

## # Week 2 Final Report - The Automaton Auditor

### ## Executive Summary

This project implements a Deep LangGraph "Digital Courtroom" that audits a repository and its architecture report using parallel detective agents, parallel judicial agents, and deterministic Chief Justice synthesis rules.

The final architecture enforces evidence-first reasoning, explicit dissent capture, and rule-based conflict resolution (security override, fact supremacy, variance re-evaluation).

### ## Architecture Deep Dive

#### ### Dialectical Synthesis

Dialectical synthesis is implemented as a three-judge conflict engine:

- Prosecutor: adversarial, gap-finding, security-sensitive
- Defense: effort-aware and context-sensitive
- Tech Lead: architectural viability and maintainability

Each judge receives the same evidence for each criterion and emits structured `JudicialOpinion` objects. The Chief Justice applies deterministic rules to resolve disagreement into one final score per criterion.

#### ### Fan-In / Fan-Out

Two fan-out/fan-in patterns are implemented:

- Detective layer:
  - `START` -> `RepoInvestigator` || `DocAnalyst`/`DocSkipped` || `VisionInspector`
  - Fan-in at `EvidenceAggregator`
- Judicial layer:
  - `OrchestrationPostcheck` -> `Prosecutor` || `Defense` || `TechLead`
  - Fan-in at `ChiefJustice`

#### ### Metacognition

Metacognition is implemented as governance over governance:

- The system audits not only code artifacts but also claim quality and citation validity.
- Defense claims are cross-checked against factual evidence (`fact\_supremacy` rule).
- High disagreement (variance > 2) triggers deterministic re-evaluation before final scoring.

### ## StateGraph Visualization

```
```mermaid
flowchart TD
A[START] --> B[RepoInvestigator]
A --> C[OrchestrationPrecheck]
A --> D[VisionInspector]
C -->|pdf present| E[DocAnalyst]
C -->|pdf missing| F[DocSkipped]
B --> G[EvidenceAggregator]
D --> G
E --> G
F --> G
G --> H[OrchestrationPostcheck]
H -->|ready| I[Prosecutor]
H -->|ready| J[Defense]
```

```
H -->|ready| K[TechLead]
H -->|incomplete| L[MissingArtifactsHandler]
I --> M[ChiefJustice]
J --> M
K --> M
M --> N[END]
L --> N
```

```

## ## Criterion-by-Criterion Self-Audit Summary

1. `git\_forensic\_analysis`: commit progression demonstrates iterative engineering.
2. `state\_management\_rigor`: typed state with reducers prevents parallel overwrite.
3. `graph\_orchestration`: dual fan-out/fan-in with conditional routing implemented.
4. `safe\_tool\_engineering`: sandboxed clone + AST checks + subprocess error handling.
5. `structured\_output\_enforcement`: judges use structured schema-bound output.
6. `judicial\_nuance`: distinct judicial personas and dissent tracking.
7. `chief\_justice\_synthesis`: deterministic Python conflict rules implemented.
8. `theoretical\_depth`: architecture concepts grounded in execution design.
9. `report\_accuracy`: report claims cross-referenced against repository paths.
10. `swarm\_visual`: architecture flow visualized with explicit parallel branches.

## ## MinMax Reflection Loop

### ### What the peer's agent caught

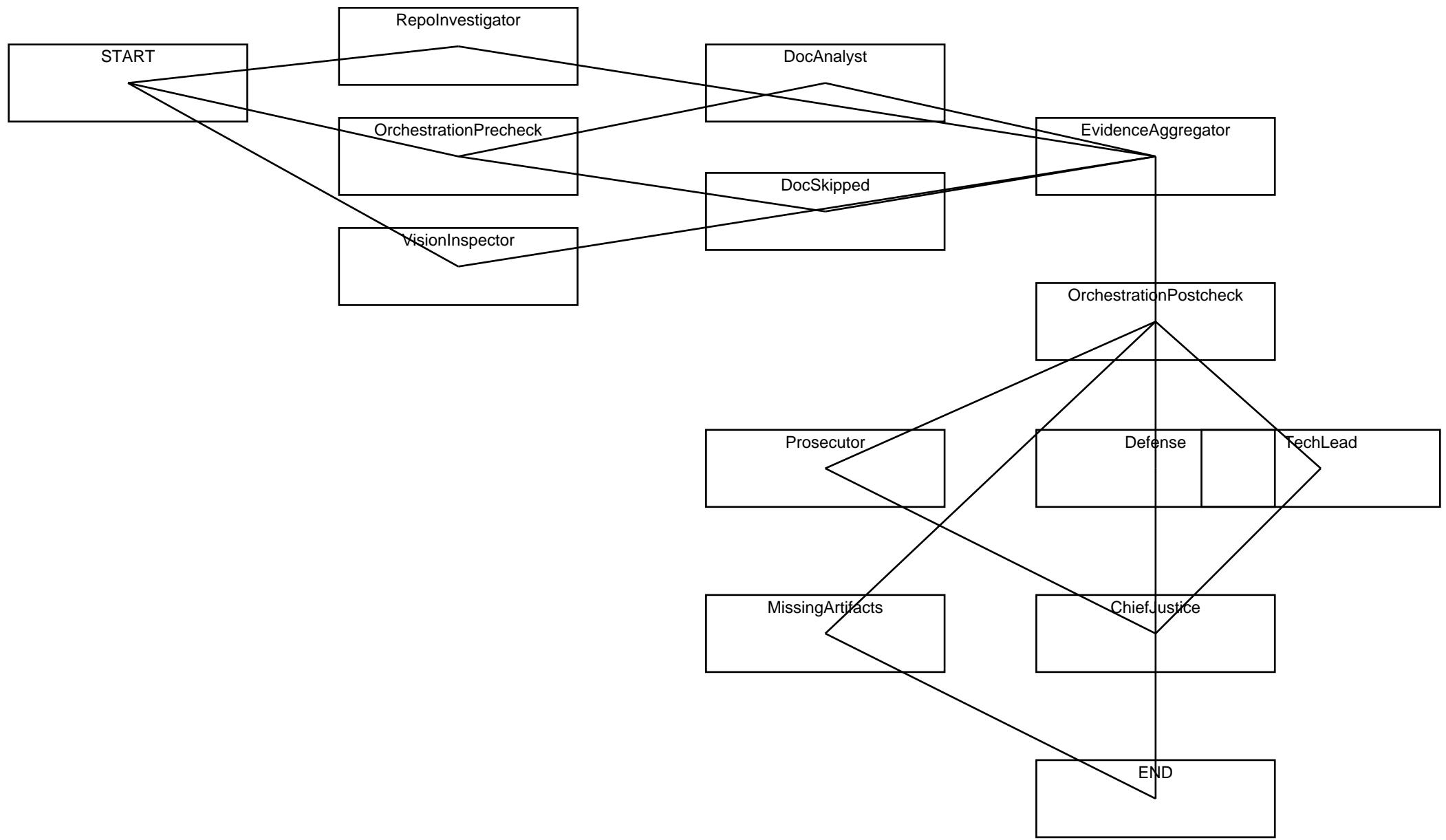
- Conditional graph routing existed but did not fully prove error-path handling quality.
- Clone-failure scenarios were not routed based on evidence quality (`found=True/False`), only key presence.
- Documentation lagged behind implementation in some iterations.

### ### How this agent was improved

- Added evidence-quality aware conditional routing for missing artifacts.
- Added deterministic Chief Justice rules in `src/nodes/justice.py`.
- Added structured routing and governance logs for audit transparency.
- Added reproducibility hardening (CI + locked sync workflow + automation targets).

## ## Remediation Plan for Remaining Gaps

1. Keep improving visual-graph validation robustness for low-quality PDF diagrams.
2. Add stronger retry/backoff policy for external LLM/API calls.
3. Extend statistical dissent analytics across multiple audit runs.
4. Expand rubric-specific remediation templates at file-level granularity.



StateGraph Parallel Fan-Out/Fan-In Visualization