



探索性研究论文

编辑日期：2026 年 10 月 22 日

从事故报告到可执行合约：软件失效知识出口能力的概念与计算研究

From Incident Reports to Executable Contracts: A Conceptual and Computational Study of Exportable Software-Failure Knowledge

科学分类：一篇概念研究论文，辅以范围明确的初步证据回顾，以及在合成用例与环境实际运行的软件概念验证。已完成：指定来源的提取、模型设计、代码编写与执行，以及结果文件检查。未完成：使用语言模型从真实报告中可靠地自动提取合约、在客户产品中开展实验、研究真实沙特公司的经济数据、独立盲测或学术同行评议。

中文摘要

本研究考察：软件与分布式系统事故中有记录的经验，能否转化为可以运行、迁移，并在开发国以外销售的技术资产。研究不以“某国承受过区域压力，便会自然形成出口产业”为前提，而是区分四个层次：描述事件；在明确限制下提取可概括的失效机制；编写带有可审计判定规则的可执行测试；证明外国买方获得的价值高于安装与维护成本。证据回顾连接三起有记录的事件：2025 年 11 月 18 日，Cloudflare 因权限变更后特征文件翻倍而中断；2025 年 6 月 12 日，Google Cloud 因配额数据存在空字段且恢复重试造成压力而中断；2024 年 2 月 22 日，AT&T Mobility 因一次影响广泛的网络配置错误而中断。随后，研究在为此构造的 16 份测试合约与 4 个实现上运行合成实验，共产生 64 个结果；四个实现分别有 7、12、11、16 个结果与参考测试一致。这些数字由软件与合约的设计决定，既不衡量生产环境有效性，也不表示事故预防概率。值得独立检验的商业假设不是“用人工智能（Artificial Intelligence; AI）制作混沌测试”——这种实践已经存在——而是语义与判定规则能否跨环境迁移、迁移成本多高，以及买方需求能否持续更新。

关键词：可执行失效合约（Executable Failure Contracts）；语义迁移（Semantic Transfer）；基于属性的测试（Property-Based Testing）；受控实验（Bounded Experiment）；推理软件（Inference Software）；混沌工程（Chaos Engineering）；产品单位经济效益（Product Unit Economics）；软件服务出口（Software Services Exports）。

English abstract

This conceptual and computational working paper examines whether documented software failures can be transformed into portable, independently inspectable test artifacts with exportable commercial value. It distinguishes incident evidence, failure-mechanism abstraction, executable-oracle construction, cross-system semantic transfer, and demonstrated foreign willingness to pay. Three primary records anchor the inquiry: Cloudflare's November 2025 feature-file propagation outage, Google Cloud's June 2025 quota-policy failure and recovery congestion, and the February 2024 US AT&T mobile-network outage investigated by the FCC. A reproducible, fully synthetic Python demonstration manually specifies 16 toy test contracts spanning input schemas, bounded feature files, authenticated versioning and retry limits. Four intentionally constructed toy adapters yield 64 executions and respectively match 7, 12, 11 and 16 declared oracle outputs. The exercise demonstrates contract encoding and repeatability, not predictive incident prevention or product-market fit; all outcomes are consequences of artificial design choices. The paper proposes prospective tests for cross-platform semantic transfer, oracle accuracy, false positives, incremental value over established chaos-testing products, integration cost and actual foreign renewals. The central claim is a falsifiable opportunity hypothesis, not evidence that a Saudi deep-tech resilience-export industry already exists.

Keywords: executable incident contracts; semantic transfer; chaos engineering; test oracle; fault injection; Saudi technology exports; reproducibility; product-market fit.

1. 科学问题，以及知道答案会改变什么？

系统可靠性文献主张理解故障并为其做准备，却不能证明每一种自称提高韧性的新工具都拥有独立市场。这里所说的“出口产业”，比本地工程团队解决自身问题的能力更具体，也更具有经济含义。我们希望检验以下假设：

某些软件失效知识可以从有记录的事故或获准环境中提取，被表示为相对独立于供应商的测试合约，以合理的边际成本迁移到不同客户系统，再反复出售给沙特境外的买方；这些买方付费，是因为产品带来了经过证明的工程收益。

这是一个复合假设，证明其中一部分，不等于证明整体。技术上或许能够制作测试，却不能在平台之间保持语义；语义或许可以迁移，但每个项目的定制成本过高；团队或许能向国际客户销售专家服务，却仍未形成可重复的软件产品模式。答案会改变专业方向与投资决策：应该资助实验室建设可复用的适配器和规范，投资专业咨询团队，还是直接使用现有工具？

科学分析单位：一份明确的测试合约，在定义清楚的系统与条件下执行，并把结果同公开判定器比较。

商业分析单位：独立外国客户在明确的销售与维护周期内购买产品或续约。两者不能等同。

2. 拟议项目出现以前，市场已经有什么？

2.1 故障实验早已有工具，不能假设它是沙特的新发现

混沌工程原则（Principles of Chaos Engineering）提出：围绕稳定状态行为建立假设，以观测为基础开展实验，并缩小实验的爆炸半径（Blast Radius）。这些原则并没有说，团队可以在不了解系统的情况下，直接从公开报告生成实验。AWS 故障注入服务（AWS Fault Injection Service; AWS FIS）自 2023 年起提供场景库，AWS 在 2024 年发布的指南中对此作了说明 [S06](#)。它支持与特定告警关联的停止条件，不只是关机开关 [S08](#)。2025 年，亚马逊（Amazon）又发布一项工程示例，在自有环境中使用大语言模型（Large Language Models; LLMs），从自然语言描述生成实验模板 [S07](#)。任何声称“把文字转成测试模板”本身具有新颖性或没有竞争的说法，都与这些证据冲突。

Jepsen 项目自 2013 年起发布数据库、队列与分布式消息系统的一致性测试报告。它运行真实实现，生成操作，注入网络分区、时钟偏移等故障，再依据公开的一致性模型检查输出历史 [S09](#)。因此，潜在增量必须精确定义：产品可能为随事故更新而变化的用例库提供统一表示，也可能提供跨供应商的语义验证，或向买方交付可复现的证明。不同方案面对不同竞争者。在审查采购行为并进行对照测试前，我们无法判定市场缺口。

2.2 初步证据中的三类失效机制

以下记录构成研究框架，但它们不是各类技术故障发生频率的代表性样本：

事故	来源记录了什么	拟议抽象机制	我们不知道什么
2025 年 11 月 18 日，Cloudflare	权限变更后，查询结果差异使特征文件翻倍；文件超过读取器接受上限，服务反复波动 S01	分布式中间输出意外膨胀，读取器存在隐藏的大小假设	在其他系统中去重测试是否足够？并不足够；元素意义与顺序可能不同
2025 年 6 月 12 日，Google Cloud	缺失字段的配额数据在全球分布式路径触发软件异常；恢复期间的重试给后端存储带来压力 S02	不完整数据激活未经测试的路径，随后引发恢复拥塞	我们没有内部系统的可执行副本，无法复现原始故障
2024 年 2 月 22 日，AT&T Mobility	一项网络组件配置错误导致全国通信中断；监管报告称 9200 万通电话被阻断，超过 2.5 万通紧急呼叫未能接通 S03	控制组件变更的影响扩散，并触发大规模重新接入	这些数据不足以定义另一运营商上的完整安全实验

前两个来源由运营方自行撰写。发布者掌握直接技术知识，但在事故叙述中也有自身利益；我们没有拿到未公开的原始运行日志进行比对。第三个来源是相对独立的监管调查，其中部分数字仍来自运营商披露。这些限制均保留在主张记录中。

2.3 证据说明出口通道存在，却不能证明这项产品已有市场

据沙特通讯社（Saudi Press Agency; SPA）报道，2025 年 4 月，28 家沙特本国科技与通信企业参加 GITEX AFRICA “沙特制造”展区 S04。2026 年 7 月，SPA 又报道 Elm 与 Huawei 在 LEAP East 期间签署谅解备忘录 S05。这些资料只记录机构参与和国际活动，没有独立测量拟议测试行业的已实现出口、经常性订阅、外国客户基础或产品利润率。它们说明企业可以寻找销售渠道，不能作为证明论点成功的因变量。

3. 定义准备出口的东西

我们把拟议的可执行失效合约（Executable Failure Contract）定义为包含五个必要部分的记录：

1. **来源（Provenance）**。提取假设所依据的记录链接、段落位置、来源独立程度，以及因果关系是来源明确声明还是研究者推断。
2. **前置条件（Preconditions）**。使实验具有意义的应用、存储与协议属性；同时设置约束，防止把实验误导性地套用到不同系统。
3. **实验刺激（Stimulus）**。合成输入、隔离环境中获准执行的变更，或具有明确范围、期限与停止边界的人造操作模式。
4. **行为判定器（Test Oracle）**。将可接受结果与不可接受结果分开的可观测属性或功能不变量，并说明如何观测行为及其不确定性。
5. **证据包（Evidence Package）**。软件版本、运行设置、权限、事件与测试日志、结果、缺失信息，以及不可据此推断的内容。

该单元需要通过**中间表示（Intermediate Representation; IR）**实现迁移，但迁移不能简化为字段名之间的机械对应。若一份测试合约假设重复元素可被删除且不改变文件含义，把它转换到有意保留重复的服务中，就会破坏合约。因此，系统需要功能描述与语义条件，不只是序列化格式转换（Serialization）。

下面给出一个说明性结构，它不是已经认可的标准：

```

incident_spec_version: proposed-v0
provenance:
  source_id: S01
  claim_boundary: documented mechanism; not original payload
applicability:
  data_semantics: order-insensitive feature set
  environment: isolated local simulator
stimulus:
  type: repeated_features
  records: [a, b, a, b, a, b, a]
oracle:
  acceptable: ACCEPT after semantic deduplication
  forbidden: uncaught parser exception
safe_execution:
  production_connection: false
  stop_after_seconds: 10
  rollback: discard isolated fixture
limits:
  does_not_reproduce: Cloudflare production code or actual outage

```

来源字段中出现“Cloudflare”，不表示该用例抹平了 Cloudflare 与其他系统之间的技术差异。如果使用者不能证明目标文件是集合，而非拥有语义顺序的列表，就应拒绝在其上应用这份合约。拒绝翻译（Abstention）优于给出自信却毫无意义的结果。

4. 最难的问题：迁移的是意义，还是只有命令？

可迁移性（Portability）包含三个不同层次：

- 语法迁移（Syntactic Portability）：文件可以被读取，并转换为另一引擎接受的有效模板。
- 操作迁移（Operational Portability）：测试能在权限边界清楚的环境中执行，并记录所需数据，不危及在线系统。
- 语义迁移（Semantic Portability）：新操作测试的是同一属性，并依照新应用的功能，对可接受与不可接受用例作出一致判断。

前两个层次可能成功，第三个仍可能失败。例如，模拟一个云可用区中断，或许会使测试网络断开，却不能等同于统一复制策略向多个区域传播时发生退化。类似地，一份包含大量时序信息的追踪记录（Trace），也不能单独揭示事务丢失或重复，除非我们先定义并发操作之间的逻辑关系。

为明确问题，设两个实验系统为 A 和 B ，执行轨迹为 τ_A 和 τ_B ，观测抽象变换为 α_A 和 α_B ，共同属性为 φ 。主张实现语义迁移，大致要求我们在适用条件内，对等价轨迹表示上的属性判定进行比较，而不能只观察两个测试是否都未抛出异常。意义要求可写为：

$$\text{MeaningMatch}(A, B, \varphi) := \text{Comparable}(\alpha_A, \alpha_B, \varphi) \wedge \text{OracleSound}(A, \varphi) \wedge \text{OracleSound}(B, \varphi).$$

这是概念表达，不是程序等价证明，也不是自动决策规则。有些属性无法在特定环境之外保持，例如金融交易语义，或在另一制度下发生变化的法律权限。

5. 实际执行的实验设计

5.1 有限目标

我们没有构建语言驱动的事故编译器，没有对 Cloudflare、Google 或 AT&T 运行测试，也没有导入其系统样本。我们制作了一项概念验证演示（Proof-of-Concept Demonstrator），只回答一个问题：能否把一小组行为不变量写成合约，使其在确定性的本地环境中，对多个独立实现产生可比较输出？

示例全部由人工编写。共 16 份合约，四个家族各四个用例。每个用例包含 `case_id`, `family`, `description`, `stimulus`, `expected` 与 `observed_property`。随后，我们编写四个测试实现模块，每个模块配备不同的防护组合。程序使用 Python 及其标准库，不连接网络。数字记录还包含测试用例配置的 SHA-256 哈希（SHA-256），以便发现输入替换；这并不构成外部安全验证。

5.2 合约家族

策略输入（Policy Schema）。正常数值配额、缺失配额、负数配额，以及无法转换为数字的配额。拟议条件是：无法解释数据时，应有意拒绝输入，而不是让未处理异常中断处理路径。这项策略并不表示每一项生产服务遇到字段缺失都必须关闭。

特征包（Feature Bundle）。小型集合、允许出现重复的集合、原始大小超过上限但去重后低于上限的集合，以及超过上限的唯一元素集合。只在这个合成示例中，我们把语义定义为元素顺序与重复不改变意义，因此可以去重。真实系统未证明这一属性时，不得迁移该合约。

更新版本（Update Version）。经过认证的新更新、经过认证的相同版本重复发送、经过认证的旧版本，以及签名无效的新版本。拟议条件是：程序不能应用回退或未经认证的更新；在结果有明确规定时，可以接受同一版本重放。我们没有模拟真实签名或密钥管理基础设施；字段 `signature_ok` 是预设的合成值。

有界重试（Bounded Retry）。直接成功；前三次失败、第四次成功；连续失败 12 次；连续失败 6 次。示例上限为总共四次尝试。判定规则区分上限内合法成功与耗尽尝试后的停止。代码不会运行真实网络或随机退避，也不测量拥塞对数据存储的影响。

5.3 四个合成实现

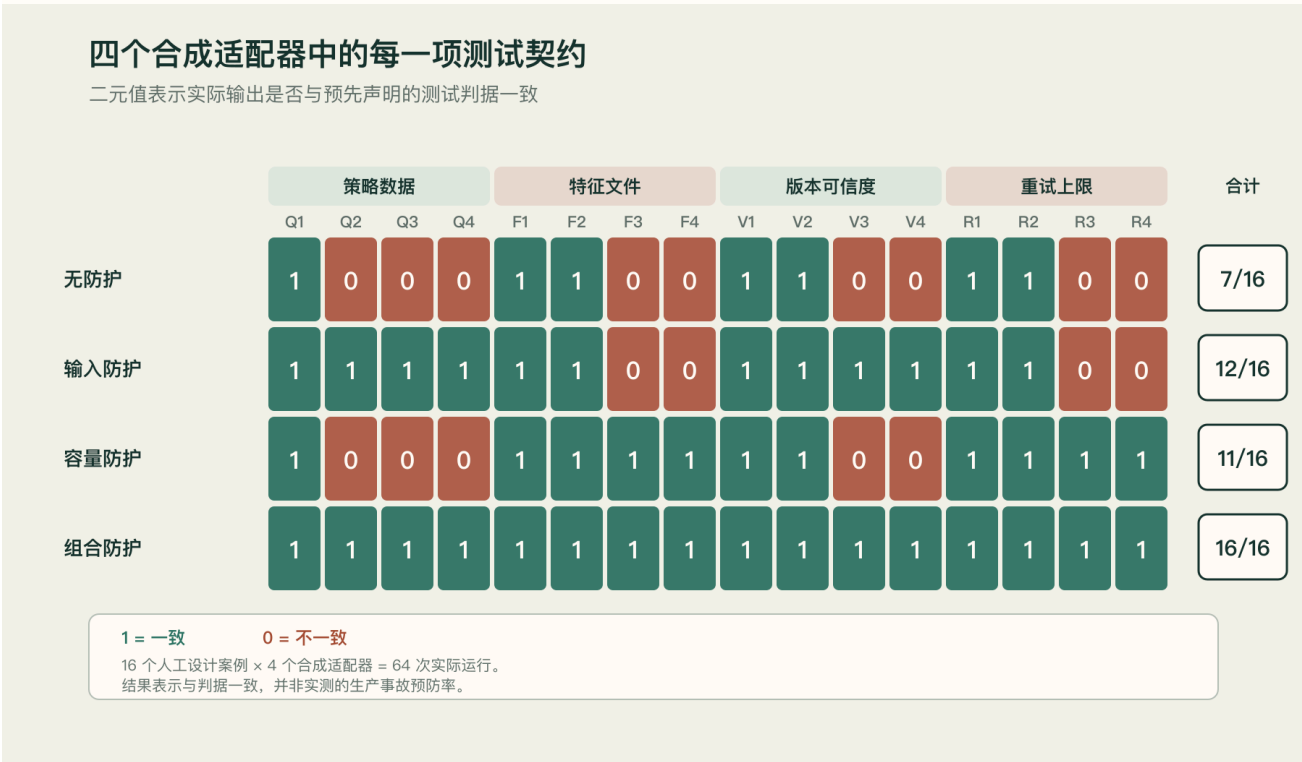
1. **无防护（Naive）**。直接转换字段，用原始列表长度和会抛出异常的上限处理特征，接受更新版本，并重试过多次数。
2. **仅输入防护（Input Guard Only）**。检查配额类型与取值、更新版本和签名；特征包与重试行为仍然脆弱。
3. **仅容量防护（Capacity Guard Only）**。处理特征重复并限制重试；策略与版本检查仍然脆弱。
4. **组合防护（Combined Guards）**。汇集示例合约明确要求的全部检查。

每个实现都是有意设计，只为展示有限差异。第一个实现不代表一家真实的糟糕公司，第四个实现也不是经过验证的安全设计。测试与防护在同一个项目中构造，因此第四个实现成功，不能成为某种开发方法在现实世界中更优的统计证据。完全可以设计出让四个实现全部失败的新用例。

5.4 执行与记录结果

运行 `analysis/reproduce.py` 后，程序生成 `contract_cases.json`、`contract_test_results.csv`、`contract_summary.csv` 与 `execution_meta.json`。单项结果数量为 16 份合约 × 4 个实现 = 64 次执行。每一行都记录预期结果、观测结果与是否一致，而不是记录“成功防止了一起事故”。

合成实现	策略输入 (满分 4)	特征包 (满分 4)	版本 (满分 4)	重试 (满分 4)	总分 (满分 16)
无防护	1	2	2	2	7
仅输入检查	4	2	4	2	12
仅资源边界	1	4	2	4	11
组合防护	4	4	4	4	16



合成数据上 64 次执行的结果矩阵

图 1. 根据 `data/contract_summary.csv` 计算的输出。分数衡量人工设计的输出与一同设计的判定规则之间的一致程度，不衡量市场有效性或真实事故概率。

可以看到，每条部分防护路径在设计用于处理的家族中成功，在其他家族中仍不完整。但这些结果几乎是预先决定的；它们没有统计假设检验意义，也不具有程序之外的泛化价值。其价值属于教学与架构层面：统一场景语言，把判定规则从执行机制中分离，并以可比较、可重新生成的形式保存每份合约的结果。16/16 不能解释为系统永不故障。

6. 控制、验证与敏感性：做了什么，没有做什么

已执行：使用不含随机性的固定规则；导出全部 64 个用例；通过程序检查合约数量与每个实现的总分；记录用例内容哈希。用户可在没有外部依赖的 Python 3.10+ 环境中运行代码，并在重跑后比较文件。这是该示例的计算可复现性（Computational Reproducibility），不是对自然事件的复现。

未执行：把合约映射到生产系统；独立审查判定器是否正确；测量非作者团队的执行时间；分析客户接受度；开展渗透测试；评估真实语言模型；以大量合成随机种子重复运行；测试自适应环境的变化。示例没有置信区间或概率值，因为这组用例不是统计样本。64 次执行不会因此变成 64 起互相独立的故事总体。

应增加的重要敏感性：如果去重语义从无序集合变成有序列表，会怎样？如果旧更新是获准回滚流程的一部分，会怎样？如果某项服务的资源不受重试伤害，超过四次重试恰是商业所需，会怎样？如果为了避免不安全放行，本来就希望未捕获异常使流程停止，又会怎样？一旦系统功能改变，第四个实现的一部分结果就可能从“正确”转为“违反合约”。这正说明每次迁移都需要独立的语义校准。

7. 能运行的测试与有用的测试之间

场景从真实新闻出发，容易让结果获得不应有的权威。事故文本并不是造成事故的软件的完整规范。运营方可能为了保护系统而省略信息，表面相似的工具可能赋予字段不同功能，调查也可能建立在不完整证据上。因此，未来实验必须区分四种结果：准确表示原事故；抽象表示一个机制家族；只模拟机制表述的教学示例；不满足适用条件、因而不应运行的实验。

当前示例依合约不同，属于第二类或第三类；它与历史系统不存在已经证明的对应关系。对错误合约而言，PASS 也可能是正确结果。人工智能驱动的工作还有一个问题：语言模型可以生成语法正确的实验，却用报告未提到的关系填补空白。它可能编造比例、字段名或重复语义。因此，来源依据准确性（Evidence Grounding）、合约编译有效性（Compilation Validity）与判定器语义有效性（Oracle Validity）必须分开验证。

7.1 好测试也应包含工具自身必须失败的情形

不知道系统功能时，产品应该拒绝生成实验。我们建议设置以下负向校准证据：

- 事故文本隐藏了必要字段的定义，系统应要求进一步调查。
- 报告中的因果解释后来被发布者撤回，系统应撤销合约或修改版本。

- 服务处理有序列表，系统不应只因相似用例在另一实现中成功便进行去重。
- 环境缺少适当的停止或恢复边界，系统不应发布可执行实验。
- 监控记录不足以证明判定结果时，测试应被标记为“无定论”，而不是“成功”。

只计算生成测试的数量，可能奖励一个说得很多、实际什么也没测的平台。更接近真实意义的指标，是通过来源、语义、执行与结果检查的独立合约数量，以及客户版本变更后仍然有效的数量。

8. 产品的初步工程架构

独立概念可以由七个边界清晰的组件检验：

A. 版本化事故语料库 (Versioned Incident Corpus)。保存链接、日期、证据类型、版本与更正。不在未经许可时重新分发受保护内容，只存储来源元数据、获准摘录或检索指针。

B. 假设提取器 (Hypothesis Extractor)。使用本地或依法可用的语言模型提出机制候选，区分观测与推论，并声明信息不完整程度。导入报告中的文字指令不会被当成系统命令。

C. 语义适用性检查器 (Semantic Applicability Checker)。询问文件功能、回滚权、版本、共享状态，以及能否定义合适的判定规则。它可以向客户工程师提出问题。

D. 中间表示编译器 (Intermediate Representation Compiler)。将获准合约转为用例集合与受控故障注入程序。它把用例类型与执行工具分开，无法执行的信息保留在 `unsupported` 字段中，不直接丢弃。

E. 有界执行适配器 (Bounded Execution Adapters)。首先在模拟或隔离测试环境中运行，使用最小权限、时长与负载上限，以及停止与回滚机制。市场工具本身也需要配置，因此不能假设这些保证绝对存在。

F. 判定与轨迹分析器 (Oracle and Trace Analyzer)。将输出与公开属性比较，并标出缺失轨迹或无定论行为。适合时可利用 OpenTelemetry [S11](#)，但不能把事务完整性缩减为成功请求率。

G. 证据复现包 (Evidence Replay Bundle)。说明系统版本、输入、判定、注意事项与限制。独立买方团队可以在自己的实验室重跑，也可以在条件不一致时拒绝运行。

从事故到测试的拟议链路

五次知识转换；每一阶段都需要证据和面向具体应用的验证



从事故记录到可审查测试产品的转换路径

图 2. 概念生产链。各阶段不是已存在的沙特系统或已发布标准；在真实环境使用前，每个接口都必须经过验证并受到限制。

潜在竞争不在于从零开发所有组件。适配器可以建在已有接口上，包括故障实验工具与开放监控项目；真正需要证明的差异价值，是降低意义迁移成本，并让不同客户都能清楚理解测试证据。平台也不只存在于云端，因此产品可能需要不连接客户网络的模块，或为客户容器软件生成输入的库。不过，这应由具体采购场景决定，而不能只是设计口号。

9. 产品经济：从第一次实验到第三位客户

我们以描述方式定义三个相互竞争的商业模式：

1. 定制可靠性咨询（Bespoke Reliability Consulting）。客户购买专家时间，由专家检查系统并设计专门实验。每份合约收入可能很高，但大多数工作会重复，利润率取决于工程师工时。
2. 许可测试包（Licensed Testpack）。客户购买一套可以有限修改、反复运行和维护的测试库。卖方必须提供适配工具、版本规范和有效性标准，否则支持请求会不断增加。
3. 订阅式执行证据服务（Subscription Execution Evidence）。客户付费，让实验持续关联不断变化的版本。若价值能够持续更新，订阅会带来经常性收入，同时也产生运行、存储、安全与隐私责任。

为避免宣布一个尚未得到证明的市场，我们为每位外国试点客户建议以下产品单位测量：

符号	变量	建议测量方式
H_i	初始设置工时	把工作与客户及版本对应起来的明确工时记录
H_r	新版本产品更新工时	不含与产品无关商业工作的升级投入
H_c	客户专属支持工时	无法推广到其他买方的介入时间
P	已付价格	已实现合同收入，不是价目表或初步谅解
V	有记录的收益	客户原有基准未发现、产品额外发现的相关用例，并附执行成本
R	付费续约	审查第一版结果后实际延长合同

可以初步定义可复用工程占比（Reusable Engineering Share）：

$$S = 1 - \frac{H_c}{H_i + H_r + H_c}$$

如果全部工时非负且分母为正，该比例位于零到一之间。但它不是毛利率（Gross Margin），也不能单独衡量产品质量。比例高，可能表示平台良好，也可能表示客户需要的支持根本没有得到提供。比较时必须一致定义期间、系统与买方。当前研究没有测得该比例的任何数字。

买方还可以用故障概率、损失规模，以及测试使事故避免概率发生的变化，构造预期价值的假设上限，再扣除实验成本、价格与副作用。然而，因果影响很难直接识别：成功测试之后可能发生全面重构；故障减少也可能来自培训改善、需求下降或架构变化；合同期内甚至可能没有事故发生。因此，我们不根据客户故事或已执行合约数量推断投资回报。

10. 如何区分可销售产品与伪装成平台的服务？

实际区分需要项目内部公开的测量面板，而不是营销称号。一个可扩展软件产品在迁移到使用相似架构的新客户时，应逐步降低平均定制投入；客户产品升级后，它仍应保留有用结果；买方工程师也应能在卖方开发者不必每次到场的情况下运行它。专业服务即使这些特征很弱，也可能取得商业成功，但那是另一种业务，采用不同的计算方式。

市场实验可以先从容易取得明确测试许可的民用行业开始：多租户企业软件、预生产环境中的非敏感订单处理平台，以及通用配置管理组件。成功得到证明后，再研究控制条件下更敏感类别。进入市场并不要求公开关键设施模拟细节，也不要求让产品连接私有运行数据。

可设置一项严格的初步进入标准，作为拟议未来协议，而非已经取得的结果：三位互相独立的国际客户，使用两个不同平台；其中至少一位能在没有卖方团队介入的情况下重跑测试包；同时记录设置成本、反馈，以及迁移失败的用例。这是为了推动可证伪测试而设定的启发式门槛，不是已经发表的公司分类科学规则。

11. 一项可能推翻这个想法的研究设计

11.1 样本与比较

初始档案应来自原始公开事故，来源独立性与细节程度须预先分类。训练机构与测试机构必须分开；若要衡量泛化能力，训练集中不能包含与保留用例同一家族的事故。事故还应按发布日期分割，不能让在更新档案上训练的语言模型，在一项声称合约发布时尚无相关信息的测试中使用未来知识。

第一项比较是工程师用当前工具手工编写测试；第二项是云服务商的预设场景库；第三项是没有额外语义检查的自然语言实验模板生成，这类实践已见于市场 [S06S07](#)。各组可用时间应尽量相同，并记录工程师技能，否则团队能力差异会被误写成工具差异。

11.2 输出与验收标准

- **机制来源准确性 (Provenance Accuracy)**：机制是否指向原始报告的具体位置，且没有增加来源中不存在的细节？
- **判定器一致性 (Oracle Agreement)**：独立专家在了解系统功能后，是否同意预期结果？分歧必须记录，不能删除。
- **适用性精确率 (Applicability Precision)**：被系统判定适用的合约中，实际核验后确实适用的比例是多少？
- **迁移产出率 (Transfer Yield)**：在定义的样本内，迁移到新环境后仍保持语义正确的合约比例；必须披露分母与排除项。
- **增量故障发现 (Incremental Fault Discovery)**：与原有工具相比，产品额外发现了多少真实、相关的缺陷；重复结果与误报应剔除。
- **边际集成成本 (Marginal Integration Cost)**：每增加一个环境和版本，卖方与客户在开发、文档和运行上投入的工时。
- **外国付费留存 (Foreign Paid Retention)**：实际使用周期后续约的外国买方数量及净付款，而不是报价或谅解备忘录数量。

证伪门槛：如果在同等时间与成本下，产品在盲测中不能比现有工具发现更多功能缺陷；如果每个项目所需咨询投入始终不下降；或外国买方拒绝续约，独立软件出口产业假设便会减弱。即使如此，产品仍可能保留教学或咨询价值。

11.3 因果推断

客户购买软件包后，即使事故减少，也不能在没有对照组或分析设计的情况下把下降归因于产品；分析必须处理版本变化、使用强度与其他修复工作。早期与产品关系最直接的输出，是发现新的、可复现的缺陷，或缩短得到可靠测试的时间。声称预防未来事故，需要更长观察期与反事实解释。严重事故本来就很少见，因此即使经济价值很大，也很难用统计方法证明事故减少的收益。

12. 权限与安全必须是执行属性，而不是管理口号

一项测试即使发现失效，只要它在授权范围外造成真实中断，就不配称为科学成功。所有软件包都应从隔离或预生产环境开始；若要在在线环境中注入故障，必须取得明确的范围许可，并设置时长、负载与资源边界，以及与有效指标相连的停止规则。AWS 文档列出了基于特定监控告警的实验停止条件 [S08](#)。这表明安全属于实验设计，而不是末尾附加的一句声明。

可以销售或发布的软件包不应保存密码、连接日志，或敏感设施的精细依赖图。公开事故名称必须与客户运行数据分开；从公开报告提取信息时，也要尊重内容权利与许可。建立全球错误分类，无须知道某家机构使用的秘密规则，更无须发布内部账户信息。使用人工智能时，应隔离不可信来源材料与运行指令，不能让文本提取结果自动获得执行权限。

13. 这项研究为沙特至 2035 年的问题增加了什么？

研究区分四个价值层次：

1. 本地专业能力（Local Expertise）：沙特团队能够解决真实问题；这一点很重要，却不足以证明出口能力。
2. 可编码知识（Codified Knowledge）：经验被转化为可检查的规范和证据；这仍不能证明可迁移性。
3. 可迁移产品（Transferable Product）：测试能以有限成本在新客户处保留近似意义；这仍不能证明需求持续存在。
4. 可重复出口产业（Repeatable Export Industry）：独立买方付费并续约，产品收入足以维持维护与改进；这是经济结果，需要不同于工程证明的证据。

这些只是供思考的拟议描述层次，不是官方指标。不能从第一层直接推断第四层，也不能用 28 家企业参加国际展会，替代对持续技术收入规模的了解 [S04](#)。

在这项假设下，较有价值的沙特研究投入或许是一座范围有限的实验室，而不是综合服务公司：建立多样、有记录的事故集合，制定条件编码协议，开发隔离实验的开放适配器，让人工编写的方案与语言模型提出、工程师审查的方案进行盲测，并设置防止能力主张超出结果的发布标准。如果在独立测量下成功，再考虑专业产品与外国客户。如果失败，也应公布失败，因为失败本身正是这个产业希望学习的研究材料。

14. 研究限制与任务边界

计算方面唯一完成的结果，是 64 个输出与预先规定的合成合约一致。外部回顾只覆盖选定的一手来源，不是完整系统综述（Systematic Review），也没有使用预先登记的排除规则筛选索引文献。我们没有计算市场规模、合同价格、事故概率或真实投资回报，也没有验证概念模型是否具有绝对学术新颖性、能否产生专利，或是否独立于现有工具。

可以接受的结论更窄，却依然严肃：我们的小型示例证明，在技术上可以把有限的输入数据、更新与负载失效家族写成可运行合约，并输出可检查证据。市场上已有成熟工具支持相近形式的故障工程及自然语言生成；因此，假设中的增量价值应在语义迁移准确性、定制成本下降与国际重复付费三方面接受检验。没有这些测试，拟议产业仍只是研究机会，而不是关于沙特新经济的既成事实。

15. 可用性与复现

`data/` 文件中的数据全部为本研究生成的合成数据，并在软件包中作为教学示例授权使用，不是第三方数据。代码位于 `analysis/reproduce.py`，除 Python 标准库（Python Standard Library）外没有依赖。在研究根目录运行脚本，即可生成相同文件及测试用例哈希。导出的图形提供矢量（SVG）与位图（PNG）格式，电子表格（Excel）用于目视检查公开公式。复现不需要云服务、登录账户或隐藏数据。

软件包不包含真实的沙特运行数据、企业秘密、针对设施的攻击代码，也不包含无法核实的冲突报告。论文对原始报告的引用均为释义，不含长篇原文或敏感执行数据。

参考文献

1. Cloudflare, Matthew Prince, *Cloudflare outage on November 18, 2025*, original incident postmortem, 18 November 2025. <https://blog.cloudflare.com/18-november-2025-outage/>
2. Google Cloud, *Incident details: June 12, 2025 service disruption*, incident status and root-cause note, 2025. <https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW>
3. Federal Communications Commission (FCC), *February 22, 2024 AT&T Mobility Network Outage*, Public Safety and Homeland Security Bureau Report, 22 July 2024. <https://docs.fcc.gov/public/attachments/DOC-404150A1.pdf>
4. Saudi Press Agency (SPA), *'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 (SPA), *Elm Signs MoU with Huawei to Advance Cooperation in AI and Digital Infrastructure*, 9 July 2026. <https://www.spa.gov.sa/en/N2630981>
6. Amazon Web Services (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. Amazon Web Services (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. Amazon Web Services (AWS), *Stop conditions for AWS FIS*, online technical documentation. <https://docs.aws.amazon.com/fis/latest/userguide/stop-conditions.html>
9. Jepsen (Kyle Kingsbury / Aphyrt), *Analyses and Research Ethics*, technical analyses and methods. <https://jepsen.io/analyses> and <https://jepsen.io/analyses/ethics>

10. Principles of Chaos Engineering, *Principles of Chaos Engineering*, foundational practitioner document.
<https://principlesofchaos.org/>
11. Cloud Native Computing Foundation (CNCF), *OpenTelemetry has graduated... Now what?*, 24 July 2026.
<https://www.cncf.io/blog/2026/07/24/opentelemetry-has-graduated-now-what/>
12. Amazon Web Services (AWS), *REL12-BP04 Test resiliency using chaos engineering*, 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 Demonstration*, version 1.0. Python code and CSV/JSON outputs are included in this package; explicitly synthetic, not independently peer reviewed.

编辑说明：本研究事故资料来自公开记录，计算结果来自作者提供的完全合成演示。文中所述机构未参与本文撰写或审阅。编辑日期：2026 年 10 月 22 日。