Provenance-First Extraction of Legacy Data: A Fidelity-Ordered, Verifiable Architecture for Migration from Obsolete Systems

Perslis Research
Extended technical report / arXiv preprint

September 2026

Abstract

A large fraction of economically and personally significant data is stranded inside obsolete software: applications whose vendors are gone, whose operating systems no longer boot on modern hardware, and whose only surviving interface is the screen. The prevailing industrial response—robotic process automation (RPA) that drives the legacy user interface and reads results off pixels, increasingly with a vision–language model in the loop—is adequate for *operating* such systems but structurally unsafe for *migrating* their data, because its dominant failure modes are silent: optical character recognition substitutes glyphs, paginated screen traversal drops or duplicates records, and no error is ever raised. We present an architecture that inverts this default. Extraction proceeds down a strict *fidelity hierarchy*: (1) direct decoding of on-disk stores from offline-mounted disk images, (2) native application export, (3) programmatic capture of the application's own control tree, and (4) pixel capture last, and then only as *witness evidence*, never as a data path. We formalize each channel as an information channel, formalize the lossless-control gate as a checkable predicate, and state the hierarchy's ordering as a set of propositions with explicit assumptions—distinguishing what is provable from what is argued. Every layer emits cryptographically signed receipts binding extracted rows to source bytes, so that a migration's completeness and integrity is a checkable property of the artifact rather than an assertion of the operator. We describe a working implementation and report a case study in which a Windows 98 system disk yielded 34,803 records across dBase, Jet, and BIFF stores with zero optical steps, validated by a 709-test suite and reconciliation against in-store record counts. Controlled benchmarks against a modeled screen-extraction baseline (fifty seeded trials) and a live-OCR baseline (a production OCR engine reading a rendered screen)—gated by a lossless-control proof and an adversarially red-teamed comparator—render the fidelity gap as measurement: the disk-direct arm scores 100% row recall on every table while real OCR silently corrupts 100% of primary keys, collapsing two relations to 0% recall. We treat the red-teaming of the measuring instrument itself, which found and fixed a gameable silent-error metric and a canonicalization flaw, as part of the method. We argue that provenance-first, disk-first extraction should be the default posture for legacy migration, and that screen-driving should be reclassified from extraction mechanism to corroborating instrument.

1 Introduction

Legacy data does not die; it becomes unreachable. Medical practices, county governments, small manufacturers, and families hold records in FoxPro directories, Access 97 databases, and spreadsheet formats last documented in the 1990s, running—if they run at all—on machines whose

replacement is the event that finally destroys the data. The migration problem is therefore not exotic: it is the mundane, ubiquitous problem of moving records out of a system that was never designed to release them, under a deadline set by failing hardware.

The tools most often applied to this problem are descendants of screen scraping: automation frameworks that operate the legacy application as a user would, paging through list views and reading fields from rendered output, increasingly with vision–language models in the loop. This approach has a seductive property—it requires no understanding of the underlying system—and a disqualifying one: **its failure modes are silent.** When an OCR stage reads a `5` as an `S`, when a row repaints mid-capture and two records blend, when a scroll reaches a false end and the final forty records are never visited, no exception is thrown. The output is a file that *looks* complete and is wrong in ways discovered only when a specific record is needed, typically after the source has been decommissioned. For interactive automation a wrong read costs a retry; for migration it becomes the permanent version of someone's data.

Operate versus migrate. This distinction is the axis of the whole design. The architecture does two jobs over one provenance spine. To *migrate* data it never touches the screen, decoding the on-disk store directly (§3). To *operate* a live legacy application it still refuses pixels, reading the application's native control tree and acting through a human-approved, receipted workflow (§4.4). Both jobs emit the same signed receipts, so a migrated row and an actuated click are each verifiable against source truth, and in both the screen is demoted to a hash-chained witness. The claims in this paper concern primarily the migration job, where errors are permanent; the operation job and its metrics appear in §4.4 and §5.

Why now. Two trends make the argument urgent rather than academic. First, the installed base of end-of-life business software is not shrinking; it is aging into a wave of forced decommissioning as the last compatible hardware fails. Second, cheap and capable vision–language models have lowered the cost of screen-driving to near zero while leaving its error profile untouched—indeed adding a new failure mode, plausible confabulation, that classical OCR lacked. The path of least resistance now produces beautiful, confidently wrong migrations at scale. If the extracted copy is destined to become the archival record, the field needs a default posture whose correctness can be *checked* rather than trusted, and a way to make the safe channel as operationally cheap as the unsafe one.

Contributions. This paper makes four contributions.

1. **A formalized fidelity hierarchy.** We define the four extraction channels as information channels, formalize the lossless-control gate as a checkable predicate over a channel, and state the hierarchy's ordering as propositions with explicit assumptions—separating an unconditional structural claim (Tier 1 has no glyph-substitution pathway) from a conditional empirical one (a bound on Tier 4 fidelity holds only under stated render conditions). The system is architecturally prevented from using a lower tier when a higher one is available (§3).

2. **Structural, not procedural, provenance.** Every extracted artifact carries signed, content-addressed receipts binding it to source bytes and to the acquisition context, so that integrity is verified by *checking the artifact*, not by trusting the pipeline. Verification is performed by a standalone, dependency-free tool that shares no code with the producing pipeline (§4.6, §5).

3. **A falsifiable assurance framework.** We define seven metrics (M1–M7) with operational measurement procedures, foreground the discriminating pair—silent-error rate and fault-detection probability—that screen-scraping baselines *cannot* score on at all, and report our honest current value on each, including the unmeasured ones (§5).

4. **A red-teamed benchmark methodology.** We contribute a benchmark whose measuring instrument is itself adversarially reviewed. It ships with a lossless-control gate proving the baseline is not a strawman, and we report the benchmark-integrity defects the review found and fixed—a gameable silent-error metric, a canonicalization flaw that hid a corruption class, and a per-process nondeterminism bug—as publishable lessons about how a benchmark can lie (§6).

We instantiate these contributions in a working system and evaluate it on a genuine obsolete machine image (§7) and in controlled benchmarks (§6). Throughout, we apply a labeling discipline that distinguishes what is deployed-and-tested from what is prototype from what is declared future work; the honesty of that labeling is, we argue, inseparable from the thesis.

Relationship to the short report. This document extends a shorter technical report of the same title. It preserves that report's pinned measured numbers verbatim—no benchmark figure has been recomputed for this edition—and adds formalization (§3), a full related-work treatment (§2), an expanded metrics framework (§5), a benchmark threat model and reproducibility protocol (§6), and a substantially harder limitations discussion (§10). Where a claim would require rerunning the benchmark to state, we mark it as future work rather than assert it.

2 Related Work

Our contribution sits at the intersection of five literatures that rarely cite one another: digital preservation and emulation, data provenance, forensic disk acquisition, document-understanding error analysis, and legacy-migration industry practice. We position against each in turn. Where we are confident of a specific work we cite it by author or system name; where we are confident only of a field or an established standard, we cite the standard rather than invent an attribution.

2.1 Digital preservation and emulation

The digital-preservation community has long recognized that bit-level survival is necessary but not sufficient: rendered meaning must survive too. The LOCKSS system ("Lots of Copies Keep Stuff Safe") [1] pioneered replication with periodic integrity auditing as the preservation primitive, and the broader field organizes around reference models for archival information packages [2]. Emulation-as-a-service [3] reframes preservation as *keeping the environment runnable* rather than migrating the artifact, so that period software can render period data on demand; large-scale software collections such as Software Heritage [4] archive source corpora with content-addressed identifiers.

Our work is complementary but pursues the opposite terminal goal. Preservation keeps the old thing runnable so a human can *read* it; migration extracts the data so the old thing can be *retired*. Emulation is, in our architecture, a means (the acquisition substrate can run an obsolete OS to reach a store or a live control tree) rather than an end. Crucially, the emulation and preservation literatures largely treat the rendered screen as the fidelity frontier—faithful rendering *is* the deliverable—whereas we treat the screen as the least trustworthy channel and insist the

authoritative bytes be decoded beneath it. The content-addressed identity that Software Heritage uses to name source objects is the same primitive we use to bind extracted rows to source bytes; we apply it downstream of extraction rather than to the source corpus itself.

2.2 Data provenance

The provenance literature supplies both a data model and a body of systems work. The W3C PROV family [5] standardizes a vocabulary of entities, activities, and agents for describing how data came to be, and provenance-aware storage systems [6] demonstrated capturing lineage at the operating- system layer so that lineage is a property of the store rather than of a cooperative application. Scientific-workflow provenance [7] records the derivation graph of computed results for reproducibility.

We adopt the PROV conceptual stance—provenance as first-class, machine-readable metadata—and specialize it in two ways relevant to migration. First, our provenance sidecar binds each extracted table not merely to a producing *activity* but to the specific *source bytes* (by digest) and the acquisition context, so a consumer can re-derive and re-check rather than only inspect the recorded lineage. Second, and more sharply, we treat provenance as an *integrity* mechanism, not only a descriptive one: a receipt whose digests re-verify is evidence about what happened, and our standalone verifier (§5) makes that evidence checkable without trusting the producer. This is closer in spirit to tamper-evident logging [8] than to descriptive lineage. We are careful (§5, M4) to separate *integrity* (digests detect corruption) from *authenticity* (a signature over the manifest attributes it to a key); we claim the former today and scope the latter as future work.

2.3 Forensic disk acquisition

The digital-forensics field established decades ago that trustworthy extraction from a storage device begins with acquiring an image under conditions that cannot alter the source. Hardware and software write-blockers, and the practice of hashing an image at acquisition and re-hashing to prove custody, are codified in guidance from standards bodies [9], and forensic tooling such as the `dd`/`dcfldd` and Sleuth Kit [10] lineage embodies image-first, read-only acquisition.

Our acquisition posture is directly descended from this tradition: we read the store *cold*, from an offline image mounted read-only, with the legacy application not merely untrusted but not running, and we record volume-level digests before any decoding begins. The novelty is not the acquisition technique—forensics has done this for decades—but the *argument* that migration, not just investigation, demands it, and the coupling of forensic- grade acquisition to a downstream, fidelity-ordered decoding pipeline that carries the acquisition digests all the way into a portable, re-verifiable package. Forensics acquires to preserve evidence for humans; we acquire to extract structured data for machines, with the same custody discipline.

2.4 OCR and document-understanding error

The document-analysis and OCR literatures have characterized recognition error for decades: confusion matrices over visually similar glyphs, the degradation of accuracy under noise, scanning artifacts, and low resolution [11], and the persistence of these errors into modern deep recognizers and document-understanding models [12]. A consistent finding is that character error rate is nonzero even on clean input and that errors concentrate on confusable pairs—exactly the digit/letter confusions (0/O, 5/S, 1/I) that dominate digit-heavy business data.

Our benchmark's modeled channel (§6) is grounded in this literature: its glyph-confusion process draws from the classic confusable-pair set, and its magnitude is deliberately conservative. But our central point about OCR is not its error *rate*; it is the error's *silence and the structural consequence of key corruption*. The recognition literature measures character and word error rates as scalar quality; we observe that when the corrupted character falls in a primary or join key, the failure is not a scatter of wrong cells but the collapse of a relation—a qualitatively worse and entirely unflagged outcome that a character-error-rate figure hides. Our live-OCR arm demonstrates this on a clean, generous render (§6).

2.5 Legacy migration and program understanding

Industrial legacy migration spans ETL tooling, RPA platforms, and a growing class of VLM-driven GUI agents; the research adjacent to it includes program comprehension and binary/source translation. Sponsored research has repeatedly targeted legacy software specifically: programs aimed at enhancing legacy software behavior, at assured micropatching of binaries, and at automated translation of legacy C to memory-safe Rust are active or recent [13]. This body of work overwhelmingly addresses *code*: how to understand, patch, or translate the program.

We address the orthogonal and comparatively neglected problem of the *data*: getting the records out, once, correctly, with checkable completeness, when the program may be un-instrumentable and about to disappear. Where GUI-agent research asks how to make an agent drive an interface reliably, we argue that for the migration task the interface should be bypassed entirely, and that when an interface *must* be driven (the operate job), the safety property is structural—propose/actuate separation with per-step receipts against a signed control-tree catalog—rather than a matter of agent accuracy. This aligns with an independent development in platform governance: mobile-platform developer policy now explicitly distinguishes deterministic, rule-based automation following a human-defined script (permitted) from autonomous initiation, planning, and execution of UI actions (prohibited) [14].

Summary of positioning. Preservation keeps the old system runnable; we retire it. Provenance describes lineage; we make lineage re-checkable and tamper-evident. Forensics acquires images for investigation; we acquire them for data migration and carry custody into the output. OCR research measures recognition error as a rate; we treat its silence and its relation-collapsing key-corruption mode as the decisive property. Legacy-software research repairs code; we rescue data. To our knowledge no prior system combines an architecturally enforced fidelity hierarchy, structural re-verifiable provenance, and a red-teamed fidelity benchmark for the migration task.

3 The Fidelity Hierarchy, Formalized

We order extraction channels by the number and severity of lossy transformations between the authoritative record and the extracted value, and we make the ordering precise enough to state which claims are provable and which are empirical. This section is deliberately conservative about epistemic status: we mark each formal statement as a *theorem* (holds by construction under its stated assumptions), a *proposition* (argued rigorously but resting on an idealization), or a *conjecture* (believed, not proven here).

3.1 Extraction as an information channel

Definition 1 (Datum, record, store)**.** Let a *store* S be the byte content of an authoritative on-disk data structure (a DBF file, a Jet database, a BIFF workbook). S determines a finite multiset of *records* $R(S)$, each an ordered tuple of *cells* drawn from a value domain $\mathcal{V}$. $R(S)$ is defined by the format specification alone: given S and the spec, $R(S)$ is a total function of the bytes.

Definition 2 (Extraction channel)**.** An *extraction channel* is a (possibly randomized) map $C : S \mapsto (\hat{R}, F)$ where $\hat{R}$ is the multiset of extracted rows over a canonical value domain and F is a set of *flags* the channel raises about its own output. We write $C(S) = (\hat{R}(S), F(S))$. A channel is *silent on an error* if it emits a wrong cell (a cell differing from the canonicalization of the corresponding truth cell) that no flag in F names.

Canonicalization is a fixed, channel-independent normalizer κ (blank, boolean, integer, trimmed-decimal, ISO-date, and padding-preserving rules; §6 details it and its adversarial hardening) applied to both truth and every channel's output, so that "the same datum via a different channel" compares equal and any divergence is a genuine extraction difference rather than a formatting artifact.

Definition 3 (Fidelity)**.** Fix a store S with truth multiset $R = \kappa(R(S))$. For a channel output $\hat{R} = \kappa(\hat{R}(S))$, define *row recall* $\rho(C, S) = |R \cap \hat{R}|/|R|$ (multiset intersection; a row with even one wrong cell is not in the intersection) and *silent-error count* $\sigma(C, S)$ as the number of wrong cells over key-matched rows that no flag in $F(S)$ names. A channel is *lossless on* S if $\rho(C, S) = 1$ and there are no spurious rows.

The pairing of ρ and σ is deliberate and is the discriminating instrument of the whole paper. A channel can be forced toward high ρ by effort; *no amount of effort lets a channel that cannot observe its own errors report a truthful* $\sigma > 0$. That asymmetry, not the raw error rate, is what the hierarchy exploits.

3.2 The four channels

We instantiate four channels, corresponding to the four tiers.

Tier	Channel	Error character	Dominant failure mode
1	On-disk store, decoded offline	Detectable, fail-fast	Unknown/corrupt format regions
2	Native export (report-to-file)	Detectable, bounded	Export truncation, locale formatting
3	Control-tree / text-buffer read	Detectable, bounded	Widget virtualization, lazy population
4	Pixel capture + OCR	**Silent, unbounded**	Glyph substitution, torn frames, missed pages

Table 1: Extraction tiers ordered by fidelity. Only Tier 4 has failure modes that are both silent and unbounded; it is therefore admissible only as evidence, or for residual fields with independent reconciliation.

Definition 4 (Tier-1 channel, C_1)**.** C_1 applies the format decoder directly to S. On a well-formed region it returns $\kappa(R(S))$ exactly; on a malformed region it raises a typed error naming the region (a flag) rather than emitting a value.

Definition 5 (Tier-4 channel, C_4)**.** C_4 renders $R(S)$ to a raster display and recovers cells by optical recognition and positional binning. It emits no flags: an OCR pipeline has no channel by which to learn that a glyph was misrecognized.

Tiers 2 and 3 (native export, control-tree read) sit between: they read text through a documented interface rather than an image, so glyph substitution is impossible, but they inherit bounded, *detectable* failure modes (export truncation, widget virtualization) that surface as short reads or explicit gaps rather than plausible wrong values.

3.3 The lossless-control gate

Before any comparison across channels is meaningful, we must exclude the possibility that the benchmark has rigged the baseline—for instance, by letting the modeled screen channel lose data even at its most benign setting. We formalize the guard as a predicate over a parameterized channel.

Definition 6 (Parameterized channel and control point)**.** Let C_θ be a channel parameterized by a fault vector $\theta \in [0,1]^k$ (for the modeled screen channel, $\theta = (p_{\text{glyph}}, p_{\text{tear}}, p_{\text{repeat}}, p_{\text{stop}})$). The *control point* is $\theta = \mathbf{0}$: all fault processes disabled.

Definition 7 (Lossless-control gate)**.** The gate is the predicate

$$\textsc{Gate}(C) \;\equiv\; \forall S \in \mathcal{S} :\; \rho(C_{\mathbf{0}}, S) = 1 \;\wedge\; \sigma(C_{\mathbf{0}}, S) = 0 \;\wedge\; (\text{no spurious rows}),$$

evaluated over the benchmark's store set $\mathcal{S}$. A harness that reports any cross-channel number without first establishing $\textsc{Gate}(C)$ is *unsound*: a deficit it attributes to the baseline's faults may instead be an artifact of a lossy render.

In our implementation the gate is not a documentation promise but a runtime precondition: the harness computes $C_{\mathbf{0}}$ first and refuses to emit any results if the predicate fails (returning a nonzero exit and the offending table). This makes the soundness condition a checkable property of every run, not a claim about the authors' care.

Proposition 1 (Gate soundness of attributed deficit)**.** *If $\textsc{Gate}(C)$ holds and the render step of C_θ is independent of θ (the same faithful render feeds every fault setting), then for any θ the deficit $1 - \rho(C_\theta, S)$ is attributable to the fault processes parameterized by θ alone, not to the render.*

Proof sketch. By construction $C_\theta = A_\theta \circ \text{render}$ where the render is θ-independent and A_θ is the fault channel. At $\theta = \mathbf{0}$, GATE forces $A_{\mathbf{0}}$ to be the identity on the rendered rows (lossless, no spurious rows), so the render itself introduces no loss. Therefore any $\rho < 1$ at $\theta \neq \mathbf{0}$ arises within A_θ, whose only mechanisms are the parameterized faults. □

Proposition 1 is what licenses reading the modeled arm's deficit as "the cost of the four documented fault processes" rather than "an unknown mix of render loss and faults." It is a statement about the *benchmark*, not about real screens; the real-OCR arm (§6) exists precisely because the fault rates in θ are chosen.

3.4 Ordering claims

We now state the hierarchy's ordering. We separate an unconditional structural claim from a conditional empirical one, because they have very different epistemic status and conflating them would be exactly the overclaim this paper warns against.

Assumption 1 (Faithful spec and cold read)**.** The Tier-1 decoder implements the format specification correctly on well-formed regions, and the store S is read from an offline image with the legacy application not running, so S is not mutated during acquisition.

Theorem 1 (Structural absence of glyph substitution in Tier 1)**.** *Under Assumption 1, for any cell whose stored representation is a digit sequence, C_1 cannot substitute a visually-confusable glyph (e.g. emit `S` for a stored `5`). Equivalently, the glyph-confusion fault process of C_4 has no analogue in C_1.*

Proof. C_1 maps stored bytes to values through the format's field-decoding function, which reads the byte value of each cell (numeric fields as packed digits or binary, text fields as codepage-decoded bytes). No step of this function consults a rendered glyph or a visual similarity metric; the byte `0x35` ('5') is decoded by its numeric or codepage meaning, and there is no code path by which its rasterized shape could influence the result. Hence the confusion map $5 \mapsto S$, which requires interpreting a *shape*, cannot occur. The argument is structural, not statistical: it does not bound a probability, it removes the mechanism. □

Theorem 1 is the paper's hardest claim and, we believe, its most important: it is not that Tier 1 has a lower glyph-error *rate* than Tier 4, but that the glyph-error *class* does not exist in Tier 1. The same argument generalizes to torn-frame and missed-page faults, which are properties of paginated visual traversal and have no counterpart in decoding a byte structure whose record count is read from its own header.

Proposition 2 (Detectability ordering)**.** *For C_1, C_2, C_3, every failure mode is detectable: it manifests as a typed error, a short read against a self-described count, or an explicit gap, so the channel can raise a flag naming it. For C_4, the dominant failure modes (glyph substitution, torn frames, missed pages) are silent: the channel completes without any signal distinguishing a correct read from a corrupted one. Consequently $\sigma(C_i, \cdot)$ can be driven to 0 by construction for $i \in \{1, 2, 3\}$ (raise the flag), while $\sigma(C_4, \cdot)$ cannot.*

Epistemic status. Proposition 2 rests on an idealization—that Tiers 1–3 read through interfaces that expose completeness (record counts, short-read signals). This holds for the formats we implement (which carry their own counts) and for control-tree reads (which enumerate a finite tree), but a store format that lied about its own record count, or a UI that reported a virtualized list as complete, would weaken it. We therefore state it as a proposition, not a theorem, and we reconcile against the store's self-description precisely to catch the case where the count itself is suspect (§4.6).

Conjecture 1 (Fidelity dominance under realistic rendering)**.** *For a broad class of real legacy displays (fixed-width list views under CRT blur, scaling, and mid-scroll repaint), a Tier-4 channel exhibits both $\rho(C_4, \cdot) < 1$ and $\sigma(C_4, \cdot) > 0$ on digit-heavy business data, whereas a correct Tier-1 channel achieves $\rho = 1$ and $\sigma = 0$ on the same data. That is, Tier 1 dominates Tier 4 on both fidelity axes simultaneously, not merely on average.*

We label this a *conjecture* rather than a proposition because it quantifies over "real legacy displays," a population we have not sampled; our evidence is one modeled channel (with chosen rates) and one live-OCR arm on a single, deliberately generous render (§6), plus Theorem 1 for the structural half. A live, public benchmark over diverse real guests (§11) is what would elevate it toward a proposition. Stating it honestly as unproven is itself an instance of the labeling discipline the paper advocates.

3.5 Why the ordering must be enforced, not advised

Two observations turn the hierarchy from a guideline into an architectural constraint. First, Tier 1 is available far more often than practice suggests: even ancient applications sit on *some* readable store—a dBase/FoxPro directory, a Jet (`.mdb`) file, Btrieve or Paradox tables, fixed-width flat files—and a disk image taken from a powered-off machine exposes all of them at full fidelity. The tier is skipped not because it is unavailable but because it demands format literacy. Second, tier degradation is *sticky*: once a pipeline is built around screen traversal, its silent error profile (Proposition 2) is inherited by every downstream consumer, and no amount of retry logic converts a silent error into a detectable one—retry re-runs a channel that still cannot see its own mistakes. The hierarchy must therefore be enforced at design time: the system is constructed so that migration data flows only through Tiers 1–3, and Tier 4 flows only into the evidence record (§4.5).

4 Architecture

The implementation comprises components with disjoint responsibilities, composed so that migration data flows only through Tiers 1–3 while Tier 4 flows only into the evidence record. We describe each layer by its *interface and guarantees*. Two subsystems—the acquisition substrate and the receipt scheme—are described abstractly by the properties they provide rather than by their internals; those internals are outside the scope of this paper (see §10 on the resulting tension with reproducibility).

4.1 The acquisition layer (abstract)

Migration begins with a *verified acquisition subsystem* that produces a byte-level image of an obsolete guest's storage together with a cryptographically signed provenance receipt. We describe it by contract, not construction:

- **Cold, offline read.** The authoritative store is read from an image with the legacy application not running—not merely untrusted but inert—so acquisition cannot mutate the source (Assumption 1). This is the forensic image-first discipline (§2) applied to migration.
- **Acquisition-time integrity.** Volume-level digests are recorded *before* any decoding begins, so the bytes the decoder sees are bound to the bytes that were acquired.
- **Signed provenance receipt.** The acquisition emits a receipt recording method, timestamp, source identity, and image digest, sealed so that the record is tamper-evident: an alteration is detectable by re-checking digests. The receipt is *testimony made tamper-evident*, not a proof that the acquisition method was honest; the independent evidence for that lives in the referenced run artifacts.

The design principle is that **acquisition is a bytes problem, not a UI problem**: no component above the acquisition layer needs the machine to be interactive. How images are produced, how guest lifecycles are managed, and how receipts are signed and stored are deliberately out of scope here; the migration pipeline consumes only the image and the receipt through a fixed interface.

4.2 The decoding layer

The decoding layer converts acquired bytes into relational data by parsing native formats directly: dBase/FoxPro `.dbf`, Jet 3/4 (`.mdb`), BIFF5/BIFF8 (`.xls`), plus delimited, INI, registry, and fixed-text sources, emitting SQLite, CSV, and JSON alongside an analyst report. Two properties are load-bearing.

Structural immunity to optical error. Because decoding operates on record structures—field descriptors, data pages, row offsets—rather than renderings, the classic OCR confusions are not merely unlikely but impossible (Theorem 1): a stored `5` has no pathway to become an `S`.

Fail-fast on malformed regions. Decoding is *input-lenient, output-strict*. Dirty legacy data becomes explicit quality flags in the manifest—a memo field that will not resolve, an out-of-range date, a precision-risk decimal preserved as exact text rather than silently rounded—and a genuinely malformed region produces a typed, localized error rather than a plausible value. The pipeline is bounded throughout (byte budgets, cell caps, cycle guards on every page and pointer walk), so a hostile or corrupt store degrades to a named failure, never a crash or an invented row. Every malformed input in a 36,000-iteration mutation-fuzz over all six format parsers produced a clean, typed refusal with zero uncaught crashes (§5, M3).

Grounded question-answering. On top of the decoded store, the layer compiles natural-language questions to read-only SQL over the extracted tables and returns answers with the supporting rows. The SQL path is hard-constrained (read-only connection, query-only mode, a step budget, row and byte caps), and all extracted content is fenced as inert, untrusted data in any model prompt. This closes the loop for a non-technical data owner without ever introducing a generative step between the store and the answer.

4.3 Control-tree capture for UI-resident residue

Some data genuinely exists only behind the running application: computed fields, values in proprietary blobs the application alone can interpret. For this residue we still refuse the pixel path. A control-tree capture component enumerates the guest's native window and control tree and emits normalized records—role, text, geometry, state—per control. Text is obtained from the windowing interface, not from an image of the text; this is the GUI generalization of reading a terminal's text buffer rather than photographing the terminal (a Tier-3 channel in the sense of §3). Controls that expose no text (owner-drawn surfaces) are explicitly marked *opaque* and demoted, never silently OCR'd into the data stream. The binary details of how the control tree is enumerated and transported are out of scope; the relevant property is that the captured text comes through a documented interface, giving it Tier-3 detectability rather than Tier-4 silence.

4.4 The operating surface

The same control-tree capture that recovers UI-resident data also powers the system's second job: operating a live legacy application. Each captured control is normalized into a structured card—role, text, geometry, state, and provenance tier—and an operating console renders the live catalog as an interactive rail beside the guest frame. Actuation is deliberately not autonomous: the AI *proposes* a step, a deterministic human-approved workflow *actuates* it one card at a time, and each actuation is receipted against the signed catalog of the guest's real control tree. The safety argument is identical to the migration job's—an action addresses a structured control by identity, never a pixel coordinate guessed from an image—so "operate" inherits the same accountability "migrate" enjoys. §5 states the metrics (M6–M7) that make this job falsifiable. A single pixel-derived view survives, the same one as in migration: a hash-chained witness (§4.5), never an action input.

Maturity, stated precisely. We separate two claims that are easy to conflate. Control-tree *capture* has been exercised against live guests from an NT-era Windows through modern Windows, enumerating full application control trees headlessly. The receipted *actuation loop* is validated at the workflow and receipt level—approved workflows, one-step actuation, per-step receipts—and demonstrated at pilot scale, but has not been run end-to-end against a live guest at pilot scale over a sustained soak. We report the former as demonstrated and the latter as pilot/future work (§5).

4.5 The screen as witness

The screen is not discarded; it is reassigned. A witness service maintains a recycling ring of periodic display captures, each stamped with a SHA-256 digest at acquisition, forming an evidence trail that the machine state during extraction matched the extracted claims: which application was open, which dataset was on screen, that no error dialog was present. The essential property is directional: **frames corroborate; they never populate.** A capture failure degrades the evidence record, not the data—validated under fault injection, where source outages degrade evidence capture without corrupting or halting extraction. This directional separation—an independent, hash-chained record of what the world looked like while the system acted, structurally prevented from becoming an input—is, we argue (§9), the most portable idea in the design.

4.6 Provenance and the Migration Package

Every layer emits signed artifacts: signed catalogs whose wire bytes are the signed bytes, provenance tiers attached per element (interface-derived vs. pixel-derived vs. synthesized), and per-frame acquisition digests. The terminal artifact of a migration is a *Migration Package*: portable DDL, open-format data files (CSV/JSON/SQLite today; Parquet/JSONL planned), a foreign-key graph, and a provenance sidecar tracing every table to source bytes and acquisition context. The DDL and foreign-key graph are assembled by a dedicated step; the graph is *evidence-backed*, not name-matched—a relationship is emitted only when the child's values are a proven subset of a unique parent key, so the graph records referential integrity that actually holds in the extracted data (on the case data, `invoice`→`customer` and `items`→`invoice`).

Verification is a property of the package. Any party holding the package can recheck digests with a published standalone tool that shares no code with the producing pipeline (stdlib-only, no

pipeline imports), so verification does not require trusting—or even possessing—the pipeline that produced the package. Its tamper-detection is tested against a flipped byte, a doctored manifest, and a path-traversal entry, all of which it rejects. The package binds to open formats, not to any destination vendor: it is deliberately cloud-neutral.

Integrity versus authenticity. We are precise about what the digests buy. They establish *integrity*: a consumer detects corruption or tampering against the recorded digests. They do not by themselves establish *authenticity*—a signature over the manifest binding it to a producing key—which requires asymmetric cryptography and a key-distribution channel and is scoped as future work (§5, M4; §11). We claim the former today and decline to claim the latter.

Reconciliation: escalate rather than guess. Because Tier-1 stores carry their own record counts (dBase headers, Jet system tables), extraction completeness is checked against the source's self-description: emitted row counts must equal in-store counts, mismatches are flagged rather than accepted, and residual Tier-3/4 fields are sampled and independently re-read. The governing rule is that any value that cannot be tied to a source-of-truth quantity is surfaced as unresolved, never defaulted. This is the operational answer to the idealization behind Proposition 2: rather than trust that a store's self-described count is honest, we make the count-vs.-output comparison a hard gate and escalate any discrepancy.

5 An Assurance Framework: Falsifiable Metrics for Legacy Control and Migration

A system that proposes to let AI agents operate and drain legacy systems must expect scrutiny of one question above all: **when it is wrong, does it know?** We answer with a metric set in which each metric is falsifiable and measurable by an independent evaluator holding only the artifacts. This section gives each metric an operational definition, a measurement procedure, and an honest current value—including the unmeasured ones. A metrics table with no gaps is a metrics table nobody tested.

5.1 Operational definitions and measurement procedures

M1 — Record recall. *Definition:* exactly recovered records divided by the source-of-truth count, where truth is the store's own self-described record count (dBase header, Jet system table), not a re-read by any extraction arm. *Procedure:* decode the store; compare emitted row count per table against the in-store count; a mismatch is a hard failure, not a warning. *Value:* 100% on the benchmark (Tab. 3) and on the 34,803-row case study (§7), reconciled against in-store metadata; fleet-scale unmeasured.

M2 — Silent-error rate. *Definition:* wrong values emitted with no flag raised, per 10^6 cells (the σ of §3, normalized). *Procedure:* over key-matched rows, count cells differing from canonicalized truth that no arm-reported flag names; a flag "covers" a wrong cell only if it names that cell's column as a whole token. *Value:* 0 measured on the disk path; the metric *itself* survived adversarial review (§6) after an initial gameable definition was corrected.

M3 — Fault detection. *Definition:* P(flagged or refused | injected fault) under adversarial corruption of inputs. *Procedure:* inject structural faults (truncated headers, corrupt pages) and measure the fraction that produce a flag or typed refusal rather than a plausible value; separately, mutation-fuzz every parser. *Value:* 100% on structural fault injection; across 36,000+ mutation-fuzz iterations over all six format parsers (seeded with real DBF/Jet/ BIFF files) every malformed input produced a clean, typed refusal with zero uncaught crashes, now pinned by a bounded regression fuzz.

M4 — Provenance verifiability. *Definition:* the fraction of emitted artifacts an independent party can re-verify from digests and signatures alone, split into *integrity* and *authenticity*. *Procedure:* run the standalone verifier (no pipeline dependency) against the package; attempt tampering. *Value:* artifact *integrity* re-verified 100% by a published standalone verifier (stdlib-only, no pipeline imports; tamper-detection tested—flipped byte, doctored manifest, path traversal all rejected). *Authenticity*—a signature over the manifest binding it to a key—remains the separate signed-channel work (§11).

M5 — Reproducibility. *Definition:* byte-identical outputs for identical inputs across independent processes and hosts. *Procedure:* run the same extraction twice and diff outputs; cross-process today, cross-host as a program deliverable. *Value:* cross-process reproducibility is pinned by a regression test (see the per-process nondeterminism bug in §6 for why this test exists); cross-host is unmeasured.

M6 — Actuation accountability. *Definition:* UI actuations that are single-step, pre-approved, and receipted, divided by all actuations. *Procedure:* inspect receipts for one-step semantics and prior approval against the signed catalog. *Value:* enforced by construction and demonstrated in pilot (approved workflows, one-step, per-step receipts); fleet/long-duration soak unmeasured.

M7 — Reversibility. *Definition:* mutating operations preceded by a dry run and covered by a tested rollback path. *Procedure:* verify a dry-run gate precedes every mutation and that a rollback restores prior state. *Value:* dry-run gate is contractual; rollback is tested for the install path only.

5.2 The discriminating pair, and why it resists gaming

M2 and M3 are the discriminating pair. A screen-scraping baseline can in principle score well on M1 on a good day—a clean render, a lucky seed—but it has no mechanism by which to score on M2/M3 at all: the scraper cannot flag what it cannot detect (Proposition 2). This is why the honest comparison is not "which arm is more accurate on average" but "which arm can even represent its own uncertainty."

Precisely because M2 is the headline metric, it is the one most vulnerable to being gamed, and the adversarial review of our benchmark (§6) found exactly that. We describe the failure class abstractly here because it generalizes beyond our harness. A silent-error metric of the form "wrong cells not covered by a flag" is only as strong as its notion of *coverage*. Two coverage bugs make it gameable:

1. **Any-flag-zeroes-everything.** If *any* flag, however unrelated, suppresses the silent count, then a channel can score a perfect M2 by emitting a single boilerplate flag ("some records were skipped") that names nothing. The fix requires a flag to name the specific column of the wrong cell it claims to cover; unrelated flags earn no credit.

Metric	Definition	Status today (pilot scale)
M1 Record recall	Exactly recovered records / source-of-truth count, reconciled against the store's own metadata	100% on benchmark and on the 34,803-row case study; fleet-scale unmeasured
M2 Silent-error rate	Wrong values emitted with no flag raised, per 10^6 cells	0 measured on the disk path; the metric itself survived adversarial review (§6)
M3 Fault detection	P(flagged | fault) under input corruption	100% on structural fault injection; 36,000+ mutation-fuzz iterations all produced typed refusals, zero uncaught crashes
M4 Provenance verifiability	Fraction re-verifiable from digests + signatures by an independent party	Integrity: 100% via a stdlib-only standalone verifier (tamper-detection tested). Authenticity (signature over the manifest): future work
M5 Reproducibility	Byte-identical outputs across independent processes/hosts	Cross-process: pinned by test. Cross-host: unmeasured
M6 Actuation accountability	Single-step, pre-approved, receipted actuations / all	Enforced by construction, demonstrated in pilot; fleet/soak unmeasured
M7 Reversibility	Mutations preceded by dry run + tested rollback	Dry-run gate contractual; rollback tested for install path only

Table 2: Assurance metrics for AI-driven legacy control and migration. "Unmeasured" entries are declared rather than extrapolated.

2. **Substring collusion.** If coverage is decided by substring match, a flag naming column `FIRSTNAME` falsely credits awareness of a wrong `NAME` cell. The fix matches whole identifier tokens, not substrings.

Both are instances of a general lesson: *a gaming-resistant metric must require the claimed awareness to be specific enough that it could not have been produced by a channel that did not actually detect the error.* A flag that any run could emit is not evidence of detection. We consider this metric-hardening part of the contribution, not an embarrassing footnote: the discriminating metric is only meaningful if it cannot be satisfied by an undetecting channel pretending to detect.

5.3 Readiness assessment

In technology-readiness terms we assess the decoding and provenance layers as laboratory-validated on real artifacts (TRL 4–5), the control-tree *capture* element as demonstrated on live guests (TRL 5), the receipted actuation loop at TRL 4 (pilot/lab-validated workflow, not yet operationally soaked), and fleet-scale operation, cross-host reproducibility, and long-duration soak as named, unstarted work. On the control side the architecture's posture—the AI *proposes* and a deterministic, human-approved, content-addressed workflow *actuates*, one step at a time, each step receipted against a signed catalog—is not a limitation grudgingly accepted; it is what makes M6 measurable at all, and it aligns with where platform governance is independently heading, distinguishing deterministic human-authorized automation from autonomous UI agents (§2).

6 Controlled Fidelity Benchmark

To quantify the hierarchy's claim rather than assert it, we built a benchmark harness that runs both extraction postures against ground truth neither can influence. This section gives the design, then a *threat model for the benchmark itself*—the ways a fidelity benchmark can lie—and finally the adversarial review that found and fixed three such lies in our own harness. All figures below are carried over from the underlying report and its results artifact; no number was recomputed for this edition.

6.1 Design

Truth rows derive from the generator functions of a deterministic synthetic legacy application (a 1995-vintage dBase III billing package: 9 customers, 25 invoices, 37 line items)—the same pure functions that pack its genuine DBF bytes, so neither arm grades its own homework. Deleted records are excluded: a correct disk parse skips them and the legacy UI never shows them, so they belong to neither arm's target set.

- **Disk-direct arm (C_1).** The production parser on the actual fixture DBF bytes, unmodified—it imports and runs the same parsing code production extraction uses. A parse exception is a *detected* total failure, reported as such.
- **Modeled screen arm ($C_4^{\mathbf{model}}$).** A faithful fixed-width render of the application's list views passed through a parameterized fault channel with four processes observed in real scrape pipelines: per-character OCR glyph confusion (p=0.002), torn-row repaints (p=0.02/page), repeated pages (p=0.01/transition), and early scroll termination (p=0.005/transition), then scraped back by column offsets. Fifty seeded trials.
- **Real-OCR arm ($C_4^{\mathbf{real}}$).** The same fixture rendered with a real monospace font and read back with a production OCR engine, binned into columns by pixel position. Nothing is invented—whatever the engine misreads is what an OCR pipeline actually misreads. One deterministic run.

Three design rules keep the comparison honest, corresponding directly to the formalization of §3. First, the modeled baseline is a *model*, labeled as such everywhere its numbers appear; it is not a live measurement of any product. Second, the *lossless-control gate* (the predicate Gate of §3) is enforced at runtime: with all fault rates at zero the channel must score 100% on every table, and the harness refuses to emit results otherwise (Proposition 1). Third, row recall is strict and disclosed as such: a row with one wrong cell counts both as an unrecovered row and as a wrong cell—two views of the same damage, not independent events.

6.2 Results

The real-OCR arm is more damning than the model, not less. On a clean, generous render—bold monospace, high resolution, no CRT blur—OCR silently substitutes the letter `O` for the digit `0` across the `INV####` invoice identifiers, corrupting 100% of the primary keys. Because those are the join keys, the failure is not a scatter of wrong cells but the collapse of the entire relation: 0% row recall on the invoice and line-item tables. The silent-wrong-cell count reads 0 there not because the read was clean but because no row survived key-matching to have its cells compared—the

Table	Arm	Row recall	Cell acc.	Silent wrong cells
customer	disk-direct	100.0%	100.0%	0
	screen (modeled)	93.8% (77.8–100.0)	99.0% (96.3–100.0)	0.5 (max 2)
	screen (real OCR)	88.9%	100.0%	0
invoice	disk-direct	100.0%	100.0%	0
	screen (modeled)	95.8% (68.0–100.0)	99.5% (96.0–100.0)	0.7 (max 5)
	screen (real OCR)	0.0%	0.0%	0
items	disk-direct	100.0%	100.0%	0
	screen (modeled)	97.7% (91.9–100.0)	99.8% (99.3–100.0)	0.2 (max 1)
	screen (real OCR)	0.0%	0.0%	0

Table 3: Three arms against the same ground truth. The modeled screen channel (fifty seeded trials) is not a live VM measurement and its ranges are min–max across trials; the real-OCR arm is a single deterministic engine read of an actual render. "Silent wrong cells" are cells differing from truth with no flag raised. Both screen arms are structurally incapable of raising flags; the disk-direct arm's errors on deliberately corrupted input arrive as named, loud failures instead of rows. The real-OCR arm's 0% invoice/line-item recall is primary-key corruption, not cell noise.

corruption manifests as total row loss, the worst outcome, not as scored wrong cells. The disk-direct arm scores 100% on the identical data. Real legacy screens are worse than this render, so this arm is a *lower bound* on the damage, not an upper one—the empirical half of the evidence for Conjecture 1.

The distribution matters more than the means. The modeled arm's worst trial (seed 5) recovered 68.0% of invoices—17 of 25—and 8 rows did not survive intact: an early scroll stop dropped 7 and a corrupted read cost the 8th, while the output remained well-formed. The disk arm's behavior under injected corruption (truncated header bytes) is a named parse error and zero rows. That asymmetry—wrong-and-unaware versus loud-and-refusing—is the thesis rendered as a measurement.

6.3 A threat model for the benchmark itself

Benchmarks that flatter their authors are a known failure mode. We enumerate the ways a fidelity benchmark can lie, because the value of our numbers is conditional on having closed each one, and because the taxonomy is reusable.

1. **Grading one's own homework.** If truth is derived from an extraction arm rather than independently, the arm scores against itself. We derive truth from the fixture's generator functions, imported directly, not from either arm's output.

2. **A rigged or lossy baseline.** If the baseline loses data even at its most benign setting, its deficit is not attributable to the phenomenon under test. The lossless-control gate (Gate) is a runtime precondition; a run that fails it emits no results.

3. **Seeding nondeterminism.** A benchmark claiming reproducibility must be reproducible across processes, not merely within one. Our fault generator was originally seeded via a hash of a string-bearing tuple, which the Python runtime randomizes per process (`PYTHONHASHSEED`): in-process determinism tests passed while independent runs diverged. This is *exactly* the class

of silent, environment-dependent error this paper exists to warn about—caught only by a cross-process reproducibility check, now a pinned test, and the reason M5 (cross-process) has a test at all.

4. **Canonicalization that hides corruption.** A normalizer applied before scoring must not equate a corrupted value with its truth. Two edge cases bit us and are instructive:

   - *Leading zeros.* Coercing zero-padded identifiers through integer parsing equates a truth `0042` with a corrupted `42`, hiding the corruption. Padding is now preserved as data: a zero-padded digit string keeps its padding and is compared as text.
   - *Signed zero and near-zero.* A float normalizer must map -0.0 and tiny negatives to the same string as 0.0, or a benign representation difference scores as a false extraction error. The trimmed-decimal rule collapses `""`, `"-"`, `"-0"` to `"0"`.
   - *Over-broad numeric coercion.* Python's `int()`/`float()` accept scientific notation, underscores (`1_000`), and non-ASCII Unicode digits, silently equating distinct source strings. The normalizer matches only ASCII decimals (a `[0-9]` class, not the Unicode-aware `\d`), keeping distinct strings distinct.

   Each is a way canonicalization can quietly launder a real difference into apparent agreement—the same silent-error pathology, one layer up.

5. **A gameable silent-error metric.** Covered in §5.2: any flag suppressing the count, and substring collusion. The fix requires a flag to name the affected column as a whole token.

6. **Rounding loss up to perfection.** Percentage formatting with round-half maps anything in $[0.9995, 1.0)$ to a displayed 100.0%—a false-perfect in a benchmark whose thesis is exactly which arm is lossless. The formatter now prints 100.0% only when the value is exactly 1.0 and otherwise clamps below the ceiling.

6.4 Adversarial review of the instrument

The harness was subjected to an independent adversarial review charged with discrediting it in either direction. The review *confirmed* the three load-bearing honesty properties—ground-truth independence, the lossless control, and the structural silence of the screen arm—and surfaced eight findings, two of them material and already described: the gameable silent-error metric (finding 5 above) and the canonicalization leading-zero flaw (finding 4). A subsequent, deeper bug-hunt across the whole harness—each finding independently reproduced before repair—fixed a further set of latent defects: a substring match that could falsely credit awareness of a wrong cell, the round-up-to-100.0% formatting, and numeric canonicalization that collapsed distinct strings such as underscore- and Unicode-digit forms. Each finding ships with its own regression test.

Two properties of this process are worth stating as method. First, **the reported numbers were unchanged by these repairs**: the fixes closed ways the harness *could* have lied, not ways it *had*. The point is that the instrument earns trust by bleeding under scrutiny and being fixed, not by appearing clean on first inspection. Second, we red-teamed the measuring instrument, not just the system—because a benchmark about silent errors that itself contains a silent error (the per-process seeding bug did exactly this) would be self-refuting. We regard adversarial review of the instrument as a required, not optional, component of a fidelity benchmark.

Remark (On the modeled arm's remaining honesty gaps). Two we disclose rather than paper over. The repeated-page fault models a scraper with *no* frame deduplication; a dedup-aware scraper detects content-identical repeats, so that one fault class—unlike the other three—is avoidable, and we say so. And the modeled arm's fault rates are *chosen*, which is precisely why the real-OCR arm exists: to replace chosen rates with a real engine's real errors on a real render.

7 Case Study: A Windows 98 System Disk

We evaluated the full path on a genuine Windows 98 machine image containing mixed-era stores. The acquisition layer mounted the disk offline; the decoding layer identified and parsed dBase, Jet 3/4, and BIFF workbooks along with delimited and configuration sources.

- **34,803 records** extracted across all stores, including a complete Jet-era Northwind database (all tables, with referential structure preserved), with row counts reconciled against in-store metadata.
- **Zero optical steps** anywhere in the data path: no OCR, no coordinate-based UI traversal, no vision model between store and output.
- **Grounded Q&A** over the result: structured questions about the extracted data were answered by compiled SQL with supporting rows returned for inspection.
- **Migration Package + standalone verification**: the bundle is assembled into portable DDL and an evidence-backed foreign-key graph (each edge proven by a referential-integrity subset check—on the case data, `invoice`→`customer` and `items`→`invoice`), sealed with per-file digests and re-verifiable by a stdlib-only third-party tool whose tamper-detection is tested.
- **709 automated tests** pin the pipeline, the benchmark harness, the sight prototype, the verifier, and the package assembler: per-format structural cases, failure-mode (malformed input) cases, and one regression test per adversarial-review finding.

7.1 Failure-mode analysis by arm

The case study is a single-machine pilot, so we do not claim a distribution over machines. What we can do is analyze, per extraction arm, *how each would fail* on the data actually present—turning the abstract error taxonomy of §3 into concrete predictions about this store.

Tier 1 (the path taken). The disk-direct path's failure modes on this store are all detectable and were exercised: a memo field that will not resolve (surfaced as a quality flag with the field named), an out-of-range or fictitious date serial (preserved with a flag rather than silently coerced), a precision-risk MONEY/NUMERIC value that cannot round-trip through binary float (preserved as exact decimal text, the affected column marked text with a flag). None of these invents a value; each escalates. The Northwind subset is a useful stress case here because it contains OLE/photo columns (yield `None` plus a quality flag, never a fabricated blob) and memo columns (real text when resolvable, flagged when not).

Tier 4 (the path refused), predicted. Had this disk been drained by screen traversal, the failure modes would be the ones the benchmark measures, now specialized to this data. The invoice and order-detail tables are keyed by identifiers with embedded digits; the real-OCR arm's 0/O confusion (§6) would corrupt those keys and collapse the joins between orders, order-details, customers, and products—the exact referential structure the Migration Package's evidence-backed graph preserves would instead be silently severed. Paginated traversal of the multi-thousand-row order-detail table would be the most exposed to early-scroll-stop and torn-row faults, precisely where the data is densest and a dropped tail is least likely to be noticed. We state this as an analysis of predicted failure, not a measured comparison on this machine; the measured comparison lives in the controlled benchmark on the synthetic fixture (§6).

7.2 What was and was not proven here

We report this as a *pilot-grade* result: single machine, single operator, formats bounded to those listed. It is nonetheless the shape of the argument: the disk that conventional practice would have paged through a screen for weeks—at unknown fidelity—was drained in one offline pass, at full fidelity, with the completeness check machine-verifiable.

Separately and precisely: the control-tree layer has been exercised against live guests from an NT-era Windows through modern Windows, enumerating full application control trees headlessly—this proves control-tree *capture* and enumeration; the receipted actuation loop (§4.4) is validated at the workflow and receipt level, not yet run end-to-end against a live guest at pilot scale over a sustained soak. The witness layer has been validated with per-frame digest verification under fault injection (source outages degrade evidence capture without corrupting or halting extraction). We are deliberate about these boundaries because the labeling discipline is part of the claim.

8 A Bounded Prototype: Terminal-Art Sight for Text-Only Models

A text-only model is blind: it cannot see a screen. The control-card layer suggests a remedy that stays on the right side of the fidelity hierarchy—render the structured control catalog into a spatial character grid ("terminal-art"), so the model reads layout from structured truth, never from pixels. We built this as a prototype and tested it honestly, including a run that *refuted* the strong version of the claim. We report it in full, refutation included, because a bounded honest result is worth more than an unbounded optimistic one—and because the refutation is itself evidence that the evaluation was not rigged.

Mechanical faithfulness. The renderer is deterministic: on a synthetic 11-control scene it recovers every card (100% label recall), preserves every unambiguous spatial relation in the grid (89/89), sources every rendered glyph to a card (complete provenance), and produces byte-identical output across runs—a guarantee a vision model's description cannot make.

Readability, measured against a confound. Readability was measured with real text-only agents (five per condition, reading only their prompt, with no access to the source cards).

- *Run 1*, on a normally-labeled scene against a reading-order list, **refuted** the strong claim: the list scored 94% on spatial questions and terminal-art 89%. A capable model answers "which

button is below *What's New*?" from the labels and reading order alone; the spatial view adds nothing, and its artifacts can distract.

- *Run 2* removed that confound—opaque controls (labels stripped) and a position-free catalog baseline, so the only source of layout is the render. Here terminal-art scored **100%** on spatial questions against **34%** for the flat catalog (10% on the pure "pick the control at this position" subset, i.e. near chance).

Hardening against two objections. That it is a fluke of one small scene: a mechanical sweep confirms the render stays fully faithful (100% recall, 100% spatial fidelity, complete provenance, deterministic) across scenes from 4 to 24 controls. And that a dense grid might be answerable from id numbering alone—which our first dense grid was: numbered row-major, *both* conditions scored 100%, because a model infers the convention ("below [1] is [5]") with no layout at all. That is a confound, not a win, and we report it as one. Re-run with ids permuted so number carries no position, the dense 16-control grid gives the sharpest result: terminal-art **100%** on spatial questions versus the position-free catalog at **0%** on picking a control by position—the catalog confidently applies the row-major convention, which the scrambled render defeats.

Honest, bounded conclusion. Terminal-art is not a universal upgrade for reading UIs—when a control's name already implies its place, the model needs no picture—but for opaque or ambiguous layouts it is the difference between near-chance and perfect, delivered deterministically and with full provenance. That is exactly the residual, owner-drawn case §4.3 flags. The prototype sits outside the shipped path and is labeled PROTOTYPE; generalizing it beyond synthetic scenes, and integrating it as a live perception channel for the operate job, is future work (§11).

9 Discussion

Why the industry defaults to the worst tier. Screen-driving wins adoption because it demands nothing of the operator: no format knowledge, no disk access, no comprehension of the system. Our position is that this is an economics problem masquerading as a technical one. Format literacy is a *capital cost*—a dBase or BIFF decoder is written once and amortized across every migration—whereas screen-scraping error is an *operating cost* paid, invisibly, by every record of every run. A fidelity-ordered pipeline exists to pay the capital cost once, publicly, and to make the operating cost visible when it is incurred.

Vision models change the temptation, not the analysis. Modern VLMs make Tier 4 dramatically easier and no safer for migration: they raise mean accuracy while leaving the error profile silent and unbounded (Proposition 2), and they add a generative failure mode (plausible confabulation) that classical OCR lacked. The hierarchy's ordering is unaffected. If anything, cheap capable Tier-4 automation makes architectural enforcement of the hierarchy *more* urgent, because the path of least resistance now produces beautiful, confidently wrong migrations at scale. A model that answers "what is this invoice number?" with a fluent, wrong string is worse than an OCR engine that garbles it, because the garble at least looks wrong.

The witness role generalizes. Separating the evidence channel from the data channel is, we believe, the most portable contribution. Any autonomous system acting on infrastructure benefits from an independent, hash-chained record of what the world looked like while it acted—provided

that record is structurally prevented from becoming an input. The discipline is not specific to legacy migration; it is a general pattern for accountable automation, and the directional constraint (frames corroborate, never populate) is what keeps the witness from quietly re-entering the data path.

Broader impact: AI training-data integrity. There is a second audience for fidelity-ordered, provenance-first extraction beyond migration: anyone assembling training or evaluation corpora from legacy or scanned sources. A screen-scraped corpus carries silent, unbounded, unflagged corruption into every model trained on it, and the corruption is exactly the kind—transposed digits, collapsed keys, dropped tails—that a downstream model will learn as if it were signal. Provenance receipts that bind each extracted row to source bytes give a corpus curator a mechanism to audit, and to *exclude by provenance tier*: admit Tier-1 rows, quarantine Tier-4 ones. We do not evaluate this application here, but the architecture supports it directly, and we flag it as a motivation for the field to take extraction fidelity seriously rather than a claim we have validated.

Broader impact: regulated-industry migration. In healthcare, finance, and government, the migrated copy is not merely archival—it is subject to retention, audit, and legal-hold obligations, and its integrity may need to be demonstrated to a regulator years later. A migration whose completeness and integrity are properties of a portable, re-verifiable artifact—rather than assertions of a vendor whose pipeline no longer exists—is a materially different compliance posture from a screen-scraped export. Realizing that posture fully requires the authenticity work (M4) and cross-host reproducibility (M5) we scope as future; we note the direction without overstating present capability.

10 Limitations and Threats to Validity

We hold this section to the standard the paper argues for: an honest limitations section is what makes the rest of the paper trustworthy, and a limitation declared is worth more than a claim inflated. We separate limitations of scope, of evaluation, and of the paper's own disclosure.

10.1 Scope of the system

Format surface is finite. The decoders cover the dominant small-business stores of the target era (dBase/FoxPro, Jet 3/4, BIFF5/8, delimited, INI, registry, text), but the long tail is real: Btrieve, Paradox, EBCDIC, and pre-relational vertical formats remain future work. The fixed-width/COBOL-copybook case is prototyped as a *spec-driven* decoder that deliberately refuses to auto-infer a layout—guessing column boundaries from bytes would be precisely the silent-error trap the hierarchy condemns—so it decodes only against an explicit copybook, and wiring it into the auto-detecting pipeline (which has no copybook-spec input path today) is open work. This is a PROTOTYPE, not a shipped capability.

Encrypted stores fall back down the hierarchy. Encrypted or password-protected stores currently require the native application, dropping those fields to Tier 2/3 and forfeiting the structural guarantee of Tier 1. The system detects and reports encryption rather than silently failing, but it cannot decode what it cannot read.

Tier 2 is the least automated tier. Native export (report-to-file) is the least developed channel in the current implementation; in practice migration data flows through Tier 1 and, for residue, Tier 3.

10.2 Threats to the evaluation

Single-guest case study. The 34,803-row result is a single machine, single operator, single OS era. It demonstrates the shape of the argument at pilot scale; it does not establish a distribution over machines, formats, or operators, and we make no fleet-scale claim (M1 fleet is explicitly unmeasured).

Single-machine benchmark on a synthetic fixture. The controlled benchmark runs on one host against a deterministic synthetic fixture, not against live legacy guests. Its disk-direct arm is production code, but its adversary is a *modeled* channel with chosen fault rates plus a single, deliberately generous real-OCR read. Conjecture 1—that Tier 1 dominates Tier 4 on both fidelity axes across *real* legacy displays—is not established by this benchmark; a live, public, multi-guest benchmark is what would test it (§11). The real-OCR arm's 0% key-corruption result is a lower bound on one clean render, not a population estimate.

No third-party replication. No party outside the author has run the benchmark or reproduced the case study. Cross-host byte reproducibility (M5) is unmeasured; we have only cross-process reproducibility within one host. The adversarial review of the harness was independent of the harness's construction but not independent of the author.

The actuation loop is not soaked. The operate job's receipted actuation is validated at the workflow and receipt level and demonstrated at pilot scale; long-duration soak (M6/M7 at scale) is unstarted. We report control-tree *capture* as demonstrated on live guests and the receipted actuation loop as pilot/future, and we have been careful throughout not to let the former's maturity borrow credibility for the latter.

10.3 The disclosure–reproducibility tension

This paper describes two subsystems—the acquisition substrate and the receipt scheme—abstractly, by the properties they guarantee (cold offline read, acquisition-time integrity, tamper-evident signed receipts) rather than by their construction. That abstraction is a deliberate choice to protect implementation details that are not the scientific contribution, but it creates a genuine tension with reproducibility that we will not paper over: **a reader cannot, from this paper alone, independently rebuild the acquisition layer or audit the signing scheme.** The consequences are concrete:

- The forensic properties of acquisition (Assumption 1) are stated as contract, not demonstrated to the reader; a skeptical reader must take the cold-read guarantee on the described interface rather than verify it.
- The receipt scheme's *integrity* guarantee is checkable through the standalone verifier (which *is* available and dependency-free), but the signing/authenticity design is not disclosed, so a reader cannot assess the authenticity story—consistent with our declining to claim authenticity (M4) until it is both built and disclosable.

- The decoding layer, the benchmark harness, the verifier, and the package assembler *are* fully inspectable in the accompanying implementation, so the core fidelity claims (Theorem 1, the benchmark, M1–M5 integrity) do not depend on the withheld subsystems. The withheld parts sit around the extraction claim, not inside it.

A proposed audit/escrow path. The honest resolution is not to publish the withheld internals but to make them auditable under controlled terms. We propose, as future work, an *audit-and-escrow* arrangement: the acquisition and receipt-signing components are placed with a trusted third party (or made available under NDA to a program evaluator), who can (a) confirm the cold-read and integrity properties against a specification, (b) re-sign and re-verify a sample package end-to-end, and (c) attest to the properties without redistributing the implementation. This preserves the moat while converting "take our word" into "an independent party checked." Until such an arrangement exists, the acquisition and signing guarantees should be read as *author-attested*, and only the integrity half of provenance (via the open verifier) as reader-checkable.

10.4 What these limitations do and do not touch

The limitations above bound the empirical reach of the evaluation and the reader's ability to audit two subsystems. They do not touch the central architectural claim, which is structural rather than empirical: Theorem 1 holds regardless of how many machines we tested, and the fidelity ordering's enforcement is a design property visible in the inspectable decoding pipeline. Where an empirical claim would over-reach the evidence—fleet fidelity, real-display dominance, cross-host reproducibility, actuation soak—we have marked it unmeasured or conjectural rather than assert it.

11 Future Work

We name the work that would elevate the paper's conjectural and unmeasured claims to measured ones, in rough priority order. Each corresponds to a specific gap identified above, and each is scoped as work not present capability.

1. **A live, public fidelity benchmark.** Replace the modeled screen channel with live legacy guests under emulation, a production-grade screen-extraction baseline (best current practice: frame dedup, retry logic, VLM reading) driving the real GUI, versus the disk-direct path, versus ground truth planted in genuine on-disk stores, with fault injection at the guest/display/timing level. Releasing the benchmark, ground-truth corpus, and scoring harness publicly makes any performer's extraction claims falsifiable and is what would test Conjecture 1 against real displays rather than a chosen model.

2. **Fleet-scale extraction.** Extend the single-machine pipeline to operator-independent, parallel acquisition across many guest images with reconciliation against each store's self-described counts as a hard gate, measuring M1/M2 at scale.

3. **Provenance authenticity (M4) and cross-host reproducibility (M5).** Add a signature over the manifest—requiring asymmetric cryptography and a key-distribution channel—so provenance carries authenticity, not only integrity; and establish byte-identical extraction outputs across independent hosts. Both are named limitations today.

4. **Actuation soak and receipt-chain formalization (M6/M7).** Run the propose/actuate/receipt loop against live guests over a sustained soak (10^4+ receipted actuations), with a formally specified, machine-checkable receipt chain (signature scheme, frame binding, one-step semantics) and adversarial perturbation of the guest UI.

5. **Format long-tail.** Bring Btrieve, Paradox, and EBCDIC into the auto-detecting pipeline, and wire the spec-driven fixed-width/COBOL-copybook decoder in through a copybook-spec input path—preserving its refusal to auto-infer layouts.

6. **Audit/escrow of the withheld subsystems.** Stand up the audit-and-escrow arrangement of §10.3 so the acquisition and signing guarantees move from author-attested to independently-attested without publishing the implementation.

7. **Terminal-art sight as a live channel.** Generalize the sight renderer beyond synthetic scenes and integrate it as a perception channel for the operate job, on the specific residual case (opaque, owner-drawn layouts) where the bounded result shows it is decisive.

12 Conclusion

Migration is the one context in which extraction errors are permanent: the extracted copy outlives its source and becomes the record. An architecture for legacy migration should therefore be judged not by how cleverly it drives an old interface but by how rarely it needs to, and by whether its output can be verified without trusting its process. The system described here reads the store, not the screen; signs what it reads; reconciles what it signs against the source's own accounting; and keeps the screen on as a witness.

We have tried to make the argument checkable rather than rhetorical. The structural core—that decoding a byte structure has no glyph-substitution pathway—is a theorem, not a hope (Theorem 1). The fidelity gap is a measurement gated by a runtime lossless-control predicate (Proposition 1), and its most damning single number—100% primary-key corruption under a clean OCR read—comes from a real engine, not a chosen rate. The claims we cannot yet prove—dominance across real displays, fleet fidelity, cross-host reproducibility, actuation soak—are labeled as conjecture or unmeasured, because a paper about silent errors that hid its own uncertainty would refute itself.

We commend the pattern—*fidelity hierarchy, structural provenance, screen as witness*—as a default posture for the coming decade of decommissioning, in which an enormous quantity of human records will either be moved carefully, once, or lost.

Status of this report: extended technical report / arXiv preprint describing a working pilot-grade system; single-machine evaluation, no third-party replication. Two subsystems (acquisition, receipt signing) are described by contract rather than construction; see §10.3 for the resulting reproducibility tension and the proposed audit path. The described components are the author's implementations; no third-party benchmark is claimed. All benchmark figures are carried over from the underlying report and its results artifact; none was recomputed for this edition.

References

[1] D. S. H. Rosenthal, T. Robertson, T. Lipkis, V. Reich, and S. Morabito. Requirements for Digital Preservation Systems: A Bottom-Up Approach. *D-Lib Magazine*, 11(11), 2005. (LOCKSS.)

[2] Consultative Committee for Space Data Systems (CCSDS). Reference Model for an Open Archival Information System (OAIS). CCSDS 650.0-M-2 (Magenta Book) / ISO 14721, 2012.

[3] D. von Suchodoletz, K. Rechert, and I. Valizada. Towards Emulation-as-a-Service: Cloud Services for Versatile Digital Object Access. *International Journal of Digital Curation*, 8(1), 2013. https://doi.org/10.2218/ijdc.v8i1.250

[4] R. Di Cosmo and S. Zacchiroli. Software Heritage: Why and How to Preserve Software Source Code. In *Proc. iPRES*, 2017.

[5] World Wide Web Consortium (W3C). PROV-DM: The PROV Data Model. W3C Recommendation, 30 April 2013. https://www.w3.org/TR/prov-dm/

[6] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer. Provenance-Aware Storage Systems. In *Proc. USENIX Annual Technical Conference*, 2006.

[7] K. Wolstencroft et al. The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research*, 41(W1):W557–W561, 2013. https://doi.org/10.1093/nar/gkt328

[8] R. C. Merkle. A Digital Signature Based on a Conventional Encryption Function. In *Advances in Cryptology — CRYPTO '87*, LNCS 293, 1988. (Merkle/hash-tree tamper-evidence.)

[9] National Institute of Standards and Technology. Computer Forensics Tool Testing (CFTT) Program: Hardware Write Blocker (HWB) and Software Write Block Tool specifications. NIST, ongoing. https://www.nist.gov/itl/ssd/software-quality-group/computer-forensics-tool-testing-program-cftt

[10] B. Carrier. *File System Forensic Analysis*. Addison-Wesley, 2005. (The Sleuth Kit lineage; image-first, read-only acquisition.)

[11] G. Nagy. Twenty Years of Document Image Analysis in PAMI. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(1):38–62, 2000. https://doi.org/10.1109/34.824820

[12] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou. LayoutLM: Pre-training of Text and Layout for Document Image Understanding. In *Proc. ACM SIGKDD*, 2020. https://doi.org/10.1145/3394486.3403172

[13] Defense Advanced Research Projects Agency (DARPA). Program pages for legacy-software efforts: V-SPELLS (Verified Security and Performance Enhancement of Large Legacy Software), Assured Micropatching (AMP), and TRACTOR (Translating All C to Rust). DARPA public program pages, accessed September 2026. https://www.darpa.mil/research/programs/verified-security-and-performance-enhancement-of-large-legacy-software; https://www.darpa.mil/research/programs/assured-micropatching; https://www.darpa.mil/research/programs/translating-all-c-to-rust

[14] Google. Use of the AccessibilityService API. Google Play Developer Program Policy (Play Console Help), accessed September 2026. https://support.google.com/googleplay/android-developer/answer/10964491 (Prohibits Accessibility-API use that autonomously initiates, plans, and executes actions, while permitting deterministic, rule-based automation that follows a static, human-defined script.)