Intent-Driven Hook Architecture Report

Date: 2026-02-21
Repository: Roo-Code extension
Scope: Intent handshake, pre/post hook middleware, semantic traceability, optimistic locking, and safety guardrails.

1) Executive Summary

This implementation resolves the "context paradox" between asynchronous IDE interaction and synchronous LLM tool execution by introducing an explicit intent checkout handshake and deterministic hook enforcement.

The delivered system now enforces:

- Intent-first mutation flow (‘select_active_intent‘ before writes/edits).
- Strong pre-hook controls (scope, checkout stage, HITL, sidecar, intent ignore, stale-file, spec-iteration guard).
- Post-hook semantic ledgering (‘agent_trace.jsonl‘ with integrity chaining and mutation-class attribution).
- Recovery and learning mechanisms (structured hook errors; verification-failure lessons captured in the repo knowledge base ‘AGENT.md‘).

2) Implemented Architecture

2.1 Components

1. Prompt Handshake Contract

- File: ‘src/core/prompts/system.ts‘
- Injects a mandatory intent protocol in the system prompt:
    - First mutating flow must call ‘select_active_intent‘.
    - Mutating tools forbidden until checkout succeeds.

2. Intent Selection Tool + Context Loader

- Tool schema: ‘src/core/prompts/tools/native-tools/select_active_intent.ts‘
- Runtime tool: ‘src/core/tools/SelectActiveIntentTool.ts‘
- Context source: ‘.orchestration/active_intents.yaml‘ via ‘IntentContextService‘.
- Hook interception (‘HookEngine.preToolUse‘) constructs and injects ‘<intent_context>‘ XML for selected intent.

3. Hook Middleware (Pre/Post)

- File: ‘src/hooks/HookEngine.ts‘
- Pre-hook controls:
    - Command classification (‘safe‘ vs ‘destructive‘).
    - Two-stage checkout state machine (‘checkout_required‘ ? ‘execution_authorized‘).
    - Contract validation for ‘.orchestration‘ control-plane files.
    - Sidecar policy (‘constraints.sidecar.yaml‘) and deny-list checks.
    - Workspace boundary and owned-scope enforcement.
    - ‘.intentignore‘ intent exclusion.
    - HITL modal/ask gate (‘Approve/Reject‘).
    - Optimistic lock stale-file blocking.
    - Numbered spec creation guard (new ‘specs/NNN-*‘ requires ‘INTENT_EVOLUTION‘).
- Post-hook controls:
    - Governance ledger append.
    - Intent history/status updates.
    - Semantic trace serialization to ‘agent_trace.jsonl‘.

4. Orchestration Persistence Layer

- File: ‘src/hooks/OrchestrationStore.ts‘

- Manages:
  - ‘active_intents.yaml‘
  - ‘agent_trace.jsonl‘
  - ‘intent_map.md‘
  - ‘governance_ledger.md‘
  - ‘constraints.sidecar.yaml‘
  - ‘.intentignore‘
- Validates agent trace schema with Zod and adds SHA-256 chain integrity (‘prev_record_hash‘, ‘record_hash‘).

5. Dispatch Shim for Write Robustness

- File: ‘src/core/assistant-message/presentAssistantMessage.ts‘
- ‘normalizeWriteToFileToolUseForDispatch(...)‘ behavior:
  - If ‘write_to_file.intent_id‘ missing and ‘task.activeIntentId‘ exists: auto-fill.
  - If ‘mutation_class‘ missing: default ‘AST_REFACTOR‘.
  - Hard fail only if no active intent exists.

6. Optimistic Concurrency + Lessons Learned

- Read hash capture: ‘src/core/tools/ReadFileTool.ts‘
- Turn-scoped hash store: ‘src/core/task/Task.ts‘
- Stale-file detection pre-write: ‘src/hooks/HookEngine.ts‘
- Verification failure lesson append: ‘src/core/tools/ExecuteCommandTool.ts‘ (repo knowledge base target: ‘AGENT.md‘)

3) Schemas and Contracts

3.1 Native Tool Schema: ‘select_active_intent‘

Source: ‘src/core/prompts/tools/native-tools/select_active_intent.ts‘

```json
{
	"name": "select_active_intent",
	"parameters": {
		"type": "object",
		"properties": {
			"intent_id": { "type": "string" }
		},
		"required": ["intent_id"],
		"additionalProperties": false
	}
}
```

3.2 Native Tool Schema: ‘write_to_file‘

Source: ‘src/core/prompts/tools/native-tools/write_to_file.ts‘

```json
{
	"name": "write_to_file",
	"parameters": {
		"type": "object",
		"properties": {
			"path": { "type": "string" },
			"content": { "type": "string" },
			"intent_id": { "type": "string" },
			"mutation_class": {
				"type": "string",
				"enum": ["AST_REFACTOR", "INTENT_EVOLUTION"]
			}
```

```
            },
            "required": ["path", "content", "intent_id", "mutation_class"],
            "additionalProperties": false
        }
}
```
‘‘‘

3.3 Agent Trace Record Schema

Source: ‘src/hooks/OrchestrationStore.ts‘

Record includes:

- ‘id‘, ‘timestamp‘, ‘vcs.revision_id‘
- ‘files[]‘ with:
    - ‘relative_path‘
    - optional ‘ast_fingerprint‘ (‘parser=tree-sitter‘, ‘summary_hash=sha256:...‘)
    - ‘conversations[]‘:
        - ‘url‘, ‘contributor‘
        - ‘ranges[]‘ (‘start_line‘, ‘end_line‘, ‘content_hash‘)
        - ‘related[]‘: ‘specification‘ and optional ‘mutation_class‘
- optional ‘integrity‘:
    - ‘chain=sha256‘
    - ‘prev_record_hash‘
    - ‘record_hash‘

Validation:

- Zod schema checks UUID, timestamp format, hash regex, range semantics.

4) Detailed Agent Flow (Turn Lifecycle)

4.1 Stage A: Intent Declaration

1. Model analyzes user request.
2. Model calls ‘select_active_intent(intent_id)‘.
3. Pre-hook intercepts and loads intent from active intents store.
4. Hook injects ‘<intent_context>‘ XML (scope + constraints).
5. ‘SelectActiveIntentTool‘ authorizes checkout for the turn (‘execution_authorized‘).

4.2 Stage B: Tool Dispatch and Pre-Hook Gating

For each tool call:

1. Dispatch shim normalizes ‘write_to_file‘ args (intent/mutation defaults).
2. Pre-hook evaluates:
    - mutating tool classification
    - checkout stage gate
    - orchestration contract drift
    - sidecar blocked tools / denied mutation paths
    - out-of-workspace mutations
    - active intent validity
    - ‘.intentignore‘ pattern exclusion
    - ‘owned_scope‘ membership
    - stale file check (optimistic lock)
    - numbered spec creation guard (‘specs/NNN-*‘ + mutation class)
    - HITL approval
3. On deny: standardized hook error or explicit scope message is returned.

4.3 Stage C: Tool Execution

- Tool executes only after passing pre-hook.
- ‘write_to_file‘ still performs its own required-param checks and approval flow.

4.4 Stage D: Post-Hook Audit and Trace

1. Governance ledger append (‘OK|FAILED|DENIED‘ metadata).
2. Intent ‘recent_history‘ update.
3. For successful mutating tools:
    - Build trace record with content hash and optional AST fingerprint.
    - Inject ‘specification‘ and ‘mutation_class‘ in ‘related[]‘.
    - Append to ‘agent_trace.jsonl‘ with chain integrity.
4. ‘attempt_completion‘ success marks intent ‘COMPLETED‘.

5) Guardrails Implemented (What Blocks What)

1. Missing Intent Gate

- Error: ‘You must cite a valid active Intent ID.‘
- Trigger: no active intent or invalid id for mutating tool.

2. Checkout Stage Gate

- Error: ‘PreToolUse denied ... intent checkout required for this turn.‘
- Trigger: mutating tool before ‘select_active_intent‘ in current turn.

3. Scope Gate

- Error: ‘Scope Violation: <REQ/INTENT> is not authorized to edit <path>. Request scope expansion.‘
- Trigger: write outside ‘owned_scope‘.

4. Intent Ignore Gate

- Structured error code: ‘INTENT_IGNORED‘
- Trigger: active intent matched in ‘.intentignore‘.

5. Stale File Gate

- Structured error code: ‘STALE_FILE‘
- Trigger: file changed since it was read this turn.

6. HITL Authorization Gate

- Structured error code: ‘HITL_REJECTED‘
- Trigger: user reject in modal/ask flow.

7. Spec Iteration Gate (new)

- Structured error code: ‘SPEC_ITERATION_REQUIRES_INTENT_EVOLUTION‘
- Trigger: creating new ‘specs/NNN-*‘ file with non-‘INTENT_EVOLUTION‘ mutation class.

6) Traceability and Integrity Notes

- Content hashing:
    - Reads/writes use SHA-256 for stale-file checks and trace ranges.
- Chain-of-custody:
    - ‘agent_trace.jsonl‘ links entries via ‘prev_record_hash‘ -> ‘record_hash‘.
- Semantic attribution:
    - ‘mutation_class‘ persisted for downstream audits (‘AST_REFACTOR‘ vs ‘INTENT_EVOLUTION‘).
- Scope + decision provenance:
    - ‘governance_ledger.md‘ records status, tool, intent, touched paths, sidecar constraints.

7) What Has Been Achieved

Completed Capabilities

- Intent-first protocol integrated into system prompt and runtime execution.
- Deterministic pre-hook governance over all mutating flows.
- Formal tool schema enforcement for intent + mutation semantics.
- Automatic write shim to reduce model/tooling friction.
- Scope and boundary protection with actionable errors.
- HITL approval integrated into pre-dispatch gate.
- Optimistic locking to prevent parallel overwrite races.
- Agent trace ledger with semantic + cryptographic integrity metadata.
- Verification-failure lesson persistence to ‘AGENT.md‘.
- New numbered-spec creation guardrail tied to mutation semantics.

Operational Impact

- Eliminates silent context drift between user intent and write execution.
- Converts failures into structured recoverable tool errors.
- Increases trust with auditable, hash-linked mutation records.
- Supports safe parallel workflows by blocking stale writes.

8) Validation Evidence

Focused tests and checks executed for this implementation:

- Typecheck: ‘pnpm -C src check-types‘
- Tests:
  - ‘src/hooks/__tests__/HookEngine.spec.ts‘
  - ‘src/core/assistant-message/__tests__/presentAssistantMessage-writeToFile-shim.spec.ts‘
- Result: Passing (including new spec-iteration deny/allow tests).

9) Known Constraints / Follow-ups

1. ‘apply_diff‘ matching is brittle in high-churn sections (‘recent_history‘); safer patch anchors are recommended.
2. The write shim applies only to ‘write_to_file‘; similar normalizers could be added for future schema-evolving mutators.
3. Telemetry and governance rollups are implemented (‘src/hooks/GovernanceRollupService.ts‘) and can be further extended with webview dashboard visualizations.

10) End to End architecture

```
flowchart TD
subgraph UI_Plane [UI Plane]
U[User]
WV[Webview UI]
end

    subgraph Logic_Plane [Logic Plane]
        WMH[webviewMessageHandler]
        CP[ClineProvider]
        TK[Task Runtime]
        API[Assistant API]
        NTP[NativeToolCallParser]
        DISPATCH[presentAssistantMessage]
        HOOK[HookEngine]
        STORE[OrchestrationStore]
    end

    subgraph Execution_Plane [Execution Plane]
        TOOLS[Tool Implementations]
        FS[Workspace File System / Terminal / MCP]
    end

    subgraph Persistence_Plane [Persistence Plane]
        ORCH[".orchestration/*"]
        ACTINT["active_intents.yaml"]
        SIDECAR["sidecar.yaml"]
        TRACE["agent_trace.jsonl"]
        MAP["intent_map.md"]
        AGENT["AGENT.md / shared brain"]
    end

    %% Connections
    U -->|interacts| WV
    WV -->|postMessage| WMH
    WMH --> CP
    CP --> TK
    TK -->|build prompt & tools| API
    API -->|streamed response| NTP
    NTP --> DISPATCH

    DISPATCH -->|preToolUse| HOOK
    HOOK -->|policy & state| STORE
    STORE --> ORCH
    HOOK -- allow/deny --> DISPATCH

    DISPATCH -->|execute| TOOLS
    TOOLS --> FS
    TOOLS -->|results| DISPATCH

    DISPATCH -->|postToolUse| HOOK
    HOOK -->|write logs| STORE
    STORE --> ORCH
    ORCH -->|data| STORE
```

11) File Index (Primary Implementation)

- ‘src/core/prompts/system.ts‘
- ‘src/core/prompts/tools/native-tools/select_active_intent.ts‘
- ‘src/core/prompts/tools/native-tools/write_to_file.ts‘
- ‘src/core/tools/SelectActiveIntentTool.ts‘
- ‘src/core/tools/WriteToFileTool.ts‘
- ‘src/core/assistant-message/presentAssistantMessage.ts‘
- ‘src/hooks/HookEngine.ts‘
- ‘src/hooks/OrchestrationStore.ts‘
- ‘src/hooks/IntentContextService.ts‘
- ‘src/core/task/Task.ts‘
- ‘src/core/tools/ReadFileTool.ts‘
- ‘src/core/tools/ExecuteCommandTool.ts‘
- ‘src/hooks/__tests__/HookEngine.spec.ts‘
- ‘src/core/assistant-message/__tests__/presentAssistantMessage-writeToFile-shim.spec.ts‘

---

github link : https://github.com/Gersum/Roo-Code-10x.git