🤖 Оптимизация архитектуры AI-агентов: от хаоса к 92%

🛠️ Архитектура агентов: от хаоса к эффективности

Ключевые тезисы:

- Сложные агенты со временем деградируют из-за накопления требований без модернизации архитектуры
- Ключ к улучшению — правильное использование примитивов: инструментов, навыков (skills) и под-агентов
- Эффективность измеряется через эвалюации (evals) и методичное «восхождение» (hill climbing) к улучшению показателей
- Claude Managed Agents (CMA) избавляет от инфраструктурных сложностей, позволяя фокусироваться на дизайне агента

🎯 Проблема: «Распухший» агент

Агент **Stock Pilot** (управление запасами) столкнулся с типичными проблемами роста:

- **Длинный системный промпт** (~400 строк) из-за постоянного добавления бизнес-требований
- **12 инструментов**, 3 из которых — обёртки над изолированными под-агентами
- **Падение эффективности (evals)**: успешность упала до 62-83%, что критично для бизнеса

Примеры провальных эвалюаций:

- **F1 (ежедневная проверка низкого запаса)**: Агент достигает цели, но неэффективным, извилистым путём.

- **F2 (процесс заказа по акции)**: Сбой коммуникации между оркестратором и под-агентом.
- **R8 (прогнозирование в промо-месяц)**: Конфликт политик в разных частях системного промпта ведёт к ошибкам в расчётах (например, использование множителя 1.35 вместо 3.1x).

🧪 Стратегия улучшения: Hill Climbing на эвалюациях

Методология «восхождения»:

1. **Запустить эвалюации** и получить базовый уровень.
2. **Проанализировать причины провалов** (можно с помощью Claude).
3. **Внести изменения в архитектуру** агента.
4. **Перезапустить эвалюации** и измерить улучшение.
5. Повторять цикл, пока не достигнете целевых показателей.

🔧 Три столпа модернизации архитектуры

📝 1. Навыки (Skills) вместо длинного промпта

Навык — это упакованная, композируемая информация, которую Клод может подтягивать в контекст только тогда, когда она нужна для конкретной задачи.

Проблема: Весь бизнес-процесс (политики, процедуры) был вшит в системный промпт, «загрязняя» контекстное окно.
Решение: Вынести специфические инструкции (например, по прогнозированию) в **Skills**. Оставить в системном промпте только самую общую, всегда необходимую информацию.
Результат: Системный промпт сократился с ~400 до ~50 строк.

> 💡 *Правило: Системный промпт — только для информации, которая нужна агенту всегда. Skills — для информации, которая нужна иногда.*

🛠️ 2. Примитивные инструменты вместо кастомных

Философия: Давайте агенту те же примитивы, что есть у человека за компьютером.

Рекомендуемая последовательность:

1. **Начните с примитивов Claude Code:** выполнение кода (bash), файловая система, веб-поиск, список задач.
2. **Удаляйте лишнее:** Если агенту не нужен веб-поиск — отключите этот инструмент.
3. **Добавляйте кастомные инструменты только по необходимости.**

Пример из кейса: Вместо создания отдельных инструментов для анализа каждого CSV-файла, агенту дали доступ к **bash**. Теперь он может написать скрипт на Python для анализа данных, что:

- ✅ **Сильно сократило использование токенов** (с >200K до минимума)
- ✅ **Снизило стоимость**
- ✅ **Увеличило гибкость**

MCP (Model Context Protocol): Подключайте только когда есть общий набор инструментов для **множества** агентов или клиентов. Часто выполнение кода (для вызова API) — более гибкая альтернатива.

🤖 3. Осознанное использование под-агентов (Sub-agents)

Когда они действительно нужны:

1. **Параллелизация:** Когда нужно «бросить много Клода» на одну сложную задачу (глубокий research, анализ кода).
2. **«Свежий взгляд»:** Когда нужно разделить процессы создания и проверки (например, один агент пишет код, другой — ревьюит).

Проблемы под-агентов: Сложная коммуникация с оркестратором, проблемы с логированием и observability.
Решение в CMA: Использовать нативную возможность **Callable Agents** для управляемых под-агентов с централизованной observability.

В кейсе Stock Pilot оставили только одного под-агента для **прогнозирования**, чтобы изолировать этот процесс от основного контекста.

🚀 Итоговая архитектура и результаты

Было:

- Оркестратор + промпт на 400 строк
- 12 инструментов (3 — под-агенты)
- Успешность эвалюаций: ~62-83%

Стало (после рефакторинга):

- Оркестратор на **Claude Managed Agents** (без забот об инфраструктуре)
- **3 примитивных инструмента:** bash, read, write
- Системный промпт: **15 строк**
- Бизнес-логика вынесена в **Skills**
- **Успешность эвалюаций: ~92%**

Ключевые улучшения:

- 📉 **Снижение использования токенов и стоимости**
- ⚡ **Рост производительности**
- 🎯 **Упрощение поддержки и архитектуры**

🎯 Ключевые выводы

1. **Начинайте с простого:** Один цикл агента + базовые примитивы (код, файлы, поиск).
2. **Используйте прогрессивное раскрытие через Skills:** Не засоряйте системный промпт. Давайте информацию по запросу.
3. **Пишите и постоянно обновляйте эвалюации:** Они — ваш компас для «восхождения» (hill climbing) к лучшим результатам.

4. **Используйте Claude Managed Agents:** Чтобы сосредоточиться на дизайне агента, а не на инфраструктуре, безопасности и масштабировании.
