



EXPLORATORY RESEARCH PAPER

Editorial date: 22 October 2026

---

# From Report to Executable Contract: A Conceptual Study and Computational Experiment in Exportable Software-Failure Knowledge

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

**Scientific classification:** A conceptual research paper supported by a bounded review of primary evidence and by an implemented software demonstrator running on synthetic cases and environments. **Status:** extraction of the selected sources, model design, code development, execution, and inspection of the result files are complete. **Not completed:** reliable automated extraction from real reports using a language model; testing on customer products; an empirical economic study of a Saudi company; an independent blind evaluation; or scientific peer review.

## Extended abstract

This study examines whether documented experience from software and distributed-systems incidents can become a technical asset that is executable, portable, and marketable outside its country of development. It does not assume that a country's exposure to regional pressures automatically creates an exportable industry. Instead, it separates four levels: describing an event; extracting a failure mechanism that can be generalized within stated limits; writing an

executable test with an auditable oracle; and demonstrating that foreign buyers receive value exceeding installation and maintenance costs. The review connects three documented events: Cloudflare's 18 November 2025 outage, caused by a feature file that doubled after a permissions change; the 12 June 2025 Google Cloud service disruption, caused by quota data with blank fields and a burdensome recovery retry wave; and the 22 February 2024 AT&T Mobility outage following a high-impact network misconfiguration. It then runs a synthetic study using 16 test contracts and 4 purpose-built implementations, producing 64 results. The implementations respectively agree with 7, 12, 11, and 16 reference-test outcomes. These figures follow from the design of the software and its contracts; they do not measure production-world effectiveness or the probability of preventing incidents. The study concludes that the commercial proposition requiring independent evaluation is not “using artificial intelligence (AI) to make chaos tests,” an existing practice, but measuring whether meaning and the test oracle transfer across environments, what that transfer costs, and whether buyers renew their demand.

**Keywords:** Executable Failure Contracts; Semantic Transfer; Property-Based Testing; Bounded Experiment; Inference Software; Chaos Engineering; Product Unit Economics; 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. The Scientific Question and What Its Answer Would Change

The literature on systems reliability calls for understanding and preparing for failure. It does not prove that an independent market exists for every new tool described as improving resilience. An “export industry,” as used here, is a more specific economic concept than a local engineering team's ability to solve its own problems. We seek to test the following hypothesis:

Some classes of software-failure knowledge can be extracted from documented incidents or authorized environments, represented as test contracts that are relatively vendor-independent, transferred among different customer systems at a reasonable marginal cost, and sold repeatedly to buyers outside Saudi Arabia who pay for demonstrated engineering value.

This is a compound hypothesis, and proving one part does not prove the whole. It may be technically possible to create a test without preserving its semantics across platforms. Transfer may be possible but require prohibitively expensive customization for every project. A team may be able to sell expert services internationally without creating a repeatable product-software model. The answer matters because it changes the decision about specialization and investment: should capital fund a laboratory that builds reusable adapters and specifications, specialist consulting teams, or simply the use of existing tools?

**Scientific unit of analysis:** a defined test contract, executed against an identified system under a set of conditions, producing an outcome compared with a declared oracle. **Commercial unit of analysis:** a contract or renewal by an independent foreign customer for a product during a specified sales and maintenance cycle. We do not equate the two units.

## 2. What Already Exists Before the Proposed Project?

### 2.1 Fault Experiments Are Existing Tools, Not a Presumed Saudi Discovery

The Principles of Chaos Engineering describe testing around a hypothesis of steady-state behavior, empirically grounded experimentation, and minimizing an experiment's blast radius. They do not suggest that an experiment can be made from public reports without system knowledge. Amazon Web Services Fault Injection Service (AWS FIS) has offered a scenario library since 2023, as a company guide published in 2024 describes [S06](#). Its stop conditions are tied to defined alarms; it is more than a switch for shutting down machines [S08](#). In 2025, Amazon published an engineering example that uses Large Language Models (LLMs)

to generate experiment templates from natural-language descriptions in its environment [S07](#). Any claim that merely turning text into a test template is novel or uncontested contradicts this evidence.

Since 2013, Jepsen has also published reports testing consistency in distributed databases, queues, and messaging systems. It uses real applications, generated operations, and faults such as network partitions and clock skew, then examines the output history against declared consistency models [S09](#). The possible contribution must therefore be located precisely. The product might provide a unified representation for a case library that changes as incident records are updated, semantic validation across vendors, or reproducible evidence for buyers. Each alternative faces different competitors, and no market gap can be declared before their purchasing behavior and comparative performance are examined.

## 2.2 Three Families of Failure Mechanisms in the Preliminary Evidence

The framing rests on the following records, which do not constitute a representative sample of the frequency of all technical failures:

| Incident                        | What the source documented                                                                                                                                                                                                             | Proposed abstract mechanism                                                                                       | What we do not know                                                                                  |
|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|
| 18 November 2025, Cloudflare    | A feature file doubled because a permissions change caused a query to return different results; the file exceeded its reader's accepted limit, and service fluctuated <a href="#">S01</a>                                              | Unexpected expansion of distributed intermediate output, combined with a reader that has a hidden size assumption | Is a deduplication test sufficient in other systems? No; meaning and element order may differ        |
| 12 June 2025, Google Cloud      | Incomplete quota data triggered a software exception in a globally distributed path; retries strained a backend store during recovery <a href="#">S02</a>                                                                              | Activation of an untested path by incomplete data, followed by recovery congestion                                | We do not have an executable copy of the internal system with which to reproduce the original defect |
| 22 February 2024, AT&T Mobility | A network-element configuration error interrupted nationwide connectivity; 92 million calls were blocked and more than 25,000 emergency calls did not reach their destination, according to the regulator's report <a href="#">S03</a> | Propagation of a control-element change and a large-scale reconnection pattern                                    | These facts do not specify a complete safe experiment for a different operator                       |

The first two sources are self-authored operator reports. Their publishers have direct technical knowledge but also an interest in the account of the incident; we did not compare those accounts with unpublished raw operational logs. The third is a relatively independent regulatory investigation, although some of its figures come from the operator's disclosure. These qualifications remain in the claim register.

## 2.3 Contextual Evidence of an Export Route, Not a Market for This Product

The Saudi Press Agency (SPA) reported that **28 Saudi technology and telecommunications companies** took part in the “Made in Saudi” pavilion at GITEX AFRICA in April 2025 [S04](#). It also reported the signing of a memorandum of understanding between Elm and Huawei during LEAP East in July 2026 [S05](#). These documents concern institutional presence and international activity only. They provide no independent measure of realized exports in the proposed testing sector, recurring subscriptions, a foreign customer base, or margins for this type of product. They indicate that sales channels can be sought; they are not a dependent variable that proves the thesis successful.

## 3. Defining What Is Supposed to Be Exported

We define the proposed Executable Failure Contract as a record with five necessary elements:

1. **Provenance.** The source record's link and passage, the source's degree of independence, and whether the causal relation is declared or inferred.
2. **Preconditions.** The application, storage, and protocol properties that make the experiment meaningful, together with constraints that prevent misleading application to a different system.
3. **Stimulus.** A synthetic input, an authorized change in an isolated environment, or an artificial operation pattern with defined scope, duration, and stop conditions.
4. **Test Oracle.** An observable property or functional invariant separating acceptable from unacceptable results, including how the behavior and uncertainty are observed.
5. **Evidence Package.** Software versions, execution configuration, permissions, event and test records, results, missing data, and an account of what must not be inferred.

The unit should be portable through an **Intermediate Representation (IR)**, but transfer is not a mechanical equation between field names. If a test contract assumes that removing two duplicate elements does not change a file's meaning, converting it for a service in which repetition is intentional corrupts the contract. Functional descriptions and semantic conditions are therefore required, rather than serialization alone.

The following is an illustrative form, not an approved standard:

```

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

```

Placing “Cloudflare” in the provenance field does not mean that the example erases the technical differences between Cloudflare and another system. If a user cannot establish that the target file is a set rather than a list whose order carries meaning, the contract should be rejected for that system. Abstention is better than a confident but meaningless result.

## 4. The Hardest Problem: Does Meaning Transfer, or Do Only Commands Transfer?

Portability has three distinct levels:

- **Syntactic Portability:** the file can be read and converted into a grammatically valid template for another engine.
- **Operational Portability:** the test can be executed within clear authority boundaries, with the required data recorded and without endangering a live system.
- **Semantic Portability:** the new operation tests the same property and treats acceptable and unacceptable cases consistently with the new application's function.

The first two stages may succeed while the third fails. Simulating a cloud-region outage may interrupt the test network, for example, yet fail to resemble the degradation of a unified replication policy distributed across several regions. A trace rich in timing data also cannot by itself reveal a lost or duplicated transaction unless a logical relation among concurrent operations has been specified.

To formalize the question, suppose we have two experimental systems  $A$  and  $B$ , execution traces  $\tau_A$  and  $\tau_B$ , abstract observation mappings  $\alpha_A$  and  $\alpha_B$ , and a shared property  $\varphi$ . A claim of semantic transfer requires, approximately, comparing the property's judgment on

equivalent trace representations under the applicability conditions, rather than simply observing that both tests ended without an exception. The semantic requirement can be written symbolically as:

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

This is a conceptual expression, not a proof of program equivalence or an automated decision rule. Some properties cannot be preserved outside a particular environment, including the semantics of financial transactions or legal permissions that differ in another system.

## 5. Design of the Experiment Actually Conducted

### 5.1 Limited Objective

We did not build a linguistic incident compiler, run tests against Cloudflare, Google, or AT&T, or import samples from their systems. We built a Proof-of-Concept Demonstrator to answer one question: can a small collection of behavioral invariants be recorded in a way that produces comparable outputs across independent implementations in a deterministic local environment?

The examples were written manually. They contained sixteen contracts: four cases for each of four families. Every case contained `case_id`, `family`, `description`, `stimulus`, `expected`, and `observed_property`. We then wrote four test implementations with four different sets of safeguards. The program is written in Python and uses only its standard library; it does not connect to a network. The digital record includes a SHA-256 hash of the test-case configuration so that changed inputs can be detected, without claiming external security validation.

### 5.2 Contract Families

**Policy Schema.** A valid numeric quota, a missing quota, a negative quota, and a quota that cannot be converted to a number. The proposed condition is that uninterpretable data should be deliberately rejected, rather than allowing an unhandled exception to kill the processing path. This policy does not mean every production service should close whenever a field is absent.

**Feature Bundle.** A small set, allowed repetition, repetition that puts raw size above the limit but falls below it after deduplication, and a unique set above the limit. For this synthetic example only, we defined the semantics such that order and repetition do not change the meaning, making deduplication valid. Transfer is impermissible if this property has not been established for a real system.

**Update Version.** A new authenticated update, a repeated equal authenticated version, an old authenticated version, and a new version with an invalid signature. The proposed condition is that the program must not apply a regressive or unauthenticated update, while accepting an identical repeat where its consequences are defined. We do not model real signature or key-management infrastructure; `signature_ok` is a ready-made synthetic value.

**Bounded Retry.** Immediate success, three failures followed by success on the fourth attempt, an extended failure lasting twelve attempts, and six consecutive failures. The example permits four total attempts. The oracle distinguishes valid success within the limit from stopping when the limit is exhausted. The code does not operate a network, implement real randomized backoff, or measure congestion effects on a data store.

### 5.3 Four Synthetic Implementations

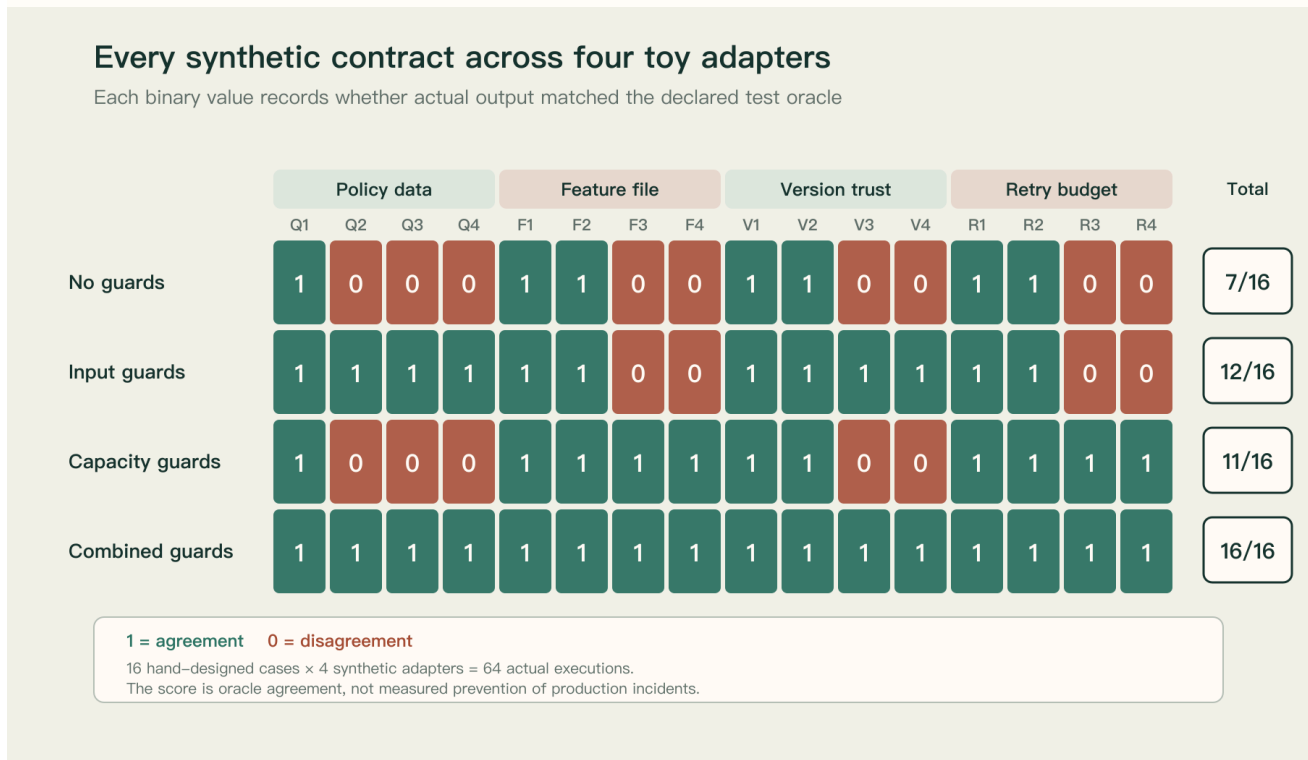
1. **Naive.** Attempts direct field conversion, uses the raw list length and an exception-based limit, accepts update versions, and retries up to an excessive count.
2. **Input Guard Only.** Adds validation of quota type and value, update version, and signature, while leaving feature-bundle and retry behavior fragile.
3. **Capacity Guard Only.** Adds feature-bundle deduplication and a retry limit, while leaving policy and version validation fragile.
4. **Combined Guards.** Combines the validations explicitly specified by the example contracts.

Each implementation was deliberately constructed to expose a limited difference. The first is not a real, poorly run company, and the fourth is not a design proven safe. **Because the tests and safeguards were constructed in the same project, the fourth implementation's success is not statistical evidence that a particular development method is superior in the world.** New cases could be devised that cause all four to fail.

### 5.4 Execution and Recorded Results

We ran `analysis/reproduce.py`, which produced `contract_cases.json`, `contract_test_results.csv`, `contract_summary.csv`, and `execution_meta.json`. The individual results total **16 contracts × 4 implementations = 64 executions**. In each row, the program recorded the expected and observed outcome and whether they agreed—not a “success in preventing an incident.”

| Synthetic implementation | Policy schema (of 4) | Feature bundle (of 4) | Versioning (of 4) | Retry (of 4) | Total (of 16) |
|--------------------------|----------------------|-----------------------|-------------------|--------------|---------------|
| Naive                    | 1                    | 2                     | 2                 | 2            | 7             |
| Input guard only         | 4                    | 2                     | 4                 | 2            | 12            |
| Capacity guard only      | 1                    | 4                     | 2                 | 4            | 11            |
| Combined guards          | 4                    | 4                     | 4                 | 4            | 16            |



Matrix of results from 64 executions on synthetic data

*Figure 1. Outputs calculated from `data/contract_summary.csv`. The score measures agreement between manually designed outputs and an oracle designed alongside them; it is not a measure of market effectiveness or real incident probabilities.*

Each partial validation path succeeded in the families it was designed to handle and remained incomplete in others. But the results are **nearly predetermined**; they carry no statistical hypothesis-test significance or external generalizability beyond this program. Their value is educational and architectural: they unify scenario language, separate the oracle from the execution mechanism, and retain the result of every contract in a form that can be compared and regenerated. A score of 16/16 cannot be interpreted as a system that does not fail.

## 6. Control, Validation, and Sensitivity: What Was and Was Not Done

**Completed:** the use of deterministic rules without randomness; export of all 64 cases; programmatic validation of the number of contracts and each implementation's total; and recording a content hash for the cases. A user can run the code under `Python 3.10+` without external dependencies and compare the files after rerunning it. This is computational reproducibility for the example, not replication of a natural event.

**Not completed:** mapping the contracts onto production systems; independent review of oracle validity; measuring the execution time of a team other than the authoring team; customer-acceptance analysis; penetration testing; evaluation of an actual language model; execution across a large number of synthetic seeds; or tests of changes in self-adapting environments. There are no confidence intervals or probability values for this example because the set is not a statistical sample. Sixty-four executions do not create a population of 64 independent incidents.

**Important sensitivities to add:** What if the semantics of deduplication change from a set to an ordered list? What if an old update is permitted under a documented rollback procedure? What if more than four retries are a business necessity in a service that does not harm its resources? What if uncaught exceptions are desirable to avoid an unsafe fail-open response? In these cases, parts of the fourth implementation may move from “correct” to “in violation” solely because the function of the system changed. This is exactly why every transfer needs independent semantic calibration.

## 7. The Gap Between a Test That Runs and a Test That Matters

Because a scenario begins with a real incident report, its result may acquire authority it does not deserve. An incident account is not a complete specification of the software that caused it. An operator may omit information to protect its system, the function of a field may differ among superficially similar tools, and an investigation may rely on partial evidence. Any future experiment must therefore distinguish four possible outcomes: an original defect represented accurately; a mechanism family represented abstractly; an educational example that only imitates the language of the mechanism; and an experiment that fails the applicability conditions and must not be run.

The current example falls into the second and third categories depending on the contract; there is no demonstrated correspondence with the historical systems. A `PASS` result can be correct for an incorrect contract. Artificial-intelligence work adds another problem: a language model may generate a syntactically valid experiment and fill gaps in the report with

a relationship the report never stated. It may invent a ratio, field name, or meaning for repetition. Evidence Grounding must therefore be separated from Compilation Validity and Oracle Validity.

## 7.1 Good Tests the Tool Itself Should Fail

The product should abstain from issuing an experiment when it does not know the function. We propose the following negative calibration cases:

- An incident account omits the necessary field definition, so the tool should request further investigation.
- A report contains a causal interpretation that its publisher later retracts, so the contract should be withdrawn or revised.
- A service processes ordered lists, so the tool should not deduplicate merely because a similar case succeeded in another application.
- An environment lacks suitable stop or recovery boundaries, so the tool should not issue an executable experiment.
- An incomplete monitoring record cannot substantiate the oracle's result, so the test should be classified as “inconclusive,” not “successful.”

Evaluating only the number of generated tests may reward a platform that speaks often and tests nothing. A measure closer to the intended meaning is the number of independent contracts that pass source, semantics, execution, and outcome review and remain valid after the customer's version changes.

## 8. Initial Engineering Architecture of the Product

An independent concept could be evaluated through seven components with clear boundaries between them:

**A. Versioned Incident Corpus.** Retains the link, date, evidence type, versions, and corrections. It does not redistribute protected content without permission; it stores provenance data, permitted excerpts, or retrieval pointers.

**B. Hypothesis Extractor.** A local or legally available language model suggests mechanisms, separates observations from inferences, and declares how incomplete the evidence is. Instructions appearing inside an imported report are not accepted as system commands.

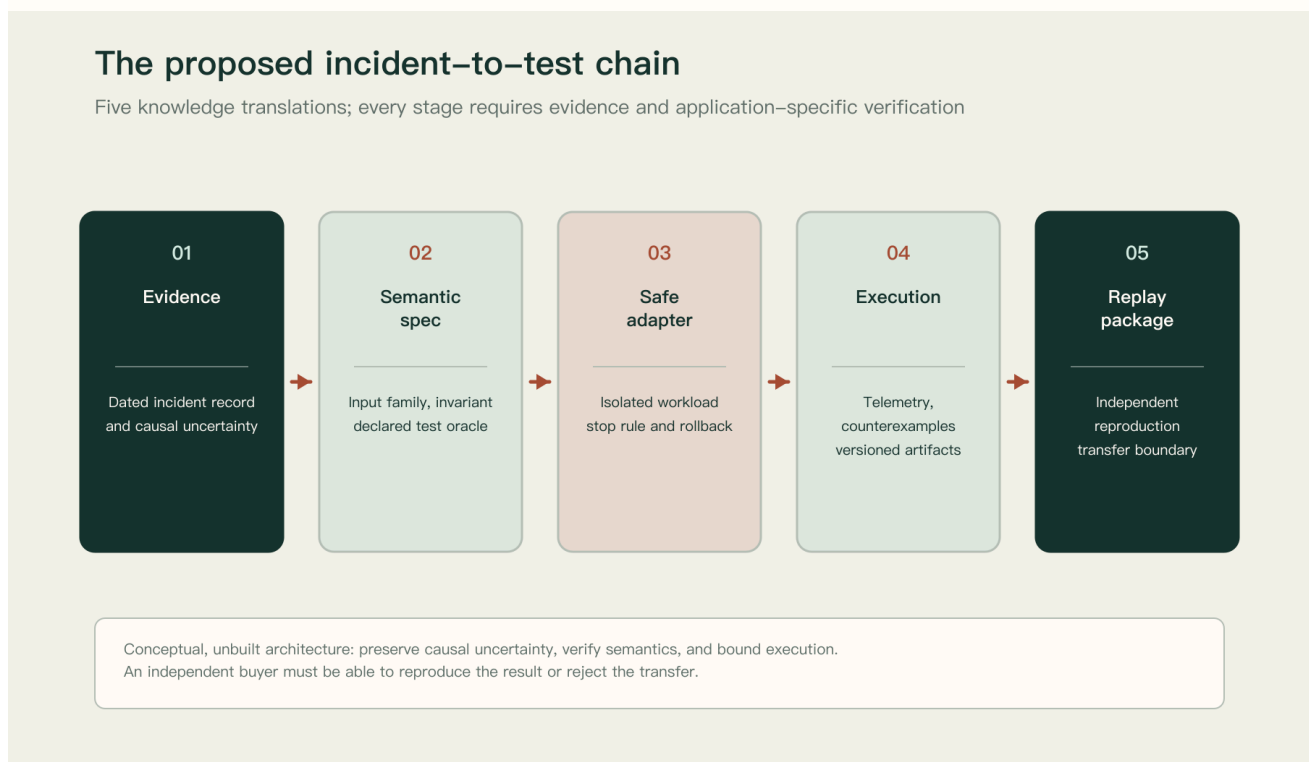
**C. Semantic Applicability Checker.** Asks about file functions, rollback rights, versioning, shared state, and whether an appropriate oracle can be defined. It may propose questions for a customer engineer.

**D. Intermediate Representation Compiler.** Converts an approved contract into case sets and bounded fault-injection programs. It separates the type of case from the execution tool and retains what it cannot implement in an unsupported field rather than discarding it.

**E. Bounded Execution Adapters.** Run first in a simulation or isolated test environment, with narrow permissions, time and load ceilings, and stop and rollback mechanisms. They do not assume these guarantees are absolute, because market tools themselves require proper configuration.

**F. Oracle and Trace Analyzer.** Compares outputs with declared properties and flags a missing trace or inconclusive behavior. Where appropriate, it uses OpenTelemetry [S11](#) without reducing transaction integrity to rates of successful requests.

**G. Evidence Replay Bundle.** States the system version, inputs, judgment, cautions, and limits. An independent buyer team can rerun the experiment in its own laboratory or reject it if its conditions differ.



Path from an incident record to a reviewable testing product

*Figure 2. A conceptual production chain. The stages do not constitute an existing Saudi system or a published standard, and every interface must be validated and constrained before use in a real environment.*

The potential competitive position is not to create every component from scratch. Adapters can be built on existing interfaces, including fault-experiment tools and open observability projects. The distinctive value that needs proof is a lower cost of transferring meaning and

clearer test evidence for different customers. Because the work of these platforms is not confined to the cloud, some purchases may require modules disconnected from customer networks or input-generation libraries for containerized software. That is an engineering choice for the purchasing case, not a design slogan.

## 9. Product Economics: From the First Experiment to the Third Customer

We define three competing business models descriptively:

1. **Bespoke Reliability Consulting.** The customer purchases expert time to inspect its system and design a custom experiment. Revenue per contract may be high, but much of the work repeats and margins depend on engineering hours.
2. **Licensed Testpack.** The customer purchases a library that it reruns and maintains with limited modification. The seller needs adaptation tools, versioning standards, and applicability criteria, or support requests will proliferate.
3. **Subscription Execution Evidence.** The customer pays to connect experiments to continuously changing releases. Subscriptions provide recurring revenue if they deliver renewable value, but create operational, storage, security, and privacy responsibilities.

To avoid declaring an unproven market, we propose the following product-unit measures for each pilot foreign customer:

| Symbol | Variable                               | Proposed measurement                                                                      |
|--------|----------------------------------------|-------------------------------------------------------------------------------------------|
| $H_i$  | Initial setup hours                    | Bounded work logs linking the effort to the customer and version                          |
| $H_r$  | Product-update hours for a new version | Upgrade effort excluding commercial work unrelated to the product                         |
| $H_c$  | Customer-specific support hours        | Intervention time that cannot be generalized to other buyers                              |
| $P$    | Price paid                             | Realized contractual revenue, not a list price or preliminary understanding               |
| $V$    | Documented benefit                     | Relevant additional cases the customer's baseline did not find, including execution costs |
| $R$    | Paid renewal                           | An actual extension after review of the first version's results                           |

The **Reusable Engineering Share** can be provisionally defined as follows:

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

If all hours are nonnegative and the denominator is positive, the share lies between zero and one. It is not gross margin and does not by itself measure product quality. A high value may indicate a strong platform, or that support customers genuinely needed was not provided. Comparisons require a consistent definition of the period, system, and buyer. This study has no measured values for the share.

A buyer could also form a hypothetical upper bound on expected value from the probability and loss of an incident, the change in its probability of avoidance due to the test, and deductions for experiment cost, price, and side effects. But the causal effect is indirect. A successful test may be followed by a comprehensive redesign. Incidents may decline because training improved, demand subsided, or architecture changed. The incident may not occur during the contract at all. We therefore infer no Return on Investment from customer stories or the number of contracts executed.

## 10. How Do We Distinguish a Saleable Product from a Service Disguised as a Platform?

The practical distinction requires a measurement dashboard published within the project, rather than marketing titles. A scalable software product should require less average customization when moving to a new customer with a similar architecture, retain a useful result after customer-product updates, and allow an engineer at the buyer to run it without the seller's developers attending every time. A specialist service may remain economically successful while these properties are weak, but that is a different venture with different economics.

We propose a market experiment beginning in civilian sectors where clear testing permission can be obtained: multi-tenant business software, nonsensitive request-processing platforms in preproduction environments, and general configuration-management components. More sensitive categories can be examined with suitable controls after success has been demonstrated. Publishing details of critical-facility simulations or connecting the product to private operational data is not a prerequisite for entering the market.

As a strict initial entry criterion—a **proposed future protocol, not a result**—consider three independent international customers on two different platforms. At least one should be able to rerun a test package without intervention from the seller's team, while setup cost, observations, and transfer failures are recorded. This is a heuristic threshold intended to encourage falsifiable testing, not a published scientific rule for classifying companies.

# 11. Designing the Study That Could Refute the Idea

## 11.1 Sample and Comparisons

An initial register would be extracted from original published incident records, with sources classified in advance by independence and detail. Training organizations should be separated from testing organizations. If the objective is to measure generalization, the training set must not include an incident from the same family as a held-out case. Incidents should also be split by publication date, so that a language model trained on a later record cannot use future knowledge in a test claiming that the information was unavailable when its contract was issued.

The first comparison is with an engineer writing a test manually using the current tool; the second, with a ready-made scenario library from a cloud provider; and the third, with natural-language experiment-template generation without additional semantic review, as shown in some market practices [S06S07](#). Teams should receive as nearly equal time as possible, and engineer skill should be recorded. Otherwise, a superior team will be mistaken for a superior tool.

## 11.2 Outcomes and Acceptance Criteria

- **Provenance Accuracy:** Is the mechanism tied to a specific passage in an original report without details being added?
- **Oracle Agreement:** Do independent experts, after learning the system's function, agree with the expected outcome? Disagreements are recorded, not removed.
- **Applicability Precision:** Of the contracts judged applicable, what proportion prove genuinely applicable when checked?
- **Transfer Yield:** What proportion of contracts retain correct semantics after transfer to a new environment within the defined sample, with the denominator and exclusions disclosed?
- **Incremental Fault Discovery:** How many additional valid defects does the product find compared with the baseline tool, after duplicates and false discoveries are excluded?
- **Marginal Integration Cost:** How many development, documentation, and execution hours do seller and customer teams spend for each new environment and version?
- **Foreign Paid Retention:** How many foreign buyers renew after an actual use cycle, and what is the net amount paid, rather than the number of proposals or memoranda of understanding?

**Falsification threshold:** If the product does not improve the discovery of functional defects in a blind test against existing tools under the same time and cost constraints; if every project requires consulting effort that does not decline; or if foreign buyers do not renew, the hypothesis of an independent export software industry weakens. Educational or consulting value may remain.

### 11.3 Causal Inference

Even if a customer's incidents decline after it acquires the package, the decline cannot be attributed to it without a comparison group or an analytical design that addresses release changes, usage intensity, and other remediation work. The earliest output most directly connected to the product is discovery of a new reproducible defect or reduced time to a trustworthy test. Preventing a future incident requires longer observation and counterfactual interpretation. Severe incidents are rare, making the benefit of reducing them statistically difficult to establish even when their economic value is high.

## 12. Authorization and Safety as Execution Properties, Not Administrative Slogans

An experiment that reveals a failure is not a scientific success if it causes a real outage outside its authorized scope. Every package should begin in an isolated or preproduction environment, require explicit scope approval if it uses injection in a live environment, set limits on duration, load, and resources, and tie stop rules to a valid indicator. AWS documents experiment stop conditions based on specified monitoring alarms [S08](#), an existing example of safety as an element of experiment design rather than an endnote.

Passwords, connection logs, and precise dependency maps for sensitive facilities should not be stored in packages intended for sale or publication. Public incident names must remain separate from customer operating data, and content rights and licences must be respected when information is extracted from published reports. Building a global taxonomy of errors does not require knowledge of an organization's secret database or publication of its internal account. Artificial-intelligence use must isolate untrusted source material from operational instructions and prevent text extraction from becoming execution authority.

## 13. What Does This Study Add to Saudi Questions Through 2035?

The study distinguishes four levels of value:

1. **Local Expertise:** Saudi teams can fix real problems. This matters, but it is not enough for export.
2. **Codified Knowledge:** Experience becomes inspectable specifications and evidence. This does not establish transferability.
3. **Transferable Product:** A test preserves approximately the same meaning for a new customer at limited cost. This does not establish sustained demand.
4. **Repeatable Export Industry:** Independent buyers pay and renew, and the product can fund its maintenance and improvement. This is an economic outcome requiring evidence different from engineering proof.

These are proposed descriptive levels, not an official index. It is wrong to infer the fourth directly from the first, or to use the participation of 28 companies in an international exhibition in place of evidence about sustained technology revenue S04.

Within this hypothesis, the most useful Saudi research investment may be a bounded laboratory rather than a comprehensive services company: a varied set of documented incidents; a protocol for encoding their conditions; open adapters for isolated experiments; blind competitions between human-written solutions and language-model proposals reviewed by engineers; and publication criteria that prevent capability claims unsupported by results. If this succeeds under independent measurement, specialist products and foreign customers can be considered. If it fails, the failure should be published, because it is itself the research material from which the proposed industry seeks to learn.

## 14. Study Limitations and the End of Its Mandate

The only completed computational result is the agreement of 64 outputs with prespecified synthetic contracts. The external review is limited to selected primary sources; it is not a comprehensive Systematic Review, and no indexed-literature screening was performed with preregistered exclusion rules. We did not estimate market size, contract price, incident probability, or actual Return on Investment. We did not establish that the conceptual model has absolute scientific novelty, would produce a patent, or is independent of existing tools.

The permissible conclusion is narrower but serious. It is technically possible to describe limited families of data-input, update, and load failures as executable contracts and produce inspectable evidence, as the small example demonstrates. Mature market tools already enable related forms of fault engineering and natural-language generation. The presumed added value must therefore be tested in **semantic transfer accuracy, lower customization cost, and repeated international payment**. Until those tests are run, the proposed industry remains a research opportunity, not news of a new Saudi economy.

## 15. Availability and Reproduction

The data in `data/` are entirely synthetic and were created for this work. They are licensed here as educational examples within the package, not as third-party data. The code is stored in `analysis/reproduce.py` and has no dependencies beyond the Python Standard Library. Run it from the research root to obtain the same files with the test-case hash. Extracted figures are provided as vector (SVG) and image (PNG) files, along with an Excel workbook for visual review and the declared calculations. Reproduction requires no cloud services, login, or hidden data.

The package contains no original Saudi operational data, company secrets, attack code for facilities, or conflict reports that could not be verified. Reports from original sources are paraphrased in this paper; it includes no long quoted passages or sensitive execution data.

## References

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 / Aphyrr), *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](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.