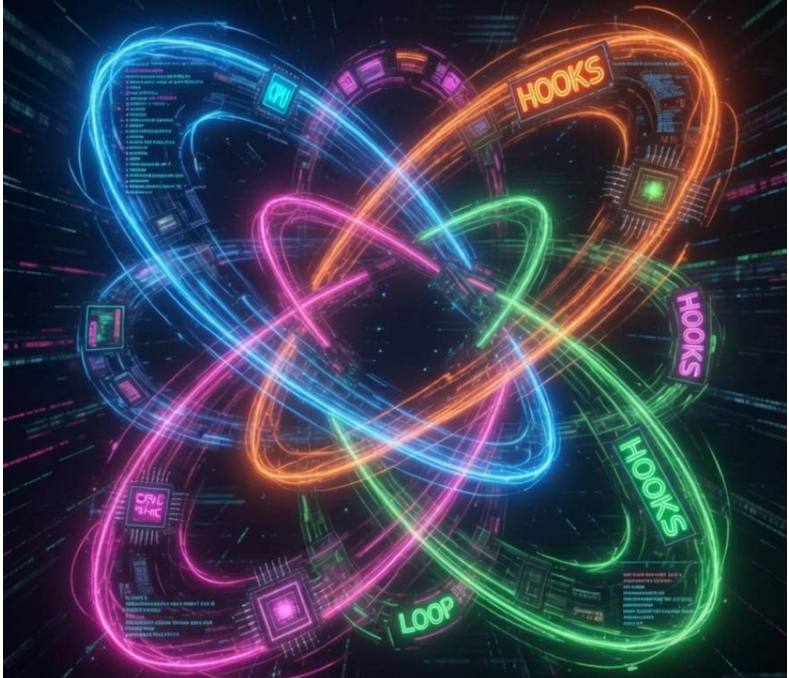
ОСНОВЫ LOOP ENGINEERING

Архитектура, циклы и хуки автономных ИИ-агентов



Павел Новопольцев

Павел Новопольцев

Основы Loop Engineering. Архитектура, циклы и хуки автономных ИИ-агентов

https://litres.ru/74230148
SelfPub; 2026

Аннотация

Основы Loop Engineering -это первое подробное руководство по петлевой инженерии, новому стандарту проектирования автономных ИИ-агентов. Эпоха ручного написания промптов уходит в прошлое. Пора строить системы, которые сами ставят задачи, пишут код, проверяют результаты и исправляют ошибки без непрерывного контроля со стороны человека.

В книге детально разбирается архитектура замкнутых циклов: изолированные рабочие области, управление долгоживущей памятью, навыки (skills) и коннекторы. Вы научитесь настраивать детерминированные хуки безопасности, жестко контролировать расход токенов и внедрять независимых агентов-верификаторов.

Особое внимание уделено скрытым рискам автоматизации: как избежать «долга понимания», предотвратить бесконечное зацикливание агентов и не потерять контроль над кодовой базой.

Внутри собраны готовые шаблоны, конфигурации и чек-листы для немедленного применения.

Содержание

Павел Новопольцев
Основы Loop Engineering. Архитектура, циклы и хуки автономных ИИ-агентов

Введение.

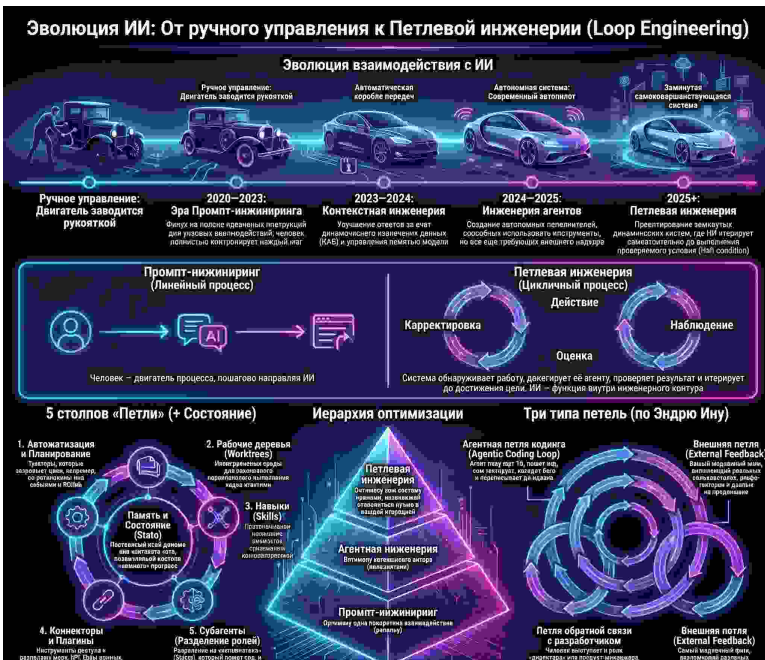
Последний промпт

Представьте себе обычный рабочий день разработчика или энтузиаста нейросетей пару лет назад. Вы открываете чат с искусственным интеллектом и пишете длинный, выверенный до запятой запрос. Нейросеть выдаёт ответ. Вы читаете, находите ошибку, пишете новый запрос: «Нет, исправь здесь и добавь вот это». Нейросеть снова отвечает.

Этот процесс очень похож на игру в настольный теннис. Вы подаёте, ИИ отбивает. Вы двигатель этого процесса. Стоит вам отойти заварить кофе, как работа замирает. Вы становитесь заложником микроменеджмента, эдаким «курьером», который переносит контекст от одного шага задачи к другому. Долгое время этот подход — промпт-инжиниринг — считался вершиной мастерства. Но в середине 2025 года всё изменилось.

В течение одной ключевой недели ведущие умы индустрии озвучили мысль, которая перевернула подход к работе с ИИ. Питер Штайнбергер, создатель OpenClaw, написал: «Вам больше не следует писать промпты для кодинг-агентов. Вы должны проектировать циклы, которые будут промптить агентов за вас». Ему вторил Борис Черный, руководитель разработки Claude Code в Anthropic: «Я больше не пишу промпты напрямую. У меня запущены циклы, которые сами ставят задачи и решают, что делать дальше. Моя работа писать циклы».

Так родилась концепция Loop Engineering — петлевая инженерия.



Эволюция происходила стремительно. Сначала мы оптимизировали единичные инструкции Prompt Engineering. Затем мы поняли, что моделям нужна память и базы знаний, и начали оптимизировать контекст Context Engineering.

Позже появились автономные агенты, способные использовать инструменты — Agent Engineering. Но агенты, какими

бы умными они ни были, оставались исполнителями одного поручения. Если агент ошибался или выдавал неполный код, в процесс снова должен был вмешиваться человек и направлять алгоритм.

Петлевая инженерия предлагает радикально иной подход. Вы больше не водитель, который крутит руль на каждом повороте.

Вы инженер, который строит автопилот и прокладывает маршрут.

Ваша задача создать замкнутую систему, которая получает цель, совершает действие, собирает обратную связь из среды, исправляет свои же ошибки и повторяет этот цикл до тех пор, пока задача не будет объективно выполнена.

Зачем тратить часы на проверку кода, тестирование и написание новых инструкций, если можно поручить агенту не только написать логику, но и самостоятельно запустить тесты, прочитать ошибки компилятора, сделать выводы и переписать решение? В этом суть нового мета-навыка: мы перестаём управлять агентами в ручном режиме и начинаем управлять системами обратной связи, в которых эти агенты функционируют.

Эпоха ручного управления нейросетями официально подошла к концу. Впереди нас ждёт мир автономных рабочих процессов, где человек устанавливает правила игры, а машины самостоятельно находят путь к победе. Но как именно произошёл этот переход от текстовых запросов к проектиро-

ванию автономных архитектур и почему традиционные методы перестали работать при масштабировании задач? Эти фундаментальные сдвиги мы подробно разберём в первой главе.


Loop Engineering
Prompt Engineering
Context Engineering
Harness Engineering
Loop Engineering
Automations
Worktrees
Skills
Connectors (MCP
Sub-agents (Maker/Checker split)
Open Loops
Closed Loops
Stop conditions
Lifecycle Hooks
Daily Triage
PR Babysitter
CI Sweeper
Multi-agent SDLC
Thrashing
Overfitting
Context Drift
Comprehension Debt

Глава 1. Эволюция: от промпта к петле

Конец ознакомительного фрагмента.