AI tools say 'may make mistakes' at the bottom. That's fine for homework. It's unacceptable for business decisions.

Yamanu makes AI accountable to your governance.

Every answer is based on YOUR institutional reasoning, not statistical probabilities.

This business case positions Yamanu as a sector-defining platform that introduces the **Meaning Layer**—a new tier in the enterprise technology stack between data and applications. Traditional AI stacks fragment data and AI across separate platforms, each requiring costly integration. Yamanu unifies all layers (data, business logic, APIs, monitoring) into a single semantic architecture where meaning is encoded at the element level—eliminating translation boundaries where context is lost or governance breaks down. While Yamanu spans existing layers such as data management, APIs, and monitoring, these components converge to form an epistemic governance architecture that enterprises are currently trying to approximate through fragmented AI add-ons and human review after the fact.

This architecture rests on foundational principles. Shannon proved in 1948 that statistical inference transmits syntax, not semantics—meaning must be explicitly represented. Kernel methods in the 1990s showed that dimensional organization enables complex reasoning in high-dimensional spaces. Yamanu synthesizes both: explicit semantic encoding structured along domain-specific dimensional relationships (for example, Institution ↔ Region or Program ↔ Occupation in YouGo), now scalable through modern ML for construction, monitoring, and refinement guided by human reasoning.

The document argues that traditional AI stacks fail enterprises because they rely on statistical inference without provenance or determinism. Foundation models like GPT and Claude cannot guarantee correct answers or explain their reasoning—fatal flaws for regulated industries and mission-critical decisions. Current enterprise responses (fine-tuning models, implementing RAG) are inadequate: they're "sharpening a butter knife when a CNC laser is needed."

Yamanu's core insight: Organizations are fundamentally linguistic enterprises. Their policies, procedures, rules, and operations are all expressions of institutional language. By explicitly encoding this language into *Information Objects*—epistemic rules bound to authoritative meaning—organizations transform their knowledge system from an implicit, drifting consensus into an explicit, governed architecture. This combats **semantic entropy**: the gradual decay, drift, and contradiction of meaning that leads to misalignment and governance failure as messages lose coherence internally and externally.

Bureaucracy is the traditional and costly countermeasure to semantic entropy. Yamanu leverages AI to exponentially enhance epistemic coherence and augment institutional governance.

The document walks through each layer of the enterprise stack, comparing traditional products (Databricks, Azure, Boomi) with Yamanu's approach, demonstrating how epistemic governance delivers deterministic reasoning, complete provenance, continuous improvement through governance feedback loops, and 1000x cost reduction versus LLM-first architectures.

INSTITUTIONAL/ENTERPRISE STACK:

1. DATA & RETREIVAL (ETL, data lakes, RAG)

What it does: Stores and retrieves the organization's authoritative data sources, both internal and external.

- Internal: structured systems such as ERP, CRM, and transactional databases, plus unstructured assets like policies, SOPs, and institutional documents.
- External: public and regulatory datasets such as IPEDS (education outcomes), CPI-U (inflation indexing), BLS labor data, or industry-specific benchmarks.

This layer provides the raw material for all institutional reasoning—the facts to which business rules and Information Objects are later applied. Critical challenge: when multiple sources define the same concept differently (e.g., "completion rate" varies across IPEDS vs. state reporting), which source is authoritative? Traditional stacks handle this through manual ETL (extract, transform, load) logic that becomes obsolete when sources change. Its integrity and semantic consistency determine the credibility of every downstream decision. Traditional approaches fail at the atomic unit: they cannot preserve record-level provenance. When multiple datasets contribute to a single answer, which specific records—down to paragraph level—produced which parts of the output? Without this, verification and reproduction are impossible.

Leading Product: Databricks *Promise:* "Lineage, quality, control, and privacy across the AI workflow—a complete set of tools to deliver any AI use case." *Reality:* Still depends on brittle ETL pipelines and manual schema alignment. Every change requires re-coding, every inconsistency becomes technical debt, and "control" lives in scripts no one fully trusts. No systematic way to resolve semantic conflicts or track which data version informed which decision.

Yamanu (Yamanu Universal Data Layer (UDL)): Ingests datasets as immutable, versioned releases with cryptographic signatures. **Authority Rules explicitly encode precedence logic:** "For California institutions, prefer state dashboard over federal Scorecard when both exist." Conflicts surfaced automatically through semantic resolution. Management approves authority hierarchies once—system applies them automatically across all queries. Every data retrieval logs which dataset version, which authority rule, which element was selected, creating complete provenance from source to output. Critically, every data retrieval logs the specific record ID and element within that record—not just 'College Scorecard 2023' but 'Record SC-2023-122755-51.3801, element: earnings_4yr_median.' This atomic provenance enables independent verification: six months later, an auditor can retrieve that exact record and reproduce the calculation.

Yamanu advantages:

1. **Versioned immutability** – every dataset timestamped, signed, traceable; can reproduce any historical decision
2. **Explicit authority rules** – management governs precedence when sources conflict, documented and auditable
3. **Semantic resolution** – conflicts identified and resolved at ingestion, not buried in transformation code
4. **Provenance from source** – every downstream output traces to specific dataset versions and authority decisions
5. **Semantic efficiency** – store only semantically-mapped data elements (typically <5% of source files). Example: 103MB datasets over 5 years = 1GB traditional data lake vs 2MB Yamanu (99.8% reduction in storage and query costs)

Message: *Databricks tracks technical lineage (table → query → table). Yamanu tracks semantic lineage (authority → reasoning → output). The difference: Databricks tells you "this number came from table X." Yamanu tells you "this number came from CA Dashboard (Authority Rule AR-047) applying ROI Classification (Epistemic Rule ER-201 v1.0).*

2. **BUSINESS LOGIC & RULES *(Yamanu's Meaning Layer)***

What it does: Encodes institutional business rules—the guidelines that govern operations, decisions, and workflows. These rules translate policies, regulations, and domain expertise into executable logic that applications use to process data and make decisions.

Leading Product: Azure Logic Apps Rules Engine *Promise:* "Low-code decision management that lets non-programmers change business rules without code. Provides consistency, clarity, and compliance while avoiding AI hallucinations." *Reality:* Rules still operate on technical "facts" (XML, .NET objects). Doesn't resolve semantic conflicts between sources. Rules are workflow-specific, not institution-wide. No determinism guarantee—same inputs can produce different outputs depending on external system state. No provenance tracking or authority resolution.

Yamanu Information Objects (IO): IO Engine executes versioned epistemic rules against semantically-resolved UDL elements. **Execution Contract**: Given identical input parameters, dataset versions, authority rule versions, and epistemic rule versions, the system produces identical outputs and identical reasoning traces—provably and reproducibly. Rules bind to meaning (semantic elements), not schemas. Authority rules resolve conflicts automatically. Every execution generates cryptographically-signed provenance enabling independent verification. Orchestration logic embedded in IO specifications (IOs chain deterministically, no separate workflow layer needed).

1. Authority Rules select the correct data source (with provenance)

2. Information Objects bind that data to Epistemic Rules

3. Epistemic Rules execute deterministic reasoning

4. Complete chain logged: source → authority resolution → reasoning → output

Two-Phase Execution:

Yamanu executes queries in two phases that traditional systems cannot replicate:

Phase 1: Dimensional Context Selection Filter data using dimensional constraints before reasoning begins.

- Example: Query "nursing program completion rates"
- Dimensions applied: Language=EN, Jurisdiction=US→California, Type=Associate, Status=Active, Location=Bay Area
- Result: Context window = 47 programs matching all dimensional constraints

Phase 2: Query Execution Within Context Execute reasoning only against dimensionally-filtered data, with authority rules resolving conflicts.

- Authority rule: "For CA institutions, prefer CA Dashboard over Federal Scorecard"
- Return: Dimensionally-scoped, authority-resolved answer with complete provenance

This compositional approach—combining dimensional relationships to create novel contexts—is structurally impossible with vector similarity search or RAG, which cannot guarantee dimensional consistency or compose constraints deterministically.

Yamanu Advantages:

1. **Determinism guarantee** – execution contract ensures reproducible outputs; same inputs always yield same results
2. **Semantic binding** – rules reference institutional meaning, not technical schemas (survive source changes)
3. **Authority resolution built-in** – conflicting sources resolved automatically per management-approved precedence
4. **Complete provenance** – every decision traceable to authoritative sources, rule versions, and reasoning chain
5. **Orchestration included** – IOs chain together deterministically within this layer (no separate orchestration needed)

Message: *Azure executes rules on technical objects (XML, .NET types). Yamanu executes rules on semantic elements with authority resolution. Same inputs always yield same outputs (execution contract), enabling retrospective verification.*

3. **ORCHESTRATION & WORKFLOW** *(Collapsed into Yamanu's Meaning Layer)*

What it does: Chains AI model calls together into reusable workflows. Manages context, memory, and tool-calling across multiple LLM invocations to accomplish complex tasks.

Leading Product: LangChain *Promise:* "Model-neutral orchestration. See every step, iterate fast, swap models without rewriting. Future-proof your stack." *Reality:* Manages probabilistic workflows—you're orchestrating unpredictable LLM responses, testing prompt chains, handling model drift.

Yamanu: No separate orchestration layer needed. IO Engine handles orchestration deterministically within the Meaning Layer. IOs chain together (Program_ROI → Earnings_IO → Cost_IO) with full provenance, but execution is predictable—no prompt management, no context drift, no model swapping because reasoning isn't statistical.

Message: *LangChain manages probabilistic workflows (prompt chains, model responses, context drift). Yamanu eliminates the need for a separate orchestration layer because Information Objects chain deterministically—no prompt management, no retry logic, no context management. Orchestration becomes rule execution.*

4. APIs & INTEGRATION

What it does: Exposes business logic and data to applications through secure, scalable interfaces. Manages authentication, routing, monitoring, and documentation for how systems communicate.

Leading Product: Boomi API Management *Promise:* "Manage the entire API lifecycle—design, security, analytics. Scale to billions of transactions with geographic distribution, caching, and load balancing." *Reality:* Manages technical plumbing—routes, authentication, throttling. Doesn't govern what the APIs actually return. No semantic validation, no provenance of outputs, no guarantee that responses are institutionally correct.

Yamanu: APIs expose IO execution endpoints. Every call includes binding specification showing which authority rules applied, which epistemic rules executed, and complete provenance chain. APIs return deterministic outputs with cryptographic signatures enabling independent verification. No need for complex caching strategies because reasoning is fast and predictable.

Yamanu Advantages:

1. **Provenance-native APIs** – every response includes full audit trail to authoritative sources
2. **Deterministic execution** – same inputs always produce same outputs (simplifies testing, debugging)
3. **Semantic validation built-in** – responses guaranteed institutionally correct, not just technically successful
4. **Record-level response granularity** – API responses include specific record IDs for every data point, enabling atomic verification and forensic reconstruction

Message: *Boomi manages API plumbing (authentication, routing, throttling). Yamanu APIs return semantically validated outputs with cryptographic provenance. Technical success vs. institutional correctness.*

5. **INTERFACES & APPLICATIONS**

What it does: Delivers functionality to end users through web apps, mobile interfaces, dashboards, chatbots, and specialized tools. This is where institutional capabilities become tangible—students evaluating programs, executives reviewing metrics, staff executing workflows.

Leading Product: Microsoft Power Apps *Promise:* "Low-code app development. Build professional apps fast. Connect to any data source." *Reality:* Apps still inherit data quality problems from below. No guarantee outputs are institutionally correct. Users get technically functional interfaces but semantically inconsistent answers. Each app handles validation independently.

Yamanu: Applications (like YouGo) consume IO APIs and inherit deterministic reasoning, semantic consistency, and full provenance automatically. Every answer includes confidence levels, data source citations, and rule versions applied. Applications don't need complex validation logic—correctness guaranteed by **Yamanu's Meaning Layer**.

Yamanu Advantages:

1. **Inherited correctness** – apps get institutionally validated answers, not raw data
2. **Provenance at UI** – users see citations, sources, confidence without backend complexity
3. **Simpler app logic** – no data cleaning, conflict resolution, or validation code needed

Message: *Power Apps builds interfaces that inherit data quality problems from underlying systems. Yamanu applications consume IO APIs and automatically inherit deterministic reasoning, semantic consistency, and complete provenance. Validation happens in the Meaning Layer, not application code.*

Institutional AI Integration Features:

Monitoring and logging:

What it does: Tracks system performance, logs API calls, detects anomalies, and provides observability across the stack. Enables debugging, compliance audits, and operational intelligence.

Leading Product: Datadog / Splunk *Promise:* "Full-stack observability. Track every API call, detect anomalies, generate alerts. Centralized logging and analytics." *Reality:* Monitors technical metrics (latency, errors, throughput) but can't assess if outputs are institutionally correct. Alerts on system failures, not reasoning failures. No mechanism to improve underlying logic based on observed patterns.

Yamanu: Governance Dashboard monitors epistemic integrity, not just technical performance. ML analyzes execution telemetry to detect edge cases (statistical anomalies, semantic rule violations, provenance gaps). Routes flagged cases to human reviewers who refine epistemic rules. Creates a closed feedback loop where the system systematically learns what it does well and improves deterministically.

Yamanu Advantages:

1. **Epistemic observability** – tracks reasoning quality, not just technical uptime
2. **Governance feedback loop** – edge cases drive systematic rule refinement (compounding intelligence)
3. **Provenance audit trail** – every execution independently verifiable with cryptographic signatures

Message: *Datadog monitors technical metrics (latency, errors, throughput). Yamanu's Governance Dashboard monitors epistemic integrity—whether outputs are institutionally correct, not just technically successful. ML analyzes execution telemetry to detect semantic rule violations and route edge cases for human refinement.*

Digital Twin:

What it does: A digital twin is a structured, synchronized digital representation of how an organization operates—showing how processes unfold in real time and how decisions play out across workflows.

Leading Product: Siemens Teamcenter Promise: "Continuous data ingestion from operations, sensors, and systems of record to reconstruct physical and procedural activity." Reality: Powerful, but constantly reconciling inconsistent data semantics across dozens of systems. Meaning is assumed, not governed. The twin shows what happened, but not whether rules, definitions, or policies were applied consistently.

Yamanu: Builds a digital twin on a governed semantic foundation—institutional meaning defined once in the Meaning Layer and reused everywhere. Yamanu calls on the same underlying data sources, but:

- Data no longer needs endless reconciliation
- Semantics no longer drift across systems
- Every data element carries governed meaning
- Every rule, definition, and authority decision is encoded in IOs

IOs provide a clean, versioned map of institutional meaning that the digital twin uses to interpret events—no reverse-engineering intent from messy, conflicting system traces.

Yamanu Advantages:

1. Governed substrate – twin built on explicit meaning, not guesswork
2. Dramatically simpler maintenance – no constant semantic reconciliation
3. Reasoning visibility – shows why decisions were made, not just what happened
4. Coherence detection – surfaces where institutional meaning is breaking down

Message: *Siemens mirrors your operations. Yamanu does this and audits them at the same time. One shows the machine running. The other shows if the machine is reasoning correctly.*

Guardrails & Compliance:

Traditional Approach (AI Stack): Post-hoc filters scanning LLM outputs for tone violations, PII leakage, regulatory red flags. Catches problems after generation—blocking harmful content but wasting compute on outputs that get discarded.

Yamanu Approach: Preventive, not reactive. Epistemic rules encode regulatory constraints and institutional policies at the reasoning layer. System cannot generate non-compliant outputs because violations are structurally impossible. PII handling, disclosure requirements, and regulatory logic baked into IO specifications.

Example:

- Traditional: LLM generates student financial advice → guardrail flags FERPA violation → block output → retry
- Yamanu: IO specification prohibits accessing protected student records without authorization → violation cannot occur

Message: *Traditional guardrails scan LLM outputs post-generation (reactive filtering). Yamanu encodes regulatory constraints in epistemic rules at the reasoning layer (preventive governance). Violations are structurally impossible rather than caught after compute is wasted.*

Cost Controls & Economics:

Traditional AI Stack: Every query burns expensive API tokens. Complex question requiring multiple LLM calls = $2-5 per answer. Fine-tuning costs $50K-500K. Vector database operations add overhead. Costs unpredictable and scale linearly with usage. Large user bases = budget nightmare.

Yamanu: Deterministic IO execution costs pennies per query—standard database operations, not API tokens. No fine-tuning costs (epistemic rules replace training). LLM usage optional and minimal (only interface parsing). **Cost comparison: $0.002 vs $2.50 per complex query = 1,000x reduction.**

Samsung replaced $1M/month integration with predictable infrastructure costs. YouGo can answer thousands of student queries for less than one GPT-4 API call costs.

Yamanu Advantages:

1. **Predictable costs** – deterministic execution, no token surprises
2. **No fine-tuning expense** – rules replace training
3. **Scales economically** – cost grows with infrastructure, not per-query API fees

Message: *LLM-first stacks charge per API token (unpredictable, scales linearly with usage). Yamanu uses deterministic execution (database operations, not API calls). Cost is infrastructure-based and predictable. Complex query: $2.50 LLM-first vs. $0.002 Yamanu = 1000× reduction.*

Semantic Datalake Efficiency: Store Only What Has Institutional Meaning:

Example: Education sector client

Uses US Department of Education datasets:

- **IPEDS Completions:** 103MB file, ~3,000 institutions × 10,000 columns
- **NCES College Scorecard:** 103MB file, similar scale, different release schedule

Traditional Data Lake (5 years of history):

- Store complete files for each annual release
- 2 databases × 103MB × 5 versions = **1,030MB (~1GB) stored**
- Every query scans full dataset (millions of unused data points)

Yamanu UDL Approach:

1. Define institutional vocabulary: "We need completion rates, earnings, costs for specific award levels"
2. Semantic mapping: **20 columns per database** (out of 10,000) map to institutional concepts
3. Authority rules: "Prefer Scorecard for earnings data"
4. Dimensional filtering: Only relevant institutions, award types, time periods

Data stored:

- 20 columns / 10,000 columns = **0.2% of original data**
- 103MB × 0.002 = 0.206MB per version
- 2 databases × 0.206MB × 5 versions = **2.06MB total**

Savings vs Traditional:

- Storage: **99.8% reduction** (1,030MB → 2MB)
- Query performance: **500× faster** (scanning 2MB vs 1GB)

- Costs over 5 years:
    - Storage: $0.28 → $0.0005 (negligible)
    - Query costs (1,000 queries/month): **$60/year → $0.12/year** = **99.8% reduction**
    - Developer time: seconds to retrieve semantically-filtered data vs minutes parsing full datasets

Complete provenance maintained: Every data point traces to exact record in original 103MB source file.

The advantage: Semantic architecture front-loads the hard work (defining institutional meaning once) but creates permanent efficiency gains. Traditional data lakes grow linearly with every source addition (another 103MB file = another 103MB stored). Yamanu grows only when institutional vocabulary expands—adding new authoritative sources costs almost nothing if they map to existing semantic elements.

This compounds: AYear 6 adds another 206MB to traditional lake (now 1,236MB). Yamanu adds 0.4MB (now 2.4MB). The efficiency gap widens over time.

Security & Audit Architecture:

Traditional Approach: Security is bolted on after system design. Role-based access controls are added at the application layer, and data is encrypted at rest and in transit. Audit logs capture API calls but not reasoning chains. When a breach occurs, or an audit is required, reconstructing *who decided what, and why* demands a costly forensic investigation.

Yamanu: Security through architecture. Every IO execution generates a cryptographically signed provenance record that captures the complete reasoning chain—data sources accessed, rules applied, authority resolutions, and outputs produced. Governance Bus is append-only, tamper-evident. Role-based access encoded in authority rules (who can invoke which IOs, access which datasets). Audit trail exists by design, not configuration.

For regulated industries:

- Complete retrospective verification: reproduce any decision from any point in time
- Provenance signatures prove data integrity
- Authority rules document who authorized each data source
- Immutable governance log satisfies SOX, HIPAA, FERPA requirements

Yamanu Advantage: *Traditional systems log events. Yamanu proves reasoning—and makes proof cryptographically unforgeable.*

Relationship to Foundation Model Evolution:

Yamanu is architecturally positioned to benefit from—not compete with—advances in foundation models and transformers.

Current State: Modern transformers (DeepSeek innovations in Llama, attention mechanisms with expanding dimensional capacity) excel at pattern recognition, entity extraction, and high-dimensional similarity matching. Yamanu leverages these capabilities out-of-the-box in three places:

1. **Construction phase:** ML extracts semantic relationships from institutional documents
2. **Interface layer:** Transformers parse natural language queries into structured parameters
3. **Observation layer:** Pattern analysis across governance telemetry

What Transformers Cannot Provide: Statistical inference cannot encode institutional authority (Shannon, 1948). Transformers will never autonomously:

- Resolve authority conflicts ("Which source is correct when they disagree?")
- Guarantee deterministic execution contracts
- Generate cryptographic provenance chains
- Encode "this is what WE mean in OUR institutional context"
- Systematically refine reasoning through governance feedback

Strategic Advantage: As commodity transformers improve (more dimensions, better pattern recognition, faster inference), Yamanu automatically benefits: better query parsing, more accurate semantic extraction, faster edge case detection. Meanwhile, governance remains proprietary—the permanent architectural moat that makes AI outputs trustworthy.

Value Proposition: Transformers are infrastructure. Governance is a competitive advantage. Yamanu provides the governance layer enterprises lack, while using best-available foundation models as tools.

The Semantic Charter for Institutional Intelligence:

Implementing Yamanu requires organizations to take ownership of their institutional language. Every policy, procedure, rule, and operational standard is an expression of organizational meaning. Left implicit, that meaning drifts—words lose precision, and semantic entropy sets in. The resulting loss of meaning erodes corporate messaging and weakens epistemic coherence—the shared consistency between what an organization knows, says, and does.

Yamanu transforms institutional language from implicit consensus into explicit, governed architecture:

- **Authority rules** encode "who decides" when sources conflict
- **Epistemic rules** encode "what we mean" for domain concepts
- **Dimensional relationships** encode "how things connect" across contexts
- **Provenance records** encode "why we decided" for every output

Yamanu only delivers its full value when the organization adopts a linguistic governance mindset—just as ERP systems depend on GAAP and disciplined data tracking. It is not a configuration exercise; it is institutional governance architecture. Through Yamanu, the organization codifies its epistemology, establishing a semantic charter that governs how data becomes information, how information becomes knowledge, and how knowledge produces decisions.

The result: institutions gain deterministic control over their reasoning systems, with continuous improvement through governance feedback loops. They fight semantic entropy through explicit encoding, versioning, and provenance—building compounding epistemic intelligence that improves with use.

This is, in essence, what corporate governance has always tried to achieve—imperfectly, through human bureaucracy, policy manuals, and procedural compliance. Yamanu uses the power of AI to extend that discipline to a scale and precision never before possible. But it still depends on people; machines can amplify judgment, not replace it.

Appendix A

Technical Foundation: Terminology Guide

For readers familiar with our patent documentation, here's how marketing terms map to technical architecture:

Business/Marketing Term	Patent/Technical Term	What It Does
Meaning Layer	IO Engine	Executes versioned epistemic rules
Universal Data Layer (UDL)	UDL Store	Maintains versioned, immutable datasets with semantic resolution
Governance Dashboard	Management Layer	Visualizes reasoning patterns and detects anomalies
Authority Resolution	Authority Rule	Resolves conflicts when multiple sources define same concept
Provenance Trail	Provenance Record	Cryptographically-signed audit log of complete reasoning chain
Business Rules	Epistemic Rules	Machine-executable institutional reasoning standards
Data Governance	Governance Bus	Append-only event stream capturing all IO invocations

Appendix B

Note on "Dimensional": In traditional data warehousing, "dimensional modeling" refers to star schemas with fact and dimension tables (Kimball methodology). In Yamanu, "dimensional" refers to semantic relationship types that persist independently of any physical schema: - Institution ↔ Region (geographic dimension) - Program ↔ Occupation (career pathway dimension) - Award_Level ↔ ROI (economic dimension) These are first-class objects in Yamanu's architecture—meaning they're explicitly defined, versioned, and governed—not derived from table joins or statistical correlations. When source schemas change, dimensional relationships persist because they represent institutional semantics, not database structure.

Appendix C

Multi-Modal Data Handling

Traditional approach: - Convert images/audio/video to vector embeddings - Lose provenance (which specific frame? which speaker? which annotation?) - No semantic validation of outputs

Yamanu approach: - Store multi-modal data with semantic metadata as versioned elements - Authority Rules govern which image/audio sources are authoritative - Provenance tracks which specific asset (down to frame/timestamp) informed which decision - Epistemic Rules validate outputs regardless of input modality

Example: Medical Imaging System - IO validates that diagnosis references board-certified radiologist's annotated scan (provenance intact), not generic stock image - Authority Rule: "For breast cancer diagnosis, prefer radiologist-annotated mammograms over AI-generated synthetic images" - Provenance Record: "Diagnosis based on scan #SC-2024-1847, frame 23, radiologist ID #R-4472, annotation timestamp 2024-03-15T14:23:47Z" - Epistemic Rule: "IF confidence < 0.85 AND no radiologist confirmation THEN flag for human review"

Key difference: Traditional systems embed images as vectors and reason statistically. Yamanu maintains atomic provenance to the specific multi-modal asset that informed each reasoning step.

Appendix D

What is Data Intelligence?

Data intelligence = analyzing metadata to gain visibility into data quality, lineage, usage patterns, and business impact. Traditional approaches track technical metadata (table schemas, row counts, query performance). True data intelligence adds semantic context: which business decisions depend on which data, who has authority when sources conflict, and how data quality affects outcomes.

How Yamanu Implements Data Intelligence: Traditional tools (e.g., Datadog, Collibra) observe: - "Table X was queried 47 times today" - "Column Y has 3% null values" - "User Z accessed dataset A" Yamanu's Governance Dashboard observes: - "Epistemic Rule ER-201 was invoked 47 times, 3 resulted in edge cases" - "Authority Rule AR-047 resolved 12 source conflicts, favoring CA Dashboard in all cases" - "Program ROI calculations for nursing programs show 95% consistency, psychology programs show 23% edge case rate" The difference: Technical observability vs. epistemic observability. Yamanu tracks whether institutional reasoning is working correctly, not just whether systems are up.

Appendix E

****Cost comparison: $0.002 vs $2.50 per complex query = 1,000× reduction****

****Calculation methodology:****

LLM-first approach: - GPT-4 API pricing: ~$0.03 per 1K input tokens, ~$0.06 per 1K output tokens - Complex query example: "Calculate ROI for nursing programs at California community colleges, considering regional cost variations and labor market outcomes" - Typical execution: 3 model calls (query decomposition → data retrieval → synthesis) - Each call: ~1K tokens input, ~500 tokens output - Cost: (3 × 1K × $0.03) + (3 × 0.5K × $0.06) = $0.09 + $0.09 = ~$0.18 per query - Add vector database operations (~$0.05), orchestration overhead (~$0.02), retry logic (~$0.25 average) = ****~$0.50 to $2.50**** depending on complexity **Yamanu deterministic execution:** - Standard database query operations: ~$0.0001 per query (Postgres/MySQL typical costs) - IO Engine execution overhead: ~$0.0005 (computational cost of rule evaluation) - API response formatting and provenance generation: ~$0.0014 - Total: **~$0.002 per query** **Result:** $2.50 / $0.002 = 1,250× reduction (conservatively stated as 1,000×) **Note:** LLM costs assume enterprise API pricing. Consumer pricing would be higher. Yamanu costs based on AWS/Azure standard compute pricing.