



CyberAgentReview

PROTECT YOUR AGENT

SML Security

Разработка с агентами: от постановки задачи до деплоя

Конвейер из шести стадий на реальном проекте:
агенты делают работу, люди принимают решения



Павел Гагарин

Эволюция разработки с LLM

Инженерия запроса prompt engineering

.....

- как сформулировать один запрос
- роль, примеры, формат ответа
- один вопрос – один ответ

Инженерия контекста context engineering

.....

- не как спросить, а что показать модели
- нужные файлы, история, инструменты в окне
- модель видит ровно то, что нужно

Инженерия цикла loop engineering

.....

- не один ответ, а управляемый процесс
- стадии, проверки, границы для людей
- агент работает в цикле и правит себя сам



Добавьте рекламный блок в интерфейс



Команда без процесса

- дни уточнений в личке
- требования всплывают на ревью
- решения - в чатах, испаряются



Агент без процесса

- уверенно реализует **не то**
- додумывает бизнес-решения молча
- бюджет - токены и время впустую
- ложная безопасность
- ухудшение архитектуры
- неконтролируемый рост

Ответ: конвейер стадий с артефактами в git

Шесть стадий, шесть артефактов

/01 Приём задачи зрелость задачи <code>intake.md</code> автор утверждает	/02 План задачи + критерии <code>plan.md</code> тимлид утверждает	/03 Выполнение цикл сабагентов <code>progress.md</code> стоп по блокеру	/04 Авто-ревью 5 агентов параллельно <code>review/.md</code> решает по значимым	/05 Деплой + приёмка dev-стенд, чеклисты <code>acceptance.md</code> тимлид · автор · дизайнер	/06 Рефлексия уроки → правки <code>retro.md</code> команда утверждает
--	--	--	--	---	---

■ агенты - работа внутри стадии ■ люди - принятие важных решений

Каждый артефакт - файл в git, история решений навсегда



Задача пришла как GitHub issue

Автор: Руководитель проекта · Приоритет: средний

Добавить в админ панель раздел «Рекламные кампании», каждая кампания содержит:

- даты начала-конца,
- richtext содержимого,
- имя рекламодателя.

Клиентам выводить блок над списком чатов

- Задача живёт в трекере - обсуждение стадии 1 комментариями к issue; решения из треда лягут в git рядом с задачей

Один абзац



Зрелость задачи: агент читает код, а не загадет

Текущее положение дел

.....

- каждое слово тикета сверяется с кодом
- «админ панель» - уже есть? частично? нет?
- пользовательские сценарии (docs/us) - черновики на утверждение автору

Критерии готовности

.....

- цель · сценарии · критерии приёмки
- дизайн: нужен? есть? хватит дизайн-системы?
- нефункциональные требования: безопасность, лимиты, внешние API
- границы: даты, пустые состояния, конфликты

Вердикт:

готово · готово с допущениями · нужен ответ · заблокировано



Пробелы делятся на два сорта

01

Бизнес →
вопросы автору

- меняет то, что увидит пользователь
- с вариантами а) б) в)
- и рекомендацией - автору остаётся выбрать букву
- уходят комментарием в `issue` - автор отвечает там же

«Как выбрать кампанию - показываем самую свежую?»

VS

02

Техника →
предположения

- агент решает сам, обоснование - одной строкой
- раздел «Предположения» в `intake.md`
- автору читать не обязательно

«Хранение: `content_src` +
санитизированный `content_html`»



План: чекбоксы – источник истины

```
docs/tasks/ad-campaigns/plan.md
```

```
### Задача 3: Конвертация и санитизация  
богатого текста
```

```
**Files:** Create: internal/api/richtext.go
```

```
- [ ] конвертер md-подмножества → HTML  
- [ ] whitelist b,i,a,br; href - http/https  
- [ ] тесты: <script>, onclick - вычищаются
```

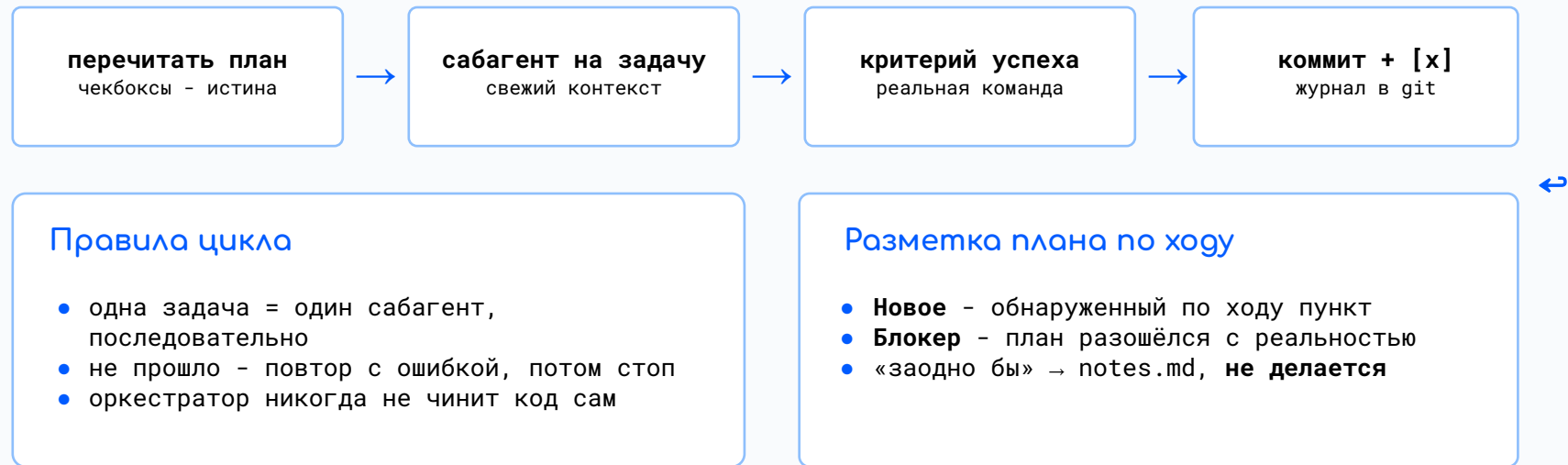
Критерий успеха: go test -run
TestRichtext - зелёный, XSS-кейсы

- единый шаблон для каждой задачи - предсказуемость
- критерий успеха = сценарий + результат /«Код написан» - не критерий/
- «Рассмотренные подходы» - почему так, для будущего
- «Вне скоупа» - контракт для scope-guard
- оценка сложности + бюджет токенов по задачам /сверим с фактом на рефлексии/

Переход:

план утверждает тимлид и его правка может изменить архитектуру

Выполнение: оркестратор не пишет код



Ответвления: когда агент зовёт человека

progress.md

```
[decision] + campaign_views «на будущее»  
[decision] whitelist: b,i,a,br - из крит. №4  
[stop] нет паттерна многострочного ввода;  
textarea в .modal / отдельный экран?  
→ разработчик: textarea, паттерн расширяем  
[deviation] баннер в app.js, фронт односкриптовый  
[usage] task 6: ~120 тыс. (план: ~50)  
[usage] task 7: ~185 тыс. (план: ~70)  
completed → отчёт: решения, принятые  
автономно - 6 пунктов, разработчику
```



Мелкая развилка → решает сам
[decision] + причина, в git



План разошёлся → СТОП
[stop]: вопрос человеку с вариантами



Финал: отчёт автономных решений
ничего не происходит молча

Человека дёргаем по делу, а каждое авто-решение он увидит списком

Пять взглядов на изменения

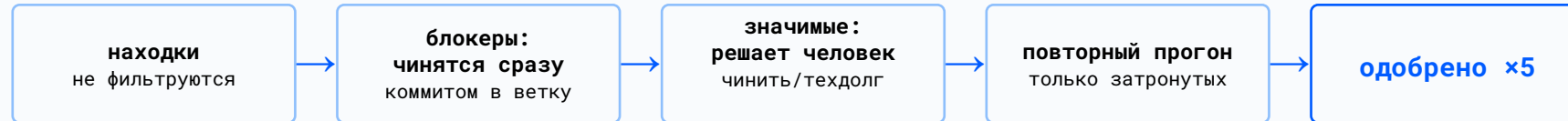
security - нашёл блокер: stored XSS - санитизация в сервисе была только на записи, не на отдаче

ux - жёстко зашитый #fff ломает тёмную тему; крестик меньше зоны тапа; нет бейджа «Реклама»

scope-guard - поймал campaign_views: автор сказал «не сейчас», исполнитель добавил «на будущее»

arch - хендлер сервиса обошёл свой интерфейс Store; индекс миграции не идемпотентен

docs - межсервисная граница ядро↔campaigns жила только в коде: нет перечня интерфейсов (маршруты, X-User-ID, зоны ответственности) → фикс: схема в доках



- Score-guard поймал то, что разработчик пропустил.
- Слоёв защиты - несколько.



Приёмка людьми: каждому – свой чеклист

Тимлид

Принял

- логи
- healthcheck campaigns
- запуск
- отчёт автономных решений

Автор задачи

Уточнение описания

- его же критерии приёмки
- без жаргона:
«сделай X – увидишь Y»

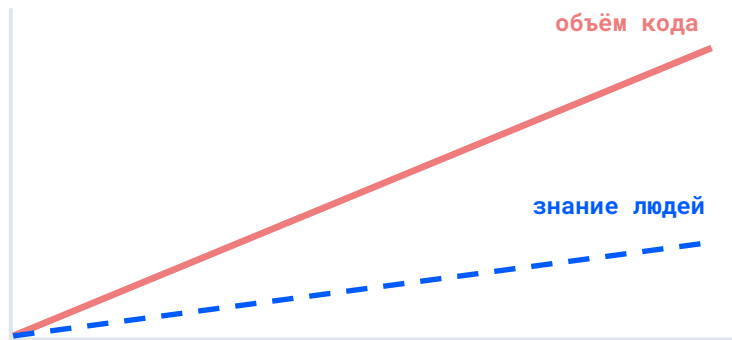
Дизайнер

«рамка в ререл неразличима»

- 4 темы
- мобильный вьюпорт
- состояния
- зоны тапа

- замечания пишутся
в **acceptance.md** –
не испаряются в чате
- дефект против критериев
→ фикс; новое пожелание
→ кандидат в тикет
- люди нашли ровно то,
что автоматика
пропустила

Код растёт быстрее понимания



Через полгода никто не может объяснить, почему система устроена так

След решений - docs/tasks/<задача>/: вопросы, ответы, отвергнутые подходы, ревью, уроки. Git - не только код

Доки после каждой задачи - CLAUDE.md и docs/us обновляются задачей плана; приём и docs-reviewer сверяют доки с кодом

Важные решения от людей - люди видят intake, план и отчёт решений

Никаких скрытых решений - каждое автономное решение агента будет в отчёте с причиной

Рефлексия: процесс правит сам себя



Каждое уточнение проходит через ответственного



Основные принципы

- **Агенты – конвейер, люди – границы.** Автономия внутри стадий, решения при переходах
- **Вопросы автору – только бизнес,** с вариантами и рекомендацией. Техника – предположениями
- **Критерий успеха – команда, а не «код написан».** Чекбоксы плана – контроль выполнения
- **Ничего не происходит молча:** [decision] / [stop] / отчёт автономных решений
- **Замечания людей – высший приоритет** и превращаются в системные правки процесса
- **Знание о системе – артефакт процесса:** след решений в git + поддержка документации

Стадии – это скиллы в git

их рецензируют, откатывают и улучшают, как код

Cyber Agent Review



Централизованный контроль, мониторинг
и защита AI-агентов в реальном времени



Отображает и логирует на едином дашборде
действия каждого агента



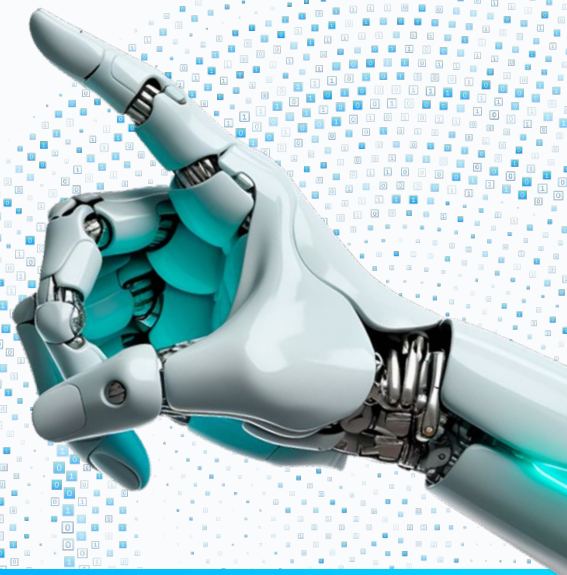
Обнаруживает и блокирует утечки данных, инъекции
промптов и другие аномалии в поведении агентов



Агент физически не может вывести конфиденциальную
информацию и использовать инструмент в обход политик



Администратор получает полный контроль
над использованием LLM в организации



Выгодно

- CISO: AI- агенты без рисков утечек и полная отчетность для аудита
- CTO: Возможность быстро внедрять новые AI-фичи без вето от отдела безопасности
- SOC: Готовые инструменты для расследования ИИ-инцидентов