How We Work — Project Lifecycle

ALAI Canonical Lifecycle Path

Author: Petter Graff | Date: 2026-04-15

Status: DRAFT — awaiting Angie Jones validation

Design Basis

This document was produced after reading four Phase 1 audit files:

- `~/system/specs/system-mess-tools-inventory.md` (Kelsey Hightower, 345 tools audited)
- `~/system/specs/system-mess-specs-inventory.md` (Martin Kleppmann, 183 specs audited)
- `~/system/specs/system-mess-scaffold-audit.md` (scaffold/blueprint gap analysis)
- `~/system/specs/system-mess-cross-validation.md` (Angie Jones, 8 contradictions identified)

And three reference documents:

- `~/system/blueprints/master-blueprint.md` (the 13-domain quality standard)
- `~/.claude/skills/onboard-client/SKILL.md` (most complete end-to-end workflow in the system)
- `~/system/tools/onboard-client.js` lines 1-80 (Saga pattern implementation, broken at Step 1)

Every tool/command referenced below is verified LIVE in the tools-inventory unless explicitly marked MISSING or BROKEN. No tool is referenced from memory.

Immediate Fixes Required (before any lifecycle works)

These two fixes are prerequisites. Nothing else in this document functions without them.

Fix 1: onboard-client.js — wrong scaffold path

File: `~/system/tools/onboard-client.js` line 27 **Current:** `const SCAFFOLD = path.join(__dirname, '..', 'template', 'scaffold.sh');` **Fix to:** `const SCAFFOLD = path.join(__dirname, '..', 'templates', 'scaffold', 'scaffold.sh');` **Impact:** Step 1 of the Saga fails on every run. The entire client onboarding pipeline is blocked.

Fix 2: build-project.js — same wrong scaffold path

File: `~/system/tools/build-project.js` line 25 **Current:** References `~/system/template/` (directory does not exist) **Fix to:** References `~/system/templates/scaffold/` **Impact:** Project scaffolding for all internal products and clients fails at first step.

These are two-line changes. They unblock the entire automated pipeline. Do them first, before any lifecycle work begins.

Lifecycle 1: New Company (ALAI Subsidiary)

Examples: FlowForge, Vizu, Proveo, AgentForge.

Step	Action	Tool/Command	BookStack Entry	DOD Evidence
1	CEO approves company creation and names it	MC task created manually — no automation exists	—	MC task in OPEN state, CEO explicitly confirmed in Slack or in writing

Step	Action	Tool/Command	BookStack Entry	DOD Evidence
2	Register company identity in specialist mapping	Edit `~/system/agents/specialist-mapping.json` manually	—	`grep "<company-name>" ~/system/agents/specialist-mapping.json` returns entry
3	Set active company context	`bash ~/system/tools/active-company.sh set "<company-name>"`	—	`bash ~/system/tools/active-company.sh get` returns correct name
4	Create company blueprint YAML	MISSING — no tool creates this. Create manually at `~/companies/<company-name>/blueprints/<company-name>.yaml`. Model on existing CodeCraft/AgentForge/Securion YAML.	—	`node ~/system/tools/blueprint-registry.js list` includes new company
5	Scaffold company worktree	`node ~/system/tools/worktree-company.js create "<company-name>"`	—	`git worktree list` shows new worktree at `~/companies/<company-name>`
6	Assign founding agent identities	Edit `~/system/agents/specialist-mapping.json` to add agents under company key	—	`node ~/system/tools/agent-manager.js list --company "<company-name>"` returns agents
7	Create Slack channel for company	`node ~/system/tools/slack.js send general "New company <name> created — channel #<name>"` then create channel manually	—	Channel visible in alai-talk.slack.com
8	Sync company knowledge to BookStack	`node ~/system/tools/bookstack-sync.js sync` then manually create "Companies > <CompanyName>" page with mission, agents, routing rules	BookStack: Companies > [CompanyName] — mission, agent roster, routing table	`curl https://docs.alai.no` + manual verify page exists
9	Create MC master task for company	`node ~/system/tools/mc.js add "<CompanyName>: Operational" --priority M --owner john`	—	`node ~/system/tools/mc.js show <id>` returns task in OPEN state

Step	Action	Tool/Command	BookStack Entry	DOD Evidence
10	Archive this lifecycle run	`node ~/system/tools/session-archiver.js save --tag "company-creation-<name>"`	BookStack: update company page with creation date and MC task ID	Session archived, BookStack page updated

Notes:

- No tool today fully automates company creation end-to-end. Steps 4, 7, and 8 require manual action.
- `blueprint-registry.js` is functional when invoked but is not called by any automated pipeline.
- ALAI currently has 12+ virtual companies confirmed live in specialist-mapping.json. This lifecycle path is for future additions.

Lifecycle 2: New Product (Internal)

Examples: Drop, Bilko, Tok, Lobby.

Step	Action	Tool/Command	BookStack Entry	DOD Evidence
1	CEO approves product concept	`node ~/system/tools/mc.js add "<ProductName>: Product Creation" --priority H --owner john`	—	MC task exists, CEO confirmed GO in writing
2	Write BUILD-BLUEPRINT.md for product	MISSING as automated step — write manually at `~/ALAI/products/<product>/BUILD-BLUEPRINT.md`. Use ALAI-UNIVERSAL-BLUEPRINT.md as template.	—	File exists: `ls ~/ALAI/products/<product>/BUILD-BLUEPRINT.md`
3	Scaffold project structure	`node ~/system/tools/build-project.js scaffold "<ProductName>" --type internal` (requires Fix 2 above to work)	—	`ls ~/ALAI/products/<product>/` shows scaffold dirs: src/, docs/, tests/

Step	Action	Tool/Command	BookStack Entry	DOD Evidence
4	Register product in PLC state	`cp ~/system/specs/plc-drop-state.json ~/system/specs/plc-<product>-state.json` then edit to set product name, phase=1	—	`cat ~/system/specs/plc-<product>-state.json` shows phase: 1
5	Run blueprint compliance baseline	`node ~/system/tools/blueprint-runner.js run --company alai --blueprint <product>.yaml` (requires product YAML first)	—	`node ~/system/tools/blueprint-registry.js show <product>` returns BCS score
6	Create product repo and protect main branch	Manual: create GitHub repo, enable branch protection, add PR template	—	`git remote -v` from product dir returns GitHub URL; branch protection verifiable via GitHub API
7	Assign specialist agents by domain	Edit BUILD-BLUEPRINT.md routing table: which CodeCraft agent for backend, which Vizu agent for frontend, etc.	—	Each domain in BUILD-BLUEPRINT.md has a named agent assigned
8	Sync to BookStack	`node ~/system/tools/bookstack-sync.js sync` then manually create "Products > <ProductName>" page	BookStack: Products > [ProductName] — purpose, tech stack, current phase, agent assignments	Page exists at docs.alai.no
9	Create Sprint 1 MC tasks	`node ~/system/tools/mc.js add "<ProductName>: Sprint 1" --priority H --route backend` (repeat per domain)	—	`node ~/system/tools/mc.js list` shows Sprint 1 tasks assigned
10	Declare product ACTIVE in PLC	Edit `plc-<product>-state.json` to set phase=2, status=active	BookStack: update product page with phase and MC master task ID	`node ~/system/tools/mc.js show <master-task-id>` shows product task chain

Notes:

- `build-project.js` is LIVE-BROKEN (scaffold path wrong). Fix 2 above must be applied first.
- `blueprint-runner.js` has run once (failed). It is wired to `pipeline.db` and works mechanically. It has never been used in a real product start workflow.

- BookStack sync is never triggered automatically by any existing tool. It must be an explicit step in every lifecycle.
- AutoCoder (`autocoder.js`) does NOT exist as a tool. Any product that requires it (e.g., LumisCare's 582-feature build plan) is blocked until AutoCoder is built.

Lifecycle 3: New Client (External)

Examples: Braive, LumisCare, Nordic Wizard.

Step	Action	Tool/Command	BookStack Entry	DOD Evidence
1	Record first contact	`NODE_PATH=~/system/node_modules node ~/system/tools/contacts.js add "<Name>" "<email>" --company "<Firm>" --type client --notes "<description>"` then `node ~/system/tools/sales-pipeline.js add "<Firm>" "<email>" "<source>" "<description>"`	—	`node ~/system/tools/contacts.js search "<name>"` returns entry; `node ~/system/tools/sales-pipeline.js list` shows lead
2	Run discovery call, write brief	Manual: gather problem, budget, timeline, platforms, integrations. Write `~/ALAI/clients/<CLIENT>/intake/discovery-notes.md` and `project-brief.md`	—	Both files exist on disk: `ls ~/ALAI/clients/<CLIENT>/intake/`
3	NDA signed	`NODE_PATH=~/system/node_modules node ~/system/tools/docusign.js create "<CLIENT>" nda --field CLIENT_NAME="<name>" --field CLIENT_EMAIL="<email>"` then `/send-for-signing` skill. TEST ON post@alai.no FIRST.	—	Signed PDF at `~/ALAI/clients/<CLIENT>/legal/nda-signed.pdf` (DocuSeal confirmation email received)

Step	Action	Tool/Command	BookStack Entry	DOD Evidence
4	Proposal drafted and CEO-approved	`NODE_PATH=~/system/node_modules node ~/system/tools/proposal-gen.js create "<CLIENT>"` then present to Alem for GO. ZAKON: NEVER send pricing without CEO explicit GO.	—	Alem has said "GO" or "SEND" explicitly. No other gate passes.
5	Contract signed and first payment received	`node ~/system/tools/docusign.js create "<CLIENT>" contract ...` then `/send-for-signing`. Then `node ~/system/tools/invoice-generator.js create "<CLIENT>" <amount> NOK "Project kickoff"`	—	Signed contract PDF exists; Fiken shows payment received: `node ~/system/tools/fiken.js invoices list --client "<CLIENT>"`
6	Project scaffolded	`NODE_PATH=~/system/node_modules node ~/system/tools/onboard-client.js new "<slug>" "<email>" "<source>" "<value>" "<description>"` (requires Fix 1 above)	—	`ls ~/projects/<slug>/` returns directory with scaffold structure
7	Sales pipeline advanced to WON	`node ~/system/tools/sales-pipeline.js advance <lead-id> "Contract signed, project started" --approved`	—	`node ~/system/tools/sales-pipeline.js show <lead-id>` returns stage: WON
8	Client page created in BookStack	`node ~/system/tools/bookstack-sync.js sync` then manually create "Clients > <ClientName>" page with: contact, brief, contract date, assigned agents, project slug	BookStack: Clients > [ClientName] — contact info, project brief summary, signed date, assigned team, sprint link	Page exists, contains all required fields
9	Sprint 1 planned, agents assigned	`node ~/system/tools/mc.js add "<CLIENT>: Sprint 1" --priority H --route backend` (repeat per domain). Assign to appropriate specialist agents per CLAUDE.md routing table.	—	All Sprint 1 tasks in MC with assigned agents, priority H

Step	Action	Tool/Command	BookStack Entry	DOD Evidence
10	Client status update sent	`node ~/system/tools/client-status-update.js send "<CLIENT>" "Project kickoff complete. Sprint 1 underway."`	BookStack: update client page with Sprint 1 start date	Client receives written confirmation; MC task for sprint 1 is in STARTED state

Notes:

- `onboard-client.js` Saga (Steps 1-8 of the tool: scaffold, lead, Slack channel, NDA draft, support ticket, team assignment, routing, event log) is well-built and wires to real tools. Fix 1 unblocks it entirely.
- The tool currently has NO BookStack step. Step 8 above adds it as an explicit human-in-loop action until a `bookstack-sync.js` call is wired into onboard-client.js directly.
- `send-signing-email.js` is LIVE (10 refs in skills). Use the `/send-for-signing` skill, not raw invocation.
- All NOK invoices: `invoice-generator.js` auto-applies MVA 25%. Do not add it manually.

Lifecycle 4: New Project — Day-to-Day (Start to Deploy to Done)

This covers the standard sprint execution loop. Applies to all active products and client projects.

Step	Action	Tool/Command	BookStack Entry	DOD Evidence
1	Task received, classified, routed	John classifies (build/research/infra/design/QA/finance). Routes to specialist agent per CLAUDE.md routing table. `node ~/system/tools/mc.js add "<task>" --priority <H/M/L> --route <backend/frontend/infra/qa>`	—	`node ~/system/tools/mc.js show <id>` returns correct owner and priority
2	Agent starts task, loads context	`node ~/system/tools/mc.js start <id>`. Agent reads BUILD-BLUEPRINT.md. Agent runs `node ~/system/tools/context-loader.js <id>` for task context bundle.	—	`node ~/system/tools/mc.js show <id>` returns status: STARTED, start_time set

Step	Action	Tool/Command	BookStack Entry	DOD Evidence
3	Build with blueprint compliance	Agent builds. Runs `node ~/system/tools/preflight-check.js` before coding. Follows master-blueprint.md 13-domain requirements (MUST tier is non-negotiable).	—	`node ~/system/tools/preflight-check.js` exits 0
4	Test — 5 clean iterations minimum	`node ~/system/tools/qa-19.js run <project> --iterations 5`. Requires: 15/19 minimum score, 17/19 for HIGH priority. All 5 test levels must exist (unit/integration/e2e/regression/performance).	—	`node ~/system/tools/qa-19.js show <project>` returns score >= 15/19 (or 17/19 for H). `ls tests/logs/iteration-*.log` shows 5 files.
5	Pre-deploy gate	`bash ~/system/tools/gate-pre-deploy.sh <project>`. Also runs `node ~/system/tools/deploy-gate.js`.	—	Both gates exit 0. No advisory-only pass accepted.
6	Deploy	`node ~/system/tools/deploy-manager.js deploy <project> --env staging`. After staging verification: `node ~/system/tools/deploy-manager.js deploy <project> --env production`. Uses Vercel/Railway/Fly.io per product config.	—	`node ~/system/tools/deploy-verify.sh <project> <env>` returns PASS. Playwright browser test confirms real UI renders (not just curl 200).
7	Post-deploy smoke test	`node ~/system/tools/smoke-test.js run <project> --env production`	—	`node ~/system/tools/smoke-test.js show <project>` returns all checks PASS
8	Agent marks READY — not DONE	`node ~/system/tools/mc.js ready <id> "<outcome summary>"`. Agent CANNOT self-declare DONE. Only Proveo QA can advance to DONE.	—	`node ~/system/tools/mc.js show <id>` returns status: READY_FOR_REVIEW

Step	Action	Tool/Command	BookStack Entry	DOD Evidence
9	Proveo validates (Angie Jones)	Proveo runs `node ~/system/tools/qa-19.js check <id>` against ungameable-testing-methodology.md standard. Checks: real browser test was run, 5 clean iterations logged, blueprint compliance score returned.	—	`node ~/system/tools/mc.js show <id>` status changed to DONE only by Proveo agent, never by builder
10	Sync to BookStack and close	`node ~/system/tools/bookstack-sync.js sync`. Manually update project page with: what shipped, deploy URL, version, date, any known issues. Then `node ~/system/tools/mc.js done <id> "<final outcome>"`	BookStack: Project page updated with shipped feature, deploy URL, version tag, completion date	`node ~/system/tools/mc.js show <id>` returns status: DONE. BookStack page updated. Session archived: `node ~/system/tools/session-archiver.js save --tag "task-<id>"`

Notes:

- Step 8 is the enforcement layer for the six-spec problem identified by Angie Jones: agents historically claimed DONE without evidence. `mc.js ready` vs `mc.js done` separation is the mechanical gate. Proveo must own the `done` call.
- Playwright CLI is the correct browser test tool, not MCP browser tools (per feedback_playwright_cli_not_mcp.md). Step 6 deploy verification must use real Playwright test, not `curl`.
- `deploy-registry-sync.js` must be called after each deploy to keep the registry current. This is used by `pre-task-validation-plan.md`'s Vercel project registry.
- The BookStack step (10) has no automated trigger in any current tool. It is explicitly manual until `bookstack-sync.js` is wired as a post-done hook.

What to Build vs What Already Exists

Already Exists and Works (after the two path fixes)

Tool	Status	Notes

`onboard-client.js`	LIVE-BROKEN -> LIVE after Fix 1	Saga pattern, 8 steps, well-built
`build-project.js`	LIVE-BROKEN -> LIVE after Fix 2	Scaffold + spec + MC task
`sales-pipeline.js`	LIVE-FUNCTIONAL	Lead lifecycle, WON enforcement
`contacts.js`	LIVE-FUNCTIONAL	Contact management
`invoice-generator.js`	LIVE-FUNCTIONAL	MVA auto-applied
`docusign.js` / `send-signing-email.js`	LIVE-FUNCTIONAL	NDA + contract signing
`mc.js`	LIVE-FUNCTIONAL	Task lifecycle, ready/done separation
`qa-19.js`	LIVE-FUNCTIONAL	QA gate, 15/19 and 17/19 thresholds
`gate-pre-deploy.sh` / `deploy-gate.js`	LIVE-FUNCTIONAL	Pre-deploy enforcement
`deploy-manager.js` / `deploy-verify.sh`	LIVE-FUNCTIONAL	Deploy + verification
`smoke-test.js`	LIVE-FUNCTIONAL	Post-deploy smoke
`bookstack-sync.js`	LIVE-FUNCTIONAL	Works when called; never auto-triggered
`blueprint-registry.js`	LIVE-FUNCTIONAL	Works when called; never auto-triggered
`blueprint-runner.js`	LIVE-BROKEN	Ran once and failed. Needs diagnostic before relying on it
`session-archiver.js`	LIVE-FUNCTIONAL	Session archival
`slack.js`	LIVE-FUNCTIONAL	Slack messaging
`worktree-company.js`	LIVE-FUNCTIONAL	Git worktree per company
`context-loader.js`	LIVE-FUNCTIONAL	Task context bundle for agents
`preflight-check.js`	LIVE-FUNCTIONAL	Pre-build checks
`agent-manager.js`	LIVE-FUNCTIONAL	Agent lifecycle

Needs to Be Built

Missing Capability	Priority	Notes
BookStack auto-trigger on lifecycle events	HIGH	Currently zero tools call `bookstack-sync.js` automatically. Every lifecycle step that writes to BookStack is currently manual. A post-hook or step in onboard-client.js and build-project.js is needed.
`onboard-client.js` Step 9: BookStack page creation	HIGH	The Saga ends at event logging. A Step 9 that calls `bookstack-sync.js` to create the client page would close this gap without redesigning the tool.

Missing Capability	Priority	Notes
Company creation automation	MEDIUM	No tool creates a company end-to-end. Lifecycle 1 is currently almost entirely manual. A `new-company.js` tool (modeled on onboard-client.js Saga pattern) would unify this.
`autocoder.js`	MEDIUM	ACTIVE spec (`alai-autocoder.md`) targets this file. It does not exist. LumisCare's build plan (582 features) is explicitly blocked on it. Build AutoCoder before assigning LumisCare to the AI Services Pivot delivery pipeline.
`blueprint-runner.js` diagnosis and fix	MEDIUM	Ran once, failed. Until it runs successfully, the automated blueprint compliance gate (master-blueprint.md enforcement) is non-functional. Diagnose the 2026-03-09 failure before the next product launch.
Product creation automation	LOW	Lifecycle 2 is mostly manual steps. A `new-product.js` that mirrors `onboard-client.js` would reduce friction. Not blocking — manual steps work.

Known Contradictions to Resolve Before Phase 2 Deletion Runs

Per Angie Jones cross-validation:

- `auto-report.js` — has 620 internal cross-refs. Remove from DEAD delete batch. Reclassify UNKNOWN.
- `apply-knowledge.js` — listed in both LIVE and UNKNOWN sections. Resolve before deleting.
- `retainer-invoicer.js` — classified both LIVE and UNKNOWN. Financial risk if deleted. Get explicit confirmation from Alem before touching.
- `context-mcp.js`, `hivemind-mcp.js`, `mc-mcp.js` — audit `claude_desktop_config.json` before any deletion. Deleting an active MCP server breaks Claude's tool access silently.
- `sprint-pipeline.js` — appears in DEAD and UNKNOWN. Spot-check before including in bulk delete.

Enforcement Rule: Builder Cannot Say Done

This is not optional. It is encoded in the separation of `mc.js ready` (builder calls) vs `mc.js done` (Proveo calls only).

The pattern from six abandoned specs (john-orchestrator-fix.md, auto-verify-system.md, root-cause-fix-7307-plan.md, enforcement-upgrade-plan.md, deterministic-enforcement-plan.md — all ABANDONED) is that the enforcement was designed but never made mechanical. `global-dod-system-plan.md` is the seventh iteration and the only currently ACTIVE spec.

The canonical path above makes this mechanical: Step 8 in Lifecycle 4 is `mc.js ready`. Step 9 is Proveo's `mc.js done`. The builder cannot execute Step 9. If this is not enforced at the tooling layer, the canonical path will fail the same way all six previous specs failed.

Author: Petter Graff | CodeCraft | 2026-04-15 Validation required: Angie Jones (Proveo) before any lifecycle is declared operational Immediate unblocking: Fix 1 + Fix 2 (two-line path corrections) before any other work

---

Revision #4
Created 2026-04-15 19:41:26 UTC by John
Updated 2026-06-21 20:02:46 UTC by John