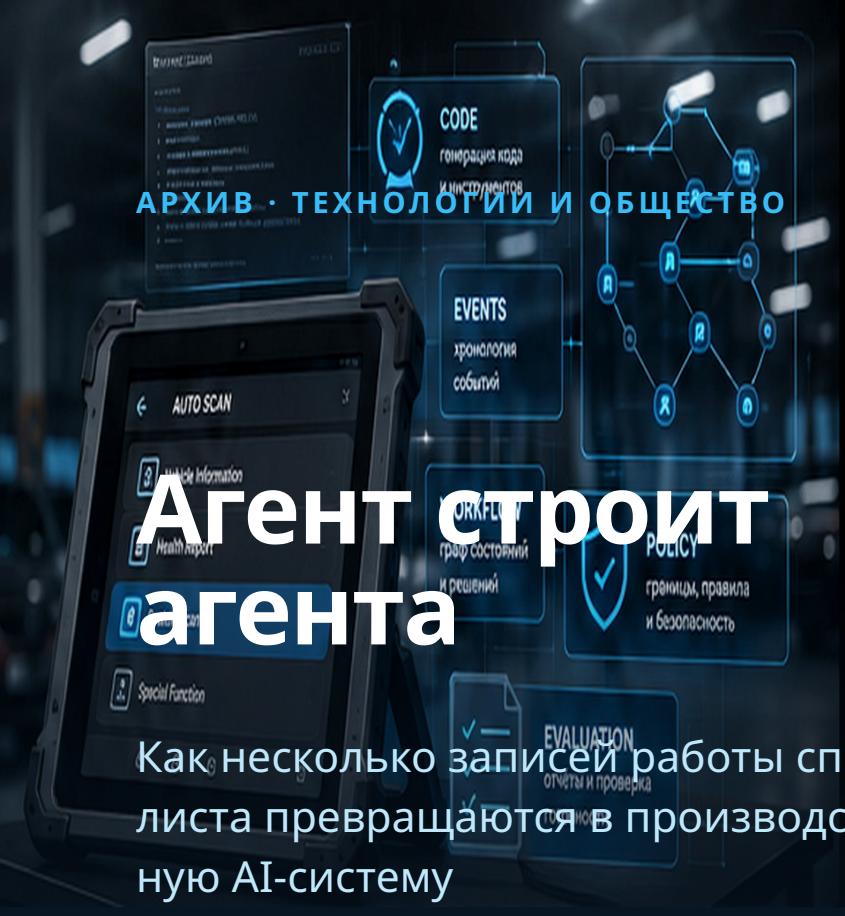

АРХИВ · ТЕХНОЛОГИИ И ОБЩЕСТВО

Агент строит агента

Как несколько записей работы специалиста превращаются в производственную AI-систему

Сергей Авдейчик

Публикационная версия · 2026

> **Одни агенты наблюдают. Другие превращают опыт в данные. Третьи проектируют правила. Четвёртые пытаются систему сломать — и только после этого пятые получают ограниченное право действовать.**

ПУБЛИКАЦИОННАЯ ВЕРСИЯ · АВГУСТ 2026

Методология построения агентов для реальных производственных процессов

Современная агентная система может перенимать ограниченный производственный workflow не через запоминание тысяч кликов, а через последовательное превращение человеческого опыта в наблюдения, семантические контракты, тесты и строго ограниченные полномочия. Важнее всего здесь то, что на каждом этапе одну агентную систему помогают строить другие.

Главный тезис: мы не обучаем модель повторять движения специалиста. Мы восстанавливаем грамматику его работы — состояния, цели, допустимые действия, исключения и доказательства результата.

Карта статьи

На экране автомобильного диагностического планшета рядом могут находиться две внешне похожие команды. Одна считывает сведения о неисправности. Другая удаляет их, запускает сервисную процедуру или изменяет состояние электронного блока.

Для опытного автоэлектрика между ними лежит пропасть. Он понимает, на каком этапе диагностики находится, что уже проверено, почему открыто именно это меню и к каким последствиям приведёт следующий шаг. Для наивной AI-системы перед глазами всего лишь несколько надписей и прямоугольников, по которым можно нажать.

Именно здесь заканчивается эффектная демонстрация и начинается инженерия.

Можно показать модели несколько видеозаписей, попросить её предсказывать следующее нажатие и получить убедительный ролик. Но такой результат будет ближе к вероятностному макросу, чем к производственной агентной системе. Он не доказывает, что модель понимает цель работы, различает безопасное и опасное, замечает изменение состояния, помнит уже проверенные узлы и способна остановиться там, где доказательств недостаточно.

В одном из своих проектов я исследую более сложный путь: как при помощи одних агентных систем построить другие. Сначала языковые модели помогают создать программы наблюдения и телеметрии. Затем мультимодальные модели превращают работу специалиста в структурированные данные. Следующий агентный контур ищет в этих данных повторяющиеся решения, исключения и признаки завершённости. Ещё один помогает проектировать системную инструкцию, контракты и политику безопасности. Отдельный агент пытается эту конструкцию сломать. И только после этого runtime-агент получает возможность работать — сначала в симуляции, затем в режиме наблюдения и лишь потом, при выполнении жёстких условий, в ограниченном исполнительном контуре.

Это не история о том, как AI научился нажимать кнопки на автомобильном планшете. Это методология превращения человеческого опыта в проверяемую производственную систему.

В инженерной реализации метод разделён на три связанных контура. **AutoDiag Replay** пассивно записывает работу специалиста. **AutoDiag Replay Evidence Artifacts** сохраняет обезличенные доказательные пакеты, отчёты и контрольные суммы. **OBDPulse 3** превращает этот материал в среду воспроизведения, оценки и поэтапного построения runtime-агента. Такое разделение важно: сбор данных, хранение доказательств и право на действие не должны сливаться в одну непрозрачную систему.

Главный тезис: мы не обучаем модель повторять движения специалиста. Мы восстанавливаем грамматику его работы — состояния, цели, допустимые действия, исключения и доказательства результата.

Профессия как скрытый язык

Большая часть профессионального знания не записана в инструкциях.

Формальный регламент может сказать: подключить диагностический прибор, выполнить общий опрос систем, открыть проблемные блоки, считать коды неисправностей и подготовить отчёт. Но он редко фиксирует десятки микрорешений, из которых на самом деле складывается работа.

Специалист замечает, что экран ещё не стабилизировался. Понимает, что список продолжается ниже видимой области. Различает одинаковые надписи, относящиеся к разным электронным блокам. Возвращается назад не потому, что ошибся, а потому, что завершил одну ветвь. Не принимает общий отчёт за доказательство полноты. Знает, какие вопросы требуют физического подтверждения человека: включено ли зажигание, подключён ли разъём, соответствует ли найденный автомобиль реальному.

Если попросить специалиста рассказать обо всём этом после работы, часть решений исчезнет из памяти как само собой разумеющаяся. Если снять только видео, останется изображение процесса, но не его машинно-проверяемая структура. Если записать лишь координаты касаний, получится сценарий, который сломается после изменения интерфейса, прокрутки или версии приложения.

Поэтому первый этап я называю **фотографированием человеческого опыта**. Но фотографией здесь служит не один скриншот и не видеоролик. Для каждого шага создаётся доказательная трасса:

- какое состояние видел человек до действия;
- какое физическое действие он совершил;
- какие промежуточные состояния возникли;
- какой экран стабилизировался после действия;
- изменился ли процесс ожидаемым образом;
- что было уже проверено и что ещё оставалось сделать;
- где наблюдение было неоднозначным.

Прокрутка хранится как упорядоченная последовательность исходных кадров, а не как удобная, но производная склейка. Повторная попытка не стирает неудачную. Исходные файлы получают контрольные суммы. Причина решения специалиста, если она нужна, записывается отдельно и не выдумывается моделью по положению пальца.

Особенно важно не смешивать четыре разных сущности:

что было написано на экране → куда фактически нажал человек → какое профессиональное действие это означало → почему специалист выбрал его именно сейчас.

Только после такого разделения запись превращается в материал, пригодный для обучения и проверки системы.

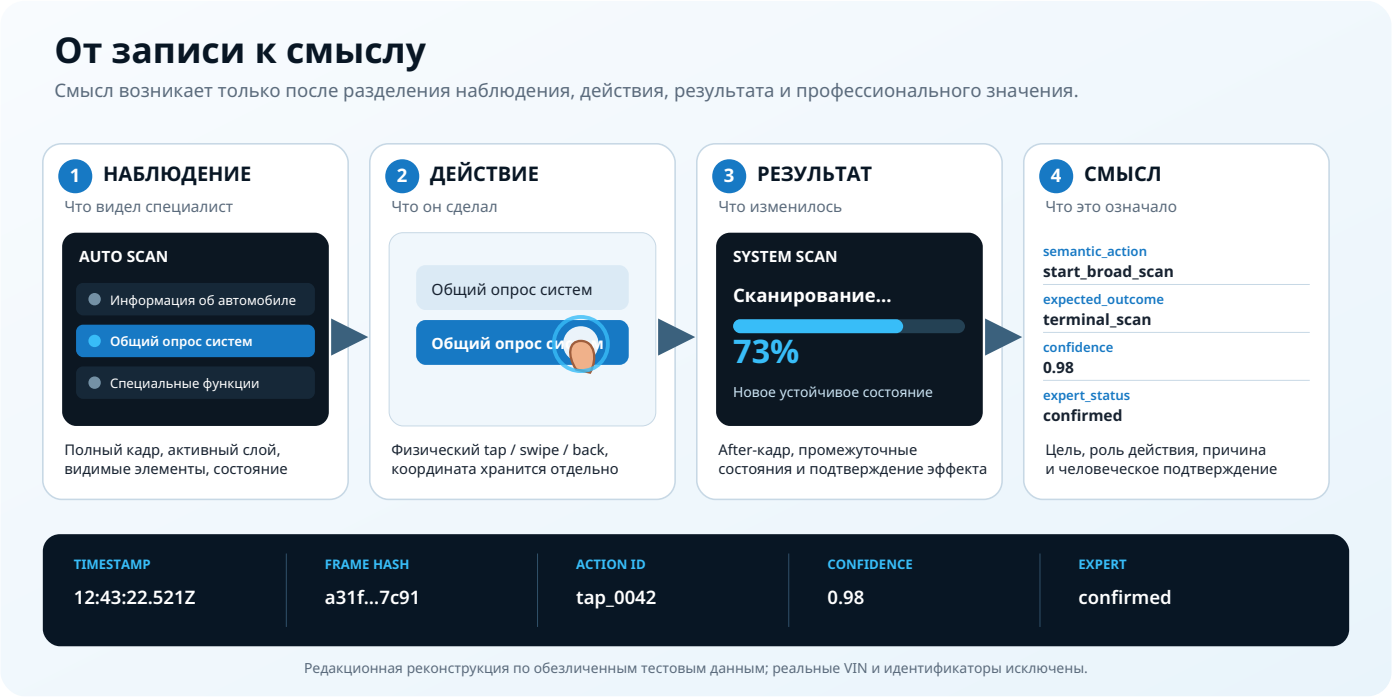

Иллюстрация 1. От записи к смыслу. Смысл не хранится в координате. Он появляется после разделения наблюдения, действия, результата и экспертного значения.

Агентная фабрика

Выражение «агент строит агента» легко понять неправильно. Оно может вызвать образ одной самосовершенствующейся модели, которая переписывает собственный код, сама себя экзаменует и затем объявляет готовой к работе.

Такую систему я как раз не строю.

Речь идёт не об одном цифровом существе, а о **фабрике с разделением труда и разделением властей**. На разных этапах могут использоваться одна и та же фундаментальная модель или разные модели, но логически это независимые роли с разными входами, полномочиями и критериями успеха.

1. Агент-разработчик создаёт наблюдателя

Сначала coding-агенты помогают исследовать устройство, писать безопасные probes, recorder, схемы данных, валидаторы, тесты и инструкции для воспроизводимого запуска.

Принципиальная граница первого контура: он наблюдает за работой оператора, но не управляет диагностическим приложением. Задача — получить надёжную телеметрию, не вмешиваясь в процесс.

2. Агент-наблюдатель переводит пиксели в структуру

Мультимодальная модель получает отдельный экран или упорядоченную последовательность экранов и строит его семантическое описание: тип экрана, активный слой, видимые тексты, элементы управления, модальные окна и неопределённости.

Она не отвечает на вопрос «что нажать». Это принципиально другая роль. Её задача — расшифровать интерфейс, а не управлять им.

3. Агент-куратор собирает корпус

Следующий контур проверяет целостность архивов, связывает события по времени, нормализует названия действий, выявляет пропуски, создаёт технические отчёты и производные наборы данных.

Первичные трассы остаются записью реальной работы специалиста. Синтетическими могут быть дополнительные вариации, обезличенные примеры и специально созданные пограничные случаи. Смешивать эти два происхождения нельзя.

4. Агент-аналитик извлекает грамматику процесса

Его интересует не последовательность координат, а повторяющаяся логика:

- как определяется текущее состояние;
- какие факты уже подтверждены;
- какие действия допустимы именно сейчас;
- какие ветви ещё не исследованы;
- что является доказательством завершённости;

- при каких условиях нужно задать вопрос человеку;
- когда необходимо отказаться от продолжения.

Из нескольких операторских трасс начинает формироваться не макрос, а граф состояний и решений.

5. Агент-архитектор проектирует будущего исполнителя

На основании корпуса он помогает подготовить системную инструкцию, типизированные входы и выходы, ограниченный словарь намерений, память прогресса и правила завершения.

Но инструкция не является единственным защитным слоем. Всё, что связано с разрешением реального действия, остаётся в детерминированном коде вокруг модели.

6. Агент-валидатор пытается доказать, что система не готова

Он ищет не подтверждение успеха, а контрпример: одинаковые надписи в разных строках, устаревшее состояние экрана, неожиданное модальное окно, повреждённый скриншот, скрытый ниже прокрутки элемент, потерю соединения, преждевременное объявление результата.

Каждая найденная ошибка превращается в новый тест. Исправленная система снова проходит весь замороженный корпус, а не только пример, на котором её поправили.

7. Runtime-агент работает внутри выданных ему полномочий

Только последний контур непосредственно решает, какой семантический шаг предложить следующим. При этом он не получает прямого доступа к устройству, shell-командам или произвольным координатам.

Он предлагает намерение. Право на действие выдаёт система вокруг него.

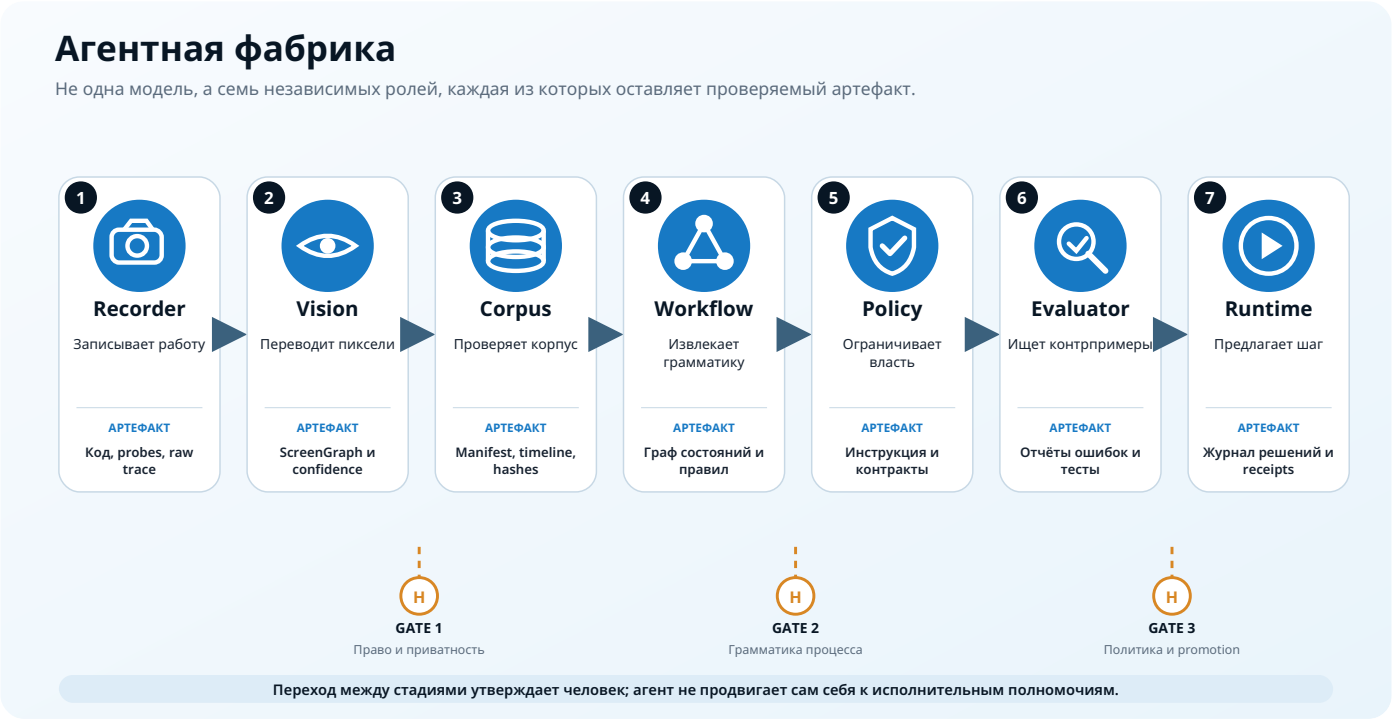

Иллюстрация 2. Фабрика агента. Каждая роль создаёт собственный проверяемый артефакт; переход между стадиями утверждает человек.

Как несколько примеров могут заменить тысячи

Здесь возникает естественное возражение. Не слишком ли смело говорить о самостоятельной работе агента, если исходных записей мало?

Для классического машинного обучения ответ часто был бы утвердительным. Если модель обучается с нуля распознавать конкретные классы экранов или напрямую связывать изображения с действиями, ей действительно потребуются большие размеченные наборы данных.

Но фундаментальная модель приходит в проект не с пустой головой. До встречи с конкретным диагностическим планшетом она уже обучалась на огромных массивах текста, изображений и кода. Она знает общие паттерны интерфейсов, языка, последовательностей действий, программных состояний и причинных связей. Локальные примеры не создают интеллект заново. Они **привязывают универсальные способности к грамматике конкретного производственного процесса**.

Исследования few-shot и in-context learning показали, что поведение большой языковой модели может существенно адаптироваться к новой задаче по инструкции и нескольким демонстрациям без обновления её весов. Работы в робототехнике добавили важное ограничение: языковое рассуждение должно быть связано с набором реально доступ-

ных и проверенных навыков, а знания, полученные при масштабном предварительном обучении, могут улучшать перенос на новые объекты и команды. Эти результаты не доказывают безопасность конкретного автомобильного агента, но объясняют, почему небольшой локальный корпус способен служить заземлением, а не полным учебником профессии.[1][2][3]

В инженерном проекте слово «обучение» поэтому имеет несколько уровней.

Могут изменяться не веса модели, а:

- системная инструкция;
- набор few-shot примеров;
- семантический граф workflow;
- память подтверждённых фактов и пройденных ветвей;
- правила host-side policy;
- библиотека допустимых возможностей;
- тестовый и adversarial-корпус.

Иными словами, обучается не только нейросеть. **Обучается вся агентная система.**

Небольшой корпус полезен не потому, что пять примеров магически описывают весь мир. Он полезен, когда каждый пример обладает высокой информационной плотностью. Один показывает нормальный маршрут. Другой — длинную загрузку. Третий — список, уходящий ниже экрана. Четвёртый — одинаковые названия в разных контекстах. Пятый — вопрос, на который имеет право ответить только человек.

Десять почти одинаковых успешных прогонов могут дать меньше знания, чем три тщательно выбранных контрпримера.

Поэтому главный ресурс — не количество изображений как таковое, а разнообразие состояний, прозрачность их происхождения и качество разделения данных. Часть трасс используется для разработки. Другая замораживается как holdout и не должна становиться подсказкой при исправлении системы.

> **Мы не обучаем модель профессии на пяти сеансах. Мы связываем уже обученную универсальную модель с локальной грамматикой работы — и затем проверяем границы этого переноса.**

Малое число примеров даёт возможность быстро построить первую версию. Оно не даёт права объявить процесс полностью освоенным. Каждый новый тип ошибки расширяет корпус, а каждый неподдерживаемый случай должен приводить не к догадке, а к безопасной передаче человеку.

Не запомнить маршрут, а перенести правило

Настоящая проверка начинается там, где агент видит сценарий, которого не было в точности ни в одной демонстрации.

У другого автомобиля меняется структура меню. Сканирование занимает больше времени. Проблемных блоков оказывается не три, а семь. Название безопасной команды сформулировано иначе. Возникает дополнительное подтверждение. Несколько одинаковых кнопок относятся к разным системам. После обновления приложения элементы немного смещаются.

Координатный макрос в таких условиях ломается. Агентная система должна работать на другом уровне абстракции.

Её намерение выглядит не как:

`нажать x=586, y=190`

а как:

`открыть следующий ещё не проверенный проблемный блок`

или:

`считать коды неисправностей в текущем подтверждённом блоке`

Модель получает текущее наблюдение, цель сессии, подтверждённые факты, историю уже выполненных действий, список непроверенных систем и ограниченный словарь допустимых намерений. Она предлагает один следующий семантический шаг.

Дальше начинается работа детерминированного host-кода. Он проверяет, существует ли на свежем экране ровно один элемент, соответствующий этому намерению. Разрешено ли действие в текущем режиме. Не изменилось ли состояние после того, как модель увидела экран. Не относится ли элемент к запрещённой функции. Не требуется ли вместо действия вопрос человеку.

Так агент получает способность комбинировать знакомые примитивы в ранее не записанных сочетаниях. Он не видел именно этот маршрут, но знает отношения между целью, состоянием и допустимым следующим шагом.

Именно в этом смысле небольшое число трасс может привести к поведению, охватывающему множество сценариев. Не любое мыслимое состояние мира, а расширяющийся класс ситуаций, описываемых одной и той же проверяемой грамматикой.

Модель предсказывает. Система разрешает

В основе большой языковой модели действительно лежит предсказание следующего токена. Этот факт никуда не исчезает после того, как мы называем модель агентом.

Но производственное действие выполняет не «следующий токен».

Между выводом модели и физическим устройством должна находиться архитектура полномочий:

свежее наблюдение → типизированное предложение агента → проверка возможности → детерминированная политика → единственный исполнитель → подтверждение результата → новое наблюдение .

Модель может распознать смысл ситуации и предложить действие. Но она не должна сама:

- выдавать себе разрешение;
- выбирать произвольные координаты;
- обходить общий исполнительный контур;
- объявлять неоднозначное действие успешным;
- повторять команду, если неизвестно, была ли она уже выполнена;
- расширять собственный список допустимых операций.

В моей архитектуре разрешение привязывается к конкретному свежему состоянию. Экран изменился — прежняя возможность истекла. Система не нашла однозначного безопасного соответствия — действие не выполняется. Потеряно подтверждение результата — автоматический повтор запрещён. Неизвестное состояние — это не приглашение к творчеству, а основание для остановки или передачи оператору.

Так интеллект модели отделяется от власти модели.

Это похоже на принцип наименьших привилегий в информационной безопасности. Сотрудник может понимать задачу, но получает доступ только к тем операциям, которые нужны ему сейчас. Агент может рассуждать широко, но действовать должен узко.

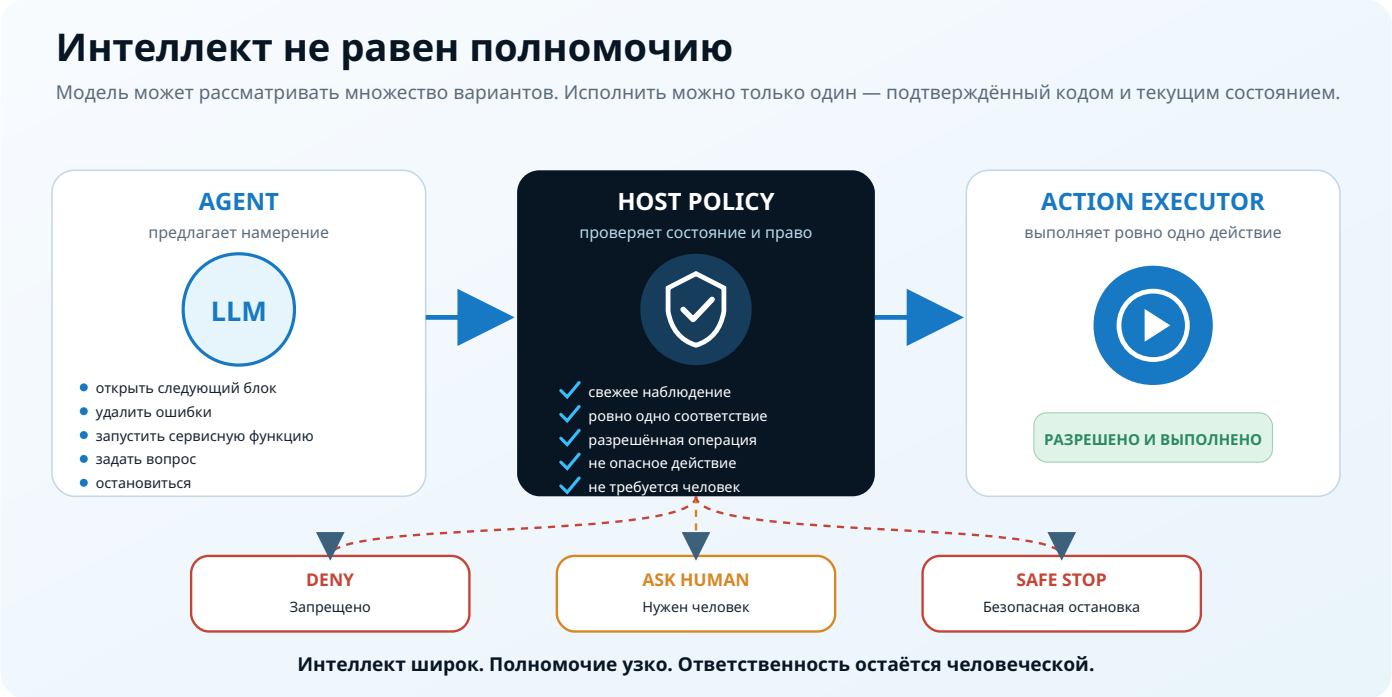

Иллюстрация 3. Интеллект не равен полномочию. Модель предлагает семантическое намерение, но право на действие остаётся в детерминированном host-контуре.

Когда агент проверяет агента

Обычное тестирование программного продукта отвечает на вопрос: выполняет ли код заранее определённые требования?

Для LLM-систем этого недостаточно. Модель может дать правильный ответ по неправильной причине. Может использовать информацию, появившуюся позже. Может случайно угадать кнопку. Может пройти знакомую трассу, но провалиться после небольшого изменения интерфейса. Может убедительно объявить работу завершённой, хотя одна из ветвей осталась ниже видимой области.

Поэтому оценка должна быть причинной.

При воспроизведении исторической сессии агент на каждом шаге видит только ту информацию, которая существовала в тот момент. Будущий экран, следующее действие механика и финальный результат ему недоступны. После предложения агента система сравнивает его не только с фактическим нажатием человека, но и с целью процесса, правилами безопасности и доказательством завершённости.

Отдельный evaluator ищет сценарии, в которых системная инструкция допускает ошибку. Он может переставлять контекст, подбрасывать похожие элементы, создавать повреждённые наблюдения, проверять потерю связи и провоцировать преждевременный `finish`.

Но и здесь агент-валидатор не становится верховным судьёй. Он формирует находки и доказательства. Ground truth утверждает названный человек. Переход к следующей стадии разрешает владелец процесса. CI подтверждает только конкретную версию кода и корпуса, а не абстрактную «надёжность AI».

На текущем этапе мой исполнительный проект сознательно остаётся в **read-only Shadow Mode**. Агент анализирует реальные или воспроизведённые состояния и предлагает решения, но не получает исполнительного канала к планшету. Его предложения сравниваются с доказательными трассами и человеческой оценкой.

Это не неудобная оговорка перед красивой презентацией. Shadow Mode — одна из важнейших производственных стадий. Он позволяет увидеть ошибки поведения до того, как ошибка станет физическим действием.

Высокая доля корректных решений полезна. Но для критических ветвей более важны другие показатели: отсутствие опасных исполненных действий, отсутствие ложного завершения и корректный отказ в неподдерживаемой ситуации.

Хороший ранний агент может часто звать человека. Плохой ранний агент уверенно продолжает.

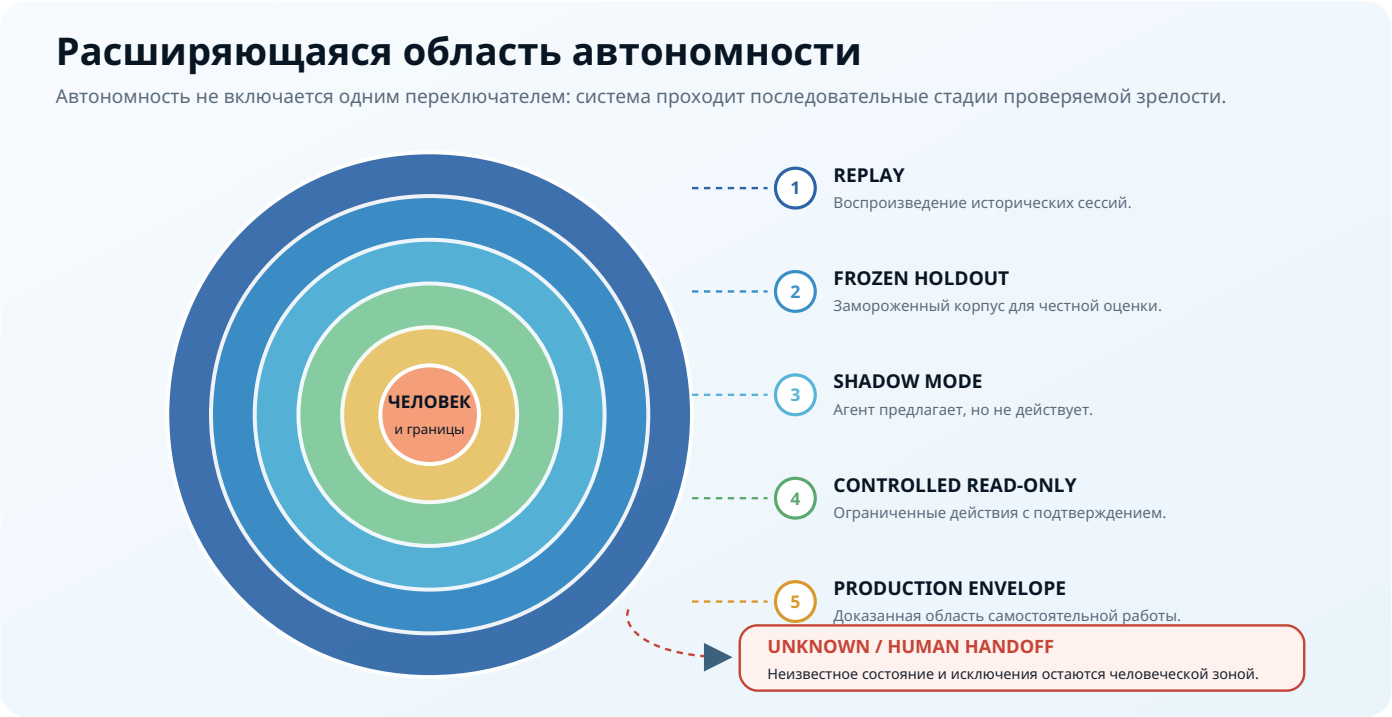

Иллюстрация 4. Расширяющаяся область автономности. Replay, frozen holdout и Shadow Mode не являются формальностью: они последовательно расширяют только доказанную область самостоятельной работы.

Кого именно заменяет такая система

Вопрос «может ли AI заменить автоэлектрика?» поставлен слишком грубо.

Если под заменой понимать всю профессию — физический осмотр, постановку диагноза, интерпретацию симптомов, ремонт, ответственность и работу с редкими неисправностями, — ответ сегодня отрицательный.

Но профессия не является неделимым объектом. Она состоит из множества задач и микрорешений.

Внутри ограниченного read-only workflow агент способен постепенно взять на себя целый операционный сегмент:

- поддерживать цель диагностической сессии;
- задавать человеку только необходимые физические или идентификационные вопросы;
- запускать допустимый маршрут общего сканирования;
- ждать подтверждённого окончания процесса;
- обнаруживать все проблемные системы, включая элементы ниже видимой области;

- последовательно открывать ещё не проверенные блоки;
- считывать коды неисправностей;
- контролировать полноту покрытия;
- формировать отчёт с доказательствами;
- останавливаться при неизвестном или опасном состоянии.

Если система выполняет все эти микрорешения самостоятельно во множестве комбинаций, это уже не «подсказка специалисту» и не один удачный клик. Это реальная автоматизация части профессионального труда.

Человек при этом не обязательно исчезает. Он перемещается на другой уровень: определяет цель, утверждает правила безопасности, разбирает исключения, проверяет спорную разметку, принимает решение о расширении зоны автономности и несёт ответственность за организацию процесса.

По мере накопления новых трасс доля стандартных случаев может расти, а специалист будет всё чаще работать с длинным хвостом исключений. Но именно в этом хвосте нередко сосредоточена самая высокая цена ошибки. Поэтому зрелая система не скрывает границу своей компетентности, а умеет её обнаруживать.

Мерой прогресса становится не процент кнопок, нажатых без человека, а размер **доказанно безопасной области самостоятельной работы**.

Методология за пределами автосервиса

Автомобильная диагностика удобна как наглядный пример. Здесь есть физическое оборудование, закрытое приложение, множество состояний, повторяемые workflow и очевидная цена неправильного действия.

Но сама методология значительно шире.

Аналогичный конвейер можно строить для лабораторных приборов, промышленных панелей, складских терминалов, систем контроля качества, медицинских административных приложений, юридических рабочих сред, ERP и другого профессионального программного обеспечения.

Для этого процесс должен обладать несколькими свойствами:

1. Его состояние можно наблюдать и фиксировать.
2. Значимые действия можно описать семантически и ограничить полномочиями.
3. Результат шага оставляет проверяемое свидетельство.

4. Неопределённость допускает остановку или передачу человеку.
5. Запись профессиональной работы возможна на законном основании и с приемлемой защитой данных.

Иногда первым полезным продуктом окажется не автономный исполнитель, а recorder. Иногда — агент контроля качества. Иногда — система, которая сравнивает действия оператора с лучшей практикой. Иногда — Shadow-агент, который месяцами учится на реальной работе, не вмешиваясь в неё.

Это важное изменение логики внедрения. Организации не обязательно начинать с вопроса: «Какой чат-бот нам купить?»

Гораздо продуктивнее спросить:

> **Как превратить наш реальный процесс в наблюдаемую, версионированную и проверяемую среду, в которой агенты смогут сначала учиться, затем советовать и только потом действовать?**

Европейское право становится частью архитектуры

GDPR

Цель, минимизация данных, privacy by design.

AI ACT

Прозрачность, AI literacy и риск-ориентированная архитектура.

DATA ACT

Доступ к данным connected products и aftermarket-сервисам.

Фотографирование опыта — это также обработка данных.

Диагностические скриншоты могут содержать VIN, идентификаторы оборудования, время визита, сведения о состоянии автомобиля, имена операторов или другие данные, которые прямо либо косвенно позволяют связать запись с человеком. Поэтому первичные трассы нельзя автоматически считать безопасным техническим материалом.

Принципы GDPR требуют заранее определить цель обработки, ограничить собираемые данные необходимым объёмом, установить срок хранения и встроить защиту данных в саму архитектуру. Практически это означает локальное или контролируемое хранение

первичных evidence, разделение идентификаторов и исследовательских ссылок, ограничение доступа, отсутствие реальных трасс в публичных репозиториях и отдельные обезличенные материалы для публикации.[4]

AI Act добавляет ещё два важных слоя. Организация должна обеспечивать достаточную AI-грамотность людей, которые эксплуатируют систему, а пользователи интерактивного AI должны понимать, что взаимодействуют с машиной. С 2 августа 2026 года применяются требования статьи 50 о прозрачности определённых AI-систем и AI-контента. Для текста на общественно значимую тему предусмотрено исключение при человеческой проверке или редакционном контроле и наличии лица, которое несёт редакционную ответственность; тем не менее прозрачное указание роли AI остаётся разумной практикой.[5][6]

Для автомобильной отрасли важен и Data Act: европейские разъяснения отдельно рассматривают доступ к данным connected vehicles и отношения между производителями, пользователями и aftermarket-сервисами. При переходе от лабораторного прототипа к коммерческому продукту эти вопросы нельзя оставлять «на потом».[7]

При этом нельзя автоматически объявлять любой диагностический AI «high-risk» или, наоборот, гарантировать, что он таким не является. Классификация зависит от intended purpose, реальной функции системы, связи с безопасностью продукта и конкретного режима применения.

Право здесь не внешний тормоз для готовой технологии. Оно влияет на то, какие данные записываются, где они хранятся, кто может дать системе разрешение, как объясняется её роль и какое доказательство остаётся после каждого решения.

Новая единица производства

Самый важный результат этого проекта для меня связан не с автомобилями.

Он показывает новую единицу организации инженерной работы.

Раньше команда вручную писала программу, собирала датасет, формулировала правила, строила тесты и затем встраивала модель в готовую конструкцию. Теперь языковые и мультимодальные агенты могут участвовать почти в каждом звене:

- проектировать средства наблюдения;
- писать и проверять код;
- превращать человеческие действия в структурированные события;
- выявлять устойчивые паттерны;

- предлагать системные инструкции и контракты;
- генерировать пограничные случаи;
- воспроизводить исторические сессии;
- искать ошибки другого агента;
- сопровождать работу runtime-системы.

Это резко сокращает расстояние между наблюдением за специалистом и первой проверяемой автоматизацией его workflow.

Но слово «агент» не отменяет инженерную дисциплину. Чем больше этапов мы поручаем моделям, тем важнее происхождение артефактов, разделение ролей, замороженные тесты, детерминированные границы и человеческое решение о готовности.

Именно поэтому формула «агент строит агента» не означает, что человек вышел из процесса.

Она означает, что человек больше не обязан вручную создавать каждый промежуточный элемент. Его задача поднимается на уровень выше: определить, какой опыт нужно зафиксировать, что считается истиной, какие действия недопустимы, какие доказательства достаточны и кто отвечает за переход от эксперимента к производству.

Будущее производственного AI, вероятно, придёт не в виде одного универсального цифрового сотрудника. Оно будет похоже на агентную фабрику, где одни системы наблюдают, другие структурируют, третьи проектируют, четвёртые проверяют, а пятые получают строго ограниченное право действовать.

> **Агент строит агента. Доказательства строят доверие. А право действовать по-прежнему создаёт человек — через архитектуру, правила и ответственность.**

Примечание автора

При подготовке структуры и редакционной версии текста использовались инструменты AI. Автор проверяет факты, выводы и окончательную публикационную версию и несёт за неё редакционную ответственность.

Описанный проект находится в исследовательской и поэтапно валидируемой стадии. Материал не заявляет сертификацию безопасности, соответствие конкретному классу AI Act или готовность к автономному управлению функциями автомобиля.

Источники

1. Brown, T. B. et al. *Language Models are Few-Shot Learners*, 2020.

   arxiv.org/abs/2005.14165

2. Ahn, M. et al. *Do As I Can, Not As I Say: Grounding Language in Robotic Affordances*, 2022.

   arxiv.org/abs/2204.01691

3. Brohan, A. et al. *RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control*, 2023.

   arxiv.org/abs/2307.15818

4. Regulation (EU) 2016/679 (GDPR), Articles 5 and 25.

   eur-lex.europa.eu/eli/reg/2016/679/oj/eng

5. Regulation (EU) 2024/1689 (AI Act), Articles 4 and 50.

   eur-lex.europa.eu/eli/reg/2024/1689/oj/eng

6. European Commission, *Guidelines on Transparency of AI-Generated Content*, 2026.

   digital-strategy.ec.europa.eu/en/policies/guidelines-transparency-ai-generate...

7. European Commission, *Guidance on vehicle data, accompanying the Data Act*, 2025.

   digital-strategy.ec.europa.eu/en/library/guidance-vehicle-data-accompanying-d...