Utilizando a IA Generativa para Apoiar a Especificação de Casos de Teste Funcionais em um Sistema de Software do Setor Público

Aline Lima de Souza França
PESC/COPPE, UFRJ
Rio de Janeiro, Brazil
limaals@cos.ufrj.br

Jessica Soares da Costa
PESC/COPPE, UFRJ
Rio de Janeiro, Brazil
jessicacosta@cos.ufrj.br

Rebeca Motta
UFF
Niterói, Brazil
rebecamotta@id.uff.br

Elaine Nunes
CAPGov - COPPETEC
Rio de Janeiro, Brazil
elaine@cos.ufrj.br

Gustavo Rocha Barreto Pinto
CAPGov - COPPETEC
Rio de Janeiro, Brazil
grbpinto@cos.ufrj.br

Rafael Maiani de Mello
UFRJ
Rio de Janeiro, Brazil
rafaelmello@ic.ufrj.br

Resumo

Os sistemas de software do setor público frequentemente envolvem regras de negócio complexas, processos críticos e integrações com múltiplos serviços externos, tornando a especificação de testes funcionais uma atividade desafiadora e demorada. Este artigo apresenta um relato de experiência sobre a transferência de tecnologia e a evolução de uma abordagem baseada em Inteligência Artificial Generativa (GenAI) para apoiar a criação automatizada de descrições de casos de teste funcionais na CAPGov. A solução inicial baseava-se na geração de cenários de teste a partir de histórias de usuário, utilizando prompts. Posteriormente, ela evoluiu para um agente de IA sensível ao contexto, enriquecido com fontes de conhecimento corporativo, incluindo documentação do sistema e capturas de tela das interfaces. A versão atual da abordagem permitiu a geração de descrições de navegação de teste mais completas e semelhantes às produzidas por humanos, reduzindo o esforço de documentação e a dívida técnica.

Palavras-chave

Casos de Teste, Grandes Modelos de Linguagem, IA Generativa, Agentes, Setor Público, Automação de Testes

1 Introdução

A elaboração manual de casos de teste funcionais representa um desafio para equipes de qualidade de software, especialmente em instituições públicas com sistemas de alta complexidade de negócio e dependências externas. Não raramente, o teste de uma única funcionalidade em tais sistemas pode envolver dezenas de cenários que precisam ser identificados e descritos passo a passo [4]. Nesse sentido, ainda que a execução dos testes funcionais seja posteriormente automatizada, a alta carga cognitiva demandada para a elaboração dos testes pode se tornar fonte de atrasos nas entregas e dívida técnica.

Com a crescente adoção e amadurecimento dos grandes modelos de linguagem (LLMs), surge a oportunidade de explorar a capacidade de esses modelos gerarem casos de teste funcionais a partir da interpretação de requisitos expressos em linguagem natural [1]. Nesse sentido, a literatura técnica tem demonstrado o interesse crescente no uso de LLMs para apoiar a prática de testes por meio de abordagens híbridas e automatizadas [3, 5].

Este trabalho relata a trajetória de transferência de uma tecnologia para geração de testes funcionais na equipe de testes que atua no principal sistema de uma instituição pública do setor jurídico [4]. A experiência mapeia a evolução dessa tecnologia alimentada pelo *feedback* contínuo do time de testes da organização ao longo de *sprints*. Inicialmente concebida e avaliada como um *prompt* estruturado e linear, a tecnologia passou a incluir a análise de imagens de navegação do sistema, combinada com o conhecimento corporativo por Geração Aumentada por Recuperação (RAG), evoluindo até se transformar em um *Agente de IA contextualizado* em produção. Neste relato, apresentamos os principais desafios da jornada com base no feedback dos analistas de teste e as principais lições aprendidas.

2 Contexto e Motivação

A instituição pública opera um sistema de grande escala para gerenciar fluxos de trabalho jurídicos e atender aos cidadãos [4]. Atualmente, o sistema possui mais de 20 mil usuários internos, 2 milhões de processos e 2,7 milhões de pessoas cadastradas. A manutenção e operação deste ecossistema são realizadas por uma equipe externa especializada do Centro de Apoio a Políticas do Governo (CAPGov), composta por aproximadamente 100 colaboradores, dentre eles doze profissionais dedicados à escrita, manutenção e automação de testes funcionais.

A especificação manual de testes consumia uma parcela significativa da capacidade da equipe de testes a cada *sprint*, deixando pouco tempo para atividades de teste exploratório e baseado em risco. Além disso, frequentemente a escrita dos testes ficava pendente para evitar atrasos na entrega, impactando diretamente na cobertura da automação dos testes funcionais. Neste sentido, o objetivo de adotar uma tecnologia para a geração de testes funcionais não era substituir os analistas, mas automatizar a identificação de cenários e a escrita descritiva para que os testes pudessem ser amplamente automatizados dentro do prazo da sprint e que os analistas pudessem se concentrar em tarefas de maior valor agregado para garantir a qualidade e cobertura dos testes.

O ponto de partida foi uma pesquisa de mestrado [6] que utilizou histórias de usuário reais do sistema para avaliar a capacidade de diferentes LLMs e técnicas de *prompting* na geração de casos de teste. O estudo conduzido nesta pesquisa não identificou diferenças significativas de desempenho *(precisão e recall)* entre os modelos e técnicas avaliadas, sugerindo que o fator determinante de qualidade reside na engenharia de *prompt*, e não na escolha do modelo.

Na fase inicial, a tecnologia era capaz de gerar apenas os *cenários* dos casos de teste(CTs), títulos e descrições sumarizadas, sem incluir passos detalhados, pré-requisitos ou dados de entrada, exigindo complementação manual pela equipe de testes. Ao longo do processo de transferência tecnológica para o CAPGov, a equipe de testes verificou que a abordagem inicial baseada em *prompts* isolados não conseguia capturar as regras de negócio implícitas nem a navegação do sistema, motivando uma evolução iterativa impulsionada pelo *feedback* contínuo dos responsáveis pela escrita de testes manuais na equipe.

3 Evolução da Tecnologia

No período de 11 meses, a tecnologia de geração de casos de teste foi evoluindo de modo incremental ao longo de quatro fases principais, orientada pelo *feedback* contínuo dos responsáveis pelos testes manuais. A cada *sprint*, os CTs gerados pelo agente são registrados em um diário de teste estruturado em uma planilha compartilhada onde cada CT é classificado quanto ao uso efetivo (*usado*/*não usado*) e ao resultado da validação manual (*funcionou*/*parcial*/*não funcionou*), acompanhado de um campo de comentário livre. Esse diário é a principal fonte de *feedback* qualitativo do processo: os desvios identificados pelos analistas, ambiguidades de requisitos, cenários ausentes ou inferências incorretas alimentam diretamente a base de conhecimento para o desenvolvimento do agente, refinando suas diretrizes e expandindo o contexto disponível para as *sprints* seguintes. Dessa forma, cada ciclo de avaliação humana não apenas corrige a saída imediata, mas também contribui para a evolução contínua da tecnologia.

Nas subseções a seguir, detalhamos cada uma dessas fases, destacando os principais desafios enfrentados e como o *feedback* dos analistas motivou cada avanço tecnológico.

3.1 Fase 1: Prompt Estruturado Simples (maio-agosto de 2025)

Na Fase 1, a engenharia de *prompt* clássica com *International Software Testing Qualifications Board* (ISTQB), especificamente no nível *Certified Tester Foundation Level* (CTFL) [2] foi aplicada sobre o texto puro das histórias de usuário. A tecnologia gerava apenas cenários, títulos e descrições sem passos detalhados, pré-requisitos ou dados de entrada, exigindo complementação manual. O *feedback* dos analistas identificou alucinações de interface: a IA previu mensagens modais de alerta onde o sistema tratava erros silenciosamente no *backend*, e campos em modo somente leitura onde a regra de negócio simplesmente ocultava a interface inteira. Esses desvios evidenciaram dois limites estruturais: a dependência de requisitos explícitos e a ausência de contexto visual e de domínio.

3.2 Fase 2: Multimodalidade (agosto-dezembro de 2025)

A Fase 2 expandiu o *input* para aceitar capturas de tela (*screenshots*), mitigando as alucinações visuais ao permitir identificar campos obrigatórios e elementos de navegação diretamente da interface. O escopo do *prompt* também evoluiu: de geração de cenários para casos de teste com passos padronizados. Contudo, o modelo tornou-se dependente do estado estático da interface, falhando em correlacionar elementos visuais com regras de negócio dinâmicas e dependências de dados nos bastidores do sistema.

3.3 Fase 3: Enriquecimento com RAG (dezembro de 2025-abril de 2026)

A Fase 3 integrou um mecanismo de Geração Aumentada por Recuperação (RAG), consultando dinamicamente a *wiki* técnica do sistema e trazendo permissões reais de perfis e mapeamentos de fluxos. Apesar do ganho de contexto, os testes piloto revelaram duas limitações cruciais:

O RAG não enxerga o que não está escrito: Em funcionalidades de alta complexidade do sistema, como o *fluxo de agendamentos*, o RAG sugeriu cobertura linear de caminhos felizes, identificando apenas sete cenários, enquanto a análise humana identificou 48 cenários para o mesmo fluxo. Deste modo, observou-se que o mecanismo recuperava regras do sistema com precisão, mas não conseguia combinar sub-estados de agendamentos com perfis de usuário e condições de dados ao mesmo tempo. Assim, percebeu-se que o analista de testes enxergava o sistema como uma matriz de combinações, enquanto que o RAG enxergava uma lista de regras.

Ausência de sequenciamento para automação: O RAG gerava cenários corretos, mas isolados e sem ordenação lógica. O especialista humano precisava reordenar os cenários para permitir o reaproveitamento de estados e dos dados ao longo da *pipeline* de automação. Assim, identificou-se a necessidade de recursos e heurísticas que permitissem ao modelo compreender o fluxo de execução, e não apenas do conteúdo dos requisitos.

3.4 Fase 4: Agente de IA com Módulos Especializados (abril de 2026)

Na Fase 4, a abordagem evoluiu para um agente de IA contextualizado responsável por orquestrar todo o processo de geração de casos de teste. A Figura 1 ilustra como o agente combina diferentes fontes de contexto antes de acionar seus três componentes internos: módulo de geração de casos de teste (encapsula técnicas ISTQB/CTFL [2] e regras de domínio), módulo de geração de planilhas Excel e módulo de geração de relatórios PDF.

O agente é configurado por três elementos: um *prompt*, que define sua identidade como especialista em testes e especifica a estrutura de saída esperada; uma *memória estruturada de sprints*, que registra regras de negócio refinadas, combinações permitidas em cadeias de *fallback* e histórias já processadas; e um *script de orquestração*, que carrega o contexto completo, envia ao modelo via API e transforma a saída aprovada em planilha padronizada. A temperatura reduzida do modelo foi adotada para favorecer consistência e minimizar a introdução de regras não previstas, reforçando o papel do agente como gerador de documentação estruturada ao invés de fonte autônoma de novas regras de negócio.

A introdução do agente de IA contribuiu para a redução do esforço da escrita mecânica dos casos de teste, permitindo que os analistas direcionem mais tempo para atividades de análise, refinamento e priorização. Em *sprints* voltadas para funcionalidades mais complexas, os casos de teste gerados passaram a servir como ponto de partida estruturado para a discussão de cenários, facilitando o

alinhamento entre equipe de testes, desenvolvimento e negócio. Além disso, a padronização dos artefatos produzidos pelo agente contribuiu para diminuir a variabilidade na documentação de testes.

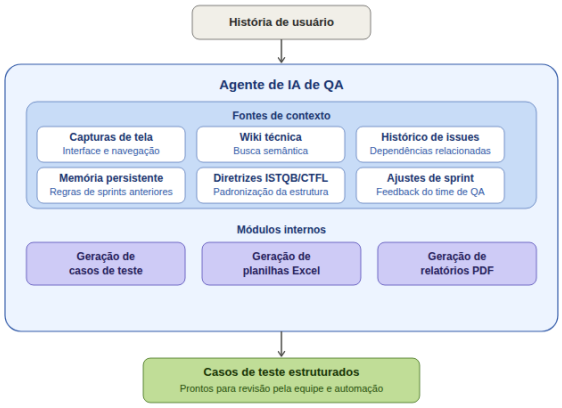

Figura 1: Visão Geral da Solução.

4 Lições Aprendidas

A experiência de evolução da tecnologia, mais especificamente da transição entre a terceira e a quarta fase, trouxe importantes lições sobre as limitações e potencialidades do uso industrial de IA Generativa na geração de testes funcionais:

O testador como curador estratégico. O valor das LLMs não está em substituir o analista, mas em automatizar a escrita mecânica descritiva. O analista atua como curador indispensável, identificando onde a IA falha em requisitos literais ou ambíguos. Nesse sentido, o analista não é substituído pela ferramenta, mas reposicionado: deixa de ser o escritor principal dos casos de teste e passa a ser o responsável por validar, selecionar e retroalimentar o agente com o conhecimento que a escrita da issue não capturou.

Conhecimento tácito é o limite real. Nenhuma combinação de RAG, *wiki* ou análise de imagens consegue capturar o conhecimento que os times acumulam em conversas de *sprint*, trocas de mensagens e acordos informais raramente documentados. Sem registro contínuo desse conhecimento, o agente não tem como incorporá-lo, o que limita diretamente seu potencial. Essa barreira ficou evidente em funcionalidades com integrações via API: vários CTs gerados não puderam ser executados por ausência de massa de dados ou restrições do ambiente de homologação, o que apontava para limites do ambiente de teste, não do modelo.

A qualidade do requisito determina a qualidade do agente. Histórias de usuário bem estruturadas tendem a gerar CTs com baixa necessidade de correções, enquanto que regras implícitas resultam em um maior desvio da especificação humana, exigindo ajustes manuais e descarte de cenários. Investir na clareza e estrutura das histórias de usuário é, portanto, uma alavanca para a efetividade da geração. O agente de IA não substitui o processo existente, mas o reorganiza, concentrando o esforço humano em decisões de maior valor agregado. Estas observações refletem trabalhos como [1], onde evidenciou-se o obstáculo do conhecimento tácito não documentado para geração de cenários de teste.

5 Conclusão e Trabalhos Futuros

Este relato de experiência apresentou a evolução incremental de uma abordagem baseada em IA Generativa para apoiar a especificação de casos de teste funcionais em uma instituição pública. A transição de *prompt* puramente descritivo para uma arquitetura de agente contextualizado permitiu mitigar lacunas de regras de negócio implícitas e reduzir o esforço de documentação do time. Os resultados obtidos reiteram que o papel das LLMs reside na otimização de tarefas mecânicas de escrita, posicionando o analista de testes humano como um curador estratégico indispensável.

A partir das lições consolidadas ao longo deste ciclo de transferência e evolução tecnológica, identificam-se três direções para trabalhos futuros. Visando mitigar a limitação do conhecimento tácito, pretendemos investigar mecanismos de captura e atualização contínua do conhecimento produzido em reuniões e conversas informais de sprint, tornando-o acessível ao agente em tempo real. Sobre a limitação de ordenação de cenários identificada na Fase 3, pretendemos utilizar a taxonomia de desvios catalogados para refinar as diretrizes do agente, inserindo instruções explícitas de particionamento ordenado e reaproveitamento de estados [3]. Finalmente, pretendemos investigar a tradução automática das planilhas geradas pelo agente para scripts executáveis [4].

Disponibilidade de Artefatos

Não são disponibilizados artefatos associados a este trabalho por razões éticas e legais. A tecnologia apresentada opera no contexto de uma instituição pública, manipulando dados sensíveis e informações protegidas por sigilo institucional.

Referências

[1] Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. 2024. Software Testing with Large Language Models: Survey, Landscape, and Vision. *IEEE Transactions on Software Engineering*, vol. 50, no. 4, pp. 911–936. DOI: 10.1109/TSE.2024.3368208.

[2] ISTQB. 2023. *Certified Tester Foundation Level Syllabus v4.0.* International Software Testing Qualifications Board.

[3] Caroline Lemieux, Jeevana Priya Inala, Shuvendu K. Lahiri, and Siddhartha Sen. 2023. Codamosa: Escaping coverage plateaus in test generation with pre-trained large language models. In *Proceedings of the 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 919–931.

[4] Bruno Souza, Rebeca Motta, Jessica Costa, Elaine Nunes, Gustavo Pinto, and Rafael Mello. 2025. A Journey of Functional Test Evolution in the Public Sector. In *Anais do X Simpósio Brasileiro de Testes de Software Sistemático e Automatizado (SAST)*. SBC, Recife/PE, Brasil, 147–149. doi:10.5753/sast.2025.13869.

[5] Zhiqiang Yuan et al. 2023. No More Manual Tests? Evaluating and Improving ChatGPT for Unit Test Generation. *arXiv preprint arXiv:2305.04207* (2023).

[6] da Silva, Leonardo Cesar. 2025. Investigando o uso de Grandes Modelos de Linguagens (LLMs) na Elaboração de Casos de Teste. Dissertação de Mestrado, Universidade Federal do Rio de Janeiro.