www.revistaft.com.br

Revista FT | ISSN 1678-0817 | v. 30, n. 164, 2026 | DOI: 10.69849/84apx086

Desenvolvimento de software assistido por inteligência artificial: uma análise comparativa entre ambientes conversacionais e ferramentas agênticas.

AI-Assisted software development: a comparative analysis of conversational environments and agentic tools.

Andrea Barboza Proto Sardi[1]
Adriano Soares de Oliveira Bailão[2]
Valber Sardi Lopes[3]
Manoel Henrique Reis de Oliveira

RESUMO: A adoção de Modelos de Linguagem de Grande Escala (LLMs) tem transformado o ciclo de desenvolvimento de software, oferecendo ganhos expressivos na prototipação rápida. Contudo, o uso de interfaces puramente conversacionais frequentemente resulta em perda de coerência arquitetural e degradação do código ao longo de múltiplas iterações. Este trabalho analisa criticamente o processo de construção de uma aplicação web de monitoramento de sensores (PHP e persistência em XML), comparando a geração inicial em ambiente conversacional (Claude Chat) com o refinamento estruturado em ferramenta agêntica orientada a código (Claude Code). Os resultados evidenciam a ocorrência do fenômeno de deriva de código (*code drift*) nas interações em chat, além de falhas de integração, vulnerabilidades de segurança (*Cross-Site Scripting*) e incompatibilidades de versão. Em contrapartida, a transição para um ambiente agêntico aliado à persistência de contexto em arquivos Markdown (CLAUDE.md e README.md) e à auditoria técnica modular reduziu o consumo de tokens, eliminou falhas críticas e assegurou a estabilidade do sistema. Discute-se ainda a inviabilidade de modelos locais (Gemma 4) em infraestruturas computacionais restritas frente à eficiência de ambientes em nuvem.

Palavras-chave: Engenharia de Software Assistida por IA; Modelos de Linguagem; Deriva de Código; Ferramentas Agênticas; Persistência de Contexto.

ABSTRACT: The adoption of Large Language Models (LLMs) has transformed the software development cycle, offering significant gains in rapid prototyping. However, the use of purely conversational interfaces frequently results in a loss of architectural coherence and code degradation across multiple iterations. This paper critically analyzes the construction

[1] Programa de Pós-Graduação em Ciências Agrárias, Instituto Federal de Educação, Ciência e Tecnologia Goiano (IF Goiano) – Câmpus Rio Verde, GO, Brasil. E-mail: andrea.sardi@ifgoiano.edu.br
[2] Programa de Pós-Graduação em Ciências Agrárias, Instituto Federal de Educação, Ciência e Tecnologia Goiano (IF Goiano) – Câmpus Rio Verde, GO, Brasil. E-mail: bailao.adriano@ifgoiano.edu.br
[3] Programa de Pós-Graduação em Ciências Agrárias, Instituto Federal de Educação, Ciência e Tecnologia Goiano (IF Goiano) – Câmpus Rio Verde, GO, Brasil. E-mail: valber.lopes@ifgoiano.edu.br

process of a sensor monitoring web application (PHP and XML persistence), comparing initial generation in a conversational environment (Claude Chat) with structured refinement in a code-oriented agentic tool (Claude Code). The results highlight the occurrence of the code drift phenomenon in chat interactions, in addition to integration failures, security vulnerabilities (Cross-Site Scripting), and version incompatibilities. In contrast, the transition to an agentic environment combined with context persistence in Markdown files (CLAUDE.md and README.md) and modular technical auditing reduced token consumption, eliminated critical failures, and ensured system stability. Furthermore, the infeasibility of local models (Gemma 4) in restricted computational infrastructures compared to the efficiency of cloud environments is discussed.

Keywords: AI-Assisted Software Engineering; Language Models; Code Drift; Agentic Tools; Context Persistence.

1. INTRODUÇÃO

A integração de ferramentas baseadas em Inteligência Artificial (IA) e Modelos de Linguagem de Grande Escala (LLMs) consolidou-se como um recurso relevante para acelerar etapas da engenharia de software, desde a concepção arquitetural até a geração e refatoração de código (Peng et al., 2023; Barke et al., 2022). Entretanto, a eficácia dessas ferramentas depende diretamente da forma como o contexto do projeto é gerenciado e da arquitetura da interface utilizada pelo desenvolvedor.

Em ambientes puramente conversacionais (*chatbots*), observa-se que, embora a geração inicial de código apresente boa estrutura sintática quando guiada por engenharia de prompt detalhada (White et al., 2023; OpenAI, 2024), a evolução iterativa de sistemas compostos por múltiplos arquivos enfrenta severas limitações (Liu et al., 2023b; Ding et al., 2023). Estudos empíricos demonstram que conversas de múltiplos turnos degradam substancialmente a precisão e a segurança do código gerado (Laban et al., 2025; Rawal et al., 2025), fenômeno exacerbado pela perda de atenção em janelas de contexto extensas (Liu et al., 2023a) e pela ocorrência de alucinações em nível de repositório (Zhang et al., 2024).

Diante desse cenário, este artigo tem como objetivo analisar empiricamente as limitações dos ambientes conversacionais no desenvolvimento de uma aplicação web completa para monitoramento de sensores em PHP e XML, avaliando os impactos da migração para um ambiente agêntico de desenvolvimento (Claude Code), a adoção de documentação leve para persistência de contexto e a viabilidade técnico-econômica de modelos locais em comparação a soluções em nuvem.

2. METODOLOGIA

O estudo foi conduzido por meio de um experimento prático de desenvolvimento incremental, estruturado em quatro etapas: (i) especificação e geração inicial via prompt detalhado em ambiente conversacional; (ii) diagnóstico de falhas de execução e deriva de código; (iii) migração para ambiente agêntico com auditoria técnica e persistência de contexto; e (iv) avaliação de viabilidade de infraestrutura computacional.

2.1 Arquitetura da Aplicação Experimental

Definiu-se como objeto de estudo uma aplicação web autônoma para registro e visualização de leituras de sensores (temperatura, umidade, pressão, luminosidade, CO_2 e movimento), desenvolvida em PHP puro (compatível com PHP 7.4+) sem uso de frameworks ou banco de dados relacional, empregando persistência

local em arquivo XML e visualização gráfica via biblioteca Chart.js. A arquitetura foi dividida em quatro artefatos principais:

- `index.php` : formulário de entrada com validação HTML5 e JavaScript, campos condicionais dinâmicos para sensores de temperatura (unidades °C, °F, K e faixas operacionais) e envio via método `POST` .
- `recebe_dados.php` : módulo de recepção, sanitização, geração de identificador único (UUID v4), cálculo automático de alertas de temperatura e gravação estruturada no arquivo XML.
- `dados_sensores.xml` : repositório persistente gerado automaticamente na primeira submissão, contendo metadados da leitura e status operacional.
- `dashboard.php` : painel analítico com indicadores-chave de desempenho (KPIs), tabela ordenável com filtros por tipo de sensor, badges de alerta e gráficos temporais e de distribuição.

2.2 Fluxo de Desenvolvimento e Critérios de Auditoria

Na primeira fase, submeteu-se uma especificação técnica detalhada (contendo paleta visual, estrutura de tags XML, requisitos de compatibilidade e regras de negócio) ao modelo Sonnet 4.6 na interface conversacional Claude Chat. Após a identificação de falhas em ambiente local (servidor Apache em sistema operacional Linux), o código-base foi transferido para a ferramenta agêntica Claude Code (Anthropic, 2026a).

Para mitigar o consumo excessivo de tokens e evitar reescritas destrutivas, estabeleceu-se uma estratégia de auditoria técnica separada da etapa de correção, delimitando o escopo de análise por arquivo e introduzindo artefatos de documentação em Markdown (`README.md` e `CLAUDE.md`). A auditoria avaliou quatro dimensões: integridade do fluxo de dados, segurança contra injeção de código, permissões de escrita em sistema de arquivos e robustez no tratamento de exceções.

3. RESULTADOS E DISCUSSÃO

3.1 Limitações do Ambiente Conversacional e Deriva de Código

Embora o prompt inicial tenha gerado três arquivos totalizando 2.231 linhas de código com estrutura visual coerente, a execução em ambiente local revelou falhas críticas de integração e compatibilidade.
Em `recebe_dados.php` , o modelo utilizou a constante `ENT_XML1` na função de sanitização — recurso introduzido apenas no PHP 8.1 — provocando erro fatal silencioso na versão PHP 7.4 especificada nos requisitos. Além disso, verificou-se inconsistência no método de envio do formulário (`GET` em vez de `POST`) e problemas de dupla codificação (*double-encoding*) antes da gravação no XML.

As tentativas subsequentes de correção por meio de pedidos incrementais no chat (tais como solicitações pontuais para ajustar trechos isolados) desencadearam o fenômeno de **deriva de código (*code drift*)**. Nesse processo, modificações locais descaracterizaram a arquitetura original, quebraram interfaces previamente funcionais entre os arquivos e introduziram inconsistências acumulativas, corroborando os achados de Jimenez et al. (2023) e Vaithilingam et al. (2022) sobre a necessidade de validação humana rigorosa e controle global de contexto.

3.2 Auditoria Técnica e Refinamento em Ambiente Agêntico

A transição para o Claude Code permitiu inspecionar o diretório do projeto de forma integrada e aplicar correções rastreáveis (Anthropic, 2025a). A auditoria técnica identificou e solucionou vulnerabilidades de segurança e falhas de concorrência, conforme sintetizado na Tabela 1.

Componente / Dimensão	Problema Identificado	Causa Raiz	Solução Implementada
index.php (Fluxo HTTP)	Exposição de parâmetros na URL e falha no processamento	Divergência entre método do formulário e recepção	Padronização estrita para envio via method="POST"
recebe_dados.php (Persistência)	Erro fatal no PHP 7.4 e risco de corrupção do XML	Uso de ENT_XML1 e ausência de bloqueio de escrita	Remoção de *double-encoding*, uso de flock(LOCK_EX) , escrita atômica (.tmp para rename()) e backup .bak
dashboard.php (Segurança XSS)	Vulnerabilidade à injeção de scripts maliciosos	Saída direta de dados externos sem escapamento	Aplicação de htmlspecialchars(..., ENT_QUOTES, 'UTF-8') em todas as saídas HTML (OWASP, 2024)
Ambiente Linux (/var/www/html)	Falha na criação automática do arquivo XML	Restrição de permissão de escrita no diretório	Ajuste de proprietário (chown wwwdata) e permissões (chmod) no servidor Apache

Sob o aspecto da segurança da informação, destacou-se a distinção entre validação de entrada e escapamento de saída: enquanto o back-end passou a validar padrões via expressões regulares (SEN-[0-9]{4}) e listas brancas (*whitelists*) de tipos de sensores, a função htmlspecialchars foi aplicada exclusivamente na camada de apresentação para mitigar ataques de *Cross-Site Scripting* (XSS) sem corromper os dados armazenados no XML (MDN, 2024; The PHP Group, 2024).

3.3 Persistência de Contexto (.md), Planejamento e Eficiência de Tokens

Embora arquivos Markdown (Gruber, 2004) não sejam requisitos de execução para sistemas PHP/XML, a criação estruturada dos arquivos README.md (contendo diagramas arquiteturais em três níveis C4) e CLAUDE.md (documentando convenções de escrita atômica, bloqueio de arquivos e regras de validação) demonstrou impacto direto na eficiência do desenvolvimento (Anthropic, 2026b; Anthropic, 2025b).

O monitoramento de consumo indicou a utilização de 37,8 mil tokens de uma janela de 200 mil (19% da capacidade total). Ao persistir as diretrizes arquiteturais em CLAUDE.md , eliminou-se a necessidade de reenviar instruções extensas a cada interação, alinhando o fluxo às práticas de otimização de cache e compressão de contexto (Anthropic, 2024; Jiang et al., 2023). Observou-se ainda que, devido à baixa complexidade arquitetural e ao escopo bem delimitado do sistema, a não utilização de módulos avançados de extensão (*skills*) mostrou-se adequada ao princípio KISS (*Keep It Simple, Stupid*), evitando sobrecarga de contexto.

Por outro lado, a análise retrospectiva evidenciou que a ausência do modo de planejamento prévio (/plan) nas interações iniciais contribuiu para o retrabalho observado (Anthropic, 2026c). A adoção de uma etapa explícita de exploração e planejamento antes da escrita de código tende a reduzir inconsistências estruturais em projetos multi-arquivo.

3.4 Viabilidade Técnica e Econômica: Modelos Locais vs. Nuvem

Avaliou-se também a viabilidade de adoção de modelos abertos executados localmente, especificamente o Gemma 4 (Google, 2026), como alternativa gratuita para ambientes educacionais e de pesquisa. Constatou-se que a gratuidade da licença do modelo não elimina custos indiretos severos de infraestrutura: a execução local de variantes capazes de lidar com geração de código multi-arquivo exige elevada capacidade de memória RAM e placas gráficas dedicadas (Unsloth AI, 2024; Google, 2025).

Sob a ótica econômica, a aquisição de hardware compatível demanda investimentos iniciais estimados a partir de R$ 7.000,00, tornando-se impeditiva para laboratórios institucionais ou equipamentos pessoais convencionais. Em contrapartida, o uso de ambientes agênticos em nuvem sob assinatura (como o plano Pro avaliado, com custo aproximado de R$ 110,00 mensais) oferece acesso imediato a modelos de fronteira, gerenciamento otimizado de janelas de contexto e estabilidade operacional sem exigência de hardware local robusto.

4. CONCLUSÃO

A transição de um protótipo gerado em ambiente conversacional para uma aplicação web validada e segura demonstra que a eficácia do desenvolvimento de software assistido por IA não reside apenas na qualidade do prompt inicial, mas na escolha de ferramentas adequadas e na aplicação disciplinada de princípios da engenharia de software. Enquanto ambientes de chat favorecem a deriva de código e falhas de integração em projetos multi-arquivo, ferramentas agênticas como o Claude Code, combinadas à persistência de contexto em arquivos Markdown (CLAUDE.md) e à auditoria técnica modular, asseguram rastreabilidade, economia de tokens e integridade arquitetural.

Conclui-se que a IA atua de forma mais produtiva como instrumento de apoio sob supervisão técnica ativa do desenvolvedor, sendo os ambientes em nuvem atualmente mais viáveis técnica e economicamente do que modelos locais para contextos com restrição de infraestrutura. Como trabalhos futuros, sugere-se a avaliação quantitativa sistemática do impacto do modo de planejamento (/plan) na redução da taxa de defeitos e do consumo de tokens em sistemas de maior complexidade.

REFERÊNCIAS

ANTHROPIC. *Prompt Caching with Claude*. 2024. Disponível em: https://www.anthropic.com/news/promptcaching. Acesso em: 29 abr. 2026.

ANTHROPIC. *Best Practices for Claude Code*. 2025a. Disponível em: https://www.anthropic.com/engineering/claude-code-best-practices. Acesso em: 29 abr. 2026.

ANTHROPIC. *How Anthropic Teams Use Claude Code*. 2025b. Disponível em: https://wwwcdn.anthropic.com/58284b19e702b49db9302d5b6f135ad8871e7658.pdf. Acesso em: 29 abr. 2026.

ANTHROPIC. *Claude Code Overview*. 2026a. Disponível em: https://docs.anthropic.com/en/docs/claudecode/overview. Acesso em: 29 abr. 2026.

ANTHROPIC. *Memory: How Claude Remembers Your Project*. 2026b. Disponível em: https://docs.anthropic.com/en/docs/claude-code/memory. Acesso em: 29 abr. 2026.

ANTHROPIC. *Common Workflows: Plan Mode for Safe Code Analysis*. 2026c. Disponível em: https://docs.anthropic.com/en/docs/claude-code/common-workflows. Acesso em: 29 abr. 2026.

BARKE, S.; JAMES, M. B.; POLIKARPOVA, N. Grounded Copilot: How Programmers Interact with Code-Generating Models. *Proceedings of the ACM on Programming Languages (OOPSLA)*, v. 6, 2022. Disponível em: https://dl.acm.org/doi/10.1145/3586030. Acesso em: 29 abr. 2026.

DING, Y. et al. CrossCodeEval: A Diverse and Multilingual Benchmark for Cross-File Code Completion. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. Disponível em: https://arxiv.org/abs/2310.11248. Acesso em: 29 abr. 2026.

GOOGLE. *Gemma 3 QAT Models: Bringing State-of-the-Art AI to Consumer GPUs*. 2025. Disponível em: https://developers.googleblog.com/en/gemma-3-quantized-aware-trained-state-of-the-art-ai-to-consumergpus/. Acesso em: 29 abr. 2026.

GOOGLE. *Gemma Model Overview*. 2026. Disponível em: https://ai.google.dev/gemma/docs/core. Acesso em: 29 abr. 2026.

GRUBER, J. *Markdown*. 2004. Disponível em: https://daringfireball.net/projects/markdown/. Acesso em: 29 abr. 2026.

JIANG, H. et al. LongLLMLingua: Accelerating and Enhancing LLMs in Long Context Scenarios via Prompt Compression. *arXiv preprint arXiv:2310.06839*, 2023. Disponível em: https://arxiv.org/abs/2310.06839. Acesso em: 29 abr. 2026.

JIMENEZ, C. E. et al. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? *International Conference on Learning Representations (ICLR)*, 2024. Disponível em: https://arxiv.org/abs/2310.06770. Acesso em: 29 abr. 2026.

LABAN, P.; HAYASHI, H.; ZHOU, Y.; NEVILLE, J. LLMs Get Lost In Multi-Turn Conversation. *arXiv preprint arXiv:2505.06120*, 2025. Disponível em: https://arxiv.org/abs/2505.06120. Acesso em: 29 abr. 2026.

LIU, N. F. et al. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics (TACL)*, v. 12, p. 157–173, 2023a. Disponível em: https://arxiv.org/abs/2307.03172. Acesso em: 29 abr. 2026.

LIU, T.; XU, C.; MCAULEY, J. RepoBench: Benchmarking Repository-Level Code Auto-Completion Systems. *International Conference on Learning Representations (ICLR)*, 2024. Disponível em: https://arxiv.org/abs/2306.03091. Acesso em: 29 abr. 2026.

MOZILLA DEVELOPER NETWORK (MDN). *Cross-site scripting (XSS) – Security*. 2024. Disponível em: https://developer.mozilla.org/en-US/docs/Web/Security/Attacks/XSS. Acesso em: 29 abr. 2026.

OPENAI. *Best Practices for Prompt Engineering*. 2024. Disponível em: https://platform.openai.com/docs/guides/prompt-engineering. Acesso em: 29 abr. 2026.

OWASP FOUNDATION. *Cross Site Scripting Prevention Cheat Sheet*. 2024. Disponível em: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html. Acesso em: 29 abr. 2026.

PENG, S. et al. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. *arXiv preprint arXiv:2302.06590*, 2023. Disponível em: https://arxiv.org/abs/2302.06590. Acesso em: 29 abr. 2026.

RAWAL, R. et al. Benchmarking Correctness and Security in Multi-Turn Code Generation (MT-Sec). *arXiv preprint arXiv:2509.21420*, 2025. Disponível em: https://arxiv.org/abs/2509.21420. Acesso em: 29 abr. 2026.

THE PHP GROUP. *PHP Manual: htmlspecialchars*. 2024. Disponível em: https://www.php.net/manual/en/function.htmlspecialchars.php. Acesso em: 29 abr. 2026.

UNSLOTH AI. *Gemma 4: Model Requirements and Performance Guide*. 2024. Disponível em: https://unsloth.ai/docs/models/gemma-4. Acesso em: 29 abr. 2026.

VAITHILINGAM, P.; ZHANG, T.; GLASSMAN, E. L. Expectation vs. Experience: Evaluating the Usability of Code

Generation Tools Powered by LLMs. *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, 2022. Disponível em: https://dl.acm.org/doi/10.1145/3491101.3519665. Acesso em: 29 abr. 2026.

WHITE, J. et al. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. *arXiv preprint arXiv:2302.11382*, 2023. Disponível em: https://arxiv.org/abs/2302.11382. Acesso em: 29 abr. 2026.

ZHANG, Z. et al. LLM Hallucinations in Practical Code Generation: Phenomena, Mechanism, and Mitigation. *Proceedings of the ACM on Software Engineering (PACMSE)*, 2024. Disponível em: https://arxiv.org/abs/2409.20550. Acesso em: 29 abr. 2026.