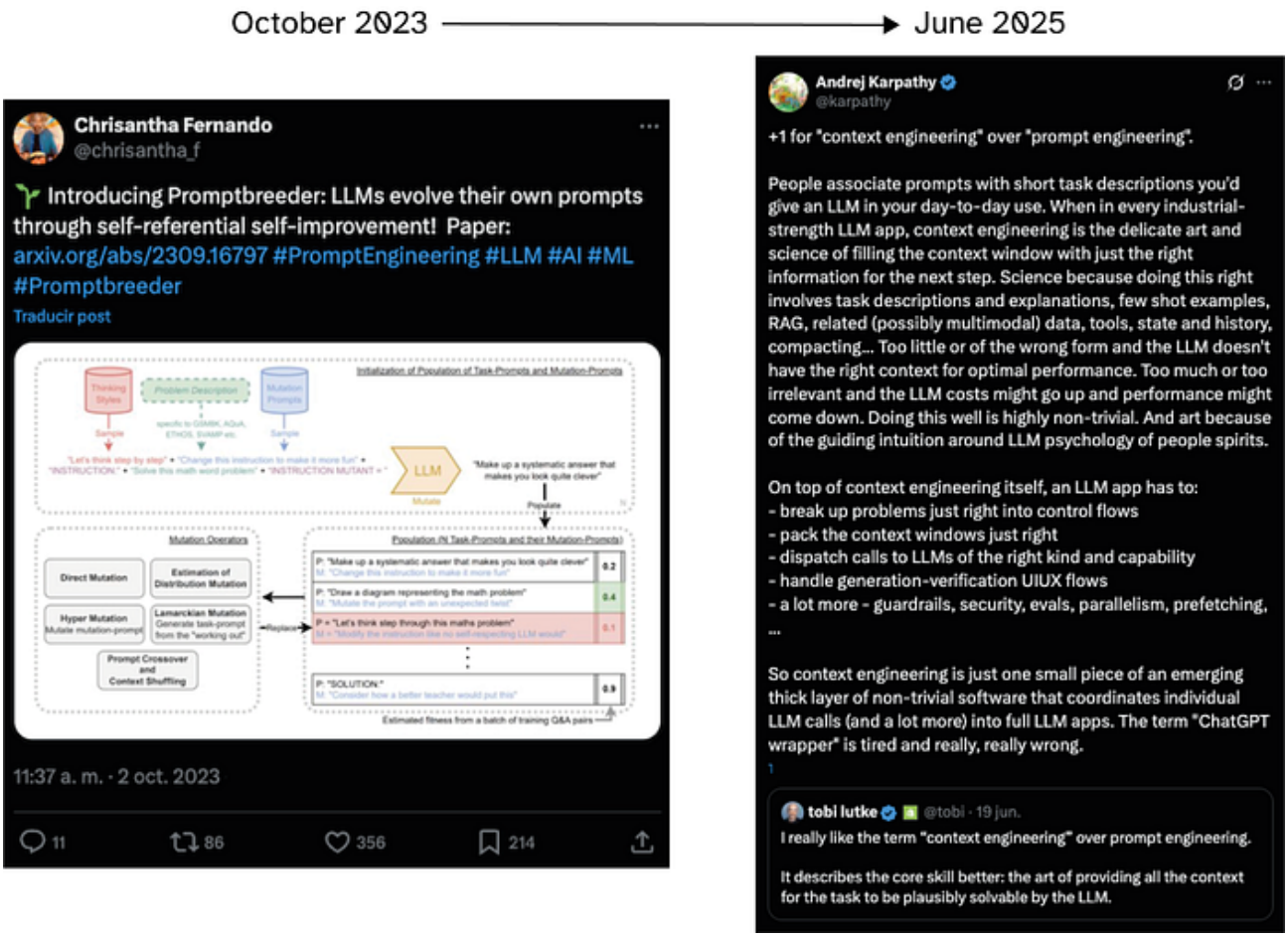
Context Engineering vs. Prompt Engineering

The work matters more than the name

Javier Marin • 9 min read • 2025-10-03

https://medium.com/data-science-collective/context-engineering-vs-prompt-engineering-3493c2925e99



Context engineering emerged in mid-2025 as the evolution of prompt engineering, gaining rapid institutional backing but facing skepticism about whether it represents genuine innovation or rebranding. While prompt engineering peaked in 2023 with six-figure salaries before declining dramatically by 2025, context engineering has secured adoption from major AI companies (Anthropic, LangChain, LlamaIndex) and addresses production challenges that simple prompting cannot solve.

Figure 1: *Prompt engineering peek and the origin of context engineering. Source: X.com*

The term become settled in June 2025 when Shopify CEO Tobi Lutke and former OpenAI researcher Andrej Karpathy publicly endorsed it on X, triggering rapid adoption. In less than a month, the first comprehensive academic survey analyzing 1,300+ papers formalized it as a distinct discipline. Yet this institutional validation coexists with pointed criticism from practitioners who view it as repackaged concepts from information retrieval, RAG, and system design.

Instructions vs. Ecosystems

Prompt engineering focuses on crafting textual instructions given to language models - refining word choice, structure, examples, and output formats within individual interactions.

> *A formal definition could be: designing task-specific instructions to enhance model efficacy without modifying core model parameters.*

Context engineering includes the complete information architecture surrounding an AI system - system prompts, conversation history, retrieved documents, available tools, memory systems, and dynamic state management. Anthropic defines it as “designing and managing the complete information ecosystem that an LLM accesses during operation”.

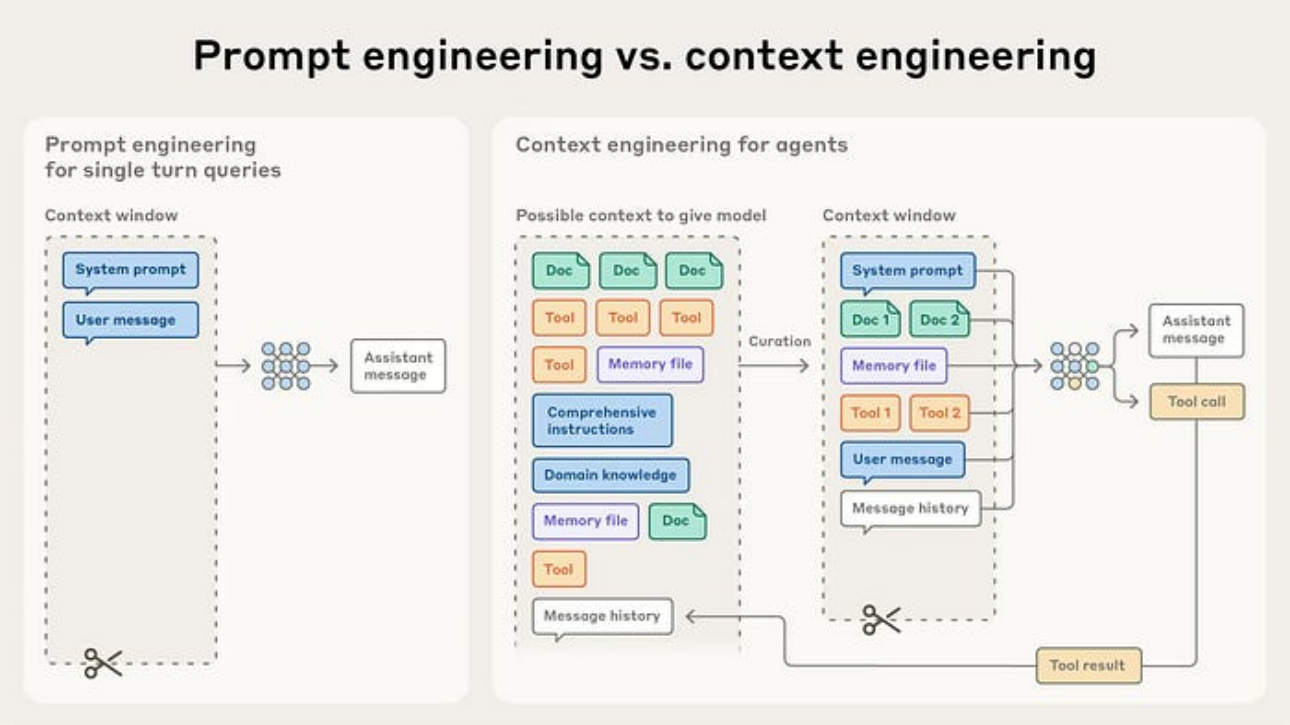


Figure 2: *Context engineering ecosystem. Source: https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents*

The technical difference is clear. Prompt engineering uses zero-shot prompting, few-shot examples, chain-of-thought reasoning, and role-based instructions. Context engineering incorporates these but adds multiple layers: system prompts (procedural memory), message history (short-term memory), episodic memory (user preferences), semantic memory (RAG-retrieved knowledge), tool schemas, and dynamic context assembly mechanisms.

Adoption And Momentum

Prompt engineering peaked in 2023 (Figure 1) and declined sharply by 2025, while context engineering is brand new, showing rapid growth from June 2025 onwards. After Lutke’s June 18 tweet supporting the term (Figure 1), industry response was swift. Anthropic published comprehensive technical guidance [here], LangChain created GitHub repositories with implementation examples [here], and by mid-2025, DataCamp, Coursera, IBM, and Microsoft offered context engineering courses. In his blog, Drew Breunig claimed that context engineering reached 25% of prompt engineering’s search volume within one month - an unusually fast adoption curve.

> We have seen how prompt engineering performs well for constrained, single-turn tasks as classification, basic summarization, straightforward Q&A, creative writing, and template-based generation. But context engineering becomes essential for complex, multi-turn, mission-critical applications requiring integration of multiple data sources, conversation history, regulatory constraints, and dynamic tool use.

Healthcare applications show success rates improving from 30% to 90% when moving from basic prompting to full context engineering with integrated patient history and episodic memory. Financial services firms implement hybrid approaches: long-term semantic memory in vector databases, short-term working memory in conversation buffers, and just-in-time retrieval. Cost optimization is critical - naive approaches can exceed $50 per query, while engineered systems achieve comparable accuracy at 4% of the cost.

Production challenges include context decay (performance drops beyond 32K tokens), context drift (conflicting information accumulating), tool confusion (models struggle with multiple tools), and lost-in-the-middle phenomenon (missing details in lengthy contexts). Solutions involve summarization, temporal ordering, RAG-based tool retrieval, and structured formatting with critical information prominently placed.

Ecosystem Maturity

Some specialized tools have immediately adopted context management approach. LangChain and LangGraph evolved from prompt chaining toward comprehensive context management, providing state management, memory systems, tool integration, and context compression. LangSmith offers observability for tracing context flows and monitoring failures.

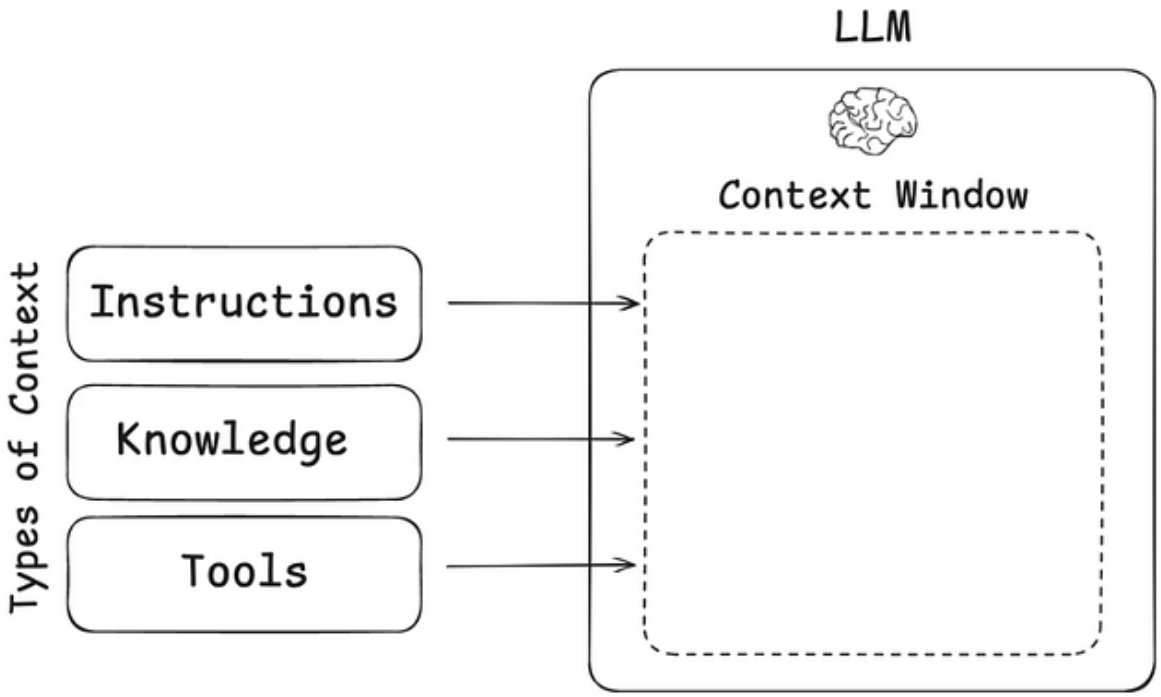


***Figure 3**: Context types commonly used in LLM applications. Source:*

https://blog.langchain.com/context-engineering-for-agents/

Anthropic's infrastructure makes context engineering first-class. Claude features 200K-token windows, the Claude Agent SDK supports context editing and memory tools, and Model Context Protocol (MCP) establishes an open standard for connecting AI models to external data sources - "USB-C for AI." MCP adoption accelerated rapidly: OpenAI, Google, enterprises including Stripe and Databricks, and IDEs like Zed and Replit integrated MCP support. Contextual Retrieval achieved 49% reduction in retrieval failures compared to standard RAG. LlamaIndex positions itself as enterprise-focused platform with RAG-as-a-service, multiple retrieval modes, and agentic retrieval. Vector databases (Chroma, Qdrant, Pinecone, Weaviate) enable semantic search infrastructure. Mem0 provides specialized memory management for AI agents.

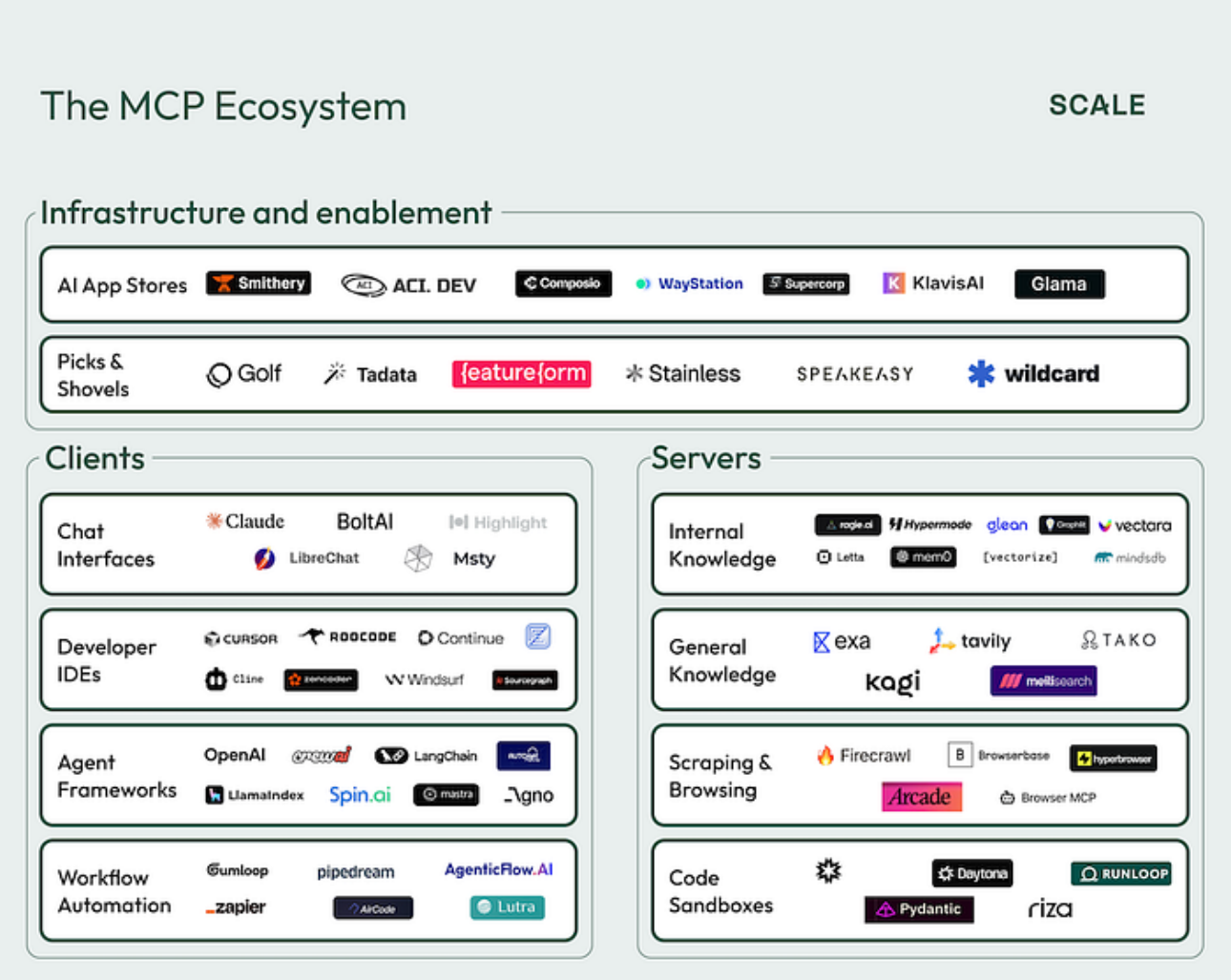


***Figure 4**: The MCP Ecosystem. Source: https://www.scalevp.com/insights/mcp-is-the-new-www/ , June 2025.*

However, the ecosystem is still fragmented and far from being mature. Multiple competing approaches exist for memory management and retrieval optimization. Integration complexity is high, requiring combination of frameworks, databases, protocols, and custom logic.

Critical Perspectives

This new approach has quickly gained its detractors. They argue that it is a "rebranding", claiming that information retrieval, knowledge management, and system integration existed long

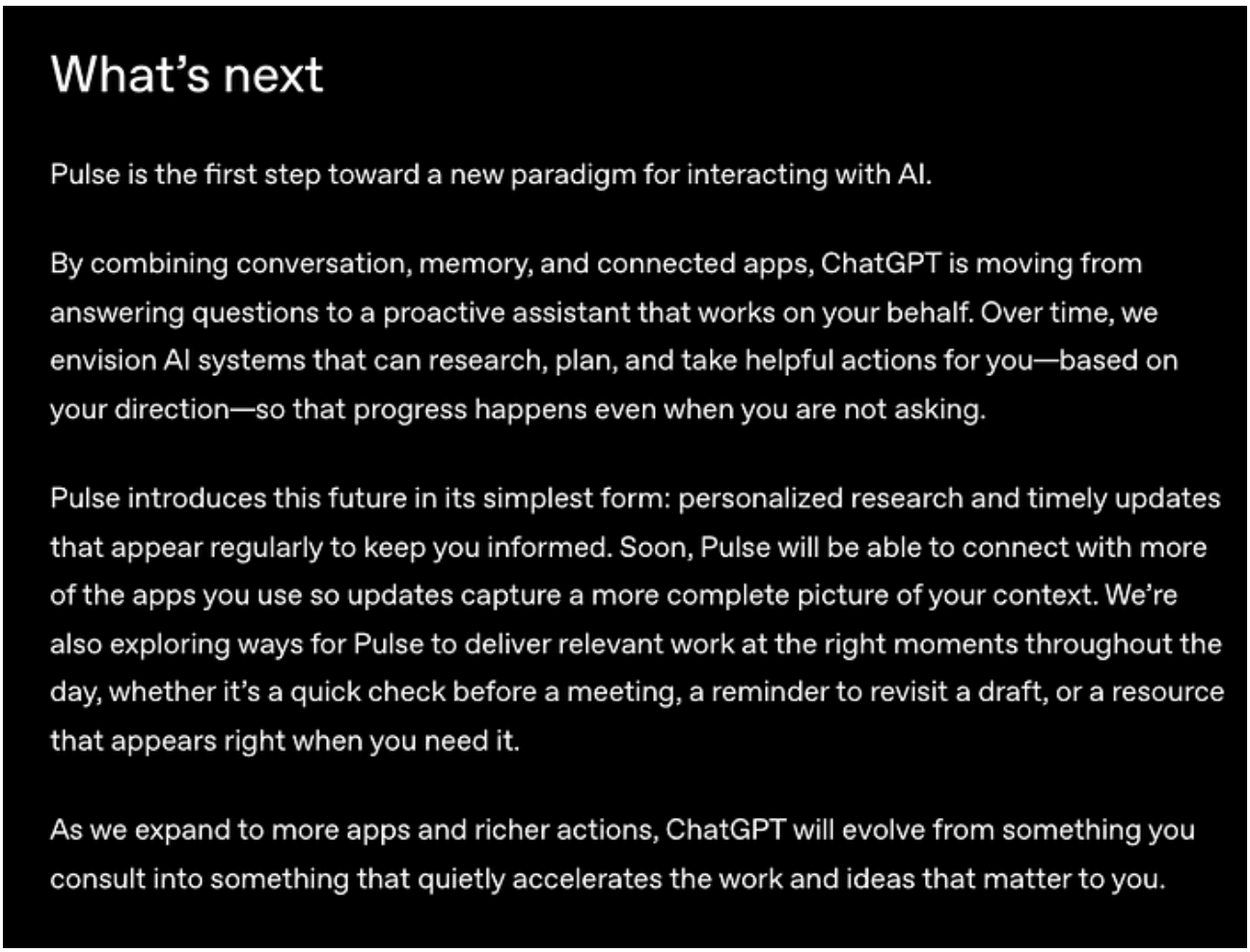
before “context engineering” terminology. RAG was established by 2020, memory systems for conversational AI date back decades. Critics view it as buzzword cycling rather than genuine advance.

The automation argument contends that manual context engineering doesn’t scale and will be superseded by automated systems. InnoGames argued it “turns developers into prompt janitors,” advocating for automated validation pipelines instead. This mirrors software engineering evolution - manual testing gave way to automation. The engineering rigor critique questions whether context engineering merits the label arguing it lacks formal methods, reproducible processes, and quantitative validation frameworks characteristic of established engineering disciplines. It’s the same questioning that prompt enginnering had before. The model improvement counterargument claims that better models will reduce context engineering needs, just as improved models made complex prompting less necessary. As models improve at handling longer contexts and understanding intent, elaborate context management should diminish.

Agentic AI: The Sustainability Argument

The strongest case for context engineering’s sustainability lies in agentic AI and multi-turn interactions. Agents plan multi-step actions, use tools dynamically, maintain state, coordinate with other agents, and adapt based on feedback. These capabilities inherently require context management beyond prompting.

Anthropic’s argues that “most agent failures are not model failures anymore, they are context failures.” Agents fail when they lack necessary information, receive contradictory context, get confused by tools, or lose track of goals - context engineering problems that won’t be solved purely by better models. Multi-turn interactions create fundamental requirements: customer service spanning multiple messages, software development across extended sessions, healthcare building longitudinal relationships. Even million-token windows fill quickly with comprehensive information, necessitating context engineering techniques. Memory systems represent requirements that won’t disappear. When AI assistants become ongoing companions rather than tools you use once and forget, they need to keep track of who you are - remembering your preferences, past conversations, and patterns they’ve learned about you. This requires memory architectures, retrieval mechanisms, and consistency management. This is the case of the new ChatGPT feature Pulse.



ChatGPT Pulse. Source: https://openai.com/index/introducing-chatgpt-pulse/

Evolution and Uncertainty

Context engineering is indeed a technical evolution but faces uncertain future as discipline. Real innovations exist - MCP protocols, LangGraph frameworks, memory systems, contextual retrieval showing 49% failure reduction. Production implementations demonstrate 30% to 90% success rate improvements, days-to-hours development compression, and 96% cost reductions.

The scope difference is genuine. Context engineering architects complete information ecosystems versus optimizing individual instructions, representing different skill sets (systems architecture vs. copywriting) and outcomes (reliable production systems vs. better responses).

Major AI companies built substantial infrastructure - Anthropic's SDK and MCP, LangChain's frameworks, LlamaIndex's platform. Production systems require these capabilities regardless of terminology. Agentic AI needs state management, memory, and tool coordination. Enterprise deployments need security boundaries and explainability - all context engineering concerns.

Despite technical validity, "context engineering" faces trajectory risks similar to "prompt engineering." Job market shows minimal direct postings - skills are being absorbed into "AI Engineer" roles. The field may follow patterns where practices outlast terminology: responsive web design became web design, DevOps became baseline expectations.

Conclusions

Context engineering skills are immediately valuable. Understanding RAG implementation, memory system design, agent state management, and tool integration enables building production AI systems that prompting alone cannot achieve.

Organizations may adopt context engineering frameworks where complexity warrants - agentic systems, multi-turn interactions, regulated industries - but integrate skills into AI/ML teams rather than creating silos.

Context engineering provides also useful framework for organizing research on RAG, memory systems, multi-agent coordination, and tool integration, but focus on fundamental problems rather than defending terminology.

The evolution in AI system design from demonstrations to production infrastructure, from one-shot interactions to persistent assistants, from isolated models to integrated systems is necessary, and ongoing.

> Whether it retains the label or becomes absorbed into broader practice, the underlying technical requirements will shape AI development for years to come. The work matters more than the name.

🙏 Thanks for reading this post

📩 **Contact me at javier@jmarin.info**

➰ Your feed-back will be highly appreciated!

References used

- Anthropic. (2025, January). *Effective context engineering for AI agents*. Retrieved from https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents
- Breunig, D. (2025, July 24). Why “context engineering” matters. *Drew Breunig’s Blog*. Retrieved from https://www.dbreunig.com/2025/07/24/why-the-term-context-engineering-matters.html
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*, 1877-1901.
- Gupta, M. (2025). Context engineering vs prompt engineering. *Medium: Data Science in Your Pocket*. Retrieved from https://medium.com/data-science-in-your-pocket/context-engineering-vs-prompt-engineering-379e9622e19d
- InnoGames. (2025). Why context engineering is overhyped (and how automation fixes it). *#InnoBlog*. Retrieved from https://blog.innogames.com/why-context-engineering-is-overhyped-and-how-automation-fixes-it/
- Ji, Y. P. (2025). Context engineering for AI agents: Lessons from building Manus. *Medium*. Retrieved from https://medium.com/@peakji/context-engineering-for-ai-agents-lessons-from-building-manus-71883f0a67f2
- LangChain. (2025). The rise of “context engineering”. *LangChain Blog*. Retrieved from https://blog.langchain.com/the-rise-of-context-engineering/

- LangChain. (2025). Context engineering for agents. *LangChain Blog*. Retrieved from https://blog.langchain.com/context-engineering-for-agents/
- Landgraf, T. (2025). Context engineering for Claude code: Mastering deep technical knowledge. *Medium*. Retrieved from https://medium.com/@tl_99311/context-engineering-for-claude-code-mastering-deep-technical-knowledge-ba
- Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys, 55*(9), 1-35.
- LlamaIndex. (2025). Context engineering - What it is, and techniques to consider. Retrieved from https://www.llamaindex.ai/blog/context-engineering-what-it-is-and-techniques-to-consider
- Lutke, T. (2025, June 18). Context engineering over prompt engineering [Tweet]. *X*. Retrieved from https://x.com/tobi/status/1935533422589399127
- McFadden, P. (2025, July). Context engineering is a mirage - The system still doesn't know. *Medium*. Retrieved from https://pmcfadden.medium.com/context-engineering-is-a-mirage-the-system-still-doesnt-know-4d8fdb07db17
- Munn, J. (2025). The rise of the context engineer. *Medium*. Retrieved from https://medium.com/@johnmunn/the-rise-of-the-context-engineer-5accaf17d5c2
- Nearform. (2025). Beyond prompt engineering: The shift to context engineering. Retrieved from https://nearform.com/digital-community/beyond-prompt-engineering-the-shift-to-context-engineering/
- Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., & Chadha, A. (2024). A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*. Retrieved from https://arxiv.org/abs/2402.07927
- Schmid, P. (2025). The new skill in AI is not prompting, it's context engineering. Retrieved from https://www.philschmid.de/context-engineering
- Schulhoff, S., Ilie, M., Balepur, N., Kahadze, K., Liu, A., Si, C., ... & Hendrycks, D. (2024). The prompt report: A systematic survey of prompting techniques. *arXiv preprint arXiv:2406.06608*. Retrieved from https://arxiv.org/abs/2406.06608
- The New Stack. (2025). Context engineering: Going beyond prompt engineering and RAG. Retrieved from https://thenewstack.io/context-engineering-going-beyond-prompt-engineering-and-rag/
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems, 35*, 24824-24837.
- Willison, S. (2025, June 27). Context engineering. Retrieved from https://simonwillison.net/2025/Jun/27/context-engineering/# Context Engineering vs. Prompt Engineering: A Comprehensive Analysis

Other mentions by Author

- medium.com | Published in Data Science Collective
- medium.com | Written by Javier Marin