



СТРАТЕГИЧЕСКАЯ СТАТЬЯ

Редакционная дата: 22 октября 2026 года

Кто купит сбой до того, как он случится?

Может ли Саудовская Аравия экспортировать исполнимое знание о технологических отказах?

The Incident That Ships: Can Saudi Arabia Export Executable Knowledge of Technological Failure?

Вместо пролога. В одних отраслях машину продают потому, что она работает. В другой, пока ещё не сложившейся отрасли, ценностью может стать испытание, которое умеет вывести машину из строя до того, как вместе с ней остановится работа людей. Речь не о торговле происшествиями и не о производстве кризисов. Вопрос в другом: можно ли извлечь инженерное знание из документированного сбоя и превратить его в тест, который покупатель в другой стране сумеет запустить, понять его пределы и отвергнуть продукт, если тот не выдержит проверки.

1. День, когда небольшой файл вырос вдвое

18 ноября 2025 года сеть Cloudflare стала вести себя как больной, у которого каждые пять минут то поднимается, то спадает температура. Сервисы отключались, возвращались и снова отключались. Сначала команда решила, что сеть подверглась масштабной кибератаке. Согласно последующему отчёту компании, причина оказалась страннее и прозаичнее: изменение прав доступа к базе данных удвоило содержимое файла, которым пользовалась система управления ботами (Bot Management). Файл распространился по сетевым узлам и превысил объём, на который было рассчитано программное обеспечение. Штатное обновление стало путём к широкому отказу S01.

Этот файл не выпускали раз в день под официальный пресс-релиз. Он автоматически обновлялся каждые пять минут по запросу к базе ClickHouse. Часть узлов уже получила новые права, часть ещё нет, поэтому система попеременно распространяла то корректный, то удвоенный файл. Временные восстановления навели наблюдателей на мысль о колеблющемся враждебном трафике. Когда большинство серверов стало выдавать дефектную версию, отказ стабилизировался. К 14:30 по всемирному координированному времени (Coordinated Universal Time; UTC) основной трафик по большей части восстановился, но о нормальной работе всех систем компания сообщила лишь в 17:06 S01.

Пятью месяцами ранее, 12 июня 2025 года, глобальный сбой затронул Google Cloud. И здесь всё началось не со взлома или атаки. Обновление политики квот внесло данные с пустыми полями, они разошлись по миру и задействовали программный путь, не способный обработать нулевой указатель (Null Pointer). Даже после обнаружения причины волна перезапусков замедлила восстановление в отдельных регионах: попытки вернуться в строй создали нагрузку на зависимую базу данных S02.

Компании и программы различались, однако у двух происшествий есть важное инженерное родство: **система не спасовала перед исключительной катастрофой; она сочла неожиданный ввод невозможным, а затем распространила последствия этого предположения с машинной скоростью и уверенностью.**

После сбоев остаются длинные отчёты. Инженеры их читают, обсуждают на совещаниях, а затем отправляют в архив человеческого опыта. Через несколько месяцев та же семья ошибок может возникнуть в другой организации, даже если её специалисты читали первый отчёт. Отсюда начинается стратегическая идея: что, если документированное происшествие превратить в исполнимый контракт отказа (Executable Failure Contract) — единицу знания, которая задаёт условие возникновения сбоя, недопустимое поведение, выявляющий его опыт и доказательство результата?

Тогда на экспорт пойдёт не просто средство наблюдения за отказами и не разовая консультация. Это будет тест, который инженер в Эр-Рияде разрабатывает, оператор в Сингапуре запускает, а покупатель во Франкфурте проверяет, не полагаясь на слово продавца. Научный и коммерческий вопрос состоит не в красоте идеи. Он в том, можно ли перенести логику сбоя между разными системами, не потеряв её смысл, и заплатит ли иностранный клиент за такой перенос второй раз.

2. Стране не нужно изобретать происшествие, чтобы основать отрасль

Идея кажется странной, если считать, что программная индустрия продаёт только то, что происходит при нормальной работе. Успешное приложение оказывает услугу, микросхема производит вычисление, сеть передаёт сообщение. Однако вокруг этих продуктов существует тихая экономика того, чего происходить не должно. Компания может ежемесячно платить за тест, который ничего не обнаружил, если у неё есть убедительные инженерные основания верить: опыт действительно отличает исправную работу от выявляемого отказа.

Этот рынок уже существует. Amazon Web Services (AWS) предлагает сервис управляемого внесения отказов (AWS Fault Injection Service; FIS). С 2023 года в нём действует библиотека готовых сценариев, описанная в опубликованном в 2024 году руководстве. Опыты могут имитировать отказ зоны доступности (Availability Zone) или последствия нарушения связи и останавливаться при выходе за установленные пределы [S06S08](#). В мае 2025 года AWS показала генерацию шаблонов экспериментов из естественно-языковых описаний с помощью Amazon Bedrock и больших языковых моделей (Large Language Models; LLMs) [S07](#).

Поэтому саудовский проект не вправе утверждать, будто он изобрёл инженерию хаоса (Chaos Engineering) или первым догадался превращать описание происшествия в опыт. Конкуренты уже есть, причём некоторые владеют платформой, которую тестирует клиент. Возможное окно для продукта уже и сложнее: **перевести смысловой механизм отказа в тест, переносимый между средами, и приложить проверяемое свидетельство наблюдаемого поведения, не привязывая проверку к одному облачному движку и не ограничиваясь случайным отключением сервера.**

Рынок может не признать это различие. Облачные поставщики способны включить такую функцию в подписку бесплатно, а покупатели — предпочесть специализированное решение для каждой платформы. Инженерно интересный пробел нельзя выдавать за доказанную инвестиционную возможность. Настоящий проект начинается с выяснения, кто заплатит за его заполнение, а не с эффектного англоязычного названия.

3. От отчёта о происшествии к «грамматике отказа»

Отчёты об отказах написаны для людей: в них есть хронология, решения, допущения, подробности распространения, первопричина (Root Cause) и меры исправления. Компьютер не поставит безопасный опыт, если сообщить ему лишь, что команда

«улучшит обработку ошибки». Ему нужны конкретное условие, исходное состояние, разрешённое действие, наблюдаемый результат, способ остановки и отката.

Предлагается считать отчёт сырьём, а не готовой программой. Он проходит пять преобразований:

1. **Отделение наблюдения от толкования.** Какие факты объявил владелец системы? Что осталось выводом? Какая часть непереносима без недостающей спецификации?
2. **Выделение причинного шаблона.** Например: изменение прав чтения в хранилище порождает дублированные строки, из-за них растёт общая структура конфигурации, а считыватель не выдерживает её размера. Это описание семейства поведения, а не копия устройства конкретной компании.
3. **Формулировка функционального инварианта (Invariant).** Например: «Смысловое дублирование записей свойств не должно останавливать основной путь обслуживания». Но и этот инвариант требует оговорки: безопасно ли удалять повторы в данном продукте? Порядок или повтор записи может быть частью её смысла.
4. **Сборка ограниченного эксперимента.** Компилятор (Compiler) создаёт синтетические входные данные в изолированной среде, задаёт ожидаемый исход, условия остановки (Stop Conditions) и откат (Rollback).
5. **Выпуск журнала свидетельств.** Какая версия программы проверялась? Каково случайное зерно, если оно использовалось? Какие случаи испытывались? Что показали распределённые трассы (Distributed Traces), журналы и метрики?

Получившуюся единицу можно представить структурированным файлом IncidentSpec с полями evidence, preconditions, stimulus, oracle, stop, rollback, environment и limitations. Это исследовательское предложение, а не принятый стандарт и не название существующего саудовского продукта. Поле oracle обозначает тестовый оракул — правило, отличающее успех от отказа. Худший недостаток умного теста не в том, что он не вызвал сбой, а в том, что он вызвал его и отпраздновал успех опыта, не зная правильного результата.

4. Почему недостаточно устраивать оборудованию искусственные землетрясения

Обычные облачные сбои вносят разрывом сети, увеличением задержки (Latency), остановкой контейнера или исчерпанием процессора. Такие проверки полезны, но сами по себе не воспроизводят события, нарушившие работу в двух первых историях.

Там причиной были слишком большой файл и пропущенное поле в политике, которая распространялась по всему миру. Система может выдержать тысячи отключений серверов и упасть при импорте одной лишней строки из базы данных.

В информатике различают отказ ресурса (Resource Fault) и смысловой отказ (Semantic Fault). Первый спрашивает, что случится без сервера. Второй — что произойдёт, если сервер исправно и быстро прочтёт данные, допустимые по форме, но нарушающие скрытое допущение программы. Такая проверка может касаться повреждения данных (Data Corruption), дрейфа схемы (Schema Drift), расхождения версий (Version Skew), нарушения идемпотентности (Idempotency Failure) или шторма повторных попыток (Retry Storm).

Не всякий плохой ввод следует обрабатывать одинаково. В финансовом сервисе нельзя всякую ошибку авторизации превращать в разрешение доступа ради непрерывности. В общедоступной службе чтения условное продолжение работы (Fail Open) может быть предпочтительнее блокировки всех пользователей. **Функциональный инвариант выводят из назначения и границ сервиса, а не копируют из новостной статьи.** В этом и состоит важнейшая профессиональная услуга рядом с продуктом: перевод смысла деятельности в измеримые условия, а не слов в конфигурационный файл.

У программного тестирования есть и другие средства. Тестирование на основе свойств (Property-Based Testing) генерирует разнообразные входы в поиске опровержения инварианта. Метаморфное тестирование (Metamorphic Testing) сравнивает результаты до и после преобразования входа, когда правильный ответ для каждого случая заранее неизвестен. Дифференциальное тестирование (Differential Testing) сопоставляет две реализации одного правила. При параллельных транзакциях задача усложняется: каждый отдельный ответ может выглядеть верным, тогда как вся история событий нарушает гарантию согласованности (Consistency Guarantee). Проект Jepsen много лет систематически испытывает реальные распределённые системы под отказами и проверяет журналы операций S09.

Перед нами не творческое агентство, которое придумывает худший день компании и просит языковую модель его нарисовать. Нужны исследователи семантики языков программирования (Programming Language Semantics), протоколов распределённых систем (Distributed Protocols) и эксплуатационные инженеры, понимающие, что условие «не упади» значительно слабее требования «выполни правильную операцию без повторов и потерь».

5. Три компании — и кто пострадал за пределами серверной?

Разговор об удвоенном файле, индексах и распределённых версиях можно принять за развлечение программиста с чашкой кофе. Иначе выглядит происшествие в американской AT&T Mobility 22 февраля 2024 года. Бюро общественной безопасности и внутренней безопасности Федеральной комиссии по связи США (Federal Communications Commission; FCC) сообщило, что после неверной настройки сетевого элемента широкое отключение сотовой связи продолжалось не менее двенадцати часов. По полученным от оператора данным были заблокированы более **92 миллионов голосовых вызовов** и свыше **25 тысяч попыток дозвониться в службу 911** S03.

Мы не придумываем пациента, который не смог вызвать скорую, и не устанавливаем судьбу каждого несостоявшегося звонка. Числа и вид услуги уже делают вопрос общественным. Изменение технического элемента повлияло на способность миллионов людей связаться друг с другом. Переносимый урок — не само число вызовов для какой-либо саудовской сети, а необходимость до выпуска проверять распространение единичного изменения и реакцию системы при восстановлении. Вред не заканчивается обнаружением ошибки: множество устройств, одновременно возвращающихся в сеть, способно продлить перегрузку уже после исправления.

Три происшествия различны: удвоенный файл свойств, квота с пустым полем и неверно настроенный сетевой элемент. Но все они ведут к одному вопросу: **когда локальная управляющая информация превращается в быстрое коллективное решение, которое трудно отменить?** Их сопоставление не доказывает, что одна платформа выявит все три. Зато оно создаёт банк научных гипотез, более честный, чем общее обещание «наш искусственный интеллект предсказывает атаки и сбои».

6. Предлагаемая фабрика хранит не происшествия, а переводит условия их возникновения

Если бы предстояло создать первый саудовский продукт, стоило бы начать не с огромной платформы и панели с картой мира и лозунгами готовности. Началом мог бы стать компилятор из происшествия в тест (Incident-to-Test Compiler), работающий с открытым реестром происшествий и тремя классами программ, которые команда хорошо понимает. К системе клиента его напрямую не подключают: сначала запрашивают абстрактную модель интерфейсов сервиса, существенные для клиента свойства и границы разрешённой среды.

Этап искусственного интеллекта (Artificial Intelligence; AI) выделяет кандидатов: «возможная причина — размер файла», «считыватель может не обрабатывать пустые поля», «повторные попытки могут усиливать нагрузку». К каждому утверждению прилагается место в отчёте и статус: подтверждено источником, вероятно или непереносимо. Языковая модель здесь — **помощник по извлечению гипотез (Hypothesis Extraction)**, а не судья, разрешающий запуск. Она может перепутать имена полей или выдумать причинную связь, поэтому её вывод нельзя непосредственно превращать в команды, влияющие на реальные сервисы.

Затем гипотеза переводится в промежуточное представление (Intermediate Representation; IR). Из него могут получиться адаптер для опытов в Kubernetes, адаптер для проверки программного интерфейса (Application Programming Interface; API) и проверка автономной библиотеки без инфраструктуры. Коммерческая подписка не имеет ценности, пока смысл не сохраняется между этими формами. Остановка группы контейнеров в первой среде может внешне напоминать отказ во второй, но не содержать общего состояния, которое сделало исходное происшествие широким.

Результаты собирают из сервисных метрик, журнала операций и сквозных трасс. OpenTelemetry предлагает один из путей к общим интерфейсам метрик (Metrics), журналов (Logs) и трасс (Traces), как поясняет Cloud Native Computing Foundation (CNCF) [S11](#). Саудовскому продукту не нужен новый национальный протокол наблюдения. Оригинальное знание может заключаться в конструкции опыта и пределах его толкования, а открытые стандарты будут переносить свидетельства.

Наконец, «паспорт свидетельств выполнения» (Execution Evidence Passport) включает версию пакета и среды, внесённые изменения, вывод тестового оракула, причину остановки и границы допустимого вывода. Это предлагаемое исследовательское понятие, а не знак независимой сертификации. Цифровая подпись (Digital Signature) защищает запись от подмены, но не делает вывод верным при слабом правиле оценки. Полностью документированный опыт тоже способен исчерпывающе ответить не на тот вопрос.

Предлагаемая цепочка от происхождения к тесту

Пять преобразований знания; каждое требует свидетельств и проверки в конкретном приложении



Это концептуальная, не построенная платформа: нужно сохранить причинную неопределённость, проверить смысл и ограничить запуск. Независимый покупатель должен иметь возможность повторить результат либо отклонить перенос.

Концептуальная схема компилятора тестов отказа: от документированного отчёта к пакету эксперимента и проверяемому результату

Рисунок 2. Предлагаемая архитектура продукта, а не опубликованная или внедрённая саудовская система. Стрелки обозначают этапы преобразования знания, требующие человеческой проверки; они не доказывают, что полная автоматизация возможна или безопасна во всех отраслях S06S07S09S11.

7. Что мы сделали в небольшой лаборатории, кроме схемы?

Чтобы тезис не превратился в доклад для конференции, мы выполнили небольшой вычислительный опыт с открыто обозначенными пределами. Были вручную составлены **16 синтетических тестовых контрактов** из четырёх семейств: политики с отсутствующими полями, повторяющиеся пакеты свойств сверх ограничения объёма, устаревшие или неподтверждённые версии конфигурации и бесконечные попытки соединения. Контракты запустили на **четырёх искусственных реализациях**: уязвимой, проверяющей только входные данные, контролирующей только границы ресурсов и объединяющей четыре требуемых барьера.

Программа действительно выполнила **64 тестовых случая** и записала для каждого ожидаемый и наблюдаемый результат. Уязвимая реализация совпала с ожидаемым исходом в **7 случаях из 16**, реализация с проверкой входа — в **12**, с ограничениями ресурсов — в **11**, объединённая — во всех **16** S13. Это результаты кода, написанного для

иллюстрации, а не доли обнаруженных сбоев в реальных компаниях. Искусственный интеллект для извлечения случаев из отчётов не применялся: контракты созданы вручную, чтобы проверить саму возможность их представления и исполнения.



Результаты 64 синтетических опытов с контрактами отказа на четырёх тестовых реализациях

Рисунок 1. Число случаев, когда вывод каждой реализации совпал с объявленным тестовым оракулом; в каждом семействе было по четыре случая. Доли не выражают ни коммерческую способность обнаруживать сбои, ни вероятность их возникновения. Код, наборы случаев и результаты находятся в отдельном приложении [S13](#).

Опыт не показывает, будто большее число защит всегда повышает устойчивость. Такой вывод был бы круговым: четвёртая реализация написана так, чтобы проходить составленные нами контракты. Результат скромнее и полезнее: **функциональное описание можно превратить во входные данные, правило решения и воспроизводимый журнал результатов, сохранив различие между отказом системы и неспособностью теста понять систему.** Переход от этой ограниченной игры к производственной базе данных, платёжной платформе или телекоммуникационной сети потребует других опытов, договоров и определений успеха.

8. Что может купить клиент: воспроизводимый отказ без копирования всей фабрики

Иностранный клиент не купит у стартапа «историю интернета»: её можно прочитать бесплатно. Не заплатит он и за пересказ отчёта, который его авторы изложили лучше. Деньги появятся, если пакет сокращает трудную работу: безопасно представляет происшествие в архитектуре клиента и связывает исход опыта с обязательствами его сервиса. Но стоимость интеграции (Integration Cost) способна поглотить лицензионную плату, если каждый заказ требует двух месяцев настройки.

У идеи есть три разных начальных продукта, и продавать их как одно нельзя:

Предлагаемый продукт	Что передаёт продавец	Возможный покупатель	Самый трудный коммерческий вопрос
Пакеты контрактов отказа (Failure Contract Packs)	Изменяемые и воспроизводимые спецификации и тесты	Команды платформ и облачных сервисов	Почему команда не напишет их сама?
Межсредовый компилятор тестов (Cross-Runtime Test Compiler)	Адаптеры и проверки смысловой эквивалентности между средами	Компании, работающие с несколькими операторами или облаками	Сколько стоит поддержка каждой новой среды?
Сервис воспроизведения свидетельств (Evidence Replay Service)	Прослеживаемый, проверяемый и повторяемый между версиями результат	Покупатель, которому нужно регулярно подтверждать эффект исправления	Доверяет ли клиент оракулу и данным?

Это **предлагаемые категории**, а не названия созданных компаний или проданных услуг. Начать можно с первого пакета и перейти ко второму, если клиенты докажут потребность в переносе между средами. Попытка сразу создать всеобъемлющий продукт, понимающий всё, скорее породит дорогой консалтинг под маской программной платформы.

9. Саудовская Аравия за пределами выставочного зала

В апреле 2025 года Управление по развитию экспорта Саудовской Аравии (Saudi Export Development Authority; Saudi Exports) участвовало в GITEX AFRICA в Марракеше с павильоном **28 национальных компаний технологического и телекоммуникационного сектора**, сообщило Saudi Press Agency (SPA) [S04](#). В июле 2026 года саудовские компании

участвовали в LEAP East в Гонконге; среди объявленных связей было подписание меморандума между Elm и Huawei о сотрудничестве в искусственном интеллекте и цифровой инфраструктуре S05.

Это реальные признаки международных каналов общения и продвижения. Однако они не являются реестром закупок саудовского продукта по устойчивости и не подтверждают регулярную выручку от иностранного клиента. Выставка помогает встретить покупателя. Экспортная отрасль возникает, когда независимый покупатель подписывает второй договор, потому что уже не хочет обходиться без первого продукта.

Чтобы занять эту нишу, Саудовской Аравии не требуется производить все элементы мировой программной индустрии. В стране велик эксплуатационный рынок, есть потенциальные пользователи в вычислениях, связи, услугах, платежах и логистике, развиваются исследовательские и инженерные кадры. Но называть эту среду «идеальной лабораторией» без проверки нельзя. Местный клиент может умело вести крупные проекты, но не разрешать экспериментальные проверки своих сервисов; продукт может остаться набором трудно повторяемых индивидуальных заказов. Готовность платить, инженерное разрешение на тест, права на данные происшествия и предельная стоимость нового клиента не выводятся из общих национальных преимуществ.

Опыт региона Персидского залива со сложными нагрузками может помочь создавать полезные случаи для цифровых цепочек поставок, сетевой синхронизации, тепловой нагрузки вычислений и смены каналов связи. Переход от сложной реальности к общему знанию правомерен лишь при работе с открытыми данными или разрешёнными средами, без чувствительных схем объектов и деталей доступа, готовых стать списком целей для нападающего.

10. Коммерческая модель выдержит более суровый экзамен, чем код

Местная компания способна построить красивую платформу, исполняющую сотню контрактов, а затем обнаружить, что послепродажное обслуживание дороже лицензии. Экспортируемому программному продукту нужна экономика единицы продукта (Unit Economics), в которой регулярная выручка не смешивается с разовым проектом разработки. Единственный опыт может потребовать нового понимания транзакций, интерфейсов и правил безопасности клиента, юридической проверки, документации на его языке и сопровождения запуска — и всё это изменится после обновления системы.

Вместо предположений о размере рынка и средней мировой подписке основателю следует измерять у настоящих пилотных клиентов четыре показателя: **часы настройки на клиента, долю контрактов без участия специалиста, стоимость поддержки каждой новой версии и долю оплаченного продления за пределами Саудовской Аравии.** Фраза «у нас есть международный клиент» недостаточна. Консультация в другой стране — экспорт профессиональной услуги, но не обязательно глубокотехнологичный программный продукт с тиражируемой экономикой.

Покупатель может приблизительно оценить верхнюю границу годовой ценности тестов: число семейств отказов, которые они способны предотвратить до выхода к пользователю, умножается на реальную вероятность и предотвращаемый ущерб; затем вычитаются цена, стоимость испытаний и их влияние на разработку. Вероятности редких сбоев и контрфактические потери установить трудно. Такая формула помогает думать о закупке, но подставлять в неё вымышленные числа и обещать гарантированную отдачу от инвестиций (Return on Investment; ROI) нельзя.

Есть и более глубокая проблема стимулов. Если продавать «успех теста», выгодно писать проверки, которые легко пройти. Если продавать «число найденных сбоев», выгодно делать случаи всё причудливее и дальше от реальности. Поэтому нужны показатели, которыми труднее манипулировать: слепые тесты, не использованные при разработке сервиса; контрпримеры, которые система не должна отвергать; сравнение с существующими средствами; полная стоимость запуска. Это не управление поверх работы, а качество самого товара.

11. Пределы искусственного интеллекта: неопределённость оставляет место и новизне, и выдумке

Языковая модель может быстро прочитать тысячи отчётов и предложить первые семейства гипотез. Модели генерации кода (Code Generation Models) способны написать адаптер или генератор случаев и запустить его в закрытой вычислительной лаборатории. Но **отчёт о происшествии не является полной спецификацией отказавшей системы.** В нём может отсутствовать внутренний интерфейс, чувствительные данные могут быть удалены, а общая формулировка — скрывать порядок событий. То, что названо «причиной», иногда оказывается временным совпадением или кратким изложением последнего управленческого решения.

Поэтому компилятор следует оценивать независимо на трёх уровнях: привязка к доказательству в отчёте (Evidence Grounding), корректность исполнимой формы (Executable Validity) и верность оценки поведения в новых системах (Cross-System

Transfer Validity). Первый уровень не доказывает второй, а исправный код не гарантирует третий. Тест может работать без ошибок и неверно судить допустимое поведение.

Здесь проявляется одна из самых выразительных парадоксальных черт искусственного интеллекта: сравнительно легко предложить машине, **что должно сломаться**; гораздо труднее научить её понимать, **что обязано остаться верным**, когда условия меняются. Это задача фундаментальной информатики — от семантики и согласованности до темпоральной логики (Temporal Logic) и моделирования состояний (State Modeling). Чем проще становится генерация кода, тем выше ценность вопроса, который человек и машина вместе формулируют так, чтобы его можно было опровергнуть.

12. Уязвимость идеи, которую нельзя скрывать от инвестора

Сильнейшее возражение состоит не в том, что идея плоха, а в том, что она **может оказаться функцией существующих инструментов, а не самостоятельной компанией**. Если облачный сервис способен сгенерировать опыт, исполнить его и показать результат внутри платформы клиента, зачем покупать внешний компилятор? Если Jepsen много лет независимо проверяет согласованность распределённых систем, где именно находится новый пробел? Если отчёт не раскрывает частные спецификации, вызвавшие отказ, сколько будет стоить их повторное открытие у каждого клиента?

Ответом не может быть заявление о превосходстве. Нужен коммерческий опыт, различающий три исхода: специализированные пакеты с ограниченной поддержкой могут победить; качественная инженерная консультация может оказаться ценной, но не масштабироваться как программа; оба варианта могут проиграть бесплатным инструментам или сервисам крупных платформ. Третий исход — не национальное поражение, а сбережённые средства и время для другой задачи.

Проверка входа на рынок проста по конструкции и трудна по существу: небольшая саудовская команда, два или три пакета по семействам публичных происшествий, два местных клиента, разрешивших безопасные опыты, а затем клиенты из других стран, не обязанные быть любезными к стране или основателю. Для каждого измеряются время установки, число изменённых строк кода, обнаружение дефекта сверх нынешнего набора, сохранение пользы через две версии и повторная оплата. Контрольный случай должен идти первым: клиент уже хорошо проходит тесты и, возможно, не нуждается в продукте.

13. Будущее, которое можно измерить, а не нарисовать на плакате

Для 2026–2035 годов можно представить три условных исхода. В первом идея поглощается продуктами облачных поставщиков, и разумнее продавать им специализированные адаптеры или передовые услуги тестирования, чем конкурировать за общую платформу. Во втором возникает рынок независимых свидетельств для отраслей с несколькими поставщиками; самостоятельный продукт умеет переносить тесты и доказывать их границы. В третьем работа остаётся разрозненным консалтингом — полезным инженерным знанием, недостаточным для отрасли с регулярной программной выручкой.

Вероятности этих исходов не оценивались. У каждого есть наблюдаемый признак: зависимость покупателей от встроенных инструментов, рост многопоставочных договоров или неизменно высокие часы настройки. Нельзя сначала написать финал, а потом отбирать новости, которые сделают его судьбой.

Главная национальная возможность здесь глубже, чем компания, применяющая искусственный интеллект. Это создание **исполнимого интеллектуального актива**: мир уже сталкивался с задачами, мы переводим их в переносимые, запускаемые и проверяемые тесты, а затем учимся на различиях между экспериментом и действительностью. Саудовская команда получает исследовательский опыт там, где слова не заменяют доказательство, а клиент — товар, метафорам которого не требуется верить.

Обычная фабрика принимает сырьё и выпускает деталь, которую можно взвесить. Предлагаемая фабрика может принимать отчёт о происшествии и выпускать условие, которое можно опровергнуть. Если это условие удастся многократно продать независимым покупателям, перед нами, возможно, зерно новой отрасли. Если нет, останется полезный ответ на вопрос, который не решают громкие названия: **какая часть знания об отказе пересекает границу, не теряя смысла?**

Источники и литература

1. Cloudflare (Matthew Prince), *Cloudflare outage on November 18, 2025*, 18 November 2025.
<https://blog.cloudflare.com/18-november-2025-outage/>
2. Google Cloud, *Incident details: June 12, 2025 Google Cloud service outage*, original post-incident timeline and root-cause updates, 2025.
<https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW>
3. Federal Communications Commission, Public Safety and Homeland Security Bureau, *February 22, 2024 AT&T Mobility Network Outage*, July 22, 2024.

<https://docs.fcc.gov/public/attachments/DOC-404150A1.pdf>

4. Saudi Press Agency, *‘Made in Saudi’ Pavilion at GITEX AFRICA Showcases 28 Leading Tech Companies*, 14 April 2025. <https://www.spa.gov.sa/en/N2297562>
5. Saudi Press Agency, *Elm Signs MoU with Huawei to Advance Cooperation in AI and Digital Infrastructure*, 9 July 2026. <https://www.spa.gov.sa/en/N2630981>
6. AWS, *Bootstrap your chaos engineering journey with AWS Fault Injection Service Scenarios Library*, 1 October 2024. <https://aws.amazon.com/blogs/mt/bootstrap-your-chaos-engineering-journey-with-aws-fault-injection-service-scenarios-library/>
7. AWS, *Chaos engineering made clear: Generate AWS FIS experiments using natural language through Amazon Bedrock*, 13 May 2025. <https://aws.amazon.com/blogs/publicsector/chaos-engineering-made-clear-generate-aws-fis-experiments-using-natural-language-through-amazon-bedrock/>
8. AWS, *Stop conditions for AWS FIS*, official documentation. <https://docs.aws.amazon.com/fis/latest/userguide/stop-conditions.html>
9. Jepsen, *Analyses and Research Ethics*, author-operated research and testing documentation. <https://jepsen.io/analyses> and <https://jepsen.io/analyses/ethics>
10. Principles of Chaos Engineering, *Principles of Chaos Engineering*. <https://principlesofchaos.org/>
11. Cloud Native Computing Foundation, *OpenTelemetry has graduated... Now what?*, 24 July 2026. <https://www.cncf.io/blog/2026/07/24/opentelemetry-has-graduated-now-what/>
12. AWS, *REL12-BP04 Test resiliency using chaos engineering*, AWS Well-Architected Reliability Pillar. https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar/rel_testing_resiliency_failure_injection_resiliency.html
13. Author-supplied materials, *Executable incident contract toy demonstration*. See `analysis/reproduce.py`, `data/contract_cases.json`, `data/contract_test_results.csv`, `data/contract_summary.csv`, and `data/execution_meta.json` in the article package. These are explicitly artificial inputs and outputs, not observed Saudi or other operators' incidents.

Ключевые понятия

Исполнимый контракт отказа (Executable Failure Contract); компилятор из происшествия в тест (Incident-to-Test Compiler); функциональный инвариант (Invariant); тестовый оракул (Test Oracle); тестирование на основе свойств (Property-Based Testing); дифференциальное тестирование (Differential Testing); метаморфное тестирование (Metamorphic Testing); шторм повторных попыток (Retry Storm); распределённые трассы (Distributed Traces); предлагаемый паспорт свидетельств выполнения (Execution Evidence Passport); экономика единицы продукта (Unit Economics); переносимая ценность (Transferable Value).

Раскрытие: редактирование и анализ выполнены при содействии искусственного интеллекта с опорой на открытые источники. Интервью и испытания производственных систем не проводились; статья и сопроводительная работа не проходили независимого рецензирования. Предлагаемые понятия не являются опубликованными продуктами или стандартами.