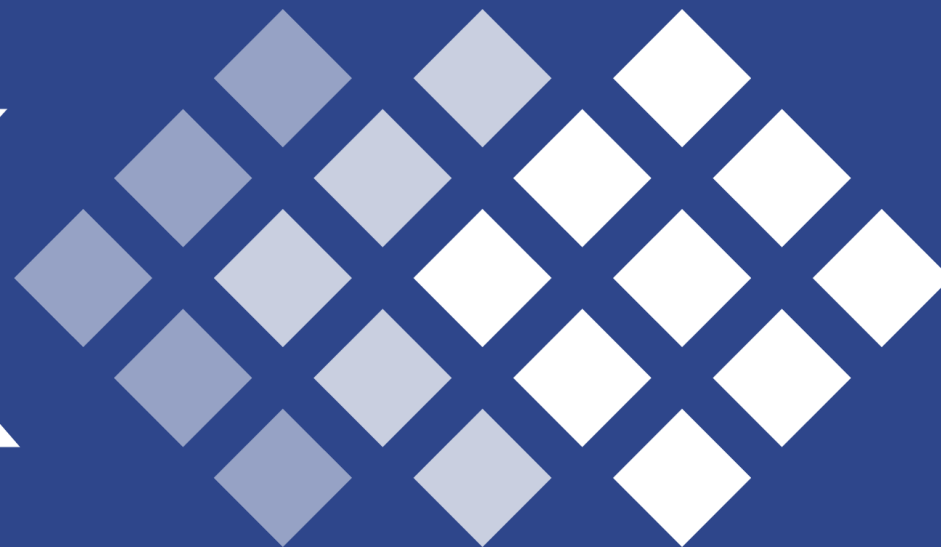


РСК



РСК Экосистема: на пути к автономному ЦОД

Константин Пономарев · Руководитель отдела разработки ПО

Группа компаний РСК

Суперкомпьютерные дни в России

2026

Агенту в ЦОД нужна опора, а не догадка

Агент объясняет систему, управляет ею, замечает изменения сам и отдаёт разобранный инцидент с доказательствами.

Для этого нужна опора — и у каждого способа своя цена.

ЧЕГО ТРЕБУЕТ ЦОД	ЧЕМ ЭТО ЗАКРЫВАЮТ	ЧЕМ ПЛАТИМ
Знание домена	больше модель	данные уходят наружу
Полнота описания	больше контекста	окно растёт, точность падает
Согласованность правил	описание отдельным слоем	правила расходятся с кодом
Связность систем	инструмент под каждую систему	интеграции растут с системами
Проверяемость	выводы делает сама модель	результат нечем проверить

Вопрос не в модели. Вопрос в среде.

Обычно: домен знает модель.

Здесь: домен несёт схема.

Среда — это схема, проверка и права, которые агент получает вместе с доступом. Схема GraphQL описывает предметную область и сама себя объясняет: сущности, связи и их смысл. Через ту же схему идёт работа — запрос, права, события. Агент читает систему так же, как читал бы её инженер.

Онтология и интерфейс — один артефакт

Онтология здесь — сама схема GraphQL, а не отдельный документ.

ОДНА СХЕМА

Типы сущностей

Связи между ними

Описания полей

Проверка запроса

Права на ветвях

События

Агент читает вместо догадки, ошибка видна до исполнения.

Среда состоит из трёх уровней ответственности

№	УРОВЕНЬ	ЗА ЧТО ОТВЕЧАЕТ	ЧТО ЭТО ЗНАЧИТ
01	Схема	Сущности, связи и описания	агент узнаёт, что есть в системе
02	Валидатор	Соответствие запроса схеме	некорректный запрос не дойдёт до исполнения
03	Бизнес-слой	Правила, права и тесты	допустимость операции решает код

Пример: «перевести узел в обслуживание».

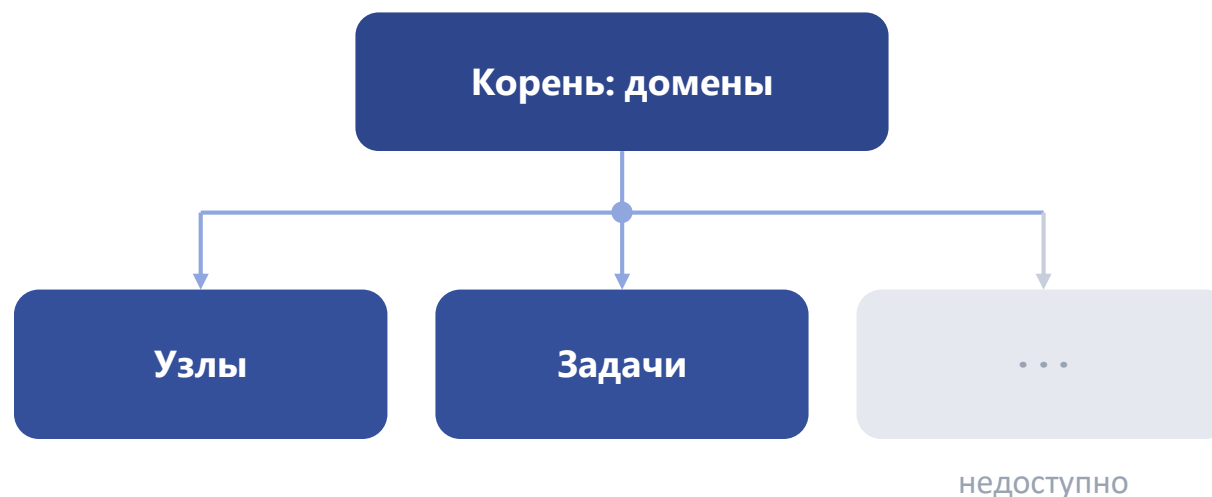
Схема показывает узел, состояние и связь — запрос построен на реальной модели. Валидатор сверяет его со схемой: ошибка возвращается агенту, система не тронута. Дальше бизнес-слой решает, кому и когда это можно.

Права определяют видимую часть графа

ЧТО ЭТО ЗНАЧИТ

Права живут в API, а не в промпте.

Права фильтруют саму интроспекцию — чтение схемы агентом. Недоступных ветвей нет ни в ответах, ни в описании: агент о них не знает.
Операции записи видны так же выборочно.



Третий домен для агента просто не существует: его нет ни в схеме, ни в запросах, ни в правах.

Агент и человек работают в одном состоянии

Общее состояние вместо переписки в чате.



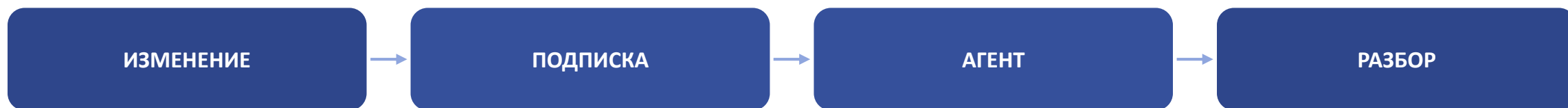
Чат вместо этого пересказывает историю заново.

Этого достаточно локальной модели

Знание и проверка в среде — тяжёлая модель не нужна.

ЧТО НУЖНО	ЧЕРЕЗ ВНЕШНИЙ КОНТУР	В СВОЁМ КОНТУРЕ
Данные не уходят наружу	не выполняется	выполняется
Оплата за каждый запрос	платит за каждый	своё железо
Размер модели	сотни миллиардов параметров	десятки миллиардов
Окно контекста	большое: домена нет в модели	малое: домен несёт схема
Доступность	зависит от чужого контура	в своём контуре
Деление задачи	один общий агент	отдельный агент на домен

Агент просыпается на событие, а не по таймеру

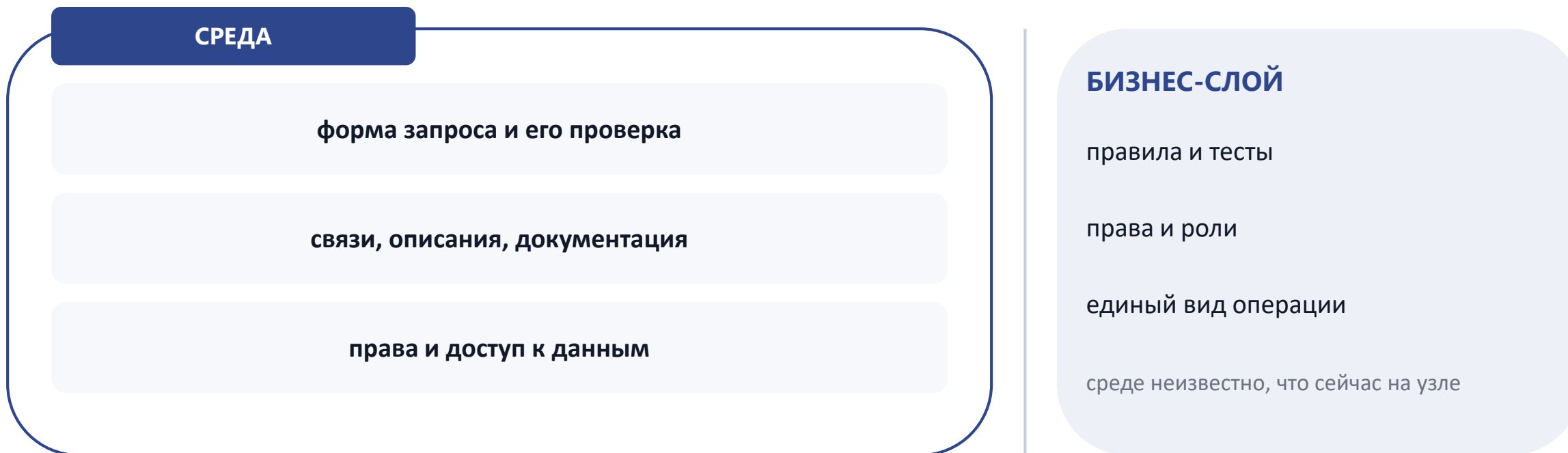


Подписка — событие из самой модели данных. Расписание и вебхук настраивают снаружи, под каждый случай отдельно.

- **Узел ушёл в ошибку** разбирает причину и приносит связный вывод, а не строку из лога
- **Задача завершилась** сверяет результат с ожиданием и пишет сводку
- **Том заполняется** предлагает перенос или расширение до отказа
- **Конфигурация разошлась** находит отличие и показывает, что именно изменилось

Границы: что остаётся бизнес-слою

Среда проверяет, можно ли так спросить. Код решает, можно ли это сделать сейчас.



Пример: «перевести узел в обслуживание» — форма верна, права есть, но на узле идут задачи.

ИТОГ

Автономность начинается со схемы, общей для агента и человека.

Одна модель объясняет систему обоим: агент по ней работает, человек по ней её видит. Академия формулирует это только сейчас, индустрия покупает дорогими моделями и чужим контуром — а здесь это уже работает.

знание

проверка

права

события

И тогда изменение обрабатывается в момент, когда произошло, а не когда до него дойдут руки.

Литература

Постановка задачи

- F. Coyle. Why Agentic Systems Need Ontologies. AI Engineer 2026
- Ontology-Constrained Neural Reasoning... arXiv:2604.00555

Эмпирика

- Symbolic Separation: Grounding Deep Agents... arXiv:2609.17107
- Knowledge-Graph Grounding Helps LLMs... arXiv:2606.22419

Механика API

- GraphQL Foundation. Authorization. graphql.org/learn/authorization
- Model Context Protocol. Subscriptions — черновик спецификации
- Anthropic. Code execution with MCP (2025); Advanced tool use (2025)

Спасибо!

rscgroup.ru · hq@rsc-tech.ru

Константин Пономарев · Руководитель отдела разработки ПО

Группа компаний РСК · Суперкомпьютерные дни в России, 2026