



НАУЧНОЕ ПРИЛОЖЕНИЕ

Редакционная дата: 18 октября 2026 года

Граница воспроизводимости: проверяемая модель независимости технологического восстановления от исходного поставщика

The Reproduction Boundary: A Testable Model of Supplier-Independent Technical Recoverability

Тип работы: концептуальное исследование (Conceptual Research Paper), включающее два реальных ограниченных вычислительных эксперимента: две детерминированные сборки небольшой программной версии (Deterministic Artifact Rebuild) и проверку доступности узлов искусственного графа зависимостей при утрате поставщика (Supplier-Loss Dependency Closure). Производственные выборки, национальные оценки и вероятностные прогнозы не использовались.

Степень завершенности: выполнены обзор перечисленных ниже свидетельств, проектирование концептуальной модели, реализация экспериментальных программ и проверка их результатов, а также схема будущего полевого исследования. Полевое сравнение, эксперименты на саудовских сервисах и эмпирические финансово-экономические исследования не проводились.

Дата отсечения доказательств: 18 октября 2026 года. Условный горизонт применения: 2026–2035.

Резюме на русском языке

Проблема: в литературе о технологической независимости большое внимание уделяется владению кодом, данными и материальной инфраструктурой. Однако этих показателей недостаточно, чтобы судить, способна ли организация воспроизвести работоспособный сервис или исправить его, если инструменты, лицензии, подписи или знания исходного поставщика недоступны. В статье предлагается понятие **границы воспроизводимости (Reproduction Boundary)** — последнего документированного перехода от исходных активов к приемлемой безопасной функции, который владелец сервиса может выполнить без необходимой зависимости от недоступного актива. Это свойство функции, пути и контекста, а не абсолютный ранг государства.

Доказательства: программный ремонт зонда (Voyager 1) в 2024 году, пути обслуживания автомобилей (Fisker Ocean) после банкротства производителя и передача знаний в (VanMoof) в 2023–2026 годах представляют разные случаи, позволяющие отделить техническую способность от географии, владения активами и прав. (Reproducible Builds), (SLSA 1.2), (SPDX) и система обновления программного обеспечения (The Update Framework; TUF) дают частичные механизмы сохранения информации, подтверждения происхождения и защиты выпусков, но ни один из них по отдельности не гарантирует независимость повторного запуска.

Выполненный метод: созданы небольшая экспериментальная программа на (Python) и два поведенческих теста. В двух разных временных каталогах собраны два пакета при фиксированных входных данных, порядке файлов и временных метках. Пакеты совпали побитно и дали одинаковый хеш (SHA-256). Кроме того, проверено искусственное дерево из **26 узлов**, включавшее локальные активы, потребности в поставщике и производные компоненты. Выполнено **восемь проверок путей восстановления**: четыре при доступном поставщике и четыре при его отсутствии. Во втором случае путям исполняемого образа, депонированного исходного кода и исходного кода с локальными инструментами сборки не хватало соответственно **3, 5 и 3** первичных входов. Условный путь переносимого ядра завершился без недостающих входов при допущении о сохранении **шести из восьми функций**, без заявления о полном совпадении с оригиналом.

Вывод: проверка даёт ограниченное вычислительное доказательство того, что большое число сохранённых файлов или наличие исходного кода сами по себе не создают полного пути к восстановлению определённой функции при отсутствии поставщика. Она не доказывает, что предложенные показатели предсказывают результаты для Саудовской Аравии, и не оценивает размер рынка или время восстановления. Поэтому предлагается независимый эксперимент на нескольких продуктах, сравнивающий

четыре испытания: сохранение материалов, пересборку, исправление поведения и разрешённый функциональный перенос. Прилагается схема проверки и опровержения, границы безопасности и структура затрат.

Ключевые слова: технологическая независимость (Technical Independence); граница воспроизводимости (Reproduction Boundary); функциональная восстановимость (Functional Recoverability); воспроизводимые сборки (Reproducible Builds); ведомость программных компонентов (SBOM); активы цепочки сборки (Build Chain Assets); зависимость от поставщика (Supplier Dependence); замыкание зависимостей (Dependency Closure); наследуемость программного обеспечения (Software Inheritability).

English abstract

Question. Does owning a service's data, source code, or compiled image establish the practical ability to rebuild, modify, validate, or replace its essential function when the original supplier is unavailable? We introduce a proposed *Reproduction Boundary*: the last independently executable and verifiable transition within a service's source-to-operation chain, conditional on specified functionality, lawful access, build tools, cryptographic trust, hardware interfaces, skills, and testing.

Completed work. An original small Python demonstration service was packaged twice at separate temporary locations using identical explicitly fixed archive metadata. Its two SHA-256 digests and raw bytes matched, and two basic functional checks passed. An entirely synthetic 26-node dependency graph and four alternative restoration targets were evaluated both with and without vendor primitives, producing eight deterministic route checks. Under the stipulated supplier-absent condition, the archived-image, source-escrow and local-toolchain routes had respectively 3, 5 and 3 unavailable primitive prerequisites and could not complete. The assumed portable-minimum route had none but retained only six of eight stipulated features. These results are properties of an invented graph and toy package; they are **not supplier-outage probabilities, measured Saudi systems, real security certifications, independent reproductions of production software, or restoration-time estimates.**

Contribution and next study. The framework separates byte-level reproducibility, behavioral acceptability, authorization to deploy, and economic attractiveness. It identifies the data and authorized technical experiments needed to learn whether an independently exercised reconstruction capability predicts shorter or cheaper vendor-exit transitions than inventories, escrow commitments, or system redundancy alone. Proposed falsification and economic decision criteria are explicit; no causal estimate or validated national index is claimed.

Keywords: sovereign recoverability; independent rebuild; software supply chain; hermetic build; functional equivalence; digital maintenance; supplier disappearance; reproducible software; Saudi Arabia.

1. Научный вопрос и единица анализа

Центральный вопрос: **при каких условиях организация способна законно, технически и с доказуемым качеством воспроизвести определённую рабочую функцию без участия исходного поставщика, от которого зависят некоторые инструменты или права на запуск?**

Единицей анализа служит не «государство» целиком, не «центр обработки данных» и не коммерческое название «приложения». Это **определённая функция или набор функций** конкретной версии продукта на известной платформе, в разрешённом сценарии утраты поставщика и при открыто заданном критерии приёмки. Простую службу отчётности можно пересобрать, тогда как компонент, зависящий от лицензированной модели или аппаратуры, — нет. Общая мера, стирающая это различие, не даёт ответа.

Понятие «цифрового суверенитета» может охватывать правовые и политические измерения, не сводимые к инженерному проектированию. Поэтому статья разделяет три вопроса:

1. **Существуют ли материалы?** Данные, код, исполняемые версии, документация, образы, пакеты и драйверы.
2. **Доступен и корректен ли путь?** Инструменты, право использования, законно принадлежащие ключи подписи, заменяемая аппаратура, поведенческие тесты, человеческая квалификация.
3. **Оправдано ли вложение в этот путь?** Стоимость сохранения и обновления способности, стоимость новой разработки или покупки замены, ожидаемый ущерб от отсутствия поставщика и срок жизни продукта.

Это разные испытания с разными знаменателями, а не уровни зрелости, присваиваемые по впечатлению.

1.1 Что способно изменить ответ?

Если санкционированное исследование покажет, что сохранённая копия кода практически не повышает способность независимых команд собрать пригодный выпуск без поставщика, изменятся и договоры разработки, и приоритеты организаций: вместо одного лишь приобретения исходников потребуется проверяемое сохранение

среды сборки и её тестов, а возможно, создание переносимого ядра. И наоборот, если сравнительные исследования покажут, что содержание независимых репозиторий увеличивает число отказов и расходы, не ускоряя восстановление определённого класса сервисов, распространять этот подход на все случаи не следует.

1.2 Предлагаемые гипотезы

- **H1 (полнота пути):** для продуктов с лицензируемыми или трудно заменяемыми зависимостями фактическое прохождение испытания пересборки лучше предсказывает способность переноса и исправления, чем само наличие копии исходного кода, после учёта сложности и размера продукта и характеристик команды.
- **H2 (границы функции):** вероятность получить приемлемую замену выше, если до исчезновения поставщика определены основные функции, независимые от второстепенных возможностей, хотя число доступных возможностей при этом может сократиться.
- **H3 (сохранность активов во времени):** сохранение исходного кода теряет эффективность, если его не сопровождают механизмы обновления пакетов, инфраструктуры компиляторов и аппаратных интерфейсов по мере их старения.
- **H4 (условная отдача):** не всякий сервис оправдывает стоимость независимого воспроизведения; ценность зависит от горизонта использования, глубины адаптации, наличия альтернативного рынка и цены ошибки.

Сильная контргипотеза: покупка современного продукта-замены и перестройка процессов могут оказаться эффективнее пересборки старой системы. Управляемые коммерческие зависимости поставщика с большой клиентской базой могут быть менее рискованными, чем слабо поддерживаемая локальная копия. Эксперимент должен допускать обе возможности, а не изолировать их как политическое возражение.

2. Обзор фактических свидетельств отдельно от выводов

2.1 Зонд (Voyager 1): огромная физическая дистанция при доступных интеллектуальных путях ремонта

22 апреля 2024 года Национальное управление США по авиации и исследованию космического пространства (National Aeronautics and Space Administration; NASA) сообщило о возобновлении передачи понятных инженерных данных. Инженеры диагностировали повреждение части памяти подсистемы полётных данных (Flight Data Subsystem; FDS), распределили фрагмент кода и его инструкции по другим областям.

Расстояние превышало 24 миллиарда километров, а односторонняя задержка сигнала составляла около 22,5 часа. Первый наблюдаемый практический успех был получен 20 апреля. Позднее, в мае, агентство сообщило о возобновлении научных данных с двух приборов; эти этапы нельзя объединять в «полное восстановление» уже в апреле. S01S02

Подтверждённое наблюдение: частичное программное исправление, выполненное дистанционно, удалось, хотя физически отремонтировать чип было невозможно.

Возможный механизм: знание интерпретации памяти, кода и внутренних связей, а также санкционированная возможность отправлять инструкции и получать доказательства проверки. **Чего случай не доказывает:** успешного коммерческого плана выхода, независимости сборки другого зонда от его поставщиков или обычной стоимости работ, переносимой на государственные сервисы.

2.2 Автомобиль (Fisker Ocean): владелец становится участником отрасли обслуживания

В 2024 году производитель автомобилей (Fisker) вступил в процедуру банкротства. В заявлении от 26 июля 2024 года, поданном в рамках дела, Ассоциация владельцев (Fisker Owners Association; FOA) объясняла, что для непрерывности требуются доступ к коду, сервисным инструментам, облачным службам, запасным частям и связи с автомобилями. Ассоциация выступала стороной, добивавшейся соглашения, а не нейтральным источником оценки результатов. Разработчиком и поставщиком приложения (OceanLink Pro) указана компания (Undercurrent Inc.); описание в (App Store) называет его официальным приложением для членов (FOA) и перечисляет функции, зависящие от оборудования и версии программ автомобиля. Ни заявление, ни описание разработчика не доказывают остановки всех автомобилей или независимости всех их функций. S08S09S10

Подтверждённое наблюдение: вне исходного завода возникла инициатива по техническому и программному обслуживанию; появился инструмент разработчика, заявляющего, что он представляет владельцев. **Возможное объяснение:** ограниченное знание, распределённое между пользователями и инженерами, превращено в организованную способность к ремонту. **Граница переноса вывода:** документы ассоциации принадлежат заинтересованной стороне; описания приложения не заменяют независимой проверки каждого автомобиля, отдельным функциям требуется специальное оборудование, а восстановление не означает решения всех неисправностей и прав на обновление. Безопасность альтернатив следует экспериментально проверять силами уполномоченных специалистов; мы не предоставляем способы доступа к автомобильным системам или обхода их защиты.

2.3 (VanMoof): три разных потрясения и сложный срок жизни технологии

Оригинальный опубликованный отчёт управляющих по банкротству (VanMoof) подтверждает, что Окружной суд Амстердама (District Court of Amsterdam) объявил банкротами нидерландские юридические лица компании 17 июля 2023 года. В те дни конкурент (Cowboy) выпустил приложение (Bikey): исходное описание выделяло локальное сохранение ключа велосипедов (S3/X3) на случай недоступности сервисов исходной компании, а история версий позднее зафиксировала поддержку (A5/S5). Страница не подтверждает полного долгосрочного охвата всех моделей. [S03S04](#)

27 февраля 2024 года новые владельцы сообщили о расширении доступа к инструкциям по ремонту и техническим чертежам поколений велосипедов в (VanMoof Archives), пояснив, что прежде часть материалов была доступна только авторизованным мастерским. Эти документы относятся к механическому и процедурному ремонту; из их публикации не следует доступность всех электронных схем, ключей выпуска или прав собственности на программы. [S05S06](#)

В 2026 году производитель объявил об изменении связи старых моделей, зависевших от сети второго поколения (2G), и установил 1 апреля как крайний срок подписки, сохранявшей противоугонное отслеживание для определённых категорий. Компания подчеркнула, что запирающие, езда и прочие автономные функции **не прекращаются**. Это событие отдельно от банкротства 2023 года: здесь стареет внешняя коммуникационная инфраструктура, хотя бренд и коммерческая деятельность возобновились. [S07](#)

Эта последовательность позволяет концептуально проверить три разные причины: исчезновение коммерческого субъекта, закрытость ремонтных знаний и старение внешнего протокола. Их нельзя смешивать в слове «поломка» или переносить последствия одной на другую. Инициативу по расширению доступа к инструкциям можно рассматривать как **частичный контрпример** к предположению, что всякий поставщик после реструктуризации будет препятствовать продолжению эксплуатации.

2.4 Саудовское свидетельство: активы доступны для обмена, но возможность сборки не предполагается

Управление цифрового правительства (Digital Government Authority; DGA) описывает программу правительственного облачного репозитория программного обеспечения (Government Cloud Software Repository; GPaaS) как средство хранения и обмена свободным программным обеспечением и открытым исходным кодом государственных органов, а также содействия их разработке. Главное управление

недвижимости (Real Estate General Authority; REGA) публикует требования и роль открытого программного обеспечения и государственной платформы депонирования. [S17S18](#)

Наблюдение важно, поскольку задаёт **реальную исходную точку** для испытаний пересборки. Оно не утверждает, что сама платформа измеряет независимость повторного запуска или что в ней находится всё программное обеспечение Королевства. До оценки эффективности переноса нужны открытые либо санкционированные испытания конкретных образцов.

3. Научное основание: различать четыре типа доказательств

3.1 Долговременное сохранение (Digital Preservation)

Архив программного наследия (Software Heritage) сосредоточен на сборе общедоступного открытого кода, истории его разработки и доступности через прослеживаемую ссылку. Эта цель отличается от предоставления полного образа производственной среды, гарантии сохранения компилятора, которым пользовалась старая версия, или лицензии на каждую коммерческую зависимость. [S15](#)

Архивирование способно спасти незаменимый актив, однако один снимок репозитория может не содержать внешних пакетов, настроек рабочей среды, состояния базы данных и секретных значений, которые вообще не следует депонировать. Распространение некоторых компонентов может быть незаконным. Активы следует разделять на материалы, которые можно сохранять и публиковать; материалы, которые можно сохранять с ограничениями; и активы, которые нельзя хранить или распространять вне законных средств контроля.

3.2 Идентичность результата сборки (Reproducible Builds)

Проект (Reproducible Builds) определяет воспроизводимую сборку как повторное создание побитно идентичных результатов при наличии того же кода, среды и инструкций. На результат способны влиять версии зависимостей, настройки локали, временная метка и путь, из которого выполняется сборка. [S11](#)

Что доказывает совпадение: при заданных допущениях и настройках из определённых входов можно повторно получить те же байты. **Чего оно не доказывает:** отсутствия дефектов, безвредности компилятора, возможности импортировать данные, законности

ключей выпуска или разрешения компетентного органа на использование. Это доказательства разных классов.

Воспроизводимость не всегда измеряется побитным совпадением. Если перенос на другую платформу или смена архитектуры неизбежно меняют байты, более уместен критерий **приемлемой поведенческой эквивалентности (Behavioral Equivalence)**, для которого необходимо определить входы, выходы, допустимые ошибки и тесты. Поведенческую эквивалентность нельзя называть бинарной идентичностью.

3.3 Происхождение и состав программного обеспечения (Provenance and SBOM)

Спецификация обмена данными о программных пакетах (Software Package Data Exchange; SPDX) определяет способы описания программных активов, связей и лицензий. Уровни защищённости цепочки поставки программных артефактов (Supply-chain Levels for Software Artifacts; SLSA 1.2) предлагают требования и уровни прослеживания происхождения исходного кода и сборки с различными гарантиями. Ни один из подходов не означает, что вторая сторона автоматически получает право или способность владеть устройством подписи, активации или процессором. [S12S13](#)

Полный реестр компонентов — важное свидетельство готовности, но он может остановиться перед «теневой зависимостью (Shadow Dependency)», не записанной в перечне пакетов: лицензией на удалённый сервис, встроенным аппаратным ключом, набором тестов у поставщика или недокументированным эксплуатационным знанием. Это примеры возможных механизмов, а не перечень уязвимостей конкретных объектов.

3.4 Безопасное обновление и построение доверия (Secure Software Update)

Система обновления программного обеспечения (The Update Framework; TUF) предлагает способ снизить риски ключей выпуска и старых либо поддельных версий обновлений с помощью подписанных метаданных, разных полномочий ключей и сроков действия. В конечном счёте запуск новой программы на устройстве остаётся обусловлен правилами доверия этой аппаратуры. [S14](#)

Здесь существенно различие между **наличием действительной подписи** и **правом изменить сторону, уполномоченную подписывать**. Корректная защита обновлений может блокировать несанкционированную программу, даже если владелец устройства имеет её исходный код. Обход такой защиты нельзя считать «решением для восстановления»: необходимо разрешение правообладателя, заранее спроектированная возможность законного обновления либо пригодная альтернативная аппаратура.

3.5 Доверие к компилятору (Trusting Trust)

Кен Томпсон (Ken Thompson) в лекции (Reflections on Trusting Trust) показал, почему одного анализа исходного кода может оказаться недостаточно для понимания исполняемой версии, если её производственные инструменты не заслуживают доверия. Позднее Дэвид Уилер (David A. Wheeler) разработал метод разнородной двойной компиляции (Diverse Double-Compiling; DDC), позволяющий при определённых допущениях проверять соответствие некоторых компиляторов предполагаемым исходникам. [S16S19](#)

Это не призыв каждой организации переписать каждый компилятор. Это объяснение того, что «возможность собрать программу» распадается на отдельные гарантии: сохранность инструментов; доступ к их исходникам; независимость проверяющих сторон; пригодность компилятора для платформы; наличие соразмерного рискам доказательства проверки.

4. Концептуальная модель и обозначения

Определим конкретный сервис s и приемлемую функцию f в сценарии утраты поставщика z . Путь их создания включает первичные активы и производные этапы в графе зависимостей артефактов (Artifact Dependency Graph; ADG); некоторые активы могут быть обязательными, а некоторые — альтернативными. Связь зависимости нельзя считать установленной без проверки реального продукта.

Основные переменные:

- $A(v, z)$: законная и независимая доступность первичного актива, права или способности v в сценарии z ; в упрощённой детерминированной модели равна нулю или единице.
- $Anc(r)$: множество первичных требований, от которых зависит путь воспроизведения r , после прослеживания зависимостей до корней.
- $V(r)$: независимое решение о том, достигли ли результаты требуемого уровня безопасности и поведения; одного успешного объединения файлов недостаточно.
- $L(r)$: независимое решение о законности выполнения пути, выпуска результата и хранения его активов по действующему соглашению.
- $F(r)$: набор функций, практически доказуемых на этом пути, а не обещанных в плане.

Предлагается следующий показатель начального условия сборки для каждого пути:

$$P_r = \prod_{v \in Anc(r)} A(v, z)$$

Значение $P_r = 1$ означает, что в данном сценарии доступны все первичные требования, а не то, что программа автоматически работоспособна, безопасна или законна. Практически полное воспроизведение требует дополнительно подтверждения $V(r)$ и $L(r)$ и наличия функции f в $F(r)$. При наличии независимых альтернативных путей один пригодный путь может сохраниться, даже когда другие отказали.

В искусственном примере измеряется число отсутствующих корневых зависимостей:

$$B_r = \sum_{v \in Anc(r)} (1 - A(v, z))$$

Это **число препятствий**, не имеющих экономических весов. Один незаменимый ключ может быть гораздо сложнее десяти файлов, которые за несколько минут восстанавливаются из другого репозитория. Значение B_r нельзя само по себе сравнивать между сервисами без определения активов и их границ.

При наличии реальных временных данных время восстановления функции можно изучать как период от **объявления об отсутствии поставщика** до первого успешного прохождения проверки безопасности и использования в альтернативной среде; случаи, не достигшие успеха до конца эксперимента, следует считать цензурированными справа (Right-Censored). Национальный процент, рассчитанный по числу сохранённых файлов, не предлагается: он смешивал бы неоднородные величины и скрывал единственное «блокирующее препятствие».

5. Первый выполненный вычислительный эксперимент: повторная сборка из известных входов

5.1 Дизайн

Мы создали файл (analysis/demo_service.py) с небольшой функцией на Python. Она проверяет наличие искусственного идентификатора в локальном наборе и возвращает идентификатор запроса, логический флаг и версию схемы. Функция не принимает реальных решений о правах и не обращается к внешним данным.

Сценарий (analysis/reproduce.py) использует стандартную библиотеку (zipfile), чтобы объединить три фиксированных элемента: код службы, небольшую схему JSON и файл с эталонным временем сборки. Имена файлов сортируются, временная метка и метаданные устанавливаются одинаковыми, после чего пакет собирается в двух разных временных каталогах. Байты двух файлов и их хеши (SHA-256) сравниваются напрямую. Зависимости из интернета не загружаются, коммерческая цепочка подписи не выполняется.

5.2 Фактические результаты

- Число пространственно независимых попыток внутри одного процесса: 2.
- Побитное совпадение: **True**.
- Хеш каждой сборки: 36c763c61b7114cffe473a230d5d583ca9acceff14ff60bdebfa3d78747fdc52.
- Небольшие положительный и отрицательный тесты службы: **оба пройдены**.
- Временный архив результата не является производственным пакетом и не публиковался как устанавливаемое обновление; его размер, хеши двух сборок и результаты двух тестов эксперимент записал в файл с именем `data/toy_rebuild_verification.json`.

Допустимый научный вывод узок: когда входные данные и сведения об упаковке в этом примере были зафиксированы, две сборки дали один результат. **Мы не называем это независимым воспроизведением, проверенным третьей стороной**: обе попытки использовали один код и стандартный интерпретатор языка в одной среде.

Эксперимент не доказывает воспроизводимость программ с разными компиляторами, лицензиями, библиотеками и аппаратурой.

5.3 Контроль отказа, не выполненный на реальной системе

Ограничения эксперимента указывают на следующие этапы: выполнить сборку в двух действительно независимых средах; проверить целостность активов до их получения; сравнить поведение результата в тестовой среде; испытать недоступность репозитория пакетов; проверить лицензированное обновление. Всё это **предлагаемые планы**, а не заявленные результаты.

6. Второй эксперимент: полнота дерева требований при отсутствии поставщика

6.1 Искусственные входные данные

Мы создали файл `data/reproduction_graph.json`, описывающий **26 узлов**. В него входят корневые активы — исходный код службы, экспорт и схема данных, локальный инструмент сборки, зеркало пакетов, — требования, зависящие от поставщика, такие как проприетарный компилятор, комплект разработки, активация, подпись и прошивка, а также узлы, производимые из предыдущих.

Это **вымышленная учебная карта**. Она не представляет конкретную компанию или класс саудовского оборудования и не раскрывает реальные точки контроля или отказа. Доступность первичного компонента отделена от успеха всей рабочей функции; каждый путь в эксперименте заранее предполагает выполнение остальных требований к тестам и правам, поэтому программа их не оценивает и не пытается обойти.

Данные определяют четыре пути:

1. `image_archive`: развёртывание сохранённого исполняемого образа с данными и конфигурацией, условиями активации и подписи и аппаратурой поставщика.
2. `source_escrow`: сборка доступного исходного кода с исходным компилятором и комплектом разработки, а также исходными требованиями среды выполнения.
3. `source_with_local_tools`: сборка исходного кода локальными инструментами и пакетами; часть требований к активации, подписи и исходной аппаратуре сохраняется.
4. `portable_minimum`: изменение конструкции части функции и её сборка из условных альтернативных компонентов с действительной лицензией, законной локальной подписью и более узким набором функций, чем в оригинале.

Каждый путь проверен в двух условиях: **поставщик доступен** и **поставщик недоступен**. Вероятности сценариев не задавались; это две детерминированные совокупности условий.

6.2 Фактические результаты

Экспериментальный путь	Поставщик доступен: отсутствующие корни	Поставщик недоступен: отсутствующие корни	Возможность завершения во втором случае	Условные функции, из 8
Сохранённый образ	0	3	Нет	8 условно
Депонированный исходный код	0	5	Нет	8 условно
Исходный код и локальные инструменты	0	3	Нет	8 условно
Переносимое ядро	0	0	Да, при условиях примера	6

Программа выполнила **8 оценок**, а не 8 реальных экспериментов на разных продуктах. Эксперимент записал названия отсутствующих требований и результаты каждого пути в файл с именем `data/closure_results.csv`. Внутренний тест подтвердил, что число недоступных корневых узлов соответствует заявленным результатам.

Числа показывают, что депонирование исходного кода недостаточно в рецепте, по-прежнему требующем активов поставщика, а переход к переносимому ядру способен уменьшить зависимость ценой отдельных функций. Результат структурный и **чувствителен к способу построения зависимостей**: при изменении узлов, прав их использования или появлении альтернатив изменится и вывод. Стоимость исправления, время переноса и безопасность восстановленной версии не измерялись.

6.3 Что изменится при другом дизайне эксперимента?

Если предположить, что перед уходом поставщик безусловно предоставляет право запуска и действующий независимый ключ обновления, часть недостающих узлов путей образа и исходного кода исчезнет. Если для переносимой альтернативы нужен код без лицензии на изменение, она не сработает, даже если дерево файлов выглядит полным. Если критерий функции требует всех восьми возможностей, путь переносимого ядра нельзя признать успехом в том же смысле, хотя его можно назвать «частичной заменой».

Точное проектирование критерия приёмки — не исследовательское украшение. Оно определяет, что именно означает слово «восстановили».

7. Экономическое измерение: почему не следует пересобирать всё?

Модель не предлагает любой ценой создавать местную отрасль для производства всего. Решение требует сравнить вариант пересборки (Rebuild Option) с вариантом покупки замены (Replacement Option), учитывая ценность сохранения каждого варианта доступным. Компания или государственный орган может сегодня доплатить за права и инструменты, которые через пять лет упростят смену поставщика, но при коротком сроке жизни программы и наличии готовых конкурирующих альтернатив затраты способны превысить пользу.

Предлагается разделять шесть групп затрат, не выдумывая национальных показателей:

1. **Стоимость сохранения собираемости**: хранение исходников и лицензированных программных материалов, обновление зеркал пакетов, документирование версий, регулярные испытания.

2. **Стоимость утраченной специализации:** снижение производительности при отказе от аппаратуры или программ, оптимизированных для определённой задачи, в пользу переносимой альтернативы.
3. **Стоимость знания:** время инженера, способного понять старую систему и изменить её поведение без новых отказов.
4. **Стоимость прав и доверия:** лицензии, признанные способы подписи, обслуживание обновлений и надлежащее раскрытие.
5. **Стоимость простоя или замены сервиса:** потеря документированной функции, задержка производства и расходы на переход.
6. **Стоимость полной замены:** покупка новой системы, преобразование данных, обучение и вывод старой системы из эксплуатации по приемлемой процедуре.

Расходы организации на сохранение способности к изменениям следует сравнивать с ценой полного отказа от технологии, а не только со стоимостью старого серверного образа. Невозвратные издержки (Sunk Costs) необходимо отделять от будущих затрат, которых ещё можно избежать: окупаемость прошлых вложений не оправдывает продолжение невыгодного продукта.

7.1 Сценарий, ослабляющий наш тезис

Если продукт предлагают несколько компаний в стандартных форматах данных; данные можно быстро перенести и проверить; функция не требует глубокой адаптации; обновления постоянно и безопасно поддерживаются, — содержание независимой фабрики для его пересборки может оказаться пустой тратой средств. Практический суверенитет тогда состоит в **простоте замены**, а не в способности организации побитно воссоздать старую программу.

8. Предлагаемый протокол саудовского исследования

8.1 Дизайн и требуемая выборка

Для реальной проверки организациям следует разрешить отбор нечувствительных сервисов или санкционированных тестовых копий в саудовских секторах, связанных с четырьмя направлениями: государственными вычислениями, экономическими технологическими сервисами, компонентами искусственного интеллекта и отдельными коммерческими продуктами цифровой инфраструктуры. Функции следует классифицировать по сложности, зависимости от аппаратуры, характеру договоров и ожидаемому сроку жизни, а не только по имени поставщика.

Команда, независимая от разработчиков, проверяет, какую часть разрешённого пакета она способна создать без учётных записей поставщика. Исследователям нельзя давать доступ к реальным пользовательским данным и эксплуатационным секретам; при необходимости следует применять изолированные лаборатории и синтетические либо обезличенные данные. Эксперименты на работающих критических системах не требуются.

8.2 Четыре регистрируемых практических этапа

Этап первый: сохранение исходника и происхождения. Команда учитывает все законно полученные исходники, схемы зависимостей, лицензии, образы, пакеты и тестовые данные. Отсутствующие материалы и известные, но недоступные активы записываются точно.

Этап второй: воспроизведение сборки. Команда пытается создать исполняемую версию или объясняет причину неудачи, сохраняя журнал входов, команд, версий инструментов и хешей. При разных результатах ожидаемые различия отделяются от нарушения воспроизводимости.

Этап третий: исправление заранее выбранной безопасной проблемы. В тестовой среде команда исправляет искусственный дефект или узко меняет функцию, используя зафиксированный до начала набор тестов. Прохождение тестов нельзя называть исчерпывающим доказательством безопасности.

Этап четвёртый: перенос ключевой функции. Команда пытается запустить выбранные основные функции на разрешённой альтернативной платформе. При наличии надёжных измерительных средств фиксируются их число, результаты приёма и различия в задержке, стоимости и энергии. Результаты рассчитываются на уровне сервиса, а не в виде единого национального показателя, создающего иллюзию однородности функций.

8.3 Что измеряется, а что нет?

Предлагаемые первичные результаты: успех воспроизведения и причина неудачи; время от начала эксперимента до первой пригодной сборки; время до первого приемлемого исправления; число прошедших тесты функций; необходимые правовые условия; фактические трудозатраты; стоимость пути по компонентам. Невозможность завершить эксперимент за период наблюдения записывается отдельным состоянием, а не вымышленным расчётным бесконечным временем восстановления.

Сравнения: наличие только исходного кода; наличие исполняемого образа; исходный код со сборочными пакетами и архивными драйверами; готовая стандартная альтернатива. Их характеристики могут пересекаться; правила отбора следует определить так, чтобы лучшие команды не получали постоянно самые простые продукты.

Анализ причинных оговорок: успешная пересборка не доказывает, что успех вызван архивированием. Команда могла быть опытнее, компонент — проще, документация — свежее, а поставщик — более склонным к сотрудничеству заранее. Потребуется предварительная классификация, повторные эксперименты и ясное сравнение реализации. Если данные не допускают причинного дизайна, статья должна описывать только связи.

8.4 Условия остановки или отказа от пилотного проекта

Если испытание требует отключить производственную систему, обойти ключ подписи производителя, получить активы без лицензии или собрать чувствительную карту эксплуатации, соответствующий шаг следует исключить. Его можно заменить моделированием или ограниченной санкционированной процедурой. Независимость воспроизведения не оправдывает нарушение гарантий безопасности или прав, которые она должна защищать.

9. Прогноз до 2035 года: три пути национальной способности

Это не расчётные прогнозы, а **условные сценарии (Conditional Scenarios)**, связанные с наблюдаемыми сигналами.

Путь первый: изобилие активов, которые нельзя воссоздать. Репозитории, программные образы и закупочная документация разрастаются, но фактическая сборка версии вне поставщика не проверяется. Признаки риска — договоры, сохраняющие исходники без пригодной среды и испытательных инструментов, и программы, для которых независимая команда не умеет выпустить новую версию.

Путь второй: повсеместная локальная сборка без избирательной эффективности. Организации пытаются воспроизвести всё приобретённое, включая то, что дешевле купить в условиях сильной международной конкуренции. Возникают расходы на сопровождение, задержки и распыление опыта по проектам, чья значимость их не оправдывает. Решающим сигналом становится рост стоимости независимости без улучшения ремонтпригодности или простоты перехода.

Путь третий: экономика наследуемого сопровождения. Местные компании и лаборатории специализируются на воспроизведении выбранных классов продуктов, средствах переноса кода, побитной проверке, восстановлении интерфейсов, лабораториях поведенческих тестов и документировании прав на изменение. Успех измеряется доказанными независимыми пересборками, выручкой от сопровождения и проверяемым экспортом, а не количеством объявленных планов.

9.1 Шесть сигналов для обсерватории

1. Доля функций, действительно прошедших испытания, а не число проектов, заявивших планы непрерывности.
2. Доля успешных и неуспешных пересборок независимыми командами по типам продуктов.
3. Фактическая продолжительность отсутствия поддержки поставщика и её влияние на способность обновить тестовую версию.
4. Доля продуктов с актуальным зеркалом пакетов и законно воспроизводимой средой сборки.
5. Разница между стоимостью фактического переноса и покупкой новой альтернативы с учётом сокращённого набора функций.
6. Саудовские продукты, клиенты которых платят за переносимость, проверку и исправление, а не только за опубликованные лабораторные тесты.

10. Границы применимости и обобщения

У работы есть шесть крупных ограничений:

1. **Отсутствие саудовских производственных данных:** у нас нет проверенных выборок закупок, сборок или переносов саудовских организаций, поэтому нельзя утверждать, что показатель доказал их уязвимость или устойчивость.
2. **Упрощение дерева:** реальные зависимости могут содержать необязательные условия, циклы, совместимость версий, общие отказы и сложные лицензионные системы; реализованная модель — упрощённое дерево первичных условий с независимыми производными.
3. **Демонстрация безопасности:** наличие требований сборки не означает, что программа безопасна или даёт требуемые результаты. Демонстрационные тесты крайне малы.
4. **Различие прав:** закон или договор могут запрещать использование сохранённого исходника, ключа или прошивки; правовая оценка конкретного продукта не

проводилась.

5. **Проблема неявного знания:** дерево не измеряет опыт инженера, время обучения или широту пространства ошибок.
6. **Отличие побитной сборки от миграции:** перенос сервиса на другую аппаратуру создаёт другие байты; стратегически он может быть успешнее, но одна проверка совпадения (SHA-256) этого не показывает.

Поэтому модель не называется ни «**индексом саудовского суверенитета**», ни «сертификатом восстановимости». Это понятия, определения, два небольших эксперимента и предложение реалистичной исследовательской программы.

11. Проверки, способные опровергнуть тезис

Научный вклад требует определить, какие результаты сделают тезис менее убедительным:

- Если независимые поставщики услуг смогут регулярно покупать новые альтернативы и безопасно запускать их быстрее и дешевле, чем пересобирать оригинал, общая ценность сохранения сборочных способностей для исследуемых классов снизится.
- Если после учёта опыта команд, простоты программ и устойчивости поставщика испытания сохранённой цепочки сборки не дадут дополнительной прогностической силы, утверждение о «границе воспроизводимости» как самостоятельной значимой мере придётся изменить.
- Если «основную функцию» нельзя определить без утраты важных прав или свойств, модель переносимого ядра не подходит сервису, даже если технически реализуема.
- Если технические данные покажут, что зависимость, названную исключительной, можно немедленно заменить проверенной и лицензированной альтернативой, её нельзя сохранять на графе как «блокирующее препятствие».
- Если большинство программ имеют короткий срок жизни и эффективно заменяются, сохранение общих инженерных компетенций может оказаться лучше хранения рецептов сборки быстро устаревающих продуктов.

В этом различие между новаторским предложением и закрытым утверждением: спросить, какой результат заставит нас изменить стратегию, прежде чем потратить на неё миллиарды.

12. Протокол воспроизведения и сопутствующие исследовательские материалы

Структура файлов исходного эксперимента в описании рукописи:

Файл	Содержание	Описание границ
analysis/demo_service.py	Искусственная служба с полностью открытой логикой	Не содержит реальных данных, клиентов или прав
analysis/reproduce.py	Две сборки пакета и проверка дерева зависимостей	Использует только стандартный Python
data/reproduction_graph.json	26 узлов, четыре пути и два проверяемых условия	Разрешённые вымышленные входы, не представляющие саудовскую действительность
data/closure_results.csv	Восемь подробных результатов и причины отсутствия зависимостей	Расчётные случаи без вероятностных весов
data/toy_rebuild_verification.json	Размер пакета, два хеша и результаты двух тестов	Локальная проверка, а не независимая межорганизационная верификация
analysis/README.md	Порядок выполнения, допущения и способы проверки	Не публикует ключи или эксплуатационные данные

Материалы, доступные с этой редакцией: Материалы исследования включают closure_results.csv, экспортированный из предоставленной рабочей книги доказательств, реестр источников source_register.csv и досье фактов news_casebook.md. Исследовательская версия в формате (PDF) открывается отдельной кнопкой чтения.

Рукопись описывает процедуру исходного эксперимента как запуск команды `python3 analysis/reproduce.py` из рабочего каталога с последующим сравнением результатов CSV и JSON. Описанная процедура не предполагает программных зависимостей помимо стандартной библиотеки (Python). При этом другая версия интерпретатора (ZIP) или (Python) в третьей среде может выявить различие файла пакета; тогда прежде чем объявлять саму идею несостоятельной, необходимо полностью описать среду.

Архивы, созданные во временных каталогах, не публикуются как готовые программы: сохраняется только результат проверки, а пакет создаётся заново при запуске сценария.

13. Итог исследования

Вывод статьи не состоит в том, что «поставщикам нельзя доверять»: именно поставщик может лучше всех выполнять повторный запуск и сохранять инновации. Не утверждается и что «открытый код решает всё»: его инструменты могут устареть, опыт — исчезнуть, а сложные права — помешать исполнению функций.

Более точный вывод: **цифровое владение отделимо от инженерной способности оживить функцию**. Организация может хранить весь материал, но не иметь законного и проверенного пути от него к функции. Небольшая сторона может сохранить эффективный путь ремонта при скромных материальных активах.

Два небольших эксперимента показали возможность дважды побитно собрать один пакет в известной среде и продемонстрировали, как утрата немногих корней в искусственной модели зависимостей разрывает пути, хотя остальные материалы сохранены. Даст ли перенос этого принципа на саудовские сервисы реальную отдачу по стоимости, времени, независимости и обслуживанию — **гипотеза, требующая независимого эксперимента**.

Если такой эксперимент будет проведён, вместо лозунга о суверенитете появится более глубокое знание, доступное проверке: что система способна создать заново, кто способен это сделать, на какой аппаратуре и при каких затратах, правах и мерах безопасности, понятных тому, кто заплатит цену неудачи.

Первичные источники и идентификаторы прослеживаемости

Подробности об источниках, даты публикаций и событий, границы выводов и ветви цитирования собраны в реестре источников (`source_register.csv`) и досье фактов (`news_casebook.md`) в составе пакета материалов. Соответствующие идентификаторы приведены ниже:

1. S01 Национальное управление по аэронавтике и исследованию космического пространства (NASA): *Voyager 1 Resumes Sending Engineering Updates to Earth*, 22 апреля 2024 года.
2. S02 Национальное управление по аэронавтике и исследованию космического пространства (NASA): *Voyager 1 Resumes Sending Science Data from Two Instruments*, 22 мая 2024 года.
3. S03 (VanMoof Bankruptcy Trustees): *Fourth Public Bankruptcy Report*, 6 июня 2024 года; зарегистрированное событие 17 июля 2023 года.

4. [S04](#) (Cowboy SA): официальная страница приложения *Bikey*; исходное описание выделяло (S3/X3), а история версий позднее зафиксировала поддержку (A5/S5), июль–август 2023 года.
5. [S05](#) (VanMoof): заявление о возобновлении работы под новым владельцем и программе восстановления, июль 2024 года.
6. [S06](#) (VanMoof): *VanMoof Archives for Legacy Models*, 27 февраля 2024 года.
7. [S07](#) (VanMoof): *What's Happening with 2G Network Shutdown*, рекомендации 2026 года.
8. [S08](#) (Fisker Owners Association): заявление ассоциации владельцев в деле о банкротстве компании в американском суде, 26 июля 2024 года, с. 2. Оригинальный текст заявления индексировался, однако файловый сервер не ответил на прямой запрос во время итоговой проверки.
9. [S09](#) (Undercurrent Inc.): страница приложения *OceanLink Pro*; описание в (App Store) называет его официальным приложением для членов (Fisker Owners Association; FOA).
10. [S10](#) (Reuters): сообщение об утверждении плана банкротства (Fisker) 11 октября 2024 года.
11. [S11](#) (Reproducible Builds): *Definitions*, техническое определение идентичной сборки.
12. [S12](#) (SLSA): *SLSA Specification Version 1.2*, версия, действовавшая на дату отсечения доказательств.
13. [S13](#) (SPDX): *Specifications*, состав программного обеспечения и его лицензии.
14. [S14](#) (The Update Framework): *Overview*, безопасность обновления программного обеспечения.
15. [S15](#) (Software Heritage): *FAQ*, цели долговременного сохранения.
16. [S16](#) Кен Томпсон (Ken Thompson): *Reflections on Trusting Trust*, 1984.
17. [S17](#) Управление цифрового правительства (Digital Government Authority): *Government Cloud Software Repository (GPaaS)*.
18. [S18](#) Главное управление недвижимости (Real Estate General Authority): *Free and Open-Source Government Software*, официальная недатированная институциональная страница.
19. [S19](#) Дэвид Уилер (David A. Wheeler): *Fully Countering Trusting Trust through Diverse Double-Compiling*, 2010.

Примечание о цитатах: текст использует атрибутированные пересказы опубликованных идей и не приписывает вымышленную прямую речь инженеру, пользователю или исследователю. Заявление заинтересованной стороны и страница приложения не представлены как независимая техническая проверка всех функций.