



СТРАТЕГИЧЕСКАЯ СТАТЬЯ

Редакционная дата: 18 октября 2026 года

Инженерия, которая остаётся после ухода создателей

Почему мы починили компьютер в 24 миллиардах километров от Земли, а владельцам современных автомобилей пришлось создать организацию, чтобы ремонтировать собственную покупку?

The Reproduction Boundary: Engineering Saudi Technology to Outlive Its Original Maker

Вступление: технологический суверенитет определяется не только местом, где стоит сервер, и не только именем в счёте за оборудование. Эта статья предлагает практическое и проверяемое определение: до какого этапа организация способна сама воспроизвести свою программу, запустить её, исправить и доказать её целостность, если компания-создатель исчезнет? В этом исследовании ремонт зонда (Voyager 1), сообщества владельцев автомобилей (Fisker Ocean) и велосипедов (VanMoof), а также дисциплина воспроизводимых сборок (Reproducible Builds) складываются в предложение о будущей саудовской отрасли, которая продаёт не одну лишь технологию, но и способность передать её тем, кому через годы придётся её чинить.

Границы доказательств: приведённые ниже факты о зонде, транспортных средствах и технических решениях датированы и опираются на первичные заявления; отдельно указано, чего они не доказывают. «Граница воспроизводимости» и «испытание технологического наследования» — аналитические предложения этой статьи, а не утверждённые саудовские стандарты. Мы действительно дважды собрали небольшой пакет и сравнили его

хеши, а также детерминированно проверили искусственное дерево зависимостей. В результатах нет данных саудовских систем или вероятностных оценок конфликта.

1. Его можно было починить, но до него невозможно добраться

20 апреля 2024 года инженеры Лаборатории реактивного движения (Jet Propulsion Laboratory; JPL) подтвердили, что компьютер зонда (Voyager 1) вновь передаёт читаемые инженерные данные после нескольких месяцев непонятных сигналов. В тот день аппарат находился более чем в **24 миллиардах километров** от Земли. Сигнал шёл до него около **22 с половиной часов** — и столько же обратно. [S01](#)

Туда нельзя было отправить специалиста, заменить неисправный чип или приобрести продление гарантии ещё на три года. Диагностика показала, что один из чипов памяти подсистемы полётных данных (Flight Data Subsystem; FDS) перестал работать исправно. Команда решила перенести части программного кода в другие области памяти. Единого свободного участка для всего повреждённого сегмента не нашлось, поэтому его пришлось разделить и переписать связи между частями. Сначала инженеры испытали фрагмент, возвращавший инженерные данные, и сообщили об успехе; остальные научные данные были восстановлены позднее, в несколько этапов. [S01S02](#)

Это не история о бессмертной машине. Зонд стар, его возможности сокращаются, а для ремонта потребовались редкие знания и исключительные средства связи. И это не эксперимент, доказывающий независимость космического проекта от поставщиков. Но он обнаруживает инженерную истину, достойную надписи над входом в технологическую лабораторию: **ремонтпригодность вещи не растёт прямо пропорционально её физической близости.**

В том же году некоторые владельцы электромобилей (Fisker Ocean) столкнулись с другим видом дистанции. Автомобиль стоял у дома, его материальные компоненты были под рукой. Но в 2024 году производитель вступил в процедуру банкротства, и вместе с этим обнажились трудные вопросы о цифровых сервисах и технических инструментах, от которых зависели отдельные аспекты обслуживания и работы функций автомобиля. **26 июля 2024 года** Ассоциация владельцев (Fisker Owners Association; FOA) подала заявление в дело о банкротстве в американском суде. В нём она объяснила необходимость сохранить жизненно важные облачные сервисы, доступ к программной архитектуре автомобилей, инструментам сервисных техников и запасным частям. Это позиция заинтересованной стороны, а не судебное решение о том, что все эти сервисы уже прекратились или автомобили перестали быть пригодными к езде. [S08](#)

Владельцы сделали нечто, на что стоит обратить внимание каждому, кто закупает тысячи процессоров или подписывает контракт на национальную систему. Они не ограничились форумом для жалоб: создали ассоциацию и привлекли инженеров. В экосистеме ассоциации появилось альтернативное приложение (OceanLink Pro), разработанное компанией (Undercurrent Inc.), чтобы вернуть часть диагностики, управления и связи — в пределах доступных устройств, программ и прав. Страница в (App Store) называет его официальным приложением для членов (FOA) и перечисляет заявленные функции, но это описание разработчика нельзя считать независимым доказательством того, что все функции каждого автомобиля восстановлены полностью или что каждое обновление безопасно. [S09](#)

Перед нами парадокс, который трудно классифицировать привычному техническому бухгалтеру: **зонд, до которого нельзя дотянуться рукой, остался ремонтпригодным; владельцам автомобиля, который стоит рядом, пришлось стать чем-то вроде изготовившей его компании.**

Здесь дистанция измеряется не километрами. Это расстояние между владением продуктом и владением знаниями и инструментами, способными заново создать утраченную функцию.

2. Срок жизни создателя, вещи и языка их общения

Обратимся к третьей истории — не потому, что она похожа на автомобильную, а потому, что показывает иное время технологии.

17 июля 2023 года Окружной суд Амстердама (District Court of Amsterdam) объявил банкротами нидерландские юридические лица производителя электровелосипедов (VanMoof). Опубликованный отчёт управляющих подтверждает решение суда, поэтому опираться на предшествовавшие объявлению слухи не требуется. [S03](#)

В те тревожные дни конкурент — компания (Cowboy) — выпустил приложение (Bikey). В исходном описании на странице приложения на первый план была вынесена возможность для владельцев велосипедов (VanMoof S3/X3) сохранить цифровой ключ на телефоне и использовать его, если сервис исходного производителя прекратит работу; в истории версий позднее появилась поддержка (A5/S5). Страница не устанавливает полного долгосрочного охвата всех моделей и не доказывает, что при падении серверов отказал бы каждый велосипед или что приложение решило все задачи ремонта. [S04](#)

Однако компания не исчезла бесследно. В 2023 году новый владелец приобрёл активы бренда, а в феврале 2024 года (VanMoof) опубликовала архив инструкций по ремонту и технических чертежей для некоторых старых моделей. В объявлении говорилось, что материалы, прежде доступные лишь отдельным мастерским, стали общедоступными,

причём некоторые файлы пришлось переводить с китайского. Это был шаг по передаче знания из заводских стен будущим ремонтникам велосипеда, но не доказательство того, что открыты весь программный код, ключи аутентификации или все запасные части. S05S06

Затем, в 2026 году, проявился ещё один вид утраты, для которого банкротство не нужно: **старение протокола.**

В разъяснениях о сворачивании сетей второго поколения (2G Network Phase-out) (VanMoof) сообщила, что на старых моделях сокращение этих сетей затрагивает противоугонное отслеживание и удалённую связь. Тем, кто хотел сохранить определённую услугу отслеживания в рамках решения, согласованного с операторами, предложили активировать подписку (Ride Pro) до **1 апреля 2026 года**. Особенно важно, что компания прямо указала: **езда, запираение и прочие автономные функции продолжают работать**. Поэтому писать, будто «велосипеды перестали работать», было бы неверно. S07

Итак, три срока жизни не обязаны совпадать: срок компании, продающей продукт; срок материальной вещи, на которой человек сидит или едет; и срок технического языка, на котором продукт разговаривает с миром, — от радиочастоты до криптографического пакета и программного интерфейса (Application Programming Interface; API).

Велосипед способен пережить банкротство своего производителя. Производитель может вернуться на рынок, тогда как сеть связи, заложенная в конструкцию старой модели, исчезнет. Покупатель может владеть каждым винтом устройства, но не сумеет воспроизвести даже один программный компонент, созданный во времена доступного компилятора и действующей лицензии.

Здесь предмет исследования становится глубже, чем «зависимость от поставщика». Мы спрашиваем о **наследуемости технологии**: может ли функция перейти к другому поколению инженеров, аппаратуры и программ, как музыкальный инструмент переходит к музыканту, который умеет натягивать его струны?

3. Понятие статьи: граница воспроизводимости

Я предлагаю термин **граница воспроизводимости (Reproduction Boundary)** для обозначения последнего этапа, который владелец цифрового сервиса способен пройти самостоятельно, законно и с проверяемым результатом, прежде чем ему потребуется актив, решение или знание, монопольно контролируемое другим поставщиком.

Проследим путь программы до её появления перед пользователем:

Исходный код (Source Code) ← компилятор и инструментарий сборки (Toolchain) ← пакеты и зависимости (Dependencies) ← исполняемый артефакт (Binary Artifact) ← подпись и разрешение на выпуск (Release Signing) ← среда выполнения и аппаратура (Runtime and Hardware) ← данные и конфигурация (Data and Configuration) ← проверка приемлемого поведения (Acceptance Tests).

Не для каждого продукта это однонаправленная цепочка: зависимости ветвятся, некоторые системы используют готовые образы, а отдельные этапы идут параллельно. Но такой схемы достаточно, чтобы увидеть распространённую иллюзию. Организация может владеть исходным кодом, хранить образы серверов и контролировать данные — и всё же остановиться на этапе обновления, потому что подпись выпуска остаётся у производителя, его комплект средств разработки (Software Development Kit; SDK) больше недоступен или аппаратный интерфейс принимает только подписанную прошивку (Signed Firmware) стороннего поставщика.

Возможен и обратный случай: исполняемая версия долго продолжает работать, но никто не способен безопасно изменить и заново собрать её. «Сейчас работает» и «я могу создать новую версию» — разные утверждения.

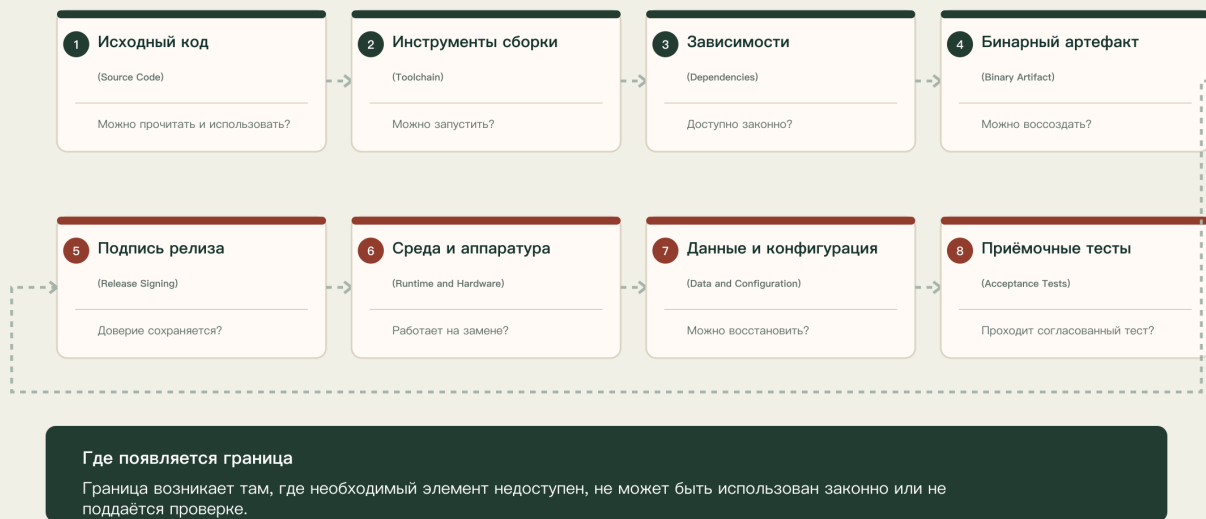
Полезно различать четыре действия, которые нельзя без разбора записывать в графу «владение»:

Действие	Настоящая проверка	Иллюзия, с которой его можно спутать
Сохранение (Preserve)	Сохранились ли необходимые активы документированными, читаемыми и законно удерживаемыми?	Наличие архива с неизвестным содержанием
Пересборка (Rebuild)	Можем ли мы независимо превратить код в исполняемую версию, получив совпадение либо документировав различия?	Само по себе владение кодом
Исправление (Repair)	Можем ли мы изменить ошибочное поведение и пройти известные тесты?	Повторный запуск старой версии
Перенос или замена (Port/Replace)	Можем ли мы сохранить ключевую функцию на иной платформе?	Перенос данных без восстановления функции

Организация может выполнить первые два действия и не справиться с третьим. Она может запустить замену с меньшим числом функций — не идентичную прежнему продукту, но удовлетворяющую согласованные базовые потребности. Это нельзя считать полным техническим совпадением; это **осознанная реконструкция функции**.

Граница воспроизводимости: цепочка технических активов

Путь от исходного материала к приемлемой функции



Общая исследовательская схема, не карта зависимостей саудовского объекта или системы.

Концептуальная схема границы воспроизводимости и этапов испытания технологии

Рисунок 1 — Предлагаемая граница воспроизводимости: какими сборочными активами мы владеем и что по-прежнему зависит от доступности, законности использования или целостности подписи у другой стороны? Это общая исследовательская схема, а не карта зависимостей саудовского объекта.

4. Можно иметь код и потерять ключ, превращающий его в программу

Более сорока лет назад Кен Томпсон (Ken Thompson) в знаменитой лекции (Reflections on Trusting Trust) сформулировал проблему, подрывающую простую уверенность в чтении исходного кода: сам компилятор (Compiler) может содержать поведение, которого нет в исходнике, предназначенном для компиляции. [S16](#)

Нет нужды пересказывать здесь весь сценарий атаки. Для нашей темы достаточно его интеллектуального следствия: **один лишь исходный код не объясняет, как родилась фактически исполняемая версия**. За ней стоит история инструментов сборки, версий библиотек, переменных среды, правил компоновки, а иногда и ошибок, которые сохранялись так долго, что стали частью ожидаемого поведения.

Поэтому «мы сохранили код» — не то же самое, что «мы независимо пересобрали приложение и доказали, что результат соответствует требуемой версии». Проект воспроизводимых сборок (Reproducible Builds) определяет воспроизводимость как возможность для разных сторон получить при одинаковых входных данных, среде и

инструкциях побитно идентичные результаты (Bit-for-Bit). Криптографический хеш (Cryptographic Hash) позволяет сравнивать версии, однако сам по себе не доказывает, что исходный код верен, программа безопасна, а право её запускать предоставлено. [S11](#)

Здесь пригодны три разных технических понятия, а не три названия для печати одного сертификата готовности:

1. **Ведомость программных компонентов (Software Bill of Materials; SBOM)**, в том числе в формате (SPDX), многое сообщает о составе программы, связях и лицензиях. Но сама по себе она не гарантирует доступность всех инструментов сборки, секретов выпуска или аппаратуры. [S13](#)
2. **Данные о происхождении сборки (Build Provenance)** в спецификации уровней защищённости цепочки поставки программных артефактов (Supply-chain Levels for Software Artifacts; SLSA) помогают зафиксировать, кто, как и при каких гарантиях собрал продукт. Однако документация не превращается автоматически в независимое право использовать все компоненты или в способность изменить любое устройство. Версия (SLSA 1.2) была утверждена 24 ноября 2025 года и оставалась утверждённой версией на дату отсечения доказательств; более ранний выпуск нельзя подставлять по умолчанию. [S12](#)
3. **Система обновления программного обеспечения (The Update Framework; TUF)** предлагает механизмы проверки подлинности обновлений и защищает пути доставки от некоторых рисков, связанных с ключами и устаревшими версиями. Она не разрешает обходить защиту производителя, а показывает, что безопасный выпуск обновления зависит от инфраструктуры доверия, продолжение которой тоже нужно проектировать. [S14](#)

Организации взрослеют, когда различают эти технологии. Забавно увидеть хранилище с тысячей печатей целостности, пока мастерская не способна изготовить предмет, на который поставлены печати. Но шутка заканчивается, когда этим предметом оказывается нужная человеку услуга или средство производства, на котором работает экономика.

5. Память программ: где прячется знание, которого нет на диске?

Если компьютер знает всё, почему ремонт (Voyager 1) потребовал многомесячной кропотливой работы людей? Потому что код, память и отчёты не устраняют потребность в **неявном эксплуатационном знании (Tacit Operational Knowledge)**: понимании того, что можно переместить, что нельзя сломать и что означает каждое изменение в системе, чьи части отдалялись друг от друга десятилетиями.

Одно знание хранится в файлах, другое — в головах, третье — в поведении устройств, которые действуют не совсем так, как ожидали проектировщики. Поэтому недостаточно скопировать репозиторий (Git Repository) с одного сервера на другой. Нужно сохранить описание среды сборки, копию лицензированных зависимостей, базовые сравнения поведения, учебные материалы, честно отмечающие уже непонятное, и возможность испытать исправление, не подвергая риску реальный сервис.

Одна из сторон решения видна в международном Архиве программного наследия (Software Heritage), который стремится сохранять общедоступный код, историю его разработки и возможность обращаться к нему в будущем. Это важный пример сохранения исходников во времени, но не обещание, что всякую сохранённую программу можно исполнить на любом будущем устройстве, и не лицензия на повторное распространение закрытых материалов. [S15](#)

Для старой информационной системы самым редким активом может оказаться не процессор или диск, а простой тестовый файл, показывающий, как система двадцать лет назад вычисляла важное число. Такой файл превращает спор двух команд из столкновения мнений в проверяемый результат. Когда не остаётся никого, кто помнит смысл числа, изменение программы становится лингвистической авантюрой: мы знаем, как пишется слово, но не знаем, что оно означало в своём договоре с реальностью.

6. Практическая проверка: воспроизвести пакет, а не вспомнить его название

Чтобы тезис не остался эссе о красоте независимости, в научном приложении мы провели два ограниченных эксперимента с явно заданными допущениями.

В первом мы собрали небольшую демонстрационную службу на языке (Python), включавшую фиктивную функцию чтения и два теста результатов. Затем без внешнего соединения создали два отдельных пакета в разных каталогах диска, зафиксировав порядок файлов, временную метку и метаданные пакета. **Байты и оба хеша (SHA-256) полностью совпали**, а два простых теста прошли успешно. Это узкий лабораторный успех: он не означает, что компилятор независим от своего создателя, что результат проверен третьей стороной или что мы воспроизвели саудовскую программу. [MODEL]

Во втором эксперименте использован абстрактный граф из **26 узлов**: исходные файлы, данные, инструменты сборки, права активации и подписи, аппаратура и работоспособные альтернативы. Затем проверены четыре пути восстановления

условной функции при допущении, что поставщик исчез. Само число узлов ничего не говорит о глубине реальной системы. [MODEL]

Предполагаемый путь восстановления	Первичные элементы, недоступные после исчезновения поставщика	Возможность завершить путь в модели
Сохранённый исполняемый образ (Golden Image)	3	Невозможно без разрешённых альтернатив
Исходный код у независимого хранителя (Source Escrow)	5	Невозможно без исходных инструментов и среды выполнения
Исходный код и локальные инструменты сборки	3	Невозможно, пока активация, подпись и аппаратура остаются у поставщика
Переносимое функциональное ядро (Portable Minimum)	0	Возможно при допущении о корректности тестов, прав и альтернатив

Последний путь не сохранил всех возможностей условной системы: мы предположили сохранение шести функций из восьми. Эти числа не описывают показатели Саудовской Аравии, время или вероятности. Это **логическая проверка условий возможности воспроизведения**, а не симуляция атак и не отчёт о компании. Сопутствующее исследование описывает структуру модели зависимостей, а пакет материалов содержит таблицу результатов для её восьми случаев. [MODEL]

Четыре рецепта восстановления при отсутствии поставщика

Результаты синтетической модели из представленной таблицы проверки

Путь восстановления	Недоступные первичные корни	Завершение пути	Предполагаемые функции из 8
Сохранённый образ	3	Заблокирован	8/8 Условно
Исходники в хранении	5	Заблокирован	8/8 Условно
Исходники и локальные инструменты	3	Заблокирован	8/8 Условно
Переносимое функциональное ядро	0	Возможен в модели	6/8

Результаты синтетической модели из 26 узлов; 0 означает отсутствие требований вне заданного рецепта в этой модели.

Сравнение недостающих звеньев в четырёх условных рецептах сборки

Рисунок 2 — Результаты расчёта по условному дереву: число недоступных первичных входов для каждого пути при сравнении полных функций, зависящих от внешнего поставщика, и более узкого альтернативного ядра. Ноль здесь не означает отсутствия реальных рисков; он лишь показывает, что в искусственной схеме не осталось требования за пределами заданного рецепта.

Может показаться, что сохранение исходного кода слабее сохранения образа: пять отсутствующих элементов против трёх. Это верно **только для двух рецептов эксперимента**, а не как общее правило. Исходный код даёт больше возможностей для будущего ремонта, но для них нужны инструменты и сборочные пакеты, которые не требуются замороженной версии при обычном запуске. Если не различать горизонт немедленного запуска и горизонт изменений, числа покажутся окончательным приговором ценности исходника.

Более того, система может иметь **99 файлов из 100**, но единственный отсутствующий файл помешает выпустить новую версию. В этом расчёте важна не доля имеющегося, а наличие полного и действительного пути от материалов к требуемой функции.

7. В эпоху искусственного интеллекта наследование сложнее, чем передача файла с кодом

Перейдём от велосипеда, которому нужна деталь, к системе искусственного интеллекта (Artificial Intelligence; AI), которую организация способна запустить сегодня, тогда как через несколько лет исчезает один из поставщиков её процессоров или программных инструментов.

Организация может сказать: мы сохранили веса модели (Model Weights). Хорошо, но сохранились ли токенизатор (Tokenizer), описание архитектуры, настройки числовой точности, реализации пользовательских операторов (Custom Operators), библиотеки ускорения (Acceleration Libraries), планировщик запросов вывода (Inference Scheduler), критерии безопасности и оценочные данные, способные выявить ухудшение при переносе?

Вес без правильного способа его интерпретации подобен музыкальному инструменту с полным комплектом струн, длина и натяжение которых неизвестны, как неизвестен и строй, для которого они созданы.

Затем возникает экономика переноса. При перемещении нагрузки с одной системы ускорения на другую могут измениться результаты вывода (Inference), скорость и энергопотребление. Недостаточно, чтобы программа просто открыла файл модели: она должна демонстрировать согласованное поведение при приемлемых затратах и задержке. Права на изменение и повторное распространение тоже нужно проверять, а не выводить из наличия файлов.

В таком случае **переносимость (Portability)** становится оплачиваемым свойством: она требует продуманных абстракций, приёмочных испытаний, а иногда компромиссов в производительности или наборе функций. В обычных условиях специализированная платформа может быть эффективнее, поэтому независимость нельзя приравнивать к отказу от всех передовых мировых технологий. Научный выбор — сравнить **выгоду специализации** с ценой старения программы вместе с её единственным поставщиком.

Представим саудовскую лабораторию, создающую продукт с искусственным интеллектом. Самым ценным стратегическим шагом может стать не только покупка дополнительной мощности, но и способность перенести определённую модель на второй путь исполнения, повторно запустить набор ключевых тестов и объяснить изменение качества, задержки и стоимости. Это не заявление о том, что перенос прост или всегда возможен. Это точная формулировка того, что следует испытать, прежде чем первый путь превратится в неизменную судьбу.

8. Саудовская Аравия: от репозитория к фабрике воспроизводимости

Стратегическое дополнение должно быть практическим. Управление цифрового правительства (Digital Government Authority; DGA) заявило программу правительственного облачного репозитория программного обеспечения (Government Cloud Software Repository / GPaaS), ориентированную на доступность, развитие и обмен свободным программным обеспечением и открытым исходным кодом государственных органов. Саудовские ведомства, включая Главное управление недвижимости (Real Estate General Authority; REGA), представляют это направление как способ повторного использования и сокращения зависимости от отдельных поставщиков. Это опубликованный институциональный факт, но он не доказывает, что любой государственный сервис можно пересобрать вне его нынешнего поставщика.

S17S18

На некоторых путях репозиторий — необходимый шаг. Но проверка им не заканчивается.

Если саудовская технологическая организация захочет превратить это направление в более глубокую отрасль, начинать стоит с того, что я называю **испытанием технологического наследования (Technical Succession Trial)**, а не с многостраничного исследования соответствия требованиям:

День первый: команда, не создававшая продукт, открывает изолированный от серверов поставщика пакет активов и выясняет, какие исходники, зависимости, условия использования, инструменты и описание среды ей доступны.

День второй: команда пытается собрать исполняемую версию; сравнивает её хеш, когда идентичность действительно требуется, и записывает причины расхождений, если в сборке есть недетерминированные факторы.

День третий: запускает тестовую версию на синтетических или обезличенных данных с разрешения владельца, изменяет небольшую функцию и повторно проходит приёмочные испытания, проверки безопасности и производительности.

День четвёртый: пытается запустить определённое ядро на технически и договорно разрешённой альтернативе, не отключая аппаратную защиту и не имитируя чужие ключи.

День пятый: фиксирует сохранившиеся функции, неудачи, стоимость исправления и право принятия решения при расхождении результатов. Дни здесь задают условную учебную последовательность, а не обещают проверить сложную систему за пять суток.

Если испытание не удалось, нужно определить причину: лицензия, отсутствующий инструмент, программная зависимость, дефицит навыков или тест, который никто не умеет интерпретировать? Каждая причина требует своего вложения. Это может быть новый контракт, разрешённое зеркало пакетов (Package Mirror), поведенческие тесты, команда по инженерии компиляторов (Compiler Engineering) или полная смена архитектуры. Слово «локализация» не объясняет ни одного из этих шагов, хотя иногда уместно для обозначения более широкой политической или экономической цели.

9. Проверка сильнейшего возражения: всему ли следует рождаться дважды?

Нет. Для многих продуктов специализированный поставщик окажется быстрее, эффективнее и безопаснее, чем попытка заново создать каждый компонент внутри организации. Локальная пересборка сложной платформы может увеличить расходы, замедлить устранение уязвимостей и породить местного монополиста, более хрупкого, чем глобальная конкуренция. В некоторых отраслях договорные права или требования безопасности изначально запрещают заменять чувствительное программное обеспечение несертифицированными решениями.

Открытая программа тоже может плохо поддерживаться, а проект — располагать всем кодом, но не иметь инженеров, которые его понимают. Нет уравнения, в котором местное всегда безопасно, открытое всегда независимо, а распределённое всегда лучше.

Какое вложение тогда разумно? **Избирательная воспроизводимость (Selective Reproducibility).**

Функции следует классифицировать по реальной отдаче от способности их исправить или перенести: насколько трудно сменить поставщика; как долго сервис сохраняет ценность; каков ущерб от прекращения поддержки; насколько редки навыки; сколько существует альтернативных платформ; какова стоимость испытаний; можно ли быстрее заменить продукт новым, чем чинить старый. В одних случаях разумно купить продукт как есть и подготовить план выхода. В других продукт нельзя приобретать без раннего доказательства того, что его функциональное ядро способно работать независимо.

Разница между этими решениями возникает не из симпатии к местному или зарубежному поставщику. Её определяют **технические испытания и экономические сравнения, которые меняются от проекта к проекту.**

10. Рынок, цену которого мы ещё не научились определять

В мире, где программы переживают некоторые компании, а протоколы связи стареют раньше устройств, может появиться спрос на новую отрасль — скорее **эволюционное сопровождение (Evolutionary Maintenance)**, чем торговлю резервными копиями.

Её продукт — не очередная панель с зелёными индикаторами. Это средства автоматической пересборки старых выпусков в изолированных средах; лаборатории по оживлению унаследованного кода; трансляторы заброшенных интерфейсов; наборы для запуска поведенческих тестов на разных поколениях аппаратуры; независимые сервисы проверки, связывающие исполняемую версию с её исходником и средой сборки; открытые компоненты, сохраняющие минимальные функции устройства или сервиса после прекращения поддержки поставщиком.

В рамках инвестиций в вычисления, цифровые сервисы и искусственный интеллект Саудовская Аравия может исследовать экспортный потенциал таких технологий для организаций, не желающих превращать свои программы в музейные экспонаты сразу по окончании контракта. Но отрасль не возникает из названия исследовательского центра. Нужны оплачиваемые заказчиками задачи, разрешённые тестовые данные, ясные права распространения и сравнение экономической целесообразности ремонта исходного решения с его заменой.

Поэтому прогноз до 2035 года становится вопросом о рынке, а не о государственных лозунгах: **кто будет продавать способность продлевать жизнь чужих продуктов за пределы их первого поколения?**

Победителем может оказаться небольшая компания, освоившая перенос старого протокола, команда, создавшая проверяемый компилятор, или лаборатория, перенёсшая модель искусственного интеллекта на вторую платформу без обвала качества. Объявить их неизбежными победителями сейчас невозможно, однако механизм создания ценности уже достаточно ясен, чтобы спроектировать измеряющий её рыночный эксперимент.

11. Заключение: наследство, которое нельзя сложить в ящик

Когда инженеры (Voyager 1) исправили старый код на расстоянии, не позволяющем сигналу поддерживать обычный разговор, они победили не только географию. У них было достаточно описаний системы, испытательных инструментов, опыта и прав на

изменение и отправку программы, чтобы построить путь к физически недоступному ремонту. S01

Когда владельцам (Fisker Ocean) после краха производителя понадобились инженерная ассоциация и инициативы по обслуживанию, проблема заключалась не в отсутствии технологий в автомобилях. Технологией они были насыщены, но часть способности к развитию оставалась привязана к экономической организации, которая уже не была прежней. S08S09

Из этой невидимой дистанции рождается тезис этой статьи: **технический продукт — расходуемый актив; способность воспроизводить, проверять и исправлять его после ухода создателя — самостоятельная экономическая компетенция.**

Создание такой компетенции может стоить денег, которых не оправдывает каждый продукт. Испытание может показать, что исходный поставщик лучше любой альтернативы. Но полезно выяснить это до того дня, когда ремонт станет необходим.

У машины две даты: дата изготовления и день, когда владельцу впервые понадобится исправить то, чего уже не может исправить производитель. Между ними государство способно овладеть редким техническим искусством: **покупать полезное сегодня, сохраняя путь к его пониманию и повторному созданию необходимой функции завтра.**

Примечания к выводам

1. Ни материалы (NASA), ни материалы (Fisker) не доказывают существования аналогичной уязвимости на саудовских объектах. Первый случай показывает успешное программное исправление в условиях жёстких ограничений; второй — социальный и практический опыт передачи части обслуживания пользователям.
2. Банкротство юридических лиц (VanMoof) в 2023 году не означает, что бренд исчез сегодня. Новый владелец возобновил работу и создал ремонтный архив. Сокращение сети (2G) в 2026 году **не остановило эксплуатацию велосипедов**: заявленное воздействие ограничено противоугонным отслеживанием и удалённой связью на некоторых моделях.
3. Эксперимент с повторной сборкой доказывает идентичность небольшого искусственного пакета в описанной среде, но не является независимой сертификацией производственного пакета. Проверка дерева не даёт вероятностей, часов восстановления или национального показателя технологической независимости.
4. Независимая сборка или перенос не дают права обходить лицензионные ограничения, подписи или требования безопасности транспортных средств. Любой

практический эксперимент на реальной системе требует полномочий, испытаний и технических мер безопасности.

5. Подробности об источниках, их ролях, датах событий и публикаций, а также границах выводов собраны в реестре источников (`source_register.csv`) и досье фактов (`news_casebook.md`) в составе пакета материалов. Предложение о «границе воспроизводимости» сформулировано автором.

Избранные источники

- Национальное управление по аэронавтике и исследованию космического пространства (NASA), *Voyager 1 Resumes Sending Engineering Updates to Earth*, 22 апреля 2024 года. [S01](#)
- Окружной суд Амстердама и управляющие по банкротству (VanMoof Bankruptcy Trustees), публичный отчёт о банкротстве, июнь 2024 года; событие 17 июля 2023 года. [S03](#)
- Компания (Cowboy), официальная страница приложения (Bikey), июль 2023 года. [S04](#)
- Компания (VanMoof), объявление о ремонтном архиве от 27 февраля 2024 года и обновления о сети (2G) за 2026 год. [S06S07](#)
- Заявление Ассоциации владельцев (Fisker Owners Association), поданное в дело о банкротстве в 2024 году, и страница разработанного (Undercurrent Inc.) приложения (OceanLink Pro), которое в (App Store) названо официальным приложением для членов (FOA). [S08S09](#)
- Проект (Reproducible Builds), техническое определение идентичной сборки. [S11](#)
- Спецификация (SLSA 1.2), утверждённая 24 ноября 2025 года и остававшаяся утверждённой версией на дату отсечения доказательств. [S12](#)
- Управление цифрового правительства (Digital Government Authority), программа репозитория государственного кода. [S17](#)