

12+
СЕРГЕЙ КИРНИЦКИЙ

Золотая лихорадка ИИ-агентов

ДЕЛЕГИРОВАНИЕ
НЕОПРЕДЕЛЁННОСТИ



Сергей Кирницкий

**Золотая лихорадка ИИ-агентов.
Делегирование неопределённости**

«Издательские решения»

Кирницкий С.

Золотая лихорадка ИИ-агентов. Делегирование неопределённости /
С. Кирницкий — «Издательские решения»,

Плоская подписка делает ночной прогон ИИ-агента как будто бесплатным. Но неопределённость, отданная вечером, возвращается утром: ревью, которое нельзя закончить, переделкой, кодом, который страшно трогать. Решения приняты — но никем. Эта книга для тех, кто ставит задачи людям и машинам: о том, почему полтора часа ясной мысли стоят больше тридцати одного часа ночной работы и почему полномочия можно делегировать, а ответственность — нет.

Содержание

Часть I. Сутки	6
Глава 1. Утро после ночной смены	6
1.1. Одна фраза вечером	6
1.2. Ночь	8
1.3. Diff	10
1.4. Над чем он работал	13
Глава 2. Длительность как признание	15
2.1. Словарь самообмана	15
2.2. Чёткая задача, которая идёт сутки	17
2.3. Кажущееся исключение: объём	20
2.4. Названная неопределённость	22
2.5. Способность и смысл	24
Глава 3. Лихорадка	27
3.1. Калифорния, 1849	27
3.2. Весна 2023: лихорадка ИИ-агентов	29
3.3. Роли вместо решений	31
3.4. Лоток с сеткой	33
3.5. Продавцы лопат	35
Конец ознакомительного фрагмента.	38

Золотая лихорадка ИИ-агентов Делегирование неопределённости

Сергей Кирницкий

Иллюстрация на обложке С использованием ChatGPT

© Сергей Кирницкий, 2026

ISBN 978-5-0071-4142-0

Создано в интеллектуальной издательской системе Ridero

Часть I. Сутки

Глава 1. Утро после ночной смены

1.1. Одна фраза вечером

Фраза пишется за несколько секунд: «переведи биллинг на подписки, сделай нормально». Человек, который её набирает, сидит за кухонным столом в двенадцатом часу. День кончился: письма клиентов разобраны, последний звонок был два часа назад, за окном давно темно, в квартире тихо, только гудит холодильник, и до сна остаётся ровно столько, сколько нужно на ещё одно дело. Ради него одного и открыт ноутбук. Написанное он не перечитывает. Отправляет — и с этой секунды задача уходит из его рук. Ещё минуту он смотрит, как на экране бегут первые строки ответа, потом опускает крышку, та закрывается с мягким щелчком, и он встаёт из-за стола с приятным чувством, что работа пошла.

Эти первые строки — уже работа. Они короткие и деловые: какой файл открыт, какая команда запущена, что она ответила, — и сменяют друг друга быстрее, чем их можно прочесть. По ту сторону фразы — ИИ-агент для программирования, и начал он с того же, с чего начал бы опытный программист, пришедший в чужой проект: нашёл в репозитории каталог с платёжным кодом, открыл файл, где создаются счета, прочитал его и запустил тесты, чтобы понять, что сейчас работает, а что нет. Его не надо вводить в курс дела и объяснять, как устроен код. Всё необходимое он найдёт и прочтает сам, а в его распоряжении весь репозиторий, не один платёжный каталог. Закрытая крышка ему не помеха: у него своя копия кода где-то в облаке, и ноутбук на столе больше не нужен. Дальше он будет делать то же самое, только без перерыва: читать файлы и править их, запускать команды и тесты, фиксировать сделанное коммитами и снова читать — шаг за шагом, столько шагов, сколько понадобится, никого ни о чём не спрашивая. Задачу ему дали словами, и других слов не будет. Человек, который их написал, уже гасит свет.

Этого человека не существует. История вымышлена и составлена из многих вечеров — тех, о которых нам рассказывали, и тех, которые мы провели сами, — поэтому у героя нет ни имени, ни названия продукта. Каждую его черту мы видели у живых людей, но все вместе они не принадлежат никому. Известно о нём немного: он основатель небольшого SaaS, сервиса, в котором клиенты работают прямо в браузере и за который платят, и он переводит биллинг на подписки. Узнать в нём можно кого угодно: бывшего коллегу, соседа по офису, себя. Мы себя узнавали. Мы тоже оставляли ИИ-агентов на ночь с одной короткой фразой, тоже закрывали ноутбук, не дождавшись первого коммита, и тоже вставали из-за стола с этим приятным чувством.

Задержимся на самой фразе — на той минуте, когда крышка ещё открыта и курсор стоит после последнего слова. Шесть слов и одна запятая — про ту часть продукта, через которую проходят все его деньги. Их не пришлось подбирать: между словами не было пауз, рука ни разу не потянулась стереть одно и поставить другое. И это не небрежность. Основатель знает свой продукт как никто: первые платежи в нём он когда-то написал сам, крупных клиентов помнит по именам и без запинки объяснит, за что они ему платят. Так ставят задачу почти все — не расписывая, одним словом давая понять, что халтура не нужна. Мы сами десятки раз писали это слово ИИ-агентам и людям, писали и слова той же породы — «по-человечески» и «как положено», — и ни разу не заметили, что за ними осталось недосказанное.

Слово «нормально» поставлено в конец фразы не для красоты. Оно значит: без временных заплаток, так, чтобы потом не переделывать и чтобы код не стыдно было показать коллеге. Между инженерами оно звучит строго, почти как проверка на профессионализм, и тот, кто его пишет, чувствует, что поднимает планку. Задачу это слово не уточняет. Им её заканчивают, как подписью заканчивают письмо, и оно создаёт ощущение, что задача описана целиком: переведи и сделай хорошо — что тут ещё скажешь. Покажи эту фразу любому инженеру, и он, скорее всего, кивнёт: что сделать — понятно, планка названа. Мы бы кивнули сами.

Но в слове «нормально» спрятан целый несделанный мир. Если его развернуть, внутри окажется по меньшей мере три вопроса, и ни один из них не уточняется мимоходом: за каждым — деньги живых клиентов и то, как эти люди будут относиться к продукту завтра.

Первый — какие вообще бывают состояния подписки. Со стороны кажется, что их два: клиент платит или не платит. Но достаточно проследить за одним клиентом хотя бы год, и состояния начинают множиться. Вот он пробует продукт бесплатно две недели — это уже подписка или ещё нет? Вот пробные две недели прошли, а карту он так и не привязал. Вот заплатил. Вот решил уйти, но оплаченный месяц ещё идёт, и он продолжает работать: он клиент или уже бывший? Вот перешёл на тариф подороже посреди месяца. Вот попросил приостановить всё на лето. Вот вернулся через полгода, и непонятно, продолжается прежняя подписка или начинается новая. Каждое такое «вот» либо становится отдельным состоянием, либо нет, и для каждой пары состояний нужно понять, можно ли перейти из одного в другое и что при этом происходит с деньгами. Состояние, о котором никто не подумал, никуда не девается: однажды оно просто наступает, и продукт ведёт себя в нём как придётся. Это похоже на мелкую бухгалтерию. На деле от этого зависит, какую сумму человек увидит в выписке и не почувствует ли он себя обманутым.

Второй — что происходит, когда списание не проходит. Первого числа банк отклоняет карту клиента, который платит уже третий год. Причина может быть любой: карта истекла, на счёте не хватило денег, банк заподозрил неладное, — а продукту нужно действовать сразу. Закрывать доступ немедленно значит, что завтра у человека, с которым три года всё было хорошо, посреди рабочего дня не откроется привычный инструмент. Не закрывать — но тогда на сколько: на день, на неделю, до конца месяца? Как сказать клиенту, что случилось: письмом, строкой в самом продукте или никак? И какими словами? А если через два дня он привяжет новую карту, за какие дни с него брать деньги? И заслуживает ли тот, кто платит три года, больше терпения, чем тот, кто пришёл вчера? Технически в этих вопросах нет ничего трудного. Трудно другое: каждый ответ — обещание, которое продукт даёт человеку в самый неприятный для того момент.

Третий вопрос тише остальных: где кончается биллинг и начинается авторизация. О первых двух рано или поздно напомнят сами клиенты — письмом или отказом платить. О третьем они не напишут никогда: снаружи видно только, пустили их или нет, но не видно, чей код это решил. Биллинг знает, кто сколько заплатил. Авторизация решает, кого и куда пускать. Пока подписок не было, эти две части продукта почти не встречались; теперь им придётся договариваться, потому что подписка обещает клиенту то, что входит в его тариф, и кто-то должен проверять, положено ли ему то, что он сейчас пытается сделать. Тариф рассчитан на троих сотрудников, а клиент приглашает четвёртого. Кто ему откажет и в какой момент: когда он нажимает кнопку или когда приходит счёт? Непонятно, чья это работа: кода, который считает деньги, или кода, который проверяет права. Непонятно, кто закрывает дверь, когда подписка кончилась: биллинг сам или он лишь сообщает об этом, а закрывает авторизация. И совсем непонятно, можно ли ради подписок трогать модуль, через который каждый клиент каждый день входит в продукт. Такую границу не найти ни в одном файле. Её можно только провести.

Эти три вопроса к тому же не лежат порознь. Клиент, у которого не прошла карта, — это новое состояние подписки или прежнее, в котором что-то пошло не так? Если закрывать

ему доступ, то закрывать будет неизвестно чей код. А сколько терпения ему положено, зависит от того, в каком состоянии он окажется. Стоит потянуть за один вопрос, и за ним потянутся два других.

Ни на один из этих вопросов у него нет ответа, и именно поэтому фраза так легко пишется.

Он не откладывает их на потом — он их не видит. Изнутри слова «нормально» их не разглядеть: оно звучит так, будто ответы на все три уже есть и проговаривать их незачем. Спроси его кто-нибудь в эту минуту, что будет с клиентом, у которого не прошла карта, он ответил бы — подумав, толково, может быть, даже хорошо. Спроси его завтра — ответил бы, возможно, иначе и тоже толково. Но такой ответ родился бы только в момент вопроса. До вопроса его нет, а спрашивать сейчас некому.

Когда мы знаем, что чего-то не знаем, это мешает: вопрос возвращается, цепляет, не даёт додумать соседнюю мысль. Незнание, которое не видно, не ощущается как пробел. Оно ощущается как лёгкость. Все мы помним, какие тексты пишутся трудно: письмо клиенту с отказом, объяснение, почему сорвался срок. Трудно их писать потому, что каждое слово в них что-то утверждает и его приходится взвешивать. Будь у основателя ответ хотя бы на один из трёх вопросов, вечерняя фраза стала бы таким текстом: в ней появились бы числа и оговорки, и каждую пришлось бы сначала сформулировать для себя. Ответы весят. Шести словам нести нечего.

Поэтому и чувство, с которым он встаёт из-за стола, неподдельное. Задача, ещё днём лежавшая на нём, теперь переложена на другого, и от этого ему легко почти физически, как бывает, когда с плеч сняли тяжёлую сумку. Держать в голове больше нечего: всё трудное уместилось в одно слово, а слово отправлено. Вопросы ушли вместе с этим словом — невидимые, похожие на ответ. В квартире всё так же тихо, и на кухне от них не осталось ничего, кроме приятного чувства, с которым опустила крышка ноутбука. Эта лёгкость и есть их след.

1.2. Ночь

Лёжа в темноте, основатель первым делом убавляет яркость телефона: экран светит так, что больно смотреть. Потом открывает панель прогона. Это та же работа, что вечером бежала строками по экрану ноутбука, только собранная в несколько чисел и один длинный список. Сверху — номер итерации, и, пока он смотрит, номер меняется: ИИ-агент закончил очередной шаг — открыл файл, поправил в нём несколько строк, запустил тесты — и взялся за следующий. Рядом — счётчик токенов. Первые цифры в нём стоят спокойно, последние мелькают так быстро, что их не прочесть, а если на пару секунд отвести глаза, всё число уже другое. Ниже идут коммиты, у каждого — короткая строка о том, что сделано, и зелёная галочка справа: тесты прошли. Новые коммиты появляются сверху и сдвигают прежние вниз. Их пока немного, и один прибавляется прямо при нём. В квартире не движется ничего, кроме цифр на экране.

Он листает список, хотя лёжа, одной рукой, это неудобно, и нажимает на один из коммитов. Тот раскрывается в цепочку шагов: какие файлы были прочитаны, прежде чем в коде поменялась хоть одна строка, какая команда запущена и что она ответила, какие тесты прошли. Перед тем как поправить одну-единственную функцию, ИИ-агент прочитал всё, что её вызывает, — так работает внимательный инженер, которому не лень. Основатель и сам работал бы так же, будь у него на это время. В строках коммитов — слова его продукта: подписка, тариф, счёт, продление. Этими словами он годами говорил с клиентами, сам когда-то расставлял их по кнопкам и письмам, а теперь их пишет не он: пока он лежит под одеялом, в его коде с ними работает ИИ-агент. Удовольствие смотреть на работу, которую делаешь не ты, ни на что не похоже. С отдыхом его не спутаешь: отдыхая, о работе не думают, а он не может думать ни о чём другом. С бездельем тоже: работа его, задача его, продукт, в котором сейчас меняется

код, тоже его. Не его здесь только руки. Его дело движется без него — и так быстро, как не двигалось никогда, пока он делал его сам.

Вечернее чувство, что работа пошла, здесь, в темноте, вырастает в другое, крупнее. Обычно его день кончается там, где кончаются силы: что не успел, то ждёт до завтра, и ночью продукт замирает, как замирает по ночам всякая работа, которую делают люди. Сегодня день не кончился. Он лежит, а дело идёт дальше, и ночь за окном ничего в нём не меняет. Как будто в его сутках стало больше часов, чем у всех остальных людей, и лишние часы работают на него. Он пробует прикинуть, сколько времени ушло бы у него самого на то, что уже лежит в списке, и бросает считать: выходит слишком много. ИИ-агент, который трудится всю ночь, — доказательство силы. Проверить это доказательство можно в любую минуту, достаточно взять телефон. Сила эта его. Она не устаёт, не отвлекается и не ложится спать, и он владеет ею так же просто, как телефоном у себя в руке.

Телефон он откладывает на тумбочку, уже засыпая. В четвёртом часу он просыпается без всякой причины, как просыпаются перед ранним поездом, и, не успев толком открыть глаза, тянется к тумбочке. Панель живая. Номер итерации ушёл далеко вперёд, в счётчике токенов прибавился разряд, сверху в списке стоят коммиты, которых он ещё не видел, и у каждого та же зелёная галочка. Он смотрит на панель недолго, кладёт телефон экраном вниз и засыпает снова, быстро и спокойно, как в детстве засыпал под негромкие голоса взрослых на кухне.

Эта ночь продлится тридцать один час — ночь, день и ещё одну ночь. Она началась за кухонным столом в двенадцатом часу и не кончается с рассветом, хотя за окном уже сереет. В неё уместятся восход и закат, целый рабочий день с письмами и звонками и ещё одна ночь с телефоном на тумбочке, а на панели всё это время будет идти работа, без перерыва на сон и на обед, без пауз длиннее нескольких секунд. Тридцать один час — больше суток. Если разложить его на обычные рабочие дни по восемь часов, выйдет почти четыре, поставленных вплотную один к другому. Ни один человек не просидит над кодом столько, не вставая из-за стола, и он сам даже в самые горячие недели перед первым запуском своего продукта не выдерживал и половины. Такое число не округляют. Его произносят целиком.

Днём у него своя работа, и её не меньше, чем всегда: письма клиентов, звонки, встреча на другом конце города. Панель прогона живёт в кармане. Он открывает её между двумя звонками, в лифте, в такси, посреди разговора, когда собеседник отвлекается на свой телефон. Движение каждый раз одно и то же — достать, разблокировать, взглянуть, — и каждый раз на панели что-то новое. Телефон в кармане весит как обычно, но ощущается иначе: в нём идёт работа. Номер итерации он уже не запоминает: к следующему взгляду тот всё равно уйдёт вперёд. Число в счётчике токенов он давно не пытается прочесть целиком, ему хватает того, что оно всякий раз больше прежнего. По верхней строке видно, в каком файле сейчас ИИ-агент и что он там делает — читает, правит или ждёт, пока пройдут тесты, — и основатель, узнавая знакомые названия, мысленно идёт за ним по своему коду, как по дому, где знает каждую комнату. Иногда он застаёт ИИ-агента посреди шага: тесты запущены, и зелёные отметки встают одна под другой, пока не зазеленеет весь столбец. Иногда новый коммит ложится наверх списка у него на глазах, и это отдельная маленькая радость, как будто успел к самому интересному. Панель он открывает чаще, чем почту, и понятно почему: почта приносит просьбы и жалобы, панель — только сделанную работу.

Около полудня звонит клиент из тех, кто давно ждёт подписок, и спрашивает, когда они появятся. Раньше на такой вопрос приходилось отвечать осторожно, прикидывая, когда дойдут руки. Сегодня основатель просто говорит: «Уже делается». Пока они разговаривают, он открывает панель, и сверху в списке появляется новый коммит, так что это буквально правда: работа идёт в ту самую минуту, когда он о ней говорит. За обедом он показывает телефон знакомому, тоже основателю небольшого продукта, и тот, пролистав список, присвистывает.

Когда за окном темнеет, он снова садится за кухонный стол и открывает панель уже на ноутбуке, чтобы увидеть всё сразу. Позади без малого сутки. За день у него сел голос от звонков и гудят ноги, телефон дважды пришлось ставить на зарядку, а на панели всё идёт с той же ровной скоростью, что и в первый час. Даже на большом экране список коммитов не помещается и уходит за нижний край, так что до первого, вчерашнего, приходится долго листать. Он проходит его снизу вверх, от первого коммита к последнему, и сутки разворачиваются перед ним ещё раз. Вот это сделано, пока он спал, это — пока ехал на встречу, это — пока обедал и показывал телефон знакомому. Он находит коммит, появившийся, пока он говорил с клиентом, и задерживается на нём с улыбкой. Каждому его часу соответствует работа, которой он не делал. Зелёный столбец не прерывается ни разу.

Мы знаем это чувство изнутри. Мы сами просыпались среди ночи и тянулись к телефону проверить, идёт ли работа, открывали панель в очереди к врачу и под столом на чужом совещании, делали снимки экрана с особенно длинным списком коммитов и отправляли их друзьям, ожидая, что те присвистнут. И сами говорили, небрежно и с плохо скрытой гордостью: «Он у меня работает вторые сутки». Первую такую ночь мы помним до сих пор: до неё сутки непрерывной работы были чем-то из чужих рассказов, после неё стали нашими. Счётчик токенов растёт, коммиты идут, тесты зеленеют, и всё это без нас, днём и ночью, без усталости. Кто хоть раз оставлял ИИ-агента на ночь, помнит это ощущение телом: тепло телефона в ладони, голубоватый свет в тёмной комнате, короткий толчок радости, когда сверху в списке появляется новая строка. Это одно из самых сильных чувств, какие даёт наше ремесло: видеть, как большая работа идёт сама, и знать, что она твоя.

Во вторую ночь основатель ложится позже обычного: всякий раз хочется дожидаться ещё одного коммита. Телефон снова в руке, и панель всё так же жива: номер итерации меняется, последние цифры счётчика мелькают, сверху появляются новые строки. Напротив новых коммитов — те же зелёные галочки, а ниже ещё и ещё, до самой первой строки. Одна сплошная зелёная полоса. Он смотрит на неё дольше, чем нужно, просто потому, что на неё приятно смотреть. В истории репозитория эти сутки с небольшим уже занимают больше места, чем несколько последних месяцев. Там, в облаке, в его коде вторую ночь подряд идёт работа, какой в его продукте не было никогда: большая, непрерывная, начатая им и сделанная не им. Каждая проверка панели — в темноте спальни, между двумя звонками, за кухонным столом — подтверждала одно и то же. Подтверждает и теперь, когда глаза у него уже закрываются, а палец ещё лежит на экране: что-то большое происходит.

1.3. Diff

Цифры на панели больше не бегут. Основатель замечает это раньше, чем успевает толком проснуться: номер итерации не меняется, последние знаки счётчика токенов стоят на месте, и сверху в списке коммитов не появляется ничего нового, сколько на него ни смотри. В комнате уже светло, за стеной соседи собираются на работу, голова тяжёлая после двух коротких ночей подряд. Верхняя строка, которая всё это время показывала, где сейчас идёт работа, теперь говорит одно: прогон завершён. Кончился он на рассвете, пока основатель спал. Под этой строкой — последнее сообщение ИИ-агента, несколько спокойных абзацев, похожих на записку аккуратного коллеги перед отпуском: что сделано, где что лежит, как проверить. Заканчивается записка тем, что все тесты проходят. А ниже — ссылка, которую за всё это время не открывали ни разу: diff, все изменения, внесённые прогоном в код, собранные в одном месте, файл за файлом и строка за строкой, — что было и что стало. Рядом со ссылкой стоит число: двести четырнадцать изменённых файлов.

Он открывает diff, не вставая с постели. На телефоне это узкая лента имён файлов, и у каждого два числа: сколько строк добавлено, сколько удалено. Он гонит ленту большим паль-

цем вверх, потом ещё раз и ещё, а она не кончается, и палец устаёт раньше, чем глаза успевают что-нибудь понять. Так двести четырнадцать файлов не прочесть. Он встаёт, забирает с кухни ноутбук и садится с ним в комнате, за стол у окна. Солнце уже добралось сюда: широкая полоса лежит поперёк столешницы, угол экрана слепит, и ноутбук приходится развернуть. Ползунок прокрутки справа сжимается в короткую чётточку, какой он бывает только в очень длинных документах, и просто пролистать список от первого файла до последнего, ничего не открывая, — дело нескольких минут. Добавленные строки отмечены зелёным, удалённые — красным. Это первый красный цвет за тридцать один час, и означает он не ошибку, а строки, которых в коде больше нет. Красного много.

Он листает. Файлы идут вперемешку, знакомые и незнакомые. Вот файл, где создаются счета: когда-то основатель написал его сам, позавчера с него начал ИИ-агент, и от прежних строк в нём осталось немного. Вот шаблоны писем клиентам — об оплате, о продлении и о том, что списание не прошло; последнего раньше не было вовсе. Вот страница, где клиент видит свою подписку, — её тоже не было. Вот файл, который должен разом перевести на подписки всех нынешних клиентов. Вот новый каталог, которого до прогона в репозитории не существовало, а в нём — собственный конечный автомат подписки, написанный с нуля, со своими состояниями и переходами. Вот модуль авторизации, и в нём тоже правки. Вот тесты, целые экраны тестов, новых и переписанных, и все они теперь проходят. Каждый файл сам по себе невелик. Вместе они как коробка, которую подхватил, не прикинув, что в ней: тяжесть доходит до рук раньше, чем до головы.

Конечный автомат он открывает первым. Сверху — состояния: «пробный период», «активна», «платёж не прошёл», «отменена, но оплачена до конца срока», «закрыта». Ниже — переходы: из какого состояния в какое можно попасть, что при этом происходит с доступом и что с деньгами, и у каждого перехода пара строк о том, почему так. Код чистый, его приятно читать. Вопросы, которых позавчера основатель не видел, встречаются его здесь уже с ответами. Кто так и не привязал карту, незадолго до конца пробных двух недель получает напоминание, а когда они выходят, оказывается в состоянии «закрыта». Клиент, решивший уйти посреди месяца, работает до последнего оплаченного дня, а потом подписка закрывается сама. Паузы на лето нет: такого состояния в списке просто не существует. Кто вернулся через полгода, начинает новую подписку, а старая так и остаётся закрытой. У кого не прошла карта, тот ещё семь дней работает как ни в чём не бывало, видит сверху экрана надпись с просьбой обновить карту, получает письмо, а на восьмой день подписка закрывается. Семь дней — для всех поровну: и для того, кто платит третий год, и для того, кто пришёл вчера.

Правки в модуле авторизации занимают совсем немного места. Но этот модуль основатель знает наизусть: через него каждое утро проходят все его клиенты. Теперь при каждом входе модуль спрашивает у конечного автомата, в каком состоянии подписка: прежде чем открыть человеку его же данные, продукт узнаёт, заплатил ли тот. Если подписка закрыта, клиент может войти и посмотреть всё, что успел сделать, но не может ничего создать или изменить; рядом в пояснении сказано, что так человек не остаётся отрезанным от собственной работы, и спорить с этим трудно. Если на тарифе для троих сотрудников приглашают четвёртого, отказ приходит в ту же секунду, когда нажата кнопка: вежливая строка о том, что тариф рассчитан на троих, и предложение перейти на тариф побольше. У каждой правки есть свой тест и своя строчка о том, зачем она. Граница, которой не было ни в одном файле, проведена — и сразу в нескольких.

Тесты он оставляет напоследок. Многие ему знакомы: часть он писал сам, ещё когда платежи были разовыми. Прежними они не остались. В одних поменялись ожидаемые значения, другие переписаны заново, нескольких нет совсем, и на их месте стоят новые. Был среди старых тест, в названии которого он когда-то сделал опечатку и так и не собрался её исправить; теперь внутри этого теста всё другое, а опечатки в названии больше нет. Все проходят. Он

смотрит на эти файлы дольше, чем на остальные, и в какую-то минуту вспоминает зелёную полосу, на которую засматривался две ночи подряд. От этой мысли тянет холодком, и он её не додумывает. Листает дальше.

Следующие часы он ищет ошибку. Читает так, как читают работу чужого инженера, прежде чем пустить её в свой продукт: медленно, с замечанием наготове, выискивая место, где можно написать «здесь не так», — пропущенный случай, неверную сумму, тест, который проходит случайно. Первым делом проверяет то, чего опасается больше всего: что станет в первое же утро с теми, кто платил по-старому, не окажутся ли они перед закрытой дверью. Не окажутся. Файл переводит их всех в состояние «активна» до конца уже оплаченного срока и каждому отправляет письмо о том, что меняется. Потом основатель запускает продукт у себя, в тестовом окружении, и за полчаса проживает целый год из жизни одной подписки: заводит клиента, даёт пробному периоду истечь, привязывает карту, которую тестовый банк отклоняет, пробует пригласить четвёртого сотрудника. Продукт ведёт себя ровно так, как написано в коде: надпись вверху экрана, семь дней, вежливый отказ. Письмо о неудачном списании приходит сразу, короткое и спокойное, написанное почти его словами. Нигде ничего не падает. Он ловит себя на том, что был бы рад хоть одному красному тесту. Ни одно место не выглядит небрежным. Так мог бы сделать толковый инженер. Может быть, так сделал бы и сам основатель. Этого он не знает. Ухватиться не за что. Каждая деталь по отдельности разумна, и это пугает сильнее, чем явная ошибка.

Принять результат не получается ни за час, ни за день: сверять его не с чем. Мешает не объём. Рядом с diff нечего положить, кроме вечернего «нормально», а оно не говорит ни «да», ни «нет».

И день идёт. Полоса солнца сползает со стола, и к полудню в комнате стоит ровный свет без теней. Встречу, назначенную на три, основатель переносит на завтра письмом в две строки. Шея затекает, от белого фона режет глаза, а он уже в третий раз открывает конечный автомат и читает его снизу вверх, будто в обратном порядке тот скажет что-то другое. Приходит письмо от клиента, который вчера спрашивал, когда появятся подписки: ну что, уже? Основатель пишет в ответ «готово» и стирает. Пишет «почти» и стирает тоже. Письмо остаётся без ответа.

После обеда он выходит пройтись вокруг дома, чтобы проветрить голову, и скоро возвращается: на улице думается о том же самом, только без экрана перед глазами. На краю стола листок, и к этому часу на нём полстраницы вопросов, которые начинаются одинаково: почему семь дней, почему поровну для всех, почему паузы нет. На каждый у него находится только встречный — а как надо? — и на встречный ответа нет тоже. Телефон с утра лежит экраном вниз: панель в нём больше ничего не покажет. Берёт он его только раз, когда звонит знакомый, вчера присвистнувший над списком коммитов, и спрашивает, чем кончилось. «Кончилось», — отвечает основатель и не находит, что прибавить. Кнопка, которой изменения принимают в основную ветку, весь день ждёт в углу экрана; нажать её — одно движение пальца. Он её не нажимает. В сумерках он зажигает настольную лампу, и там, где утром лежала полоса солнца, теперь жёлтый круг, а в нём листок с вопросами. К вечеру этот код он знает почти так же хорошо, как свой, и всё так же не может сказать ни «да», ни «нет».

Такие дни мы просиживали и над своими diff — с тем же ползунком, сжатым в чёрточку, с тем же третьим прочтением одного файла, с тем же письмом, которое никак не отправить. И так же, как он, мы не смогли бы объяснить постороннему, что нас держит: всё ведь работает.

Ближе к ночи меняется то, как он смотрит на экран. Утром перед ним была чужая работа, которую нужно проверить. Теперь он видит в ней свой завтрашний день. Если нажать кнопку, то однажды первого числа, когда у того, кто платит третий год, не пройдёт карта, продукт даст ему семь дней — ровно столько же, сколько вчерашнему новичку. Клиент напишет не ИИ-агенту, а основателю, и объяснять, почему семь дней, а не месяц, и почему поровну для всех, придётся ему, как объясняют своё. Когда позвонят с жалобой на отказ четвёртому сотруднику,

основатель скажет «так у нас устроено» — о правиле, которое сам узнал сегодня утром. Письмо с вопросом «ну что, уже?» всё ещё открыто в соседнем окне. Ответить на него «готово» — значит поставить свою подпись под всеми двумястами четырнадцатью файлами. Каждая строчка на этом экране станет обещанием, которое его продукт даёт людям, и отвечать за каждое будет он.

Человек держит в руках решения, которых не принимал.

1.4. Над чем он работал

Ответ на вопрос, над чем тридцать один час работал ИИ-агент, кажется очевидным, и ещё вчера его, не задумываясь, дал бы кто угодно: над биллингом. Так было сказано в первых же словах задачи, об этом говорили строки коммитов — подписка, тариф, счёт, продление, — и в двухстах четырнадцати файлах нет ни одного, который не касался бы денег клиентов или того, кого куда пускать. Но вечерняя фраза состояла из двух половин. Первая, «переведи биллинг на подписки», называла, что сделать. Вторая, «сделай нормально», стояла на месте трёх ответов, которых у основателя не было, и в ночь она ушла пустой. Места в этой пустоте хватало на деньги всех его клиентов и на всё, что продукт им обещает. Снаружи её не было видно: фраза выглядела законченной, как письмо с подписью.

В коде такое место пустым остаться не может. Там нельзя написать «нормально»: где у клиента не прошла карта, должно стоять число дней, а где на тариф для троих приглашают четвёртого — момент, когда ему откажут. ИИ-агент не решал задачу про биллинг — он заполнял пустоту, оставленную вечерней фразой, правдоподобным содержимым. У пустоты была форма, форма трёх вопросов, и содержимое повторяет её в точности. Где вечером прятался вопрос о состояниях подписки, утром лежит новый каталог с конечным автоматом; где прятался вопрос о неудачном списании — семь дней и письмо; а на месте вопроса о границе биллинга и авторизации — готовая граница, проведённая правками в модуле авторизации.

Если смотреть на ночь отсюда, от вечерней фразы, каждый шаг на панели — открыл файл, поправил несколько строк, запустил тесты — укладывал в пустую половину фразы ещё немного, и к рассвету третьего дня места там не осталось. Пустота вернулась к основателю заполненной, и весь день он смотрел как раз на неё: туда, где должны были стоять его ответы, а стояли чужие.

Тридцать один час ушёл не на написание кода, а на выбор того, что должен был выбрать человек. Семь дней для того, у кого не прошла карта, и семь для всех поровну — для клиента третьего года и для вчерашнего новичка. Отказ четвёртому сотруднику в ту же секунду, когда нажата кнопка. Лето без паузы, и этот выбор не занял в diff ни одной строки. Закрытая подписка, при которой в свою работу можно смотреть, но ничего в ней нельзя тронуть. Каждый из этих выборов мог быть иным: четырнадцать дней или месяц, больше терпения тому, кто платит давно, отказ в день счёта, пауза до осени. Ни один не следовал из слова «нормально», ни один не был ошибкой, и каждый касался людей, которых основатель помнит по именам.

Чтобы код появился, кто-то должен был выбрать. Выбирал ИИ-агент, и опереться ему было не на что, кроме тех же шести слов. Выбирал на третьем часу и на двадцатом, пока основатель спал, звонил, ехал на другой конец города. Вечером за одним вопросом тянулись два других; ночью за одним выбором тянулись следующие. Раз в конечном автомате появилось состояние «закрыта», модулю авторизации понадобилось знать, что в этом состоянии можно, а чего нельзя, а тестам — чего ждать от модуля.

Ни в diff, ни на панели выбор ничем не выделялся. В diff семь дней — одна цифра в одной строке, и выглядит эта строка так же, как тысячи соседних, где переименована переменная или поправлен отступ. В записке, которой закончился прогон, семь дней числятся среди сделанного, в одном ряду с тем, где что лежит и как это проверить. Основатель думал, что

панель, в отличие от почты, приносит только сделанную работу. Выбор она приносила в том же виде: короткой строкой коммита с зелёной галочкой справа. Рядом со словами его продукта в этих строках были только слова работы: добавлено, исправлено, перенесено, обновлено. Слова «выбрано» не было ни в одной.

Может быть, семь дней клиенту третьего года были отпущены как раз в ту минуту, когда основатель говорил в трубку «Уже делается», и коммит, на котором он вчера в сумерках задержался с улыбкой, был именно этим. Может быть, нет: по строке в списке этого не узнать. Тогда же, проходя список снизу вверх, он находил напротив каждого своего часа работу, которой не делал, и радовался. Список с тех пор не изменился. Напротив каждого его часа стоит выбор, которого он не делал.

Ночное восхищение от этого не становится ошибкой. ИИ-агент работал всерьёз, внимательно, не прерываясь ни на минуту, и что-то большое и правда происходило. Не была ошибкой и утренняя растерянность. Легко принять их за реакции на две разные вещи — на силу в телефоне и на чужой код в diff. Но вещь была одна, и её увидели дважды. Список коммитов, уходящий ночью за нижний край экрана, и ползунок, сжатый утром в короткую чёрточку, показывают одну и ту же длину. Телефон, который основатель две ночи не выпускал из рук, почти весь день пролежал экраном вниз, хотя в нём всё тот же список. То большое, что тридцать один час подтверждала панель, и есть решения, которых человек не принимал. Ночью панель показывала их размер, утром diff показал их содержимое.

В первую ночь ему казалось, что в его сутках стало больше часов, чем у всех остальных людей, и лишние часы работают на него. Лишние часы были. Уходили они на то, чтобы выбрать за него. Номер итерации, разряд, прибавившийся в счётчике токенов к четвёртому часу, зелёная полоса от первой строки до последней — всё это мерило одно: сколько пришлось выбрать там, где вечером осталась пустая половина фразы. Он владел силой, не знавшей ни усталости, ни сна, держал её в руке вместе с телефоном, и сила эта была настоящей. Сила, которой любовались ночью, оказывается мерой того, чего владелец задачи не знал.

Мы так же любовались и так же сидели — и долго не складывали одно с другим. Ночь помнилась отдельно, утро отдельно, как два разных случая из жизни ремесла: в одном мы гордились инструментом, в другом часами разбирали чужой код и винили в усталости объём. В наших телефонах всё ещё лежат снимки экрана с самыми длинными списками коммитов, те, что мы когда-то отправляли друзьям. На снимках ничего не изменилось: те же строки, та же сплошная зелёная полоса. Только теперь в каждой строке проступает ещё и то, чего мы тогда не решили, и отделить одно от другого уже не получается. Фраза «Он у меня работает вторые сутки» и сейчас верна, просто в ней слышно, чем он был занят. Гордость не прошла. Рядом с ней встало неудобство, и оно не проходит тоже.

Листок под лампой к ночи исписан почти до конца, и все вопросы на нём обращены к diff: почему семь дней, почему поровну, почему нет паузы. Новых среди них нет. Это вопросы из слова «нормально», только теперь у каждого есть чужой ответ. Одного вопроса там нет: почему в тот вечер шесть слов, за которыми не было ни одного ответа, легли так легко, что основатель опустил крышку ноутбука с приятным чувством, что работа пошла, и почему тридцать один час чужого выбора он носил в кармане как собственную силу и показывал за обедом знакомому. В diff об этом нет ни строки, а ему самому такой вопрос в голову не приходит. Незнание остаётся при нём таким же, каким было за кухонным столом, — без имени и без причины.

Позавчера у него не было ответа ни на один из трёх вопросов, и он этого не видел. Сегодня ответы лежат перед ним — двести четырнадцать файлов ответов, — и ни одного из них он не давал. Этого человека не существует, он собран из многих вечеров, и наших тоже. Число его ночи от этого не становится вымышленным. Его по-прежнему не округлить.

Тридцать один час — это не время работы ИИ-агента. Это размер того, чего вы не знали.

Глава 2. Длительность как признание

2.1. Словарь самообмана

«Мощь ИИ». «Он исследует». «Задача сложная». «ИИ-агент ищет оптимальное решение». «Посмотрим, что он придумает».

Эти фразы мы слышали много раз и много раз говорили сами. Их пишут под снимком панели прогона, когда счётчик токенов переваливает за очередной порог. Их произносят в разгар прогона, объясняя коллеге, почему ветка ещё не готова. Их говорят себе вечером, отправляя задачу, и среди ночи, когда встают посмотреть, как там идёт. Одну — с гордостью, другую — устало, третью — с любопытством, как о посылке, которую вот-вот принесут. Но ни одна из них не звучит как оправдание. Каждая звучит как хорошая новость. Глупости в них нет: так говорят люди, которые хорошо знают своё дело, и мы среди них. Нередко фразы идут и вместе, одной связной речью: «Задача сложная, он исследует, ищет оптимальное решение — вот она, мощь ИИ. Посмотрим, что он придумает». Каждая подпирает соседнюю, и вместе они убеждают.

Поэтому попробуем выслушать каждую по отдельности — без спора и без усмешки — и задать ей один вопрос: о чём ты на самом деле? Под каждой лежит не оценка и не настроение, а вполне конкретное незнание: несделанный выбор, неназначенный критерий, непонятая система. Его можно показать пальцем — вопросом, на который вечером никто не ответил.

Когда прогон идёт долго, его хочется назвать мощью или исследованием, и оба слова превращают саму длительность в достоинство: одно — в силу инструмента, другое — в работу познания. «Мощь ИИ» рождается у панели. Итерации сменяют друг друга, коммиты идут, счётчик токенов растёт, и к исходу суток ощущение, что владеешь силой, становится почти телесным. В нём нет ничего постыдного, и слово приходит само. Но что оно измеряет? В технике мощность — это работа, делённая на время: сильный двигатель делает то же дело быстрее слабого. Ночью мы восхищаемся мощью как раз за то, что дело длится долго. Это первая странность. Вторая в том, что работу считают от результата, а панель знает только расход: шаги, токены, коммиты. Она при этом не лжёт: всё это действительно было. Сколько сделано от целого и сколько осталось, она показать не может: и то и другое считается от финиша, а финиш никто не назначил. Что должно быть правдой в конце, чтобы мы сказали «готово»? Какой сценарий обязан пройти от начала до конца, от формы заказа до письма с подтверждением? Что у нынешних клиентов должно работать так же, как вчера? Какое число в отчёте должно сдвинуться и насколько? Пока ответа нет, мерить результат нечем, и мы меряем то, что меряется. Большой расход и получает имя мощи. Под этим словом лежит неназначенный критерий. По такой мере самой мощной была бы ночь, которая не кончается никогда.

«Он исследует» звучит, пока прогон ещё идёт и надо объяснить, что происходит: ветку пока не трогаем, он исследует. Слово взято из хорошего словаря — из лабораторий и экспедиций, где у работы всегда есть вопрос: его записывают одной строкой до начала, и по нему потом видно, что работа кончилась. Какой вопрос здесь? Где та строка, которую записали, отдавая задачу? Что мы будем знать завтра, чего не знаем сегодня? Честный ответ обычно неловок. Мы не знаем, кто, кроме модуля заказов, читает таблицу с адресами доставки. Не знаем, какой из двух похожих обработчиков уведомлений ещё вызывается, а какой остался от прошлой переделки. Не знаем, что сломается в ночном отчёте, если поменять формат даты. А раз строки нет, нет и признака, что работа кончилась: исследовать можно сколько угодно. Самое странное в этой фразе — её предмет. Это не дальний материк и не новое вещество, а наш собственный

репозиторий, в котором мы работаем каждый день. Знания о своём же коде у нас нет, и слово «исследует» превращает непонятую систему в его занятие.

Обе фразы — похвала ИИ-агенту: одна хвалит его силу, другая — его любознательность. О задаче ни одна как будто ничего не сообщает. Но сведения о задаче в них есть, только записаны наоборот: вместо «нам нечем мерить» — мощь, вместо «мы не понимаем» — исследование.

Третья и четвёртая фразы ничего не хвалят. Они объясняют длительность и звучат трезвее, как инженерная оценка, поэтому им верят охотнее. «Задача сложная» переносит незнание на саму задачу и говорится так, будто сложность — свойство вроде веса: одна задача весит килограмм, другая центнер, и ничего с этим не поделать. Что именно в ней сложно? Чаще всего за этим словом стоят не трудные строки кода, а вопросы, на которые можно ответить по-разному. Можно ли менять структуру таблицы заказов, или прежние записи должны читаться как раньше? Должны ли старые версии мобильного приложения по-прежнему понимать ответы сервера? Где граница, за которую ИИ-агенту заходить нельзя, и что внутри неё разрешено ломать? На каждый вопрос есть два-три разумных ответа, и каждый ведёт к своему коду. Пока ни один не выбран, в задаче спрятана не одна задача, а несколько, наложенных друг на друга, и она действительно выглядит тяжёлой. Проверить это легко: стоит ответить на эти три вопроса, и задача на глазах становится проще, хотя в ней не изменилось ни строчки кода. Сложность, которую мы ей приписываем, — это число невыбранных ответов.

«ИИ-агент ищет оптимальное решение» переносит незнание на занятие ИИ-агента и звучит почти научно. Фраза обещает, что где-то уже есть лучший вариант и его осталось найти. «Оптимальное» — слово из математики, а в математике оно не стоит одно: оптимальное всегда оптимально по чему-то. По скорости, с которой пользователь проходит оформление? По числу изменённых файлов? По тому, насколько легко будет прочесть этот код через полгода? По стоимости серверов? Каждый признак даёт своё решение, и лучшее по одному нередко оказывается худшим по другому. Даже вопрос о шагах — с чего начать, что переписать целиком, что поправить точно, в каком порядке выкатывать — не имеет ответа, пока не сказано, что значит «лучше»: кто бережёт время, начнёт с одного, кто бережёт читаемость — с другого. По какому признаку сравнивать, не сказал никто, и вся фраза значит только одно: ищет долго. Под ней — неназначенный критерий.

Пятая фраза отличается от остальных. В «посмотрим, что он придумает» нет ни силы, ни сложности, ни строгости. В ней впервые появляемся мы сами — и появляемся зрителями. «Посмотрим» говорят в зале кинотеатра, на трибуне, у окна перед грозой. А эту фразу чаще всего говорят последней, когда задача уже отправлена и остаётся только уйти. В ней заложена вся ночь. Её обычно произносят с лёгкой улыбкой, будто говорящий сам слышит, что сказал. Она почти прозрачна. Это не стиль постановки, а признание, и из пяти фраз только в этой его слышно без перевода. В остальных четырёх оно тоже есть, но его приходится доставать вопросами.

Если поставить пять фраз рядом, видно, что это не случайные оговорки, а то, как устроен язык. В четырёх из них нет того, кто отдал задачу: подлежащее в них — мощь, он, задача, ИИ-агент. В пятой говорящий есть, но только как тот, кто ждёт. Ни одна не говорит «я не выбрал», «я не понимаю», «я не сказал, что считать готовым», хотя любая из этих коротких фраз была бы точнее всех пяти. Каждая берёт пустое место и делает из него достоинство: невыбранное становится сложностью, непонятое — исследованием, неназначенное — мощью и оптимальностью, ожидание — интригой. И в каждой остаётся приоткрытая дверь: вдруг он найдёт, вдруг придумает.

Общее у них и время. Мощь и сложность стоят в настоящем, как свойства, которые были всегда: задача будто родилась сложной, а не стала такой, когда вопросы о ней остались без ответа. «Исследует» и «ищет» — тоже в настоящем, но как процесс, которому не видно конца.

«Посмотрим» — в будущем. Прошедшего времени нет ни в одной. Ни одна из пяти фраз не говорит о вечере, когда задачу отдавали, хотя всё, что они закрывают, должно было решиться именно тогда.

Расскажем ту же ночь без этих фраз. Вечером мы не сказали, что будет считаться готовым. Не решили, что можно ломать. Не знаем, кто ещё читает нашу таблицу. Не назвали, что значит «лучше». Посмотрим. Ни одно слово здесь не ложно, и ни одно не противоречит пяти фразам. Ночь та же, и часы на панели те же, изменились только слова: хорошая новость превратилась в перечень несделанного. Язык, которым мы говорим о долгих прогонах, устроен так, чтобы незнание звучало как возможность.

Такие фразы — маски: привычные слова, которыми незнание выдаётся за возможность. Лгать маске для этого не нужно, достаточно закрыть то, что под ней. Произносим мы её вслух, для других, но прячем незнание прежде всего от самих себя: первым эти слова слышит тот, кто их говорит. Изнутри это выглядит буднично. Мы садимся дописать к задаче, что будет считаться готовым, начинаем фразу и упираемся в вопрос, что делать с заказами, оформленными по старым правилам. Ответа нет, и в голове сама собой звучит: задача сложная. Вопрос не решён, но уже и не мешает. Строка остаётся недописанной, задача уходит, а мы искренне уверены, что всё учли. Коллега, услышав «он исследует», ещё может переспросить: что именно? Себя мы не переспрашиваем. Поэтому словарь этот — словарь не обмана, а самообмана. Называть длительность мощью, исследованием или сложностью задачи — самообман.

Всё, что нашлось под пятью масками, одного рода. Такого же рода было незнание, которое тридцать один час светилось силой на панели основателя и утром обернулось двумястами четырнадцатью изменёнными файлами: то самое, чего он не знал. Его вопросы были о другом — о состояниях подписки, о неудачном списании, о границе между биллингом и авторизацией, — но устроены так же. И имя у всего этого одно — неопределённость: всё, что на момент передачи задачи не известно и не решено. Слово длинное и неуклюжее, и в этом его честность: в нём нет ни похвалы, ни обещания.

Долгий прогон — всегда неопределённость в том или ином виде: несделанный выбор, неназначенный критерий, непонятая система, а чаще всё сразу. Не иногда, не в неудачную ночь, не у тех, кто плохо ставит задачи, — всегда. Хорошая новость, которой звучала каждая из пяти фраз, была новостью о ней. Мы называли её мощью, исследованием, сложностью, оптимальным решением и интригой, но под каждым из пяти имён прогон двигала она. У долгого прогона нет другого топлива.

2.2. Чёткая задача, которая идёт сутки

Задача была короче некуда: добавить в API ещё один метод, по образцу соседних. В ней не было ни одного расплывчатого слова, и понять её неправильно, казалось, невозможно. Мы отдали её вечером, рассчитывая утром первым делом прочитать короткий diff и закрыть вопрос. Прогон по ней шёл сутки. Ошибки в самой формулировке потом так и не нашлось: слова стояли на своих местах, и каждое значило ровно то, что мы имели в виду.

Удивление здесь законное. Интуиция «чётко поставил — быстро получил» верна, и каждый, кто отдавал такие задачи, знает это по опыту. Тест упал после вчерашнего переименования функции: ИИ-агент читает сообщение об ошибке, находит старое имя ещё в двух местах, правит, запускает тесты, и через пять минут всё снова зелёное. Форма регистрации принимает адрес почты без домена: добавить валидацию и тест на неё — минуты. В библиотеке журналирования закрыли уязвимость: обновить зависимость, запустить тесты, закоммитить. Ни одной из этих задач не нужно больше нескольких минут: правка сделана — и прогон окончен. Быстрыми их делает не только малый объём. В каждой уже всё решено. Упавший тест сам говорит, что считать правильным. Адрес без домена — не адрес, спорить тут не о чем. Номер новой

версии назван, а старые тесты покажут, сломалось ли что-нибудь. Такая задача приходит вместе со своим критерием, и остаётся только сделать.

Шкала на этом не кончается. Бывают задачи по-настоящему большие: модуль загрузки файлов, который режет большой файл на части, докачивает их после обрыва связи и сверяет собранное с отправленным. Если заранее решено, какого размера части, где держать недокачанное и сколько его хранить, и назначен признак готовности — скажем, файл в несколько гигабайт доходит целым через соединение, которое трижды рвет посреди загрузки, — такой модуль занимает несколько часов. Кода в нём много, тестов много, и ИИ-агент всё это время работает, а не топчется на месте: пишет разбиение на части, докачку и сверку, запускает тесты с оборванным соединением, чинит то, что они находят. Каждый час прогона виден в коде. Эти часы — не долгий прогон. Это норма: ровно столько требует объём.

Отсюда мера. Если решение известно, устройство выбрано, критерий назначен, задаче хватает времени, соразмерного её объёму: минут на правку, нескольких часов на сложный модуль. Долгий прогон — тот, что длится дольше, чем требует объём задачи, в которой всё это уже решено. Восемь часов на модуль загрузки — не долгий прогон. Восемь часов на валидацию одного поля — долгий. Сами по себе сутки ещё не мера: сутки на что? Мерой их делает шкала: только рядом с ней видно, какая часть часов ушла на объём, а какая — на что-то другое.

Теперь поставим рядом другие задачи из той же недели, тоже в одну строку. Обновить библиотеку, которая сжимает картинки, — в одном сервисе это заняло минуты, а в соседнем прогон шёл до конца следующего дня. Добавить валидацию — минуты; добавить метод API «по образцу соседних» — сутки. Модуль загрузки по выбранному устройству — несколько часов, а удаление учётной записи, работа в разы меньше, шло сутки с лишним. Вторая задача в каждой паре, когда мы её писали, казалась ничуть не менее чёткой, чем первая, и писалась так же быстро. Такие пары мы встречаем то и дело, когда отдаём ИИ-агенту мелкие задачи пачкой: почти все возвращаются за минуты, а одна-две не возвращаются и через сутки.

Утром diff у суточных задач оказывается не больше, чем у их быстрых пар. В соседнем сервисе — только новый номер версии. Новый метод — не длиннее любого из соседей. Удаление учётной записи — десяток файлов. Объём сопоставимый, а то и меньше; время несравнимое. Проще всего списать разницу на случай: не повезло, ночь не задалась, прогон свернул не туда. Или сказать, что задача оказалась сложнее, чем выглядела, и тем самым вернуть маску, без которой только что обходились. Но ИИ-агент был тот же, что днём справлялся за минуты, и ставили задачи мы не хуже.

Если задачи сопоставимого объёма разделяют минуты и сутки, это не загадка, а улика. Время прогона не берётся ниоткуда: раз его не съел объём, его съело то, чего задаче не хватало. Значит, если «чёткая» задача идёт несоразмерно своему объёму, чёткость была иллюзией, и неопределённость просто спряталась. Маска ей для этого не понадобилась: в этих парах есть только задача и время. И время говорит то же, что говорили маски. Улика ещё не показывает, где пряталась неопределённость. Она показывает только, что неопределённость была.

Спрятаться ей было где, и ни одно из этих мест не видно в самой задаче: её можно перечитывать сколько угодно и не заметить ни лишнего слова, ни недостающего. У каждой из трёх суточных задач место было своё.

Сама библиотека и в соседнем сервисе обновилась за минуты, и на ноутбуке тесты прошли. На сервере сборки упал тест, который сравнивает готовую миниатюру с эталонной: миниатюра вышла на едва заметный оттенок темнее. Пал он при каждом запуске, без исключений. ИИ-агент перебирал версии библиотеки, пересобирав, откатывал, собирал снова, клал рядом миниатюры с ноутбука и с сервера и сравнивал их точка за точкой. Каждая пересборка занимала минут двадцать, и за ночь их набралось несколько десятков. Код один, версия одна, исходная картинка одна, а цвет разный, и причины он не находил. К утру стало видно, в чём дело: на сервере под нашей библиотекой работает системная библиотека другой версии, и цвет

она считает чуть иначе. Формулировка «обновить библиотеку» была чёткой. Капризным оказалось окружение, в котором её предстояло выполнить, и как оно себя поведёт, не знал никто — ни мы, ни он. Неопределённость пряталась снаружи, в мире. Там же живёт и редкий баг: задача починить его бывает короткой и чёткой, а когда и отчего он случается, из неё не узнать.

С методом API окружение ни при чём. Соседей у него два десятка, и написаны они по-разному: половина ходит к данным через старый слой, половина — через новый; одни сообщают об ошибках в одном виде, другие в другом; где-то длинный список отдают по номерам страниц, где-то — с меткой, откуда продолжать. В команде правило знает каждый: новое пишется только через новый слой, списки отдаются с меткой, а старые методы не трогают, пока их не перепишут целиком. Ни в одном файле оно не записано. Оно живёт в старых замечаниях на ревью, в истории переделок, в привычке всей команды.

ИИ-агент взял за образец ближайшего соседа — старого — и повторил его во всём. Потом стал читать историю соседних файлов, сверять даты и авторов правок, заметил, что за последний год к старому слою не прибавилось ни одного нового обращения, и переписал метод через новый. Потом нашёл два способа отдавать списки, долго сличал их с окружающим кодом и выбрал тот, что встречался чаще, — старый. Утром метод работал, тесты проходили, а первое замечание на ревью было коротким: «у нас так не делают». Сутки ушли на то, чтобы по косвенным следам угадывать правило, которое любой из команды назвал бы за полминуты, — и угадать наполовину. Для нас «по образцу соседних» значило одно, а для того, кто правила не знает, — любое из двух десятков. Правило действует давно, но в репозитории, а не в задаче, и для задачи, ушедшей вечером, его не существовало. Неопределённость пряталась в общем коде — в том, что знают все и не записал никто.

Последнее укрытие ближе всего. «Дать пользователю удалить свою учётную запись» мы пишем, отчётливо видя результат: кнопка в настройках, окно подтверждения, записи больше нет. Ночью ИИ-агент дважды переделал удаление, сначала только помечая запись, потом стирая насовсем; в утреннем diff она исчезает сразу и безвозвратно. Вместе с ней стёрты все комментарии пользователя, и обсуждения, где ему отвечали другие люди, теперь обрываются на полуслове. Общие документы, которые он когда-то создал, удалены тоже, хотя писали в них вдесятером. Читая это, мы уверенно поправляем: удалённую запись месяц можно восстановить, комментарии остаются с подписью «пользователь удалён», это же само собой разумеется. На документах мы запинаясь. Что с ними делать, мы, оказывается, не знаем и сами. Кнопка в этих сутках — малая часть; остальное ушло на то, чего в формулировке не было: сразу или со сроком, стереть или обезличить, чей труд уходит вместе с записью. Часть ответов у нас была, но осталась в голове и до задачи не дошла. Части не было вовсе — было только ощущение, что она есть. Вечером одно от другого не отличалось: всё казалось одинаково понятным. Неопределённость пряталась в голове автора.

Ясность для себя ещё не означает определённости задачи. Определённость — свойство задачи в момент передачи: решения приняты, критерий назначен. Ясность — состояние головы человека, и она может обманывать. Определённость уходит вместе с задачей. Ясность остаётся у нас. В задачах на минуты они совпадали: мы видели результат, и задача несла в себе всё, что нужно, чтобы его получить. В задачах на сутки ясность была ничуть не меньше — такая же спокойная и полная. Изнутри головы пары не различались. Различались задачи.

Чёткость, которую мы чувствовали, была ясностью, и ясностью настоящей: результат мы действительно видели. Определённости не было ни в одной из трёх суточных задач: как поведёт себя сервер сборки, не знал никто; правило о соседях знала команда, но не задача; а что делать с общими документами, не решил никто. Сверх того, что требовал объём, прогон шёл столько, сколько в задаче было неопределённости, а не столько, сколько у нас было ясности. «Чёткая» задача, которая идёт сутки, была ясна только своему автору. Определённой она не была.

2.3. Кажущееся исключение: объём

В пятистах файлах нашего кода стоит вызов, которого в следующем выпуске сетевой библиотеки уже не будет: прежний интерфейс, через который код ходит к чужим сервисам, объявлен устаревшим. Авторы библиотеки приложили руководство по переходу: таблица в две колонки, слева старый вызов, справа новый. Переписать пятьсот файлов под новый API — работа, о которой не спорят на планёрке и которую никто не хочет делать руками: одна и та же правка, повторённая сотни раз, скучная ровно настолько, насколько предсказуемая. Мы отдаём эту работу ИИ-агенту вечером, одной фразой со ссылкой на руководство, и уходим спокойно: думать тут не о чем.

Взглянем на неё такой, какой она видится вечером. Масок здесь нет: никто не называет эту работу сложной и не ждёт, что ИИ-агент придумает что-то своё. Спрятаться неопределённости тоже негде. Что делать, сказано в руководстве, и сказано не нами, а авторами библиотеки, которые знают её лучше всех. Как делать, очевидно: открыть файл, найти старый вызов, заменить по таблице, перейти к следующему. Когда готово, понятно заранее: старых вызовов не осталось ни одного, сборка идёт без ошибок, тесты проходят. Каждый файл по отдельности — дело нескольких минут, и ни в одном нет ничего, над чем стоило бы задуматься. Их просто пятьсот.

И прогон по ним идёт долго: не час и не два, а больше суток. Здесь и звучит возражение, и отмахнуться от него нельзя. При чём тут неопределённость? Незнания в пятистах файлах не больше, чем в одном; больше только самих файлов. Работа долгая не потому, что в ней что-то не решено, а потому, что она большая. Ведь если время прогона соразмеряют с объёмом, то вот он, объём, и время ему соразмерно: пятьсот файлов по несколько минут и складываются в ночь. Будь файлов пять, прогон кончился бы за полчаса, и никто не заподозрил бы в нём никакого незнания. Вот оно, исключение: долгий прогон, под которым ничего не спрятано.

Но откроем файлы по одному. В первом вызов меняется ровно по таблице, во втором тоже, в десятом тоже. В двенадцатом старый вызов при сбое сети просто возвращал пустой ответ, и код ниже на это рассчитывал: пустота означала «показать то, что сохранилось с прошлого раза». Новый вызов в той же ситуации бросает ошибку. Руководство прямо об этом предупреждает, но что с ошибкой делать, сказать не может: оно описывает библиотеку, а не наш код. Можно поймать её и вернуть пустоту, как раньше. А можно дать ей подняться выше, туда, где её, возможно, давно ждут и ни разу не дождались. Здесь дорога, которая до сих пор не ветвилась, расходится, и дальше нужно выбрать. Такое место и есть развилка.

Дальше развилок становится больше. Старый вызов сам повторял запрос, если соединение обрывалось; новый не повторяет. Где мы только читаем данные, повтор безвреден, и его можно вернуть. Где запрос что-то создаёт на той стороне, повтор может создать это дважды, — и прежнее поведение, возможно, годами было тихой ошибкой, которую теперь можно исправить, а можно бережно перенести. Старый ждал ответа сколько угодно; новый требует назвать срок, и для запроса, ответа на который человек ждёт перед экраном, разумный срок один, а для фоновой выгрузки совсем другой. Старый сам шёл по перенаправлению на другой адрес; новый останавливается и возвращает этот адрес нам. Старый отдавал ответ готовым текстом, сам угадывая кодировку; новый отдаёт сырые байты, и угадывать теперь нам. Ни на один из этих вопросов не отвечает руководство, и ни на один — соседний файл: у каждого места свой код вокруг вызова и свои ожидания.

Пятьсот файлов — это пятьсот маленьких развилок, и чем их больше, тем больше молчаливых решений. Не буквально по одной на файл: где-то развилок нет вовсе, где-то три в одной функции. Но число работает против возражения, а не за него. Эти файлы писали разные люди в разные годы, и каждый по-своему опирался на старое поведение: один рассчитывал

на пустой ответ, другой — на повтор, третий о сбоях просто не думал. Каждый следующий файл добавляет не только замену, но и ещё одно место, где старый вызов может не перейти в новый один к одному. Сам по себе такой выбор мал: минута мысли того, кто знает, как этот сервис должен вести себя при сбое. Но в пятистах файлах такие минуты складываются в сотни выборов, разбросанных по разным местам, и ни один из них не прозвучал вслух.

В diff эти выборы не отличить от механических замен: ушёл один вызов, пришёл другой. В итоговом сообщении прогона все они уместятся в одну фразу: старые вызовы заменены в пятистах файлах. На эти развилки и тратится время прогона. Сама замена занимает мгновение; часы уходят на то, чтобы в каждом файле прочитать код вокруг, понять, чего он ждал от старого вызова, выбрать и проверить выбранное.

А если развилки действительно нет? Допустим, новый интерфейс изменил только имена: каждый старый вызов переходит в новый один к одному, поведение осталось прежним до мелочей, и мы знаем это наверняка. Тогда вся работа укладывается в одно правило замены: где стоит это, поставь то. Правило, которое записывают один раз и применяют пятьсот раз, — это скрипт: замена по шаблону или короткая программа, которая разбирает код и переставляет в нём вызовы. Такой скрипт пройдёт по репозиторию за несколько минут, при втором запуске сделает в точности то же, что при первом, и его целиком можно прочитать до запуска. Ему не нужно понимать код вокруг вызова и не нужно ничего выбирать: ему хватает шаблона. Если в задаче нет неопределённости, ей нужен скрипт, а не ИИ-агент.

Механическую замену ИИ-агент сделает тоже, и сделает аккуратно. Но всё, чем он отличается от короткой программы, — умение прочесть незнакомое место, понять, чего от него ждали, и выбрать — в работе, где выбирать нечего, не пригодится ни разу. Отдавая работу ему, а не скрипту, мы зовём того, кто выбирает, и тем самым признаём, что выбирать есть что. ИИ-агент нужен только там, где есть что решать или искать.

Третьего не дано. Можно сказать: в тех нескольких файлах, что мы успели открыть вечером, вызов менялся ровно по таблице; развилки, скорее всего, нет, но за все пятьсот мы не поручимся, потому и отдаём работу ИИ-агенту, а не скрипту. Это и есть неопределённость: на момент передачи не известно, в каких из пятисот файлов дорога одна, а в каких расходится. Знать, что развилки нет, — значит уметь записать правило замены целиком. Если записать его целиком не получается, значит, где-то замена перестаёт быть заменой, и часы прогона уйдут на то, чтобы выяснить, где именно, — файл за файлом. Большая механическая работа бывает либо с развилками, либо под скрипт. Долгой и при этом определённой она не бывает.

Вечером мы видели в этой работе только объём. Но из всего, что в ней было, объём оказался самым коротким: скрипт проходит его за минуты. Всё остальное время забрали развилки, доставшиеся прогону невыбранными. Будь в тех же пятистах файлах вдвое больше развилки, ночь вышла бы длиннее; не будь их вовсе, она сжалась бы до минут скрипта. Возражение говорило: долго, потому что много. Точнее сказать: долго, потому что много развилки. А нерешённая развилка — неопределённость в самом прямом смысле слова: выбор, который отдали вместе с работой, не сделав.

Долгий прогон — всегда неопределённость, и теперь за этим «всегда» стоят пятьсот файлов: двенадцатый с его пустым ответом, запросы, которые нельзя повторять вслепую, срок ожидания, который никто не назвал, кодировка, которую больше никому угадывать. Большая механическая работа не обошла правило, а показала его с другой стороны: здесь неопределённость не пряталась ни под одним словом, она была разложена по файлам. Исключение, разобранный честно, подтверждает правило.

2.4. Названная неопределённость

Раз в два-три дня сервис личных кабинетов перезапускается сам. Ничего не падает с грохотом: память растёт медленно, ровной линией, пока процесс не упирается в предел и его не перезапускают, и пользователи замечают только несколько секунд без ответа. На тестовых данных рост не повторяется, в журналах виден лишь сам перезапуск. Отдавая этот баг ИИ-агенту, мы пишем всё, что знаем, и отдельно — чего не знаем. Знаем, что началось это три недели назад, после выпуска, в который вошло два десятка изменений. Знаем, что в будни память растёт быстрее, чем в выходные. Знаем, что две самые очевидные догадки уже проверены и обе оказались мимо. Не знаем одного: отчего. Ни у кого из нас ответа нет: он есть только у самого сервиса, и получить его можно, лишь расспрашивая сервис. Прогон идёт двое суток. ИИ-агент поднимает сервис на стенде, пропускает через него записанные за неделю запросы, ускорив их во много раз, делает снимки памяти через час работы и через три. Потом откатывает изменения выпуска половинами, затем половинами половин, и смотрит, без каких линия перестаёт ползти вверх. Под конец второго дня подозреваемый остаётся один: таблица в памяти, куда сервис заводит запись для каждого нового сочетания параметров запроса и откуда ничего не удаляет. Исправление после этого — дело минуты.

Старый сервер, на котором годами по расписанию запускались фоновые задания, выводят из работы, и задания переезжают в новый планировщик. Список заданий есть, расписание есть, код каждого лежит в репозитории; сам перенос — работа на час. Но за годы на одном сервере задания срослись с ним и друг с другом, и ни одна из этих связей не видна, пока смотришь на задание по отдельности. Одно берёт файл, который соседнее оставило на локальном диске часом раньше. Другое отсчитывает начало дня по часовому поясу сервера, а не по тому, что указан в настройках. Третье годами держалось на том, что его никогда не запускают дважды одновременно: сервер был один, задание успевало закончиться задолго до следующего запуска, и второму экземпляру взяться было неоткуда. Что такие связи есть, известно наверняка: без них старые серверы не живут. Где они и сколько их, неизвестно, и в задаче так и написано. ИИ-агент запускает задания в новом окружении рядом со старым, следит, что каждое читает и что пишет, несколько дней подряд сравнивает, что получилось там и там, и разбирает каждое расхождение, пока задания в новом планировщике не начнут вести себя так же, как на старом сервере. Связей находится семь, и о двух из них не помнил даже тот, кто когда-то писал эти задания. Вместо часа прогон занимает три дня, и почти всё это время съедают связи, о которых до начала не подозревали.

Страница со списком товаров отвечает за полторы секунды, а должна укладываться в треть секунды, на тех же запросах, что пользователи присылали за последний месяц, и с тем же содержимым. Число названо, замерить его можно в любую минуту. Какими правками к нему прийти — вопрос открытый. Может быть, хватит одного индекса в базе данных. Может быть, на каждый товар в списке уходит отдельный запрос к базе, и таких запросов сотни. Может быть, половину времени съедает не база, а подготовка ответа. А может быть, всё сразу, и каждое изменение даст понемногу. Какая из догадок верна, заранее не узнать: это выясняется только замерами, а сколько даст правка — только когда она сделана. ИИ-агент пробует одно, другое, третье. Индекс даёт немного. Один запрос к базе вместо сотен даёт почти половину. Готовые ответы, отложенные про запас, ускоряют страницу вдвое, но товары в ней встают не в том порядке, и эту попытку приходится откатить. К утру в коде остаются четыре правки из двух с лишним десятков попыток, и страница отвечает за четверть секунды.

Редкий прод-баг, миграция со скрытыми зависимостями, оптимизация с ясной метрикой: здесь долгий прогон законен. Этих прогонов не сократишь, подумав вечером подольше. Сколько ни сиди над задачей, причина перезапусков не придёт в голову, связи между задани-

ями не проступят в коде, а нужные правки не назовут себя сами. Сэкономить эти часы можно, только согласившись на догадку вместо ответа. Прятать здесь нечего, и красивое слово не нужно: время уходит туда, куда мы его отправили.

Но ни один из трёх прогонов не обходится без того, чего до его начала не знал никто. Это тоже неопределённость — пути и мира. Незнание бывает разным, и у того, с которым работают эти три прогона, два коротких имени. Путь — какими шагами дойти до цели. Мир — как на самом деле ведут себя код, зависимости и окружение, а не как они описаны. Эту часть незнания и можно отдавать машине: это и есть честная работа ИИ-агента. Отдавая её, нельзя сказать заранее, чем кончится прогон, сколько он продлится и найдётся ли вообще то, за чем мы отправили ИИ-агента. Утечка могла оказаться не в нашем коде, а в чужой библиотеке, которую нам не исправить; страница могла упереться в половину секунды, что с ней ни делай. Законная неопределённость остаётся неопределённостью.

Где же тогда граница между таким прогоном и самообманом? Не в задаче. Тот же баг с памятью можно отдать иначе. Вечером уходит та же задача, только без того, что мы о ней знали: без выпуска, после которого всё началось, без будней и выходных, без двух догадок, которые мы уже отбросили. ИИ-агент добросовестно начинает с самых очевидных догадок и тратит первые полсутки на то, чтобы проверить обе заново. Не зная, что память растёт по будням, он гоняет сервис под нагрузкой выходного дня и долго не видит роста. Не зная, когда начался рост, откатывает изменения за полгода, а не за три недели. Каждый такой крюк стоит ему часов. А на второй день, когда в еженедельной сводке надо написать, почему исправления всё ещё нет, мы пишем коротко: там редкий баг. Такой ответ устраивает всех, и нас самих тоже. Фраза звучит как объяснение, но ничего не объясняет. Слово «баг» говорит, что что-то сломано, слово «редкий» — что сломано не каждый раз; отчего, где и с какого выпуска, не говорит ни одно. Опровергнуть её нельзя — баг действительно редкий и действительно там, — и потому её не слышно: это маска, только сделанная из инженерных слов. Под ней та же неопределённость, что и в честном прогоне, — как ведёт себя наш собственный сервис, — и прогон по ней идёт те же двое суток, если не дольше. Баг тот же, и часы не короче.

То же с фразой, которая в словаре самообмана звучала почти научно: «ИИ-агент ищет оптимальное решение». В ней сказано о занятии ИИ-агента и ни слова о нашем незнании: одно заслоняет другое. Но в прогоне со страницей товаров он и правда подбирает шаги, которые заранее были неизвестны. Скажем о нём ту же фразу, назвав незнание: мы не знаем, какими правками прийти к трети секунды, и отдали ему именно это. Занятие прежнее; только под словом «оптимальное» теперь стоит число, а под «ищет» — названное незнание, и маски больше нет. Фраза из оправдания стала отчётом о работе.

Граница, выходит, проходит не по задаче, а по знанию о задаче. Честный прогон отличается от самообмана не тем, что в нём нет незнания, а тем, что его источник назван: мы знаем, чего не знаем, и знаем, почему отдали это машине. Вопрос не в том, отдали ли мы неопределённость, — отдали, раз прогон длинный. Вопрос в том, какую именно, и признаёмся ли мы себе в этом. Честность начинается с того, что мы знаем, какую отдали. Не с удачи: честный прогон может вернуться ни с чем, а прогон под маской — случайно наткнуться на ответ, но удача задним числом не переписывает вечер. И не с короткого срока: прогон с названной неопределённостью тоже может идти сутками.

У работы ИИ-агента в таких прогонах есть своё имя, и это не «он исследует». Этим словом мы называли прогон, у которого не было вопроса: оно обещало знание, не говоря, о чём. Законная работа над путём и миром называется поиском. «Исследует» говорит о том, чем занят ИИ-агент; «ищет причину перезапусков» — о том, какого ответа ждём мы. Искать можно только названное, и у поиска вопрос есть, заданный до начала прогона: отчего растёт память, что задания молча берут друг у друга, какими правками прийти к трети секунды.

Защита поиска не отменяет тезис, а уточняет его. Неопределённости в трёх законных прогонах не меньше, чем под любой маской; нет в них только молчания о ней.

Хорошим долгий прогон от этого не становится, как не был и плохим. Он всегда неопределённость, и спрашивать о нём стоит не «сколько», а «какая» и «признана ли». Двое суток под словами «там редкий баг» не отвечают ни на один из этих вопросов. Двое суток над утечкой, отданной вместе с выпуском, буднями, отброшенными догадками и честным «отчего — не знаем», отвечают на оба ещё до начала: неопределённость — в том, как ведёт себя наш сервис, и она записана в самой задаче. Такой поиск — промывка на известной жиле: законная работа ИИ-агента, и тоже неопределённость, только названная.

2.5. Способность и смысл

Способность ИИ-агента работать сутками — реальное достижение, и индустрия справедливо им гордится. Удерживать задачу на сутки — значит идти шаг за шагом и ни на одном шаге не потерять, куда идёшь. Длинная работа редко ломается на самом трудном месте. Чаще она рассыпается от мелочей: от забытого условия, от попытки, которую уже делали, от правки, которая молча спорит с правкой шестичасовой давности. ИИ-агент, который не рассыпается, на двадцатом часу помнит, о чём была задача и что из сделанного в начале прогона трогать уже нельзя. Он упирается в ошибку, которая десятый раз подряд всплывает в новом месте, и разбирает её заново, а не бросает работу и не начинает всё сначала. Помнит, что в полночь откатывал похожую правку, и не повторяет её на рассвете. Замечает, что его собственное давнее изменение теперь мешает, возвращается к нему и переделывает. И всё это время сам запускает сборку, сам читает, что она вывела, и сам решает, что делать дальше: подсказок ему не шлёт никто.

Здесь не о чем спорить и нечего умалывать. Восхищение, с которым мы смотрим на такую работу, — не наивность: оно видит настоящее умение и видит его верно. Для того, кто смотрит на ИИ-агента, долгая работа значит именно это — что он умеет.

Но для владельца задачи эти сутки значат другое: не что умеет ИИ-агент, а сколько незнания было в задаче в момент передачи. Владелец — не должность, а отношение к результату: это тот, кому результат нужен и кто утром отвечает за слово «принято». Чаще всего это тот же человек, что ночью любовался работой. Человек тот же — меняется вопрос, который он задаёт прогону: ночью — на что способен ИИ-агент, утром — что было в задаче, которую он сам отдал.

Тридцать один час основателя можно прочесть дважды. Первое прочтение — как достижение: всё это время ИИ-агент не выпускал из рук одну задачу, провёл её через двести сорок-пятьдесят файлов, выстроил собственный конечный автомат подписки, дошёл до модуля авторизации и не бросил работу на полпути. Второе — как мера задачи. Конечный автомат — это ответ на вопрос, какие у подписки бывают состояния. Правки в модуле авторизации — ответ на вопрос, где кончается биллинг. Даже сам конец прогона — ответ на вопрос, что считать готовым. Во фразе «переведи биллинг на подписки, сделай нормально» этих ответов не было. Их дала ночь. Факты в обоих прочтениях совпадают: ни один не прибавлен и не убран; меняется только то, к чему их относят, — к инструменту или к задаче. Каждый час, который первое прочтение записывает ИИ-агенту в заслугу, второе записывает задаче — как ещё одно нерешённое в ней.

Основателю второе прочтение досталось утром, как неприятная новость о его ночи. Но оно не принадлежит одной ночи и одному человеку. Так читается любой долгий прогон, и оба прочтения всегда верны сразу. И это не состязание, в котором кто-то должен проиграть: ИИ-агент не становится хуже оттого, что те же часы говорят и о задаче, а мы не становимся умнее оттого, что это заметили.

Отделим друг от друга две вещи, которые мы привыкли смешивать: характеристику инструмента и характеристику собственной ясности. Вернее, не самой ясности — изнутри головы всё обычно казалось ясным, — а того, что из неё успело стать определённой задачей: какие решения приняты и назначен ли критерий. Первая характеристика — о том, что ИИ-агент может: как долго держит задачу, сколько кода прочтёт, не потеряв нити, как аккуратно правит чужое. Вторая — о том, с чем мы его оставили.

Смешиваем мы их не по глупости. Это привычка, и заработана она честно: почти у всех наших инструментов время работы и правда говорит о них самих. Сборка идёт долго — значит, медленный компилятор или слабый сервер; копирование тянется — значит, медленный диск. Другой инструмент — другое время; другая работа того же размера на том же инструменте — почти то же время. С ИИ-агентом иначе. Один ИИ-агент, получив две задачи в один вечер, возвращает одну через минуты, а другую, не больше первой, через сутки, и способность у него в обоих случаях одна. Меняется задача — меняется время, а ИИ-агент остаётся прежним. К тому же ночью перед глазами только он: задача ушла вечером и больше не видна, а он на виду работает, правит, запускает тесты. Часы идут при нём, и мы записываем их ему.

Можно возразить: не умей ИИ-агент держать задачу так долго, прогон кончился бы раньше. Кончился бы — и вернул бы задачу недоделанной: неопределённость в ней осталась бы той же, её просто не прошли бы до конца. В задаче про биллинг это значило бы не меньше нерешённого, а меньше пройденного: вопросы о состояниях подписки и о границе с авторизацией остались бы на месте, только без ответов, которые дала ночь. Выносливость не создаёт этих часов, она только не даёт им оборваться. Способность работать сутками — не мощь, а длина строительной рулетки. Пока рулетка достаёт до стены, она показывает длину комнаты, а не свою. Короткая не делает комнату меньше, она обрывается, не дойдя до стены. Длинная доходит — и показывает комнату целиком.

Первое слово словаря самообмана, «Мощь ИИ», и есть язык этого смешивания. Оно складывается из двух правд и одного перехода. «Он работал сутки» — правда. «Сутки работы по силам только инструменту, который умеет держать задачу» — тоже правда. «Значит, сутки — от его мощи» — здесь и совершается подмена: часы задачи переходят на счёт инструмента и встают в один ряд с его настоящей способностью. Потому эта маска и держится так крепко: всё в ней правда, кроме того, кому отданы часы. На вопрос, чьи это сутки, «Мощь ИИ» отвечает не задумываясь: его.

Ответ на этот вопрос лежит раньше ночи: в минуте, когда текст задачи отправлен и ноутбук закрыт. Эта минута и есть вечер прогона, даже если задача ушла в полдень: время суток тут ни при чём. В эту минуту всё решённое уходит вместе с задачей, и всё нерешённое — тоже. Дальше прогон идёт без нас, и ничего своего мы к задаче уже не прибавим: каждое несделанное решение достанется ночи и заберёт свои часы. Длительность задана уже здесь. Ночь её только выговаривает — час за часом.

Представим четыре прогона одинаковой длины, идущих рядом: один под словами «задача сложная», другой над задачей, которая вечером казалась чёткой, третий над пятьюстами файлами, четвёртый над утечкой памяти, про которую в задаче записано и что известно, и чего нет. Утром на часах у всех четырёх стоит одно число, и по часам их не различить. По ИИ-агенту тоже: во всех четырёх он работал одинаково добросовестно. Способность у него одна на всех, и как раз поэтому о самих прогонах она не говорит ничего. Прогонки разные, и за каждым стоит своя неопределённость. Различает их не ИИ-агент и не число часов, а то, какая неопределённость в них была и названа ли она в минуте передачи.

Поэтому длительность — признание при любом имени, которое мы ей даём. Скажем ли мы «Мощь ИИ», «задача сложная» или честное «он ищет причину перезапусков» — сутки всё равно рассказывают одно: сколько в задаче оставалось нерешённого. Разница лишь в том, кто признаётся первым, — и только это зависит от нас. Когда неопределённость названа, мы при-

знаёмся сами, вечером, своими словами, и прогон только подтверждает сказанное. Когда она спрятана за маской, признаётся сам прогон — утром, языком diff, и уже не нашими словами.

Почему гордость настоящим умением так легко становится надеждой, что оно заменит знание, сутки не скажут. Зато о самой задаче они не ошибаются. Длительность — это не свойство ИИ-агента. Это свойство задачи в ту минуту, когда вы её отдали.

Глава 3. Лихорадка

3.1. Калифорния, 1849

Человек сидит на корточках у горной реки и держит в руках лоток. Он зачерпывает со дна песок вместе с мелким гравием, заливают водой и начинает медленно вращать: лёгкое выплёскивается через край, тяжёлое оседает на дно. Через несколько минут в лотке остаётся горсть тёмного осадка, и человек наклоняется над ним. Если в этом песке было золото, на дне блеснут жёлтые крупинки. Если не было, он выплеснет осадок и зачерпнёт снова. Выше и ниже по течению сидят такие же люди с такими же лотками, и каждый повторяет одно и то же движение с утра до темноты.

Так выглядела Калифорния в 1849 году. Годом раньше на лесопилке в предгорьях нашли золото, и новость, которая сперва казалась слухом, дошла до Орегона, до Мексики, до Чили. Через год к рекам хлынули десятки тысяч людей: одни шли через весь континент по суше, другие плыли морем. Большинство из них никогда прежде не держало лотка в руках. Лоток и не требовал выучки: широкая неглубокая миска, вода и терпение. Позже на реках появились качалки, длинные деревянные желоба, шлюзы, в которые отводили целые русла, но в памяти лихорадка осталась именно такой: река, лоток и толпа людей, промывающих песок.

В этом движении было что-то захватывающее, и делать вид, будто его не было, нечестно. Люди бросали хозяйства и ремесло, проводили в пути месяцы, чтобы встать по колено в воду чужой реки и начать с нуля. Никто не гнал их силой. Их вела картина, которую легко понять: золото лежит в реке, реку можно промыть, промывать может каждый. В этой картине нет ни одного звена, которое требовало бы особого знания. Нужны только руки, лоток и время, а рук и времени у толпы было сколько угодно.

В ней же легко увидеть историю успеха. Толпа у воды сама выглядит доказательством: раз столько людей бросили дома и приехали сюда, значит, золото есть и его хватит на всех. Историки лихорадки рассказывают другое. Самое доступное золото вымыли в первые же сезоны, и каждый следующий год у реки приносил меньше предыдущего. Часть старателей в итоге получила скромную прибыль, если вычсть дорогу, еду и снаряжение. Большинство разбогатеть не смогло, а те, кто приехал позже, чаще уезжали ни с чем или в убытке. Золото в реках было, и добыли его много. Просто толпа и золото встречались гораздо реже, чем обещал вид самой толпы.

В этом есть жестокая логика. Первый сезон прощал незнание: золото лежало так близко, что лоток находил его и в руках новичка, если тот успел прийти рано. Именно такие сезоны и порождают слух, который разносится по свету. Но слух идёт медленнее, чем пустеет река. Толпа ещё прибывала, а самое лёгкое уже вымывали, и то, что поначалу было необязательным, стало решающим: теперь нужно было не просто мыть, а знать, где мыть. Толпа приезжала за условиями первого сезона и заставляла условия второго.

Кого же отличала удача? Не лоток: лотки у всех были почти одинаковые, и вода в них вращалась одинаково. Первыми на реках оказались те, кто жил неподалёку, и те, для кого добыча металла была ремеслом ещё дома, — горняки из мексиканской Соноры и из Чили. Чилийцы приезжали со своими лотками, работали артелями и, в отличие от основной массы приезжих, часто были опытными горняками. Многие лучшие участки к приходу этой массы уже были заняты, а новичков добывать золото учили как раз те, кто умел это раньше. Их преимущество лежало не в руках и не в инструменте. Опытный горняк смотрит на реку иначе: он видит, где течение замедляется, где на излучине оседает тяжёлое, где берег похож на тот, у которого

золото уже находили. Новичок видит воду и песок. Умение, которое отличало удачливых, было умением знать, где искать.

Важно, чего это умение не делает. Оно не ускоряет промывку и не делает лоток лучше. Оно избавляет от тысячи лотков, промытых там, где промывать было нечего. Представим двух людей на одной реке, в сотне шагов друг от друга. У них одинаковые лотки, одинаковые руки, одинаковое упорство. Один моет в заводи за большим камнем, где вода веками замедлялась и роняла тяжёлое. Другой моет на быстрине, потому что там было свободно и близко к лагерю. Со стороны их не различить: оба с рассвета согнуты над водой, оба сделают за день одинаковое число промывок. Различит их только вечер, да и то не всегда. Лоток добросовестно промывает и песок с золотом, и песок без него, и по самой работе разницы не видно: те же движения, та же ломота в спине, тот же тёмный осадок на дне. А пустой вечер легко объяснить невезением и утром выйти к реке снова.

Что оставалось делать человеку, который не знал, где искать? Только одно: мыть больше. Больше лотков в день, больше дней у воды, больше мест вдоль реки. Нам легко смотреть на эту толпу свысока, зная из учебников то, чего не знал человек, стоявший по колено в воде. Изнутри всё выглядело иначе. Это не жадность. Это единственная величина, которую новичок держал в своих руках: знания у него не было, а время и силы были. Он не мог купить знание места, но мог встать раньше и лечь позже. Если нельзя понять, где золото, остаётся надеяться, что достаточно большое количество песка само его покажет. Когда таких людей десятки тысяч, их общая надежда становится видимой, у неё появляется пейзаж: берега, усеянные людьми, которые не знают, где искать, и поэтому ищут везде.

Эту надежду и называют лихорадкой. Лихорадка — это массовая надежда, что количество попыток заменит знание жилы. Не жар одного человека, а общее состояние толпы, где каждый видит вокруг себя сотни таких же лотков и принимает их число за довод. Лихорадка заразна, она охватывает сразу многих, и изнутри её трудно отличить от здравого смысла: так ведь делают все. Каждый новый человек на берегу укрепляет веру остальных, хотя о реке он знает не больше других. Лихорадка не выдумывает золота: оно действительно есть в реке, и именно поэтому надежда так убедительна. У лихорадки один признак, по которому её можно узнать в любую эпоху: уверенность, что если промыть достаточно песка, знать, где мыть, уже не понадобится.

Из этой картины достаточно взять два вопроса. Первый: знал ли старатель, где искать. Второй: была ли у лотка сетка. Первый вопрос — о человеке, который стоит у реки, второй — о том, что он держит в руках. Смешивать их нельзя: самый совершенный лоток не выбирает реку, а знание места само по себе ничего не промывает. Показательно и то, чего среди этих вопросов нет: сколько людей стояло у воды, сколько песка прошло через лотки, сколько часов длился день у реки. Именно эти числа лихорадка считает главными, потому что их видно издалека и их легко сложить. И именно они ничего не говорят о том, будет ли на дне золото. В 1849 году первый вопрос решал почти всё, а второй почти не звучал: лоток был простой миской, и всю остальную работу делали вода, тяжесть металла и глаз самого старателя, склонившегося над осадком.

Мы, оставляющие ИИ-агентов работать на ночь, тоже стоим у реки. Лоток стал другим: его больше не держат в руках, и работает он, пока мы спим. Река тоже другая, в ней нет воды и гравия, только код и задачи. Но вопросы к нашей ночи остались теми же, что у воды в калифорнийских предгорьях. Названия рек, имена, число людей на берегу меняются от эпохи к эпохе и почти ничего не объясняют. Объясняют два вопроса.

3.2. Весна 2023: лихорадка ИИ-агентов

Весной 2023 года, с разницей в несколько дней, в открытом доступе появились две программы. Одну, AutoGPT, выложил один разработчик, основатель игровой компании. Другую, BabyAGI, собрал венчурный инвестор. Устройство обеих уместается в одну строку: цель, план, действие, наблюдение. Человек пишет цель словами. Программа просит языковую модель разбить эту цель на задачи, берёт первую и выполняет её: ищет в интернете, записывает файл, запускает код. Результат возвращается в модель, та смотрит, что получилось, и составляет следующий план. BabyAGI вела себе список дел: выполняла задачу, по её итогам дописывала в список новые и заново сортировала его. Один такой проход, от плана до наблюдения, — итерация. Итерации шли одна за другой, пока их что-нибудь не останавливало.

ИИ-агентами здесь названы именно эти системы 2023 года, хотя код они писали не обязательно. Нередко они собирали сведения, составляли заметки и отчёты, а код был лишь одним из действий, которые им разрешалось выполнять. ИИ-агентами их делает не предмет работы, а её форма: задача дана словами, дальше много шагов подряд без участия человека.

Главное в этой строке не шаги, а место, где стоит человек. Цепочка от плана к наблюдению замыкается сама на себя: то, что модель выдала, возвращается к ней же, и она сама оценивает, куда двигаться дальше. Человек написал цель в самом начале и больше в этой цепочке не появляется. Он стоит вне контура. Ни в одной точке между планом и наблюдением нет места, где его суждение входило бы в работу. Программа не могла ни о чём его спросить: спрашивать было некого. Если на третьей итерации выяснялось, что цель можно понять двумя способами, выбор делала модель, и этот выбор тут же становился основанием для четвёртой итерации, пятой, десятой. Это не было недосмотром. В этом и состояла вся притягательность замысла: машина, которой достаточно сказать, чего ты хочешь, а дальше она справится сама. Смотреть на экран, где программа сама себе пишет следующий шаг, было захватывающе, и восхищение тех первых недель было честным.

Отсюда и ночи. Многие из нас той весной впервые оставили ИИ-агента работать без себя. Вечером человек вписывал цель, широкую, как желание, — что-нибудь вроде «собрать всё о рынке и составить план запуска продукта», — запускал программу в непрерывном режиме и уходил спать. ИИ-агент 2023 года трудился, пока никто не смотрел, и именно этого от него хотели. Прелесть была в том, что утром можно проснуться к готовому.

Утро выглядело иначе. Человек открывал экран, и сначала всё было обнадеживающе: журнал длинный, в рабочей папке прибавились файлы. Первые строки журнала и правда были похожи на работу: разумный план, первые запросы, первые заметки. Чем ниже, тем меньше в них было движения. Потом начиналось заикливание. Тот же запрос к поисковой системе, тот же вывод, тот же вывод из вывода, и снова тот же запрос, десятки раз подряд. На это в обсуждениях на странице AutoGPT жаловались уже весной 2023 года: программа застревала и раз за разом повторяла один и тот же поиск. Файлы в папке при ближайшем взгляде оказывались черновиками одного и того же, начатыми заново под чуть разными именами. Рядом рос счёт: каждая итерация была отдельным платным обращением к модели, и за ночь кредиты сгорали на повторы. А финиша не было. Цель не достигнута, отчёта нет, или есть что-то, отдалённо на него похожее, чем нельзя воспользоваться. Если к утру программа и стояла, то не потому, что дошла до цели, а потому, что кончились деньги, которые ей с вечера разрешили потратить.

Самое поразительное в том, что всё это было сказано заранее. В описании самого AutoGPT весной 2023 года авторы предупреждали, что работа на модели может обойтись дорого из-за расхода токенов, советовали заранее ставить лимит расходов, а непрерывный режим прямо не рекомендовали: в нём программа может работать вечно. Слова «работать

вечно» точнее любого пересказа тех ночей. Люди читали предупреждение и всё равно запускали: хотелось увидеть своими глазами, и в этом нет ничего постыдного.

Утро 2023 года выглядело так: заикливание, сожжённые кредиты и отсутствие финиша. Провал был грубым и прямым: его было видно с первого экрана, по одинаковым строкам журнала и по счёту за ночь. Разбирать его неделями не требовалось, хватало минуты, чтобы понять, что ничего не сделано. Он не притворялся работой.

Соблазнительно списать ту весну на наивность: люди поверили шуму вокруг новинки. Но честнее увидеть в ней другое. Те, кто запускал AutoGPT и BabyAGI, по большей части понимали, что программы сырые. Их вёл не обман, а вопрос, который стоило задать: что, если модели уже достаточно одной цели, а остальное она достроит сама? Это был эксперимент в чистом виде, такой, какой в инженерии удаётся поставить редко. Из него убрали всё, что обычно смешивается с результатом и мешает его прочесть: человека в контуре, внешнюю проверку, сколько-нибудь определённую задачу. Машине отдали всё, включая цель и критерий готовности. Цель ей дали фразой. О каком продукте идёт речь, для кого план, насколько он должен быть подробным, сколько сведений о рынке считать достаточным — всё это оставалось внутри фразы нерешённым, и решать приходилось той же модели. Когда работа готова, тоже решала она сама. Если у этого эксперимента должно быть имя, то вот оно: чистый эксперимент по делегированию неопределённости.

Зададим ему два вопроса из 1849 года. Старателем здесь был человек, который вечером вписал цель и ушёл спать. Знал ли он, где искать? Нет. Он не выбирал места: вместе с целью он отдал машине и вопрос о том, где копать, а заодно и что считать золотом. Лоток должен был сам выбрать реку, сам выбрать место на ней и сам решить, что там искать. Была ли у лотка сетка? Тоже нет. Ничто вне машины не отделяло результат, который можно принять, от песка. Каждую итерацию оценивала та же модель, что её выполнила, по тем же представлениям о цели, которые она сама себе составила. Лоток промывал и сам же решал, что блесит. Два вопроса, два «нет».

Ценность этого эксперимента в том, что ответ на него получился ясным. Когда из опыта убрано всё лишнее, результат не на что списать: ни на плохие тесты, ни на чужую ошибку в постановке, ни на коллегу, который вмешался не вовремя. Всё, что случилось той ночью, случилось между целью и машиной. Эксперимент показал, что делает лоток, когда кроме лотка нет ничего. Лоток промывает. Усердно, оплачивая каждый проход, по той же схеме снова и снова. Список дел BabyAGI пополнялся сам из себя, и никто не держал в руках его последнюю строку. Весна 2023 года задала машине вопрос, может ли она обойтись без человека на обоих концах работы, и получила ответ за несколько недель. Ответ был написан в журналах тех ночей.

А рядом с экспериментом звучало слово. В те же месяцы пресса называла бум вокруг языковых моделей «gold rush»: так писали о деньгах, которые потоком шли в новые компании, о спешке, о страхе опоздать. Позже, когда в центре внимания оказались ИИ-агенты, выражение перешло и на них. Сравнение с золотой лихорадкой не выдуманно задним числом: индустрия и те, кто о ней писал, произнесли его сами, с первых месяцев бума. Чаще его произносили с воодушевлением, как обещание. Но и те, кто предостерегал, говорили о деньгах и о спешке, а не о том, кто из пришедших уедет ни с чем. Метафора пришла раньше осмысления.

Что слышали в этом слове тогда? Золото: размер добычи, которая достанется первым. Спешку: надо успеть, пока участки не разобраны. В английском выражении слышен бег, в русском — жар, и оба оттенка хорошо ложились на ту весну. Но из всей истории, которую несёт слово, слышали только начало, где золото лежит в реке и бежать к нему нужно немедленно. Остального не слышали: что к рекам приходят десятки тысяч, а золото достаётся немногим; что отличает этих немногих не лоток; что толпа держится на надежде, будто количество промытого песка рано или поздно заменит знание жилы.

А ведь всё это происходило в те же самые недели, почти в тех же лентах новостей. Заголовки говорили «gold rush», а ИИ-агенты весны 2023 года тем временем ночами промывали песок там, где жилы никто не знал, и без сетки. Эти две вещи почти никто не сложил вместе. Индустрия назвала сюжет, но не прочла его. Она взяла из лихорадки её блеск и не заметила её механику. Прочсть этот сюжет значило бы спросить не о том, сколько в реке золота и как быстро к нему добраться, а о том, кто из пришедших знает, где его искать, и чем он отделит золото от песка. Слово оказалось точнее, чем думали те, кто его произносил.

3.3. Роли вместо решений

Урок весны прочитали быстро и по-своему. Если одиночный ИИ-агент теряется, рассудили многие, то потому, что он один: ему приходится быть сразу всеми — тем, кто задумывает, тем, кто делает, и тем, кто проверяет. На это у инженерии давно есть ответ, и ответ проверенный: разделение труда. Во второй половине 2023 года появились системы, где одну и ту же модель раздавали по ролям, как по штатному расписанию. Одна из них описывала себя как виртуальную компанию по разработке программ, другая — как сборочный конвейер, где каждая роль получает от предыдущей готовый документ и передаёт следующей свой. За ними последовали другие, и весь следующий год системы для программирования охотно собирали из ролей. Менеджер, архитектор, кодер, тестер.

ИИ-агентом в такой системе была каждая роль. Под каждой стоит та же языковая модель, только с другим промптом и другим названием, и каждая работает так же, как система весны: получает задачу словами и делает свою часть без участия человека. Названия у ролей взяты из человеческих профессий, но людей за ними нет. Есть модель, которой сказали, кем ей быть.

Смотреть на работу такой системы было убедительно. В журнале шёл разговор, похожий на совещание: менеджер излагал требования, архитектор отвечал документом, кодер переспрашивал, тестер возвращал замечания, и кто-то подводил итог. Если держаться картины реки, лоток теперь был не один: их поставили цепочкой и каждому дали должность. Перенести разделение труда на машину было естественным ходом мысли.

Возьмём одну роль — архитектора. Текст её промпта от системы к системе менялся, но смысл обычно сводился к одному: ты опытный архитектор программного обеспечения; по требованиям составь описание системы — модули и их обязанности, интерфейсы между ними, структуры данных; пиши так, чтобы по документу мог работать программист. На вход приходило требование человека, часто в одну-две фразы. Допустим, сделать сервис для учёта заказов небольшого магазина.

Ответ приходил быстро и выглядел безупречно. Документ с разделами: обзор, модули, интерфейсы, данные, выбор технологий. Модулей четыре: заказы, склад, пользователи, уведомления. Данные хранятся в реляционной базе. Модули обмениваются запросами через программный интерфейс. Каждое предложение утвердительное, ни одной оговорки, ни одного вопроса к заказчику. Документ звучал так, как звучат документы опытных архитекторов, и неудивительно: модель прочла их великое множество.

Теперь посмотрим, чего в нём нет. Нет вариантов, между которыми выбирали. Почему модулей четыре, а не два; почему склад отделён от заказов; что делать, если заказ оплачен, а товара на складе уже нет. Каждая строка документа — ответ, и ни за одной строкой не стоит выбор. Слово «решение» здесь расходится на два своих смысла. Решение как ответ задачи в документе есть: вот модули, вот данные, вот связи. Решения как акта выбора нет: никто не взвесил, чем одно устройство лучше другого для этого магазина, и никто не отказался от остальных. А архитектура — это и есть результат решений об устройстве, то есть именно таких выборов. Роль архитектора в промпте не принимает архитектурного решения. Она даёт исполнителю имя, а имя ничего не выбирает.

У людей порядок обратный. Архитектором человека называют потому, что он уже принимал такие решения и знает, чем за них расплачиваются через год. Название идёт следом за выборами. В промпте название пришло первым, и от него ждали, что выборы последуют сами. Последовал жанр: модель, которой сказали «архитектор», пишет текст, похожий на архитектурный документ. Жанр воспроизводим. Выбор в жанре нет.

Проверить это можно простым опытом, и многие из нас его невольно ставили. Дать системе ту же фразу про магазин ещё раз. Архитектор снова ответит быстро и снова безупречно, но документ будет другим: модулей окажется три или пять, склад сольётся с заказами, база данных сменится, а уверенность тона останется прежней. Решение в смысле выбора так себя не ведёт. Если выбор сделан, за ним стоят причины, и те же причины завтра приведут к тому же устройству. Здесь причин не было, и каждый запуск доставал из жанра новый правдоподобный вариант. Два документа нельзя даже сравнить по существу: ни в одном не сказано, от чего он отказался и почему.

Дальше документ уходил кодеру, и для кодера это были уже не предположения, а указания: раз архитектор написал «четыре модуля», значит, так решено. Строки несли вес роли, а не выбора, которого за ними не было. Неопределённость, пришедшая в систему вместе с фразой человека, никуда не делась. Она прошла через роль архитектора и вышла из неё в облике решённого.

То, что случилось с архитектором, случилось с системой целиком. Её строители воспроизвели форму организации: штатное расписание, передачу документов из рук в руки, замечания, итоговые сводки. Всё, что видно в организации со стороны, было на месте. Не хватало того, ради чего организация существует, — принятых решений. Отдел, где работают люди, держится не потому, что в нём есть должности. Должности — это места, куда складывают принятые решения: что мы строим, для кого, что будет считаться готовым, чего нельзя трогать. Когда живой архитектор пишет, что склад стоит отдельно от заказов, за этой строкой обычно лежит знание о деле: скажем, что скоро складом начнут пользоваться ещё два магазина. Когда такие решения приняты, роли распределяют работу по их исполнению. Когда не приняты, роли делят между собой только текст. Организацию имитировали, а определённости в ней не появилось.

Поэтому надёжность таких систем и оставалась слабой, и упиралась она не в число ролей. Добавить в схему ещё одну роль — второго проверяющего, ещё одного архитектора, руководителя над руководителем — значило добавить ещё одну копию той же модели с тем же незнанием задачи, только под новым названием. Переписка становилась длиннее, согласования подробнее, а определённости оставалось ровно столько, сколько было во фразе, с которой всё началось. Прибавить её роли не могли: каждая получала на вход то, что выдала предыдущая. По сути, роль менеджера получала ту же фразу-желание, какую весной вписывали в одиночную программу, и дальше эта фраза только переходила из рук в руки, обрастая документами. Разделение труда перенесли по форме, не заметив, что работает оно на содержании.

Спросим эту эпоху о том же, о чём спрашивали реку в 1849 году: знал ли старатель, где искать. Прямого ответа она не даёт. Вместо ответа она показывает три лотка, которым раздали должности: один размечает участок, другой моет, третий смотрит на дно. Движения распределены толково, никто никому не мешает, со стороны это похоже на хорошо поставленную добычу. Но лоток, которому поручено размечать, не знает, где жила. Он знает только, что его дело — размечать, и проведёт границы участка уверенно, потому что уверенность входит в его роль. Место он выберет там, где удобно, а остальные будут мыть там, где он разметил. А старатель, который должен был знать место, оставил у воды одну-две фразы и ушёл. На вопрос о знании эпоха ответила распределением обязанностей.

В этом и была сила её ответа. Когда ролей много, кажется, что знание где-то между ними: если не знает архитектор, то знает менеджер, а если не знает и менеджер, значит, требования и так ясны. Каждая роль смотрит на соседнюю, и никто не смотрит на реку.

Распределение обязанностей не заменяет знания. Оно умеет донести знание до каждого, кому оно нужно, если знание есть: тот, кто понимает, чего хочет, может раздать части работы и не держать в голове каждую мелочь. Если знания нет, распределение разносит по ролям пустые места, и каждое получает своё название. Чем стройнее схема, тем труднее разглядеть, что в её основании ничего не решено: пустое место с табличкой «архитектура» выглядит занятым.

Эпоха ролей спрашивала, кто что делает. Вопрос, кто знает, остался за пределами схемы, там же, где весной остался человек. Все перемены того времени произошли внутри лотка. Цепочка лотков работала слаженнее одного лотка. Но как ни дели движения между лотками, ни одно из них не говорит, где мыть.

3.4. Лоток с сеткой

Следующая перемена пришла не от тех, кто строил ИИ-агентов, а со стороны. Осенью 2023 года учёные из Принстона выложили SWE-bench — набор задач, по которому можно проверить, умеет ли языковая модель чинить настоящие программы. Задачи для него не придумывали. Их взяли из жизни двенадцати популярных репозиторий с открытым кодом: кто-то когда-то сообщил, что программа ведёт себя неправильно, кто-то другой исправил ошибку и приложил к исправлению тесты. Таких историй набралось больше двух тысяч. Модели давали код репозитория и описание ошибки словами и просили исправить. Потом на её правке запускали тесты из того самого исправления, которое когда-то сделал человек: те, что до исправления падали, должны были пройти, а остальные не должны были сломаться. Оценивали не объяснение словами, а то, как ведёт себя код.

Тесты, которые должны пройти. В этом скромном условии главное слово то, которого в нём нет: внешние. Весной того же года критерий готовности отдавали машине вместе с целью, и решала, закончена ли работа, та же модель, что её делала. Здесь критерий стоял вне её досягаемости. Его написали люди, которые эту ошибку однажды действительно исправили, и написали задолго до того, как модель увидела задачу. Модель могла прочесть код, переписать функцию, попробовать ещё раз, но не могла повлиять на то, что будет считаться исправлением. Впервые у настоящей задачи для ИИ-агента появилась проверка снаружи.

Разница видна на одной-единственной попытке. Модель на SWE-bench, закончив правку, ни о чём себя не спрашивала. Вопрос ей задавали тесты, а тестам было всё равно, насколько убедительной правка кажется ей самой. Спросить себя, удалась ли попытка, и получить ответ от того, кто знал задачу раньше тебя, — два разных действия, хотя со стороны оба выглядят как проверка.

Что это меняет, легко проглядеть, потому что сами попытки выглядели как прежде. Модель по-прежнему читала, правила, запускала, ошибалась. Но теперь у каждой попытки был исход: прошла или не прошла. Каждая попытка получает оценку, и попытки сходятся. Первые результаты были скромными: в статье, где авторы описали SWE-bench, лучшая из проверенных моделей справилась примерно с двумя задачами из ста. Но эти два процента были сведениями другого рода, чем утро весны 2023-го. Про то утро можно было сказать только, что финиша нет. Про два процента можно было сказать, какие именно задачи решены, а какие нет, и через полгода проверить, стало ли решённых больше.

Стало. Экзамен из настоящих ошибок настоящих программ быстро превратился в общую мерку, и новые ИИ-агенты один за другим предъявляли свои результаты на нём. В 2024 году одна компания объявила о «первом ИИ-программисте» по имени Devin, и в тот же день появился его открытый двойник, позже получивший имя OpenHands. Часть авторов SWE-bench выпустила SWE-agent, где главной находкой был не новый ум, а удобный для ИИ-агента способ обращаться с компьютером: как открывать файлы, как листать код, как запускать команды. К 2025—2026 годам ИИ-агенты перебрались в терминал разработчика — Claude

Code, Codex, — а оттуда в облако. Фоновый ИИ-агент работает в облаке без открытого сеанса: задачу отдают, и он делает её сам, пока человек занят другим или спит.

Теперь ИИ-агент — это снова ИИ-агент для программирования: он сам читает репозиторий, правит файлы, запускает тесты и коммитит. Это уже не программа с широкой целью вроде «собери всё о рынке» и не должность в штатном расписании, а лоток, сделанный под одну работу: промывать код.

Последняя перемена коснулась уже не самого ИИ-агента, а работы человека. К 2026 году в профессиональном языке появилось выражение «инженерия циклов». Цикл — обвязка, которую человек строит вокруг ИИ-агента: повторять попытки, оценивать их, решать, продолжать или остановиться. Человек пишет не промпты, а циклы. Раньше всё, что человек вкладывал в работу, умещалось в текст, обращённый к ИИ-агенту. Теперь часть вложенного живёт между попытками: в том, что запускается после каждой из них и что решает, будет ли следующая.

Так история и приходит к ночной смене 2026 года. Вечером человек описывает задачу, запускает цикл и уходит. Ночью ИИ-агент в облаке делает попытку, цикл запускает проверку, смотрит на итог и либо возвращает работу с пометкой, что она не прошла, либо переходит к следующему шагу. Попытки идут до утра, и никто на них не смотрит. К утру в облаке лежит история десятков таких попыток, и человек, открывший её, видит не одну попытку, а целую ночь чужого труда.

Если спросить, что изменилось между весной 2023 года и этой ночью, проще всего ответить: всё. Модели сильнее, инструменты удобнее, ИИ-агент дольше держит задачу и реже теряет нить. Но из всех перемен одна отличается по роду. Изменилась не надежда, а лоток: у него появилась сетка, отделяющая золото от песка. Сетка — это проверка снаружи, которую лоток не выбирал и не может передвинуть. Песок — результат, который нельзя принять; золото — тот, что прошёл сетку. Тесты SWE-bench стали первой сеткой, которую увидели все, и почти всё, что пришло после, от удобных интерфейсов до циклов, выросло вокруг неё. Все прежние перемены касались того, как лоток промывает. Здесь лоток впервые получил то, чем промытое можно проверить. В 1849 году эту работу делал сам старатель, разглядывая то, что осталось на дне. Весной 2023-го её отдали самому лотку, и он сам оценивал собственный осадок. Сетка вынесла её обратно наружу, только теперь это не человек над осадком, а тесты, которые не устают к вечеру и не обманываются блеском.

Надежда же осталась той самой, что на калифорнийских берегах: количество попыток заменит знание жилы. Больше того, сетка её подкрепила. На задачах с сеткой количество действительно работает: ещё одна ночь, ещё один прогон, и решённых становится больше. Числа на экзамене росли у всех на глазах, и легко было перенести это наблюдение на любую задачу — решить, что теперь лотку достаточно дать побольше ночей, а остальное он намоет. Тем более что цикл можно поставить вокруг любой задачи. Сетку — только вокруг той, где заранее известно, чем проверить результат.

Была ли у лотка сетка? Впервые за всю историю лихорадки ответ — да, по крайней мере там, где проверка существует. Знал ли старатель, где искать? Здесь не изменилось ничего. Сетка — свойство лотка, а не реки. Она судит о промытом, но не говорит, где мыть. Жила — место, где промывка честна, и для этого ей нужна не только сетка. Нужно, чтобы кто-то заранее решил, чего мы хотим и как устроено то, что мы строим, а сетка проверяла попытки уже на решённом. Сетка закрывает только половину условия. Старатель 2026 года стоит у реки с лучшим лотком из всех, что у него были, и знает о реке ровно столько же, сколько знал новичок на калифорнийском берегу.

Посмотрим ещё раз на задачи SWE-bench. В каждой из них вторая половина уже была. Кто-то написал, что программа ведёт себя неправильно, — значит, было известно, чего от неё хотят. Репозиторий был устроен задолго до задачи, модули разграничены, и исправление должно было уложиться в эти границы. Тесты написал человек, который ошибку однажды уже

исправил. ИИ-агенту оставалось найти путь, и здесь попытки вместе с сеткой работали в полную силу. Это была честная промывка: известно, чего хотим, известно, где искать, и есть чем отличить найденное. Лоток с сеткой впервые показал, на что способен, на реке, где жилу до него нашли другие. В этом нет упрёка экзамену: он и был задуман, чтобы проверять лоток, а не выбор реки. Но в нашей собственной задаче за нас этого никто не делает.

Там, где сетки нет, ИИ-агент 2026 года ведёт себя как AutoGPT с хорошими манерами. Изменились манеры, а не поведение. Весной 2023 года ночь без сетки кончалась заикливанием, сожжёнными кредитами и отсутствием финиша, и о провале утро сообщало само. ИИ-агент 2026 года не заикливается. Он пишет аккуратный код, разбивает работу на осмысленные коммиты с внятным описанием, и к утру на панели горят зелёные галочки. Он заканчивает не потому, что кончились деньги, а потому, что сам счёл работу готовой. Финиша, который можно принять, как не было, так и нет. Есть конец, который выбрал лоток.

Поведение — это то, что остаётся, если снять манеры. А под ними та же схема, что весной: цель словами, план, действие, наблюдение, и наблюдение оценивает тот же, кто действовал. Человек снова вне контура, только контур стал длиннее и опрятнее. Если в задаче не решено, чего мы хотим, ИИ-агент решит это сам, как решала программа весны. Разница лишь в том, что весной такой выбор был нелепым и бросался в глаза, а теперь он пристойный, грамотный и выглядит так, будто его кто-то сделал обдуманно.

Такое утро легко узнать. Двести четырнадцать изменённых файлов, и каждая деталь по отдельности разумна. Ни одного повтора, ничего, за что глаз зацепился бы с первого взгляда. Весной беду выдавало однообразие. Теперь её скрывает разнообразие: каждый файл отличается от соседнего, каждый сделан со знанием дела, и вместе они не отвечают на вопрос, которого никто не задал.

Манеры — это то, чему лоток мог научиться сам, и научился он им блестяще. Опрятность кода, уверенный тон коммитов, привычка разбивать работу на части — всё это умения лотка, и за три года они выросли больше, чем кто-нибудь ждал весной. Но знание жилы не растёт вместе с ними, потому что оно никогда не было умением лотка. Хорошие манеры не заменяют сетку. Они только делают её отсутствие незаметным. AutoGPT хотя бы заикливался у всех на виду. ИИ-агент с хорошими манерами промывает песок так чисто, что утром его трудно отличить от золота.

3.5. Продавцы лопат

У любой лихорадки есть участник, которого не видно на картине с рекой. Он не стоит в воде и не держит лотка. Он стоит за прилавком: в лагере у прииска, в городке, куда сходятся тропы с берегов, на пристани, куда приходят корабли с новичками. Он продаёт лотки и лопаты, муку и сапоги, место в повозке до реки. Всё, что старатель вычитал из своей скромной прибыли, в этой лавке записывали в доход. Современные работы историков калифорнийской лихорадки подтверждают то, что звучит как поговорка: торговцы заработали на ней гораздо больше старателей. Самым богатым человеком Калифорнии первых лет лихорадки обычно называют не того, кто намыл больше всех золота, а хозяина лавок, открытых у самых приисков.

В этом нет ни загадки, ни хитрости. Есть механика. Лоток покупают до того, как им зачерпнут первый песок. Старатель ставит на реку. Лавочник на реку не ставит ничего: его доход зависит не от того, сколько золота в воде, а от того, сколько людей пришло к воде. Пустой вечер старателя для лавки обычный торговый день. И ещё одно: тот, кто знает, где мыть, покупает инструмент один раз и работает им на своём месте. Тот, кто не знает, уходит выше по течению, потом ниже, берёт муки ещё на неделю, меняет сношенное на новое и снова идёт искать. Никто этого не замышлял, и никто не должен был замышлять. Так устроена любая лихорадка,

начиная с 1849 года: надёжнее всех зарабатывают те, кто продаёт инструмент, потому что инструмент покупают независимо от того, есть ли в реке золото.

Лопаты этой лихорадки — токены и инструмент, который их тратит. Из токенов сделана каждая промывка: каждая итерация ИИ-агента, каждый прочитанный им файл, каждая попытка, которую цикл отправил на проверку. Продавцы лопат и здесь получают своё так же, как лавочник у прииска: за лопату платят одинаково, чем бы ни кончилась ночь, принятым diff или горой песка. Весной 2023 года эту лопату было видно. Каждое обращение к модели шло отдельной строкой в счёте, и утром счёт лежал рядом с результатом ночи. Он ничему не учил, но задавал вечеру вопрос: стоит ли эта задача ночи? Вопрос неудобный и полезный. Чтобы на него ответить, приходилось хотя бы прикинуть, есть ли на этом месте что мыть.

Плоская подписка устроена иначе. За неё платят раз в месяц, и сколько ни копай, больше не заплатишь. Лопата от этого не стала бесплатной: за неё заплачено. Но она стала бесплатной на ощущение. Отдельная ночь больше ничего не стоит в ту минуту, когда человек вечером прикидывает, запускать или нет. И вечерний вопрос переворачивается. Раньше он звучал так: стоит ли задача ночи. Теперь иначе: чем бы занять ночь. Она всё равно оплачена, и оставить её пустой кажется потерей. Такую ночь «жалко не использовать».

Это повод, и повод сильный, потому что у него нет задачи, зато есть время. Под свободную ночь задача находится всегда. Чаще всего не та, где известно, что искать, а та, что давно лежит в голове недодуманной: переписать как следует, привести в порядок, наконец сделать. Её и отдают, не выяснив сперва, где копать: копают без разведки. Ночь, отданная большой промывке, кажется использованной полнее, чем ночь, после которой человек только узнал бы, где мыть. Лопата, бесплатная на ощущение, убирает ровно то, что раньше давал счёт, — паузу перед тем, как копнуть.

У лихорадки есть и публичная сторона. Счётчик токенов, на который ночью так приятно смотреть, недолго оставался личным делом. Появились таблицы, где люди сравнивают свой расход: кто сжёт больше токенов за неделю, за месяц, у кого ИИ-агенты работали дольше и шире. Наверху — тот, чей счётчик крутился быстрее всех. Рядом с его именем стоит число, и число честное: столько токенов действительно ушло. Но посмотрим, что именно оно считает. Токены тратятся на промывку, а не на золото. Они одинаково уходят на ночь, после которой утром было что принять, и на ночь, после которой принять было нечего. В таблице нет колонки «принято». Лидерборды расхода токенов меряют количество промытого песка и превращают его в статус.

Здесь таблица делает то, чего от неё никто не хотел. Тот, кто знает, где копать, обычно тратит меньше: он моет на одном месте, у него есть чем проверить промытое, и заканчивает он раньше утра. Тот, кто не знает, моет всю ночь и следующую, переходит с места на место, и его счётчик растёт быстрее. В рейтинге расхода он выше. Знание места выглядит в такой таблице бедностью, незнание — размахом. Трудно представить старателя, который гордился бы числом лотков, промытых за сезон: гордились тем, что осталось на дне. И как толпа на калифорнийском берегу, эта таблица убеждает самым своим размером.

Прочесть всё это как обвинение было бы ошибкой. Лавочник 1849 года продавал настоящие лотки и настоящую муку, и без его лавки ничего не намыл бы и тот, кто знал, где мыть. Он продавал то, о чём его просили. Те, кто продаёт токены сегодня, тоже продают вещь настоящую. Лопата не зло, и продавец не злодей. Дешёвая лопата в руках того, кто знает, где жила, — чистое благо. Промывка на известной жиле — честная работа, и чем больше попыток можно сделать там за ночь, тем лучше. Для такого человека подписка, при которой можно мыть свой участок сколько нужно и не считать каждый лоток, — подарок, а не ловушка. Лопата — полезный инструмент, если знаешь, где копать.

Всё держится на этом «если». Лавка продаёт одну и ту же лопату опытному горняку из Соноры и новичку, который неделю назад впервые увидел реку. Продать вместе с ней знание

места лавочник не может, да и не обязан. Ему не нужно знать, где золото: его товар хорош на любом берегу. Знание места никогда и не было товаром. Его нельзя отвесить, как муку, и нельзя выдать вместе со сдачей: оно бывает только у того, кто сам стоит у реки, и остаётся при нём. Это условие возвращает к 1849 году, к тому, что отличало удачливых: не лоток и не лопата, а умение знать, где искать. На прилавке его не было тогда. Нет и теперь.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, [купив полную легальную версию](#) на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.