2º Encontro da SBPC em MS/
XI ENEPEX / XIX ENEPE/
22ª SNCT - UEMS / UFGD 2025

TESTES UNITÁRIOS DE SOFTWARE COM APOIO DE LARGE LANGUAGE MODELS

Instituição: Universidade Estadual de Mato Grosso do Sul.

Área temática: Ciência da Computação.

SOUZA, Igor Roberto Michalski de[1] (rgm47539@comp.uems.br); **RECALCATTI,** João Pedro[2] (rgm49117@comp.uems.br); **PRATES,** Jorge Marques[3] (jprates@uems.br).

[1] – Discente do Curso de Ciência da Computação, UEMS;
[2] – Discente do Curso de Sistemas de Informação, UEMS;
[3] – Docente do Curso de Sistemas de Informação, UEMS.

A geração de testes de software é uma atividade essencial para garantir a qualidade e a confiabilidade dos sistemas, porém, o processo manual é demorado e propenso a falhas. A ascensão dos Modelos de Linguagem de Grande Escala (LLMs) oferece uma abordagem promissora para automatizar e aprimorar esta tarefa, gerando economia de recursos ao identificar defeitos no início do desenvolvimento. O objetivo principal deste projeto foi comparar quantitativamente a eficácia da ferramenta tradicional Pynguin com três LLMs modernos, GPT-4o, Gemini 2.5 Pro e DeepSeek, na geração de testes unitários para funções em Python. Como objetivos secundários, buscou-se medir a capacidade de cada abordagem em alcançar alta cobertura de ramos, avaliar a eficácia na detecção de falhas por meio de testes de mutação e analisar o impacto de diferentes estratégias de engenharia de prompt. As três estratégias de prompt testadas foram: uma abordagem básica (zero-shot simples), uma detalhada (zero-shot com contexto e requisitos) e uma com exemplos (few-shot), para avaliar como a sofisticação da instrução afeta a qualidade do teste gerado. A metodologia consistiu em um experimento controlado e repetível, utilizando um conjunto diversificado de sete funções-alvo. A seleção buscou variar a complexidade e o domínio de aplicação, incluindo um algoritmo clássico (Crivo de Eratóstenes) e funções realistas que abrangem validação de dados, manipulação de strings e lógica financeira, garantindo que os resultados fossem verificáveis e que os testes operassem sobre tipos de dados padrão do Python. A qualidade das suítes de teste foi avaliada por um conjunto abrangente de métricas, incluindo a cobertura de ramos, o volume de testes gerados e a natureza dos mesmos (como o uso de falhas esperadas, ou xfail). A métrica central, no entanto, foi a pontuação de mutação, obtida com a ferramenta Cosmic Ray, para medir a capacidade real de detecção de falhas. Os resultados revelaram uma narrativa clara e consistente. Embora a ferramenta tradicional e os LLMs tenham alcançado uma cobertura de ramos similarmente alta e quase perfeita, com médias superiores a 98%, a capacidade de detecção de falhas apresentou uma grande disparidade. A pontuação de mutação média do Pynguin foi de apenas 38,0%, enquanto os LLMs, em média, superaram 86%, demonstrando uma eficácia drasticamente superior. Essa diferença origina-se de uma divergência fundamental de abordagem: o método puramente estrutural do Pynguin, que otimiza para a cobertura, contrasta com a capacidade dos LLMs de inferir a semântica e a intenção do código. Os LLMs demonstraram maior aptidão para gerar asserções que validam a lógica de negócio e o "caminho feliz" do código, não apenas seus caminhos de erro. A formulação dos prompts também se mostrou um fator crítico, com estratégias mais elaboradas geralmente produzindo testes mais robustos, embora a tendência não seja estritamente linear, variando conforme o modelo e a função. Conclui-se que os LLMs não apenas automatizam, mas evoluem a geração de testes de uma otimização estrutural para uma validação semântica, demonstrando maior eficácia na criação de testes que refletem a lógica de negócio.

PALAVRAS-CHAVE: Engenharia de Software, Teste de Software, Modelos de Linguagem de Grande Escala.

AGRADECIMENTOS: Agradecemos à Universidade Estadual de Mato Grosso do Sul (UEMS) pelo apoio institucional ao desenvolvimento deste projeto de pesquisa.