GitHub for Prior Auth: The Most Defensible Health AI Org of the Coding Agent Era Might Not Automate Auth but Version, Test & Distribute the Payer Coverage Rules Everyone Rebuilds From Scratch

JUN 26, 2026

🎧 Part I Podcast free on Spotify.

Thoughts on Healthcare Markets & Technology

Part I: GitHub for Prior Auth: The Most Defensible Health AI Org of the Coding Agent Era Might Not Automate Auth but Version, Test & Distribute the Payer Coverage Rules Everyone Rebuilds From Scratch

Every provider org, RCM vendor, and clearinghouse rebuilds the same prior auth logic from the same payer PDFs. In isolation. Every time. That is the infrastructure problem hiding in plain sight...

Listen now

a day ago · Thoughts on Healthcare

🎧 Part II Podcast episode for paid subscribers only. Also available on Spotify.

Thoughts on Healthcare Markets & Technology

Part II: GitHub for Prior Auth: The Most Defensible Health AI Org of the Coding Agent Era Might Not Automate Auth but Version, Test & Distribute the

Payer Coverage Rules Everyone Rebuilds From Scratch

Every provider org, RCM vendor, and clearinghouse rebuilds the same prior auth logic from the same payer PDFs. In isolation. Every time. That is the infrastructure probl

Listen now

a day ago · Thoughts on Healthcare

To listen to paid episodes in Apple or Spotify, li
settings on those platforms (instructions inside th
Podcast).

Thoughts on Healt
Technology is a re
publication. To recei
support my work, co
free or paid s

Table of Contents

Abstract

- Thesis: the strongest low capital healthcare company buildable with coding agents is not an auth bot. It is a version controlled, testable, machine readable layer for payer coverage and prior auth rules. Working names: CoverageOS or PolicyHub.
- Core question it answers: for THIS patient, payer, plan, CPT/HCPCS, ICD-10, site of care, what must be documented before the order is placed?
- Why better than ghost network / directory products: sits in the revenue cycle, ROI is countable (denials, resubmissions, peer to peers, write offs, abandonment), recurring per order usage vs quarterly audits.
- Regulatory catalyst: CMS Interoperability and Prior Authorization final rule. Impacted payers (MA, Medicaid, CHIP, QHP on the FFEs) must stand up a Prior Authorization API by Jan 1 2027, and report PA metrics plus specific denial reasons starting 2026. HL7 Da Vinci splits the work into CRD, DTR, PAS.
- Architecture: GitHub mapped one to one. Repo equals policy. Commit equals dated policy change with semantic diff. Branch equals product/jurisdiction variation. PR equals proposed correction with evidence. Issue equals unresolved ambiguity. CI equals continuous testing of structured criteria.
- pWedge: advanced imaging PA in Florida (MRI, CT, PET, nuclear cardiology) across MA, FL Medicaid managed care, large commercial, and the delegated vendors (Carelon, EviCore, Cohere, RadMD). No PHI needed at MVP.
- Open core: schema, CLI/SDK, Medicare NCD/LCD repos, synthetic test suite, provenance spec are open. Commercial payer graph, verified releases, contract specific branches, observed denial outcomes, and integrations are paid.
- Pricing range: practice $3k to $12k, imaging/infusion org $15k to $50k, regional group $25k to $100k, RCM/auth vendor $50k to $250k, enterprise API/data $100k plus.
- Moat: longitudinal policy corpus, normalized clinical ontology, code to policy graph, observed denial behavior, contributor reputation. The model commoditizes. The graph does not.
- Hard boundary: ship a research and change management product, not a guarantee of payment. Do not automate the final adjudication on day one.

One. The question that quietly breaks every revenue cycle

There is a single question buried under every imaging order, every biologic start, every sleep study and home oxygen setup, and it sounds trivial until somebody tries to answer it at scale. For this exact patient, on this exact plan, for this exact procedure and diagnosis at this exact site of care, what specifically has to be in the chart before the service gets ordered. That is the whole ballgame. Get it right and the claim sails. Get it wrong and the practice eats a denial, a resubmission, a peer to peer, a delay in scheduling, sometimes a patient who just gives up and never comes back. The maddening part is that the answer exists. It is sitting in a payer medical policy, a prior auth code list, a delegated vendor criteria set, a provider manual, a bulletin published two Tuesdays ago. It is just scattered across thousands of PDFs and portals in a format that makes it functionally invisible at the moment a clinician is finishing a note.

Most of the companies trying to fix healthcare administrative pain aim at problems that are real but hard to monetize cleanly. A ghost network or directory accuracy product, for instance, finds genuine compliance failures, but a buyer struggles to say what fixing any one bad record is actually worth. Coverage intelligence does not have that problem. The value drops straight onto the revenue cycle, where everything is already measured in dollars. An orthopedic group, an infusion center, a radiology shop can do the arithmetic on a five percent improvement in clean authorization rate in about the time it takes to finish a coffee. Avoided denials, fewer resubmissions, fewer peer to peers, lower staff labor, faster scheduling, fewer write offs, less treatment abandonment. These are line items, not vibes. And unlike a directory audit that happens monthly or quarterly, the coverage question gets asked every single time somebody orders advanced imaging, a specialty drug, DME, genetic testing, rehab, a cardiac procedure, ortho surgery, radiation oncology, or home health. High frequency plus countable ROI is the combination that makes software sticky, and coverage policy has both.

Two. Why coverage policy behaves like source code that nobody put under version control

Here is the uncomfortable mental model. Payer coverage policy is software. It is conditional logic that takes inputs (codes, diagnoses, durations, prior treatments, site of care) and returns an output (required, not required, covered, denied, document these seven things first). The catch is that this software is distributed the way code would be distributed if every company in an industry emailed each other forty seven page PDF attachments and then each maintained its own undocumented fork in somebody’s head. A rule can change without anyone understanding precisely what

changed. Different payer products run different versions of the same base logic. Local operational interpretation drifts away from the source. There is no canonical diff. There are almost no automated tests. Ambiguities get resolved privately, over the phone, with a reference number scribbled on a sticky note, and then re solved an hour later by a different nurse at a different practice who has never met the first one. Every provider, every RCM vendor, every clearinghouse, every auth company reconstructs roughly the same logic from the same documents and keeps the result to themselves.

Software engineering solved this category of pain decades ago, and not by being smarter about reading. It solved it with infrastructure. Version control so changes are attributable and reversible. Diffs so a change is legible. Dependency management so you know which version of which thing you are actually running. Automated testing so a change cannot silently break behavior that used to work. Code review so a proposed change has to survive scrutiny before it ships. Issue tracking so an unresolved question has a home. Release channels so the bleeding edge and the stable build are different things. The bet worth making is that coverage policy has comparable complexity to a midsize software project, and it is being run with none of these mechanisms. A vector database does not fix this. Throwing the PDFs into embeddings helps somebody find a passage, which is genuinely useful, but it is the search box, not the operating system. Finding the paragraph is not the same as knowing the rule, knowing which version of the rule applies on a given date, and knowing that the rule still does what it did last month.

Three. Taking the GitHub analogy literally instead of using it as a logo

The temptation is to slap GitHub on the pitch deck as branding and move on. The more interesting move is to take the analogy at face value and let it dictate the actual architecture, because the mapping turns out to be almost suspiciously clean. A repository is a coverage policy, not as a stored PDF but as a structured object that carries the source documents, effective dates, retired versions, applicable products, jurisdictions, code mappings, medical necessity criteria, exclusions, documentation requirements, site of service rules, step therapy, the delegated UM vendor, machine readable logic, a human readable summary, test cases, known ambiguities, and provenance for every single asserted fact. A commit is a policy change. When a payer moves its lumbar MRI policy on the first of the month, the system records what actually changed, which is a different artifact entirely from the email everyone currently gets that says a payer updated an imaging policy and here is the attached forty seven pages, good luck. The product shows two diffs side by side. The source diff shows which passages, tables, and code lists changed in the original document. The

semantic diff shows what the rule actually did, that an exception for progressive motor weakness got added, that chiropractic got dropped as a separately enumerated modality, that the required conservative therapy window went from four weeks to six. The semantic diff is where the coding agent earns its place at the table, because file diff tools can tell you paragraph fourteen changed while a model can propose that the substantive change was a new exception for progressive neurologic deficit, and then a human reviewer approves or rejects that read.

A branch is a plan or jurisdiction specific variation, which matters more than it sounds because the question does this payer cover this is almost always underspecified. The honest answer depends on legal entity, product, funding arrangement, state, employer benefit design, member contract, the delegated vendor, site of care, network status, and the effective date. Branches make those forks explicit instead of letting one generic policy summary cosplay as universal truth. The Medicare side maps cleanly too, where national coverage determinations behave like upstream rules, local coverage determinations behave like jurisdictional branches, billing and coding articles behave like implementation files, and the Medicare Administrative Contractors behave like the maintainers of jurisdiction specific distributions. A pull request is a proposed correction with evidence attached, submitted by a verified auth nurse who noticed the structured rule says six weeks of PT is always required when the source policy actually allows a documented physician directed home exercise program. The PR does not silently become truth. It creates a proposed change, supporting evidence, a discussion, a reviewer requirement, an auditable decision, and a permanent history, which neatly solves the central failure of naive crowdsourcing where contributions get treated as gospel the moment they arrive. An issue is an unresolved ambiguity, the policy that requires recent imaging without ever defining recent, which different regions appear to interpret as twelve months or six months with varying confidence and no official clarification anywhere, and the honest resolution is to label it ambiguous rather than force a model to manufacture a number.

The most underrated piece is continuous integration. Every structured policy carries test cases. A case might describe an MRI of the lumbar spine for radiculopathy with eight weeks of symptoms, no progressive motor weakness, six weeks of completed conservative therapy, no red flags, and assert that prior auth is required and documentation looks complete. Another might describe two weeks of symptoms with progressive motor weakness and no completed conservative therapy, and assert that the exception applies and the required documentation is objective motor deficit plus neuro exam. When the policy updates, the tests run, and the system flags scenarios that used to qualify and no longer do, new documentation elements, contradictions

between the narrative criteria and the code list, rules that cannot even be evaluated from the stated inputs, and divergences between two payers implementing what looks like the same criterion. That is the moment the thing stops looking like content management and starts looking like infrastructure.

Four. The 2027 mandate that nobody is building the content layer for

Timing is not usually a moat, but a forced industry wide format migration is close to one, and there is a big one already on the calendar. Under the CMS interoperability and prior authorization rule, impacted payers, which covers Medicare Advantage, state Medicaid and CHIP fee for service and managed care, and qualified health plans on the federally facilitated exchanges, have to stand up a Prior Authorization API beginning primarily on the first of January 2027. The API is meant to let providers determine whether authorization is even required, surface the documentation requirements, and exchange the request and the decision. Ahead of that, starting in 2026, those same payers have to give specific reasons for prior auth denials and publicly report a set of authorization metrics like approval rates and turnaround times. The denial reason data is the sleeper asset here, because it is the raw material for eventually learning which policy language actually predicts a denial rather than which language merely sounds important.

This creates a textbook transitional window. Today the requirements live in scattered PDFs, portals, and vendor sites, which is the painful but tractable status quo. Through 2026, providers need readiness tooling to survive the gap. In 2027, the same product can start ingesting the payer APIs right alongside the documents, treating the API as one more source rather than a replacement for everything. Later, the real denial reasons flow back in and sharpen the rules. So the company can launch as a policy intelligence layer and grow into the orchestration layer underneath the exchanges, which matters because the standards themselves do not solve the hard part. HL7 Da Vinci already cut the workflow into three named functions, Coverage Requirements Discovery to figure out if auth is needed and what is required, Documentation Templates and Rules to gather the right documentation, and Prior Authorization Support to move the request and decision. Those define the pipes. They say nothing about whether the rule being shipped through the pipe was interpreted correctly, whether the underlying policy is current, or whether the implementation matches what the payer actually does when a real claim shows up. FHIR can transport a rule. It does not vouch for the rule. That gap is the entire business.

Five. Where a coding agent actually earns its money, and where it should be kept on a leash

The wrong instinct is to point a model at a policy and ship a prior auth chatbot, because that capability is going to be everywhere and worth roughly nothing on its own. The high value job for a coding agent is maintaining the policy supply chain, which is closer to running a continuous build system than answering questions. Medical policies sit in a sweet spot of structure, regular enough that they can be parsed into criteria trees and code mappings and effective dates, but variable enough that hand written parsers are expensive to build and miserable to maintain, which is exactly the regime where a code generating agent shines. It can crawl payer document libraries, find the policy indexes and document endpoints and revision histories, propose a payer specific ingestion adapter with methods to discover documents and detect versions and map product scope, parse the PDFs and pages, pull out CPT and HCPCS and ICD-10 codes, separate the mandatory criteria from the explanatory throat clearing, compare versions, and generate edge case tests where the patient meets the general rule, qualifies through an exception, is missing a required duration, trips a contradiction between two criteria, or falls into a gap the policy never addresses.

The discipline that keeps this from becoming malpractice is that the model interprets and generates code, but it is never the unverified source of truth, and the final rules engine stays deterministic. Every extracted assertion stays chained to an exact source passage, a document URL, a version, an effective date, an extraction confidence, and a human review status. The review protocol is the actual product quality, not the model. The agent extracts the rule, a second independent model checks it, deterministic validation confirms every cited code and date resolves, a human reviews anything materially ambiguous, and the application always shows the original source evidence so a skeptical auth manager can audit the chain themselves. When a source changes, the agent runs the whole software development lifecycle on it, detect the change, pull the new document, diff against the prior version, propose a semantic commit, identify the affected rules and tests, run the regression suite, route material changes to a reviewer, publish an approved release, alert the watchers, and flag the specific customers whose workflows just got disrupted. The biggest line item in this company is not compute or hosting. It is clinical policy quality assurance, the human and procedural apparatus that makes a coverage assertion trustworthy enough that a practice will reorganize its scheduling around it.

Six. The Florida imaging wedge, and why starting narrow beats starting grand

The failure mode for a project this conceptually large is trying to boil the entire ocean of every payer and every service on day one, which produces a thin, untrustworthy layer of everything and a moat of nothing. The disciplined start is prior authorization requirements for advanced diagnostic imaging in one state, covering MRI, CT, PET, and nuclear cardiology across the major Medicare Advantage plans, the Florida Medicaid managed care plans, the large commercial payers, and the delegated management environments that actually run the gate, which means Carelon, EviCore, Cohere, and the RadMD style setups. Advanced imaging is the right beachhead for a stack of boring practical reasons that all compound. The code universe is constrained, so the problem is bounded. Authorization is common, so the product gets used constantly. The documentation requirements repeat across cases, so structuring one policy pays off across many orders. Denials directly delay scheduling, which is the kind of pain a practice manager feels in their spine. Imaging centers have sharp financial incentives and both the ordering practice and the imaging facility benefit, so there are two buyers per transaction. And critically, the product can launch with no PHI at all, because a coverage requirement reference does not need a patient to exist, which keeps the security and compliance burden of the MVP close to nothing and lets a human validate extracted policies efficiently.

The second wedge worth eyeing once the muscle is built is specialty infusion drugs, where the value per avoided denial is dramatically higher because the drugs are expensive, but the policy complexity and the drug specific change velocity are also nastier, so it is a fast follow rather than a starting line. The MVP itself does not need EHR integration to be useful. A searchable coverage database where a user enters payer, plan type, procedure code, diagnosis, and state and gets back the authorization requirement, the managing vendor, the clinical criteria, the required documentation, the applicable policy, the effective date, the recent changes, and the source citations is already valuable. A weekly policy change alert that tells an organization that eleven policies affecting them moved this week, three of which added prior auth, two of which changed site of care, four of which changed documentation, and two of which revised covered indications, is arguably the easiest thing to sell first because it requires zero workflow integration and slots straight into an inbox. A documentation checklist generator that turns a planned service into a structured list of what the note needs, duration of symptoms, objective findings, prior treatment and dates, contraindications, relevant labs, prior imaging, the required scoring instrument, rounds out a trio of products that can ship without ever touching a patient record.

Seven. What gets open sourced, what stays behind the paywall

Open source usually looks like a way to weaken a data business, and the instinct to keep everything locked up is strong, but here the open core model is the correct strategy precisely because it strengthens the moat if the line is drawn in the right place. The line is not give away the whole payer database and pray someone buys support. The line is open the common language, the tooling, and the public policy foundation, and charge for verified content, workflow integration, and proprietary operational intelligence. The schema should be public, a documented data model for what a coverage policy is, its scope and codes and criteria and provenance, because a shared schema invites adoption, lowers vendors fear of lock in, positions the company as the neutral representation layer, and lets third parties build complementary tools, and the schema was never the moat anyway, adoption of the schema is what makes the moat deeper. The command line tools and SDK should be open too, the validators and diff viewers and local test runners and FHIR converters and synthetic case generators and reference connectors for the CMS data, because every developer who pulls a policy, validates a file, runs a test suite, or compiles to FHIR using these tools is one more person treating this format as the default unit of exchange. The Medicare repositories are the strongest candidates for a genuinely open community layer, since national and local coverage determinations and the associated articles are public material that CMS already exposes through the Medicare Coverage Database and a coverage API, and the value the project adds is turning that raw material into versioned repos with structured criteria, code mappings, human readable diffs, test cases, and issue tracking, which generates search distribution, developer adoption, academic use, credibility, and a training ground for contributors without trying to charge for government data that sophisticated buyers can already pull themselves. The synthetic test cases and the provenance standard round out the open layer, the former because real patient cases bring privacy and contracting baggage while synthetic ones can become an interoperability conformance suite for the whole industry, and the latter because industry trust depends on being able to inspect exactly how a conclusion was reached.

What stays paid is the part that costs continuous money to maintain, requires privileged data, or carries enterprise accountability. The commercial payer policy graph is the core revenue, not because any individual PDF is hard to find but because the continuously maintained graph connecting payer to legal entity to product to network to jurisdiction to service to code to diagnosis to requirement to documentation to effective date to source is expensive to keep alive and current. Verified production releases are paid, where the community edition carries machine

extracted and community reviewed rules with transparent confidence levels while the enterprise edition carries clinically reviewed releases with defined validation, service level agreements, update guarantees, role based approvals, audit logs, and liability boundaries, which is the same distinction as a free open source project versus a supported enterprise distribution, and the paying customer is buying assurance, not access. Plan and contract specific applicability is paid and deeply embedded, because the public policy may say one thing while a provider's specific contract or self funded employer plan or delegation arrangement changes the operational reality, so enterprise customers upload their contracts and plan documents and internal payer matrices and the system builds a private branch off the public upstream that no competitor can replicate. The observed authorization and denial outcomes may end up the most defensible dataset of all, the accumulated knowledge of which documented criteria actually get enforced, which requirements trigger pends, which language predicts denials, where real payer behavior diverges from published policy, and which supporting documents lift approval odds, all of it carefully separated into published requirement versus structured interpretation versus customer specific workflow versus observed behavior so that anecdote never gets laundered into official policy. And the integrations, the EHR and SMART on FHIR and CDS Hooks and RCM and auth vendor connections, are paid because the open project tells you what the rules mean while the commercial product makes those rules execute inside a live workflow.

Eight. The community that turns ten thousand nurses into maintainers

A social layer makes sense here, but it should resemble a serious technical community rather than a consumer feed, because the goal is to take the thousands of people who currently interpret payer rules in isolation and wire them into a distributed maintenance network. Contributors establish expertise by role and domain rather than by posting volume, an auth nurse with a track record in advanced imaging and Florida Medicaid and Carelon workflows, a certified coder, a UM physician, a health law attorney, a FHIR engineer, each with a profile that shows accepted policy corrections, source citations added, test cases authored, and reviewer acceptance rate, and reputation built strictly on accepted evidence backed work rather than likes or popularity. Users watch the things they care about, a payer, a specific policy, a CPT family, a specialty, a state, a UM vendor, a drug, a site of care category, and their feed fills with policy commits, effective date changes, new auth requirements, changed documentation criteria, open ambiguities, payer clarifications, and reported implementation discrepancies, which is a legitimately useful product because the feed is operationally meaningful change rather than generic healthcare content.

Discussions attach to policies, and the responses get classified by epistemic status, source backed, official payer communication, operational observation, unverified anecdote, superseded, product specific, jurisdiction specific, so the platform never collapses ten specialists reporting the same payer behavior into an official coverage rule, because that consistent report is valuable intelligence and is also not a rule, and keeping those two things distinct is the entire trust proposition. Bounties let organizations fund the long tail, a few hundred dollars to structure and validate every current Florida Medicaid advanced imaging policy, a few thousand to build a complete test suite for Medicare home oxygen criteria, with specialty societies and patient advocacy groups and provider groups and RCM vendors and even health plans as potential sponsors, and the company taking a transaction fee or folding bounty administration into enterprise subscriptions, which is especially handy for policies that matter intensely to a narrow community but would never climb the central roadmap. Eventually a payer can claim and maintain its own repository with an official maintained badge, which creates a genuinely two sided dynamic where providers want reliable policies and payers want fewer junk submissions, but it introduces an obvious neutrality risk, and the only honest answer to that risk is radical provenance, where payer authored content is labeled as payer authored, community observations stay separate, the platform's own validation stays separate from payer certification, and customers can always see the disagreements and the open issues rather than a laundered consensus.

Nine. The thing prior auth has always secretly been: a dependency resolution problem

There is a second software analogy lurking under the GitHub one, and it might be the more useful frame for explaining what the product actually does. The company can resemble a package registry, npm or Docker Hub for coverage logic, where a customer installs a versioned policy package for a specific payer, product, and state, and the package ships with the structured requirements, code mappings, documentation templates, tests, provenance, effective dates, deprecation notices, and FHIR compatible artifacts. An application resolves a query by payer, product, state, code, and date and gets back the exact applicable package and version. This framing matters because prior authorization is, underneath all the human suffering, a dependency resolution problem, and almost nobody manages it as one. The real answer for a given order depends on the payer policy version plus the plan version plus the member benefit version plus the code set version plus the clinical guideline version plus the state rule version plus the federal rule version plus the UM vendor configuration, and

healthcare organizations resolve that dependency tree by hand, over the phone, with tribal memory, every single time. A package manager that pins those versions and resolves them deterministically is not a flashy product, but it is the kind of unsexy infrastructure that becomes load bearing and impossible to rip out once a workflow depends on it, which is exactly the position a durable company wants to occupy.

Ten. The moat hiding inside an open core, and the one way to blow the whole thing up

The reason the open strategy deepens rather than dilutes defensibility is that every community contribution mints a durable asset, a corrected mapping, a policy interpretation, a source citation, a synthetic test, a resolved ambiguity, a product specific variation, a payer clarification, a historical observation, a contributor reputation signal, and in aggregate the community is not just producing content, it is producing a labeled policy validation dataset. Over time the company learns which document structures produce extraction errors, which criteria are reliably ambiguous, which contributors are trustworthy in which specialties, which policies change most often, which machine interpretations need human review, which rule structures generalize across payers, and which changes actually disrupt operations, and all of that makes the automation better and the next policy cheaper to structure, which is a flywheel where more open repos bring more users who file more issues and corrections that produce better structured policies that make a more trustworthy enterprise product that drives more integrations and observed outcomes that yield better operational intelligence that makes the community more valuable. The eventual high value asset is not what the payer policy says, which is increasingly a commodity any model can read, it is the answer to which clinical and administrative facts are actually necessary to get this service approved under this payer product configuration, and that is genuinely hard to recreate because it requires the corpus, the network, the test suite, the provenance system, and the observed results all in one place. The model layer commoditizes. The graph does not.

The way to destroy all of this is to forget that the product can become dangerous the instant it presents a probabilistic interpretation as an authoritative coverage determination, because real coverage depends on the patient's exact benefit plan, product specific rules, state mandates, employer benefit design, delegated UM vendors, effective dates, contract terms, the actual medical facts, the coding, provider participation, and plain payer discretion, and no amount of elegant version control changes the fact that a confident wrong answer here causes real financial and clinical harm. So the company describes itself as a coverage requirement research and change

management system, never a guarantee of payment, and it resists the enormous gravitational pull toward automating the final approval determination, which is the most clinically sensitive, contractually tangled, and adversarial corner of the entire process. The first product stays in the lane of which policy appears to apply, which version applies on the relevant date, whether prior auth is required, what the published policy asks for, what relevant information is missing, what changed, what is ambiguous, and what evidence supports each answer, and that restraint is not a limitation, it is the whole business, because that is already an enormous market and it is the part where being right is achievable. The genuinely contrarian thesis is that the next important healthcare AI company may not automate prior authorization at all. It may build the version control, testing, and distribution infrastructure that every authorization system, including all the agents that are about to flood the space and each rebuild the same policy corpus and repeat the same extraction errors, quietly depends on. That is the more ambitious company, and it is the more defensible one, which is a rare and pleasant combination to find sitting in plain sight underneath a pile of forty seven page PDFs

.

← Previous

Next →

Discussion about this post

Comments Restacks

Write a comment...

Alex Openstone 2d

This is the right frame — and the analogy holds even more literally than you describe.

In CLIA/CAP-regulated clinical laboratory autoverification, the same problem exists one layer deeper: not just which policy applies, but whether a specific specimen disposition decision was made correctly, by what logic, and with what evidence — all auditable to the CAP inspector.

The implementation I've been building (LabInTrace) uses a knowledge graph with Git-bound schema. Schema changes, node/edge mutations, topology diffs — all committed, attributable, reversible. The audit trail isn't produced by wrapping decisions in a logger. It's a structural consequence of how the graph is maintained.

You note that FHIR can transport a rule but doesn't vouch for it. Same problem inside the lab: an LLM can traverse an autoverification state machine but can't prove it did so coherently. The KG enforces traversal as a hard execution constraint — not a retrieval source — which is what makes the audit record meaningful rather than decorative.

You've described the architecture. Here's what it looks like when the domain requires it structurally: https://doi.org/10.5281/zenodo.20348934

LIKE REPLY SHARE