ENGINEERING TECHNICAL NOTE

For Engineers

CGAM, Witness, Oracle Degradation, Human Gate, Rollback

Kotov Ivan
Bruxelles Belgique
2026

Document boundary

Status: Short engineering note. Technical implementation surface.
Boundary: Not legal advice. Not certification. Not a conformity assessment. Not a formal Article 50 compliance claim. Formal compliance remains deployment-specific and subject to legal review.

Academic PDF edition. Prepared for reviewable technical reading.

For Engineers

Status: Short engineering note. Technical implementation surface.
Boundary: Not legal advice. Not certification. Not a conformity assessment. Not a formal Article 50 compliance claim. Formal compliance remains deployment-specific and subject to legal review.

Purpose

This note describes a compact engineering pattern for AI-assisted workflows that may use local models, external frontier APIs, CLI/cloud agents, retrieval systems, validators, and human reviewers.

The problem is not only that an AI model may generate text. Modern AI-assisted workflows can create files, modify repositories, generate metadata, run validators, prepare release notes, build PDF/HTML artifacts, call tools, and affect what becomes public. Once agents can touch artifacts, the engineering question changes:

```
What was the agent allowed to do?
Which oracle or model was used?
What did the system change?
Was the change recorded?
Who reviewed it?
Who accepted responsibility?
Can the artifact be corrected or rolled back?
```

The proposed answer is a small control plane:

```
AI-assisted workflow
-> task contract
-> permission scope
-> sandbox / worktree
-> action
-> witness event
-> provenance sidecar
-> human review
-> human gate
-> release / rejection
-> rollback / correction route
```

This is not a replacement for legal review or compliance assessment. It is evidence infrastructure for engineers.

1. CGAM: C-Governed CLI Agent Mesh

The C-Governed CLI Agent Mesh, or CGAM, should be treated as a workflow governance layer for executable AI assistance. It is not the public-facing AI system by itself. Its job is to make AI-assisted tool use bounded, inspectable, and reversible.

Minimum CGAM objects:

```
task_contract.yaml       # what the agent is asked and allowed to do
permission_scope.yaml    # read/write/execute/publish/secret boundaries
sandbox_record.json      # worktree, temp dir, container, or execution boundary
witness.jsonl            # append-only boundary events
provenance.sidecar.json  # machine-readable artifact route
human_review.yaml        # what human reviewed, accepted, rejected, or froze
release_manifest.json    # final artifact inventory
rollback_plan.md         # correction and restoration path
```

A task contract should be created before agent work starts. It should name the system, artifact, allowed actions, forbidden actions, required human gate, and responsible human or organisation.

Example:

```
task_id: "CGAM-DEMO-001"
system: "AI-assisted publication workflow"
artifact: "input/draft.md"
agent_mode: "bounded_assistance"
allowed_actions:
  - "suggest edits"
  - "create metadata"
  - "prepare release artifacts"
forbidden_actions:
  - "publish without human gate"
  - "access secrets"
  - "silently remove disclosure"
  - "claim legal compliance"
human_gate_required: true
responsible_human: "Kotov Ivan"
```

The principle is simple: no agent should have more authority than the task requires.

2. Witness events

A witness event records that a boundary transition occurred. It is not a diary, not a chat transcript, not surveillance, and not a substitute for user-facing disclosure.

A useful witness event is small:

```
{
  "event_id": "WIT-20260620-001",
  "time": "2026-06-20T14:00:00+02:00",
  "actor": "agent",
  "action": "draft_assisted",
  "artifact": "input/draft.md",
  "sha256": "example_hash",
  "oracle_id": "local_llm_or_mock",
  "permission_scope": "contracts/permission_scope.yaml",
  "human_gate_required": true,
  "privacy_class": "boundary_metadata"
}
```

The event should answer only the audit question: what transition happened, to which artifact, under which boundary, and with which evidence reference. It should not store private prompts, secrets, full conversations, or unnecessary personal data.

Witness gives engineers an audit spine. It does not prove truth, legality, safety, or wisdom. It proves that a recorded transition exists and can be inspected.

3. Provenance sidecar

A provenance sidecar describes the artifact route. It is machine-readable, versioned, and attached to the build or release package.

Minimum fields:

```
{
  "artifact_id": "ARTICLE-001",
  "artifact_type": "markdown_to_html_pdf",
  "source_files": ["input/draft.md"],
  "generated_files": ["output/article.html", "output/article.pdf"],
  "ai_assistance": true,
  "oracle_id": "local_llm_or_mock",
  "agent_assistance": true,
  "witness_log": "witness/witness.jsonl",
  "human_review_record": "review/human_review.yaml",
  "release_manifest": "release/release_manifest.json",
  "visible_disclosure_required": true,
  "legal_review_completed": false
}
```

Do not confuse provenance with disclosure. A sidecar helps machines, reviewers, and auditors. A person still needs a visible notice where the workflow or publication requires it.

4. Oracle degradation

An oracle is any replaceable capability provider: local LLM, external frontier API, retrieval system, judge model, validator, database, search service, expert system, or human expert.

No oracle should define identity, authority, responsibility, or release state. Oracles provide candidate signals. The workflow control plane decides how those signals are accepted, rejected, downgraded, or routed.

Recommended oracle tiers:

```
Tier 0: mock oracle / no model
Tier 1: local small model
Tier 2: local medium model
Tier 3: EU-hosted or institutional model
Tier 4: external frontier API
Tier 5: high-risk specialised oracle requiring separate review
```

Degradation protocol:

```
1. Detect oracle failure, restriction, timeout, or policy block.
2. Record oracle_unavailable witness event.
3. Stop external calls to the unavailable oracle.
4. Switch to local fallback, mock mode, cached context, or human-only review.
5. Mark capability downgrade in provenance sidecar.
6. Prevent silent equivalence: downgraded output must not be presented as identical to full-oracle output.
7. Require renewed human review before release.
8. Preserve witness, provenance, task contract, review state, and rollback route.
```

The core test is:

```
oracle loss may reduce capability;
oracle loss must not erase evidence, responsibility, or review state.
```

5. Human gate

The human gate separates generation from acceptance. It should be explicit, recorded, and scoped.

A useful human review record should contain:

```
review_id: "HR-001"
reviewer: "Kotov Ivan"
role: "Author / responsible human reviewer"
artifact:
  - "output/article.html"
  - "metadata/provenance.sidecar.json"
scope:
  - "visible disclosure"
  - "claim boundary"
  - "provenance sidecar"
  - "witness events"
  - "release manifest"
decision: "approved_for_demo_release"
legal_review_completed: false
correction_route: "release/CORRECTION_LOG.md"
```

A checkbox saying “human reviewed” is weak. The record should say who reviewed, what was reviewed, under which scope, what decision was made, and where correction happens.

6. Rollback and correction

Rollback is not shame. It is an engineering requirement.

Rules:

```
Do not silently overwrite released artifacts.
Do not delete bad evidence to look cleaner.
Do not hide oracle downgrades.
Do not pretend a corrected artifact was the original artifact.
```

A correction path should:

```
1. Record issue.
2. Identify affected artifact.
3. Create correction witness event.
4. Update artifact or revoke release.
5. Recompute hashes.
6. Update release manifest.
7. Require human review.
8. Publish correction note where needed.
```

Minimal release package:

```
release/
  RELEASE_MANIFEST.json
  SHA256SUMS
  CORRECTION_LOG.md
  ROLLBACK_PLAN.md
```

Explicit engineering bridge

The bridge is:

```
model / oracle output
-> bounded agent action
-> witness event
-> provenance sidecar
-> human gate
-> release manifest
-> correction or rollback path
```

This bridge makes AI-assisted workflows inspectable without pretending that logs, metadata, or hashes are legal compliance.

Quiet bridge I: control variety

A workflow that can generate, edit, execute, publish, and revise needs matching controls. A single label cannot control all failure modes. Use task contracts for authority, permission scopes for capability, sandboxing for mutation control, witness for boundary events, provenance for artifact route, human gates for responsibility, and rollback for repair.

Quiet bridge II: information discipline

Evidence should carry enough signal for review without becoming a privacy leak. Store boundary metadata, not private life. Store artifact route, not unnecessary raw prompts. Store human decision scope, not personal confession. Engineers should design the evidence channel, not dump everything into logs because storage is cheap.

Earth paragraph

In a workshop, a robot arm may cut metal, a measuring tool may verify the part, a logbook may show the operation, and a foreman may sign the job card. These are different functions. If the robot fails, the tool lies, the logbook is missing, or the foreman never looked at the part, the finished object is not trustworthy. AI-assisted workflows have the same structure. The model is the machine, the witness is the logbook, provenance is the routing tag, the human gate is the foreman’s signature, and rollback is the spare part shelf. No romance. Just a system that can be inspected when something goes wrong.

Final engineering rule

Do not ship from:

```
agent output
```

to:

```
public artifact
```

without:

```
task contract
permission scope
sandbox or worktree boundary
witness event
provenance sidecar
human review record
human approval gate
release manifest
rollback / correction route
```

The model may be powerful. The workflow still needs brakes.