Complaint-grounded provenance for AI-mediated software work

Alan N. Pham[1]

[1]AO Labs, Worcester, MA, USA

Human–AI software work increasingly fails not because no artifact is produced, but because the user must reconstruct why an artifact changed, whether the public surface actually deployed, and whether the change relieved the complaint that started the work. Existing provenance systems record derivation, logs record events, observability systems expose metrics and traces, and AI-interaction guidelines emphasize feedback and intelligibility; none of these, by themselves, preserve the user's complaint as a first-class operational datum. Here we introduce Progress, a production AO Labs monitor that links public source movement to complaint-grounded issue records. Progress scans public pages, selected APIs, paper PDFs, a CV PDF, and a private planning-text export; stores compact source fingerprints internally; renders plain field-level movement publicly; exposes a manual refresh control; renders capture health, recurring pattern families, and a top-level work log independent from scan diffs; and attaches structured notes containing the complaint, issue being solved, Codex-side change, observed source change, Spec reuse note, provenance, commit, and snapshot binding. A live read on May 18, 2026 at 5:17 PM EDT reported 33 configured sources, 33 online sources, five changed sources, and zero offline sources before this paper route itself was added. A May 19 correction broadened the AO Labs source graph to include Spec, League, and A3 surfaces and made same-day detector-only rows a logging failure to backfill. A later May 19 correction added a separate Work logged section so Curtis and League work records remain visible even when the latest scan body diff is elsewhere. A May 22 update added working-fallback and preferred-domain tracking for a new microphone alarm app; the live scan at 7:39 PM EDT reported 51 configured sources and 48 healthy sources, with the new `dbalarm_custom_domain` row correctly showing unresolved DNS while `aolabs.io/dbalarm/` was live. A May 24 source update added a password-gated Research ledger, separated its icon from Progress, and then moved the hub from fallback to the canonical `research.aolabs.io` domain after DNS resolved; the live scan at 5:48 PM EDT reported 54

configured sources and 51 healthy sources, with `research_home`, `research_fallback`, and `research_summary` all returning 200. On May 25, Alan rejected the renamed Todo list as maintenance pressure and asked to stop and remove it; Codex deleted the Railway project, removed the hub tile and fallback routes, and removed the three Research/Todo tracking sources. Later the same day, Alan clarified that he did need a minimal two-column Todo table with one fixed-height status window per row and an empty new-item row integrated into the table itself. Codex restored Todo on Railway, restored the AO Labs hub fallback route, and tracked the working fallback, Railway service, public-safe summary API, and unresolved custom domain as separate Progress rows. The live scan at May 25, 2026, 12:14 PM EDT reported 57 configured sources and 54 healthy sources; `todo_fallback`, `todo_service`, and `todo_summary` were healthy, while `todo_custom_domain` remained unresolved. A follow-on May 25 correction moved the persistent blank new-item row to the bottom of the table, compacted the header and rows, and verified that a saved row appears above a fresh blank row after item creation. A later May 25 visual correction treated the working UI as incomplete because the buttons, typography, layout, and color still looked low quality; Codex rebuilt Todo as a quieter database-like table inspired by notes-app surfaces, kept the row-menu deletion guard, and verified desktop and mobile rendering on `todo.aolabs.io`. The final repeated polish complaint exposed a remaining table-affordance gap, so Codex added a quiet drag handle with persisted row order, preserved inline item-name editing, collapsed empty saved status cells, and added an intentional loading row for first paint. A June 9 correction adds a top-level Capture audit that reports changed sources with no attached issue note, recent work events missing required fields, and the boundary that active Codex threads are not captured unless Codex harvests them into an event. A June 11 correction adds source-bounded pattern recognition over structured work events and current scan gaps, so repeated burdens such as capture gaps, public closure, interface shape, source evidence, Spec/paper sync, and finance queue language become visible families rather than private inference work. The contribution is a source-bounded architecture for turning user burden into inspectable provenance: complaint → issue → implementation → observed public change → current state → recurring pattern. The current evidence is a single production system record, not a population study, but it establishes a falsifiable interface primitive for AI-mediated work: a change is not complete until the reason it was started, the public movement it produced, and the repeated failure family it belongs to are visible together.

May 25 domain update: after Alan completed the Porkbun and Railway setup, `todo.aolabs.io` resolved to Railway, served Todo health and summary APIs, and became the hub tile target. Progress scan `d83b6f7220303235` showed all four Todo sources healthy while the overall source count remained 57 and the healthy count remained 54 because unrelated sources were still unavailable.

May 25 interaction update: Alan then rejected the always-visible blank row and the bare `x` delete affordance. Codex replaced the blank row with a bottom "New row" button, moved deletion behind a three-dot row menu, added an inline "Are you sure you want to delete?" confirmation, and verified on the live custom domain that only a temporary test row was deleted while Alan's saved `fluxcell` row remained.

May 25 design update: Alan then said the Todo surface still looked amateur, specifically naming the buttons, text, layout, and color, and asked for a more elegant Notion- or Obsidian-like treatment. Codex replaced the heavy form styling with a quiet database table surface, subtler borders and shadows, smaller header typography, left-aligned add-row behavior, restrained action controls, and a mobile menu placement fix. When Alan said the result still did not look polished, Codex treated polish as an identity-and-composition failure rather than another color tweak: the Todo favicon and AO Labs hub tile received a sharper table mark, the header scale was reduced, the bottom add control became a quiet row spanning both table columns, the status panes became shorter embedded windows, and the mobile item column was widened so item names stayed readable. When Alan repeated that the design still did not look polished, Codex changed the layout again so the table fills the working viewport, continues its row and column grid through empty space, and removes fake status placeholder text from saved rows. The final repeated correction added the missing database affordances: a subtle grip handle for click-drag row reorder, a persisted order endpoint, inline item-name editing, compact empty status cells, and a loading row so first paint is not an empty shell. The live site served the new stylesheet, JavaScript, HTML, and reorder API; temporary production rows verified drag persistence and were removed; and desktop plus mobile screenshots verified the rendered workspace.

May 25 archive update: Alan then decided the Todo surface did not match his real memory pattern: Messenger and the PhD organization document keep enough ambiguity to make him revisit what matters, while a clean Todo table would create maintenance pressure and reduce

that useful friction. Codex archived Todo rather than deleting it: the AO Labs hub tile was removed, `/todo/` became an archived-state page, the Railway deployment was stopped while the project and volume were preserved, the repository README records the archive state, and the active Todo rows were removed from Progress tracking while the work log preserves the reason.

June 9 capture update: Alan said Progress was still not capturing enough of the repeated frustration across AO Labs. Codex changed the summary API and public page so capture coverage itself is visible: latest work appears first, changed source rows without attached issue notes are counted, incomplete work events are listed with missing fields, and the page states that active Codex threads require explicit harvesting into Progress events.

June 11 pattern update: Alan then asked for Progress to capture every nuance and recognize patterns. Codex added a source-bounded `patterns` object to the summary API and a compact Patterns section to the public page. The recognizer groups recent structured events and current scan gaps into recurring families, repeated source clusters, strongest examples, and concrete next logging actions without claiming to read private active chats automatically.

1 Introduction

AI assistants have made software work faster, but they have also exposed a new reliability gap. A user can ask for a public page to be made clearer, a paper to be updated, a deployment to be verified, or a stale source to be removed; the assistant can make several commits; the live site can change; and the user can still be left asking what actually happened. The residue is not only technical uncertainty. It is cognitive work: reconstructing the original complaint, checking whether the assistant understood it, separating implementation changes from source changes, and verifying whether the public artifact now expresses the intended state.

This problem sits between several mature literatures. Cognitive artifacts and distributed-cognition accounts show how external records can stabilize action and memory across people and tools (*1–3*). Situated-action work shows that plans and instructions acquire meaning inside lived activity rather than through abstract specification alone (*4*). Boundary-object theory explains how records can coordinate work across communities without requiring every participant to share the same mental model (*5*). Provenance research and standards describe how data products can be connected to the entities, activities, and agents that produced them (*6–9*). Software observability, site-reliability practice, and distributed tracing make production systems legible through events, metrics, logs, and request paths (*10*, *11*). Human–AI interaction guidelines emphasize making system state, uncertainty, correction, and feedback pathways visible (*12*, *13*). Dataset and model documentation work shows that technical artifacts need explicit records of composition, limits, intended use, and accountability (*14–16*).

What remains under-specified is the complaint itself. In AI-mediated work, the complaint is often the highest-value datum: it states the burden the user was carrying. Yet ordinary logs do not know it; hashes do not express it; commit messages compress it; dashboards turn it into a metric; and model traces usually describe inference, not why the work was started. This is especially costly when interaction load is already part of the failure. Cognitive-load theory and executive-function accounts make clear that unnecessary reconstruction can be a real performance burden, not a mere inconvenience (*17*, *18*). When a user is autistic, has ADHD, or simply operates under high context and high standards, the difference between a raw technical record and a cognitively usable record becomes a product requirement (*19*). The problem is sharpened by large language models because fluent explanations can sound complete while remaining weakly grounded in the actual artifact or missing the communicative intent behind the user's request (*20*). Similar lessons appear in data-centric

AI work: the unglamorous work of preserving data context, lineage, and quality often determines whether downstream models and interfaces are trustworthy (*21*).

Progress is a deliberately small answer to this gap. It is not an analytics warehouse, an autonomous evaluator, or a generic observability platform. It is a complaint-grounded provenance monitor for AO Labs. It periodically scans the public and selected private-facing sources that matter to the AO Labs suite, detects source movement, and renders a calm public ledger. Its central design choice is to separate four things that often collapse together: the complaint that started the work, the issue being solved, the Codex-side implementation change, and the observed source movement. In the latest visible interface, complaint and issue are merged into one compact "Issue" block, but the API preserves the fields separately so Spec and future records can reuse them without reconstructing intent from prose.

The novelty is not that Progress checks URLs. Uptime monitors, change detectors, logs, and provenance models already do pieces of that. The novelty is the typed link from subjective friction to public source movement inside an operational monitor. Progress treats a user's complaint as a recordable cause in the practical sense relevant to AI-mediated work: the human reason the change was started. The system can then ask whether the source that moved is the source that should have moved, whether the change note is attached to the right snapshot, whether retired sources are still polluting current status, and whether the UI is showing evidence or merely numbers.

2 Results

2.1 Progress converts complaint into a provenance object

The core data model is a directed record of work rather than a score. A monitored source may change for many reasons: a deploy, an upstream site edit, a PDF rebuild, a private export failure, or unrelated third-party page churn. Progress detects that movement through response metadata and internal fingerprints. The human reason is attached separately through an event. Figure 1 shows the resulting chain.

Complaint: user burden that started the work.

Issue: problem being solved in operational terms.

Codex change: implementation, source-list, paper, or deployment action.

Source change: observed public/API/PDF movement in the scan.

Current state: latest source fact served by Progress.

Figure 1. Complaint-grounded provenance chain. Progress preserves the user's complaint and the issue being solved separately from source movement. The public page merges complaint and issue into one "Issue" block for readability, then shows the Codex-side change, observed source change, and current source state as separate evidence.

This structure addresses a failure common to AI-assisted implementation. A source diff can prove that something changed, but it cannot prove that the assistant solved the right problem. Conversely, a polished explanation can claim the right intent while the live surface remains stale. Progress requires both sides to be inspectable. The note says why the work was started; the scanner says what the world now serves.

2.2 A live scan establishes the first bounded evidence state

The source-of-truth read used for this manuscript was the production summary endpoint on `progress.aolabs.io`. The latest snapshot before the paper route was added had identifier `76fde4e141e9193b` and was created on May 18, 2026 at 5:17 PM EDT. It reported 33 configured sources, 33 healthy responses, five changed sources, and no offline sources. The changed sources were `progress_home`, `progress_summary`, `imagineer_ops`, `curtis_ops`, and `youtube_nalalan`. These values are deliberately dated. They are evidence of one production state, not a stable performance claim.

Field	Production value used here
Snapshot	`76fde4e141e9193b`
Scan time	May 18, 2026, 5:17 PM EDT
Configured sources	33 before `progress_paper` was added
Healthy sources	33 of 33 online
Offline sources	None in this read
Changed sources	Progress page; Progress summary API; Imagineer state API; Curtis ops API; YouTube `@nalalan` page
Scan reason	Manual

Table 1. Live Progress state used as manuscript evidence. The paper route introduced by this work is expected to increase the configured source count after deployment and the next successful scan.

The source list includes public AO Labs pages, project PDFs, selected operational APIs, the public CV PDF, working fallback routes, preferred custom-domain boundary checks, and a private planning-text export stored server side. It also reflects recent correction: a retired `relaylive.aolabs.io` source was removed because it produced a 404 that no longer represented current Relay state. This matters because monitoring can create false burden when it preserves obsolete expectations. Removing retired sources is therefore part of provenance hygiene, not only source-list maintenance.

The next correction exposed the opposite failure. Progress had source status for some AO Labs work while leaving other active surfaces outside the source graph. League animation and recording work, Spec revision state, and A3 public state could therefore disappear from the user's progress record even when those apps had active public routes. The May 19 source update adds Spec home, Spec summary, Spec paper, League home, League recordings, League paper, and A3 home to the monitored set. It also adds a public refresh button so Alan can force a current read without waiting for the worker cadence.

A May 22 source update tests the same rule under a route-boundary condition. Alan asked for

`dbalarm.aolabs.io` to appear on the AO Labs hub and to provide a microphone-triggered alarm for high sound levels. The preferred subdomain did not yet resolve in DNS, so the implementation shipped a working fallback route at `https://aolabs.io/dbalarm/`, configured a standalone GitHub Pages repository with `CNAME=dbalarm.aolabs.io`, and added two Progress sources: `dbalarm_home` for the working fallback and `dbalarm_custom_domain` for the preferred but blocked domain. The production scan at May 22, 2026, 7:39 PM EDT reported 51 configured sources and 48 healthy sources. The unhealthy rows were `dbalarm_custom_domain` because DNS did not resolve, `sleep_custom_domain` because the certificate did not match the hostname, and `sarrus_paper` because the direct PDF route returned 404. The important result is not that every source is healthy; it is that the monitor distinguishes a live fallback from a blocked preferred route instead of making the user remember that split.

A May 24 source update briefly repeated this boundary pattern for a password-gated Research ledger and then a renamed Todo list, but the May 25 corrections show that source tracking must both remove abandoned surfaces and restore a clarified one without confusing it with old Research state. Alan first said the Todo list would create pressure to maintain a system he would not use and asked to stop and remove it. The cleanup removed the AO Labs hub tile, deleted the `/todo/` and `/research/` fallback routes, deleted the Railway project that served the app, and removed `research_home`, `research_fallback`, and `research_summary` from Progress. The production scan at May 25, 2026, 1:45 AM EDT reported 52 configured sources and 49 healthy sources with no Research or Todo source rows. Alan then clarified that the useful artifact was not analysis or a work-pressure ledger, but a two-column private Todo table: one item name column, one fixed-height status cell whose scroll position keeps the newest text visible like a message thread, and an always-empty new-item row inside the table. Codex restored that narrowed app on a new Railway project, restored `https://aolabs.io/todo/` as a working fallback route, and added four Todo-specific Progress rows. The production scan at May 25, 2026, 12:14 PM EDT reported 57 configured sources and 54 healthy sources: `todo_fallback`, `todo_service`, and `todo_summary` returned 200, while `todo_custom_domain` remained unresolved because `todo.aolabs.io` was not yet attached. A later May 25 deploy moved the empty row to the bottom rather than the top, made the header and row heights smaller, and verified the live behavior by creating and deleting a temporary production row without leaving test data behind.

The custom-domain closure happened later on May 25. After Alan completed the Porkbun and Railway

setup, `todo.aolabs.io` resolved to Railway and returned the Todo health and public summary endpoints. Codex switched the AO Labs hub tile and `/todo/` fallback from the temporary Railway URL to the custom domain. Progress scan `d83b6f7220303235` then showed `todo_fallback`, `todo_service`, `todo_summary`, and `todo_custom_domain` all healthy.

The interaction itself changed again in the same work cycle. Alan said the persistent blank row should instead be a designed new-row button, and that the delete control should not be a visible `x`. Codex replaced the always-empty row with a bottom "New row" button that opens one draft row on demand, hid deletion under a three-dot row menu, and required a compact inline confirmation before deleting. The production verification on `todo.aolabs.io` created a temporary row, opened the menu, cancelled deletion once, confirmed deletion once, and verified that the real saved `fluxcell` row remained.

The visual-quality complaint was a separate source-of-truth event rather than a cosmetic preference. Alan said the table worked but still looked amateur and low quality, with weak buttons, text, layout, and coloring. The follow-on design change kept the same two-column product shape but removed the heavy beige form language, reduced the header scale, made cells read like editable database fields, turned the add control into a quiet left-aligned row action, softened the menu and destructive confirmation, and fixed the mobile menu so it no longer spilled off the viewport. A second polish correction followed when Alan said the result still did not look polished: the bottom add control stopped behaving like a boxed call-to-action and became a quiet table row spanning both columns, the Todo favicon and hub tile were redrawn as a sharper literal table mark, the header and status panes were tightened, the mobile item column was widened, the menu surface stayed opaque instead of letting text ghost through, and viewport-scaled type stayed out of the main UI. A third correction treated the remaining failure as a page-structure problem: the table now occupies the working viewport, faint row and column grid lines continue through the empty area, and saved rows with empty status cells no longer show fake placeholder text. The final correction addressed direct table manipulation itself: Alan wanted click-drag row rearrangement and explicit item-name editing, so Codex added a quiet grip handle, a persisted reorder endpoint, inline editable names, compact empty status cells that expand on focus, and a loading row for first paint. Live verification on `todo.aolabs.io` confirmed that the new stylesheet, JavaScript, HTML, and reorder API were served; two temporary production rows could be dragged into a new saved order and then removed; the three real rows remained; the menu still required confirmation; and desktop plus mobile renderings had no horizontal overflow. The next correction changed the operational state rather than the interface: Alan decided the clean table would not fit how

he actually remembers tasks, so Codex archived the app as inactive, removed its hub tile and active Progress sources, replaced the fallback redirect with an archive-state page, and stopped the Railway deployment while preserving the project, source, volume, and saved data for restore.

The May 26 Idle Shroom motion correction applies the same rule to alias routes. Alan repeatedly said the game looked cheap, unpolished, poorly made, and then identified the animations as bad. Codex changed the public tap loop and verified three playable routes: `https://aolabs.io/idleshroom/`, `https://aolabs.io/mushroom-boop/`, and `http://idleshroom.aolabs.io/`. Progress already tracked the main AO Labs route and standalone route, but the `/mushroom-boop/` alias was missing from the source graph. The source list therefore adds `idle_shroom_mushroom_boop` so the monitor follows every public route touched by the game-quality fix rather than leaving Alan to remember an untracked alias.

2.3 Public rows show exact movement without exposing raw fingerprints

Progress uses response fingerprints internally to detect body movement. The public interface does not show those fingerprints. This design follows a simple distinction: a hash is good evidence for detection, but bad evidence for comprehension. The public row therefore translates movement into plain fields when possible: response size, status transition, title transition, content-type transition, and compact JSON-field movement. For the Progress summary API, for example, compact fields include latest scan time, healthy count, source count, changed count, and snapshot count. For domain-specific APIs, Progress stores only a bounded public summary rather than mirroring full payloads.

When no smaller field-level diff is available, the row says so. This is an important negative result. HTML and PDF body movement can be real without being interpretable at the public-row level. Third-party pages can change for reasons unrelated to AO Labs. Private text exports can fail because of upstream availability. The system's credibility depends on not converting those limits into confident narrative.

2.4 Issue notes prevent detector-only explanations

The issue-note model is the mechanism that makes Progress more than a change detector. An event can name one or more `source_ids`, bind to a `snapshot_id`, and store `complaint`, `issue`, `codex_change`, `changed`, `spec_note`, `provenance`, and `commit`. The summary builder attaches notes to sources in the current snapshot. Notes with a snapshot identifier apply only to that snapshot. Notes without one apply only near the scan time, preventing stale intent from being silently reused for an unrelated future change.

Field	Function
`source_ids`	The monitored sources to which the note applies.
`snapshot_id`	Exact scan binding when available; prevents later source movement from inheriting old intent.
`complaint`	The user's complaint or nearest verified prompt that started the work.
`issue`	The human problem being solved; shown with complaint as one visible "Issue" block.
`codex_change`	What Codex changed in code, paper, source list, deployment, or public copy.
`changed`	What the monitored source actually changed to, independent of intent.
`spec_note`	Why the record is reusable by Spec or another AO Labs archive.
`provenance`	Basis for the note, such as thread, rollout, live scan, commit, or source route.
`commit`	Optional implementation commit identifier.

Table 2. Issue-note fields. The API preserves data separation; the UI reduces visible repetition by merging complaint and issue into one block.

The latest Progress corrections illustrate the value of this model. Alan complained that the page showed pointless numbers, raw technical fragments, and a "WHY" section that explained why Progress detected a change rather than why he started the work. The resulting implementation changed both the data model and the interface: raw hashes were removed from public summary output, field diffs were rendered in readable language, complaint and issue notes were attached to rows, Codex-side changes were separated from source movement, and Relay Live was retired from current tracking. A subsequent complaint exposed a second failure: Curtis and League work existed in the event ledger but was not visible enough because the main surface still privileged the latest source diff. Progress therefore added a Work logged section above the changed-source table. A later complaint exposed a third failure: even a work log is insufficient if missing fields and unharvested chat evidence are invisible. Progress now adds a Capture audit above the work log, counts changed rows without issue notes, lists incomplete work records, and states the thread-harvest boundary. The newest correction exposes a fourth failure: a complete note can still leave Alan to infer the repeated pattern across events. Progress therefore computes a source-bounded pattern object from recent structured work records and current scan gaps. It groups recurring burdens, repeated sources, representative examples, and the next logging action without claiming private chat omniscience. Progress records the complaint as the

reason the page changed, the scanner records the public movement produced by the change, the event ledger keeps recent work visible when scan diffs move elsewhere, and the pattern layer shows when several records are the same failure family.

2.5 The interface is intentionally boring

The visual design is part of the method. The page avoids celebratory dashboards, oversized cards, abstract scores, and raw technical evidence. It uses a state-first layout: current scan, capture status, recurring patterns, work logged, needs attention, changed this scan, tracked groups, tracked sources, and recent log. The Capture section summarizes whether source movement and work records have enough attached intent to be useful. The Patterns section summarizes the strongest recurring family, open unnoted changed rows, recent event-window size, and a short list of family/action rows. The work log shows recent issue events as work records rather than hiding them inside scan history. The changed row has four human-facing blocks: “Issue”, “Codex changed”, “Source changed”, and “Current”. This is a small but important interface claim. A monitor for cognitive relief should not require the user to decode a monitor.

This design aligns with human–AI interaction guidance that systems should show status, make uncertainty visible, support correction, and expose what the system can and cannot do (*12*, *13*). It also aligns with research on AI documentation and accountability: visible records should describe artifact boundaries and intended interpretation, not only output values (*14–16*). Progress applies those principles at the level of live software work rather than static model release.

The refresh control is intentionally small. It is not a dashboard flourish; it is an accountability affordance. If Alan suspects that the record is stale, he can request a new scan directly. The UI reports scanning, cooldown, failure, and last-updated states in plain text. The same correction also changes the logging standard: a same-day Codex-authored change that appears as “No issue note found”, as a detector-only body change, or as an incomplete work event is treated as missing work to backfill, not as an acceptable monitor state.

2.6 The paper becomes part of the monitored system

This work adds the paper route itself to Progress. The app now serves `/paper`, `/paper.pdf`, `/paper/source.tex`, and `/paper/references.bib`. The source list also adds `progress_paper`, a PDF source at `https://progress.aolabs.io/paper.pdf`. After deployment and the next successful scan, Progress should monitor the public paper that describes Progress. This closes a recursive but useful loop: the provenance monitor now treats its own public research record as a source whose availability and movement can be inspected.

3 Discussion

Progress proposes a narrow primitive for AI-mediated work: complaint-grounded provenance. The primitive is needed because the work product is no longer just code. It is a shifting relation among a user's complaint, an assistant's interpretation, a set of source edits, a deployment, a public surface, and a later record. Existing tools capture pieces of this relation. Git captures file history. Logs capture execution. Observability captures system behavior. Provenance models capture derivation. Human–AI guidelines capture design ideals. Progress binds these to the complaint that made the work necessary.

The claim is intentionally bounded. Progress does not infer private intent from source bodies. It does not decide whether Alan is satisfied. It does not turn every public source movement into a known cause. It does not make AI-generated explanations reliable by default. Instead, it creates a structure in which missing intent remains visibly missing and attached intent has a provenance field. That restraint is what makes the system scientifically useful. The monitor can be wrong, stale, or incomplete in ways that can be inspected.

The deeper contribution is to treat user friction as data with operational status. This matters for AI systems because friction is often the only signal that the assistant handed work back to the human. In ordinary workflows, that signal disappears into chat history. In Progress, it can become a typed event that changes future presentation. A complaint about raw hashes becomes a rule against showing raw fingerprints. A complaint about Relay Live becomes removal of an obsolete source. A complaint about "WHY" becomes a distinction between detector reason and human issue. This is an experimental pattern: if future complaints decline for the same class of burden, the record has functional value; if they do not, the issue-note model is insufficient.

The limitations are substantial. The current evidence is a single production deployment maintained by

one user and one AI-assisted workflow. There is no controlled study of cognitive load, no population-level evidence, and no benchmark comparing Progress to standard monitoring tools. The source list is curated, not discovered. Third-party pages can introduce noise. Issue notes depend on Codex writing accurate records. The next empirical step is therefore not a larger claim; it is measurement: correction frequency before and after issue-note rendering, number of rows with attached human issue notes, time to identify why a change occurred, stale-note rate, and user-reported reconstruction burden.

Still, the architecture is compelling because it makes a hidden unit of AI work observable. The unit is not a prompt, a commit, or a page. It is the relief loop from complaint to public state. A system that can track that loop can support stronger papers, cleaner dashboards, better instruction archives, and lower-friction software collaboration. Progress is the first AO Labs implementation of that idea.

4 Materials and Methods

System

Progress is a FastAPI application deployed at `progress.aolabs.io`. The backend defines `TRACKED_SOURCES`, a fixed list of source records with identifiers, names, lanes, kinds, purposes, URLs, and optional public-facing links. The scanner uses an asynchronous HTTP client with redirects enabled and a configured timeout. For each source it records check time, HTTP status, success state, response size, compact content type, and an internal SHA-256-derived response fingerprint. HTML sources receive a compact title. Text sources are decoded and stored server side. JSON sources are reduced through source-specific public summary functions. The source list is manually curated and now includes Spec, League, A3, Progress, Curtis, Imagineer, Relay, selected papers, selected APIs, working fallback routes, preferred-domain boundary checks, and other AO Labs surfaces.

Snapshots and public summary

Each scan creates a snapshot containing source records, healthy-source count, source count, lane counts, and deltas relative to the previous snapshot. Source movement is detected by comparing internal response fingerprints. The public summary compacts snapshots, filters retired source identifiers from public history, omits raw fingerprints, attaches issue notes, exposes current source documents and recent ledger entries, computes a capture-status object, and computes a pattern-status object. The capture object counts changed source identifiers without attached issue notes, audits recent work events for required fields, and names the boundary that active Codex chats are not captured unless Codex writes an event or performs a thread harvest. The pattern object uses keyword-defined, source-bounded families over recent structured work events and current scan gaps; it reports strongest recurring families, repeated source clusters, representative examples, open unnoted source changes, and the next logging action. The page consumes the summary endpoint rather than independently refetching monitored sources.

Issue notes

Codex-authored events are written to `/api/progress/events`. The event schema includes lane, kind, title, body, URL, source identifiers, complaint, issue, changed, Codex change, Spec note, provenance, commit, and snapshot identifier. The summary builder treats `change_issue`, `codex_change`, and `source_issue` as attachable issue-note kinds. Snapshot-bound notes apply only to the matching snapshot; unbound notes apply only near the scan time.

Interface

The public interface is a static HTML/CSS/JavaScript surface served by the same backend. It renders the current scan, manual refresh control, capture audit, pattern recognition, work log, needs-attention list, changed-source list, tracked groups, tracked sources, and recent log. The capture renderer reads the summary capture object and shows latest work, unnoted changed rows, incomplete work records, and the explicit thread-harvest boundary. The pattern renderer reads the summary pattern object and shows state, strongest family, open gaps, event-window size, repeated source clusters, family counts, source ids, examples, and actions. The work-log renderer reads recent `change_issue`, `source_issue`, and `codex_change` events directly from the summary endpoint, so logged work remains visible even when it is not part of the latest changed-source set. The change-story renderer merges complaint and issue into a single visible "Issue" block, then separately renders Codex-side implementation change, observed source movement, and current source fact. The same change also updates the script cache key so deployed browsers load the revised renderer. Manual refresh calls `/api/progress/scan/run`, handles cooldown and failure messages, and reloads the summary after completion.

Paper route

The manuscript uses the AO Labs default LaTeX article scaffold. The app serves the paper page at `/paper`, the PDF at `/paper.pdf`, the TeX source at `/paper/source.tex`, and the bibliography at `/paper/references.bib`. The paper PDF is added to `TRACKED_SOURCES` as `progress_paper`. Evidence values in this manuscript were taken from the production summary endpoint and the local Progress source tree on May 18, 2026, with source-boundary updates checked against the production summary endpoint on May 22, 2026 at 7:39 PM EDT and May 24, 2026 at 5:48 PM EDT.

5 Acknowledgements

The Progress design was shaped by Alan's repeated complaints that the page showed numbers, raw technical fragments, and detector-centered explanations without stating what was tracked, why the work started, what Codex changed, and what the source actually did. Those complaints are treated here as design evidence.

6 Funding

No external funding supported this system record.

7 Author contributions

A.N.P. defined the AO Labs workflow, supplied the complaint-driven requirements, and evaluated the visible artifact. Codex implemented the Progress monitor changes, manuscript route, and paper draft under A.N.P.'s direction.

8 Competing interests

The author declares no competing interests.

9 Data availability

The public Progress summary is available from `progress.aolabs.io` at `/api/progress/summary`. The endpoint exposes compact source state and issue notes. Raw internal fingerprints, private source text, secrets, and write tokens are not public data.

10 Code availability

The implementation source is maintained in the AO Labs Progress application repository. Public paper source is served from `progress.aolabs.io` at `/paper/source.tex` after deployment.

11 Additional information

This paper is a living system record. Substantive changes to Progress source tracking, issue-note semantics, capture audit semantics, pattern recognition, public scan presentation, or paper routes should update the manuscript and rebuilt PDF in the same work cycle.

References

1. D. A. Norman, in *Designing Interaction: Psychology at the Human-Computer Interface*, ed. by J. M. Carroll (Cambridge University Press, Cambridge, 1991), pp. 17–38.

2. E. Hutchins, *Cognition in the Wild* (MIT Press, Cambridge, MA, 1995).

3. J. Hollan, E. Hutchins, D. Kirsh, Distributed cognition: Toward a new foundation for human-computer interaction research. *ACM Transactions on Computer-Human Interaction* **7**, 174–196 (2000).

4. L. A. Suchman, *Plans and Situated Actions: The Problem of Human-Machine Communication* (Cambridge University Press, Cambridge, 1987).

5. S. L. Star, J. R. Griesemer, Institutional ecology, translations and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907–39. *Social Studies of Science* **19**, 387–420 (1989).

6. P. Buneman, S. Khanna, W.-C. Tan, presented at the Database Theory – ICDT 2001, pp. 316–330.

7. L. Moreau *et al.*, The Open Provenance Model core specification (v1.1). *Future Generation Computer Systems* **27**, 743–756 (2011).

8. L. Moreau, P. Missier, *PROV-DM: The PROV Data Model*, World Wide Web Consortium (`https://www.w3.org/TR/prov-dm/`).

9. J. Cheney, L. Chiticariu, W.-C. Tan, Provenance in databases: Why, how, and where. *Foundations and Trends in Databases* **1**, 379–474 (2009).

10. B. H. Sigelman *et al.*, presented at the Technical Report, Google.

11. B. Beyer, C. Jones, J. Petoff, N. R. Murphy, Eds., *Site Reliability Engineering: How Google Runs Production Systems* (O'Reilly Media, 2016).

12. S. Amershi *et al.*, presented at the Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, pp. 1–13.

13. B. Shneiderman, *Human-Centered Artificial Intelligence: Reliable, Safe and Trustworthy* (Oxford University Press, 2020).

14. T. Gebru *et al.*, Datasheets for datasets. *Communications of the ACM* **64**, 86–92 (2021).

15. M. Mitchell *et al.*, presented at the Proceedings of the Conference on Fairness, Accountability, and Transparency, pp. 220–229.

16. I. D. Raji *et al.*, presented at the Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, pp. 33–44.

17. J. Sweller, Cognitive load during problem solving: Effects on learning. *Cognitive Science* **12**, 257–285 (1988).

18. R. A. Barkley, Behavioral inhibition, sustained attention, and executive functions: Constructing a unifying theory of ADHD. *Psychological Bulletin* **121**, 65–94 (1997).

19. D. E. M. Milton, On the ontological status of autism: The double empathy problem. *Disability & Society* **27**, 883–887 (2012).

20. E. M. Bender, T. Gebru, A. McMillan-Major, S. Shmitchell, presented at the Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, pp. 610–623.

21. N. Sambasivan *et al.*, presented at the Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, pp. 1–15.