



SORTCEROR

A BRIEF HISTORY OF ORDERED CHAOS

THE COMPLETE DEVELOPMENT RECORD

07 MAY 2026 - 26 AUGUST 2026

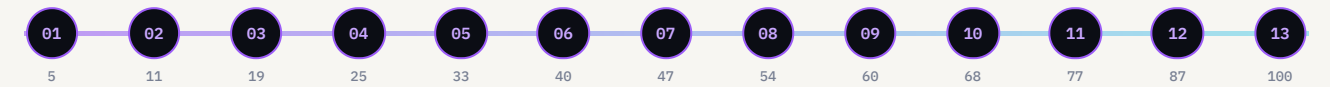
“The original idea was not to build some fucking cathedral of abstractions. I wanted to stop dumping plugins into a pile, and I wanted to fix it without wrecking the database.”

CONCEIVED, DESIGNED AND BUILT BY CHASE

A factual chronology, a design-history record, and a brutally honest post-mortem.

Contents

Thirteen architectural ages, followed by the pivots that mattered, the architecture that survived, the current package truth and the laws I would now refuse to build without.



01	THE MANUAL MESS	07-23 MAY 2026	5
02	THE SAFE AUDITOR	28 MAY-04 JUNE 2026	11
03	THE LOCAL EVIDENCE PIVOT	08-13 JUNE 2026	19
04	THE REWRITE AND CORPUS-TRUTH ERA	15-26 JUNE 2026	25
05	SAFE EXECUTION BECOMES A PRODUCT	05-12 JULY 2026	33
06	THE PRODUCT-SPINE CORRECTION	13-18 JULY 2026	40
07	CONSUMERISATION AND RUNTIME TRUTH	19-26 JULY 2026	47
08	THE INTELLIGENCE PROGRAMME	26-30 JULY 2026	54
09	THE GREEN MACHINE THAT WASN'T	30 JULY-08 AUGUST 2026	60
10	ONE SPINE, ONE OWNER	09-13 AUGUST 2026	68
11	BURNING THE BUREAUCRACY	13-20 AUGUST 2026	77
12	THE PACKAGE IS THE PRODUCT	16-22 AUGUST 2026	87
13	CONSUMER TRUTH, AGAIN	23-26 AUGUST 2026	100
FINAL RETROSPECTIVE			112-120

AUTHOR'S NOTE

Why this document exists

This document records the first 112 days of Sortceror: what I built, why I built it that way, what broke, what changed my mind, what the system actually proved, and what it was still failing to do at the final documentary cut-off.

It is intentionally not a sanitised release note. The dead ends are included because they explain the architecture better than the clean checkpoints do. A lot of the strongest laws came from me first doing the opposite, watching it fail, and then refusing to let a polished report rewrite what physically happened.

“Chroma is not a fucking reverb.”

THE FIRST HONEST LESSON

That tiny correction was the earliest proof that a plausible classification can still be confidently wrong. The rest of the architecture is, in one way or another, an attempt to stop the machine from hiding that kind of mistake.

THE LATER HONEST LESSON

A package can pass thousands of tests, reproduce cleanly, launch, render beautifully, and still fail before a normal producer can select a fucking folder. Green means the ordinary human journey works. Nothing else gets to borrow the word.

CHRONOLOGY NOTE

Dates follow the retained project record. Work that crossed midnight is grouped under the date on which its result or gate was recorded. Exact hashes, timings and counts appear where they materially establish identity or proof. Where the record does not support false precision, I do not invent it.



Executive Overview

The project in one page

Sortceror began as manual FL Studio folder triage and became a local-first plugin intelligence system with canonical SQLite memory, evidence-separated authority, reversible execution, a packaged Windows product spine, a distinctive visual language, and a very expensive education in the difference between internal proof and consumer truth.

START Manual folder triage A producer taxonomy, VST cleanup and vendor scripts exposed the identity and safety problem.	FOUNDATION Evidence before authority Reports, hashes, reviewed knowledge, canonical memory, no-overwrite and rollback became permanent law.
PRODUCT One reversible spine Workspace, scan, immutable snapshot, resolution, plan, execute, verify, History and Undo.	AUGUST Architecture on trial The repository was simplified, competing owners were removed, package inputs were frozen and the packagers were forced to prove themselves physically.
PROOF 241 reversible moves A complete copied-root journey passed: first run, review, planning, physical movement, verification, exact Undo, Scan New and relaunch.	TODAY Rolling package remediation The latest consumer package is not yet installer or live-user accepted. Startup, picker ownership and progress truth have been repaired; the current returning-user resolver path is still being forced back to consumer-scale behaviour.

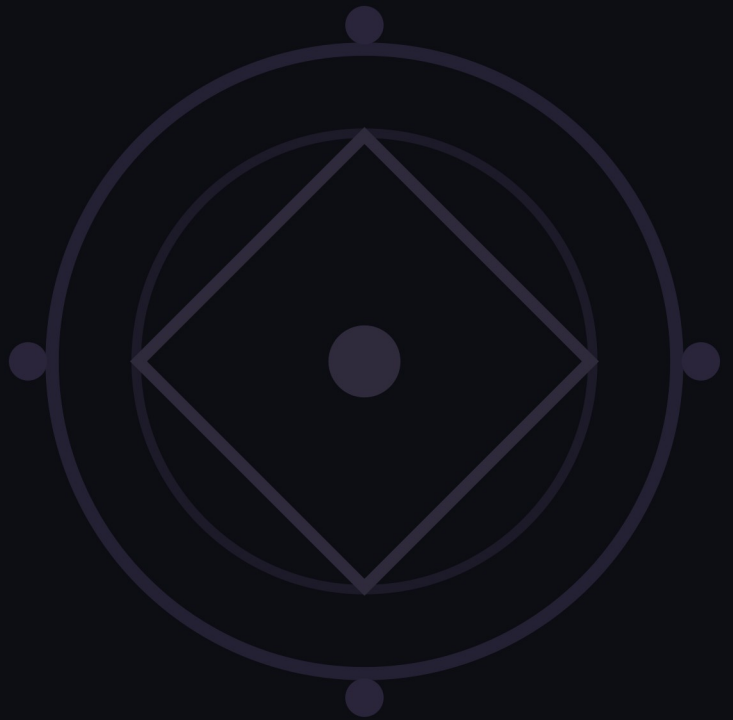
THE CONTROLLING FACT	Sortceror is much further along than a folder sorter and less finished than its largest green test run ever made it look.
---------------------------------	---

PART I

THE MANUAL MESS

07–23 MAY 2026

Before there was a product, there was a folder tree,
too many plugins, and a growing suspicion that names
were lying to me.



ARCHITECTURAL AGE 01

PART 01 / 07-23 MAY 2026

THE MANUAL MESS

Sortceror did not begin as software. It began as me trying to clean up a ridiculous FL Studio Plugin Database by hand. I had thousands of entries, duplicate formats, misleading names, vendor variants, empty folders and loose plugins sitting in places that made no musical sense. The first job was not "build intelligence". It was: decide what the fuck the taxonomy even is, then stop breaking it while moving things around.

07 MAY 2026

Manual taxonomy becomes a real system

WHAT CHANGED I started by placing loose plugins into a practical producer-facing folder tree: dynamics, EQ, filters, stereo and spatial tools, distortion and saturation, lo-fi, delay, reverb, modulation, vocals, guitar and bass, multi-effects, mastering, controllers, Patcher, visual tools and gain. This was not an academic ontology. It was based on the question I actually ask while producing: "What job am I trying to do right now?"

WHY I DID IT The immediate goal was retrieval speed. A plugin collection is only useful when I can find the right tool before the musical idea dies. Vendor-first sorting was not enough; broad "Effects" folders were not enough; and FL Studio's default discovery layout certainly was not enough.

HINDSIGHT This was the first durable Sortceror principle: taxonomy must serve the producer's decision, not the plugin vendor's marketing copy. The mistake was assuming the taxonomy alone would solve the problem. It only defined the destination; it did not solve identity, evidence or safe movement.

EVIDENCE SNAPSHOT

- Initial functional roots established
- Producer task, not vendor, became the primary organising axis

07 MAY 2026

The taxonomy expands under pressure

WHAT CHANGED As actual plugins were reviewed, the tree grew. Pitch processing needed its own root. Instruments could not be left mixed with effects. Restoration tools such as UNVEIL, UNCHIRP and UNFILTER did not belong under vocals merely because they often touch recorded audio. Channel strips and specialist mastering processors needed clearer separation. Scaler, VFX Sequencer and similar tools belonged under controllers; Fruity LSD, Serum and Vital belonged under instruments; visual mixers belonged under visual tools.

WHY I DID IT A useful taxonomy has to survive real edge cases. Every awkward plugin exposed either a missing category, an overloaded category or a false assumption about what users look for.

HINDSIGHT The good part was allowing the tree to change. The bad part was changing it reactively without yet having a canonical schema, migration rules or validation. At this stage the taxonomy lived mostly in my head and in conversation history - which was obviously not going to scale.

EVIDENCE SNAPSHOT

- New roots included Fx-Pitch, Instruments, Fx-Restoration and Fx-Channel Strip

07 MAY 2026 ARCHITECTURAL PIVOT

"Chroma is not a fucking reverb"

WHAT CHANGED A bad classification put Chroma under reverb. I immediately corrected it to multi-effects. That sounds tiny, but it was the first clean demonstration that plugin names, presets and marketing language can produce confidently wrong category guesses.

WHY I DID IT I wanted exact placement, not plausible-sounding placement. A plugin sorter that is wrong with confidence is worse than a folder of unsorted plugins because it hides the error inside an apparently organised system.

HINDSIGHT In hindsight, this sentence was the seed of the entire evidence doctrine. Static name matching was always going to hit a wall. The correct response was not a bigger regex pile. It was to gather better evidence, represent uncertainty and refuse authority when the evidence did not support it.

EVIDENCE SNAPSHOT

- Misclassification corrected: Chroma → Fx-MultiFX

09 MAY 2026

The VST3-over-VST2 cleanup problem

WHAT CHANGED I wanted FL Studio to prefer VST3 versions without forcing me to re-sort thousands of database entries. I backed up the full Plugin Database, rescanned with Verify plugins, Rescan previously verified plugins, Combine 32/64-bit versions and Combine presets for VST/VST3 versions, then investigated whether sorted .fst entries could be replaced from verified VST3 sources while preserving the existing folder tree.

WHY I DID IT The requirement was already recognisably Sortceror-shaped: preserve the user's organisation, make the smallest safe change, record what changed and avoid touching actual plugin installations.

HINDSIGHT The important discovery was that an .fst file is an FL Studio database preset or shortcut, not proof of plugin format. The database entry, the binary implementation and the displayed product identity are related but not interchangeable. Later entity resolution work came directly from this distinction.

EVIDENCE SNAPSHOT

- Full backup: Plugin database BACKUP
- Rescan combined VST/VST3 and 32/64-bit variants
- Only database entries were in scope

09 MAY 2026

The leftover report proves that UNKNOWN is not a verdict

WHAT CHANGED After matching what could be matched exactly, 130 entries remained: 110 marked UNKNOWN, 20 marked LIKELY VST2 and zero exact VST3 source matches left. Manual-check candidates included Auto-Tune Evo, Effectrix, Guitar Rig 6, HalfTime, RC-20, ShaperBox and older VolumeShaper variants. Fruity Blood Overdrive was a useful false alarm because it was native FL, not a legacy VST problem.

WHY I DID IT The report was intended to reduce a huge migration to a finite review queue. It did that, but it also exposed how weak a label such as UNKNOWN can be when the underlying parser does not understand native tools, wrapper metadata or product variants.

HINDSIGHT This was an early lesson in honest states. UNKNOWN should mean "the current evidence cannot establish the answer", not "probably broken" and definitely not "safe to delete". That distinction later became blocked, unresolved, review-only and fail-closed states throughout the product.

EVIDENCE SNAPSHOT

- 130 leftovers
- 110 UNKNOWN
- 20 LIKELY VST2
- 0 exact VST3 matches remaining

12 MAY 2026

The first reusable sorting script

WHAT CHANGED I built Sort-Acustica-FL.ps1 to move Acustica .fst entries out of broad folders into a more useful category, subtype and vendor structure. It operated only inside the FL Studio Plugin Database, backed up the affected tree, wrote CSV logs and never touched installed binaries. The first category set included dynamics, channel strips, distortion and saturation, mastering, reverb, delay and tools.

WHY I DID IT Manual sorting had become repetitive enough that a reusable rule-driven tool was justified. The aim was not yet autonomous recognition; it was consistent application of decisions I had already made.

HINDSIGHT The script was valuable because it established the first executable boundary: database presets could move, plugin installs could not. The weak point was that classification still depended on handwritten knowledge and naming patterns. It automated labour, not understanding.

EVIDENCE SNAPSHOT

- Script: Sort-Acustica-FL.ps1
- Backups and CSV logs
- Moves limited to .fst database entries

12 MAY 2026 ARCHITECTURAL PIVOT

I initially rejected the dry run

WHAT CHANGED My first instinct was to skip the preview and apply the moves directly. I had already reviewed the plugins and wanted the machine to get on with it. That instinct was understandable and still wrong as a product default.

WHY I DID IT At the time, the direct-apply preference came from impatience with repetitive work. I was the only user, I knew the folders, and a backup existed. The perceived cost of review felt higher than the risk.

HINDSIGHT This is one of the clearest things I would reverse from day one. A backup is not the same as a legible plan. Safe automation needs preview, explicit authority, source validation, no-overwrite rules, execution journalling and rollback. Sortceror eventually adopted all of that, but only after I had first argued against the friction that later became one of its strongest features.

EVIDENCE SNAPSHOT

- Early preference: direct apply
- Later product law: review-first and token-gated mutation

12 MAY 2026

ZL stops being treated as a category

WHAT CHANGED While restructuring Acustica entries, I realised that ZL meant zero-latency variant. It described an implementation characteristic, not a musical function. Normal and ZL editions could be separated or nested, but they should not distort the core taxonomy.

WHY I DID IT Variant labels had to be represented without confusing them with product role. A low-latency mastering compressor is still a mastering compressor; ZL is a variant attribute.

HINDSIGHT This small correction foreshadowed the later identity graph: vendor, product, version, edition, format, implementation and functional role are separate axes. Collapsing them into one plugin name string guarantees subtle errors.

EVIDENCE SNAPSHOT

- ZL redefined as a variant/implementation attribute

12 MAY 2026

Vendor-wide restructuring and the first accuracy demand

WHAT CHANGED The scope expanded beyond Acustica. I wanted vendor folders such as Eventide, UJAM, SSL, iZotope and Plugin Alliance inside the correct functional subfolders; loose Installed\Effects\New entries needed to land in the main Effects tree; empty vendor folders needed cleanup; duplicates needed scrutiny; and the audit CSVs needed to be accurate one plugin at a time.

WHY I DID IT The point was to turn an ad hoc cleanup into a repeatable library-maintenance workflow. The same pattern - scan, classify, preview, move, clean residue, report - kept recurring.

HINDSIGHT The phrase "100% accuracy" was directionally right but operationally undefined. I had no held-out benchmark, no authority model and no measured false-positive rate. Later work had to turn that demand into explicit metrics, abstention and proof instead of a confident slogan.

EVIDENCE SNAPSHOT

- Expanded vendor/category coverage
- Loose New-folder ingestion
- Empty-folder and duplicate cleanup requirements

23 MAY 2026 ARCHITECTURAL PIVOT

Raw chats are a messy evidence pile

WHAT CHANGED I explicitly stopped treating conversation history as the project database. The working analogy was blunt: the chats were a messy evidence pile; the real project sources needed to be the cleaned case file. I proposed canonical files such as Plugin_Inventory.csv and FL_Plugin_Database_Structure.md, with fields for vendor, product, format, type, use case, favourites, low-latency variants, VST3 preference, target folder and notes.

WHY I DID IT The immediate goal was retrieval and consistency. I wanted a machine-readable inventory that could answer specific questions instead of relying on screenshots, memory and vibes.

HINDSIGHT This was the conceptual bridge to canonical memory. CSV was the first target because it was simple and inspectable. SQLite later replaced it as authority because relationships, provenance, conflicts, revisions and execution history needed more structure than flat files could safely provide.

EVIDENCE SNAPSHOT

- Canonical inventory concept established
- Structured data separated from narrative documentation

PART 1 RETROSPECTIVE

What this phase actually changed

The prehistory matters because almost every later subsystem already existed here in primitive form. There was a taxonomy, a scanner, a move operation, a variant problem, a duplicate problem, a review queue, backups, logs and unknowns. What did not exist was a coherent model connecting them. I was still treating the task as "sort files accurately" when the real problem was "establish plugin identity, infer function from evidence, translate that into a user taxonomy and mutate a live database without lying about certainty."

WHAT I WOULD DO DIFFERENTLY

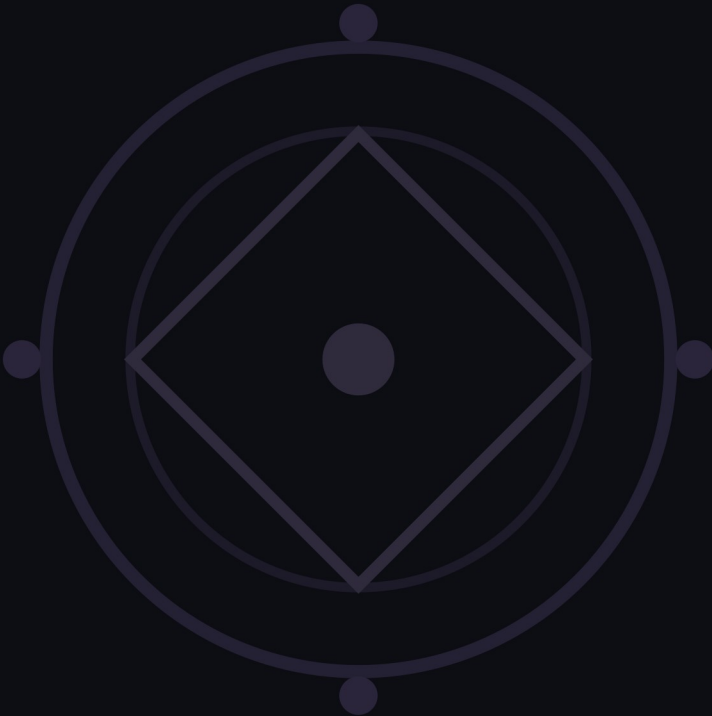
1. I would have written the canonical taxonomy and identity schema before writing vendor-specific movers.
2. I would have made dry-run plus no-overwrite the default from the first script, regardless of my own impatience.
3. I would have separated product identity, implementation format, variant and musical function immediately instead of allowing names to carry all four.
4. I would have created a small labelled evaluation set before claiming anything close to 100% accuracy.

PART II

THE SAFE AUDITOR

28 MAY-04 JUNE 2026

The first real system: reports first, mutation last,
evidence over vibes, and no trust granted merely
because a script ran.



ARCHITECTURAL AGE 02

PART 02 / 28 MAY-04 JUNE 2026

THE SAFE AUDITOR

The formal project began when the cleanup scripts became a review-first auditor with explicit safety law. This was the first version that could plausibly be handed to another person without relying on them understanding my entire folder history. It was also the point where the product name, application shape, evidence vocabulary and knowledge lifecycle started to emerge.

28 MAY 2026

The Safe FL Studio Plugin Database Auditor

WHAT CHANGED The first formal artifact was Invoke-FLPluginDatabaseAudit.ps1. By default it produced reports only. It discovered the normal Documents and OneDrive FL Studio Plugin Database roots, inventoried .fst entries, identified duplicates, generated move-plan.csv and move-plan.json, wrote a summary and produced a rollback script. Duplicate candidates went to a run-specific quarantine plan rather than being deleted.

WHY I DID IT The project needed a trustworthy boundary before it needed cleverness. I had already seen how easy it was to confuse presets, binaries, variants and native entries. The first product promise therefore became: tell me exactly what you intend to do before you do it.

HINDSIGHT This was the right foundation. The later mistake was allowing the proof machinery around safety to grow faster than the actual recognition quality and user journey. Safety was necessary; it was never sufficient.

EVIDENCE SNAPSHOT

- Reports first
- Inventory, duplicate and move-plan artifacts
- Rollback script generated
- No delete operation

28 MAY 2026

The mutation contract becomes explicit

WHAT CHANGED Execution required the exact token "I REVIEWED THE FL STUDIO PLUGIN DATABASE MOVE PLAN". Rollback required its own exact token. The system backed up before audit and again before execution, refused overwrite, rejected a move if the source SHA256 changed after planning, required every destination to remain under the database root and limited mutation to .fst entries.

WHY I DID IT The rationale was to make authority visible and testable. A button click or a boolean flag was too easy to trigger accidentally; the plan needed to be reviewed against the exact source state that would be changed.

HINDSIGHT This contract survived every rewrite. The implementation changed from PowerShell to Python and from scripts to a desktop runtime, but immutable planning, source revalidation, journalled execution and rollback remained core. That continuity is evidence that the safety model was not ceremony; it was genuine product law.

EVIDENCE SNAPSHOT

- Execute token and rollback token
- Pre-audit and pre-execution backups
- SHA256 source revalidation
- No overwrite
- Root containment

28 MAY 2026

The master rule table

WHAT CHANGED The first serious ruleset used Priority, Enabled, MatchType, Pattern, DestinationRelative, Vendor and Notes. There were 137 enabled rules: 127 regex and 10 exact, with 122 using a {Vendor} placeholder. Rules were evaluated by ascending priority.

WHY I DID IT The rule table converted scattered decisions into an inspectable, editable source. It was vastly better than hard-coded if/else logic and made review possible without reading the script.

HINDSIGHT The limitation became obvious almost immediately: a rule table is a classification mechanism, not a knowledge system. It cannot establish that a vague token refers to a particular product, resolve a renamed edition, explain a contradiction or distinguish evidence from authority. It remained useful later as a generated cache, but it could not be the brain.

EVIDENCE SNAPSHOT

- 137 enabled rules
- 127 Regex
- 10 Exact
- 122 vendor-templated destinations

28 MAY 2026

Subtype-aware paths replace generic category dumping

WHAT CHANGED The default destination originally looked like Effects\Category\Vendor\Plugin.fst. I corrected it to Effects\Fx-Category\Subtype\Vendor\Plugin.fst, with an equivalent generator structure. RootOne, for example, belonged under low-end and specialty EQ, then Leapwing Audio, not merely under a broad EQ bucket.

WHY I DID IT The point was not aesthetic nesting. Subtypes are what make a large collection navigable. Without them, category roots become new junk drawers.

HINDSIGHT This correction later forced taxonomy-node validation, exact destination reconciliation and profile mapping. It also created a maintenance burden: subtype labels had to be canonical, not merely plausible strings produced by a classifier.

EVIDENCE SNAPSHOT

- Detailed destination shape established
- Category → subtype → vendor → plugin

29 MAY 2026 ARCHITECTURAL PIVOT

Sortceror gets a name and becomes an app concept

WHAT CHANGED I chose Sortceror over the earlier "Sorta" idea and gave it its own application data area. The scope expanded from one audit script into a Windows desktop application supporting Effects, Generators and multiple "New" source roots.

WHY I DID IT A real user should not need to remember command-line modes, CSV locations or tokens scattered across a folder. The interface had to turn the backend into a legible workflow without becoming a second, contradictory sorting engine.

HINDSIGHT Naming the product mattered more than expected. It created pressure for a coherent experience, a visual language and a stable product contract. It also made it easier to over-design screens before the underlying journey was finished - a mistake that would recur in July.

EVIDENCE SNAPSHOT

- Product name fixed: Sortceror
- Windows desktop product direction established

29 MAY 2026

Detailed and Simple profiles

WHAT CHANGED I defined two sort profiles. Detailed preserved the category, subtype, vendor and plugin hierarchy. Simple reduced the structure to broad Simple-EffectType or Simple-GeneratorType roots with vendor folders. The same knowledge needed to translate into more than one user-facing organisation scheme.

WHY I DID IT The profile concept separated what a plugin is from how a particular user wants it displayed. That was an important architectural distinction: classification should not be hard-wired to one folder tree.

HINDSIGHT The first implementation still allowed profile mappings and taxonomy labels to drift. Later normalization work was required because a "profile" is only reliable when every target node is canonical and conflicts are surfaced rather than silently mapped.

EVIDENCE SNAPSHOT

- Detailed and Simple profile contracts

29 MAY 2026

The GUI is declared a wrapper, never a second brain

WHAT CHANGED The first screen map included Dashboard, Scan New, Dry Run Sort, Move Plan Preview, Execute Reviewed Plan, Knowledge Base, Review Queue, Profiles, Rollback, Settings and an Advanced/Behind the Curtain area. Advanced mode exposed the backend command, run folder, move plan, proposed knowledge, research evidence, logs and rollback artifacts.

WHY I DID IT The interface needed to make the evidence and safety state understandable, not hide it. I explicitly separated enrichment from execution and required the GUI to call the same backend used by tests and scripts.

HINDSIGHT This rule was correct but not consistently enforced. Later prototypes had screens that looked real while some buttons had no live service behind them. The July product-spine correction was essentially the expensive enforcement of this May decision.

EVIDENCE SNAPSHOT

- Single backend principle stated
- Enrichment separated from execution
- Advanced evidence visibility planned

29 MAY 2026

Rule-source labels and the first provenance model

WHAT CHANGED Every decision needed a source label: Knowledge Base, Alias Match, Exact Rule, Vendor Rule, Generic Classifier, Internet Evidence, Manual Review, Unresolved or Conflict. For unfamiliar products, enrichment proposals needed identity, vendor, root type, subtype, destination, confidence, sources, evidence summary, date, aliases and ambiguity notes.

WHY I DID IT I wanted the product to answer "why" rather than merely output a folder. Provenance was also necessary to stop low-trust enrichment from silently becoming move authority.

HINDSIGHT This was the right direction, but confidence was still too loosely defined. A percentage without calibration, independence groups and contradiction handling is decoration. Later audits repeatedly caught "confidence" values that looked scientific without carrying enough epistemic meaning.

EVIDENCE SNAPSHOT

- Decision-source vocabulary introduced

- Proposed knowledge required source and ambiguity fields

29 MAY 2026

Starter seeds expose the difference between volume and authority

WHAT CHANGED Starter work produced proposed effect and generator knowledge, aliases and profile mappings. Recorded dry-run figures included 100 effect knowledge rows, 111 generator rows, 6 aliases, 85 profile-mapping conflicts and one duplicate skipped. A forensic check then found that the active backend did not yet contain every expected seeding or health mode.

WHY I DID IT The purpose was to give the system enough initial coverage to be useful. The audit was necessary because generated files and design documents were beginning to outrun the code that actually consumed them.

HINDSIGHT This became a recurring failure mode: artifact production can look like progress while authority remains unchanged. I eventually learned to ask a harsher question: what new correct decision can the live product make today that it could not make yesterday?

EVIDENCE SNAPSHOT

- 100 proposed effect rows
- 111 proposed generator rows
- 6 aliases
- 85 profile conflicts
- 1 duplicate skipped

02 JUNE 2026 ARCHITECTURAL PIVOT

The external corpus appears

WHAT CHANGED The project gained modes for building an external plugin corpus and generating proposed knowledge from it, backed by tables for source manifests, products, evidence, fetch logs, product aliases and vendor aliases. The corpus was explicitly non-authoritative: it could produce evidence and proposals but could not mutate active knowledge or move files.

WHY I DID IT The goal was to stop relying on a tiny handwritten rule set. A broad product corpus could reduce unknowns, especially for plugins not installed locally or named ambiguously.

HINDSIGHT The non-authoritative boundary was correct. The early corpus was still mostly a safe skeleton. Breadth without source quality, role-specific claims and conflict handling did not create human-grade understanding.

EVIDENCE SNAPSHOT

- External corpus tables and modes added
- Corpus prohibited from direct knowledge or move authority

02 JUNE 2026

Evidence roles become explicit

WHAT CHANGED Evidence was split into IdentityEvidence, VendorEvidence, RootTypeEvidence, FunctionEvidence, SubtypeEvidence and FormatEvidence. Evidence sets could establish consensus, generated rules carried provenance and staleness, and move-plan rows became more traceable.

WHY I DID IT Different claims require different proof. A vendor page may establish identity and function but say nothing reliable about the user's preferred taxonomy. A local binary may establish implementation and version but not musical use. Treating every source as interchangeable was no longer acceptable.

HINDSIGHT This was one of the strongest architectural decisions in the whole project. The remaining flaw was that some proposed rows still leaned on selected candidate fields without preserving a clean source-to-role chain. The later evidence graph fixed that structurally.

EVIDENCE SNAPSHOT

- Six evidence roles
- Evidence-set consensus
- Provenance and staleness tracking

03 JUNE 2026 **ARCHITECTURAL PIVOT**

Live enrichment fails honestly

WHAT CHANGED Three probe products were used to test enrichment. Rift 2 was confidently assigned the wrong subtype: saturation and harmonic colour instead of creative multi-mode destruction. MODO BASS 2 remained unresolved. The Sauce stayed review-only, which was the correct outcome.

WHY I DID IT The test was designed to answer whether the system could safely graduate vague external evidence into usable knowledge.

HINDSIGHT The answer was no, and that was useful. Rift 2 proved that confidence can be wrong; MODO BASS proved that missing evidence must remain missing; The Sauce proved that abstention is a successful result. This is where fail-closed behaviour stopped being a safety slogan and became an intelligence requirement.

EVIDENCE SNAPSHOT

- 3 live probes
- 0 import-eligible
- 1 confident subtype error
- 1 unresolved
- 1 correctly review-only

03-04 JUNE 2026

The Review Queue becomes the learning loop

WHAT CHANGED The workflow crystallised into Update Knowledge → Review Knowledge → Import Knowledge → Execute. ReviewQueue lifecycle repair, manual seed import and template export were added, alongside vendor and product registry structures and the beginnings of a Python enrichment brain.

WHY I DID IT Corrections needed to change future behaviour rather than disappear into one-off move plans. The queue was intended to turn human intervention into durable knowledge, aliases and evidence.

HINDSIGHT The key idea was right. The first implementation still made the user review too much and did not yet prove that imported corrections flowed through the active resolver. Review became a product feature before review compression became an intelligence objective.

EVIDENCE SNAPSHOT

- Review lifecycle formalised
- Manual seed import/export
- Python brain skeleton and early tests

04 JUNE 2026 ARCHITECTURAL PIVOT

The first mixed-language architecture

WHAT CHANGED At this point the intended split was: PowerShell for execution, backup, rollback, packaging and hard safety; SQLite as the canonical data target; Python for corpus, enrichment, evidence and proposals; CSV only for import, export and review. The rule was already that the knowledge base was canonical and generated rules were disposable.

WHY I DID IT This was a staged migration rather than a rewrite. The existing executor was proven and the new intelligence work was easier to express in Python.

HINDSIGHT The split was sensible as a transition and poor as a destination. Too much behaviour remained duplicated across languages and files. Within a week, the project would make the cleaner decision: Python owns the application; PowerShell becomes a thin shell and then legacy debt.

EVIDENCE SNAPSHOT

- SQLite named as canonical target
- Python assigned intelligence work
- PowerShell retained execution/safety temporarily

PART 2 RETROSPECTIVE

What this phase actually changed

This phase created the project's moral centre: evidence is not authority, proposals are not knowledge, knowledge is not move permission and a successful process is not proof of a correct decision. It also created the first major source of bloat. Every new uncertainty produced another CSV, report, mode or gate. The system became safer and more inspectable, but the product risked becoming an evidence factory that made the user do the thinking.

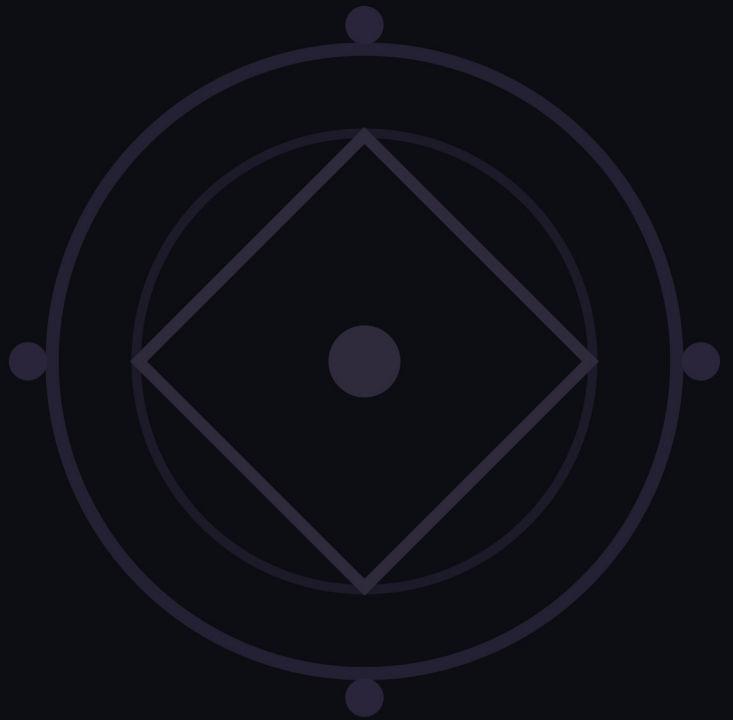
WHAT I WOULD DO DIFFERENTLY

1. I would have defined the authority hierarchy as a first-class schema earlier instead of spreading it across modes and files.
2. I would have built a tiny end-to-end product slice before designing the full screen map.
3. I would have treated calibration and abstention as intelligence features from the first enrichment experiment.
4. I would have measured active knowledge imported and future asks eliminated, not just proposals generated.

THE LOCAL EVIDENCE PIVOT

08–13 JUNE 2026

The project stopped pretending a rule table could understand a plugin and started building a local machine-evidence graph.



ARCHITECTURAL AGE 03

PART 03 / 08-13 JUNE 2026

THE LOCAL EVIDENCE PIVOT

External research was useful, but the machine already contained a huge amount of safer evidence: plugin binaries, VST3 module metadata, Windows version resources, installation paths, registry records, sidecars, hashes and the user's existing sorted corpus. The next pivot was to stop asking the internet questions the local system could answer first.

08 JUNE 2026

Alias repair and the first real 157-file scan

WHAT CHANGED Ambiguous exact aliases were cleared and the active scan was tested against 157 files. Stop-after-scan completed in roughly 1.56 to 2.01 seconds; capped 20-file runs took roughly 3.6 to 3.9 seconds. Live enrichment validation, research-batch export, cache-repair dry run and proposal generation all reported success.

WHY I DID IT The point was to move from synthetic fixtures to the actual unsorted set and establish that the pipeline could at least observe it quickly and repeatably.

HINDSIGHT One review-pack artifact was later found invalid because a fallback runner had produced an untruthful result. That mattered: the proof layer itself can lie. From then on, provenance and identity had to apply to validation artifacts as much as to plugin evidence.

EVIDENCE SNAPSHOT

- 157 real files
- Scan-only: ~1.56-2.01 s
- 20-file capped run: ~3.6-3.9 s

09 JUNE 2026

The FL Studio reality model changes

WHAT CHANGED I confirmed that unsorted .fst entries do not live only under Installed\Effects\New and Installed\Generators\New. FL Studio can auto-transfer entries into loose folders across Installed, Effects and Generators. Favourites needed exclusion, sidecars needed preservation, and the scanner had to distinguish approved organised roots from merely existing paths.

WHY I DID IT A sorter that scans only the obvious New folders will miss the exact mess it claims to fix. The source model had to reflect what FL actually does, not what the folder names imply.

HINDSIGHT This was an important move from path assumptions to an explicit reality model. It also taught me that "exists on disk" is not the same as "valid source" or "valid taxonomy node" - a distinction that later prevented empty approved folders from being deleted as junk.

EVIDENCE SNAPSHOT

- Source coverage expanded across Installed, Effects and Generators
- Favourites excluded
- Sidecars preserved

10 JUNE 2026 ARCHITECTURAL PIVOT

A six-step local intelligence roadmap

WHAT CHANGED The new plan was explicit: model FL reality, parse VST3 moduleinfo.json, extract static binary metadata, resolve .fst entries to binaries, store the evidence in canonical SQLite and add ML-ready ranking without

granting ML authority. VST3 class metadata could expose names, vendors, versions, categories and compatibility without loading the plugin. CLAP descriptors offered a similar route.

WHY I DID IT The aim was to reduce questions by using deterministic, local and static evidence before external enrichment. Loading arbitrary third-party code inside the main process was deliberately out of bounds.

HINDSIGHT This roadmap was the first genuinely scalable intelligence plan. The phrase "ML-ready, not ML authority" also survived. The future system could use learned models to rank and retrieve while still requiring structured evidence and policy gates for consequential actions.

EVIDENCE SNAPSHOT

- VST3 moduleinfo parser planned
- Static PE/binary evidence
- FST-to-binary resolver
- SQLite evidence graph
- ML ranking without authority

11 JUNE 2026

The first 80 manual seed

WHAT CHANGED I prepared the first 80 rows of the 157-plugin set as a manual research seed. The schema captured canonical name, plugin key, product identity, vendor, root type, approved root, subtype, destination and explicit authority flags. All 80 rows were forced to CanAutoMove FALSE, NeedsReview TRUE and HumanReviewRequired TRUE. Preparation yielded 80 rows, 80 documents and 480 claim drafts with zero active mutation.

WHY I DID IT The seed was intended to create reviewed gold data safely. Taxonomy validation and conflict detection were added so the import could not smuggle invalid destinations into active memory.

HINDSIGHT Technically this was solid. Strategically it exposed a wall: I did not know every plugin well enough to hand-author its truth, and a product designed to eliminate manual sorting could not begin by asking me to research every unknown product. The seed became valuable benchmark material, but not the main scaling strategy.

EVIDENCE SNAPSHOT

- 80 rows
- 80 documents
- 480 claims
- 0 active mutation
- 0 CanAutoMove

11 JUNE 2026 ARCHITECTURAL PIVOT

Ask reduction is not ask elimination

WHAT CHANGED The manual seed workflow produced review templates, case files and an ask-reduction estimate. That wording triggered a more important product question: why was the goal merely to reduce my work? The product needed to acquire evidence, rank hypotheses and abstain intelligently so that I only saw genuinely irreducible exceptions.

WHY I DID IT At the time, manual research looked like the safest path to gold knowledge. It was safe, but it shifted the labour back onto the user.

HINDSIGHT This was the first moment I recognised that review volume is itself a product defect. Human review should be an exception lane and a learning signal, not the normal operating mode.

EVIDENCE SNAPSHOT

- Manual research strategy rejected as primary scaling path

12 JUNE 2026 ARCHITECTURAL PIVOT

The major Python and SQLite cutover

WHAT CHANGED I made the clean architecture decision. Python would own scan, parse, classify, evidence, SQLite and move-plan generation. SQLite would be canonical memory. PowerShell would be limited to shell-level calls, backup, move, rollback, packaging and gates. CSV would exist only as a review/export/import/audit artifact. Generated rules would be disposable. The GUI would be a cockpit over the same application spine.

WHY I DID IT The mixed architecture had become harder to reason about than the product. Python offered better parsing, testing, data modelling and future ML integration; SQLite offered transactional state, relationships and auditable provenance.

HINDSIGHT This was the most important technical pivot in the project. I should have made it earlier, but the earlier PowerShell work was not wasted: it supplied the safety contract and executable requirements that the Python system had to preserve.

EVIDENCE SNAPSHOT

CHECKPOINT 01c165d / reviewed seed active import preparation

- Branch: data-driven plugin intelligence
- Python owns product logic
- SQLite canonical
- CSV non-authoritative
- PowerShell demoted

12 JUNE 2026

Reviewed import and rollback safety are re-proven

WHAT CHANGED Fixture ActiveExecute and rollback were validated under the new architecture. The recorded gate was 50 PASS, 0 FAIL. The real Plugin Database remained untouched, no active generated rules were introduced and no CanAutoMove rows were created.

WHY I DID IT A rewrite is not allowed to wave away prior safety. The new spine had to prove that authority, backup and rollback semantics survived the language change.

HINDSIGHT This was a good example of targeted regression proof. Later the project temporarily forgot this economy and began rerunning enormous suites for unrelated changes. The eventual validation-economy doctrine was a return to this more disciplined idea.

EVIDENCE SNAPSHOT

CHECKPOINT 40a655c / reviewed import and rollback safety

- 50 PASS / 0 FAIL
- Real active mutation count: 0
- CanAutoMove rows created: 0

13 JUNE 2026

The local install intelligence graph explodes in scale

WHAT CHANGED The local-first scan expanded across linked installs, Program Files, ProgramData, AppData, Common Files, public documents and the FL database. The resulting LocalInstallIntelligence.sqlite was about 1.377 GB and passed integrity and foreign-key checks. It contained 639,799 file observations, 1,628,480 identifier candidates, 237 VST3 module observations, 707 VST3 class observations, 769 uninstall records, 2,477 existing sorted labels and 7,399 candidate identities or destinations linked by 68,998 evidence edges.

WHY I DID IT The objective was to turn the machine into its own evidence corpus. Existing sorted placements could become weak training labels; binaries and registry records could help resolve products; module metadata could reduce name ambiguity.

HINDSIGHT The scale was impressive and misleading. The graph also contained taxonomy leakage, invalid candidate roots and evaluation leakage. It lacked manual/document parsing and stronger product identity rules. This became the project mantra: scale is not intelligence. A million rows of contaminated evidence are just a larger pile of bullshit.

EVIDENCE SNAPSHOT

- SQLite ~1.377 GB
- 639,799 file observations
- 1,628,480 identifier candidates
- 707 VST3 classes
- 2,477 sorted labels
- 68,998 evidence links
- 128 pytest + 176 Pester passed

13 JUNE 2026

Environment truth becomes part of product truth

WHAT CHANGED The project gained a stricter development-environment verifier and clean checkpoints for local observation and the intelligence graph. Python interpreters, virtual environments, compilers and required tools had to be identified by exact path and version rather than assumed from PATH.

WHY I DID IT Reproducible evidence is impossible when the runtime producing it is ambiguous. The same command can mean different packages, tests or compilers depending on the shell.

HINDSIGHT This looked like internal hygiene, but it later became directly relevant to packaging and detached clean-room builds. The weakness was that environment verification also became another place where the project could over-invest compared with the user journey.

EVIDENCE SNAPSHOT

- Local observation checkpoint
- Clean local-intelligence-graph checkpoint
- Integrity and safety counters green

PART 3 RETROSPECTIVE

What this phase actually changed

The local evidence pivot changed the project from a sorter with a knowledge file into a data system. The right mental model became: observe the machine, preserve claims separately, resolve entities probabilistically, translate conclusions through a canonical taxonomy and only then consider authority. The wrong move was to treat graph size and test count as proxies for recognition quality. They were not.

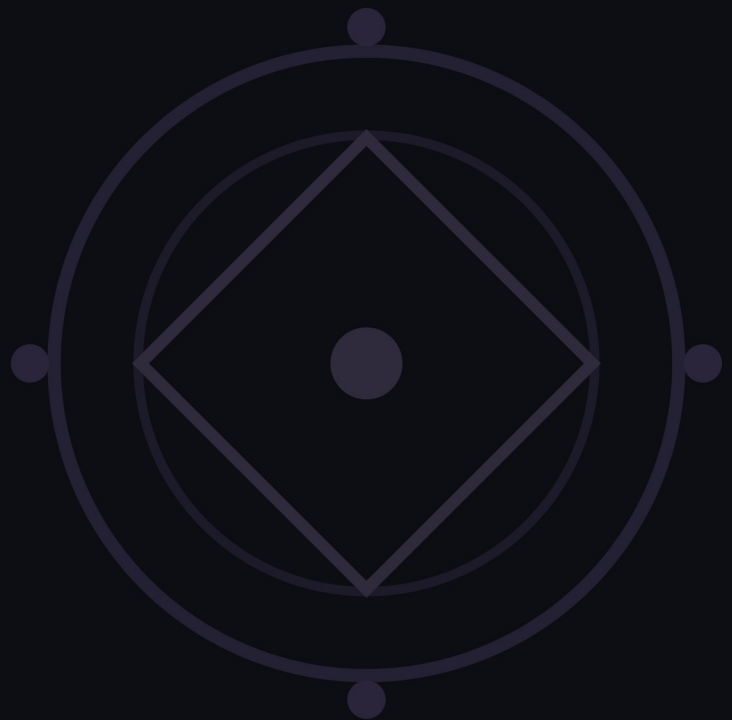
WHAT I WOULD DO DIFFERENTLY

1. I would have designed leakage-resistant evaluation before ingesting the existing sorted corpus as labels.
2. I would have capped scan scope and evidence retention earlier so the graph could be iterated quickly.
3. I would have prioritised identity-resolution precision over collecting every possible file observation.
4. I would have connected each evidence adapter to a measurable product decision instead of celebrating row counts.

THE REWRITE AND CORPUS-TRUTH ERA

15–26 JUNE 2026

PowerShell became legacy, SQLite became memory, Python became the application, and historical labels were finally put on trial.



ARCHITECTURAL AGE 04

PART 04 / 15-26 JUNE 2026

THE REWRITE AND CORPUS-TRUTH ERA

The next eleven days were less glamorous and probably more important. The repository had to stop carrying every historical mechanism forever. Validation had to become cheaper, artifacts had to stop eating the disk, and the existing sorted corpus had to be audited before it could be treated as training truth. This was the move from "data-driven" in name to actual authority discipline.

15 JUNE 2026

The environment verifier stops pretending

WHAT CHANGED The verifier was corrected to distinguish PATH Python from the required interpreter, treat missing global pytest as a warning when the project virtual environment was healthy, require the exact virtual-environment Python, pytest and compileall, and verify 7-Zip through the real executable path.

WHY I DID IT The goal was simple: report the environment that actually runs the project, not an idealised machine state.

HINDSIGHT This was a good correction and a warning about validators. A gate that fails for irrelevant reasons teaches people to ignore gates. A gate that passes on the wrong interpreter is worse. Validation has to be both strict and relevant.

EVIDENCE SNAPSHOT

- Interpreter and tool paths made explicit
- Irrelevant global-package failure downgraded

16 JUNE 2026 ARCHITECTURAL PIVOT

ThinDelta replaces ten-gigabyte review packs

WHAT CHANGED Review packs had grown beyond 10 GB. I introduced ThinDelta as the default, added a FastDevGate, stored large artifacts by reference and tightened checkpoint staging. A representative thin pack was about 76.3 MB with five large external references and roughly 6.89 GB excluded from inline packaging. The fast development gate completed in 48.434 seconds; the full thin-delta live pack reported 58 PASS, 0 FAIL.

WHY I DID IT The evidence system had become too expensive to review, copy and regenerate. The product needed identity-addressed evidence rather than repeatedly zipping the world.

HINDSIGHT This was the birth of validation economy, though I did not yet apply it consistently. The lesson is obvious now: proofs should be reusable by exact identity. Rebuilding unchanged evidence is not rigor; it is waste.

EVIDENCE SNAPSHOT

- Thin pack ~76.3 MB
- ~6.89 GB excluded inline
- Fast gate 48.434 s
- 58 PASS / 0 FAIL

17 JUNE 2026 ARCHITECTURAL PIVOT

The 81 GB artifact-hygiene incident

WHAT CHANGED The repository ballooned to roughly 82 GB, with about 81.3 GB under temporary review packs and the artifact store. Old live-gate packs and duplicated evidence were the main offenders.

WHY I DID IT The project was preserving everything because deletion felt risky. That instinct produced a different operational risk: the development environment became slow, expensive and nearly unusable.

HINDSIGHT This should have been prevented by retention law from day one. Evidence needs manifests, hashes, lineage and retention classes - not infinite duplication. The incident forced a cleaner rule: preserve accepted identities, discard reproducible intermediates and never confuse "more files" with "more proof".

EVIDENCE SNAPSHOT

- Repository ~82 GB
- Temporary artifacts ~81.3 GB
- Retention and manifest law tightened

18 JUNE 2026

Exact taxonomy and the corpus ladder

WHAT CHANGED A large taxonomy audit scanned 5,891 rows: 2,031 were approved, 3,860 rejected, 18 valid controller placements identified, 92 invalid-folder rows found and 34 overly broad folders rejected. The corpus ladder then narrowed 5,891 observed rows to 2,031 taxonomy-valid, 1,953 conflict-free and 1,834 candidates.

WHY I DID IT Before using historical placements as intelligence, I needed to know which labels were actually valid under the current taxonomy. Existing organisation was evidence, not automatic truth.

HINDSIGHT This was the correct move and should have happened before the first large evidence graph. It exposed that the user's own corpus contains legacy decisions, obsolete nodes and convenience placements. Gold data is made, not inherited.

EVIDENCE SNAPSHOT

- 5,891 observed
- 2,031 approved
- 3,860 rejected
- 1,953 conflict-free
- 1,834 candidates

18 JUNE 2026 ARCHITECTURAL PIVOT

A huge staging pipeline imports zero authority

WHAT CHANGED The project could generate a 157-row move-and-learn packet, a 500-row sorted corpus packet and 237 generated rules with no blank or invalid destinations. Reviewed-import SQLite contained 894 plan and 894 outcome rows with clean integrity. Yet authoritative knowledge imported remained zero. Measured production asks were 5,578; the forecast after approval was 894; measured reduction was still zero.

WHY I DID IT Every safety barrier had done its job: nothing unreviewed became authority. But the pipeline was now visibly stalled. It was excellent at preparing decisions and poor at graduating correct ones.

HINDSIGHT This is one of the most important failures in the timeline. I had built a bureaucracy around learning. The system was safe, but safety scaffolding had become a substitute for shipping intelligence. The answer was not to remove authority gates; it was to build better evidence, evaluation and controlled promotion.

EVIDENCE SNAPSHOT

- 894 plan rows
- 894 outcome rows
- 0 imported authoritative knowledge
- 5,578 measured asks
- 0 measured ask reduction

18 JUNE 2026 ARCHITECTURAL PIVOT

The strangler rewrite decision

WHAT CHANGED I considered restarting. The actual decision was more disciplined: treat the first system as a requirements mine, test oracle, data source and cautionary tale, then build the new Python-native product beside it and retire old behaviour in slices.

WHY I DID IT A clean rewrite would have thrown away hard-won safety semantics and fixtures. Continuing to patch the old spine would preserve duplication and confusion.

HINDSIGHT The strangler approach was right. The mistake was not declaring retirement criteria aggressively enough. Legacy modes remained visible in tests and documentation long after they stopped representing the product, which later created stale-contract failures.

EVIDENCE SNAPSHOT

- V1 retained as requirements/test/data source
- V2 built beside legacy runtime

19-20 JUNE 2026

The V2 environment and Python-native law

WHAT CHANGED The development environment was standardised around Python 3.12, with analytics, retrieval, document, binary and VST-resource tooling available. DuckDB, Polars, PyArrow, sentence-transformers, vector stores, entity-resolution libraries, graph tools, PDF/document parsers, PE parsing and image utilities were verified. A side environment covered specialised deduplication libraries. CI moved to Python 3.12 and editable installs.

WHY I DID IT The objective was to make the architecture capable of evidence extraction, analytics, retrieval and learned ranking without improvising dependencies during each pass.

HINDSIGHT This platform work created real optionality. The caution is that installed libraries are not implemented intelligence. Later audits correctly refused to credit a capability merely because the dependency existed.

EVIDENCE SNAPSHOT

- Python 3.12 environment
- Analytics/retrieval/binary/document stacks verified
- CI green

20 JUNE 2026 ARCHITECTURAL PIVOT

Python is Sortceror

WHAT CHANGED The V2 doctrine was written plainly: Python is the application. PowerShell is legacy debt and receives no new behaviour. SQLite is canonical authority and memory. DuckDB and Polars are analytical tools. SQLite FTS5 provides deterministic retrieval. Embeddings and vectors can suggest, not decide. Weak supervision, entity resolution and label-quality tools produce review signals, not authority.

WHY I DID IT The architecture needed one centre of gravity. Every duplicate implementation multiplied the chance that the GUI, tests and executor would disagree.

HINDSIGHT This law held. The later product-spine and intelligence passes were refinements of it, not reversals. It is probably the clearest example of the project getting a foundational decision exactly right.

EVIDENCE SNAPSHOT

- Stable checkpoint: 98931bb
- Python-native law
- PowerShell decommissioning
- CI green

20 JUNE 2026

The authority hierarchy is formalised

WHAT CHANGED The project separated DISCOVERY, INTELLIGENCE, KNOWLEDGE AUTHORITY, MOVE AUTHORITY and EXECUTION AUTHORITY. CorpusValidatedClassification could establish knowledge; ReviewedMoveDecision could establish movement; execution still required its own token and live source validation.

WHY I DID IT The same evidence can be useful at one layer and insufficient at another. An embedding match may help discovery but should never authorise a file move.

HINDSIGHT This hierarchy became the backbone of every later fail-closed decision. It also made honest product messaging possible: the interface could say what the system knows, what it suspects and what it is permitted to do.

EVIDENCE SNAPSHOT

- Five authority layers made explicit

22 JUNE 2026 ARCHITECTURAL PIVOT

Human-on-the-loop replaces review-everything

WHAT CHANGED I corrected the autonomy model. Chase should not review every proposed decision. The product should act autonomously where deterministic authority exists, request exception review where uncertainty remains, require an explicit token where files mutate and use sample auditing to calibrate trust.

WHY I DID IT Review-everything was safe but not a product. Full autonomy without abstention was reckless. The middle path was selective autonomy with visible authority.

HINDSIGHT This is still the correct target. The remaining challenge is earning a large enough autonomous coverage set without contaminating evaluation or hiding false positives.

EVIDENCE SNAPSHOT

- Autonomy model: act, abstain, token-gate, sample-audit

23 JUNE 2026

The V3 corpus-truth sequence is locked

WHAT CHANGED The next sequence was deliberately ordered: Corpus Audit and Trust-Tier Snapshot → Taxonomy Node Normalization → 157 Move Plan Reconciliation → Evidence Text Builder → structured enrichment and embeddings → Hybrid Resolver → simulated New-folder replay → active review packets → reviewed import and safe movement.

WHY I DID IT The previous pipeline had tried to enrich before fully validating the labels and taxonomy it would learn from. The new sequence put truth preparation first.

HINDSIGHT This sequencing was excellent. It converted a vague "make it smarter" goal into a dependency graph. The later temptation was to add more intelligence mechanisms before the product spine was complete; the roadmap itself was not the problem.

EVIDENCE SNAPSHOT

- Stable baseline: c4311ae
- V3 corpus-truth branch established

25 JUNE 2026

Corpus Audit v1

WHAT CHANGED A deterministic corpus-audit package wrote proof artifacts into SQLite, included no-mutation evidence and added 20 focused tests. The full suite reached 476 passed. The audit archive had its own identity.

WHY I DID IT The goal was to grade historical labels by trust tier rather than treating every sorted entry as gold.

HINDSIGHT This was a small but structurally important milestone. It established that training data and authority data need explicit lineage, not just a folder path that happens to look organised.

EVIDENCE SNAPSHOT

- 20 new tests
- 476 full-suite passes
- No-mutation proof

25 JUNE 2026

Taxonomy Node Normalization v1

WHAT CHANGED Canonical taxonomy nodes were normalised and checkpointed. Legacy spellings, broad folders and invalid roots could no longer pass as equivalent merely because their text looked close.

WHY I DID IT The resolver, review tools and move planner all needed one destination vocabulary. Without normalization, two correct-looking components could still disagree on the actual path.

HINDSIGHT This removed an entire class of hidden mismatch. It also demonstrated why taxonomy belongs in canonical data, not duplicated constants across scripts and UI.

EVIDENCE SNAPSHOT

- Checkpoint: a7a47bf
- Canonical node normalization

26 JUNE 2026

The 157 Move Plan Reconciliation

WHAT CHANGED The reconciliation package added SQLite tables for runs, source rows, matches, destination validation, reconciliation items, review seed groups and safety counters. A 12-row fixture produced 10 matches: 8 exact, 1 weak and 1 ambiguous; one row was unmatched. Ten destinations were valid, two invalid; one promotion candidate emerged; six were blocked; two needed review; one was ready for evidence text. All safety counters remained zero and no network calls were made.

WHY I DID IT The purpose was to compare the legacy 157-row plan against canonical evidence and taxonomy rather than importing it wholesale.

HINDSIGHT This was the practical point where old work became auditable input to the new system. It also showed why reconciliation is not binary: exact, weak, ambiguous, unmatched, valid, blocked and promotable are distinct states with different authority consequences.

EVIDENCE SNAPSHOT

- 12-row proof fixture
- 10 matched
- 8 exact
- 1 weak
- 1 ambiguous
- 1 unmatched
- 2 invalid destinations

- 0 safety violations

26 JUNE 2026

Evidence Text Builder v1

WHAT CHANGED Role-separated observations were converted into reproducible evidence text and provenance packets before external enrichment. The builder created a stable bridge between structured claims and retrieval or review surfaces.

WHY I DID IT Embeddings and external research work better when the input is explicit about identity, function, source and contradiction rather than a concatenated plugin name.

HINDSIGHT This was a necessary enabling layer, but it was still infrastructure. Its value depended on being connected to the active resolver and measured against held-out decisions - a standard I would apply more aggressively in the later intelligence passes.

EVIDENCE SNAPSHOT

- Evidence text/provenance ledger established

PART 4 RETROSPECTIVE

What this phase actually changed

By the end of June, Sortceror had stopped being a collection of scripts and become an architecture with laws. The project knew what counted as evidence, which data was canonical, how authority graduated and why historical labels were untrusted. It also learned a brutal operational lesson: safe staging can still be a dead end. A pipeline that never promotes correct knowledge is not cautious intelligence; it is a filing cabinet.

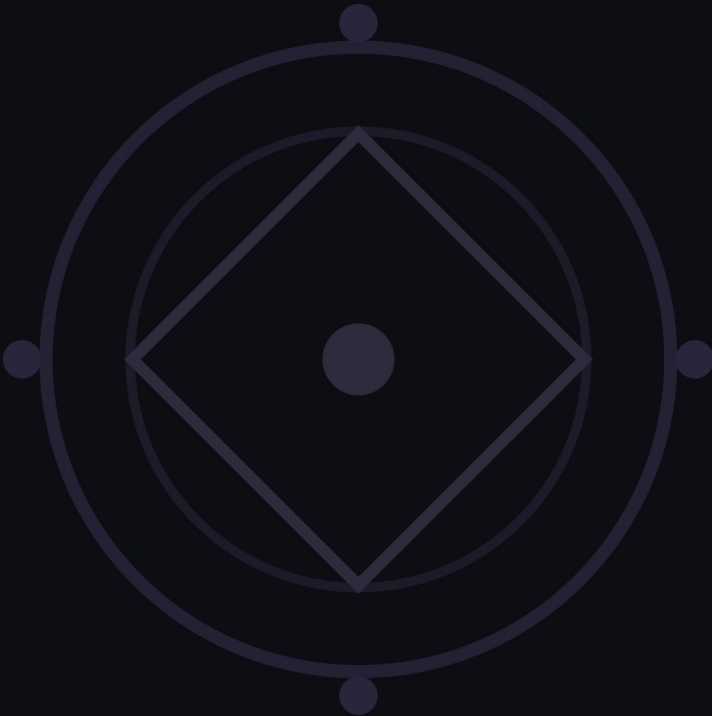
WHAT I WOULD DO DIFFERENTLY

1. I would have defined retirement dates for every legacy mode and test.
2. I would have set a hard budget for artifact volume and validation time before the repository reached 82 GB.
3. I would have required every infrastructure milestone to name the exact active-product decision it unlocked.
4. I would have measured autonomous coverage, false-positive rate and review minutes from the start of V2.

SAFE EXECUTION BECOMES A PRODUCT

05–12 JULY 2026

The 157-plugin plan moved from theory to controlled authority, then the intelligence spine and desktop cockpit began to converge.



ARCHITECTURAL AGE 05

SAFE EXECUTION BECOMES A PRODUCT

July began with the first controlled live movement of the 157-plugin plan, then accelerated into local intelligence, every-plugin accounting, a real desktop shell and packaged executables. This was the phase where Sortceror finally stopped being only an internal analysis system and started becoming something a producer could launch.

05 JULY 2026

The reviewed 157 authority-gated planner

WHAT CHANGED A dedicated planner implemented plan, execute and rollback with backup manifests, rollback manifests, sidecar and collision gates, SQLite source validation and an exact approval token. CSV, raw JSON and manual seed files were explicitly blocked as live movement authority.

WHY I DID IT The 157-row set had been studied for weeks. The goal was to convert that work into a real movement path without allowing convenient artifacts to bypass canonical review.

HINDSIGHT This was the correct culmination of the authority hierarchy. It proved that safety did not have to mean permanent simulation. The hard part was not writing `shutil.move`; it was proving that the row being moved was the reviewed row, from the reviewed source state, into a valid destination, with a reversible journal.

EVIDENCE SNAPSHOT

CHECKPOINT 5f1e4e8 / reviewed 157 sort authority-gated planner

- Exact execute token
- SQLite-only move authority
- Sidecar/collision/source validation

05 JULY 2026

The canonical ledger initially blocks the plan

WHAT CHANGED The first plan run correctly refused to proceed because no canonical reviewed ledger existed. Reviewed157Sort.sqlite was then built and validated: one plan run, 157 move-plan rows and 157 operations; 148 were eligible, 9 blocked, with zero execution, backup or rollback events at that point.

WHY I DID IT The token was necessary but deliberately insufficient. Authority had to exist in the database before a human phrase could activate it.

HINDSIGHT This is one of the best examples of fail-closed design working exactly as intended. The system did not reward effort or assume that a familiar CSV was trustworthy. It demanded the missing authority record.

EVIDENCE SNAPSHOT

- 157 rows
- 148 eligible
- 9 blocked
- 0 execution events
- SQLite integrity OK

05 JULY 2026

Medium-risk approvals and the 154-move execution

WHAT CHANGED Six medium-risk products - Console 1, REF SEND, Rolling Sampler, Scaler 3 Audio, Tectonic and ZRev - received explicit approval. After re-planning, 154 rows were eligible and 3 remained blocked: two ambiguous matches and one missing source. I executed the 154 planned moves and verified destination files and hashes.

WHY I DID IT The goal was to move everything that had earned authority while preserving honest blocks for unresolved rows.

HINDSIGHT This was the first substantial real-world proof that the safety model could ship action rather than only reports. It also reinforced that "almost all" is a valid success state when the remaining rows are genuinely ambiguous. Forcing 157/157 would have made the product less trustworthy.

EVIDENCE SNAPSHOT

- 154 executed
- 3 blocked
- 6 explicit medium-risk approvals
- Post-execution hash verification

05 JULY 2026

Empty is not invalid

WHAT CHANGED Post-move cleanup revealed another subtle rule: an empty folder can be an approved taxonomy bucket. Cleanup therefore had to delete known legacy junk, not every empty directory.

WHY I DID IT The earlier cleanup scripts treated residue as clutter. The product now needed schema-aware cleanup that understood the difference between an empty valid node and an obsolete path.

HINDSIGHT This tiny correction captures a broader principle: filesystem shape is not self-explanatory. Meaning comes from canonical taxonomy and history, not from whether a directory currently contains a file.

EVIDENCE SNAPSHOT

- Approved empty taxonomy nodes preserved

06-07 JULY 2026

Local Resource Intelligence and ML Entity Resolution v1

WHAT CHANGED The local-only intelligence pass scanned .fst files, VST3, VST2, CLAP and AAX binaries, sidecars, module metadata, PE resources, strings, install paths and local documents. It resolved products from machine evidence first and kept all learned or semantic outputs non-authoritative. Across ten replay modes, false-confident errors were recorded as zero; the external-research queue fell from 258 to 219.

WHY I DID IT The objective was to recognise more products without network calls and without loading third-party plugins in the main process.

HINDSIGHT This was a meaningful intelligence gain, but the replay set was still close to the known corpus. Zero false-confident errors in a constrained benchmark did not prove open-world intelligence. The later autonomy audit was right to revoke stronger claims.

EVIDENCE SNAPSHOT

- External queue: 258 → 219
- 10 replay modes
- 0 false-confident errors
- 364 tests passed

07 JULY 2026

All 573 Sort-New candidates receive a disposition

WHAT CHANGED The Sort-New plan expanded to 573 candidates. A representative controlled dry run classified 19 as ready, 255 as review and 299 as blocked. Proposal rows carried predicted identity, vendor, format, taxonomy path, candidate alternatives, evidence, weights, contradictions, confidence and authority status.

WHY I DID IT The goal was complete accounting. Every object needed a reasoned state even when it could not move.

HINDSIGHT This was good product truth: blocked and review-only rows were first-class outcomes. The weakness was still coverage. A system that safely blocks most items may be trustworthy, but it is not yet autonomous enough to be valuable at scale.

EVIDENCE SNAPSHOT

- 573 candidates
- 19 ready
- 255 review
- 299 blocked

09 JULY 2026

The read-only cockpit reaches operability

WHAT CHANGED A read-only cockpit exposed the intelligence results. Forty-eight items were sort-ready, grouped into 46 decisions, and 18/18 GUI parity checks passed. The full suite reached 522 tests with hard safety counters at zero.

WHY I DID IT The interface needed to prove that the backend's decisions could be rendered consistently and navigated as a product workflow.

HINDSIGHT The cockpit was useful, but it could still become an attractive operator console rather than a consumer product. The next necessary step was review-decision import and execution through the same bridge - not more dashboards.

EVIDENCE SNAPSHOT

- 48 sort-ready
- 46 grouped decisions
- 18/18 GUI parity
- 522 tests

10 JULY 2026

Every plugin-like object is accounted for

WHAT CHANGED The readiness audit dispositioned 7,026 plugin-like objects with zero unexplained residue and accounted for all 5,846 live .fst files. Plan v5/v6 covered all 573 Sort-New candidates. Source validation found 60 Installed-mirror .fst entries and 513 binary-only candidates, with none yet proven as movable user-zone sources. The corpus audit retained 8 original suspects, surfaced 3 hidden Evolve misfiles and produced 11 meaningful placement findings.

WHY I DID IT Before building the GUI, I wanted proof that the product understood the denominator - not merely a curated subset.

HINDSIGHT This was one of the strongest completeness milestones. The important caveat was source authority: knowing a binary exists is not the same as having a valid .fst source to move. The system correctly kept that distinction.

EVIDENCE SNAPSHOT

- 7,026 plugin-like objects
- 5,846 live .fst accounted
- 573 candidates covered
- 60 Installed-mirror FSTs
- 513 binary-only

10 JULY 2026 ARCHITECTURAL PIVOT

GUI work is finally authorised

WHAT CHANGED The verdict changed from `gui_should_wait` to `gui_can_start_now`. A 12-screen desktop cockpit was built with 26/26 data dependencies loaded and 739 tests passing.

WHY I DID IT The decision was based on denominator proof and a functioning read-only intelligence spine, not on impatience to make screenshots.

HINDSIGHT Authorising the GUI was reasonable. The later problem was treating screen completion as product completion. A screen that renders data is not finished until its controls, state transitions, failures and recovery are connected to the live service spine.

EVIDENCE SNAPSHOT

- 12 screens
- 26/26 data dependencies
- 739 tests

10 JULY 2026

Two Windows packaging paths are proven

WHAT CHANGED The desktop shell moved from browser fallback to native Windows packaging. `pywebview`, `PyInstaller` and `Nuitka` were installed and verified. The `PyInstaller` executable was about 12.1 MB; the `Nuitka` build about 61.4 MB. Both passed version, smoke, doctor, launch-check and visible GUI launch. The full suite reached 848 passes, with no provider calls or file-movement authority enabled.

WHY I DID IT The product target was a double-click Windows application, not a local web page that required development tooling.

HINDSIGHT Proving both packagers was useful, but maintaining two release paths became expensive. `PyInstaller` eventually became the practical default; `Nuitka` was paused after a compiler out-of-heap failure. The better move would have been to choose one primary packager earlier and treat the second as a later optimisation experiment.

EVIDENCE SNAPSHOT

- `PyInstaller EXE` ~12.1 MB
- `Nuitka EXE` ~61.4 MB
- 848 tests passed
- Visible native launch proven

11-12 JULY 2026

The interface is forced to do real work

WHAT CHANGED The product requirement sharpened: clicking Scan, Research, Review or Organise had to call live services, settle into truthful state and survive navigation. Route identity became content-addressed; changing

workspace, profile, source or decisions invalidated stale ready data. Back/Forward proof expanded, and native reconciliation corrected 73 removed rows including 3x Osc, BooBass and BassDrum.

WHY I DID IT The cockpit could no longer be a beautiful report viewer. It had to become the single product surface for the actual runtime.

HINDSIGHT This was the beginning of the product-spine correction. The work was technically sound, but the project still underestimated how many screens had been built around incomplete vertical slices.

EVIDENCE SNAPSHOT

- Content-addressed plan identity
- Route invalidation
- 73 stale/removed rows reconciled

PART 5 RETROSPECTIVE

What this phase actually changed

This phase proved that Sortceror could move real files safely, account for the entire library, package a native desktop app and expose the intelligence spine. It also created the next trap: an enormous amount of backend truth was now visible through a GUI that looked further along than the end-to-end product actually was. The system was becoming impressive. It was not yet shippable.

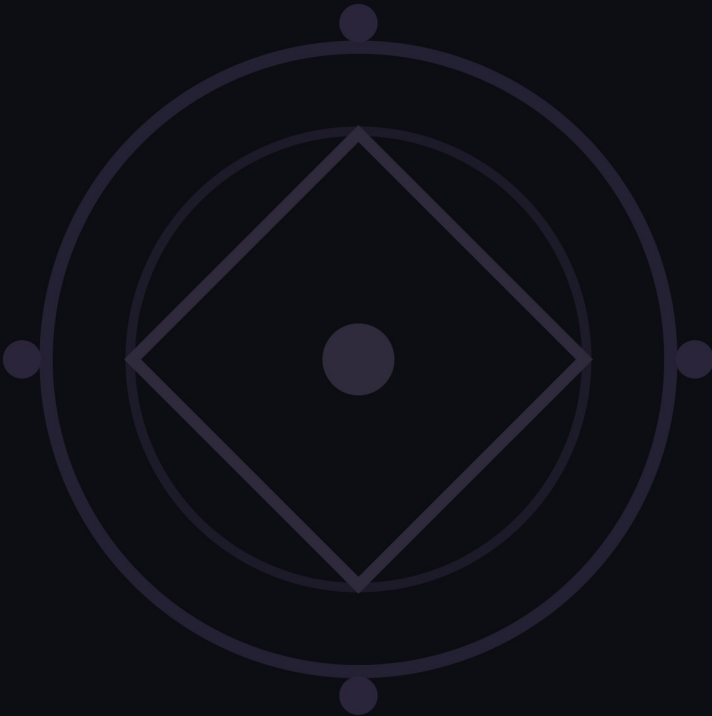
WHAT I WOULD DO DIFFERENTLY

1. I would have chosen one packaging path and frozen it until the product journey was stable.
2. I would have required every screen to demonstrate one complete user action before adding the next screen.
3. I would have introduced open-world held-out cohorts before describing replay results as autonomy evidence.
4. I would have tracked time-to-first-audit and time-to-undo as primary product metrics alongside test counts.

THE PRODUCT-SPINE CORRECTION

13-18 JULY 2026

An impressive system was still not a usable product.
The architecture was forced through one coherent
end-to-end spine.



ARCHITECTURAL AGE 06

PART 06 / 13-18 JULY 2026

THE PRODUCT-SPINE CORRECTION

The most useful review in the project was also the rudest to the existing architecture: despite thousands of passing tests and a sophisticated cockpit, a fresh user could not reliably complete the fundamental job. The answer was not another feature. It was one coherent spine from workspace selection to undo.

13 JULY 2026 ARCHITECTURAL PIVOT

The adversarial review says: not shippable

WHAT CHANGED The review found three critical failures. Sort-New could not execute a real plan. A cold first run could dead-end before the first audit. The no-overwrite decision was vulnerable to a state race. Human prompting was also far above the autonomy target.

WHY I DID IT The purpose of the review was to judge the product as a producer would experience it, not as a repository of components.

HINDSIGHT This was the moment I had to stop rewarding infrastructure. A product can have excellent evidence, tests and architecture diagrams and still fail its only job. The review was correct: the missing end-to-end spine was the highest-priority defect.

EVIDENCE SNAPSHOT

- Critical: Sort-New execution missing
- Critical: cold-start dead end
- Critical: no-overwrite race
- Major: excessive prompting

13 JULY 2026 ARCHITECTURAL PIVOT

One spine replaces competing realities

WHAT CHANGED The recommended model became: Workspace → ScanRun → immutable source snapshot → Audit → FST-only SortPlan → Exceptions → Execution Journal → Verification → Undo. Non-FST Sort-New rows, duplicate packagers and operator-only surfaces were to be deprecated or contained.

WHY I DID IT The product needed one definition of current workspace, one source snapshot, one plan identity and one execution history. Anything else invited stale UI state and contradictory answers.

HINDSIGHT This spine is still the correct architecture. It also gave the project a ruthless prioritisation test: if a feature did not improve this journey or the intelligence feeding it, it could wait.

EVIDENCE SNAPSHOT

- Canonical product spine defined

14 JULY 2026

Phase 0 establishes a clean product baseline

WHAT CHANGED Product-spine adoption, legacy containment, a clean baseline, PyInstaller consolidation and a cold-audit launchpad were checkpointed. The next gate was deliberately narrow: can a fresh packaged user select a workspace and get a truthful read-only audit?

WHY I DID IT The phase reduced scope after the architecture had spread too widely. Packaging and startup were treated as part of the product contract, not deployment trivia.

HINDSIGHT This was a strong reset. The naming of numbered phases also helped, although later phase/pass terminology became overly dense. The underlying sequencing - cold audit before mutation - was exactly right.

EVIDENCE SNAPSHOT

- Checkpoint: 134f5dc
- Phase 0 product spine accepted

14 JULY 2026

Phase 1 fails on real bundled knowledge

WHAT CHANGED Ten of twelve gates passed. The failing gates were knowledge matching and package parity. Against the real bundled GlobalKnowledge, the fixture expected 9 recognised and 1 unknown but produced 1 recognised and 9 unknown. A synthetic reviewed lane achieved the expected 9/1, proving the engine worked while the shipped knowledge did not.

WHY I DID IT The gate was designed to catch the difference between a correct algorithm and a useful package. Synthetic success could no longer stand in for bundled product truth.

HINDSIGHT This was an excellent failure. It prevented a common lie: "the feature works in tests" when the actual package lacks the data required to make it work. Product inputs are code in every meaningful sense.

EVIDENCE SNAPSHOT

- 10/12 gates passed
- Real bundle: 1 recognised / 9 unknown
- Synthetic lane: 9 recognised / 1 unknown
- Phase blocked

15 JULY 2026

Cold read-only audit is accepted

WHAT CHANGED After repairing bundled knowledge parity, the packaged product could install a workspace, perform a cold read-only audit, persist a Library view and recognise 9 of 10 golden products while safely leaving one unknown. Live mutation remained locked.

WHY I DID IT The first useful consumer milestone was deliberately non-destructive. A producer could launch the app and receive truthful library information without granting movement authority.

HINDSIGHT This was the right minimum product. It should probably have been built several weeks earlier, before the 12-screen cockpit. A small real journey reveals architecture defects faster than a large mock surface.

EVIDENCE SNAPSHOT

- Commit: aecbba1
- Tag: phase1 cold read-only audit accepted
- 9 recognised / 1 safely unknown

15 JULY 2026

Phase 2A adds durable knowledge and profiles

WHAT CHANGED GlobalKnowledge consolidation, a durable user-owned overlay, complete sort-profile behaviour, producer correction workflows, external read-only alpha hardening, privacy-tiered exports, false-identity measurement, offline cohort evidence and a packaged alpha candidate were accepted. Two full suites each reported 2,160 passed and 4 skipped.

WHY I DID IT A read-only audit becomes more useful when the user can correct identity or placement and preserve that correction without editing project files.

HINDSIGHT The feature set was legitimate, but the project again risked horizontal expansion. The missing vertical slice was now obvious: Organise, Execute, Verify, Undo. More alpha infrastructure would not substitute for it.

EVIDENCE SNAPSHOT

- 2,160 passed / 4 skipped twice
- User-owned knowledge overlay
- Profiles and correction workflow

15 JULY 2026 **ARCHITECTURAL PIVOT**

I call out the over-auditing loop

WHAT CHANGED I stopped accepting endless deferral. The project kept proving adjacent infrastructure while the actual mutation journey remained unfinished. I demanded a working vertical slice and a cleaner, darker, producer-facing UI with less compliance theatre.

WHY I DID IT The opportunity cost had become unacceptable. Every broad re-audit delayed the moment the product could solve the original problem.

HINDSIGHT This was a necessary management correction. The project needed validation economy and a stronger definition of done: one working journey beats five immaculate reports about why the journey is not ready.

EVIDENCE SNAPSHOT

- Priority reset: Organise → Execute → Verify → Undo

16 JULY 2026

Phase 3 delivers the first working vertical slice

WHAT CHANGED Organise, Execute, Verify and Undo were implemented and accepted against controlled data. Live user-root activation remained pending, but the product finally contained the complete reversible action path.

WHY I DID IT This closed the oldest gap in the project: turning an audited plan into a journalled, verified and reversible result through the product.

HINDSIGHT The slice was a major milestone, but controlled fixtures still did not prove the installed-copy or clean-room realities. The project correctly continued with progressively more realistic environments rather than calling it done.

EVIDENCE SNAPSHOT

- Commit: 185ddfa
- Tag: phase3 intelligent organise/execute/undo accepted

16 JULY 2026

Phase 3.1 proves installed-copy Scan New

WHAT CHANGED The vertical slice was extended to an installed copy of the Plugin Database. Scan New could operate against a realistic copy while the live user root remained untouched.

WHY I DID IT The product needed the filesystem shape, sidecars and permissions of a real installation without risking the production library.

HINDSIGHT Installed-copy testing was the correct intermediate environment. The next clean-room suite then exposed just how much hidden coupling still existed.

EVIDENCE SNAPSHOT

- Commit: 34de454
- Installed-copy Scan New accepted

16 JULY 2026

The clean-room suite goes properly red

WHAT CHANGED The clean-room run reported 2,369 passed, 11 skipped, 62 failed and 23 errors. Packaging was held. Failures included resolver-aware producer projection, Waves variant identity and category behaviour, variant-scoped Local Overrides, denominator proof and evidence-verifier defects.

WHY I DID IT A clean room removes the accidental support of the developer machine: stale caches, known paths, prior state and hidden dependencies.

HINDSIGHT This failure was painful and valuable. The project did not lower the gate or cherry-pick a green subset. It treated the red suite as a bounded repair loop. That is exactly how a release candidate should behave.

EVIDENCE SNAPSHOT

- 2,369 passed
- 11 skipped
- 62 failed
- 23 errors
- Package held

17-18 JULY 2026

Phase 3.2 repairs producer interaction and runtime truth

WHAT CHANGED The correction was kept narrow: make projection resolver-aware, close variant identity, scope Local Overrides correctly, prove the real-library denominator and repair contradictions and verification. The accepted checkpoint landed at commit c57a4b0 with a clean synchronized tree.

WHY I DID IT The purpose was not to opportunistically add features. It was to make the existing vertical slice truthful under clean conditions.

HINDSIGHT This was a good bounded recovery. It also showed why product state, identity state and variant state must be typed rather than inferred from display strings.

EVIDENCE SNAPSHOT

- Commit: c57a4b0
- Tree: 024262a
- Phase 3.2 producer interaction/runtime accepted

18 JULY 2026 ARCHITECTURAL PIVOT

The first hybrid-autonomy audit rejects the clever bit

WHAT CHANGED The Phase 4 intelligence foundation combined reviewed authority, local evidence, FTS retrieval, embeddings, cached external claims, hybrid ranking, contradiction constraints and replay calibration. The audit found the HYBRID_AUTONOMOUS path unsafe: independence groups were not applied, contradictions were incomplete, the runtime ranker did not match the persisted artifact, policy was hard-coded, probability claims were not genuinely calibrated, replay leakage existed and provider readiness was overstated.

WHY I DID IT The goal was a single resolver that could use every evidence source while preserving authority. The audit asked whether the final autonomous state was actually earned.

HINDSIGHT Rejecting the hybrid autonomy path was correct. FTS and semantic discovery were useful; embeddings remained discovery-only; unresolved workbench and user learning were accepted. The important principle was preserved: a sophisticated ranker does not become authority because it produces a probability-shaped number.

EVIDENCE SNAPSHOT

- Hybrid autonomy disabled
- FTS/semantic discovery accepted
- Embeddings discovery-only
- Provider readiness limited

18-20 JULY 2026

Phase 4.1 is accepted fail-closed

WHAT CHANGED The intelligence and producer-runtime foundation was repaired, resealed and checkpointed at b7483ff. Full validation accounted for 2,581 nodes exactly once: 2,556 passed and 25 expected skips. Source and packaged navigation both passed 38/38. A packaged Product Detail refresh defect was caught, fixed and re-proven before checkpoint. Hybrid autonomy remained disabled.

WHY I DID IT The objective was not to force the ambitious intelligence path through. It was to preserve the useful foundation and ship only the states the evidence justified.

HINDSIGHT This is a mature milestone because it narrowed the claim instead of weakening the gate. The foundation was accepted; autonomy was not. That distinction is exactly what the product's authority model was built to support.

EVIDENCE SNAPSHOT

- Commit: b7483ff
- Tree: d548a1c
- 2,556 passed / 25 expected skips
- 38/38 source navigation
- 38/38 package navigation
- Hybrid autonomy fail-closed

PART 6 RETROSPECTIVE

What this phase actually changed

The product-spine correction is where Sortceror became honest about the difference between a technical platform and a usable product. It also proved the value of adversarial gates: bundled knowledge, clean-room state and packaged DOM refresh each failed in ways that unit tests alone did not reveal. The broader lesson is that architecture has to terminate in a human action. If the producer cannot launch, understand, act and undo, the architecture is unfinished.

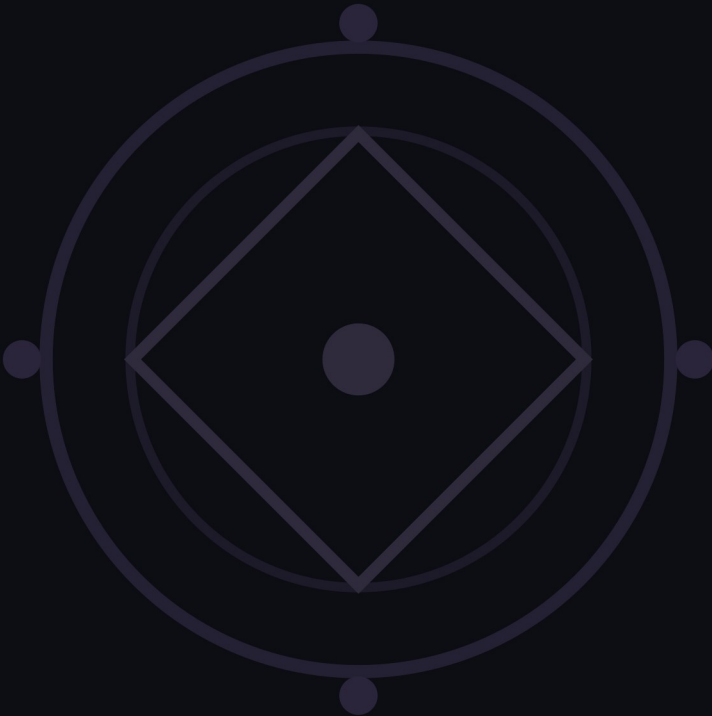
WHAT I WOULD DO DIFFERENTLY

1. I would have built the cold-audit vertical slice before the full cockpit.
2. I would have made clean-room packaging a routine lane much earlier.
3. I would have kept phase naming simpler and tied each phase to one consumer-observable capability.
4. I would have treated refresh, navigation and restore state as core runtime logic rather than frontend polish.

CONSUMERISATION AND RUNTIME TRUTH

19–26 JULY 2026

Visual polish, packaging, state ownership, recovery, performance, and the ugly work of making “green” mean usable.



ARCHITECTURAL AGE 07

PART 07 / 19-26 JULY 2026

CONSUMERISATION AND RUNTIME TRUTH

With the intelligence foundation fail-closed and the vertical slice real, the remaining work looked deceptively like polish. It was not. Packaging, navigation, visual state, startup ownership, recovery, progress truth, schema migration and query performance are all part of whether a desktop product can be trusted. This phase was the slow conversion from developer-correct to consumer-reliable.

19-20 JULY 2026

Phase 4.1 survives privacy reseal and navigation defects

WHAT CHANGED An initially accepted product pack needed privacy resealing, and a later navigation flicker superseded the candidate. The work was held until both the exact package inputs and the real route behaviour were corrected. The final checkpoint remained b7483ff rather than preserving an earlier almost-green pack.

WHY I DID IT A release identity has to include its privacy surface and its actual packaged behaviour. A visually intermittent defect is still a defect even when the underlying data is correct.

HINDSIGHT This was the right refusal to average away faults. The project had begun to treat exact package identity, not merely source commit, as the release object.

EVIDENCE SNAPSHOT

- Candidate superseded after privacy/navigation issues
- Final checkpoint remotely verified

20 JULY 2026

Cleanup reclaims 20.71 GB

WHAT CHANGED A staged cleanup targeted 139 generated roots, fully removed 132 and left seven ACL-protected shells under a copied Waves Surround path. The operation reclaimed 20.71 GB on C:.

WHY I DID IT The earlier artifact blowout had made cleanup a release concern. The process needed transparency about exactly what would be removed and what permissions prevented removal.

HINDSIGHT The improvement was not merely free disk space. Cleanup became manifest-driven and bounded. The remaining weakness was that evidence retention still required active attention rather than being enforced automatically by policy.

EVIDENCE SNAPSHOT

- 139 targets
- 132 removed
- 7 protected shells retained
- 20.71 GB reclaimed

20 JULY 2026

The Scan Sigil design substrate is frozen separately

WHAT CHANGED A self-contained design substrate for the Scan Sigil and Home Glyph was committed under a design-only path: 83 files, 81/81 manifest validation, no runtime or package-input changes. This kept the visual source distinct from production wiring.

WHY I DID IT The animated scan state was intended to make long analysis legible and specific to Sortceror without allowing experimental assets to contaminate the runtime.

HINDSIGHT Separating design authority from integration was correct. The mistake would have been to let implementation convenience redraw or simplify the accepted motion system. The later integration pass explicitly prevented that.

EVIDENCE SNAPSHOT

- 83 design files
- 81/81 manifest
- 0 runtime bytes changed

20 JULY 2026 **ARCHITECTURAL PIVOT**

Validation economy becomes explicit

WHAT CHANGED A bounded integration pass classified 403 test files exactly once. FAST ran 992 passed and 1 skipped in 78.646 seconds; INTEGRATION ran 1,028 passed and 2 skipped in 520.791 seconds. Historical, Windows/GPU, release and packaging lanes were deliberately not run because their inputs had not changed.

WHY I DID IT The project had wasted too much time rerunning every proof after small changes. The new law was one appropriate lane per affected surface, with broader gates only when relevant bytes changed.

HINDSIGHT This was not reduced rigor. It was better causal reasoning. Validation that cannot explain why it is running is theatre.

EVIDENCE SNAPSHOT

- 403 test files classified
- FAST: 992 + 1 skip
- INTEGRATION: 1,028 + 2 skips

21 JULY 2026

Scan Sigil and Home Glyph reach the live product

WHAT CHANGED The runtime integration used one active WebGL context, same-job polling through the existing state mechanism and the accepted assets without visual redesign. A 14.755-second capture contained 867 frames at 58.758 effective fps. First shell paint was 111.1 ms; first fallback paint 2,466.5 ms; first complete WebGL frame 2,900.4 ms; 100 polls across five sequences produced zero visible deviations.

WHY I DID IT The scan animation needed to be smooth, truthful and tied to the actual job state rather than functioning as a generic loading loop.

HINDSIGHT The integration proved the visual system could be exact and performant. It also exposed how easy it is for a stale test contract to block valid architecture when tests forbid all script tags rather than distinguishing pinned local assets from remote code.

EVIDENCE SNAPSHOT

- 867 frames
- 14.755 s
- 58.758 fps
- First shell paint 111.1 ms
- 1 active WebGL context

21 JULY 2026

A stale safety test blocks packaging

WHAT CHANGED FAST failed because an old test rejected every <script src> tag, including pinned local Scan Sigil assets. Packaging stopped immediately; no EXE or downstream lanes ran until the contract was corrected. The final integration checkpoint landed at 3735291 with a clean remote-synchronised tree.

WHY I DID IT The safety intent - no uncontrolled remote scripts - was valid. The implementation of the test had become broader than the actual policy.

HINDSIGHT This is why historical tests cannot automatically define the current product. Tests are executable law, but laws can become stale. The correct response is to repair the test against the real safety invariant, not to work around it or delete it blindly.

EVIDENCE SNAPSHOT

- Packaging correctly stopped
- Commit: 3735291
- Tree: 74ee0b2

21-22 JULY 2026

The product-wide visual overhaul is judged against references

WHAT CHANGED The visual pass covered the full product rather than only the scan screen. The first source candidate was rejected because Product Detail materially diverged from the reference and Home and History had noticeable departures. The source candidate remained frozen and unpackageable until visual parity and interaction states were corrected.

WHY I DID IT The product needed a coherent visual system: dark, premium, legible, tactile and restrained. The purpose was not decorative novelty; it was to make dense evidence and state understandable to a producer.

HINDSIGHT This was a good use of visual acceptance. The risk was spending too much time on reference parity while consumer reliability defects remained. That tension became explicit the next day when the packaged app failed normal usage.

EVIDENCE SNAPSHOT

- Source visual candidate initially rejected
- Product Detail, Home and History corrected
- Packaging held pending visual acceptance

23 JULY 2026 ARCHITECTURAL PIVOT

The latest package is rejected as a consumer product

WHAT CHANGED The package showed intermittent normal launch, a false Ready state, a disabled Plugin Database button, obscured route loading, blank route handoff, Scan New inheriting VERIFIED state, weak Product Detail variant tactility and general slowness. A missing horizontal seam also broke the intended 4x2 Home metric grid.

WHY I DID IT The app looked close in controlled captures but failed the mundane actions a real user performs. The recovery pass therefore stopped all new design and packaging work.

HINDSIGHT This was another reminder that screenshot quality and product quality are different axes. Consumer reliability had to outrank the urge to keep polishing visuals.

EVIDENCE SNAPSHOT

- Package invalidated
- Startup, route, state, control and performance defects identified

23 JULY 2026

Runtime Consolidation Pass 1

WHAT CHANGED Pass 1 added a RuntimeAtlas with ten entries covering eight screens, startup tracing, safe mode, last-known-good recovery, interrupted-job recovery, single-instance ownership, non-restorable-window handling, bounded heartbeats and canaries. FAST reported 1,073 passed and 1 skipped; Integration 1,030 passed and 2 skipped; current-law 75 passed. Physical relaunch succeeded 20/20. Warm visible shell measured 672.607 ms; first usable screen 2,191.54 ms; maximum startup gap 1,868.173 ms.

WHY I DID IT The immediate goal was to make launch, ownership and recovery observable before attempting another consumer package.

HINDSIGHT This was proper runtime engineering. The checkpoint at 6cdd995 gave the product a measurable startup model instead of anecdotal "it opened for me" confidence.

EVIDENCE SNAPSHOT

- Commit: 6cdd995
- 122 paths
- 20/20 relaunch
- Warm shell 672.607 ms
- First usable 2,191.54 ms

24 JULY 2026

Runtime Consolidation Pass 2

WHAT CHANGED Pass 2 moved more application state into typed, durable runtime ownership and tightened navigation, capability truth, workspace persistence, recovery and job-state projection. It was checkpointed at cbe81a8 with schema v13 as the accepted baseline for the data-plane work.

WHY I DID IT The recurring UI defects were symptoms of duplicated or inferred state. The fix was to make state ownership explicit rather than patch each screen independently.

HINDSIGHT This pass is less visually dramatic than the ones around it and more architecturally valuable. Typed state and one ownership path reduce entire families of bugs: stale readiness, inherited terminal state, blank handoff and impossible combinations.

EVIDENCE SNAPSHOT

- Commit: cbe81a8
- Tree: e65f970
- Runtime Consolidation Pass 2 accepted

24-25 JULY 2026

Runtime Consolidation Pass 3: release-scale data plane

WHAT CHANGED Pass 3 upgraded the operational schema from v13 to v14: 99 to 111 tables, 71 to 84 indexes and 177 to 179 triggers. Migration took about 58 ms and reapplication about 10 ms. No-change incremental scans measured p95 0.446 s at 10,000 items and 4.90 s at 100,000; representative one-change and 100-change 10k runs remained under a second. The package-input set contained 699 files and a diagnostic executable and sealed audit archive were produced.

WHY I DID IT The product needed incremental filesystem scanning, a primary write model, dependency invalidation and query performance that would not collapse on a large library.

HINDSIGHT This was the right time to optimise because the product spine and state ownership were finally stable enough to measure. Earlier optimisation would have entrenched the wrong model.

EVIDENCE SNAPSHOT

- Schema 13 → 14
- 99 → 111 tables
- 71 → 84 indexes
- 10k no-change p95 0.446 s
- 100k no-change p95 4.90 s
- Commit: 6fe9e52

25 JULY 2026

Runtime Consolidation Pass 4: convergence

WHAT CHANGED Pass 4 decomposed frontend/runtime responsibilities, consolidated single-state ownership, introduced virtualised producer workspaces, tightened transactional execution and recovery, retired legacy paths and prepared the consumer candidate. Scan progress moved to one backend-owned field used by the ring, internal percentage and accessibility progress bar. Source visual proof reached 65/65 and the fresh execution → relaunch → History → byte-identical Undo journey was sealed before checkpoint.

WHY I DID IT The goal was to make the product one system, not a UI layered over competing runtimes. Progress, cancellation and terminal state needed one source of truth.

HINDSIGHT This was the culmination of the runtime work. It also produced the validation-economy rule now used across routine changes: only affected lanes run unless a product, runtime or package-input byte changes.

EVIDENCE SNAPSHOT

- Source visual matrix 65/65
- Backend-owned progress truth
- Execution/relaunch/History/Undo journey
- Checkpoint: c0cfd58

26 JULY 2026

Runtime Consolidation Pass 5: architecture truth

WHAT CHANGED The final correction moved the operational schema to version 16, removed unused generic transaction methods such as `commit_step_intent`, `record_filesystem_applied` and `verify_step`, repaired route authority, bounded-list truth, package/capability truth and stale documentation. It deliberately avoided a schema redesign, executor rewrite, new provider work, live-user mutation or a second packager.

WHY I DID IT After several consolidation passes, the remaining risk was documentation and API surface claiming capabilities the live product did not use.

HINDSIGHT Removing dead abstractions is as important as adding features. The project had accumulated generic mechanisms for imagined futures. Pass 5 made the architecture describe the product that existed.

EVIDENCE SNAPSHOT

- Schema version 16
- Unused transaction surface removed
- Capability and route truth corrected

PART 7 RETROSPECTIVE

What this phase actually changed

Consumerisation forced the project to accept that startup, state restoration, navigation, progress and package identity are not peripheral engineering. They are the user's evidence that the product knows what it is doing. The visual work was worthwhile, but the strongest result of this phase was not the sigil; it was the convergence on one state model, one progress truth and one reversible product journey.

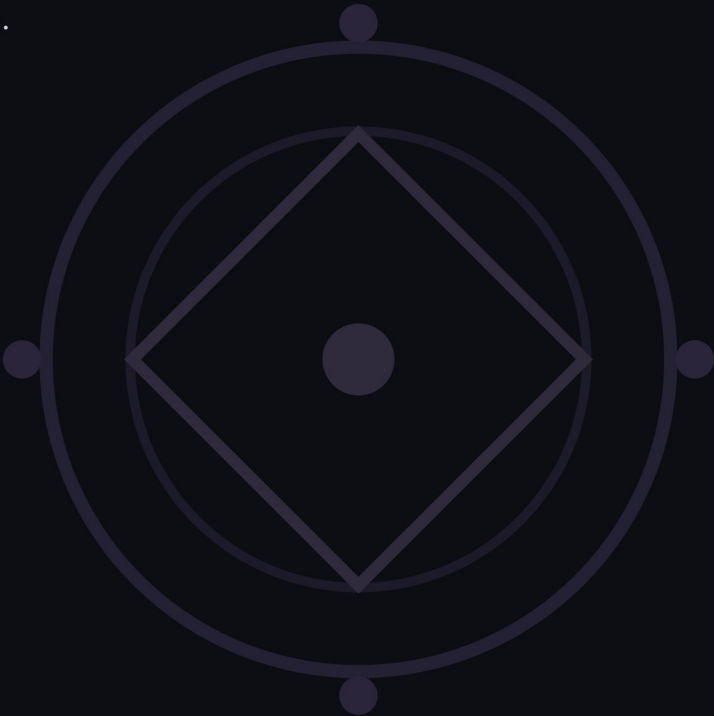
WHAT I WOULD DO DIFFERENTLY

1. I would have started runtime ownership and recovery work when the first desktop shell was created.
2. I would have frozen the visual system after the first accepted language and spent more time on interaction states.
3. I would have made package identity a first-class database record earlier.
4. I would have formalised validation economy before the July mega-suites consumed days.

THE INTELLIGENCE PROGRAMME

26–30 JULY 2026

Safety was ahead of intelligence. The next pivot built an open-world evidence platform—and then the human gate exposed what remained broken.



ARCHITECTURAL AGE 08

THE INTELLIGENCE PROGRAMME

By late July the safety and runtime architecture were substantially ahead of the actual intelligence. That was the new bottleneck. Sortceror could preserve evidence, refuse unsafe authority, package a desktop runtime and undo verified movement. It still needed to recognise unseen products, distinguish variants and roles, acquire the right evidence, calibrate uncertainty and reduce review without cheating on the benchmark.

26 JULY 2026 ARCHITECTURAL PIVOT

The autonomy audit revokes the flattering story

WHAT CHANGED A dedicated intelligence review concluded that the safety and authority model were ahead of the intelligence. The autonomy certification was not valid. The resolver was still too closed-catalogue, the production-shaped benchmark contained leakage and the active request path had not proved that it used the deeper local-evidence plane.

WHY I DID IT The audit was designed to answer a different question from the runtime passes: not "does the machinery work?" but "does the system know enough to deserve autonomy on unseen plugins?"

HINDSIGHT This was the correct diagnosis. The project had been too willing to describe replay precision as intelligence. A resolver that recognises known families under format noise is useful; it is not open-world understanding.

EVIDENCE SNAPSHOT

- Autonomy certification revoked
- Closed-catalogue bias identified
- Benchmark leakage identified

26 JULY 2026

Pass I narrows and repairs the intelligence claim

WHAT CHANGED Pass I repaired role, version and vendor semantics; replaced string-based autonomy classes with structured predicates; built an adversarial grouped benchmark and ablation baseline; and explicitly disabled unseen autonomy. Dynamic probes, learned reranking, fine-tuning, conformal prediction and live provider research were deferred from this bounded pass.

WHY I DID IT Before adding more mechanisms, the project needed a benchmark that could tell whether the existing resolver actually generalised.

HINDSIGHT This was disciplined. It traded headline ambition for a trustworthy baseline. The later programme could then build on known weaknesses instead of hiding them inside aggregate accuracy.

EVIDENCE SNAPSHOT

- Reviewed exact identity: 2,612/2,612
- Format-noise recovery: 97.62634%
- Unseen-vendor-group coverage: 22.320061%
- 0 false autonomous identity/destination errors

27 JULY 2026

Intelligence Pass I is checkpointed

WHAT CHANGED The accepted baseline landed at commit 3a33b3a. It preserved exact identity for all 2,612 reviewed products, recovered most format-noise variants, kept role-translation, semantic-version and cross-vendor autonomous errors at zero, and left unseen autonomy disabled. The audit archive and benchmark semantic content were hashed separately.

WHY I DID IT The objective was to freeze a truthful starting line for the larger programme.

HINDSIGHT The most important number was not 2,612/2,612; it was 22.32% unseen-vendor-group candidate coverage. That is where the real product gap lived. The baseline made that gap impossible to hand-wave.

EVIDENCE SNAPSHOT

- Commit: 3a33b3a
- Tree: 4548902
- Archive SHA begins bc9732
- Unseen autonomy disabled

27 JULY 2026

Sortceror becomes a public product identity

WHAT CHANGED The project secured the primary domain and major social handles, moved the repository under the Sortceror organisation and codified the brand system: canonical SVG sources, obsidian and purple palette, raster export pipeline, icons, avatars and platform-safe assets.

WHY I DID IT The product had reached the point where its external identity needed to be as deliberate as its internal architecture.

HINDSIGHT Brand work did not improve recognition accuracy, but it mattered commercially and forced asset provenance and repeatable exports. The key was keeping branding changes physically separate from the active intelligence checkpoint so one could not contaminate the other.

EVIDENCE SNAPSHOT

- sortceror.com secured
- Official organisation and handles
- Canonical SVG → generated export pipeline

27-28 JULY 2026

Intelligence Pass II builds the product identity graph

WHAT CHANGED Pass II introduced a SQLite-backed graph across vendors, products, versions, editions, formats and implementations. Local evidence adapters brought filenames, hashes, binaries, PE metadata, VST3 classes, signatures, registry records, sidecars, reviewed knowledge and prior decisions into one evidence spine. The resolver kept separate conclusions for product identity, semantic function, taxonomy placement and move permission, surfaced contradictions and alternative hypotheses, generated review candidates and replayed corrections.

WHY I DID IT The project needed to stop collapsing every question into one label. "What binary is this?", "what product family is it?", "what does it do?", "where should it go?" and "may I move it?" are related but different conclusions.

HINDSIGHT This was the most substantial intelligence architecture since the Python/SQLite pivot. Its weakness was performance on genuinely held-out families: roughly half top-1 in the retained family test, with weaker independent semantic accuracy. The open-set detector was promising but not identity-capable.

EVIDENCE SNAPSHOT

- Identity graph across vendor/product/version/edition/format/implementation
- Separate identity, semantic, taxonomy and authority conclusions
- Alternative hypotheses and contradictions

28 JULY 2026

Pass II is published with exact scope separation

WHAT CHANGED The accepted checkpoint landed at a7b6cad with tree 6f005421. It committed 147 paths: 62 modified and 85 added, with no deletions. Package inputs totalled 757. Intentional branding work remained outside the checkpoint as exactly 28 dirty paths, preserving the intelligence tree's identity.

WHY I DID IT The goal was to publish the architecture without accidentally absorbing unrelated brand assets or line-ending noise into the checkpoint.

HINDSIGHT This is a good example of source-control truth serving product truth. A large pass is only reviewable when its physical scope matches its declared scope.

EVIDENCE SNAPSHOT

- Commit: a7b6cad
- 147 paths: 62 modified / 85 added
- 757 package inputs
- 28 branding paths excluded

28 JULY 2026 ARCHITECTURAL PIVOT

Pass III raises the bar to real-format and behavioural intelligence

WHAT CHANGED Pass III targeted contained VST3 and CLAP introspection, a real installed-plugin cohort, dynamic capability packets, semantic gold truth, calibrated learned reranking, official-source research, active learning, shadow autonomy and an inactive signed knowledge-release candidate. Live file movement and production knowledge activation remained locked.

WHY I DID IT The evidence graph now needed richer observations and better generalisation. Static metadata alone cannot always distinguish products with vague names or overlapping marketing language.

HINDSIGHT The contract was ambitious and appropriately separated intelligence from mutation authority. The risk was trying to prove too many new subsystems in one pass, making it harder to tell which evidence actually caused an improvement.

EVIDENCE SNAPSHOT

- Contained native probe host
- Real plugin cohort
- Learned reranking
- Official research
- Active learning
- Shadow autonomy
- Signed knowledge candidate

28-29 JULY 2026

Pass III is implemented in substance, then repeatedly re-proven

WHAT CHANGED Focused validation reached 80/80. FAST reported 1,086 passed and 1 skipped; CURRENT later reached 2,259 passed and 3 skipped; WINDOWS_GPU reached 51 passed and 3 skipped. A release-current error

exposed a legacy helper that assumed a .git directory existed inside a detached package source, invalidating that freeze and forcing a new package source. Further audit work challenged network-denial proof, semantic-gold independence, research-claim precision, active-learning evidence and shadow-autonomy calibration.

WHY I DID IT The purpose was to prove that the new intelligence stack worked in a clean detached package, not only inside the repository.

HINDSIGHT The repeated invalidations were frustrating and correct. A signed or sealed artifact is meaningless if the build helper reaches back into repository state or the benchmark reuses evidence from the answer key.

EVIDENCE SNAPSHOT

- Focused: 80/80
- FAST: 1,086 + 1 skip
- CURRENT: 2,259 + 3 skips
- WINDOWS_GPU: 51 + 3 skips
- Detached-package defect found

29 JULY 2026 ARCHITECTURAL PIVOT

The diagnostic package passes automation and fails the human gate

WHAT CHANGED The final diagnostic ZIP contained 964 files. Its ZIP SHA256 began cbba7928; the EXE began 87c7ca2f; the package-input aggregate began cafed33b. Automated proof was green across the recorded lanes: 61 focused checks, 1,089 fast, 2,294 current, 52 Windows/GPU and 92 release-current. The package was explicitly marked diagnostic-only.

WHY I DID IT The final gate was a real double-click consumer sequence. That gate found what the automated evidence missed.

HINDSIGHT This is the current defining lesson: the human gate is not anecdotal QA after the "real" tests. It is a first-class acceptance test. The correct response was to invalidate the package and suspend checkpoint publication.

EVIDENCE SNAPSHOT

- ZIP: cbba7928...
- EXE: 87c7ca2f...
- 964 files
- Automated proof: 61 / 1,089 / 2,294 / 52 / 92
- Checkpoint withheld

29-30 JULY 2026

The exact consumer failures

WHAT CHANGED Start contained analysis was disabled because the plugin implementation was not bound and surfaced UNKNOWN / NOT_BOUND. Queue official research reported NETWORK DISABLED. Re-selecting the same Plugin Database root triggered ambiguous workspace invalidation and endless cyan loading. Sidebar routes could remain in loading state. A clean-room scan stalled at 81% without useful progress. Closing and relaunching could produce "Sortceror could not restore the app."

WHY I DID IT These were not cosmetic defects. They broke the new intelligence features, workspace truth, progress observability and recovery - the exact surfaces Pass III claimed to advance.

HINDSIGHT The package therefore has no release authority and Pass III has no accepted checkpoint. The accepted baseline remains Pass II. The current job is to reproduce each failure from captured state, repair the active service bindings and workspace state machine, rebuild from a clean source and rerun the human sequence before any publication.

EVIDENCE SNAPSHOT

NOT_BOUND | NETWORK DISABLED | SAME-ROOT HANG | 81% STALL | RESTORE FAILURE

PART 08 RETROSPECTIVE

What this phase actually changed

The July intelligence programme established the right graph, the right authority separation and the right refusal to grant unseen-product autonomy. It also ended on a false horizon. The diagnostic package failed because the new capability was not reliably bound into the ordinary product journey. That was not the end of the story; it was the beginning of a month spent discovering how many different ways a technically sophisticated system can still fail to be a usable application.

The green lanes were not necessarily lying. They were answering smaller questions than the producer was asking.

WHAT I WOULD DO DIFFERENTLY

- 01 I would have treated the July 30 document as an interim volume, not a present-day conclusion.
- 02 I would have moved the physical double-click journey ahead of the final large evidence archive.
- 03 I would have refused any phase whose user-visible capability was not already bound through the packaged service spine.
- 04 I would have recognised sooner that repeated adjacent runtime defects were evidence of ownership failure, not bad luck.



PART IX

THE GREEN MACHINE THAT WASN'T

30 JULY-08 AUGUST 2026

The package had immaculate internal evidence and a user journey that fell apart. August began by refusing to call that a near miss.



ARCHITECTURAL AGE 09

SORTCEROR / DEVELOPMENT TIMELINE

30 JULY 2026

Automation green, product red

A million green assertions do not add up to one working click.

WHAT CHANGED

The final Intelligence Pass III diagnostic package carried green focused, fast, current, Windows/GPU and release-current evidence. The actual package then reported contained analysis as not bound, official research as network disabled, same-root reselection as an endless loading state, scan progress frozen at 81 percent, and restore failure after relaunch. I withheld the checkpoint and kept the package diagnostic-only.

WHY I DID IT

The purpose of the final package gate was to answer whether a producer could use the capability, not whether each subsystem could demonstrate internal consistency. The human sequence found exactly the failures the sealed automation could not.

HINDSIGHT

This became the defining August rule: the user journey is not anecdotal quality assurance after the real proof. It is the proof root. A package that cannot invoke, display and recover its own capability has no release authority, regardless of how beautiful the test summary looks.

EVIDENCE SNAPSHOT

- Contained analysis: NOT_BOUND
- Same-root reselect: endless loading
- Relaunch restore: failed
- Official research: NETWORK DISABLED
- Clean-room scan: 81% stall

02 AUGUST 2026

I refused to march into Phase IV

WHAT CHANGED

The natural programme sequence said the next thing was more intelligence. I stopped it. The preceding corrections had a pattern: every local fix exposed another neighbouring failure in startup, phase ownership, projection, recovery or state publication. That was no longer a list of unrelated bugs. It was evidence that the runtime architecture had too many competing truths.

WHY I DID IT

Building more capability on top of an unstable product spine would have multiplied the number of ways the product could disagree with itself. The correct intervention was not another feature or another package attempt. It was an adversarial architecture and ownership audit.

HINDSIGHT

This was one of the most valuable uses of delay in the project. I was not deferring the product; I was refusing to bury a structural problem under another layer of impressive machinery. The cost was another week of internal work. The alternative cost was months of debugging every new capability through the same broken ownership seams.

EVIDENCE SNAPSHOT

- Phase IV held
- No provider activation
- No new intelligence owner
- Architecture review authorised
- No live-user mutation

03 AUGUST 2026

101 of 101 assemblies trusted, and the app still failed

The capability existed. The product had forgotten the words for it.

WHAT CHANGED

A Runtime Consolidation R8 package failed during startup. The managed-runtime trust closure itself passed 101 of 101 assemblies. The actual defect was smaller and more embarrassing: the shell emitted four real startup phases - PYTHONNET_TRUST_CACHE_READY, MANAGED_RUNTIME_TRUST_CLOSURE_READY, WEBVIEW_MANAGED_ASSEMBLIES_LOADED and WEBVIEW_RUNTIME_AVAILABLE - that the central startup phase registry did not recognise.

WHY I DID IT

The package needed one owner for the lifecycle vocabulary. Trust was not broken. The consumer and registry simply disagreed about legal states. I corrected the phase contract rather than touching the managed-runtime machinery that had already proved itself.

HINDSIGHT

This was August in miniature. A large, expensive subsystem could be correct while one tiny duplicated vocabulary made the whole product look dead. The lesson was not to add more validation around the registry. It was to eliminate the reason two places could disagree in the first place.

EVIDENCE SNAPSHOT

- Managed trust: 101/101
- Failure class: producer/registry drift
- Four missing legal startup phases
- Correction kept bounded

03-04 AUGUST 2026

Attempt 08 went green; Attempt 09 stalled the projection child

WHAT CHANGED

The next sequence proved the repaired startup path, then failed later at PROJECTION_CHILD_STALLED. A dedicated P09-06 correction loop and an eight-by-eight correction matrix closed the known failure modes without reopening Product Detail or the progress architecture wholesale.

WHY I DID IT

The narrow objective was to distinguish a genuine resolver or projection failure from another lifecycle/reporting mismatch. I kept mutation and intelligence expansion out of the pass and treated the exact child state as the unknown.

HINDSIGHT

The correction matrix going green did not mean the architecture was now settled. It meant the current defect cluster was understood. The fact that the programme kept moving one layer deeper after every package was precisely why the broader audit remained necessary.

EVIDENCE SNAPSHOT

- Attempt 08 gate: green
- P09-06 reruns completed
- Attempt 09: PROJECTION_CHILD_STALLED
- Correction matrix: 8/8

05-08 AUGUST 2026

The real defect was fragmented ownership

WHAT CHANGED

Across startup, scan, projection, navigation and recovery, the same pattern repeated: one component owned durable state, another inferred it, a third rendered an approximation, and a fourth tried to recover from a state it did not actually own. Each piece looked locally reasonable. Together they produced impossible combinations such as Ready while the active job was unresolved, a previous Library beside a failed current projection, and progress percentages that stopped meaning anything.

WHY I DID IT

I needed a product architecture with one semantic owner per fact and explicit translation to every consumer. The target became one versionless application-service spine and one run-scoped operational transaction, with the frontend reduced to presentation and the execution path bound to exact current state.

HINDSIGHT

This was the point where the project stopped treating complexity as evidence of seriousness. The architecture would survive only where it could explain its unique responsibility, its real consumer and what broke if it disappeared.

EVIDENCE SNAPSHOT

- One fact -> one semantic owner
- One current run lineage
- Frontend presentation only
- No screen-specific truth engines

08 AUGUST 2026

The drive taught its own architectural lesson

More retained bytes were not more proof. They were a different failure mode.

WHAT CHANGED

Generated clean rooms, copied worktrees, evidence trees and build residue had filled the external work drive. A bounded cleanup deleted 37,277,818,468 logical bytes across 131,691 files and 26,499 directories in 119 classified entries, with zero deletion failures. Free space rose by 171,923,472,384 physical bytes - about 160.12 GiB - because the exFAT volume was charging enormous allocation overhead for vast numbers of small files.

WHY I DID IT

The immediate outcome was space. The deeper outcome was another proof that preserving every reproducible byte is not safety. It can make the environment slower, more fragile and harder to reason about. I retained unique evidence and exact survivors, then backed up and moved the work substrate to NTFS so the filesystem stopped multiplying small-file cost.

HINDSIGHT

The deletion numbers looked absurd because the storage representation was absurd. That distinction matters. I had not discovered free data; I had stopped paying one-megabyte blocks for tiny files. The same one-fact/one-owner principle applies to disks: retain the unique truth, not every physical representation that ever touched it.

EVIDENCE SNAPSHOT

- 37.28 GB logical deleted
- 131,691 files
- 119 classified entries
- 160.12 GiB physical space recovered
- 26,499 directories
- 0 failures

PART 09 RETROSPECTIVE

What this phase actually changed

The first week of August changed the diagnosis. The problem was no longer that one ambitious intelligence pass had a few rough edges. The product had accumulated enough duplicated state, lifecycle vocabulary and validation ceremony that locally correct pieces could assemble into globally impossible behaviour. The only credible route forward was to simplify the ownership topology before asking the package to carry anything else.

WHAT I WOULD DO DIFFERENTLY

- 01 I would have recognised the repeated adjacent failures as an architecture signal one package earlier.
- 02 I would have treated generated storage as temporary by default before exFAT turned small files into a 160 GiB tax.
- 03 I would have required one owner for lifecycle vocabulary before managed-runtime packaging began.
- 04 I would have stopped using the phrase near-release while the normal user journey remained unproven.



PART X

ONE SPINE, ONE OWNER

09-13 AUGUST 2026

The repository was pulled apart until every surviving runtime fact had one owner, one route and one reason to exist.



ARCHITECTURAL AGE 10

SORTCEROR / DEVELOPMENT TIMELINE

09 AUGUST 2026

The architecture audit begins from zero entitlement

WHAT CHANGED

I challenged the product as though none of its infrastructure deserved to survive. Every runtime service, bridge method, plan family, route owner, recovery mechanism, test fixture and package surface had to show a real current consumer and a unique responsibility. Historical importance was not enough.

WHY I DID IT

The target was not a prettier architecture diagram. It was a smaller product that could be held in one mental model: Workspace -> ScanRun -> ImmutableSnapshot -> ResolutionRun -> SortPlan -> Decisions -> ExecutionJournal -> Verification -> UndoRun.

HINDSIGHT

The audit preserved capability rather than code shape. Useful behaviour could migrate behind the new spine; obsolete owners could become test donors or history. This avoided the two stupid extremes: keep everything because it once worked, or rewrite everything because the current form was ugly.

EVIDENCE SNAPSHOT

- Capability preservation before deletion
- One versionless service seam
- One operational transaction
- No broad rewrite authorised

09-10 AUGUST 2026

Projection lifecycle gets one current lineage

WHAT CHANGED

The projection path was reconciled so workspace, scan, audit, resolver child, publication, snapshot and request identity formed one exact lineage. Equal-looking revision numbers from another run could no longer pass as current. Late responses could not resurrect stale frontend state, and a failed child could not silently inherit a successful parent label.

WHY I DID IT

Most visible progress and navigation defects were really lineage defects. The interface was rendering data without being able to prove it belonged to the current user action. Exact lineage made stale suppression and recovery mechanical rather than visual guesswork.

HINDSIGHT

This was not glamorous, but it deleted whole categories of impossible state. Once every result had to identify the run that produced it, many defensive branches became unnecessary. The product stopped asking, "does this look recent?" and started asking, "is this the exact current fact?"

EVIDENCE SNAPSHOT

- Exact run lineage
- Child/publication identity bound
- Stale response suppression
- No numeric-revision shortcut

10-11 AUGUST 2026

The brain and engine stop being separate products

WHAT CHANGED

Resolver, evidence, planning and maintenance behaviour were reconciled behind the application service rather than exposed as competing packet, plan and audit families. Existing intelligence was retained; normal product callers stopped reading its internal representations directly.

WHY I DID IT

The aim was to keep the deep intelligence while removing the requirement that every screen and executor understand how the intelligence was manufactured. The application service would own the current answer. Internal packets, hypotheses and provenance remained available where they had an actual evidence or review consumer.

HINDSIGHT

This distinction later became crucial in the performance work. A representation can be useful and still be the wrong permanent operational shape. "A current screen reads it" is necessary evidence for retention, not automatic permission to keep an expensive topology forever.

EVIDENCE SNAPSHOT

- Resolver remains primary intelligence owner
- Internal packets hidden from normal UI
- Application service owns product answer
- Execution authority unchanged

12 AUGUST 2026

The producer, frontend and lifecycle closure lands

WHAT CHANGED

The first major closure completed with a 205-of-205 contract ledger, 30-of-30 acceptance gates and eight-of-eight final sections, with no open or grounded-not-applicable residue. The installed operation surface was reduced to 69 real product operations plus five consumer-backed host utilities.

WHY I DID IT

The point was to make the product one coherent client of one service spine. The work removed nineteen zero-consumer public facades, reduced AppServices public methods from 121 to 102, route owners from two to one, frontend store slices from nineteen to nine, canonical frontend modules from fifteen to eleven, and package/static representations from sixty to forty-four.

HINDSIGHT

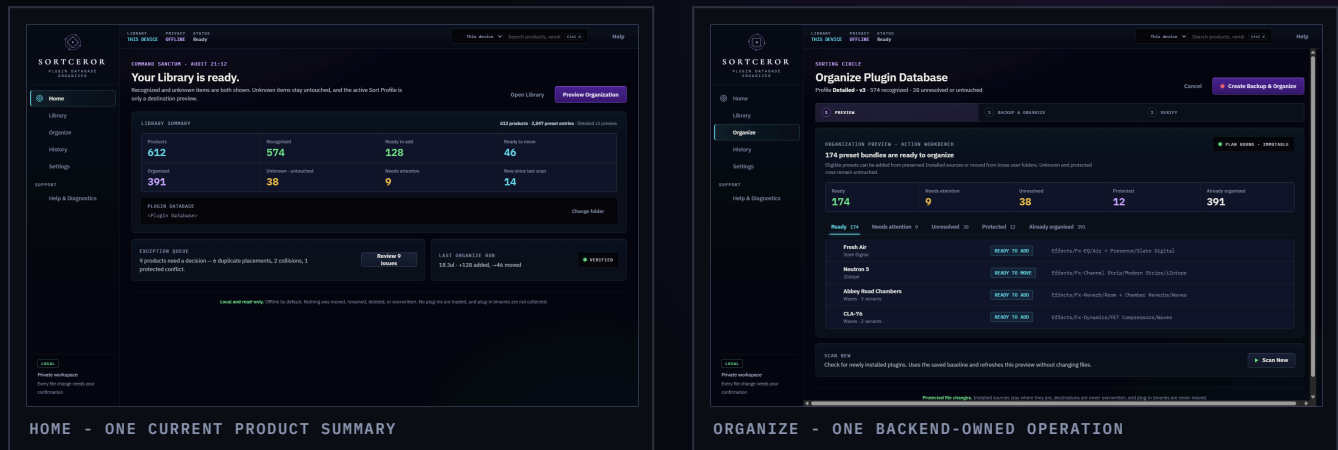
The net permanent authored product source fell by 5,084 lines. That number mattered because capability was preserved while ownership shrank. This was not line-count golf; it was evidence that the product could do the same work with fewer places to disagree.

EVIDENCE SNAPSHOT

- Contract ledger: 205/205
- Installed operations: 69 + 5 host utilities
- Focused proof: 237 passed
- Acceptance gates: 30/30
- Permanent product source: -5,084 lines

The interface stopped inventing state

One pure presentation derivation now drove visible text, Scan Sigil state, progress, severity, ARIA output and enabled controls. The frontend could remember focus and navigation. It could not manufacture lifecycle truth.



BLOCKED ≠ INTERRUPTED ≠ SUPERSEDED

UNAVAILABLE ≠ IDLE

TERMINAL PROGRESS DOES NOT RESET TO ZERO

12-13 AUGUST 2026

Validation was forced to justify its own existence

WHAT CHANGED

The next closure turned on the test and evidence apparatus itself. Every current surface was classified, stale assertions were compared against current product truth, selected-test support became an exact identity, and broad CURRENT validation was demoted from ritual to escalation.

WHY I DID IT

The project had spent too much time rerunning enormous suites because a document, helper or stale test changed. The new rule was causal: identify which evidence the changed bytes invalidate, run that evidence, and stop. A failing test proved disagreement, not automatically a product requirement.

HINDSIGHT

One final CURRENT lane still took 1,360.21 seconds against a 720-second target: 2,264 passed and 47 skipped, with the integration subset accounting for almost 89 percent of the time. I kept the miss honest. Green correctness did not magically turn a two-times runtime failure into success.

EVIDENCE SNAPSHOT

- Current lane: 2,264 pass / 47 skip
- Target: <720 s - missed
- No broad rerun after unchanged inputs
- Wall: 1,360.21 s
- Integration share: ~88.8%

13 AUGUST 2026

Anti-Ouroboros becomes development law

WHAT CHANGED

The project formalised a new pass shape: freeze the baseline, complete in-scope discovery, inventory defects, cluster shared causes, design once, implement in dependency order, run cheap causal tests, freeze final bytes, validate once, seal evidence last, perform one independent sanity audit and move on.

WHY I DID IT

This was a direct response to weeks of audit -> fix one -> re-audit -> fix another loops. Each loop recreated context, reran expensive proof and generated another layer of temporary machinery. The process looked rigorous while it delayed the external outcome.

HINDSIGHT

The law did not weaken safety. It separated load-bearing safety from evidence theatre. Live mutation, authority, data integrity and rollback still demanded exact proof. A local reversible code correction did not need forensic chain-of-custody machinery merely because such machinery could be imagined.

EVIDENCE SNAPSHOT

- Discovery before mutation
- Causal validation tiers
- One independent sanity audit
- One coherent defect inventory
- Evidence sealed last

PART 10 RETROSPECTIVE

What this phase actually changed

By 13 August the runtime and the development process had both been simplified around the same idea: one fact, one owner, everything else derived or disposable. The product kept its intelligence, interface, profiles, evidence and safety behaviour. What it lost was the fiction that every historical representation deserved permanent citizenship.

The product did not need more guardians. It needed fewer places capable of telling different stories.

WHAT I WOULD DO DIFFERENTLY

- 01 I would have imposed one current lineage and one service seam before the first twelve-screen cockpit.
- 02 I would have treated public methods without current consumers as defects, not harmless flexibility.
- 03 I would have made CURRENT an escalation lane from the day the test suite became expensive.
- 04 I would have written the Anti-Ouroboros law after the first repeated correction loop, not after the twentieth.



BURNING THE BUREAUCRACY

13-20 AUGUST 2026

The architecture was challenged by credible replacements, the context was rebuilt around current truth, package inputs were frozen, and the hardening machinery was told it did not get to become the product.



ARCHITECTURAL AGE 11

13 AUGUST 2026

The surviving architecture goes on trial

Apparently I had built the best possible architecture across the board. That smelled like shit.

WHAT CHANGED

I launched a forest-level dominance review with a deliberately hostile premise: assume nothing survives. For every generic infrastructure domain, compare deletion, absorption, platform primitives, existing dependencies, mature libraries and bounded replacement against the current bespoke mechanism. Prototype where uncertainty is empirical. Count whole-system owners, states, dependencies, packaging burden and defect classes - not just lines of code.

WHY I DID IT

The purpose was to avoid fossilising the refactor merely because I had spent a lot of time on it. If a mature mechanism could delete a thousand lines and three owners without creating a worse package, the current implementation deserved to die.

HINDSIGHT

The first result claimed all six domains were already dominant, with no replacements and zero product delta. It was tidy, comprehensive and implausibly flattering. I rejected the result as insufficiently adversarial rather than rewarding it for being convenient.

EVIDENCE SNAPSHOT

- Six architecture domains examined
- Initial verdict: all dominate
- Initial replacements: none
- Result challenged before context freeze

14 AUGUST 2026

The clean verdict fails closed

WHAT CHANGED

A fresh contrary review reproduced the evidence and rejected the easy pass. Seven findings remained blockers and one remained material. Candidate universes, quality vectors, platform-native prototypes, lightweight framework comparisons, package-architecture comparisons and whole-system scorecards were incomplete or insufficiently bound.

WHY I DID IT

The correct response was not to defend the original report. I preserved it as superseded evidence, marked the programme fail-closed and resumed the same bounded correction lineage. No product claim was allowed to inherit authority from the false pass.

HINDSIGHT

This was a major cultural win. The project had enough process machinery that it could easily have used its own artifact volume to overwhelm a challenge. Instead, the contradiction was made part of the record. A clean-looking audit that does not survive hostility is not a clean audit.

EVIDENCE SNAPSHOT

- Contrary review: FAIL_CLOSED
- Original evidence preserved as superseded
- 7 blockers + 1 material finding
- No architecture freeze on false result

14 AUGUST 2026

The corrected review closes after bounded replacements

WHAT CHANGED

The candidate work was completed, the missing comparisons were run, and bounded replacements were implemented where they genuinely dominated. The final independent validator passed 74 of 74 checks. The architecture closed as PASS_AFTER_BOUNDED_REPLACEMENTS with no major cross-owner pivot required.

WHY I DID IT

This was the outcome I actually wanted: not "everything survives" and not "rewrite the product," but a smaller architecture that had defeated serious alternatives on whole-system evidence. The retained decisions included explicit revisit conditions so current dominance would not become permanent religious law.

HINDSIGHT

The pass also proved that an architecture review can be adversarial without becoming another endless programme. Once the missing evidence and bounded replacements closed the actual blockers, the correct action was to stop.

EVIDENCE SNAPSHOT

- Terminal validator: 74/74
- Major architectural pivot: none
- Outcome: PASS_AFTER_BOUNDED_REPLACEMENTS
- P03C remained conditional

14 AUGUST 2026

I stopped rediscovering my own project

WHAT CHANGED

After the architecture settled, I rebuilt the durable project context around current source truth. Twenty-three existing notes were censused; twelve were refreshed, nine reverified, six appended and one superseded. Current architecture, programme routing, subsystem ownership and retrieval canaries were reconciled in one place.

WHY I DID IT

The point was not to create a second source of truth. Durable context would narrow future discovery and point to current owners. Git, SQLite, source, packages and physical behaviour remained authoritative. The context existed to stop expensive archaeology, not replace reality.

HINDSIGHT

This became the Established-Architecture Delta Verification rule: recover the accepted owner and flow, verify only causally relevant drift, isolate the exact unknown, run the cheapest direct proof, and reopen architecture only when current evidence actually contradicts it.

EVIDENCE SNAPSHOT

- Existing notes: 23
- Reverified: 9
- Superseded: 1
- Refreshed: 12
- Appended: 6
- Terminal gates: 84/84

15 AUGUST 2026

P04 freezes what can actually ship

WHAT CHANGED

The package-source and environment pass classified the exact distributable runtime inputs, private and build-only exclusions, canonical knowledge database, generated assets, legal notices, hidden imports, native and managed trust members, dynamic libraries, hooks, wheelhouse, Python runtime and packager recipes. The final package source contained 150 exact members.

WHY I DID IT

A source checkout and a dependency lock are not a package. P04 existed to make every build input explicit before the packagers touched it. PyInstaller was the practical reference lane; Nuitka remained a comparative lane at this historical boundary.

HINDSIGHT

The freeze also separated source identity, environment identity, recipe identity, build execution identity and physical output identity. Those are related facts, not interchangeable hashes. This later made package defects attributable instead of mystical.

EVIDENCE SNAPSHOT

- Package-source members: 150
- PyInstaller reference lane
- No package built during P04
- Operational schema: v18
- Nuitka comparative lane

15-16 AUGUST 2026

The freeze had a blind spot: it had never baked the recipe

WHAT CHANGED

Before the first real build, the frozen environment had to be corrected so `Path.home()` resolved to an empty synthetic profile rather than the real user profile. That correction re-bonded the environment and recipe identities without changing product bytes. It closed a packaging precondition that source analysis alone had missed.

WHY I DID IT

The deeper flaw was structural: P04 could prove that two recipe definitions agreed with each other without executing either packager. That meant a recipe could be perfectly frozen and still exclude a module the runtime imported, or point the package at a development-root artifact.

HINDSIGHT

This did not make the freeze useless. It defined the exact object that the physical build could falsify. The lesson was narrower: a manifest can prove what it says; only the package can prove whether what it says is sufficient.

EVIDENCE SNAPSHOT

- Synthetic `USERPROFILE` required
- Product bytes unchanged
- Build still required as external proof
- `Path.home()` bound to isolated profile
- Recipe identities advanced

18-20 AUGUST 2026

Beautiful Simplicity becomes hard law

Maximum ceremony is not maximum safety.

WHAT CHANGED

The development doctrine was rewritten around the external outcome, actual causal bottleneck, cheapest decisive falsification, full intervention space and smallest sufficient intervention. Deletion, acceptance, deferral, isolation, bypass, absorption, platform mechanisms and existing project owners all came before building another abstraction.

WHY I DID IT

The aim was to stop the project from hardening the hardening. A safeguard that generated its own recurring problems would be simplified or deleted before another safeguard was wrapped around it. Assurance would scale with consequence, blast radius, uncertainty and irreversibility rather than defaulting to maximum ceremony.

HINDSIGHT

This was not minimalism as style. It was minimum whole-system complexity: fewer semantic owners, representations, durable states, dependencies, lifecycle machines, permanent artifacts and recurring instructions. The test became brutally practical: would a competent team, knowing everything I now knew, deliberately choose to own this machinery?

EVIDENCE SNAPSHOT

- DELETE / ACCEPT / DEFER first
- Proportional assurance
- Stop when the outcome is done
- BUILD exceptional
- Temporary by default

19 AUGUST 2026

ATTENTION_ONLY finally means what it says

ATTENTION_ONLY did not need a new system. It needed to be admitted by the system that already owned it.

WHAT CHANGED

The organiser already distinguished ACTIONS_READY, NO_CHANGES_NEEDED and ATTENTION_ONLY. Two admission sites in live activation incorrectly treated every zero-action plan as though it had to be NO_CHANGES_NEEDED, which blocked maintenance-baseline adoption whenever unresolved items still needed attention.

WHY I DID IT

I changed exactly those two existing sites to admit both legitimate zero-action outcomes. ATTENTION_ONLY remained ATTENTION_ONLY. Unresolved knowledge stayed unresolved, move and execution authority stayed unchanged, and the safe filesystem/observation baseline could still be established for the next Scan New.

HINDSIGHT

The final correction was two tuples, four product lines, one focused regression and no new state, cache, schema or owner. It is one of the purest Beautiful Simplicity examples in the record: the architecture already contained the answer; the bug was a gate that had forgotten half of it.

EVIDENCE SNAPSHOT

- Commit: f3acdad
- +4 product lines
- Move/execution authority unchanged
- Two admission sites
- Schema: v18 unchanged

PART 11 RETROSPECTIVE

What this phase actually changed

This phase made the project intellectually smaller. The architecture survived a real dominance challenge, durable context stopped forcing repeated rediscovery, package inputs became explicit, and the development law finally put outcomes ahead of machinery. The most important result was not another control artifact. It was permission to stop creating them when an existing owner or a four-line correction was enough.

WHAT I WOULD DO DIFFERENTLY

- 01 I would have made package execution part of the source-freeze falsification instead of waiting for the next gate.
- 02 I would have challenged the first all-domains-dominant verdict before it reached a terminal report.
- 03 I would have separated durable context from current status much earlier.
- 04 I would have adopted Beautiful Simplicity before the project learned to build bureaucracy faster than product.



THE PACKAGE IS THE PRODUCT

16-22 AUGUST 2026

The source was finally forced through real packagers, real runtimes and a complete disposable producer journey. Every missing file and twelve-minute resolver was now a physical fact.



ARCHITECTURAL AGE 12

16 AUGUST 2026

The first real package dies on one frozen exclusion

WHAT CHANGED

Attempt 10 reconstructed the frozen source and environment, activated the build key and produced a real PyInstaller package in 29.6 seconds. The package contained 286 members and about 90.04 MB. Twenty-two of twenty-three physical, trust, legal and confidentiality gates passed.

WHY I DID IT

All four runtime commands then failed at the same import chain because `sortceror.gui_prototype.safety` had been explicitly excluded from the recipe while `builder.py` imported it at module load. The PyInstaller warning file stated the contradiction directly. The Nuitka recipe carried the same exclusion, so this was a shared recipe defect rather than a packager defect.

HINDSIGHT

The package did exactly what physical proof is supposed to do: it found the one line that every source-only gate had permitted. I stopped the lane, preserved the package and corrected the package recipe through the proper owner rather than editing the frozen build in place.

EVIDENCE SNAPSHOT

- Build time: 29.6 s
- Physical gates: 22/23
- Network: 0
- Package: 286 members / ~90.04 MB
- Failure: missing imported safety module

16 AUGUST 2026

The consumer runtime tried to reach back into D:

WHAT CHANGED

A later package advanced farther and then failed because runtime code attempted to load the package manifest from a development/build-root path on D:. The exact classification mattered: either the manifest was genuinely runtime-required and omitted, build-only data was being read at runtime, or a packaged equivalent existed and path resolution was wrong.

WHY I DID IT

I traced the call chain instead of blindly copying another file into the package. The permanent invariant was simple: a consumer package must be self-contained and must not depend on repository, worktree or build-root files existing on the development machine.

HINDSIGHT

This was another case where the easiest apparent fix - bundle the missing file - could have cemented the wrong ownership. Package construction data and consumer runtime data are different responsibilities. The product should carry the exact runtime fact it needs, not the build system that happened to know it.

EVIDENCE SNAPSHOT

- Consumer dependency on D: rejected
- No source-root fallback
- Build-time and runtime ownership separated
- Package self-containment required

PHYSICAL PACKAGER COMPARISON

PyInstaller and Nuitka finally had to earn themselves

Both lanes used the same final source and declared package inputs. The comparison measured the actual cost of owning each architecture.

	PYINSTALLER	NUITKA STANDARD
Build A / B	28.83 s / 27.02 s	891.97 s / 858.76 s
Package	90,186,449 B / 288 members	152,886,949 B / 227 members
Executable	11,067,963 B	76,408,320 B
Startup to bridge	2,859.3 ms median	2,835.2 ms median
Representative workload	10.778 s	10.805 s
Private memory	712.0 MB	712.0 MB
Reproduction	286 / 288 byte-identical	225 / 227 byte-identical
Permanent burden	Python build environment	MSVC + SDK + SCons + helper cache + jobs tuning

WINNER

PYINSTALLER

Nuitka was roughly 30 times slower to build, 69.5% larger and operationally heavier, while normal startup and the representative workload were effectively the same.

16 AUGUST 2026

Compiling Python did not move the real bottleneck

Sortceror was not slow because Python had failed to become C.

WHAT CHANGED

Nuitka improved only the interpreter and first-party import slice, which accounted for about 28.3 percent of startup. The rest was managed-runtime loading and WebView2 bridge/DOM work shared by both packages. Its small gain in the compiled slice was lost elsewhere, leaving normal startup effectively tied.

WHY I DID IT

The comparative lane had now answered its question. Maintaining two packagers would add a compiler toolchain, Windows SDK, helper cache, memory tuning and much longer feedback without improving the producer outcome. PyInstaller became the required primary architecture; Nuitka remained historical capability, not a parity obligation.

HINDSIGHT

This was a textbook dominance decision. Nuitka did reduce exposed Python bytecode, but both packages still carried the distributable knowledge database and visible runtime strings. Partial extraction resistance did not justify owning a second build architecture before commercial hardening had a concrete requirement.

EVIDENCE SNAPSHOT

- PyInstaller selected
- Nuitka package: +69.5%
- Nuitka permanent lane paused
- Normal runtime delta: negligible
- Build time: ~30x

17-20 AUGUST 2026

The full 8,304-entry producer run exposed the real cost

WHAT CHANGED

The six-stage producer generated 65,496 identity-hypothesis rows, 10,916 evidence packets, 139,854 bindings and 8,304 final decisions. A one-line score-clamp correction changed legal-input rejection from 148 of 303 to zero and allowed the golden path to complete.

WHY I DID IT

The cold run then took 750.955 seconds end to end: 21.257 seconds for audit/scan, 715.647 for the resolver, 8.072 for compile and 5.731 for projection. The resolver missed its sixty-second gate by 11.93 times.

HINDSIGHT

This was not permission to rewrite the resolver. It was the first honest full-scale measurement of where time went. The next question became which work was causally necessary for changed observations, which work was run-global, and which durable rows existed only because the current representation happened to materialise them.

EVIDENCE SNAPSHOT

- Hypotheses: 65,496
- Bindings: 139,854
- Resolver: 715.647 s
- Packets: 10,916
- Decisions: 8,304
- Total: 750.955 s

17-20 AUGUST 2026

The decision distribution was mostly obvious - the work was not

Scale is not intelligence, and persistence is not free merely because SQLite accepts the rows.

WHAT CHANGED

Of the 8,304 decisions, 8,164 were RESOLVED_EXACT_GLOBAL. Only 140 fell into ambiguous, unknown, local-intelligence or likely-version-sibling outcomes. The expensive path was therefore spending most of its time around products whose accepted identity was already exact.

WHY I DID IT

I tested the large persistence families rather than assuming impressive row counts were useful. Some apparently redundant claims and hypotheses did have a live Product Detail consumer. That proved they could not simply be deleted. It did not prove the current normalised representation was the best permanent design.

HINDSIGHT

The product distinction became operational provenance versus research candidate graph. An exact known product may need an explanation. It does not automatically need to persist a miniature relational research dataset every time the library is rebuilt.

EVIDENCE SNAPSHOT

- Exact global: 8,164
- Unknown: 38
- Likely version sibling: 20
- Ambiguous: 52
- Local intelligence: 30

20-21 AUGUST 2026

Two interventions dominate the cold path

WHAT CHANGED

The cheapest successful direction was not a new cache architecture. Unchanged observations could reuse prior resolver semantics, while truly new or dependency-invalidated observations took the expensive path. Run-global immutable work was built once rather than reconstructed per observation. A crash in the optimisation could still fall back to a correct full resolve.

WHY I DID IT

This preserved the existing durable job and operational database owners. No second recovery cache, registry or lifecycle was introduced. A later 40-plus-one proof showed the intended shape: forty unchanged observations reused prior semantics and one genuinely new observation alone entered the expensive resolver work.

HINDSIGHT

The performance programme produced an acceptance-shaped candidate around 187.224 seconds and a canonical warm gate around 57-58 seconds. Those results did not make every future package fast; they established the accepted mechanisms that later source must continue to use.

EVIDENCE SNAPSHOT

- Reuse unchanged semantics
- Run-global immutable work hoisted
- Warm gate: ~57-58 s
- Resolve only new/invalidated work
- No new durable cache owner

20 AUGUST 2026

A reproducible PyInstaller candidate is finally frozen

WHAT CHANGED

The required PyInstaller A/B lane produced a 288-member package of 90,111,662 bytes. Both builds passed four of four smokes, twenty of twenty physical proofs and ten of ten reproducibility checks. One immutable candidate was frozen for packaged consumer acceptance.

WHY I DID IT

The candidate was not declared a product merely because the build was coherent. PCA remained blocked on a known product issue, and the package stayed immutable while that issue was investigated. The build and the consumer journey were deliberately separate authorities.

HINDSIGHT

This was the first point where the project could stop saying "the source should package" and point at an exact executable tree. It was also the point where every subsequent claim had to be tied to the package lineage actually exercised, not to whatever source happened to be newest.

EVIDENCE SNAPSHOT

- Package members: 288
- Smokes: 4/4 each
- Reproducibility: 10/10
- Package bytes: 90,111,662
- Physical proof: 20/20 each
- PCA: blocked on known issue

21 AUGUST 2026

The simplicity law itself had been truncated

The hardening had managed to break the hardening. Naturally.

WHAT CHANGED

The canonical Beautiful Simplicity implementation file was discovered to contain only fifty-nine lines, while downstream projections claimed a complete rule set. I repaired the canonical owner and regenerated the dependent copies through their existing path. All twenty-three current skills then reconciled green.

WHY I DID IT

The irony was useful. The project had just adopted a law against duplicated representations and immediately found the law itself split across inconsistent copies. The correct fix was not another synchronisation framework. It was one canonical owner and derived projections.

HINDSIGHT

A governance mechanism does not become exempt from its own architecture. If it needs permanent machinery merely to keep its copies agreeing, the mechanism has failed the same retention test as product code.

EVIDENCE SNAPSHOT

- Canonical file found at 59 lines
- Single owner repaired
- Derived projections regenerated
- 23/23 current skills green

22 AUGUST 2026

The complete disposable producer journey passes

WHAT CHANGED

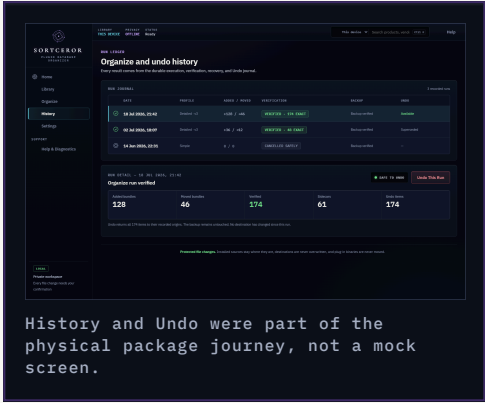
A later immutable PyInstaller lineage completed the full internal copied-root journey: the 8,304-entry first run, review, planning, 241 physical moves, verification, History, exact Undo, Scan New, persistence and relaunch. The final relaunch reached usable Home in 2.956 seconds, displayed correct Home and History Undo truth, reported no typed failures and shut down cleanly.

WHY I DID IT

The copied root made real movement and exact rollback testable without granting authority over the live user library. This was the first end-to-end proof that the product could carry intelligence all the way through a physical reversible outcome in a package.

HINDSIGHT

The scope remained explicit. Internal copied-root acceptance was complete. Live-user acceptance, installer behaviour and distribution hardening were not. A safe success in one authority domain does not silently inherit permission in the next.



EVIDENCE SNAPSHOT

- Source: b774b2b8...
- Executable: 9c382374...
- Physical moves: 241
- Usable Home: 2.956 s

- Build input: 0693efaf...
- Decisions: 8,304
- Exact Undo: passed

22 AUGUST 2026

The package had finally done the original job

Only took four billion thousand seven hundred and ninety leventeen tokens to get here.

WHAT CHANGED

For the first time, the project could state the original promise in physical terms: observe a real Plugin Database shape, resolve the entries, show the producer what would happen, move the authorised set, verify the result, retain History and restore the exact prior state.

WHY I DID IT

That did not erase the preceding failures. It made them legible as the cost of earning a reversible consumer transaction instead of stopping at a clever sorter. The original May scripts had backups and logs. The August package had identity, authority, durable state, verification, recovery and a user-facing journey around the same physical act.

HINDSIGHT

This was the emotional high point of the internal programme and still not the release. The copied-root package proved that the product architecture could work. It did not prove that the next visual/state revision, the ordinary default profile or the live user machine would preserve the same result.

EVIDENCE SNAPSHOT

- Original outcome achieved on copied root
- Live root untouched
- Installer/distribution: not authorised
- Package immutable during proof
- Provider/network calls: 0

PART 12 RETROSPECTIVE

What this phase actually changed

This phase converted abstract package readiness into physical outcomes. The packager comparison eliminated a costly second lane, the full-scale resolver measurement exposed the real bottleneck, reuse and run-global ownership produced the right optimisation shape, and the copied-root package finally performed 241 verified reversible moves. It was a genuine milestone - and still only an internal one.

WHAT I WOULD DO DIFFERENTLY

- 01 I would have executed one real packager during the source/environment freeze instead of letting recipes certify each other.
- 02 I would have selected PyInstaller earlier and treated compilation as a commercial-hardening experiment rather than a release prerequisite.
- 03 I would have run full-scale resolver cardinality before celebrating smaller replay timings.
- 04 I would have kept internal copied-root acceptance and ordinary default-profile acceptance visibly separate from the first successful journey.



CONSUMER TRUTH, AGAIN

23-26 AUGUST 2026

The product finally looked like itself, passed a deep internal journey, and then failed before I could select a folder. The record ends where the package is being made honest again.



ARCHITECTURAL AGE 13

23-24 August 2026

The product finally looked like Sortceror

WHAT CHANGED

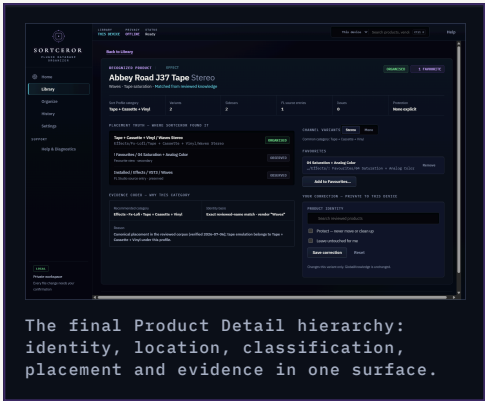
I completed the product-wide visual system: IBM Plex Sans and Mono, the Cormorant wordmark, the obsidian surface ladder, restrained violet and cyan semantics, machined controls, the master sigil, the full operational Scan Sigil, Product Detail hierarchy, History, Search, Library and Organize. The accepted design removed generic SaaS cards and made dense evidence feel like one instrument.

WHY I DID IT

The interface had to communicate depth without exposing internal bureaucracy. Normal mode would show outcomes and clear actions. Product Detail and Behind the Curtain could reveal evidence, relationships and authority when requested. Mutation language stayed explicit even when the atmosphere leaned hard into the arcane identity.

HINDSIGHT

This visual work mattered. It made the product legible, distinctive and emotionally real. It also created a dangerous temptation: a coherent screen can make a partially coherent product feel finished. The next physical double-click destroyed that illusion immediately.



The final Product Detail hierarchy: identity, location, classification, placement and evidence in one surface.

EVIDENCE SNAPSHOT

- IBM Plex Sans + Mono
- Obsidian material system
- Product-wide current visual language

- Cormorant wordmark only
- Full Scan Sigil

24-25 AUGUST 2026

Product Detail J1-J9 and a new internal package go green

WHAT CHANGED

The visual/state convergence repaired fresh startup, historical schema upgrade, second launch, Product Detail, Similar Product, placement, correction, policy, favourite response-loss recovery, cleanup Undo, active-operation fences and relaunch recovery. Final PyInstaller A/B and reproducibility passed again.

WHY I DID IT

The frozen internal copied-root package contained 288 files and 90,221,198 bytes. Its executable hash began 0ea00c43 and its source was commit 2de20f65. The static package identity, build manifest and current visual frames all agreed.

HINDSIGHT

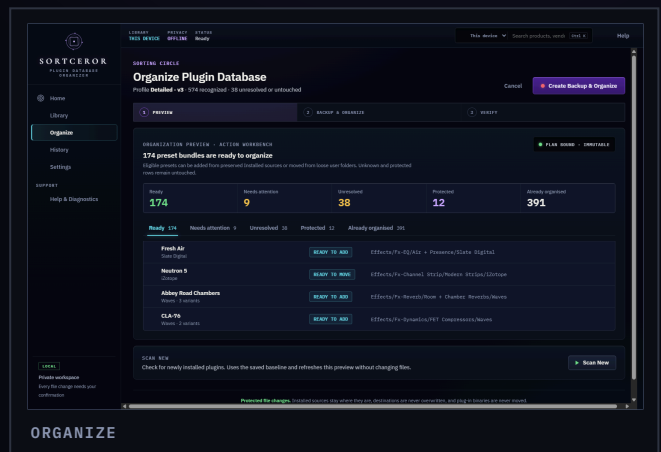
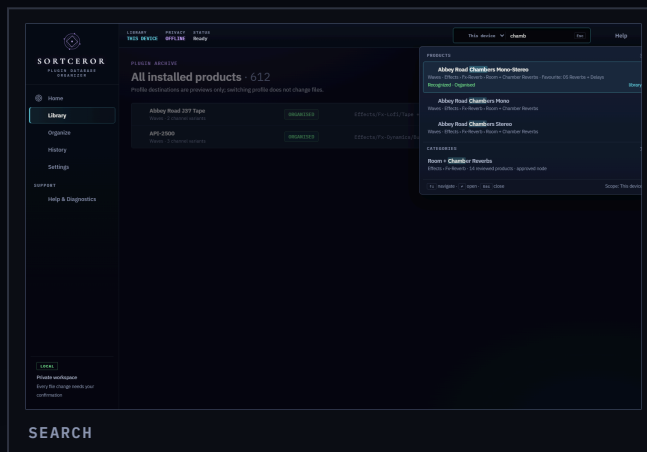
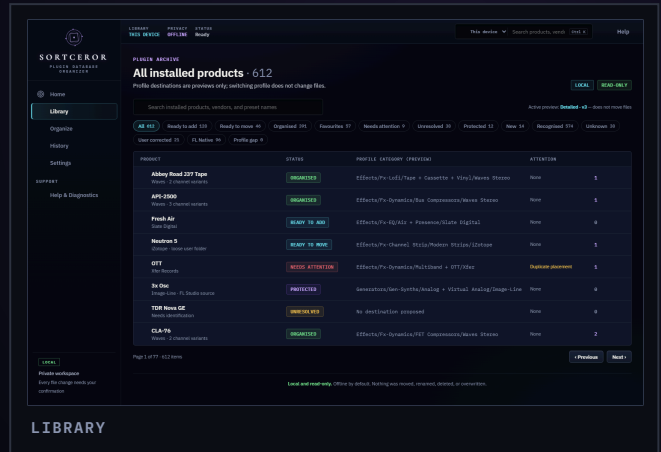
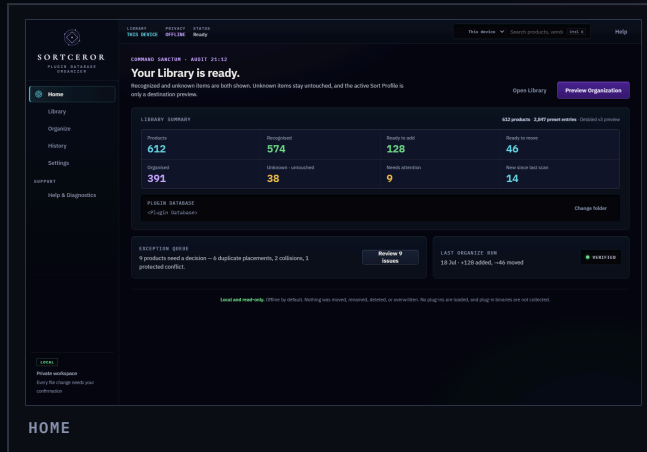
The handoff correctly kept live-user acceptance and distribution hardening locked. The mistake came afterward: the breadth of J1-J9 and the internal copied-root evidence created confidence that ordinary default-profile startup would be routine. It was not.

EVIDENCE SNAPSHOT

- Source: 2de20f6557...
- 288 files / 90,221,198 bytes
- Live-user acceptance: unperformed
- Executable: 0ea00c4397...
- J1-J9: green

THE PRODUCT-WIDE VISUAL LANGUAGE

Beautiful, coherent, and not allowed to hide broken behaviour



THE ATMOSPHERE CAN BE MAGIC. THE PRODUCT TRUTH CANNOT.



25 AUGUST 2026

I double-clicked the EXE and could not select a folder

I had built immaculate proof for an application that still could not select a fucking folder.

WHAT CHANGED

The first launch opened, but the Plugin Database control showed a red forbidden cursor and would not accept input. Repeated launches then fell into the recovery screen. This happened after an almost nine-hour internal convergence run that had described the package as functional and frozen.

WHY I DID IT

The only useful response was to revoke the package gate immediately. No amount of J1-J9 detail mattered if the first normal user action was impossible. The screen recording and diagnostics outranked the handoff summary.

HINDSIGHT

This was the most infuriating recurrence in the project because it was not an exotic edge case. I had built immaculate proof for an application that still could not select a fucking folder. The opportunity cost of internal acceptance without ordinary external proof was now impossible to ignore.

EVIDENCE SNAPSHOT

- Ordinary double-click: failed
- Repeated launch: recovery shell
- Installer: blocked
- Folder picker control: unavailable
- Package gate: reopened

25 AUGUST 2026

Persistent state was blocking the first interaction

WHAT CHANGED

The exact historical v18 predecessor reproduced the failure. Five coupled causes emerged: a migration identity conflict, whole-database validation extending the blocking startup path, close waiting behind the installed-application lock, picker readiness unnecessarily depending on full AppServices readiness, and a rejected second folder selection being able to schedule an audit against the previously configured workspace.

WHY I DID IT

I corrected those existing owners: validation was scoped to the exact v18 upgrade, SQLite work became cooperatively cancellable, shutdown made itself visible before waiting on the lock, lease cleanup ordering was fixed, the native picker was decoupled from full application readiness, and audit scheduling required one successful current configuration.

HINDSIGHT

No new runtime framework or second owner was added. The defect cluster was a textbook example of why the first-interaction path must be isolated from long maintenance work. A producer should be able to choose a folder while a historical database upgrades in the background.

EVIDENCE SNAPSHOT

- v18 migration identity conflict
- Close waited behind app lock
- Rejected B could schedule audit against A
- Whole-DB validation blocked startup
- Picker coupled to full readiness

25 AUGUST 2026

The successor launches and the picker becomes real

WHAT CHANGED

The corrected package contained 291 members and 88,065,458 bytes; its executable hash began 07229659. On the active maintenance specimen, the shell became visible in 2.30-3.04 seconds and the picker was usable about 0.95-0.99 seconds later. It opened, cancelled, reopened and selected an owned empty folder while migration remained active.

WHY I DID IT

Close during migration exited in 2.72-2.95 seconds, released ownership, produced no WAL residue and rolled the interrupted upgrade back to a recoverable schema boundary. A strict zero-argument launch reached shell in 2.369 seconds and first usable screen in 2.871; relaunch took 2.436 seconds. Four rapid launches produced exactly one visible owner.

HINDSIGHT

These were real wins. They proved the startup and ownership corrections. They did not prove the current data journey. The package remained promotion-pending rather than accepted, and the next ordinary audit proved why that distinction mattered.

EVIDENCE SNAPSHOT

- 291 members / 88,065,458 bytes
- Picker usable: +0.95-0.99 s
- Relaunch: 2.436 s
- Shell: 2.30-3.04 s
- Close during migration: 2.72-2.95 s

25 AUGUST 2026

The current Library and current audit contradicted each other

WHAT CHANGED

The successor package launched into an existing default profile, showed 8,304 entries and an older Library projection, then raised MAINTENANCE_RESOLVER_REUSE_BINDING_MISMATCH. After folder selection, audit lifecycle became unrecognised. A manual retry reached 81 percent and ended in Verification failed.

WHY I DID IT

The product was simultaneously saying "Your Library is ready," "the current maintenance projection could not be rebuilt" and "the audit could not finish." The previous Library was useful historical state, but it was not current Organize authority. The interface needed to preserve the old view without pretending the refresh had succeeded.

HINDSIGHT

The package was downgraded again. Startup, picker and process ownership were green. Default-profile audit, resolver reuse, current organization preview and Organize readiness were red or unestablished. Installer work stayed blocked.

EVIDENCE SNAPSHOT

- Maintenance reuse: binding mismatch
- Current audit: failed
- Consumer-functional package: no
- Previous Library: still browsable
- Progress: 81% terminal failure

25 AUGUST 2026

Cold fallback proved correctness and exposed a 48 minute catastrophe

Correct and fifty minutes late is still broken.

WHAT CHANGED

The stale reuse binding was classified as derived knowledge-state drift rather than cross-workspace corruption. Refusing the stale result and running a cold resolver rebuild preserved the historical database, completed all 8,304 decisions and produced a current plan and projection.

WHY I DID IT

It took 48 minutes and 56 seconds. The package was moving at roughly 2.5-2.6 observations per second. Correct fallback semantics had been proved, but the fallback was functionally unusable for a producer waiting to Organize.

HINDSIGHT

This was not a packaging-only failure. The source-side recompute was already catastrophically slow. The question was no longer whether cold recomputation was valid; it was why the current lineage had diverged from the accepted indexed, pruned and hoisted resolver path.

EVIDENCE SNAPSHOT

- Cold decisions: 8,304
- Current projection published
- Throughput: ~2.5-2.6 observations/s
- Historical state preserved
- Cold wall: 48m 56s

25-26 AUGUST 2026

This was not normal cold work

WHAT CHANGED

Retained performance evidence had already proved a 10,000-FST generated pipeline with 16.612 seconds in the resolver and 71.446 seconds total. Candidate evaluations had been reduced from 26.12 million to 1.858 million, with a maximum pool of 234 out of 2,612 and no decision-semantic or knowledge-binding mismatch.

WHY I DID IT

The current roughly fifty-minute path was therefore not a larger version of the accepted path. Its observed throughput was about 230 times slower. The only responsible diagnosis was missing or bypassed accepted optimisation, a new dominating per-observation cost, or a package/runtime path that differed from current source.

HINDSIGHT

I stopped historical hypothesis shopping. The next proof had to delta-compare the current call path against the accepted candidate index, pruning, run-global knowledge construction, reuse and hoisted immutable work. One short profile and bounded subset could answer that; another 8,304-entry diagnostic run could not.

EVIDENCE SNAPSHOT

- Accepted resolver: 10,000 in 16.612 s
- Candidate evaluations: 26.12M -> 1.858M
- Broad rediscovery rejected
- Accepted total pipeline: 71.446 s
- Current throughput delta: ~230x slower

26 AUGUST 2026

Rolling packaged remediation restores truthful progress

WHAT CHANGED

The first rolling checkpoint corrected the visible product around the active full-scale resolver: removed the purple wash, externalised the Scan Sigil script for CSP, removed the inline loader, made the backend own the 0/1 lifecycle truth, aggregated progress through the existing service and removed the premature 81 percent publication. A header line-height correction fixed clipped descenders and alignment.

WHY I DID IT

The package then showed durable resolver progress advancing from 1,028 to 4,620 to 5,851 of 8,304 while public progress moved 77 -> 78 -> 79. That closed the old 77/81 frozen-state defect. A remaining quantisation bug snapped backend checkpoints to 100 of 10,000, causing a roughly fifty-nine-second flat 79 percent while 2,064 real items completed.

HINDSIGHT

The smallest correction reduced the existing checkpoint quantum. The same observed pair that previously rendered 79.0 -> 79.0 now rendered 79.0 -> 79.9. The Scan Sigil stayed continuously alive, Cancel remained enabled, CSP stayed intact and the header stayed clean.

EVIDENCE SNAPSHOT

- Checkpoint: ac206127...
- Progress observed: 1,028 -> 4,620 -> 5,851 / 8,304
- Public cadence: 79.0 -> 79.9
- Rolling build input: 8140bf26...
- Quantisation fix: fc3fc0f7...

PART 13 RETROSPECTIVE

What this phase actually changed

The final four days repeated the project's oldest lesson at the largest scale. A beautiful, internally accepted package failed the first normal click. The startup fix then passed while the current data journey failed. The cold fallback then produced the right answer at a fifty-minute latency. By the documentary cut-off, the package was materially more truthful - startup, picker ownership, progress cadence, Cancel state, CSP and Scan Sigil motion were physically better - but the latest lineage was still under rolling consumer remediation rather than installer or live-user acceptance.

The record closes honestly: better, running, still not accepted.

WHAT I WOULD DO DIFFERENTLY

- 01 I would have made zero-argument default-profile startup the first gate of every supposedly final package.
- 02 I would have required the current data refresh, not merely a persisted previous Library, before calling Organize ready.
- 03 I would have bound resolver performance evidence to the exact source lineage so later packages could not silently bypass accepted optimisations.
- 04 I would have frozen the visual system only after the ordinary consumer journey, not allowed beautiful coherence to create release pressure.



CROSS-CUTTING RETROSPECTIVE

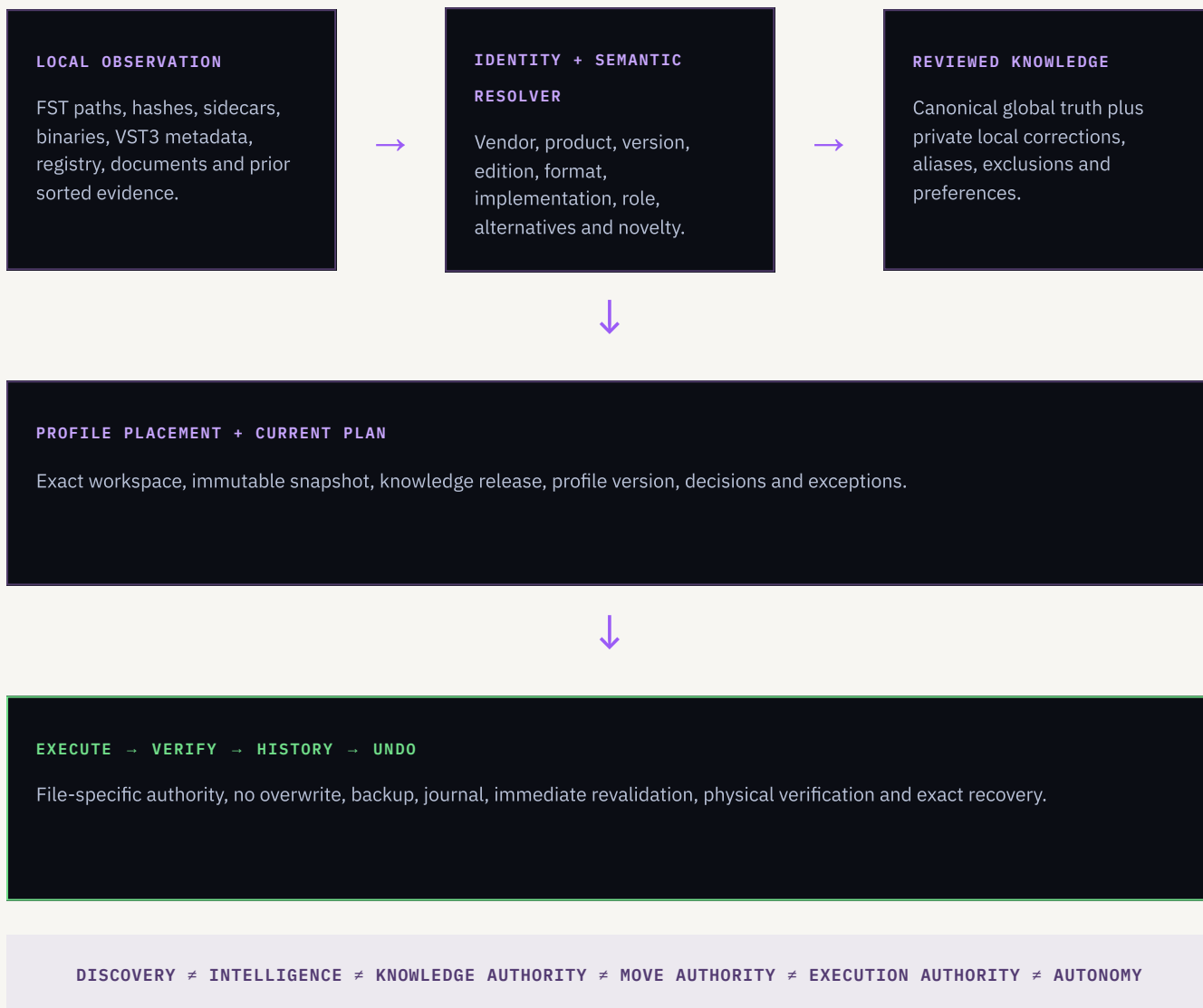
The architectural pivots that actually mattered

These were not rebrands. Each changed what counted as truth, who owned state, or what the product was allowed to do.

07 MAY	Name/category guesses	→	Evidence-backed classification	Chroma proved plausible labels can be confidently wrong.
12 MAY	Direct apply	→	Review-first mutation	Backup alone was not a legible plan.
23 MAY	Conversation memory	→	Canonical structured data	The chat history was a messy evidence pile.
12 JUN	Mixed scripts	→	Python + SQLite application	One language and one durable data owner.
18 JUN	Safe staging	→	Measured authority promotion	894 prepared decisions still imported zero knowledge.
13 JUL	Component platform	→	One product spine	A sophisticated cockpit could not complete the job.
20 JUL	Mega-validation	→	Causal validation economy	Rerunning everything was not more rigorous.
29 JUL	Automated green	→	Human package authority	The double-click journey invalidated the package.
09 AUG	Patch adjacent symptoms	→	One fact, one owner	Repeated defects exposed fragmented state ownership.
14 AUG	Architecture self-belief	→	Adversarial dominance	The first flattering audit failed closed.
18 AUG	Hardening the hardening	→	Beautiful Simplicity	Complexity had to earn permanent ownership.
22 AUG	Source correctness	→	Physical copied-root proof	241 moves and exact Undo finally closed the action loop.
25 AUG	Internal acceptance	→	Ordinary consumer truth	The package still could not select a folder.

What Sortceror is now

Evidence can flow into hypotheses. Hypotheses can flow into reviewed knowledge. Reviewed knowledge can flow through a Sort Profile into a plan. None of those layers can skip directly to execution.



BRUTALLY HONEST RETROSPECTIVE

What I got right

I separated evidence from authority.

Names, embeddings, local metadata, official research, confidence and prior folders could inform a decision without silently becoming permission.

I made mutation reversible.

Backups, source identity, no-overwrite, journalling, verification and Undo survived every architecture change.

I stayed local-first.

The machine became its own evidence corpus, and unknown third-party code stayed outside the main process.

I allowed honest failure states.

Unknown, blocked, review-only, attention-only and fail-closed states repeatedly stopped the product manufacturing certainty.

I eventually forced one product spine.

The application stopped pretending a pile of capable subsystems automatically formed one product.

I let physical evidence overrule prestige.

Large green suites, beautiful screens and accepted internal journeys were revoked whenever the actual package contradicted them.

The strongest thing in the project is not that it rarely fails. It is that the architecture gives failure somewhere honest to go.

BRUTALLY HONEST RETROSPECTIVE

What I should have done differently

01	Define the identity model before the first vendor script.	Product, version, edition, format, implementation, role and taxonomy were allowed to collapse into one string for too long.
02	Build the cold vertical slice before the cockpit.	A believable twelve-screen interface arrived before the first normal producer journey was durable.
03	Measure active learning, not prepared paperwork.	Hundreds of immaculate proposals still imported zero authoritative knowledge.
04	Design leakage-resistant evaluation before ingesting the corpus.	Closed-catalogue replay made intelligence look more general than it was.
05	Adopt retention and validation economy on day one.	The 82 GB repository, exFAT amplification and repeated mega-suites were avoidable.
06	Choose one packager early.	Nuitka consumed build time and architecture attention before it had a product outcome to justify.
07	Run the zero-argument package first.	The ordinary default-profile click should have happened before the nine-hour internal convergence victory lap.
08	Bind performance to source lineage.	An accepted fast resolver is useless if the current package can silently take a fifty-minute path.

CURRENT RELEASE TRUTH / 26 AUGUST 2026

What is accepted, what is not

ACCEPTED DURABLE ARCHITECTURE

One application-service spine, canonical SQLite ownership, evidence-separated authority, Sort Profiles, reversible execution, current visual system and the Beautiful Simplicity development law.

ACCEPTED HISTORICAL PHYSICAL PROOF

The 22 August copied-root package completed 8,304 decisions, 241 physical moves, verification, History, exact Undo, Scan New, persistence and relaunch.

LATEST ROLLING PROOF

Startup and picker ownership are repaired. The packaged full-scale resolver now publishes durable progress and a truthful moving percentage; Scan Sigil motion, CSP, Cancel and header presentation are physically improved.

NOT ACCEPTED

No current installer candidate. No latest live-user consumer acceptance. No live Plugin Database mutation authority. No public distribution, licensing or activation. The returning-user resolver path still requires terminal consumer-scale proof.

THE ABSENCE OF A CURRENT RELEASE CANDIDATE IS NOT A FAILURE TO RECORD PROGRESS. IT IS THE CORRECT PRODUCT TRUTH.

What happens next

01 Close the current resolver delta

Prove the packaged lineage is using the accepted candidate index, pruning, reuse and run-global ownership. Fix only the exact missing/bypassed path or new dominant cost. Same semantics, consumer-scale throughput.

02 Finish the ordinary packaged journey

Zero-argument startup, folder selection, current audit, current Library, Organize readiness, navigation, close and relaunch on the exact immutable successor package.

03 Build the minimal Windows installer

Use the frozen PyInstaller package as payload. No updater, licensing, activation, service or custom installer framework. Prove install, relaunch, upgrade-state preservation and uninstall behaviour.

04 Installed-product acceptance

Exercise the installed copy against fresh and copied historical state. No dist-folder assumptions, no developer paths, no hidden bootstrap dependency.

05 Live-user acceptance

Start read-only. Verify the real selected root and current plan. Grant one narrowly bounded live mutation only after backup, execution, verification, History and Undo are ready.

LATER Resume the Intelligence Programme

Only after the packaged product is durable. Improve held-out identity and semantics without using new intelligence to hide a broken consumer application.

CONCLUSION

The long road, accurately stated

Sortceror is much further along than the original folder sorter and not yet a current consumer release. Both statements are true.

The difficult bones now exist: local-first evidence, canonical product and implementation identity, reviewed knowledge, profile-separated placement, immutable planning, action-scoped authority, physical backup, journalled execution, verification, History, exact Undo, packaged Windows runtime state, a coherent visual language and an architecture capable of refusing certainty when it has not been earned.

The project has also learned what those bones do not guarantee. They do not guarantee that the package imports the file its recipe excluded. They do not guarantee the first folder picker is usable while a historical database upgrades. They do not guarantee a stale derived binding falls back quickly. They do not guarantee a green internal journey still works after the next source lineage changes.

The product wins when the producer stops managing the library - not when the repository becomes extremely good at describing why the producer still cannot.

The immediate work is therefore not another abstract rewrite and not another grand intelligence subsystem. It is to restore the accepted fast resolver path in the current package, complete the ordinary consumer journey, freeze one exact package, install it, and prove the live-user boundary carefully. Then the intelligence work can resume from a product that deserves it.

THE PRESENT-DAY RULE

No new subsystem gets to call itself progress unless it improves a real producer outcome and survives the exact packaged journey that consumes it.

Appendix A / Checkpoint Ledger

Major ledges used in the chronology

12 JUN	01c165d / 40a655c	Python/SQLite cutover and rollback safety	20 JUN	98931bb	Python is Sortceror
05 JUL	5f1e4e8	Reviewed 157 planner	15 JUL	aecbba1	Cold read-only audit accepted
16 JUL	185ddfa	Organize / Execute / Verify / Undo	20 JUL	b7483ff	Fail-closed intelligence/runtime foundation
25 JUL	c0cfd58	Consumer journey convergence	28 JUL	a7b6cad	Identity graph and evidence spine
12 AUG	4a6136b	Producer/frontend/lifecycle product checkpoint	13 AUG	d6b2d123	Validation economy closure checkpoint
14 AUG	PASS_AFTER_BOUNDED_REPLACEMENTS	Architecture dominance closed	14 AUG	CTX01_PASS	Persistent project cognition current
15 AUG	P04_PASS_DUAL_LANE_FREEZE	Package source/environment frozen	19 AUG	f3acdad	ATTENTION_ONLY baseline admission
22 AUG	ae00a7da handoff	Copied-root PCA journey accepted	24 AUG	2de20f65	Post-design internal package source
26 AUG	ac206127 / fc3fc0f7	Rolling progress and quantisation corrections			

130	VST migration leftovers	1.377 GB	first local evidence graph	154	reviewed live moves
7,026	plugin-like objects accounted	2,612	reviewed exact products	8,304	full-scale decisions
241	packaged copied-root moves	48m56s	latest cold fallback wall	59 s	flat 79% quantisation interval

Rejected and superseded packages remain historical evidence. A checkpoint proves the exact scope it closed; it does not grant future package or live-user authority.

15 development laws

01	Discovery is not intelligence.	02	Intelligence is not knowledge authority.
03	Knowledge authority is not move authority.	04	Move authority is not execution authority.
05	A token is necessary and never sufficient.	06	Unknown is a valid answer; forced certainty is a defect.
07	The frontend may expose the brain but may not invent a second one.	08	One fact gets one semantic owner.
09	Unchanged evidence is reused by identity, not regenerated for theatre.	10	A package build is not consumer acceptance.
11	Internal copied-root authority never leaks into the live user root.	12	The human double-click journey can invalidate every automated green lane.
13	Performance that makes the action unusable is a product defect.	14	Temporary machinery expires unless a recurring consumer earns it.
15	Stop when the external outcome is done.		

