Relatório de Análise Comparativa
COMPARANDO SBST VS. LLMS

11322E0026 - Marcelo Bispo dos Santos

Instituto de Ciências Exatas – Universidade Federal de Juiz de Fora (UFJF)
Departamento de Ciência da Computação
Disciplina: 1322008 - Inteligência Artificial na Engenharia de Software

1 INTRODUÇÃO

Este relatório compara a eficácia e manutenção de testes gerados por EvoSuite (SBST) e IA Generativa (LLM). As ferramentas foram executadas, os testes gerados e posteriormente validados via PITest (Mutation Testing).

Link do repositório GitHub: https://github.com/santosmarcelob/sbst-vs-llms

2 TABELA COMPARATIVA DE MÉTRICAS

A tabela abaixo resume os dados obtidos na execução real das ferramentas.

Métrica	**SBST (EvoSuite)**	**LLM (Assistida)**
Cobertura de Linhas (Line Coverage)	100%	100%
Cobertura de Ramos (Branch Coverage)	100%	100%
Eficácia Real (Mutation Score)	77% (23/30 mortos)	93% (28/30 mortos)
Mutantes Vivos (Falhas não detectadas)	7	2

Observação: A suíte LLM demonstrou eficácia superior (93% vs 77%) ao detectar casos de borda sutis que a abordagem evolutiva não capturou, mesmo com alta cobertura estrutural.

3 EXEMPLOS DE BUGS - MUTANTES ENCONTRADOS

A análise de mutação revelou diferenças críticas na capacidade de detecção de falhas:

1. Mutante de Condição de Borda (**ConditionalsBoundaryMutator**):
   - Cenário: Alteração do operador “**>=**” para “**>**” na verificação de peso.
   - Resultado SBST: O mutante sobreviveu. O EvoSuite gerou inputs que satisfaziam a cobertura do ramo, mas não testaram o valor limite exato, permitindo que a falha passasse despercebida.
   - Resultado LLM: O mutante foi morto. A suíte LLM incluiu testes explícitos para limites (ex: **testFragileWeightBoundary)**, garantindo que a lógica estivesse correta no ponto de transição.
2. Mutante de Negação (**NegateConditionalsMutator**):
   - Cenário: Inversão de lógica em condicionais simples.
   - Resultado: Ambas as suítes foram eficazes, matando 100% destes mutantes gerados.

4 ANÁLISE CRÍTICA SOBRE MANUTENÇÃO E ESFORÇO DE CORREÇÃO

Comparação qualitativa focada no esforço exigido do desenvolvedor.

4.1 SBST (EvoSuite)

- Legibilidade: O código gerado é ilegível para humanos (ex: variáveis **double0**, **class0**), focado apenas em satisfazer a máquina.
- Esforço de Correção Manual: Altíssimo. Quando um teste falha ou a regra de negócio muda, entender o que o teste **test04** está validando requer engenharia reverso do teste. A correção geralmente implica em descartar a suíte e gerar uma nova, perdendo histórico.
- Manutenção: Inviável para ser mantido manualmente ("Brittle tests").

4.1 LLM (Geração Assistida)

- Legibilidade: Código semântico, com nomes de métodos descritivos (ex: **shouldThrowExceptionForInvalidInputs**) e uso de recursos modernos do JUnit 5 (**@ParameterizedTest**). Funciona como documentação viva.
- Esforço de Correção Manual: Baixo. Se uma regra muda (ex: taxa aumenta de 10% para 15%), o desenvolvedor localiza facilmente o teste relevante e atualiza a asserção, similar ao trabalho com código escrito manualmente.
- Manutenção: A suíte é sustentável a longo prazo e pode ser integrada ao codebase principal.

5 CONCLUSÃO

A comparação demonstra que, para testes de unidade focados em lógica de negócio:

1. LLM superou SBST em eficácia real, matando mais mutantes de borda.
2. LLM é vastamente superior em manutenção, gerando artefatos que agregam valor ao projeto, enquanto SBST gera artefatos descartáveis úteis apenas para validação pontual (smoke testing/crash detection).

REFERÊNCIAS

EvoSuite. Automatic Test Suite Generation for Java. Disponível em: https://www.evosuite.org/. Acesso em: 20 jan. 2026.

PITest. Mutation Testing for Java and the JVM. Disponível em: https://pitest.org/. Acesso em: 20 jan. 2026.