



战略文章

编辑日期：2026 年 10 月 18 日

在创造者离场之后仍能延续的工程

为什么我们能修复一台远在 240 亿公里外的计算机，现代汽车的车主却要成立组织来修理自己买下的产品？

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

引言：技术主权并不只取决于服务器放在哪里，也不只取决于硬件发票归谁所有。本文提出一个可操作、可检验的定义：如果制造软件的公司消失，一家机构能独立完成软件再生产、运行、修复并验证其完整性的链条，究竟能走到哪一步？本文把旅行者 1 号（Voyager 1）的修复、菲斯克 Ocean（Fisker Ocean）车主与 VanMoof 自行车车主的互助实践，以及可复现构建（Reproducible Builds）联系起来，设想一种面向未来的沙特产业：它销售的不只是技术，也包括把技术交给多年后负责维修者的能力。

证据边界：下文关于探测器、车辆及技术决策的事实均有日期和原始材料支持，并明确说明这些材料不能证明什么。“再生产边界”和“技术继任试验”是本文提出的分析概念，并非沙特已经采用的标准。我们实际进行了两项有限测试：将一个小型软件包构建两次并比较哈希值，以及对一棵虚树依赖树进行确定性检查。结果不含任何沙特系统数据，也不估计冲突发生的概率。

1. 它可以修，却无人能够抵达

2024 年 4 月 20 日，喷气推进实验室（Jet Propulsion Laboratory; JPL）的工程师确认，旅行者 1 号（Voyager 1）的计算机在连续数月发回无法解读的信号后，重新开始发送可读的工程数据。当天，探测器距离地球超过 240 亿公里；信号单程约需 22.5 小时。[S01](#)

没有人能派维修员过去，无法更换故障芯片，也不能购买三年的延长保修。诊断表明，飞行数据子系统（Flight Data Subsystem; FDS）的一块存储芯片已无法正常工作。团队决定把部分程序代码迁到其他存储区域。由于没有一处空间足以容纳整个受损部分，他们必须拆分代码，并调整各部分之间的引用。工程师先测试恢复工程数据的部分；这一部分成功后，其他科学数据才在后续阶段逐步恢复。[S01S02](#)

这不是一台永不损坏的机器。探测器已经老化、能力不断减弱，修复还依赖稀缺经验和特殊通信工具。这也不是证明航天项目已经摆脱供应商的实验。它揭示的工程事实是：一个东西能否修复，并不与我们离它有多近成正比。

同一年，一些菲斯克 Ocean（Fisker Ocean）电动汽车车主遇到了另一种距离。汽车就在家门口，物理部件触手可及，但制造商在 2024 年进入破产程序，车辆维护和部分功能所依赖的数字服务与技术工具随之成为棘手问题。2024 年 7 月 26 日，菲斯克车主协会（Fisker Owners Association; FOA）向美国破产法院提交声明，说明车主需要关键云服务继续运行，还需要获得车载软件架构、维修技师工具和零配件。这是一份利益相关方的陈述，并非法院已经裁定所有服务确已中断，或所有车辆均已无法驾驶。[S08](#)

这些车主做了一件值得所有大规模采购处理器、签署国家级系统合同的人关注的事。他们没有止步于诉讼论坛，而是组织协会、聘请工程师；这一体系中还出现了由 Undercurrent Inc. 开发的 OceanLink Pro 替代应用。应用商店说明把它称为菲斯克车主协会会员的官方应用，并称其在现有硬件、软件和权利允许的范围内提供部分诊断、控制和连接功能。但应用页面来自开发者，不能据此独立证明每一辆车都恢复了全部功能，或每次更新都安全。[S09](#)

这形成了一个传统技术会计很难归类的悖论：人手无法触及的探测器仍可修复；人手能够触及的汽车，却迫使车主变成近似于原制造商的组织。

这里的距离不是公里数，而是拥有产品与拥有重建其缺失功能所需知识和工具之间的距离。

2. 制造者的寿命、物件的寿命，以及两者沟通语言的寿命

再看第三个故事。它的价值不在于像汽车案例，而在于揭示了技术的另一种时间尺度。

2023 年 7 月 17 日，阿姆斯特丹地区法院（District Court of Amsterdam）宣布电动自行车制造商 VanMoof 在荷兰的相关实体破产。破产受托人公布的报告记录了法院裁定，无需依赖公告前的传言。[S03](#)

在那段不安时期，竞争对手 Cowboy 推出了名为 Bikey 的应用。上线说明重点介绍 VanMoof S3/X3 用户把数字钥匙保存到手机，并在原厂服务中断时使用；版本记录随后又写明加入了 A5/S5 支持。该页面没有证明服务器一旦停运所有自行车都会失效，也没有证明应用长期支持所有车型或解决了全部维护需求。[S04](#)

但这家公司并未从世界上消失。新的所有者在 2023 年收购品牌资产；2024 年 2 月，VanMoof 又公开了部分旧车型的维修手册和技术图纸档案。公告称，一些过去只向特定维修点提供的材料开始开放，部分文件还需要从中文翻译。知识由工厂内部转向未来维修者，这是重要一步，却不能证明全部软件代码、认证密钥或零配件都已公开。[S05S06](#)

2026 年又出现了一种不以企业破产为前提的损失：协议老化。

VanMoof 在关于第二代移动通信网络退网（2G Network Phase-out）的说明中指出，旧车型的防盗追踪与远程连接会受到所依赖网络缩减的影响。希望借助其与网络运营商协商的方案保留特定追踪服务的用户，需要在 2026 年 4 月 1 日前启用 Ride Pro 订阅。更关键的是，VanMoof 明确表示，骑行、上锁以及其他独立功能仍能使用；因此不能把这件事写成“自行车停止工作”。[S07](#)

由此可见，三种寿命未必一致：销售产品的公司能存在多久；人们乘坐或驾驶的实体能使用多久；产品与外部世界沟通所用的技术语言能维持多久——从无线频率、加密套件到应用程序接口（Application Programming Interface; API）。

自行车可能在制造商破产后继续存在；公司可能重返市场，而旧车型设计所依赖的通信网络却已经消失。买家也可能拥有设备的每一颗螺丝，却无法再生产一段软件，因为当年使用的编译器已经不可获得，许可证也不再有效。

问题因此比“依赖供应商”更深。我们真正要问的是技术可继承性：一种功能能否传给下一代工程师、设备和软件，就像乐器传给懂得调弦的演奏者？

3. 本文的核心概念：再生产边界

本文提出再生产边界（Reproduction Boundary）：数字服务的所有者在合法且可验证的条件下，能够独立完成的最后一步；越过这一步，便必须依赖另一个供应商独占的资产、决定或知识。

先看软件在呈现给用户之前所经历的路径：

源代码（Source Code）← 编译器与构建工具链（Toolchain）← 软件包与依赖项（Dependencies）← 二进制制品（Binary Artifact）← 发布签名与发布授权（Release Signing）← 运行时与硬件（Runtime and Hardware）← 数据与配置（Data and Configuration）← 验收测试（Acceptance Tests）。

这并非所有产品都遵循的单向链条。依赖会分叉，一些系统使用预构建镜像，一些环节并行发生。但这条路径足以揭示一种常见错觉：机构可能拥有源代码、服务器镜像和数据，却无法更新程序，因为发布签名掌握在制造商手中；或制造商的软件开发工具包（Software Development Kit; SDK）已经不可获得；或硬件接口只接受由另一方签名的固件（Signed Firmware）。

反过来也可能发生：二进制版本可以长期运行，却无人能够安全地修改和重新构建。“现在能运行”和“我能再造一个新版本”是两个不同的判断。

因此需要区分四个不能一概归入“所有权”的动作：

动作	真正的检验	容易混淆的假象
保存（Preserve）	必要资产是否仍有文档、可读取，并且可以合法保存？	有一个内容不明的压缩包
重新构建（Rebuild）	能否把源代码转换为独立的可执行版本，并做到一致或记录差异？	只有源代码
修复（Repair）	能否改变错误行为并通过已知测试？	重新启动旧版本
迁移或替代（Port/Replace）	能否在不同平台上保留关键功能？	只迁移数据而不恢复功能

一家机构可能完成前两项，却做不到第三项；也可能运行一个功能较少、不与旧产品完全一致，但满足约定核心需求的替代品。它不能被称为完全技术等同，而是有意识地重组功能。



再生产边界及技术检验阶段概念图

图 1——拟议的再生产边界：我们实际拥有多少构建资产？哪些资产的可获得性、合法使用或签名完整性仍受制于第三方？本图是一般性研究设计，并非任何沙特机构的依赖关系图。

4. 你可能带走了代码，却丢失了把它变成程序的钥匙

四十多年前，肯·汤普森（Ken Thompson）在著名演讲《Reflections on Trusting Trust》中提出一个动摇“阅读源代码即可放心”这一信念的问题：编译器（Compiler）本身可能携带源代码中看不见的行为。^{S16}

这里无需重述攻击场景，只需保留其与本文相关的思想结论：仅凭源代码，无法解释实际运行版本是怎样产生的。软件背后还有构建工具、库版本、环境变量、汇编规则的完整历史，有时连长期保留的缺陷也已经成为预期行为的一部分。

所以，“我们保存了代码”不同于“我们独立重建应用，并证明产物与目标版本一致”。可复现构建（Reproducible Builds）把可复现性定义为：不同主体使用相同输入、环境与指令，能够得到逐比特一致（Bit-for-Bit）的产物。此时，加密哈希（Cryptographic Hash）可以比较版本，却不能单独证明源代码正确、程序安全，或运行权利可用。[S11](#)

这里有三个彼此不同的技术概念，不能把它们当成同一张“准备就绪”证书的三种名称：

1. 软件物料清单（Software Bill of Materials; SBOM），包括 SPDX 格式，能说明进入软件的众多组件、关系及许可证。但它不能单独保证所有构建工具、发布秘密或硬件都可获得。[S13](#)
2. SLSA（Supply-chain Levels for Software Artifacts）规范中的**构建来源证明（Build Provenance）**有助于记录谁以何种方式、在什么保证条件下构建了产品；但文档本身不会转化为独立使用全部组件的权利，也不会赋予改变每台设备的能力。我们审阅的是当时有效的 SLSA 1.2，而非默认套用更早版本。[S12](#)
3. **软件更新框架（The Update Framework; TUF）**提供机制，用于验证更新真实性，并降低密钥和旧版本带来的部分风险。它不是绕过制造商保护的许可，而是在说明：安全发布更新依赖一套信任基础设施，这套基础设施自身也必须具备延续设计。[S14](#)

机构真正成熟的标志，是能把这些技术区分开。保险库里放着一千枚安全印章，车间却造不出要盖章的东西，听来或许荒诞；当这个“东西”是人们依赖的服务，或经济生产所用的工具时，荒诞便会变成代价。

5. 软件的记忆：磁盘没有保存的知识藏在哪里？

如果计算机知道一切，为什么修复旅行者 1 号仍需要人类持续数月的精细工作？因为代码、内存和报告无法消除**隐性运行知识（Tacit Operational Knowledge）**的重要性：哪些部分可以移动、哪些部分不能破坏，以及一个跨越数十年、各部件渐行渐远的系统中，每个变化究竟意味着什么。

有些知识存在文件里，有些存在人的头脑里，还有些存在设备的实际行为里——它们未必完全符合设计者的预期。因此，只把一个 Git 代码库（Git Repository）从一台服务器复制到另一台远远不够。还要保存构建环境说明、获得许可的依赖项副本、基本行为对照、坦率标明未知部分的培训材料，以及在不危及真实服务的情况下进行测试性修复的能力。

软件遗产（Software Heritage）档案是全球范围内照亮部分解决方案的项目。它致力于保存公开源代码及其开发历史，并让人们能够回溯。它是跨时间保存源代码的重要示范，却不承诺每一份已保存程序都能在未来任意设备上执行，也不构成重新分发闭源材料的许可。[S15](#)

对于老旧信息系统，最稀缺的资产可能不是处理器或磁盘，而是一个简单测试文件：它说明二十年前系统怎样计算某个敏感数字。这个文件能把两个团队相互冲突的意见转化为可检验的结果。当再也没有人知道这个数字原本表示什么时，修改软件就成了一场语言冒险：我们知道一个词怎么写，却不知道它在与现实订立的契约中意味着什么。

6. 实际检验：再生产一个软件包，而不是只记得它的名字

为了不让论点停留在赞美独立性的文章层面，我们在科学附件中实际执行了两项假设公开、范围有限的实验。

第一项实验用 Python 编写一个小型演示服务，包含一项虚构读取功能和两个结果测试。随后在磁盘上两个不同路径中分别创建软件包，不连接外部网络，并固定文件顺序、时间戳和软件包元数据。两次构建的原始字节与 SHA-256 哈希完全一致，两个简单测试也都通过。这只是一个狭窄的实验室成功：它不表示解释器独立于其制作者，不表示第三方验证了结果，也不表示我们重建了任何沙特软件。[MODEL]

第二项实验使用一个包含 26 个节点的抽象图，节点涵盖源文件、数据、构建工具、激活与签名权、硬件以及可运行的替代组件。随后在假定供应商消失的条件下，检查恢复虚构功能的路径。节点数量本身不代表真实系统的深度。[MODEL]

假定恢复路径	供应商离场后不可用的原始要素	按模型能否完成
已保存的运行镜像（Golden Image）	3	没有获准替代项则不能完成
第三方托管的源代码（Source Escrow）	5	原有工具和运行要素缺失，不能完成
源代码加本地构建工具	3	激活、签名及硬件仍由供应商掌握，不能完成
可迁移的最小功能核心（Portable Minimum）	0	在测试、权利和替代项均成立的假设下可以完成

最后一条路径没有保留虚构系统的全部能力；我们假定八项功能中保留六项。这些数字不描述沙特的表现、时间或概率。它们是对再生产成立条件的逻辑检查，不是攻击模拟，也不是企业评估。配套研究说明了依赖模型的结构，证据包则提供其八种情境的结果表。[MODEL]

供应商缺席时的四种恢复方案

依据所附验证表呈现的合成模型结果

恢复路径	不可用的原始根节点	路径完成	假设功能（共 8 项）
保存的可执行镜像	3		8/8 有条件
源代码托管	5		8/8 有条件
源代码与本地工具	3		8/8 有条件
可移植功能核心	0	模型 	6/8

结果来自一个含 26 个节点的合成模型；0 仅表示该模型的既定方案没有留下外部要求。

四种虚构重组方案缺失环节对比

图 2——在虚构依赖树上计算的结果：各路径不可用的原始输入数量，并比较依赖外部供应商的完整功能与范围较窄的替代核心。这里的零不表示现实中没有风险，只表示在给定的虚构图与方案中，没有把任何要求留在方案之外。

从数字上看，保存源代码似乎比保存镜像更弱：缺失五项而非三项。这只在本实验的两种方案内成立，并非普遍规则。源代码为将来的修复提供更广空间，但实现这种可能性需要构建工具和软件包；冻结镜像直接运行时未必需要这些东西。如果不能区分即时运行的时间尺度与未来修改的时间尺度，就会误把这些数字当成源代码价值的最终判决。

更重要的是，一个系统即使拥有 100 个文件中的 99 个，唯一缺失的那个也可能足以阻止新版本发布。这里计算的不是我们拥有材料的比例，而是从材料到目标功能之间是否存在一条完整、有效的路径。

7. 人工智能时代：继承比保存一个代码文件更难

从一辆缺少零件的自行车，转向一个机构今天可以运行、但其处理器或软件工具供应商可能多年后消失的人工智能（Artificial Intelligence; AI）系统。

机构可能会说：我们保存了模型权重（Model Weights）。很好，那么分词器（Tokenizer）、架构定义、数值精度设置、自定义算子（Custom Operators）、加速库（Acceleration Libraries）、推理调度器（Inference Scheduler）、安全标准，以及揭示迁移后性能退化的评估数据呢？

不知道如何正确解释的权重，就像一件弦数齐全的乐器，却没人知道每根弦的长度、张力以及它原本要演奏的音阶。

接下来还有迁移经济性。工作负载从一套加速系统迁到另一套时，推理（Inference）结果、速度或能耗都可能变化。程序能够打开模型文件还不够；它还必须在可接受的成本与延迟下产生约定行为。允许修改和重新发布的权利也需要检验，不能因为文件存在就默认权利存在。

在这里，**可移植性（Portability）**是一项需要付费获得的属性：它要求经过设计的抽象、验收测试，有时还要牺牲性能或功能。专用平台在正常条件下可能效率更高，因此，独立性不能等同于放弃所有先进的全球技术。科学决策要比较专业化带来的回报，与软件跟随唯一供应商一起老化的代价。

设想一家开发人工智能产品的沙特实验室。最有战略价值的投入可能不只是采购更多算力，而是让某个模型能够迁移到第二条执行路径，重新运行一组核心测试，并解释质量、延迟和成本的变化。这不是宣称迁移容易或总能实现，而是准确说明：在第一条路径变成无法更改的命运之前，我们应当检验什么。

8. 沙特：从代码仓库走向再生产工厂

这里的战略建议必须具有操作性。沙特数字政府管理局（Digital Government Authority; DGA）公布了政府云软件仓库（Government Cloud Software Repository / GPaaS）项目，旨在让政府自由与开源代码能够在机构间提供、开发和交换。包括房地产总局（Real Estate General Authority; REGA）在内的沙特机构，也把这一方向表述为促进复用、减少对部分供应商依赖的方式。这是已公开的制度实践，却不能证明每一项政府服务都能脱离当前供应商重新构建。[S17S18](#)

在一些路径中，代码仓库是必要的一步，但它不是检验的终点。

如果沙特技术机构要把这一方向发展成更深的产业，可以从本文所称的**技术继任试验（Technical Succession Trial）**开始，而不是写一份标题很多的合规报告：

第一天：一个未参与产品开发的团队打开一套与供应商服务器隔离的资产包，确认其中实际包含哪些源代码、依赖项、使用合同、工具和环境说明。

第二天：团队尝试构建可执行版本；若二进制一致是正确条件，则比较哈希值；若构建包含非确定性因素，则记录差异原因。

第三天：团队在获得数据所有者许可后，以合成或匿名数据运行测试版本，修改一项小功能，并重新通过验收、安全和性能测试。

第四天：团队尝试在技术与合同均允许的替代平台上运行一个明确的功能核心，不关闭硬件保护，也不仿造他人的密钥。

第五天：团队记录保留的功能、无法完成的部分、修复成本，以及结果发生分歧时谁有权作出决定。这里的五天只是教学性的顺序，不承诺复杂系统能在五天内完成全部试验。

如果试验失败，就应确定失败原因：许可证、缺失工具、软件依赖、技能不足，还是无人知道如何解释的测试？每种原因对应不同的投入：新合同、获准的软件包镜像（Package Mirror）、行为测试、编译器工程（Compiler Engineering），或彻底改变架构。“本地化”一词无法解释这些具体步骤，虽然它有时适

合描述更广泛的政策或经济目标。

9. 面对最强反对意见：所有东西都值得出生两次吗？

不值得。对许多产品而言，专业供应商可能比内部重造每个组件更快、更高效，也更安全。本地重建复杂平台可能提高成本、降低安全修复速度，甚至造出一个比全球竞争更脆弱的本地垄断供应商。在某些行业，合同权利或安全要求本来就不允许用未经认证的软件替换敏感软件。

开源软件也可能维护不足。项目即使拥有全部代码，也可能没有理解代码的工程师。不存在一条公式能保证本地产品必然安全、开源产品必然独立，或分布式系统永远更好。

那么，什么才是明智投入？答案是选择性可复现（Selective Reproducibility）。

需要按照独立修复或迁移能力的实际回报来分类功能：替换供应商有多难；服务价值能维持多久；支持终止会带来多大损失；技能有多稀缺；替代平台有多少；测试成本多高；购买新产品是否比修理旧产品更快。有时，理性的选择是照常购买产品，同时准备退出方案；另一些情况下，若不能尽早证明核心功能可独立运行，就不应采购。

两种判断的差异，不取决于对本地或外国供应商的情感，而取决于随项目变化的技术试验和经济比较。

10. 一个尚未学会准确定价的市场

当软件的寿命超过部分供应商、通信协议又可能早于设备本体老化时，市场可能需要一种更接近演化式维护（Evolutionary Maintenance）、而非备份销售的新产业。

它的产品不是另一块亮绿灯的仪表盘，而是能够在隔离环境中自动重建旧版本的工具、复活遗留代码的实验室、转换废弃接口的工具、跨硬件世代运行行为测试的软件包、把二进制版本与源代码和构建环境对应起来的独立验证服务，以及在供应商停止维护后保留设备或服务最低功能的开放组件。

沙特可以在计算、数字服务和人工智能投资中，研究把这类技术出口给不愿在合同终止时看着软件变成博物馆藏品的机构。但一个产业不会因为研究中心取了名字就自动出现；它需要付费客户提出的挑战、获准使用的测试数据、清晰的分发权利，以及修复原系统与彻底替换它之间的可行性比较。

因此，面向 2035 年的前瞻问题是一个市场问题，而不是国家口号：谁会出售一种能力，让别人的产品在第一代生命周期结束后仍能继续生存？

赢家可能是一家精通旧协议迁移的小公司，一个造出可审计编译器的团队，或一家把人工智能模型迁到第二个平台且没有造成质量崩塌的实验室。现在无人能宣布它们注定胜出，但其价值机制已经足够清楚，可以设计市场实验加以检验。

11. 结语：无法装进箱子的遗产

旅行者 1 号的工程师在遥远到无法进行普通往返对话的距离上修复旧代码时，克服的不只是地理。他们拥有足够的系统说明、测试工具、经验，以及修改并上传程序的权利，因此能为一个物理上无法触及的对象开辟修复路径。[S01](#)

菲斯克 Ocean (Fisker Ocean) 车主在制造商陷入困境后成立工程协会和维护项目，并不是因为汽车缺乏技术。恰恰相反，汽车充满技术；问题在于，它继续演化的部分能力仍与一家不再如昔的经济组织绑定。[S08S09](#)

本文的论点由这段无形距离产生：技术产品是一项会消耗的资产；在制造者离场后再生产、验证和修复它的能力，则是一项独立的经济能力。

建立这种能力可能需要一笔并非每个产品都值得投入的资金。试验也可能证明原供应商仍优于所有替代者。但最好在真正需要修复之前，就知道答案。

机器有两个历史时刻：它被制造的那一天，以及所有者第一次需要修复某个连制造者也无法修复的问题的那一天。两者之间，一个国家可以学习一种稀缺的技术艺术：购买今天有用的东西，同时保留一条道路，使自己明天仍能理解它，并重新造出其中不可替代的功能。

推理说明

1. NASA 与 Fisker 的材料都不能证明沙特设施存在同类薄弱点。前者显示了在严苛限制下成功开展软件修复；后者显示了用户把部分维护能力转移到自身的社会与实践过程。
2. VanMoof 在 2023 年的相关实体破产，并不表示该品牌目前已经消失。新的所有者接手后建立了维修档案。2026 年 2G 网络缩减也没有使自行车无法骑行；公开说明中的影响仅限于部分车型的防盗追踪和远程连接。
3. 重复构建实验只证明一个小型虚构软件包在所述环境中得到一致产物，不构成对生产软件包的独立认证。依赖树检查不提供概率、恢复小时数或国家级技术独立评分。
4. 构建或迁移能够独立完成，不等于有权绕过许可、签名或车辆安全限制。任何真实系统的执行试验都需要授权、测试和工程安全保障。
5. 来源细节、作用、事件与发布日期以及推理边界，汇集在证据包内的来源登记表 ([source_register.csv](#)) 与事实案例簿 ([news_casebook.md](#)) 中。“再生产边界”提案由本文作者提出。

精选来源

- 美国国家航空航天局 (NASA) , *Voyager 1 Resumes Sending Engineering Updates to Earth*, 2024 年 4 月 22 日。[S01](#)

- 阿姆斯特丹法院及 VanMoof 破产受托人，公开破产报告，2024 年 6 月；所记事件发生于 2023 年 7 月 17 日。[S03](#)
- Cowboy, Bikey 应用原始页面，2023 年 7 月。[S04](#)
- VanMoof，2024 年 2 月 27 日维修档案公告及 2026 年 2G 网络更新。[S06S07](#)
- 菲斯克车主协会（Fisker Owners Association）2024 年向破产案提交的声明，以及 Undercurrent Inc. 提供的 OceanLink Pro 应用页面；页面说明称其为 FOA 会员的官方应用。[S08S09](#)
- Reproducible Builds 项目，可复现构建的技术定义。[S11](#)
- SLISA 1.2 规范；该版本于 2025 年 11 月 24 日获批，并在证据截点时保持为已批准版本。[S12](#)
- 沙特数字政府管理局（Digital Government Authority），政府代码托管项目。[S17](#)