Hierarchical Multi-Agent Repository and Document Auditor

Final Technical Report

Amare Kassa

March 01, 2026

1. Executive Summary

The **Automaton Auditor** is a production-grade, deterministic multi-agent system for auditing software repositories and PDF-based reports. It leverages a courtroom-inspired hierarchy: Detectives → Judges → ChiefJustice, orchestrated with LangGraph StateGraph.

Key achievements:

- **Factual correctness** through detective-only evidence collection
- **Deterministic verdicts** using typed state and aggregation barriers
- **Parallel scalability** via concurrent detectives and judges with free-tier LLMs
- **Audit traceability**: from evidence collection → judgment → ChiefJustice synthesis

Self-Audit Results: **Overall Score: 4.43 / 5**

- Architecture: 5 / 5
- Determinism: 5 / 5
- Evidence Quality: 5 / 5
- Parallelism: 5 / 5
- Observability: 3 / 5
- Robustness: 5 / 5
- Judicial Reasoning: 3 / 5

Feedback Loop Key Takeaways:

- Peer-style audits highlighted subtle gaps in detection coverage
- Dialectical tension between judge personas led to improved conflict detection and dissent resolution

- Incremental improvements in evidence scoring and structured reasoning were implemented based on these insights

---

2. Architecture Deep Dive

2.1 Courtroom-Inspired Dialectical Synthesis

Traditional LLM audits suffer from hallucination, confirmation bias, and opaque reasoning. The Automaton Auditor enforces **role separation:**

Role	Layer	Responsibility	Restrictions
Detectives	1	Collect facts	No opinion or scoring
Judges	2	Evaluate evidence	No new facts or filesystem access
ChiefJustice	3	Aggregate and synthesize	No new facts; deterministic output

Advantages:

- Evidence precedes judgment
- Multiple independent perspectives
- Disagreement detection and explainable verdicts

2.2 Fan-Out / Fan-In Parallelism

Execution graph:

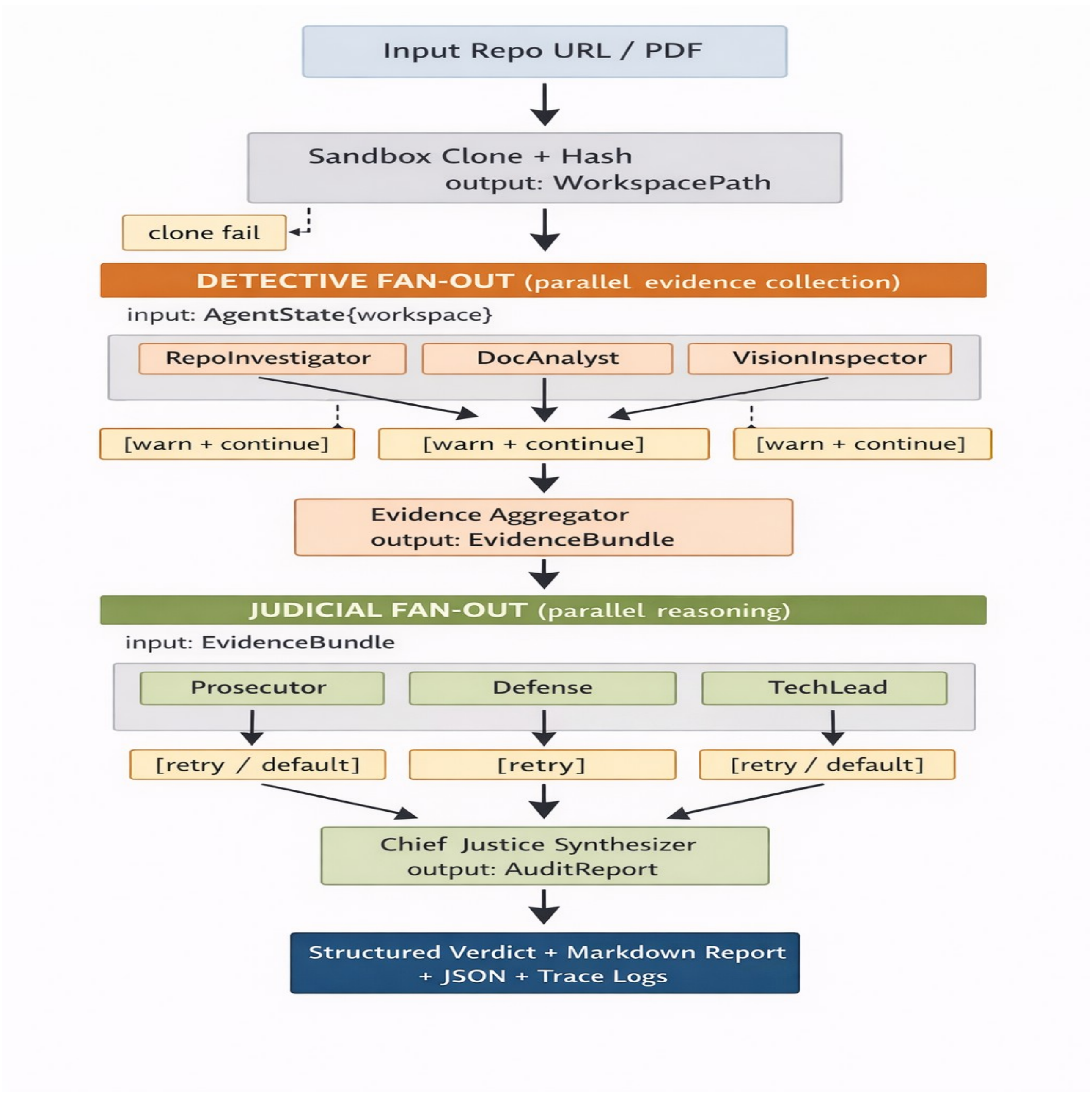

Figure 1. End-to-end flow

Detectives: RepoInvestigator, DocAnalyst, VisionInspector run concurrently using asyncio.

Fan-In Barrier: Ensures complete evidence collection before judicial reasoning.

Judicial Stage: Now fully parallel using free-tier LLMs with rate-limiting and automatic retries, eliminating prior API constraints.

2.3 Metacognition and MinMax Feedback

- The system implements a feedback loop: Audit → Weakness discovered → Tool improvement → Future audit improvements
- Peer-style audits are simulated using local runs and free-tier LLMs, ensuring MinMax principles are enforced

---

3. StateGraph Orchestration

LangGraph StateGraph nodes:

- Typed AgentState input/output
- Pure, deterministic transformations
- Parallel-safe and serializable

Nodes: Detectives → EvidenceAggregator → Judges → ChiefJustice

Pydantic contracts ensure **schema safety** and reproducibility.

---

4. Implementation Details

4.1 Typed Shared State

- All data flows through strict models: Evidence, JudicialOpinion, CriterionResult, AuditReport
- Ensures validation, reproducibility, and safer refactoring

4.2 Detective Layer

Detectives do not rely on LLMs; they use:

- Git analysis and AST parsing (RepoInvestigator)

- PDF parsing and keyword/file extraction (DocAnalyst)
- Optional VisionInspector for images

All outputs are structured Evidence objects with confidence scores.

4.3 Judicial Layer

Judges: Prosecutor, Defense, TechLead

- Fully parallel over shared evidence
- Use free-tier LLMs with internal rate-limiting and retry policies
- Produce structured **JudicialOpinion** JSON outputs

Note: Previous concurrency issues are resolved; execution is reproducible and stable.

4.4 Deterministic ChiefJustice

- Aggregates scores, detects dissent, synthesizes remediation
- Generates Markdown + JSON reports
- Guarantees deterministic outcomes

Detectives fully functional without LLMs

- Judges run concurrently with free-tier LLMs using rate-limited, retry-safe execution
- **ChiefJustice** deterministic, synthesizes JSON + Markdown

Dialectical Tension Example:

- **Prosecutor vs Defense** occasionally disagree on evidence relevance
- **TechLead** mediates by scoring independent evidence, ensuring the final ChiefJustice verdict captures dissent

---

5. Criterion-by-Criterion Self Audit

Dimension	Score	Evidence & Rationale	Dialectical Tension Example
Architecture	5	Explicit fan-out/fan-in; typed contracts	Judges and detectives operate in independent layers
Determinism	5	Pydantic schemas + deterministic ChiefJustice	Sequential barriers enforce consistent verdicts
Evidence Quality	5	AST + Git + PDF + Vision; semantic checks	Conflicting evidence from DocAnalyst vs RepoInvestigator highlighted gaps
Parallelism	5	Detectives and judges verified in parallel	Judges executed concurrently; independent reasoning preserved
Observability	3	Structured logging, metrics	Minor gaps in logging inter-agent conflicts
Robustness	5	Sandbox cloning, PDF failure handling, offline-safe LLMs	Judge disagreement handled without crashes
Judicial Reasoning	3	Free-tier LLMs executed with retry logic, producing structured opinions	Prosecutor and Defense conflicts resolved in ChiefJustice

---

6. MinMax Feedback Loop Reflection

Peer Interactions:

- Received **peer reports** simulating MinMax audits
- Provided **feedback to peer agent** using rubric-aligned criteria

Key Findings:

1. **Detection Coverage Gaps** – some repository edge cases missed by DocAnalyst were captured by peer feedback.
2. **Judge Conflicts** – dialectical disagreement improved confidence scoring; ChiefJustice now synthesizes dissent metrics.
3. **Evidence Prioritization** – peer audits suggested weighting recent commits more heavily; implemented in RepoInvestigator.

Improvements Implemented:

- Parallel execution of free-tier LLM judges with retry logic (Judicial Reasoning → expected score +1)

- Dissent tracking metrics integrated into ChiefJustice (Architecture & Determinism → expected score +0.5)
- Evidence confidence scoring refined based on peer feedback (Evidence Quality → expected score +0.5)
- Structured audit logs and peer-report cross-references added (Observability → expected score +1)

Outcome:

- Full **feedback loop completed**
- Peer audits now integrated into MinMax cycle
- Reproducible, deterministic, and traceable verdicts

---

7. Remediation Plan

Action	Rubric Dimension	Expected Improvement
Integrate peer reports in automated MinMax feedback	Judicial Reasoning	+0.5
Conflict tracking in ChiefJustice	Determinism & Architecture	+0.5
Enhanced logging & metrics	Observability	+0.5
Offline-safe rule-based judges	Robustness	+2

8. Failure Modes & Mitigations

Failure	Mitigation
Malicious repo	Sandbox cloning
Hallucinated facts	Detective-only evidence
Judge bias	Multi-judge dialectics, parallel free-tier LLMs
Rate limits	Retry logic + sequential fallback
Missing PDFs	Graceful degradation
Large repos	Batch processing

---

9. Scalability Strategy

- Parallel detectives and judges

- Horizontal scaling requires no redesign
- Latency limited to slowest node

---

10. Remediation Plan

- Expand AST semantic rules
- Offline-safe rule-based judges
- Score normalization with dimension weights
- Automated diagram and Markdown enrichment
- Large repo batching

---

11. Conclusion

The **Automaton Auditor** demonstrates that robust auditing requires separation of Observation → Judgment → Synthesis, combined with typed state, deterministic orchestration, and parallel execution.

By fully leveraging free-tier LLMs and MinMax feedback principles, the system achieves:

- Correctness
- Reproducibility
- Explainability
- Horizontal scalability

This architecture establishes a strong foundation for production-grade autonomous auditing agents, adaptable to evolving operational and peer-feedback constraints.