidade Prática. Ao final, apresenta-se uma síntese consolidada do desempenho geral dos modelos.

Na História de Usuário 1, o ChatGPT *Thinking* obteve 9 pontos, enquanto o Gemini *Flash* alcançou 8 pontos (tabela 1).

Tabela 1 - História de Usuário 1

Critério	**ChatGPT *Thinking***	**Gemini *Flash***
Cobertura Funcional	2	1
Relevância dos Testes	2	2
Clareza Textual	1	2
Cenários Alternativos	2	1
Aplicabilidade Prática	2	2
Total	**9/10**	**8/10**

Fonte: Elaborado pelo autor.

Na História de Usuário 2, o ChatGPT *Thinking* obteve 8 pontos, enquanto o Gemini *Flash* alcançou 6 pontos (tabela 2).

Tabela 2 - História de Usuário 2

Critério	**ChatGPT *Thinking***	**Gemini *Flash***
Cobertura Funcional	2	1
Relevância dos Testes	2	1
Clareza Textual	1	2
Cenários Alternativos	2	1
Aplicabilidade Prática	1	1
Total	**8/10**	**6/10**

Fonte: Elaborado pelo autor.

Na História de Usuário 3, o ChatGPT *Thinking* obteve 9 pontos, enquanto o Gemini *Flash* alcançou 7 pontos (tabela 3).

Tabela 3 - História de Usuário 3

Critério	**ChatGPT *Thinking***	**Gemini *Flash***
Cobertura Funcional	2	1
Relevância dos Testes	2	1
Clareza Textual	2	2
Cenários Alternativos	2	1
Aplicabilidade Prática	1	2
Total	**9/10**	**7/10**

Fonte: Elaborado pelo autor.

Na História de Usuário 4, o ChatGPT *Thinking* obteve 9 pontos, enquanto o Gemini *Flash* alcançou 8 pontos (tabela 4).

Tabela 4 - História de Usuário 4

Critério	**ChatGPT *Thinking***	**Gemini *Flash***
Cobertura Funcional	2	1
Relevância dos Testes	2	2
Clareza Textual	1	2
Cenários Alternativos	2	1
Aplicabilidade Prática	2	2
Total	**9/10**	**8/10**

Fonte: Elaborado pelo autor.

Na História de Usuário 5, o ChatGPT *Thinking* obteve 9 pontos, enquanto o Gemini *Flash* alcançou 8 pontos (tabela 5).

Tabela 5 - História de Usuário 5

Critério	**ChatGPT *Thinking***	**Gemini *Flash***
Cobertura Funcional	2	1
Relevância dos Testes	2	2
Clareza Textual	1	2
Cenários Alternativos	2	1
Aplicabilidade Prática	2	2
Total	**9/10**	**8/10**

Fonte: Elaborado pelo autor.

Na História de Usuário 6, o ChatGPT *Thinking* obteve 8 pontos, enquanto o Gemini *Flash* alcançou 6 pontos (tabela 6).

Tabela 6 - História de Usuário 6

Critério	**ChatGPT *Thinking***	**Gemini *Flash***
Cobertura Funcional	2	1
Relevância dos Testes	2	1
Clareza Textual	1	2
Cenários Alternativos	2	1
Aplicabilidade Prática	1	1
Total	**8/10**	**6/10**

Fonte: Elaborado pelo autor.

Na História de Usuário 7, o ChatGPT *Thinking* obteve 8 pontos, enquanto o Gemini *Flash* alcançou 7 pontos (tabela 7).

Tabela 7 - História de Usuário 7

Critério	**ChatGPT *Thinking***	**Gemini *Flash***
Cobertura Funcional	2	1
Relevância dos Testes	2	1
Clareza Textual	1	2
Cenários Alternativos	2	1
Aplicabilidade Prática	1	2
Total	**8/10**	**7/10**

Fonte: Elaborado pelo autor.

Na História de Usuário 8, o ChatGPT *Thinking* obteve 9 pontos, enquanto o Gemini *Flash* alcançou 8 pontos (tabela 8).

Tabela 8 - História de Usuário 8

Critério	**ChatGPT *Thinking***	**Gemini *Flash***
Cobertura Funcional	2	2
Relevância dos Testes	2	1
Clareza Textual	2	2
Cenários Alternativos	2	1
Aplicabilidade Prática	1	2
Total	**9/10**	**8/10**

Fonte: Elaborado pelo autor.

Na História de Usuário 9, o ChatGPT *Thinking* obteve 9 pontos, enquanto o Gemini *Flash* alcançou 8 pontos (tabela 9).

Tabela 9 - História de Usuário 9

Critério	**ChatGPT *Thinking***	**Gemini *Flash***
Cobertura Funcional	2	1
Relevância dos Testes	2	2
Clareza Textual	2	2
Cenários Alternativos	2	1
Aplicabilidade Prática	1	2
Total	**9/10**	**8/10**

Fonte: Elaborado pelo autor.

Na História de Usuário 10, o ChatGPT *Thinking* obteve 9 pontos, enquanto o Gemini *Flash* alcançou 8 pontos (tabela 10).

Tabela 10 - História de Usuário 10

Critério	**ChatGPT *Thinking***	**Gemini *Flash***
Cobertura Funcional	2	1
Relevância dos Testes	2	2
Clareza Textual	2	2
Cenários Alternativos	2	1
Aplicabilidade Prática	1	2
Total	**9/10**	**8/10**

Fonte: Elaborado pelo autor.

3.1 SÍNTESE DOS RESULTADOS

Para consolidar os resultados obtidos, foi elaborada uma matriz de tabulação contendo as pontuações atribuídas a cada modelo em todas as histórias de usuário avaliadas. A partir desses dados, realizou-se uma análise estatística descritiva, considerando a pontuação total e a média final de desempenho de cada modelo.

A Tabela 11 apresenta o desempenho geral dos modelos avaliados.

Tabela 11 - Desempenho geral dos modelos avaliados

Modelo	**Pontuação total**	**Média final**
ChatGPT *Thinking*	87/100	8,7
Gemini *Flash*	74/100	7,4

Fonte: Elaborado pelo autor.

Observa-se que o ChatGPT *Thinking* obteve desempenho superior ao Gemini *Flash* no conjunto das dez histórias de usuário analisadas. O modelo alcançou média final

de 8,7 pontos, enquanto o Gemini *Flash* obteve média de 7,4 pontos, indicando diferença de 1,3 ponto entre as médias gerais.

A Tabela 12 apresenta a distribuição das pontuações finais obtidas por cada modelo em cada história de usuário.

Tabela 12 - Pontuação por história de usuário

História de Usuário	**ChatGPT *Thinking***	**Gemini *Flash***
HU1: Login no Sistema	9	8
HU2: Atualizar Dados Pessoais	8	6
HU3: Solicitar Licença	9	7
HU4: Aprovar Solicitação de Licença	9	8
HU5: Registrar Ponto de Trabalho	9	8
HU6: Cadastrar Novo Funcionário	8	6
HU7: Criar Vaga de Emprego	8	7
HU8: Gerar Relatório de Licenças	9	8
HU9: Visualizar Lista de Funcionários	9	8
HU10: Fazer Logout do Sistema	9	8

Fonte: Elaborado pelo autor.

A análise da pontuação por história de usuário evidencia que o ChatGPT *Thinking* obteve resultado superior em todas as avaliações realizadas. O modelo apresentou notas entre 8 e 9 pontos, demonstrando desempenho mais consistente ao longo do experimento. O Gemini *Flash*, por sua vez, apresentou variação entre 6 e 8 pontos, indicando desempenho satisfatório, porém menos regular.

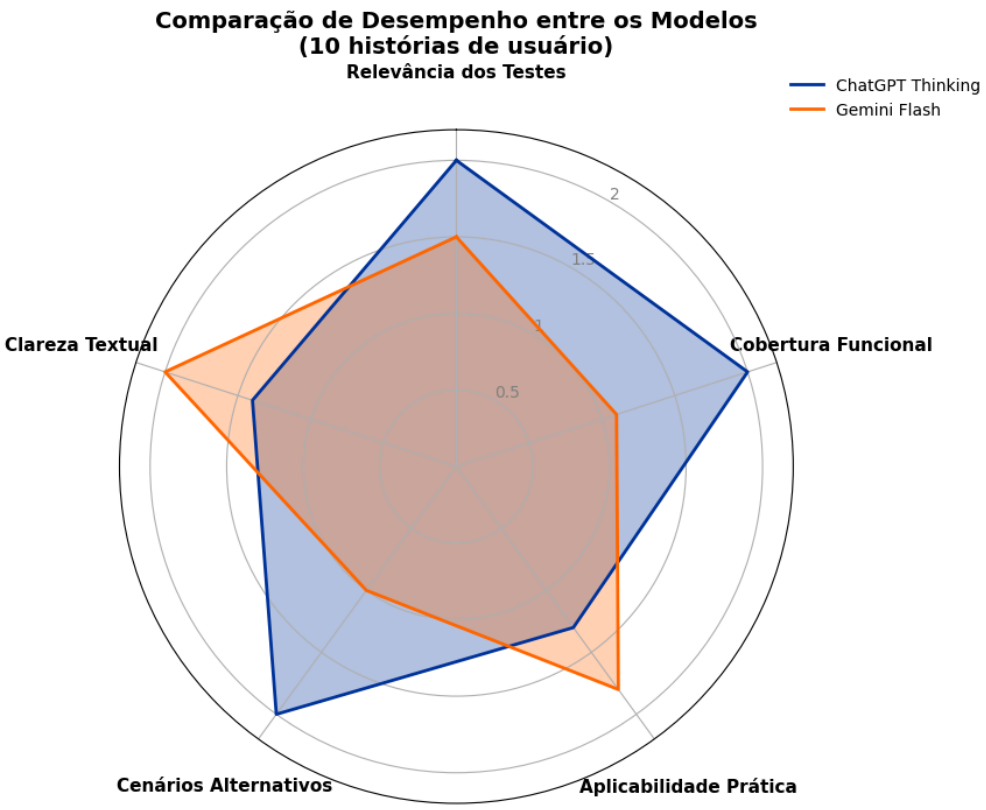
A Tabela 13 apresenta uma síntese qualitativa do desempenho observado por critério de avaliação.

Tabela 13 - Síntese do desempenho por critério avaliado

Critério	**Tendência observada**
Cobertura Funcional	Melhor desempenho do ChatGPT *Thinking*
Relevância dos Testes	Melhor desempenho do ChatGPT *Thinking*
Clareza Textual	Melhor desempenho do Gemini *Flash*
Cenários Alternativos	Melhor desempenho do ChatGPT *Thinking*
Aplicabilidade Prática	Desempenho semelhante entre os modelos

Fonte: Elaborado pelo autor.

Figura 2 - Comparação de desempenho entre os modelos (10 histórias de usuários)



Fonte: Elaborado pelo autor.

De modo geral, os resultados indicam que o ChatGPT *Thinking* apresentou maior capacidade de exploração dos requisitos, geração de casos de teste e identificação de cenários alternativos. O Gemini *Flash*, por sua vez, destacou-se pela clareza textual e objetividade das respostas, embora tenha apresentado menor profundidade em critérios relacionados à cobertura funcional e à variedade dos testes sugeridos.

Assim, os resultados demonstram que ambos os modelos podem contribuir para atividades de apoio à Garantia de Qualidade de Software, especialmente na análise inicial de histórias de usuário e na geração preliminar de casos de teste. No entanto, o ChatGPT *Thinking* apresentou desempenho mais robusto no conjunto avaliado, enquanto o Gemini *Flash* mostrou-se mais adequado para respostas objetivas e de rápida compreensão.

Os resultados obtidos demonstram que os *LLM* avaliados possuem potencial para apoiar atividades relacionadas à Garantia de Qualidade de Software, especialmente na análise de requisitos, geração de casos de teste, identificação de cenários alternativos e apoio à priorização de riscos. Considerando as dez histórias de usuário analisadas, o ChatGPT *Thinking* obteve média geral de 8,7 pontos, enquanto o Gemini *Flash* alcançou

média de 7,4 pontos. Esse resultado indica desempenho superior da ferramenta no conjunto avaliado, sobretudo nos critérios de Cobertura Funcional, Relevância dos Testes e Cenários Alternativos.

O desempenho superior do ChatGPT *Thinking* pode ser explicado pela maior profundidade das respostas geradas. Ao longo das histórias de usuário avaliadas, o modelo demonstrou maior capacidade de explorar fluxos principais, propor cenários negativos, identificar condições de exceção e relacionar os testes às regras de negócio. Esse comportamento é relevante para atividades de QA, pois indica maior capacidade de transformar histórias de usuário em artefatos de teste mais abrangentes e tecnicamente detalhados. Em contrapartida, o Gemini *Flash* também produziu artefatos úteis, porém com respostas mais objetivas e concisas, destacando-se principalmente no critério de Clareza Textual.

Esse resultado converge com estudos recentes sobre o uso de *LLM* em atividades de teste de software. Wang et al. (2024) destacam que a geração de casos de teste é uma das principais aplicações investigadas nesse contexto. De forma complementar, Celik e Mahmoud (2025) apontam que esses modelos podem produzir casos de teste coerentes a partir de descrições textuais de requisitos, reduzindo o esforço inicial de elaboração dos artefatos de teste. No presente estudo, esse comportamento foi observado em ambos os modelos, embora o ChatGPT *Thinking* tenha apresentado maior profundidade, completude e abrangência nas respostas geradas.

A diferença entre ambos também pode ser interpretada à luz de estudos empíricos sobre geração automática de testes. Guilherme e Vincenzi (2023) observaram que o ChatGPT apresentou potencial para geração de testes unitários, embora ainda demande validação humana. De forma semelhante, Schäfer et al. (2023) identificaram que *LLM* podem obter bons resultados em tarefas de geração automatizada de testes, mas apresentam limitações em cenários mais complexos. Tang et al. (2023), ao comparar ChatGPT com técnicas de *Search-Based Software Testing*, também indicaram que modelos de linguagem podem produzir testes compreensíveis e úteis, mas nem sempre alcançam a mesma sistematicidade de ferramentas especializadas. Esses achados dialogam com os resultados deste trabalho, nos quais os modelos produziram bons artefatos iniciais, mas ainda exigem revisão técnica antes de sua aplicação prática.

Enquanto ao Gemini, à partir dos resultados deste experimento, observou-se que suas respostas foram geralmente mais diretas e organizadas, favorecendo a leitura rápida e a aplicação inicial das sugestões. Entretanto, essa concisão frequentemente veio acompanhada de menor cobertura funcional e menor exploração de cenários alternativos. Esse comportamento sugere que, embora o modelo possa ser útil como apoio inicial

à documentação e estruturação de ideias, suas respostas tendem a demandar maior complementação por parte do profissional de QA quando o objetivo é obter uma análise mais ampla e tecnicamente detalhada.

Outro aspecto relevante observado foi a capacidade dos modelos de apoiar a análise de requisitos. Em diferentes histórias de usuário, os modelos apontaram ambiguidades, lacunas e possíveis interpretações alternativas dos requisitos apresentados, ainda que com níveis distintos de profundidade. Esse resultado dialoga com Ronanki, Berger e Horkoff (2023), que investigaram o potencial do ChatGPT em processos de elicitação de requisitos, observando bons resultados em atributos como consistência, correção e compreensibilidade, embora com limitações relacionadas à ambiguidade e viabilidade. Também se aproxima da revisão de Marques, Silva e Bernardino (2024), que identifica o ChatGPT como um modelo promissor para apoiar atividades de elicitação, documentação e validação de requisitos. Além disso, Almeida et al. (2025) reforçam que diferentes ferramentas podem apresentar desempenhos distintos na geração de requisitos, o que contribui para justificar a comparação realizada entre ChatGPT *Thinking* e Gemini *Flash* neste estudo.

A abordagem *zero-shot* adotada no experimento também merece destaque. Como os modelos não receberam exemplos prévios nem contexto adicional específico para cada tarefa, os resultados refletem o desempenho das ferramentas em uma condição inicial de uso, baseada apenas nas instruções fornecidas no *prompt* padronizado. Essa escolha metodológica permitiu simular uma situação próxima ao uso inicial por profissionais da área e forneceu uma linha de base para comparação entre os modelos. Segundo Wang et al. (2024), estratégias como *zero-shot* e *few-shot prompting* são recorrentes em estudos sobre *LLM* aplicados a testes de software. Além disso, Celik e Mahmoud (2025) destacam que a estrutura e o conteúdo dos *prompts* podem influenciar a qualidade dos testes gerados, impactando aspectos como cobertura, acurácia e detecção de falhas. Assim, embora a padronização dos *prompts* tenha contribuído para maior controle experimental, ela também limitou a exploração de refinamentos sucessivos que poderiam melhorar a completude e a aplicabilidade dos artefatos gerados.

Apesar dos resultados positivos, o estudo também evidenciou limitações importantes. Em alguns casos, os modelos produziram sugestões genéricas ou assumiram regras de negócio não explicitadas nas histórias de usuário. Esse comportamento, observado no presente experimento, reforça a necessidade de validação humana, especialmente em contextos que envolvem permissões de acesso, segurança, regras específicas do domínio e impactos operacionais. Essa necessidade também é apontada por Marques, Silva e Bernardino (2024), que destacam que *LLM* podem produzir informações

imprecisas, ambíguas ou enviesadas, comprometendo a confiabilidade e a qualidade dos artefatos gerados em Engenharia de Requisitos caso não haja revisão crítica.

Além das limitações inerentes aos modelos, deve-se considerar uma limitação metodológica do presente estudo. A avaliação das respostas foi realizada por um único avaliador com experiência profissional em Garantia de Qualidade de Software, utilizando uma rubrica estruturada previamente definida. Embora essa abordagem tenha permitido uma análise técnica consistente e padronizada, a atribuição das pontuações pode estar sujeita a certo grau de subjetividade. Estudos futuros podem ampliar a confiabilidade dos resultados por meio da participação de múltiplos avaliadores independentes e da utilização de métricas de concordância interavaliadores.

Também é importante considerar que o experimento foi conduzido com dez histórias de usuário derivadas de funcionalidades do sistema OrangeHRM. Embora essas histórias representem cenários relevantes para processos de gestão de pessoas e recrutamento, os resultados não devem ser generalizados automaticamente para todos os domínios de software. Sistemas com maior criticidade, maior complexidade regulatória ou requisitos de segurança mais rigorosos podem demandar análises adicionais e validações mais específicas.

De modo geral, os resultados indicam que os *LLM* podem contribuir de forma relevante para aumentar a produtividade do Analista de QA, especialmente na etapa inicial de análise de requisitos e elaboração de casos de teste. O ChatGPT *Thinking* apresentou melhor desempenho geral, destacando-se pela profundidade técnica e pela exploração de cenários alternativos. O Gemini *Flash*, por sua vez, mostrou-se útil pela clareza e objetividade das respostas, podendo ser empregado como ferramenta de apoio rápido à estruturação inicial de artefatos. Contudo, os achados reforçam a visão predominante na literatura de que os *LLM* devem ser utilizados como ferramentas complementares, e não como substitutos da atuação humana especializada. A combinação entre a capacidade generativa dos modelos e o julgamento técnico do profissional de QA tende a representar a abordagem mais adequada para obtenção de resultados confiáveis, úteis e aderentes às necessidades reais dos projetos de software.

4 CONCLUSÃO

O presente trabalho atingiu seu objetivo geral ao avaliar o uso de Modelos de Linguagem de Grande Escala (LLM) como ferramentas de apoio à Garantia de Qualidade de Software, analisando seu potencial na interpretação de requisitos, na identificação de riscos e no suporte à elaboração de artefatos de teste. Por meio de um experimento controlado com dez histórias de usuário baseadas no sistema OrangeHRM, comparou-

se o desempenho do ChatGPT *Thinking* e do Gemini *Flash* em abordagem *zero-shot*, utilizando uma rubrica estruturada composta por cinco critérios técnicos.

A análise dos resultados demonstrou desempenho superior do ChatGPT *Thinking* no conjunto de histórias de usuário avaliadas. Embora não tenha alcançado nota máxima em todas as avaliações individuais, o modelo obteve as maiores pontuações na maior parte dos cenários analisados, destacando-se pela profundidade técnica, pela maior cobertura funcional e pela capacidade de explorar cenários alternativos e riscos associados às histórias. O Gemini *Flash*, por sua vez, apresentou respostas mais concisas e organizadas, com bom desempenho em Clareza Textual e Aplicabilidade Prática, mas demonstrou limitações recorrentes quanto à profundidade da análise e à variedade dos casos de teste gerados.

Em relação aos objetivos específicos, o estudo permitiu identificar aplicações práticas de *LLM* no ciclo de teste, comparar o desempenho entre diferentes modelos e avaliar a utilidade das sugestões geradas para apoiar a tomada de decisão do Analista de QA. Evidenciou-se que os modelos podem acelerar a criação inicial de artefatos de teste e auxiliar na identificação de lacunas nos requisitos, desde que suas respostas sejam revisadas criticamente por um profissional qualificado.

Dentre as limitações observadas, destacam-se a possibilidade de *hallucinations*, a dependência excessiva das sugestões geradas, a variabilidade das respostas e as preocupações relacionadas à privacidade de dados. Também deve ser considerada a limitação metodológica relacionada à avaliação por um único especialista e ao uso de uma amostra composta por dez histórias de usuário de um único sistema. Tais fatores reforçam que essas ferramentas possuem caráter complementar e não substitutivo. A validação humana permanece indispensável para garantir a acurácia dos artefatos produzidos, a aderência às regras de negócio e a segurança dos processos de QA.

Como principal contribuição, este trabalho apresenta uma comparação empírica entre dois *LLM* comerciais aplicados ao apoio de atividades de QA a partir de histórias de usuário, utilizando uma rubrica estruturada voltada à cobertura funcional, relevância dos testes, clareza textual, cenários alternativos e aplicabilidade prática.

Como sugestão para trabalhos futuros, recomenda-se a ampliação da amostra com requisitos de maior complexidade técnica, a inclusão de outros modelos, como Claude ou Grok, a comparação entre abordagens *zero-shot* e *few-shot*, e a integração das respostas geradas por *LLM* com ferramentas de gestão e automação de testes. Em última análise, os *LLM* representam uma fronteira promissora para a Engenharia de Software, podendo elevar a produtividade do Analista de QA quando utilizados sob rigorosa supervisão técnica.

REFERÊNCIAS

ALMEIDA, C. et al. From elicitation interviews to software requirements: Evaluating llm performance in requirement generation. In: **Proceedings of the 28th Workshop on Requirements Engineering (WER2025)**. [S.l.: s.n.], 2025.

CELIK, A.; MAHMOUD, Q. H. A review of large language models for automated test case generation. **Machine Learning and Knowledge Extraction**, v. 7, n. 3, p. 97, mar 2025. Acesso em: 10 abr. 2026.

GAROUSI, V.; MäNTYLä, M. V. A systematic literature review of literature reviews in software testing. **Information and Software Technology**, v. 80, p. 195–216, dec 2016. Acesso em: 15 jun. 2025.

GUILHERME, V. H.; VINCENZI, A. M. R. An initial investigation of ChatGPT unit test generation capability. In: **SAST 2023**. Campo Grande, MS, Brazil: [s.n.], 2023.

KHAN, M. E.; KHAN, F. Importance of software testing in software development life cycle. **International Journal of Computer Science Issues**, v. 11, n. 2, p. 120–123, mar 2014. Acesso em: 15 jun. 2025. Disponível em: <https://www.ijcsi.org/papers/IJCSI-11-2-2-120-123.pdf>.

MARQUES, N.; SILVA, R. R.; BERNARDINO, J. Using ChatGPT in software requirements engineering: A comprehensive review. **Future Internet**, v. 16, n. 6, p. 180, 2024.

RONANKI, K.; BERGER, C.; HORKOFF, J. **Investigating ChatGPT's potential to assist in requirements elicitation processes**. 2023.

SCHÄFER, M. et al. **An empirical evaluation of using large language models for automated unit test generation**. 2023.

TANG, Y. et al. **ChatGPT vs SBST: A comparative assessment of unit test suite generation**. 2023.

WANG, J. et al. Software testing with large language models: Survey, landscape, and vision. **IEEE Transactions on Software Engineering**, v. 50, n. 4, p. 911–936, apr 2024. Acesso em: 10 abr. 2026.

APÊNDICE A - Prompt de Avaliação

"Assuma o papel de um Analista de QA Sênior especializado em testes funcionais e análise de requisitos de software. Você receberá uma História de Usuário referente ao sistema OrangeHRM. Sua tarefa é realizar uma avaliação técnica completa da história apresentada e responder de forma estruturada com as seções abaixo:

1. **Análise de Requisitos**
    - Ambiguidades ou inconsistências
    - Informações ausentes
    - Regras de negócio implícitas
    - Principais riscos de entendimento
2. **Casos de Teste**
   Crie uma tabela *Markdown* com os seguintes tipos:
    - Fluxo principal (*Happy Path*)
    - Fluxos alternativos
    - Validações negativas / Dados inválidos
    - Permissões e perfis de acesso
3. ***Edge Cases* e Riscos**
   Liste os principais *edge cases* e riscos técnicos (limites, concorrência, segurança, usabilidade).
4. **Priorização**
   Classifique todos os casos de teste como: Alta | Média | Baixa
5. **Avaliação Final**
   A história está pronta para desenvolvimento? (Sim / Parcialmente / Não). Justifique em até 2 frases.

Diretrizes obrigatórias:

- Seja técnico e objetivo;
- Organize a resposta em tópicos e tabelas quando útil;
- Não invente requisitos sem indicar hipótese;

- Considere boas práticas de QA;
- Responda em português do Brasil.”

APÊNDICE B - Histórias de Usuários

H1: Login no sistema

Como usuário, eu quero fazer login com usuário e senha válidos para acessar a página inicial da aplicação.

Critérios de Aceitação:

- Página de login é exibida;
- Usuário informa credenciais válidas;
- Clica no botão *Login*;
- Sistema redireciona para o *Dashboard*.

H2: Atualizar dados pessoais

Como funcionário, eu quero atualizar minhas informações pessoais para manter meus dados sempre atualizados.

Critérios de Aceitação:

- Acessar o menu *My Info*;
- Editar campos como telefone, endereço ou foto;
- Salvar as alterações;
- Mensagem de sucesso é exibida.

H3: Solicitar licença

Como funcionário, eu quero solicitar uma licença para registrar minhas ausências.

Critérios de Aceitação:

- Acessar o módulo *Leave*;
- Selecionar tipo de licença e período;
- Enviar a solicitação;
- Solicitação aparece como "*Pending*".

H4: Aprovar solicitação de licença
Como supervisor, eu quero aprovar ou rejeitar solicitações de licença para gerenciar as ausências da minha equipe.
Critérios de Aceitação:

- Acessar lista de solicitações pendentes;
- Visualizar detalhes da solicitação;
- Escolher Aprovar ou Rejeitar;
- Status da solicitação é atualizado.

H5: Registrar ponto de trabalho
Como funcionário, eu quero registrar meu ponto (*Punch In/Out*) para controlar minhas horas trabalhadas.
Critérios de Aceitação:

- Acessar o módulo *Time*;
- Clicar em *Punch In* ao iniciar o trabalho;
- Clicar em *Punch Out* ao finalizar;
- Horários são registrados corretamente.

H6: Cadastrar novo funcionário
Como administrador, eu quero cadastrar um novo funcionário para incluí-lo no banco de dados do RH.
Critérios de Aceitação:

- Acessar menu *Admin* ou PIM;
- Preencher dados básicos do funcionário;
- Salvar o cadastro;
- Funcionário aparece na lista de *employees*.

H7: Criar vaga de emprego
Como administrador de RH, eu quero criar uma nova vaga de emprego para iniciar um processo seletivo.
Critérios de Aceitação:

- Acessar o módulo *Recruitment*;
- Preencher título, descrição e requisitos;
- Publicar a vaga;
- Vaga fica disponível para candidatos.

H8: Gerar relatório de licenças
Como administrador de RH, eu quero gerar relatório de licenças para analisar as ausências dos funcionários.
Critérios de Aceitação:

- Acessar o módulo *Reports*;
- Selecionar o relatório de *Leave*;
- Aplicar filtros (período, tipo);
- Gerar e visualizar o relatório.

H9: Visualizar lista de funcionários
Como usuário autorizado, eu quero visualizar a lista de funcionários para consultar informações da equipe.
Critérios de Aceitação:

- Acessar *Employee List*;
- Aplicar filtros quando necessário;
- Visualizar nome, cargo e departamento;
- Poder acessar detalhes do funcionário.

H10: Fazer logout do sistema
Como usuário, eu quero fazer logout da aplicação para sair com segurança.
Critérios de Aceitação:

- Clicar no ícone de usuário no topo;
- Selecionar a opção *Logout*;
- Ser redirecionado para a tela de *login*;
- Sessão é encerrada.

APÊNDICE C - Rubrica de Avaliação

Tabela de Critérios de Avaliação

Critério	Descrição
1. Cobertura funcional	Avalia se o modelo contempla os requisitos, fluxos principais, ambiguidades, lacunas e riscos da história de usuário.
2. Relevância dos Testes	Avalia a qualidade, utilidade e variedade dos casos de teste gerados para o contexto de QA.
3. Clareza textual	Avalia a organização, coesão, legibilidade e aderência da resposta ao formato solicitado.
4. Cenários alternativos	Avalia a capacidade de identificar fluxos alternativos, exceções, dados inválidos, riscos técnicos e *edge cases* não óbvios.
5. Aplicabilidade prática	Avalia se a resposta pode ser aproveitada na rotina real de teste e documentação de software.

Escala de Pontuação

Nota	Significado
0	Não atende ao critério
1	Atende parcialmente
2	Atende completamente