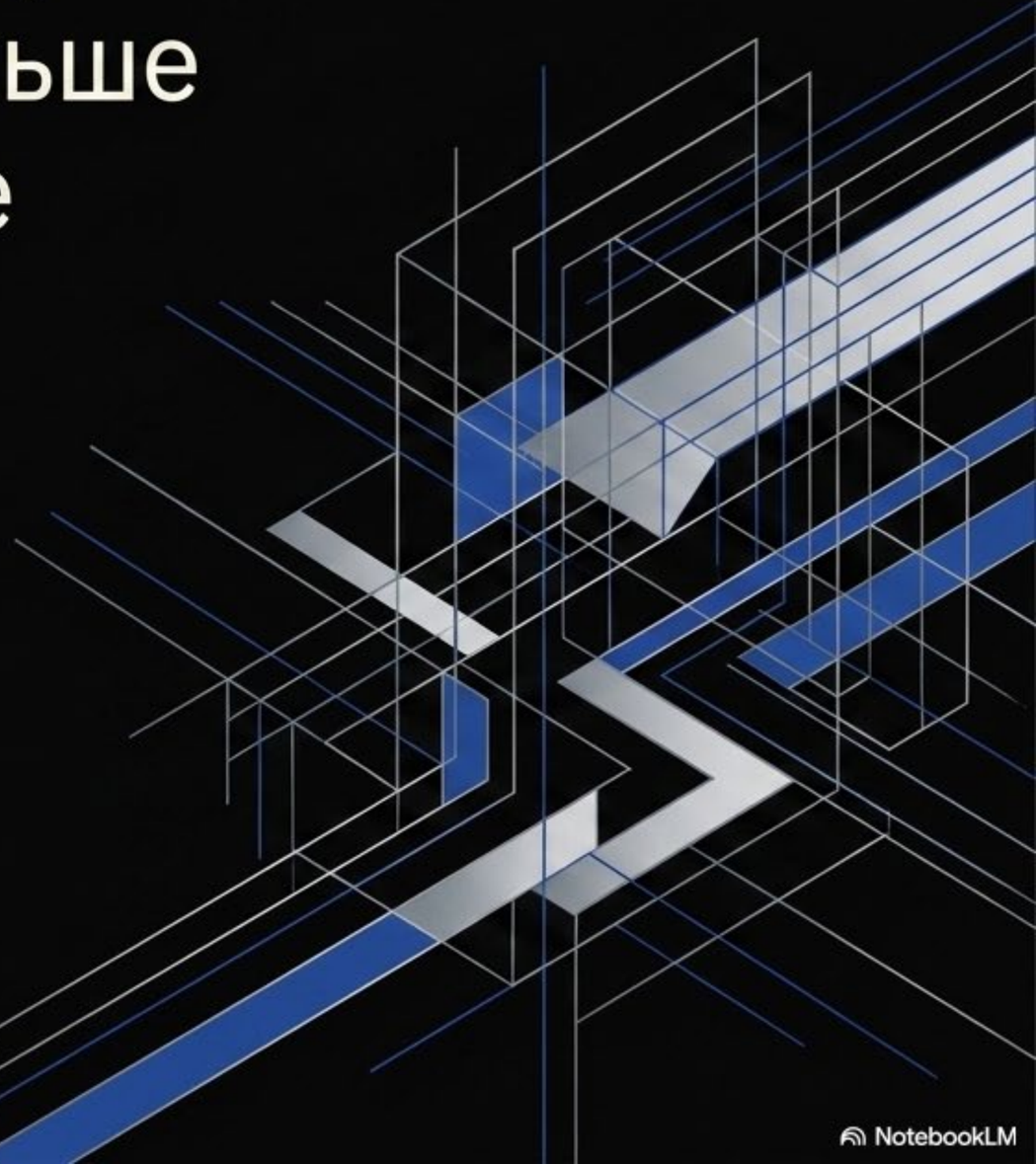
За пределами IVR. Как управлять AI агентами через бизнес-политики

Олег Зельдин Апекс Берг

LLM-агенты в контакт-центре: почему task completion больше не доказывает соблюдение бизнес-правил

JourneyBench, SOP graph и метрика User Journey Coverage Score для оценки policy-aware agents



Источник:

Beyond IVR: Benchmarking Customer Support LLM Agents for Business-Adherence

Sumanth Balaji, Piyush Mishra, Aashraya Sachdeva, Suraj Agrawal

https://arxiv.org/abs/2601.00596

1. Новый риск гибкости: агент может помочь клиенту и нарушить процесс

Почему в LLM-поддержке
task completion уже недостаточен

IVR: Иллюзия контроля

Характеристики:

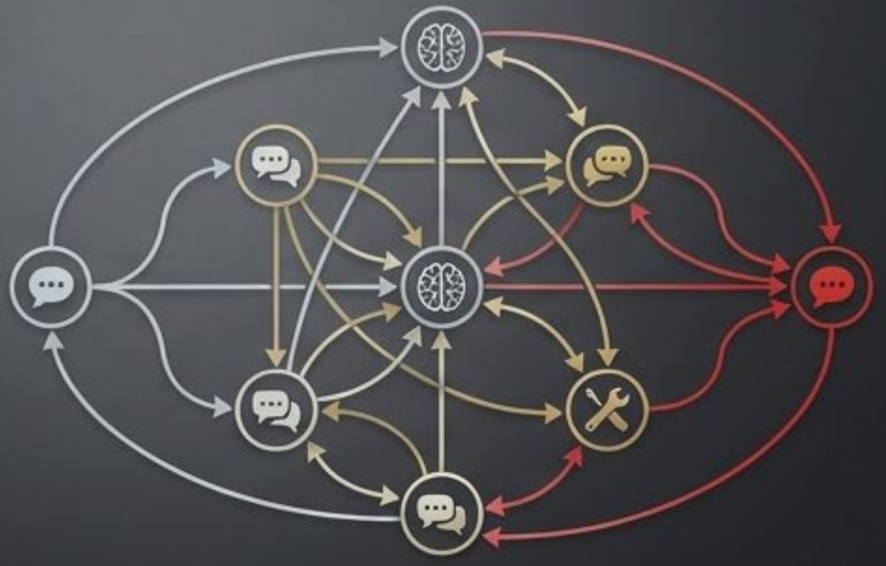
- Жесткие меню, статическое дерево решений, низкая гибкость, раздражение клиента.

Бизнес-ценность:

- Идентификация до действия, обязательные проверки, предсказуемая маршрутизация, минимум импровизации.

Высокий контроль процесса

LLM-агент: Риск свободы



Характеристики:

Многоходовой диалог, естественный язык, tool use, адаптация к клиенту.

Утрата контроля:

Может продолжить без права, может выбрать неправильный путь, может скрыть нарушение хорошим ответом.

Иллюзия успеха: как нарушение SOP выглядит со стороны

Финальный успех разговора **не доказывает правильность** и **легитимность** пройденного пути.

Матрица операционного риска AI

Стандартные метрики:

Containment

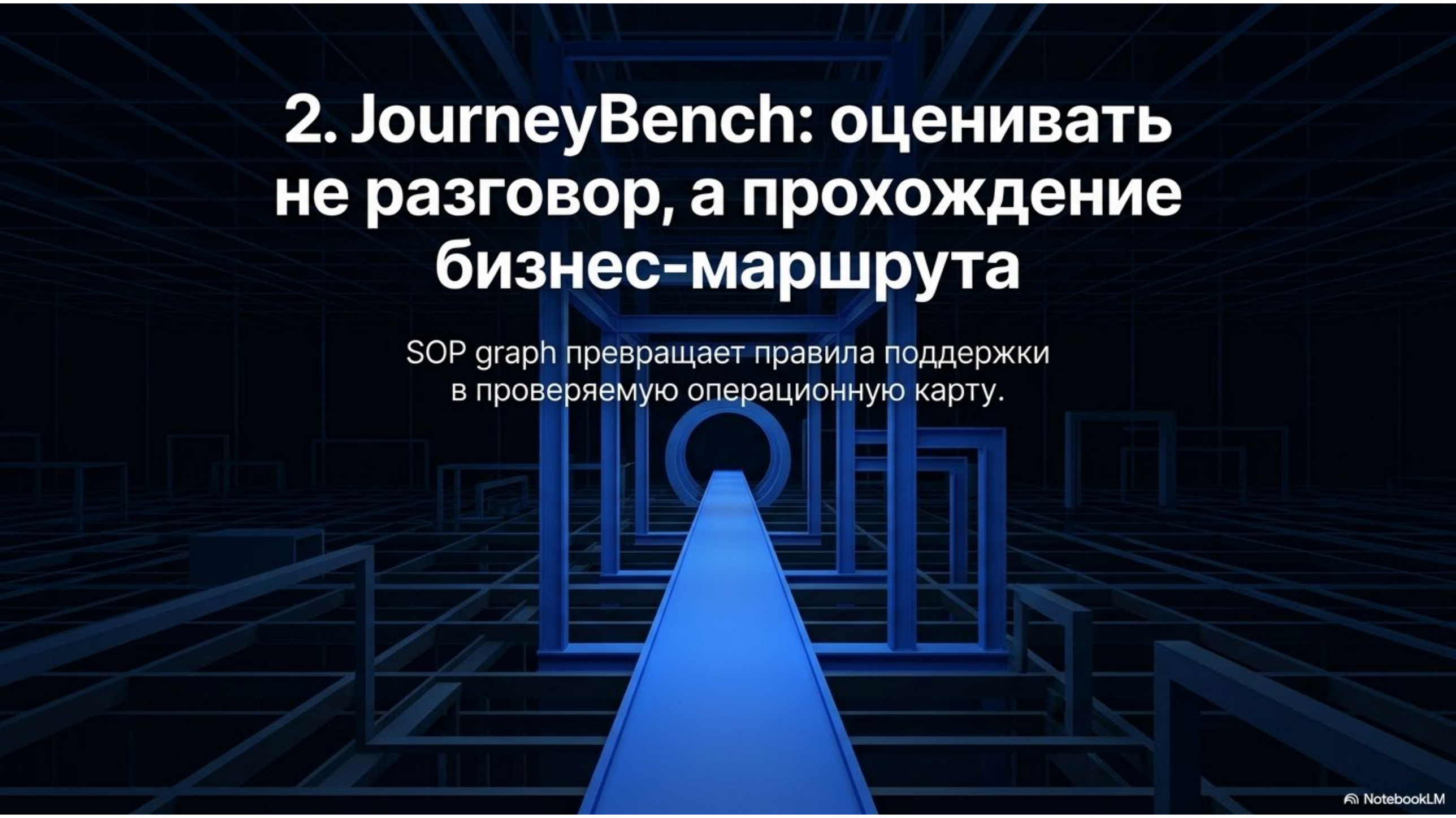
Resolution

AHT

CSAT

QA score

Где метрика Journey Adherence?


2. JourneyBench: оценивать не разговор, а прохождение бизнес-маршрута
SOP graph превращает правила поддержки в проверяемую операционную карту.
NotebookLM

Архитектура фреймворка JourneyBench

SOP Graphs

Граф процесса
(Формализация бизнес-правил).

Nodes

Узел
(Атомарная задача с инструментами и правилами перехода).

User Journeys

Маршрут клиента
(Один конкретный, легитимный путь через граф).

Scenarios

Сценарий
(Тестовый диалог с начальным состоянием и ожидаемой трассой).

Тестирование policy-aware агента начинается не со случайного чата, а со строгой операционной карты процесса.

SOP Graph: Исполняемая операционная карта

SOP graph — это не абстрактная математика, а исполняемая операционная карта. Это не рекомендация для LLM, а жесткое процедурное правило.

Анатомия Узла: Управление на микроуровне

Карта и территория: Маршрут против Сценария

User Journey

Маршрут по процессу. Последовательность узлов и разрешенных вызовов инструментов (tool calls).

Из одного маршрута генерируется несколько стресс-сценариев.

Scenario

Полный тестовый случай. Включает начальное состояние, профиль пользователя, заранее заданные ответы инструментов и Expected Tool Trace.

Масштаб и строгость датасета JourneyBench

703
ДИАЛОГА В ДАТАСЕТЕ

41
ИНСТРУМЕНТ (TOOLS)

3
БИЗНЕС-ДОМЕНА
(E-COMMERCE: 232 | LOAN: 230 | TELECOM: 241)

10.91
ХОДА В СРЕДНЕМ НА ДИАЛОГ

3.34
ВЫЗОВА ИНСТРУМЕНТА В СРЕДНЕМ

ДАТАСЕТ СФОКУСИРОВАН НА POLICY-DRIVEN ПРОЦЕССАХ: ЗАВИСИМОСТИ, ОБЯЗАТЕЛЬНЫЕ ПАРАМЕТРЫ И СЛОЖНЫЕ ВЕТВЛЕНИЯ, А НЕ ОДНОШАГОВЫЕ ЗАПРОСЫ.

3. Как строится JourneyBench: от SOP-графа до стресс-сценариев

Генерация, экспертная проверка, маршруты, синтетические сбои.

Human-in-the-Loop: Генерация и Валидация

Фаза 1: Генерация (LLM как ускоритель)

1. LLM генерирует черновые candidate SOP graphs для 10 доменов.

→

2. Автоматическая проверка ацикличности и связности графа → Автоисправления.

→

3. Обогащение узлов промптами и параметрами инструментов.

Фаза 2: Ручная приемка (Governance)

5 экспертов контакт-центра независимо оценивают графы.

- ☑ Logical Structure (выполнимость)
- ☑ Coherence (согласованность данных)
- ☑ Complexity (баланс сложности)

Правило приемки:
Строгий Unanimous 5-of-5 pass.
(Из 10 графов прошло только 4).

Фаза 3: Построение тестовой среды

Маппинг (BFS)

Алгоритм обхода в ширину (Breadth-First Search) систематически вычисляет все допустимые user journeys через граф.

Simulated User (GPT-4o)

На основе выбранного пути создается user seed — профиль клиента, начальные данные и инструкции вести естественный диалог.

Tools как Черные Ящики

Агент вызывает API, но вместо реальной базы получает заранее подготовленный JSON. Ответ подбирается по логике condition-driven value generation, чтобы принудительно направить агента по целевой ветке.

Фаза 4: Стресс-тестирование compliance

Correct Context
(Счастливый путь)

Сценарий: Все данные присутствуют, API отвечают корректно.

Цель: Пройти стандартный маршрут без галлюцинаций.

Missing Parameter
(Скрытые данные)

Сценарий: Обязательный параметр скрыт пользователем.
Следствие: Downstream вызовы удаляются. Агент обязан остановиться и запросить данные.

Failing Function
(Сбой системы)

Сценарий: Tool call падает (API-ошибка).

Следствие: Невозможные вызовы аннулируются. Агент должен корректно отработать исключение вместо импровизации.

От диалога к дисциплине: новый стандарт AI-операций

Проблема	Решение	Метрика
Успешное завершение разговора (Task Completion) маскирует нарушения бизнес-правил.	Переход на SOP Graphs как проверяемую операционную карту контакт-центра.	Оценка AI-агентов по метрике Journey Adherence — строгому следованию легитимным маршрутам
Успешное завершение разговора (Task Completion) маскирует нарушения бизнес-правил.	Переход на SOP Graphs операционную карту контакт-центра.	Оценка AI-агентов по метрике Journey Adherence — строгому следованию легитимным маршрутам даже в условиях стресса.

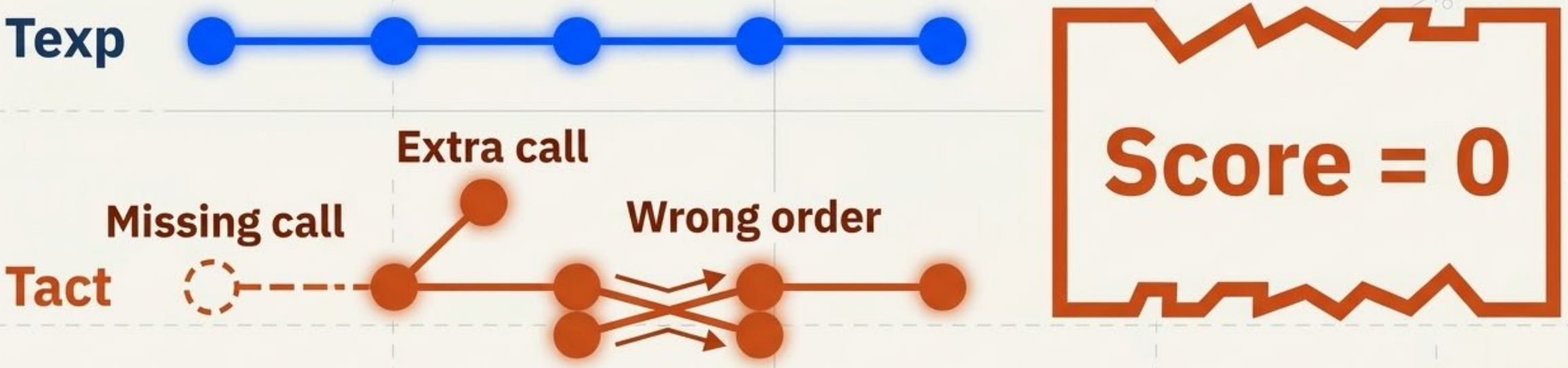
Гибкость LLM должна служить UX, а не разрушать compliance.

UJCS: Жесткая линейка для AI

Метрика, которая ловит неправильный путь к правильному финалу. Измерение дисциплины, а не эмпатии.

Уровень 1: Tool Trace Alignment (Нулевая терпимость)

Фундамент compliance**-логики.** Мы сравниваем **фактическую последовательность** вызовов (Tact) с **ожидаемой трассой** (Texp).



Если **нарушен порядок**, разговор **получает итоговый балл 0**. Процедура **не терпит примерного прохождения.**

Уровень 2 и Итог: От точности параметров к UJCS

Шаг 1: Tool Call Accuracy

Если трасса совпала, проверяем параметры. Пример: creditScore 720 вместо 700 снизит балл, но не обнулит его.

$$TCA_{\text{conv}} = \frac{\sum \text{Корректных параметров}}{\sum \text{Ожидаемых параметров}}$$

Шаг 2: User Journey Coverage Score

Итоговая метрика. Усреднение journey adherence по всему набору симулированных сценариев. Показывает стабильность агента.

$$UJCS = \frac{1}{N} \times \sum TCA_{\text{conv}}$$

Как читать UJCS бизнесу

Высокий UJCS: Агент идеально держит маршрут и передает точные параметры. Безопасен для регламентированных процессов.

Средний UJCS: Маршруты соблюдаются частично. Требуется донастройка графа или промптов.

Низкий UJCS: Агент часто нарушает SOP, путает порядок шагов. Риск compliance-нарушений.

Важно:

UJCS измеряет процесс, а не эмпатию. Метрика не заменяет CSAT, но гарантирует операционную безопасность (task completion).

Дизайн агентов: Двигатель контроля

Почему архитектура оркестрации (DPA) важнее, чем размер языковой модели. Развилка проектирования.

Развилка архитектур: SPA против DPA

Static-Prompt-Agent (SPA)

- **Логика:** Лабиринт целиком. Один гигантский системный промпт. Ветвления через if-then.
- **Доступ:** Все инструменты доступны одновременно.
- **Уязвимость:** Высокий риск context overload. Агент может вызвать инструмент не того этапа или проигнорировать стоп-факторы.

Dynamic-Prompt-Agent (DPA)

- **Логика:** Система шлюзов. SOP работает как машина состояний (state machine).
- **Доступ:** Агент видит только текущий узел и релевантные ему инструменты.
- **Преимущество:** Оркестратор динамически обновляет промпт, открывая доступ только к следующему легальному коридору.

Двигатель DPA: Роль Оркестратора

Шаг 4: Оркестратор оценивает условия и переводит систему на следующий узел.

Шаг 1: Агент получает локальный промпт и доступ к 1-2 инструментам.

Шаг 2: Агент вызывает инструмент (Tool Call).

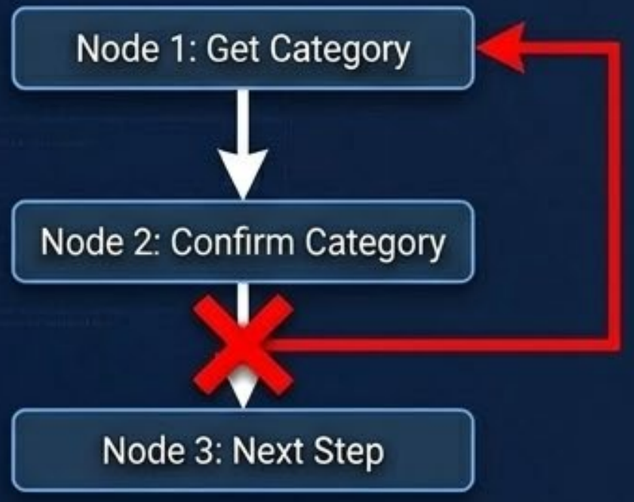
Шаг 3: Инструмент возвращает ответ (Tool Response).

Оркестратор встраивает жесткую операционную дисциплину прямо в поведение модели.

Mid-flow correction: Гибкость без хаоса

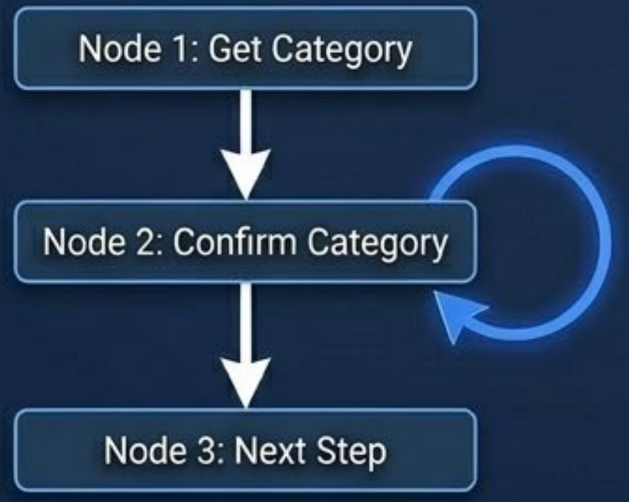
Сценарий: Пользователь указывает категорию дохода Salaried, а затем внезапно исправляет на Self-Employed в середине процесса.

Старый IVR



Жесткий сброс. Клиент вынужден возвращаться назад или начинать ветку заново.

Динамический Агент (DPA)



Обновление данных внутри текущего узла. Оркестратор пересчитывает переменные и мягко продолжает процесс.

Вывод: Гибкость становится безопасной, только когда она управляется строгой машиной состояний.

Резюме: Три кита надежного Enterprise AI

Регламентированный чертеж (JourneyBench)

AI помогает в создании SOP, но эксперт 5-of-5 остается финальным фильтром.

Строгий аудит (UJCS)

Мы не измеряем разговор, мы измеряем математическую точность следования процессу (Tool Trace & Tool Parameters).

Динамический контроль (DPA)

Отказ от лабиринта в одном промпте в пользу пошагового оркестратора и машины состояний.


Архитектура надежных LLM-агентов: от экспериментов к governance
Анализ бенчмарков JourneyBench, классификация скрытых сбоев и операционная модель внедрения DPA-агентов в контакт-центрах.
Executive Summary для C-level, Head of AI и директоров клиентского сервиса.
NotebookLM

Эксперименты доказали, что структура оркестрации перевешивает размер модели

DPA (Dynamic Process Automation)

В операционной среде контакт-центра то, как агент встроен в процесс, важнее количества параметров в самой LLM.

SPA (Static Prompt Approach)

Дашборд исследования JourneyBench: 4 модели, 2 дизайна, стресс-тестирование

LLM Engines

- ✓ GPT-4o
- ✓ GPT-4o-mini
- ✓ Claude 3.5 Haiku
- ✓ Llama 3.3

Среда и Условия

- 3 бизнес-домена
- 3 типа сценариев
- Строгий лимит — 40 реплик (turns)

Надально

Prementaцiи

Архитектура

- SPA (Static Prompt)
- DPA (Dynamic Process)

Метрика и Симулятор

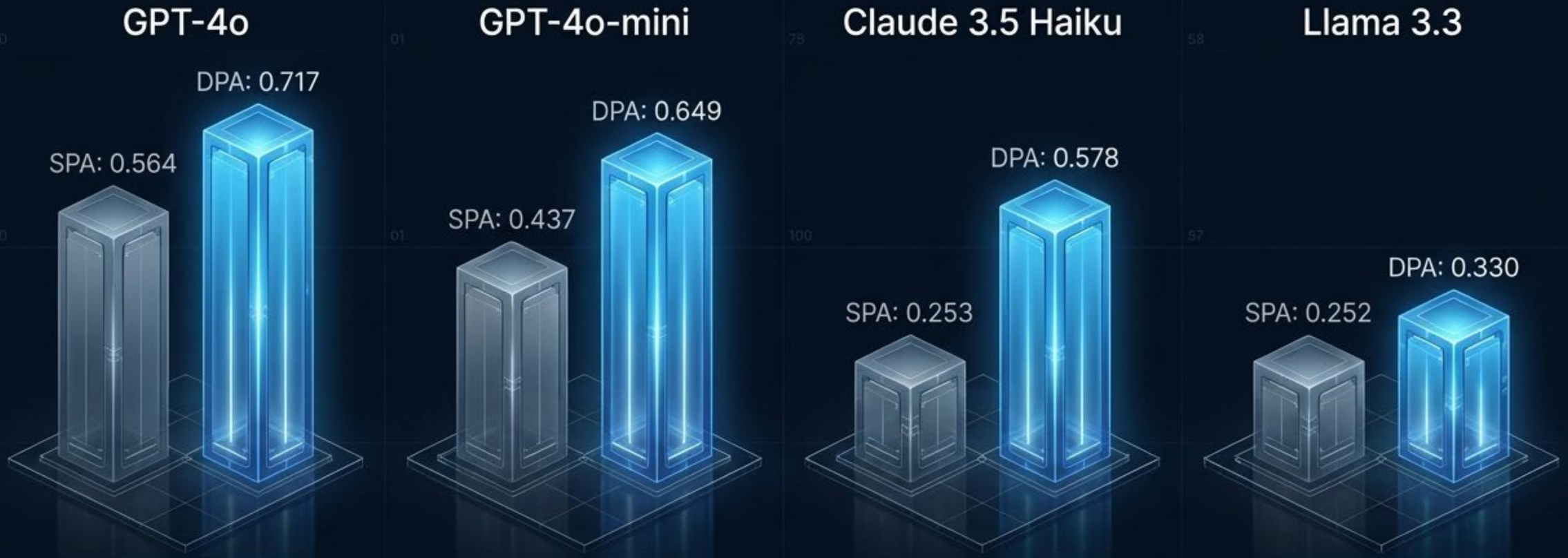
Главный KPI:

UJCS

(User Journey Completion Score)

Симулированный клиент: GPT-4o

Управляемый процесс повышает результативность любой базовой модели



Вывод: Динамическая привязка к процессу стабилизирует и улучшает показатели независимо от провайдера и архитектуры самой LLM.

На «счастливом пути» разницы нет. При сбое системы — статика рушится

Correct Context
(Идеальные условия)

GPT-4o:	SPA: 0.871	DPA: 0.873
GPT-4o-mini:	SPA: 0.720	DPA: 0.718

Показатели почти идентичны.

Failing Function
(Сбой API / Отказ системы)

GPT-4o:	SPA: 0.511 (провал)
	DPA: 0.857 (устойчиво)
GPT-4o-mini:	SPA: 0.326 (обрушение)
	DPA: 0.816 (спасение процесса)

Инсайт: Когда API падает или данных не хватает, агент без явной оркестрации начинает импровизировать. DPA блокирует галлюцинации.

Парадокс эффективности: меньшая модель с рельсами бьет большую без рельс

Дорогая модель без рельс

GPT-4o + SPA

Индекс UJCS: 0.564

>

Бюджетная модель со строгим процессом

GPT-4o-mini + DPA

Индекс UJCS: 0.649

Холодный душ для стратегии «купим модель побольше».
Если рельсы (SOP) заданы жестко, меньшая модель работает надежнее, предсказуемее и кратно дешевле в масштабах контакт-центра.

Проверка реальностью: архитектура DPA в production-среде

Масштаб (Scale)

Развернуто в реальных контакт-центрах

6000+

Обрабатывает звонков ежедневно

Realism Validation (Оценка качества)

Оценка через LLM-as-a-judge (QA-рубрика)

Overall Реалистичность:	**84.37%**
Conversational Proficiency (CP):	**82.33%**
Goal Attainment (GA):	**87.78%**

Резюме: Академический бенчмарк жестко привязан к реалистичной QA-практике корпоративного уровня.

Анатомия ошибок агентов: что должно видеть QA кроме финального ответа

1. Нарушения зависимостей (SOP violations)

2. Галлюцинации параметров

3. Сбои симулятора симулятора (слепые зоны тестирования)

Пробой логики: игнорирование обязательных бизнес-зависимостей

Вывод: Dependency violation — критическая зона риска. DPA блокирует движение без прохождения обязательных гейтов.

Генеалогия параметра: скрытые галлюцинации в значениях

Блок Пользователя

«Мой credit score — **720**»

Блок Системы (Инструкция)

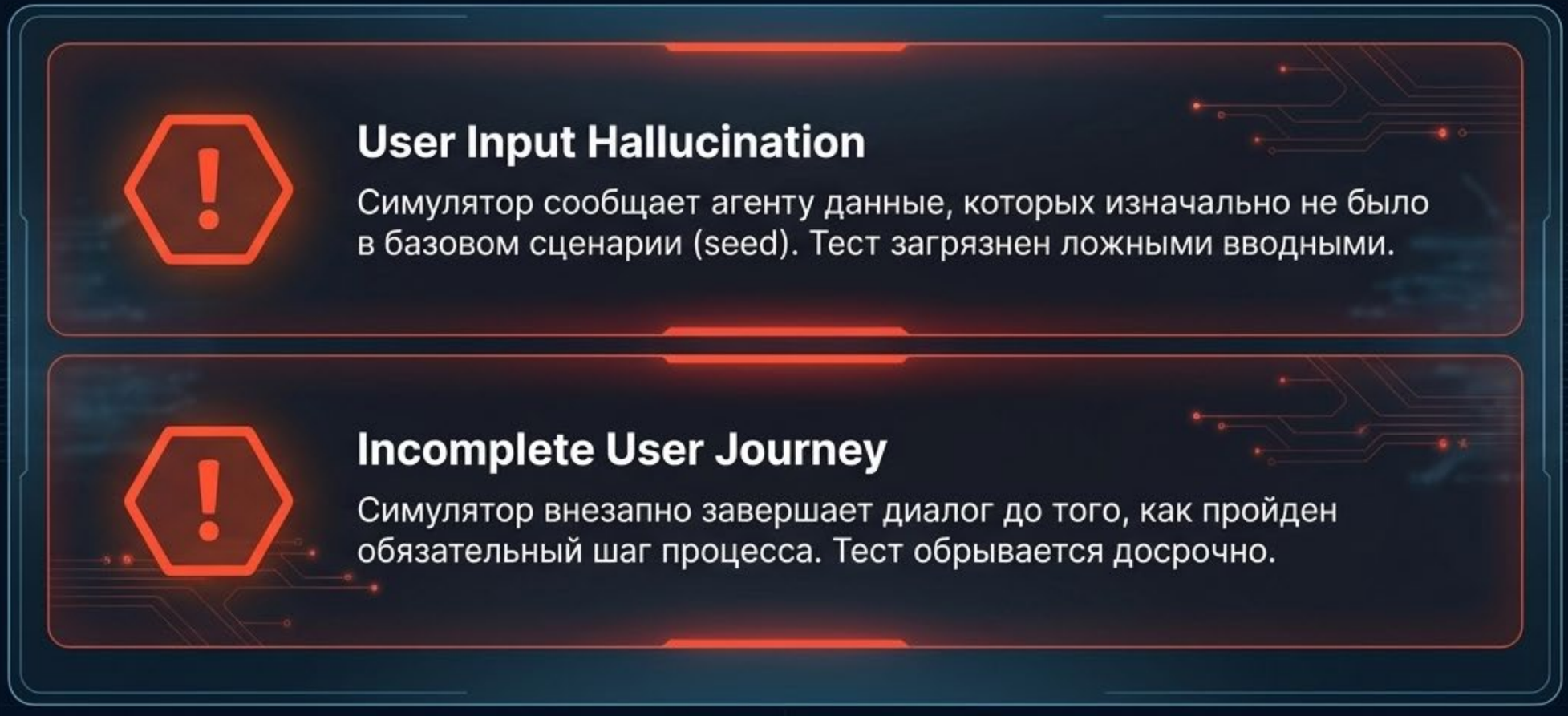
Описание инструмента: подсказка example value = **700**

Блок Агента

Вызывает инструмент со значением: **700**

Правило для QA: Проверять не только факт вызова инструмента (**tool call**), но и жесткую трассируемость каждого значения обратно к **seed**-данным клиента.

Слепая зона тестирования: сбои пользовательского симулятора



Ключевой тезис: При построении синтетического QA вы рискуете измерять шум. Валидировать необходимо не только тестируемого агента, но и самого тестового клиента.

Матрица контроля: четырехуровневая пирамида AI-QA

Фундамент полигона

Валидность теста: проверка стартового seed, качества симулятора и ожидаемой трассы (expected trace).

Практика внедрения: архитектура Governance для контакт-центров

1. SOP (инструкции) как строго исполняемая карта.
2. Трансформация процессов QA по трассе графа.
3. Обязательные стресс-тесты до выхода в production.

Инженерный арсенал: от статичной инструкции к симуляции

A: Данные узлов
user seed, tool response, node definition.

B: Генерация SOP
промпт для трансформации текста в SOP-граф.

C: Валидация
автоматическая проверка структуры графа.

D: Ветвления
condition-driven value generation.

E: Шаблоны
template статического промпта (защита от перегрузки контекста).

F: Симулятор
промпт симулированного пользователя (с жестким запретом на галлюцинации данных).

Готовый набор заготовок для развертывания внутреннего AI-QA стенда.

Чеклист приемки SOP-графов: правило «5-of-5 pass»

Автоматический сканер

- Единый стартовый узел (One start node)
- Достижимость всех узлов (Reachability)
- Отсутствие мертвых циклов (Cycle detection)
- Валидация переменных и выражений

Экспертный аудит

- Логическая структура (Logical Structure)
- Согласованность бизнес-смысла (Coherence)
- Адекватность сложности (Complexity)
- Реалистичность диалогов (QA-оценка CP и GA)

В production отправляется только граф, прошедший абсолютный барьер «5-of-5 pass».

Операционная модель AI: единый часовой механизм контакт-центра

Ограничения подхода и управление операционными рисками

Зависимость от логики

DPA работает ровно настолько хорошо, насколько идеально описан процесс. При частых изменениях графы требуют непрерывной поддержки.

Пределы симуляций

Синтетический QA не способен покрыть 100% непредсказуемости живых клиентов.

Экономика оценки

Масштабное LLM-тестирование имеет прямую стоимость (в рамках бенчмарка — $388.88 за цикл), что диктует выбор моделей.

Влияние на персонал

Внедрение требует перепрофилирования агентов-людей на более сложные задачи (эскалации) и ответственного change management.

Пять золотых правил внедрения AI-агентов

1. SOP должны стать исполняемыми операционными картами, а не текстом.
2. QA обязано проверять трассу инструментов и параметры, а не только тон ответа.
3. Тесты должны включать пропуски данных и сбои API, а не только «счастливый путь».
4. Оркестрация логики — это ядро операционной модели, а не деталь для разработчиков.
5. Классический Task Completion необходимо дополнить метрикой Journey Adherence.

«Вопрос успешного внедрения звучит так: не просто решил ли агент задачу, а имел ли он право решать ее именно так»