УДК 004.8

Бадертдинова Елена Радитовна, доктор технических наук, ФГБОУ ВО «Казанский национальный исследовательский технологический университет», г. Казань

Совгачев Александр Андреевич, магистрант, ФГБОУ ВО «Казанский национальный исследовательский технологический университет», г. Казань

PROMPT ENGINEERING КАК ИНЖЕНЕРНАЯ ПРАКТИКА В РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Аннотация

Prompt engineering (PE), практика целенаправленного формирования инструкций для управления поведением больших языковых моделей (LLM), за 2022–2026 годы прошёл путь от эмпирического подбора формулировок до зарождающейся инженерной дисциплины. В работе представлен нарративный обзор 32 источников, охватывающий эволюцию техник PE, их влияние на качество генерации кода, индустриальные инструменты и системные ограничения дисциплины. Наиболее эмпирически обоснованными остаются детализация алгоритмических требований, уточнение формата ввода/вывода и security-ориентированные префиксы. Параллельно зафиксирован системный кризис воспроизводимости: большинство PE-техник не воспроизводятся при смене модели, а автоматические методы оптимизации (DSPy, EPiC) стабильно превосходят ручной PE.

Annotation

Prompt engineering (PE), the practice of deliberately constructing instructions to control the behavior of large language models (LLMs), has evolved over 2022–2026 from empirical trial-and-error into an emerging engineering discipline. This paper presents a narrative review of 32 sources covering the evolution of PE techniques, their impact on code generation quality, industrial tools, and systemic limitations of the discipline. The most empirically supported techniques for code

generation remain detailed algorithmic requirements, input/output format specification, and security-oriented prefixes. A systemic reproducibility crisis has been identified: most PE techniques fail to generalize across model changes, while automated optimization methods (DSPy, EPiC) consistently outperform manual PE.
Ключевые слова: prompt engineering, большие языковые модели, генерация кода, chain-of-thought, автоматическая оптимизация промптов

Keywords: prompt engineering, large language models, code generation, chain-of-thought, automatic prompt optimization

Введение

Появление больших языковых моделей общего назначения (GPT-3, 2020; GPT-4, 2023; Claude 3/4 и Llama 3, 2024–2025) радикально изменило практику разработки программного обеспечения. Качество результата оказалось критически зависимым от формулировки запроса, что породило prompt engineering (PE) — систематическое конструирование инструкций для управления поведением модели. К 2024 году появились специализированные инструменты, учебные курсы и академические обзоры; рынок PE-инструментов оценивается в 505 млн долларов с прогнозом роста до 6,7 млрд к 2034 году [31]. Одновременно оформляется критическая позиция: исследование ICLR 2024 показало, что большинство PE-техник не воспроизводятся за пределами исходной модели [22], а концепция «Promptware Engineering» (2025) характеризует текущее состояние как «promptware crisis» [20]. Цели работы: (1) систематизировать техники PE и их эффекты для генерации кода; (2) описать индустриальные инструменты; (3) критически оценить статус PE как инженерной дисциплины.

Эволюция техник Prompt Engineering

Развитие prompt engineering происходило поэтапно: от простого использования примеров в промптах до появления специализированных методов оптимизации, структурированных рассуждений и инженерных практик разработки. Табл. 1 отражает ключевые этапы эволюции и наиболее

значимые работы, сформировавшие современный подход к проектированию промптов.

Год	**Событие**
2020	Few-shot prompting — управление LLM через примеры в промпте
2020	AutoPrompt — первый метод градиентной оптимизации промптов
2022	Chain-of-Thought — цепочки рассуждений в few-shot примерах
2022	Least-to-Most Prompting — декомпозиция задач от простого к сложному
2022	Self-Consistency и ReAct
2023	Tree of Thoughts, DSPy, OPRO
2024	EPiC — эволюционная оптимизация промптов для кода
2025	Promptware Engineering — SE-практики для промптов
2026	Midolo et al. — первые доказательные рекомендации по PE для генерации кода

Таблица 1 - хронологическая шкала

Ключевые техники

Few-shot prompting (Brown et al., 2020) [1] заложил основу: модели предъявляются примеры «вопрос–ответ» в тексте запроса, что позволяет управлять поведением LLM без дообучения. Chain-of-Thought (Wei et al., 2022) [2] расширил принцип: добавление промежуточных шагов рассуждения повышает точность на GSM8K с 18% до 57% (PaLM 540B); модели 7–13B с instruction tuning воспроизводят CoT-эффекты устойчиво [13]. Self-Consistency (Wang et al., 2022) [3] добавил ансамблирование: N цепочек генерируются независимо, ответ определяется голосованием (GSM8K +17,9%). Tree of Thoughts (Yao et al., 2023) [5] заменил линейные цепочки деревом с поиском BFS/DFS: CoT + GPT-4 на Game of 24 — 4%, ToT + GPT-4 — 74%. OPRO

(Google DeepMind, 2023) использует сам LLM как оптимизатор промптов, превосходя ручных авторов на GSM8K и Big-Bench Hard. DSPy (Stanford NLP) [25] трактует промпты как декларативные модули и оптимизирует их автоматически. Role prompting, несмотря на рекомендации Anthropic и OpenAI [28, 29], не улучшает точность для code/math задач (Zheng et al., EMNLP 2024 [6], 162 роли, 4 семейства LLM), оставаясь эффективным лишь для форматирования ответа.

Эмпирическое влияние на качество генерации кода

Midolo et al. [14] на бенчмарках HumanEval+, MBPP+ и BigCodeBench установили: наиболее эффективный паттерн — добавление алгоритмических деталей (57% случаев успешной оптимизации), уточнение формата ввода/вывода получило наивысшую оценку разработчиков (88%), автоматическая оптимизация через EPiC успешна в 61–75% случаев [16]. В рандомизированном эксперименте Peng et al. (95 разработчиков) группа с GitHub Copilot выполнила задачу за 71 мин против 161 мин в контроле (ускорение 55,8%, p=0,0017) [12]. Bruni et al. показали, что security-focused prefix снижает уязвимости до 56% для GPT-4o, итеративный prompting устраняет 41,9–68,7% ранее сгенерированных дефектов [15]. SWE-agent [17] набирает +10,7 п.п. на SWE-bench исключительно за счёт дизайна Agent-Computer Interface, что ставит под вопрос приоритет PE перед качеством инструментального окружения. Sclar et al. (NeurIPS 2024) [11] фиксируют разброс точности до 76 п.п. на LLaMA-2-13B при минорных форматных изменениях промпта при неизменном смысловом содержании.

Индустриальные практики и инструменты

По мере усложнения LLM-приложений появились специализированные инструменты. В Табл. 2 представлены наиболее распространённые решения, используемые в современной практике prompt engineering.

Инструмент	**Особенность**
Microsoft PromptFlow	Визуальный редактор, интеграция с Azure AI Studio

LangSmith / LangChain	Prompt Hub, evaluation, tracing
Langfuse (open-source)	Observability + A/B-тесты, self-hosted
Braintrust	Staged dev→stage→prod, CI/CD интеграция
DSPy	Декларативные модули, автоматическая оптимизация

Таблица 1 – экосистема инструментов

Концепция «промпт как код» с версионированием, тестированием и staged deployment (dev → staging → production) стала консенсусной практикой среди Anthropic, OpenAI, Google и Microsoft. По данным Stack Overflow Developer Survey 2025 [27], 84% разработчиков используют AI-инструменты, 51% — ежедневно, однако позитивный sentiment снизился с 70%+ до 60%. Спрос на «Prompt Engineer» вырос на 135,8%, но опрос Microsoft (31 000 сотрудников) показал, что это вторая с конца роль по планируемому найму: на практике PE интегрируется в позиции AI/ML Engineer.

Ограничения и критика

Системная хрупкость PE задокументирована на нескольких уровнях. «Butterfly Effect» (arXiv 2024) [21] показал, что добавление одного пробела в конец промпта изменяет ответы в 500+ тест-кейсах из 11 000 на 11 различных LLM. Sclar et al. (NeurIPS 2024) [11] фиксируют разброс до 76 п.п. при сугубо форматных изменениях. PromptBridge [23] демонстрирует: при переносе промпта между моделями одного семейства (Llama-3.1-70B → 8B) точность падает на 50–70%; авторы резюмируют: «there is no such thing as prompt portability right now». Кризис воспроизводимости подтверждён на уровне ICLR 2024 [22]: 6 техник (включая CoT и EmotionPrompting), проверенных на 7 моделях и 5 бенчмарках, демонстрируют «general lack of statistically significant differences». Концепция «Promptware Engineering» [20] фиксирует структурные препятствия для признания PE дисциплиной: прежде всего нетестируемость недетерминированных систем и принципиальная неточность естественного языка как среды спецификации требований.

Практические рекомендации

Для генерации кода наиболее стабильный результат даёт детализация алгоритмических требований и явное указание формата ввода/вывода [14]. Смена базовой модели требует полного переаудита промптов [23]; для production-управления зрелы Langfuse и Braintrust с версионированием и A/B-тестированием. Security-focused prefix обязателен для кода, связанного с аутентификацией и валидацией: снижение уязвимостей до 56%, устранение до 68,7% дефектов итеративным prompting [15]. При наличии 50+ размеченных примеров переход к DSPy или EPiC даёт более воспроизводимые результаты, чем ручной PE [25] [16].

Заключение

За 2020–2026 годы prompt engineering оформился в практику с собственной таксономией, инструментальной экосистемой и консенсусными рекомендациями от ведущих AI-компаний. Вместе с тем заявить о полноценном дисциплинарном статусе не позволяет кризис воспроизводимости: большинство задокументированных эффектов не обобщаются за пределы исходной модели, а автоматические методы оптимизации (DSPy, EPiC, OPRO) в контролируемых условиях стабильно превосходят ручной PE. Роль инженера при этом смещается: не составитель промптов, а проектировщик пайплайнов и метрик оценки качества. Дальнейшие исследования целесообразны в направлении измеримых метрик переносимости промптов между моделями и формализации lifecycle management для promptware-систем.

Список литературы

1. Brown T., Mann B., Ryder N. et al. Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems (NeurIPS 2020), 2020. URL: https://arxiv.org/abs/2005.14165 (дата обращения: 05.06.2026).
2. Wei J., Wang X., Schuurmans D. et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. Advances in Neural Information Processing Systems (NeurIPS 2022), 2022. URL: https://arxiv.org/abs/2201.11903 (дата обращения: 05.06.2026).

3. Wang X., Wei J., Schuurmans D. et al. Self-Consistency Improves Chain of Thought Reasoning in Language Models. International Conference on Learning Representations (ICLR 2023), 2023. URL: https://arxiv.org/abs/2203.11171 (дата обращения: 05.06.2026).
4. Yao S., Zhao J., Yu D. et al. ReAct: Synergizing Reasoning and Acting in Language Models. International Conference on Learning Representations (ICLR 2023), 2023. URL: https://arxiv.org/abs/2210.03629 (дата обращения: 05.06.2026).
5. Yao S., Yu D., Zhao J. et al. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. Advances in Neural Information Processing Systems (NeurIPS 2023), 2023. URL: https://arxiv.org/abs/2305.10601 (дата обращения: 05.06.2026).
6. Zheng C., Liu Z., Xie E. et al. Is "A Helpful Assistant" the Best Role for Large Language Models? Proceedings of EMNLP 2024, 2024. URL: https://arxiv.org/abs/2311.10054 (дата обращения: 05.06.2026).
7. Schulhoff S., Ilie M., Balepur N. et al. The Prompt Report: A Systematic Survey of Prompting Techniques. arXiv:2406.06608, 2024. URL: https://arxiv.org/abs/2406.06608 (дата обращения: 05.06.2026).
8. Sahoo P., Singh A.K., Saha S. et al. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. arXiv:2402.07927, 2024. URL: https://arxiv.org/abs/2402.07927 (дата обращения: 05.06.2026).
9. Vatsal S., Dubey H. A Survey of Prompt Engineering Methods in Large Language Models for Different NLP Tasks. arXiv:2407.12994, 2024. URL: https://arxiv.org/abs/2407.12994 (дата обращения: 05.06.2026).
10. Chu T., Zeng Z., Wang J. et al. A Survey of Chain of Thought Reasoning: Advances, Frontiers and Future. Proceedings of ACL 2024, 2024. URL: https://arxiv.org/abs/2309.15402 (дата обращения: 05.06.2026).
11. Sclar M., Choi Y., Tsvetkov Y., Suhr A. Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design. Advances in Neural

Information Processing Systems (NeurIPS 2024), 2024. URL: https://arxiv.org/abs/2310.11324 (дата обращения: 05.06.2026).

12. Peng S., Kalliamvakou E., Croft P., Auer M. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. arXiv:2302.06590, 2023. URL: https://arxiv.org/abs/2302.06590 (дата обращения: 05.06.2026).
13. Yang C., Yang J., Gao S. et al. Chain-of-Thought in Neural Code Generation: From and for Lightweight Language Models. IEEE Transactions on Software Engineering, 2024. URL: https://arxiv.org/abs/2312.05562 (дата обращения: 05.06.2026).
14. Midolo G., Falkowski M., Gorla A. Guidelines to Prompt Large Language Models for Code Generation. arXiv:2601.13118, 2026. URL: https://arxiv.org/html/2601.13118 (дата обращения: 05.06.2026).
15. Bruni R., Goretti F., Merlo E. Benchmarking Prompt Engineering Techniques for Secure Code Generation. IEEE/ACM FORGE, 2025. URL: https://arxiv.org/abs/2502.06039 (дата обращения: 05.06.2026).
16. Taherkhani A., Fard F.H. EPiC: Cost-effective Search-based Prompt Engineering of LLMs for Code Generation. arXiv:2408.11198, 2024. URL: https://arxiv.org/abs/2408.11198 (дата обращения: 05.06.2026).
17. Yang J., Jimenez C.E., Wettig A. et al. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering. International Conference on Learning Representations (ICLR 2025), 2025. URL: https://arxiv.org/abs/2405.15793 (дата обращения: 05.06.2026).
18. Errica F., Siracusano G., Sanvito D., Bifulco R. What Did I Do Wrong? Quantifying LLMs' Sensitivity and Consistency to Prompt Engineering. Proceedings of NAACL 2025, 2025. URL: https://arxiv.org/abs/2406.12334 (дата обращения: 05.06.2026).
19. Geng B., Dohan D., Maheshwari P. et al. Generating Structured Outputs from Language Models: A Survey. arXiv:2501.10868, 2025. URL: https://arxiv.org/html/2501.10868v1 (дата обращения: 05.06.2026).

20. Eghbali A., Pradel M. Promptware Engineering. arXiv:2503.02400, 2025. URL: https://arxiv.org/html/2503.02400v2 (дата обращения: 05.06.2026).
21. Mizrahi M., Kaplan G., Malkin D. et al. State of What Art? A Call for Multi-Prompt LLM Evaluation. arXiv:2401.03729, 2024. URL: https://arxiv.org/html/2401.03729v2 (дата обращения: 05.06.2026).
22. Loya S., Bhatt P., Chatterjee M. et al. Prompt Engineering a Prompt Engineer. International Conference on Learning Representations (ICLR 2024), 2024. URL: https://openreview.net/forum?id=bgjR5bM44u (дата обращения: 05.06.2026).
23. Guo Q., Zhang R., Wei F. et al. PromptBridge: Cross-Model Prompt Adaptation via Contrastive Learning. arXiv:2512.01420, 2025. URL: https://arxiv.org/html/2512.01420v1 (дата обращения: 05.06.2026).
24. Battle L., Gollapudi A. AI Prompt Engineering Is Dead. IEEE Spectrum, 2024. URL: https://spectrum.ieee.org/prompt-engineering-is-dead (дата обращения: 05.06.2026).
25. Khattab O., Singhvi A., Maheshwari P. et al. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. International Conference on Learning Representations (ICLR 2024), 2024. URL: https://dspy.ai/ (дата обращения: 05.06.2026).
26. Zhao R., Li X., Chen W. et al. Towards Large Language Models Robustness to Changes in Prompt Format Styles. arXiv:2504.06969, 2025. URL: https://arxiv.org/html/2504.06969v1 (дата обращения: 05.06.2026).
27. Stack Overflow. Developer Survey 2025: AI Section. 2025. URL: https://survey.stackoverflow.co/2025/ai (дата обращения: 05.06.2026).
28. Anthropic. Prompt Engineering Overview. 2025. URL: https://platform.claude.com/docs/en/build-with-claude/prompt-engineering/overview (дата обращения: 05.06.2026).
29. OpenAI. Prompt Engineering Guide. 2024. URL: https://developers.openai.com/api/docs/guides/prompt-engineering (дата обращения: 05.06.2026).

30. Google Cloud. What is Prompt Engineering. 2024. URL: https://cloud.google.com/discover/what-is-prompt-engineering (дата обращения: 05.06.2026).
31. Braintrust. Top 5 Prompt Versioning Tools in 2025. 2025. URL: https://www.braintrust.dev/articles/best-prompt-versioning-tools-2025 (дата обращения: 05.06.2026).
32. ZenML. Best Prompt Management Tools for 2025. 2025. URL: https://www.zenml.io/blog/best-prompt-management-tools (дата обращения: 05.06.2026).