AXIOVEX SYSTEMS · WHITEPAPER

VEXA

A Projected Constraints Stack by Axiovex Systems

Whitepaper v1.1 · 2026-11-05 · Axiovex Systems, LLC
Written from the ratified VEXA program record v1 (ratified 2026-11-04). Revised 2026-11-16.

Abstract

Research programs lose track of what they know. Claims accumulate in papers, notebooks, dashboards, and chat threads until nobody can answer three simple questions from the record: what does the program hold, what would change its mind, and what has it already killed? VEXA is a stack built so those questions always have mechanical answers. Every material claim lives in one append-only epistemic ledger, with its source, its confidence stated in the source's own terms, and the observation that would falsify it. Every view of the program (a matrix, a map, and a walkable three- dimensional scene) is generated from that ledger by script and is blocked from publishing whenever a verification gate fails. An AI agent works inside those views. It fetches literature, drafts claims, launches registered studies, and reads verdicts back from the record. It never issues a verdict of its own. VEXA grew out of a foundational research framework called projected constraints, and out of a governance pattern that several independent programs adopted and stress-tested before it was centralized. This paper describes the stack, the evidence for its parts, its spatial interface, its rules for honest immersion, and the gated plan by which the programs that bred it will adopt it.

Axiovex takes its name from axiom and the Latin vexare, to shake. The name is the method: shake the axioms until what is actually true falls out, then build on that ground. VEXA is that method made into software a researcher can stand inside.

1. The problem: programs lose track of what they know

A research program's knowledge does not live in one place. It lives in the paper that states a result carefully, the slide that states it boldly, the notebook that states it numerically, and the memory of whoever ran the study. Those four statements drift apart, and the drift is invisible until someone builds on the wrong one. Four failure modes recur often enough to design against.

First, claims without falsifiers. A claim that no observation could refute is not knowledge; it is decoration. In most programs nothing forces a claim to name what would kill it, so decoration accumulates beside results and is quoted with the same confidence. Second, narrative confidence quietly becomes a number. A source says "low," a spreadsheet needs a value, and 0.3 appears, carrying precision nobody measured. Third, status by summary. Verdicts get reported from summaries of summaries, and an interim or mistaken count can travel further than the record it misstates. Fourth, history gets edited. A correction made by rewriting the original entry destroys the evidence that the program ever believed otherwise, and with it the ability to learn from the error.

VEXA treats all four as engineering defects with mechanical countermeasures, specified in its constitution and requirements and checked by tests, not as matters of care or culture.

2. Three layers

VEXA is a stack of three layers, and the separation matters.

Layer 1 is Projected Constraints, a candidate foundational framework. It works in a primitive vocabulary (designated A0 to A9 in the program's records), takes a single founding axiom (one primitive dimensionful scale, because from order and counting alone only dimensionless combinations are identifiable), and has produced kernel-checked necessity results in a formal core (the formal core's theorems: order-and-count observables are invariant on rescaling orbits; no estimator built from causal-set data alone can recover absolute scale; one datum of known nonzero weight fixes the orbit). The framework assumes no candidate structure, causal sets included, is true. It identifies what must be true and where the gaps are. Its largest named gap is dynamics. VEXA depends on none of its physics being correct. The framework is in the stack because it is where the discipline was forged, under the hardest conditions the fleet works in: claims that can be killed by a computation.

Layer 2 is the programs. Independent research programs (a projected-constraints physics program, a computational decipherment program, a systems atlas, and others in engineering and workforce domains) each keep their own ledger, their own data, and their own verdicts, in their own repositories, under their own direction. Nothing about a program's truth centralizes. What is shared is tooling and discipline.

Layer 3 is the stack proper: spec-kit for the development lifecycle; the Applied Epistemic Engineering engine (published, version 1.6.0) for claim evaluation; a provider-neutral evaluator contract; the epistemic ledger as a maintained product (version 0.15.0); the Tekmerion study-execution surface; and the pattern that binds them, called the winning combination: one append-only ledger as the sole entry point, a matrix generated from the ledger, a layered map as the primary surface, an executable engine that computes the matrix's implications, and a verification gate that blocks publication on failure. Truth flows one way: ledger, matrix, map.

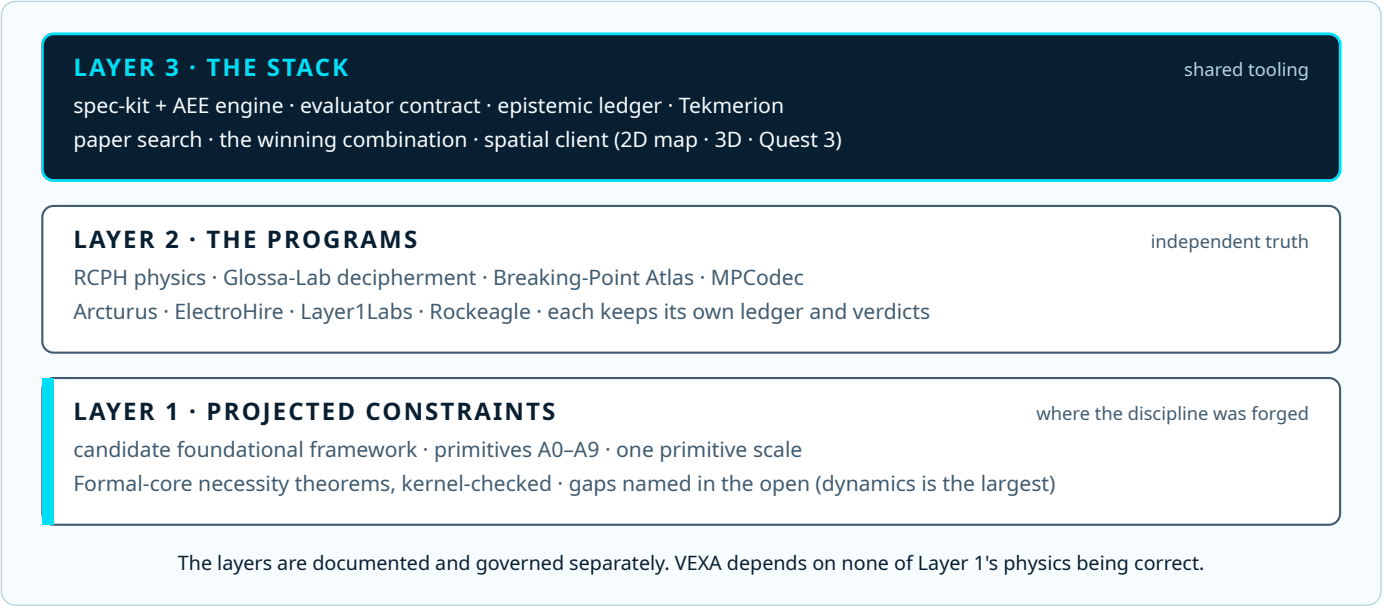

Figure 1 — The three layers of VEXA, documented and governed separately.

3. The ledger discipline

The ledger is an append-only file of claim entries. Each entry carries a statement, a source, a date, a status, a confidence, a falsifier, and a line stating what the claim does not establish. Confidence is carried verbatim: if the source says "low," the ledger says "low," in a dedicated field, forever. Any tool that maps narrative confidence to a number fails the stack's tests by design.

The ledger's schema is pinned in a versioned file beside it, and a validator enforces that schema at intake and in continuous integration. The validator recomputes each entry's proposition fingerprint (a hash over the normalized statement) rather than trusting the stored value, refuses schema violations loudly, and writes nothing on a refusal. Duplicate handling is declared per ledger: a claim ledger (one entry per proposition) refuses a duplicate fingerprint at intake; an assertion or evidence ledger (counting independent assertions as corroboration) accepts duplicates and counts them in a published census, grouped by fingerprint. Neither kind permits a duplicate to pass uncounted. On every validation run, coverage counts for the required fields are published, so a legacy gap is a standing number, never silence, and never backfilled with invented content.

Corrections are append-only reclassifications. A killed claim stays in the ledger, marked killed, with its history intact. The history is not overhead; in the spatial surface described below, it becomes something a researcher can walk through.

These rules are not aspirational. When the schema-pinning rule was proposed as a pattern amendment, the proposing program's first full run validated 871 entries, reported falsifier coverage of 831 of 840 with the 9 nulls carried as a counted, named gap, and the gate caught a real upstream collision and 21 stale source hashes on the same day. The founding program's ledger stands at 44 entries under its pinned schema, each bound to a pinned fingerprint manifest that a deliberate act alone can regenerate.

4. Truth flows one way: matrix, map, scene

From the ledger, a script generates the matrix: claims against evidence, studies, and relations. The matrix is executable. Its engine settles sign and status assignments mechanically, propagates constraints, and projects consequences, so a program's implications are computed, not narrated. From the matrix, a builder generates the map, and now the scene (Section 6). No downstream artifact is ever edited by hand. If a map disagrees with the ledger, the map is wrong, and the fix goes into its generator.

A verification gate stands between generation and publication. It recomputes each downstream layer from its source by an independent route and fails on any divergence. Three of its check classes carry named failure modes the fleet learned the hard way. Regeneration losslessness: rebuilding both layers from the ledger must reproduce the published artifacts byte for byte, on any platform, which requires hashing the canonical form of a file (line endings normalized, UTF-8) rather than its raw bytes; before that rule, the same ledger passed on Linux and failed on Windows over line endings alone. Relation-carrying citations: every citation is checked for carrying the specific relation it is cited for, not merely for existing and being relevant, because a gate once passed 202 checks and failed exactly one edge whose citation was about the right subject under the wrong relation. The check class has a numeric arm as well: a citation whose measured value violates the claim's own stated bound is refused. Gate write-freedom: a gate must never mutate the tree it checks, bytecode caches included, proven by hashing the published tree before and after a red run. A red gate leaves every published artifact byte-untouched. Nothing ships red.

At their recorded adoption points, the pattern's reference implementations reported gate results including 207 checks passing in plain mode and 208 with regeneration (the Atlas implementation, with falsifier coverage 543 of 543) and a layered map whose publication runs only through the gate (the Glossa implementation, five screens, generated from its ledger). Those figures are cited from their records with their dates in the underlying records; this paper claims them for those programs at those times, and for nothing else.

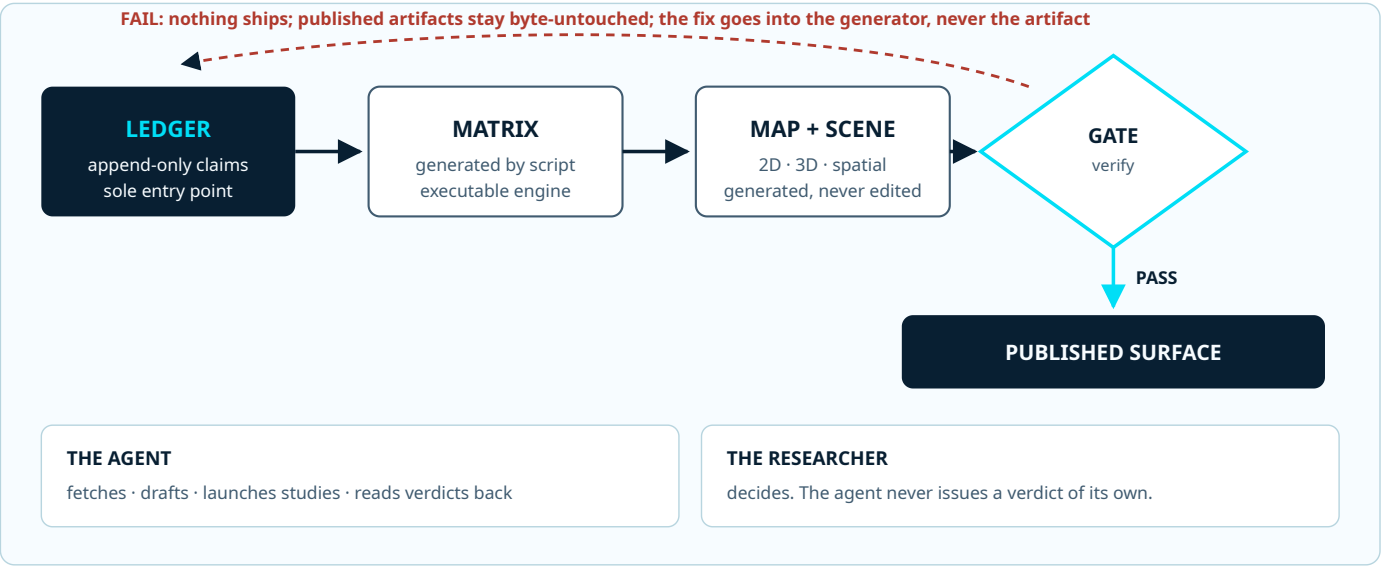

Figure 2 — Truth flows one way: ledger, matrix, map and scene, through the gate.

5. Studies that judge themselves

Claims enter the ledger from studies, and VEXA's study discipline is the ledger discipline applied earlier in time. A study is registered before it runs, with its falsifier and its verdict criteria stated in advance. Its runner is frozen and its hash recorded. Its results are lumped by the run: one run's outputs never pool silently into another's, and when two comparable runs diverge (different results, or results from one and none from the other) the junction between them gets a formal divergence analysis before either result may be used. A verdict exists only where the machinery issued one against a valid frozen run. INCONCLUSIVE is a result. UNMEASURED is a state that renders as itself in every surface, never smoothed into an estimate.

Verdict reporting carries the same discipline one level up (a standing rule, ratified 2026-11-04 after a fleet incident in which a job's wrapper reported success while its payload never ran). Every wrapped execution in the stack reports both numbers: the wrapper's completion state and the payload's own verdict. A non-zero or absent payload verdict reads FAILED or UNVERIFIED in those words; a zero-step run blocked by billing reads NO VERDICT, never green; a study whose frozen runner never ran is UNMEASURED. A report that carries a success the payload did not earn fails the stack's tests by design.

Tekmerion is the stack's study-execution surface: it registers claims with mandatory falsifiers, verifies citations, freezes registrations before evaluation, issues verdicts only from valid frozen runs, and keeps the whole record append-only and hash-verified. A companion literature surface (a multi-source scholarly search tool covering arXiv, Crossref, OpenAlex, Semantic Scholar, and others) fills the retrieval gap beside it: finding what exists, so claims can be checked against their nearest neighbors before they are registered.

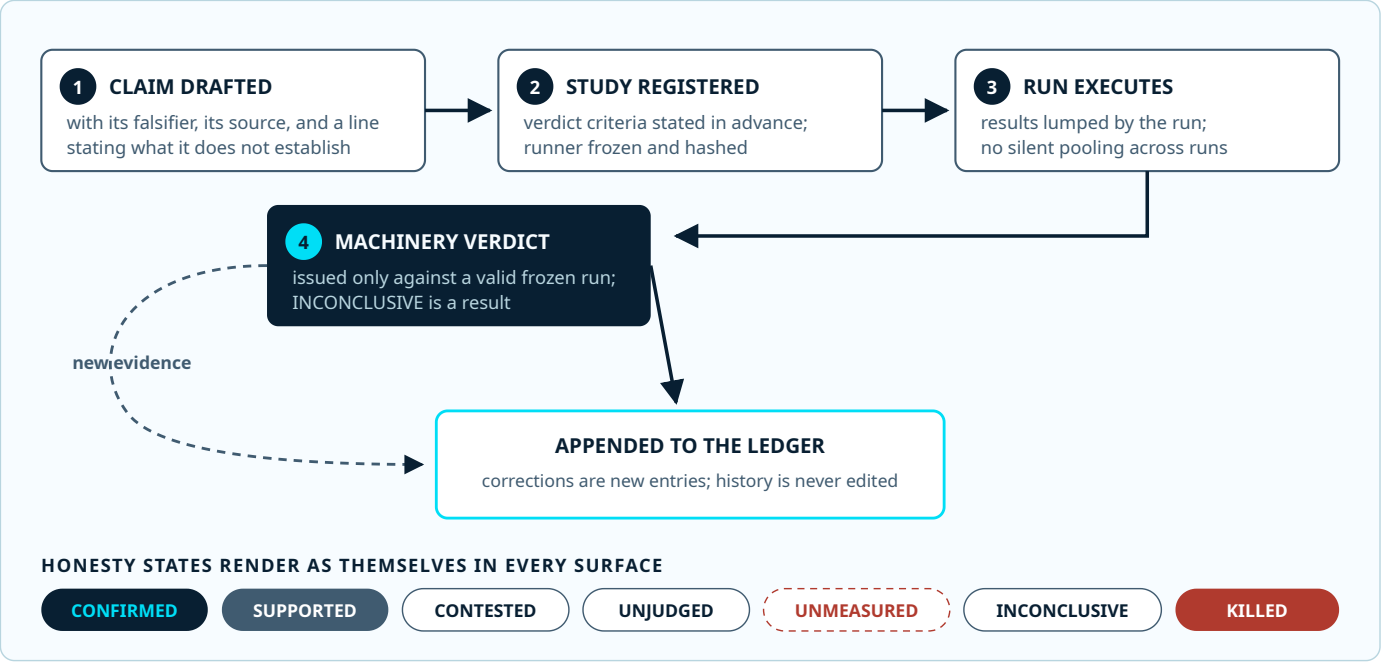

Figure 3 — The epistemics in play: from a drafted claim to a machinery verdict to an append-only record.

6. The spatial interface

A claim graph is not a spreadsheet, and a table is a poor way to stand in front of one. VEXA's primary surface is a graph: nodes for claims, evidence, studies, datasets, domains, and gaps, with typed, citation-carrying links between them. The node kinds and edge relations are a closed, shared

vocabulary; new ones enter only by ratified amendment. Containment is collapsible at every level, so a domain node is its whole subgraph, and expanding it in place is how a researcher moves from system to component to evidence. Selecting a link shows the exact ledger entries that carry it. Selecting a node re-centers the scene on its typed neighborhood. A builder-enforced visible-node budget keeps the scene readable: the knowledge-graph literature documents the global graph collapsing into an uninterpretable hairball past a few hundred nodes, and the field's remedies (focus graphs, typed and filtered links, collapsible containment, structured cards beside the graph) are VEXA's design, adopted deliberately. Because VEXA's links are already typed and gate-checked, it starts from the precondition that literature names as what makes a graph useful.

The same generated scene description renders three ways: a 2D map in a browser, a 3D view on a desktop, and an immersive scene through WebXR. On a Meta Quest 3, in the Quest Browser, the program graph can be anchored to a real table or wall in passthrough augmented reality and walked around, or taken into full virtual reality at scales no room could hold. Rendering uses Babylon.js over the WebXR standard, delivered from a URL with no install, and no headset is ever a blocker: every spatial function has a desktop equivalent and every datum a 2D one. History becomes literal in this surface: because the ledger is append-only, a time control renders the program as it stood on any date, and corrections can be watched landing.

One precedent deserves naming. Nanome demonstrated that scientists work well inside spatial scenes with an AI copilot that runs tools by plain-language request and returns results into the same scene, keeping a record of each run. VEXA adopts those two patterns. The difference is the object: a molecule is natively three-dimensional, and a claim graph is not. What makes VEXA's scene different from every spatial tool surveyed for this paper is that the scene is a gated, generated artifact of an epistemic ledger. If the scene and the ledger disagree, the scene is wrong, full stop. No surveyed tool makes that promise, because no surveyed tool has the ledger.

7. The agent in the scene

An AI agent is present in every surface, voice-first in the headset. It navigates and narrates paths through the graph, fetches literature and attaches it as evidence candidates, drafts claims with their falsifiers for intake, launches and harvests registered studies through the program's own machinery, and reads verdicts, scores, and honesty states back from the record. Its constraints are implemented as code and tests, not as instructions it might forget. Verdict language is templated from record fields, so a claim the record calls UNJUDGED is spoken as unjudged. Its summaries are labeled as summaries. Its writes go through the same validators and gates as a human's, are logged per run, and are attributable. When the machinery refuses, the agent surfaces the refusal as issued.

The rule can be stated in one line, and it governs the agent in every surface, chat included: the agent points; the researcher decides.

8. Immersion with clean hands

Immersion is a tool for understanding structure, and VEXA uses the honest half of what immersion practice knows: how to hold attention on one thread until its shape becomes clear. A guided session opens with an entry ritual of about a minute: the room fades, the scene builds around a question the researcher chose, and the agent states the session goal and the falsifier of the claim under study, aloud

and in text. Guided walks through the graph are pre-registered artifacts: path, stops, and what would change the walker's mind are stated before the walk runs, and deviations are logged as deviations. Inside a headset there are no notifications, no feeds, and no engagement metrics. Sessions have natural stopping points, and every guided session ends with an exit ritual: a read-back of what was decided, what was killed, and what is still open, written to the program's record. A researcher leaves with the ledger's version of the session, not the experience's version.

The other half of persuasion craft is refused, by name, in the ratified program record: no covert suggestion, no confidence priming toward a verdict, no selective omission in the agent's narration, nothing the researcher could not endorse if it were stated plainly. A finding that a shipped pattern does one of these is a severity-1 defect, checked by adversarial review at every release. Immersion is never a reason to believe a claim. Belief is the ledger's business, and the ledger does not care how a scene felt.

9. The extension-of-mind mandate

VEXA is designed, by its owner's ratified mandate (ratified 2026-11-04), as an extension of one researcher's mind, and the mandate is constitutional rather than decorative. It names four cognitive functions and assigns each to machinery: memory is the ledger and the record, which never silently edits; attention is the agent, which watches sources and surfaces exceptions instead of waiting to be asked; reasoning support is the drafting, computation, and adversarial checking the agent performs on labeled proposals; perception is the map and the scene, the surfaces a researcher actually stands in front of.

Delegation inside that design is maximal and bounded at the same time. Inside standing grants recorded in a registry, the agent completes work end to end and asks only about ends no grant covers. Control stays structural rather than attitudinal: every grant is revocable in one action, revocation is itself ledgered, and no grant may expand itself, so an ambiguous scope halts the agent rather than stretching the grant. The researcher stays informed without being interrupted: decisions arrive one at a time as a queue, exceptions and provenance digests are pushed, and any action the agent took can be retrieved with its evidence within two steps. Two things are never delegated. Truth stays mechanical: verdicts come from gates, validators, and frozen runs, never from the agent's judgment. Authority stays the owner's: a short reserved list (external sends, purchases, publishing, lifting gates) changes only by his own act.

10. A framework that improves itself, on the record

VEXA's programs share one pattern, and the pattern has a maintained way to change. When any adopting program finds an improvement or a failure mode in its own work, it writes a short pattern note and delivers it to every other program and to the VEXA home. The note is triaged on the record as an adoption into the shared components, a candidate amendment to the canonical pattern for the owner's ratification, or a local-only finding with its reason. Adopted changes propagate back with versions and tests, and landing is verified from each program's repository on a monthly cycle, not taken from anyone's report.

The mechanism has a track record. The relation-carrying citation rule came from a systems atlas's first gate run and was ratified as Amendment 1. Canonicalize-before-you-hash came from the tooling program's cross-platform failure and was ratified as Amendment 2. Schema pinning came from the

decipherment program's anchor-file discipline, generalized, and was ratified as Amendment 3. A ruling on duplicate propositions, scoped by ledger type, came from the ledger product's practice. Six further findings from a codec program (a gate must never mutate its tree, narrative confidence stays verbatim, and four more) were adopted by the decipherment program through the same loop. None of these improvements was designed centrally. All of them are now load-bearing in every adopting program. That is the sense in which VEXA is a bred framework, not a drawn one: the harmonization loop is its selection mechanism, and each program's working implementation is a candidate ancestor.

The loop's largest test so far is the generalization study. The skeleton abstracted from the physics instantiation was carried into a second domain, decipherment, as a registered study (Glossa-Lab, completed 2026-11-04). The domain stated its record system and projection before results (the first bar), taking no primitive absolute at all, a disposition sustained under adversarial attack. Two necessary conditions were derived from that projection (the second bar): a non-certification condition at full strength, and a recorded-profile condition at consistency strength, reported at exactly that strength. Two further candidates were not derived, with verified obstructions; one refusal produced a finding in its own right, that register certification is bookkeeping rather than a projection-class fact. The study's level is INSTANTIATES-path: the verdict over the full five-part bar belongs to an independent reader and has not been issued, and the study also fed two clarification candidates back into this registry, which is the loop working as designed.

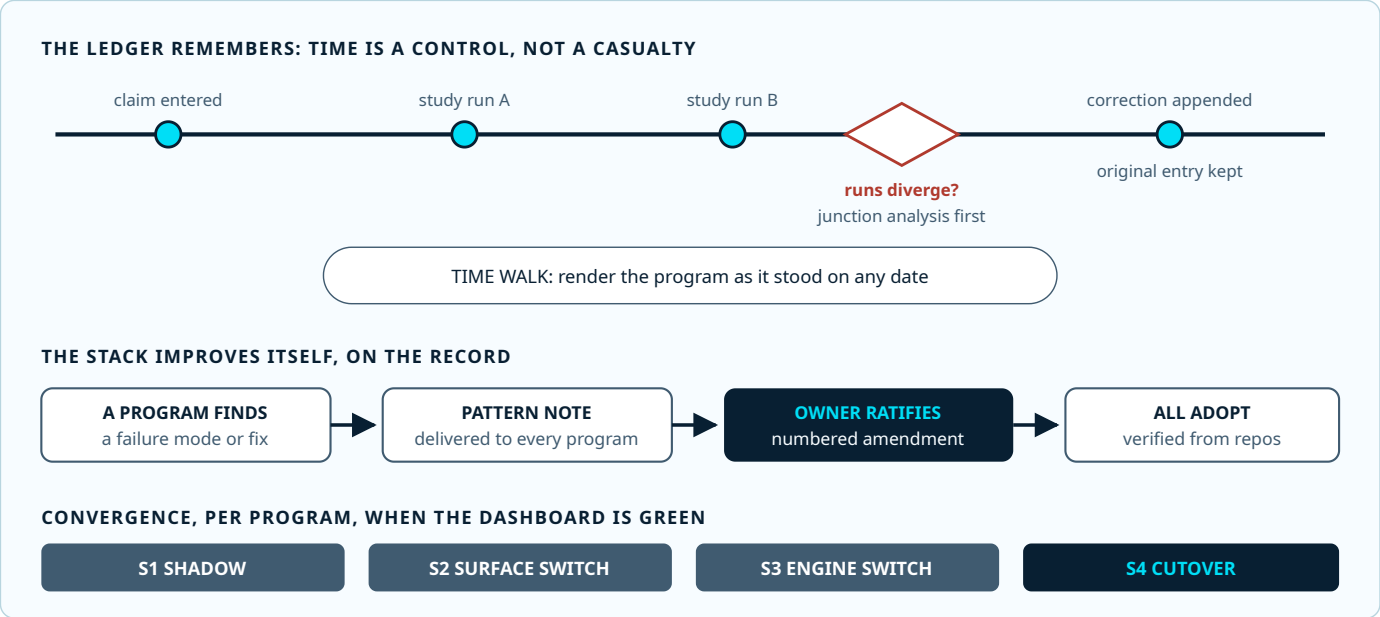

Figure 4 — Research continuity: an append-only history, a self-improving pattern, and gated convergence.

11. Convergence: replacing local stacks, when the time is right

Because each program built its implementation locally, under the pattern, the fleet today runs several local stacks that agree in discipline and differ in code. The canonical pattern named its own end state in 2026: once every program's plan has succeeded and the improvement feedback is flowing, the common framework is centralized as its own component, and the programs consume it instead of carrying local copies. VEXA is that centralization, and it proceeds in gated stages, per program: a shadow stage, in which the shared components run beside the local stack on the program's real ledger and must prove equivalence over consecutive gated runs; a surface switch; an engine switch, with historical verdicts reproduced on a frozen slice of the program's ledger; and a cutover, at which the local implementation

retires to read-only archive. Every stage is reversible until cutover, and the rollback is drilled in a scratch clone before cutover is allowed. Cutover is decided per program from a dashboard of six measured criteria (including that the program is between campaigns, that shadow equivalence has held, and that the rollback drill is current) and the owner's explicit go. Tooling converges; program truth never moves.

As of this paper's date, no program has entered the shadow stage. The founding programs still run their local implementations, in full health, under the harmonization loop. The paper states this plainly because the plan working as designed looks, from outside, exactly like nothing happening, and the distinction matters: convergence is gated on evidence, and the evidence has not been gathered yet.

12. What VEXA does not establish

This section is required by the stack's own constitution, and it is not boilerplate. VEXA does not make any program's claims true. A green gate proves that the matrix, map, and scene agree with the ledger, that citations carry their relations, and that regeneration is lossless. It proves consistency, not correctness. The underlying science stands or falls in its own studies. The Layer 1 framework remains a candidate: its formal results are kernel-checked as mathematics, its physical identifications are a research program's claims, and its largest gap, dynamics, is open. The efficiency claim one might expect for the development flow (that spec-kit with the epistemic engine outperforms spec-kit alone) is, on the benchmarking program's own record of 2026-10-29, not proven: the instrument is proven, component results stand, and the headline comparison earned a nil under its frozen bars. This paper does not claim it. The spatial and immersive benefits described in Sections 6 to 8 are design commitments with defined device test bars (sustained frame rate, task completion, spoken-verdict fidelity, session discipline), not measured results; until those test cases run on hardware, they are stated as commitments, and the record's traceability matrix says exactly which tests would settle each one. Finally, the generalization result of Section 10 establishes that the skeleton's first two bars can be met in a second domain; it does not establish instantiation, which requires all five bars and an independent reader, and it establishes nothing about the Indus script. Finally, the name: VEXA is used in the lockup "VEXA, a Projected Constraints Stack by Axiovex Systems," and commercial use beyond research and program work awaits a trademark clearance that this paper does not substitute for.

13. Availability

VEXA is in active development at Axiovex Systems. Its published components include the Applied Epistemic Engineering engine and the epistemic ledger (both on PyPI), the spec-kit extensions for the engine and the evaluator contract, and the Tekmerion study engine. The spatial client and the centralized shared components are in development under the ratified program record summarized here. Programs adopting the pattern today do so through the harmonization loop described in Section 9.

References

1. The winning combination, canonical pattern note, RCPH / projected-constraints program, 2026-11-01, with Amendments 1 to 3 and the layer-scoped duplicates ruling of the same date.
2. VEXA program record v1: constitution, requirements, architecture, improvement loop, convergence plan, test cases. Axiovex Systems, ratified 2026-11-04.

3. Projected Constraints Program charter, ratified 2026-10-30; Minimal primitive formal note (Lean, bare core).
4. applied-epistemic-engineering 1.6.0 (PyPI) and repository; spec-kit-aee and spec-kit-evaluator extension repositories; spec-kit-aee-benchmark repository and program record, including the owner verdict of 2026-10-29.
5. epistemic-ledger 0.15.0 (PyPI) and repository; tekmerion 0.2.0 (PyPI) and repository.
6. Glossa-Lab study records (winning combination adoption; node system tightening; full-picture map and gap register), BitConcepts/glossa-lab.
7. Breaking-Point Atlas records: README, ledger schema, and pattern note on the gate-and-settle discipline (2026-11-01), with adoption records of 2026-11-02.
8. WebXR Device API and Meta Quest Browser capability documentation; Babylon.js WebXR documentation.
9. Nanome platform materials on spatial computing for research and its AI copilot; immersive analytics literature, including ASCRIBE-VR (Quest 3) and interactive molecular dynamics in VR.
10. Knowledge-graph and personal-knowledge-management graph-view critiques and remedies, as surveyed 2026-11-04.

© 2026 Axiovex Systems, LLC. All rights reserved. Whitepaper v1.1 · 2026-11-05
AXIOVEX SYSTEMS, LLC · 6633 18 Mile Road, Sterling Heights, MI 48314 · axiovexsystems.com · start@axiovexsystems.com
VEXA whitepaper v1.1, 2026-11-05 (revised 2026-11-16). The name VEXA is used in the lockup "VEXA, a Projected Constraints Stack by Axiovex Systems"; commercial use beyond research and program work awaits trademark clearance.