# Agentic Engineering ? ALAI AI Factory Roadmap (2026-05-26)

# Agentic Engineering ? ALAI AI Factory Roadmap

**Date:** 2026-05-26
**Source video:** https://www.youtube.com/watch?v=2KcITKKJikA
**Video title verified via yt-dlp:** “Top #1 Opportunity for Senior Engineers: Agentic Engineering”
**Channel:** IndyDevDan
**Duration:** 1582 seconds (~26m 22s)
**Transcript evidence:** `/tmp/alai/youtube-2KcITKKJikA/2KcITKKJikA.en.vtt`
**Related ALAI closure evidence:** `/tmp/alai/p2p-ai-factory-v1-closure-20260526.md`

## Executive summary

The video’s core thesis is that senior engineers should stop treating AI as one-off “vibe coding” and instead build **agentic engineering systems**: harnesses, software factories, verifier loops, always-on agents, and domain-specific agent teams.

ALAI already has most foundations:

- Mission Control for task state and gates.
- Virtual companies for domain routing.
- Event Bus for async workflow.
- Company Mesh for P2P agent communication.
- P2P Pair Programming V1: main coder + independent verifier before final QA.
- BookStack and discover.js for operational knowledge.
- Memory / RAG plumbing, with LightRAG intended as canonical graph backend.

The missing layer is not “another agent”; it is a **clean AI Factory Experience Layer** that turns these components into a repeatable operator workflow and visible product.

# Video-derived pillars mapped to ALAI

| Video pillar | Meaning | ALAI current equivalent | Gap |
|---|---|---|---|
| Agent harnesses | Own the environment around the model, not just prompts | Pi, Claude Code hooks, skills, tools, prompt injection | Need a polished factory command/UI |
| Software factories | Build the system that builds the system | MC + Event Bus + virtual companies + Company Mesh | Need standard workflow runner and metrics |
| Extensible software | Agents improve through tools/hooks/skills | `~/system/tools`, Pi skills, hooks, BookStack | Need clearer extension templates and test gates |
| Always-on agents | Agents run in background and react to events | LaunchAgents, daemons, event handlers, MC resolver | Reliability backlog and stalled-task recovery |
| Agentic access | Give agents safe access to context and tools | discover.js, BookStack, LightRAG, memory, MC evidence | LightRAG health must be reliable/default-on |
| Verifier harness | Independent agent checks another agent | P2P Pair Programming V1 + Proveo + MC gates | Need metrics and controlled expansion |

# Current ALAI baseline

## Already done

1. **P2P Pair Programming V1 closed**
   - Evidence: `/tmp/alai/p2p-ai-factory-v1-closure-20260526.md`
   - Default: prewire + prompt injection + MC ready/done gate.
   - Deferred: auto Company Mesh send at dispatch.
2. **Company Mesh exists**
   - Used for bounded peer verifier loops.
   - Mission Control can require mesh evidence for risky tasks.
3. **Virtual companies exist**
   - CodeCraft, Vizu, FlowForge, Proveo, Securion, AgentForge, etc.
   - Routing source: `node ~/system/tools/discover.js routing "<task>"`.
4. **Knowledge system exists**
   - BookStack for canonical docs.
   - discover.js for tool-first lookup.
   - LightRAG wrapper exists, but current live status check timed out on 2026-05-26.

# Strategic recommendation

Build **ALAI AI Factory V2** as an internal product first.

Do not start by building an external SaaS. First make the internal factory flow undeniable:

```
CEO idea/request
  → Mission Control parent task
  → plan/spec page in BookStack
  → route to virtual company
  → main coder + P2P verifier
  → final QA gate
  → evidence package
  → demo dashboard/status
  → memory/RAG writeback
```

# Target experience

Alem should be able to say:

> “Napravi product demo za Bilko mobile companion.”

And the factory should produce:

1. MC parent task and subtasks.
2. Architecture/spec page in BookStack.
3. Routed builder/verifier companies.
4. Pair-programming pre-verifier thread where required.
5. Evidence paths, cost, progress, blockers.
6. Final QA review before “done”.
7. Knowledge writeback to BookStack + memory/RAG.

# Implementation roadmap

## Phase 0 — report + tracking (today)

- Create this roadmap/report.
- Publish it to BookStack.
- Create MC tracking task.
- Confirm memory/LightRAG current status.

# Phase 1 — Factory workflow MVP (1–3 days)

Deliver a single command or documented workflow:

```
node ~/system/tools/ai-factory.js start "<goal>" --priority H --domain
backend|frontend|product|infra
```

Minimum behavior:

- Create MC parent task.
- Classify route via discover/company route.
- Generate BookStack/spec stub.
- Generate execution plan and subtasks.
- Apply P2P Pair Programming policy for risky tasks.
- Record evidence file.

# Phase 2 — Operator cockpit (3–7 days)

Build a simple dashboard/status surface:

- Parent goal.
- Current step.
- Assigned company/agent.
- P2P verifier status.
- Evidence paths.
- Cost so far.
- Blockers.
- Next action.

Can start as CLI/Markdown; UI can come later.

# Phase 3 — Reliability hardening (1–2 weeks)

- Standard timeout handling for local models.
- Retry/split strategy for paused agent runs.
- Stalled-task resolver improvements.
- Better evidence quality scoring.
- LightRAG health/retry and fallback rules.

# Phase 4 — External/productizable layer (6–10 weeks)

Only after internal flow is stable:

- Multi-tenant isolation.
- Auth + billing.
- Secret isolation.
- Hosted agent runners.
- Customer onboarding.
- Audit logs and compliance.

# Work packages

## WP1 — Factory CLI / workflow runner

Owner: AgentForge + CodeCraft
Goal: Implement `ai-factory.js` MVP that creates/tracks a factory workflow from one goal.

Acceptance:

- Creates MC parent task.
- Creates or links BookStack page.
- Creates subtasks for plan/build/verify/docs.
- Emits JSON evidence package.
- No production mutation by default.

## WP2 — Factory BookStack templates

Owner: Skillforge / Lexicon
Goal: Standardize pages for factory plans, architecture notes, evidence, and postflight.

Acceptance:

- Template for AI Factory request.
- Template for workflow status.
- Template for evidence package.
- Template for postflight/lessons learned.

## WP3 — P2P metrics and verifier quality

Owner: Proveo + AgentForge
Goal: Measure whether P2P verifier loops reduce rework.

Acceptance:

- Track mesh thread id, verifier end-state, cost, retry count, evidence quality.
- Report per MC task.
- Identify timeout/false-pass patterns.

# WP4 — Memory + LightRAG writeback

Owner: AgentForge / FlowForge
Goal: Make knowledge writeback reliable.

Acceptance:

- MC done writes durable summary to memory/HiveMind/LightRAG outbox.
- BookStack page is indexed or queued for indexing.
- If LightRAG is down, queue remains durable and alert is emitted.

# WP5 — Demo scenario

Owner: John + AgentForge
Goal: Create one clean demo that mirrors the video’s thesis using ALAI’s own system.

Recommended demo:

- “Build Bilko Mobile Companion architecture-first workflow” or
- “Fix H backend task with main coder + peer verifier + final QA”.

Acceptance:

- Screen-recordable flow.
- Clear before/after.
- All evidence paths exist.
- No unsupported claims.

# Risks and guardrails

1. **Do not auto-send verifier too early**
   - Keep V1 default: prewire + prompt injection + MC gate.
   - Auto-send only later as opt-in after implementation artifacts exist.
2. **Avoid cost explosion**
   - Default verifier cap: $0.25, max $1 without cost review.
   - Today’s cost check already showed non-trivial Opus spend, so V2 should use Sonnet/local models where possible.
3. **Do not treat memory as evidence**
   - Memory/LightRAG can guide retrieval.

- Evidence must remain files, commands, logs, tests, BookStack URLs, MC state, or live health checks.

4. **LightRAG must fail safely**
   - Current status check timed out on 2026-05-26.
   - Factory workflow must queue writeback when LightRAG is unavailable instead of blocking product work.

# Timeline estimate

- **Useful internal demo:** 4–8 hours.
- **Repeatable internal workflow:** 3–5 days.
- **Operator cockpit / stable internal product:** 2–3 weeks.
- **External SaaS-grade product:** 6–10 weeks minimum.

# Decision

Proceed with internal **ALAI AI Factory V2** as a tracked MC initiative.

Default implementation mode:

```
Prewire + prompt injection + MC gate + final QA
```

Not default yet:

```
Automatic Company Mesh send at dispatch time
```

# Evidence paths

- Video metadata/transcript directory: `/tmp/alai/youtube-2KcITKKJikA/`
- Transcript: `/tmp/alai/youtube-2KcITKKJikA/2KcITKKJikA.en.vtt`
- P2P V1 closure: `/tmp/alai/p2p-ai-factory-v1-closure-20260526.md`
- P2P system evidence: `/tmp/alai/p2p-pairing-system-integration-evidence-20260525.md`
- Claude Code injector evidence: `/tmp/alai/p2p-cc-userprompt-injector-evidence-20260526.md`

---

Revision #2
Created 2026-05-26 12:59:49 UTC by John
Updated 2026-06-28 20:04:50 UTC by John