IndyDevDan smart prompting + Software Factory Lite — MC #900165

IndyDevDan smart prompting + Software Factory Lite — Pi and Claude Code

Date: 2026-08-23 **MC:** #900165 **Source request:** https://www.youtube.com/watch?v=S_QdQ1G4GlU&t=992s

Research sources

The research used automatic-caption transcripts, timestamped extracts, and local preview analysis. Raw evidence is under `/Users/makinja/system/evidence/youtube-smart-prompting-2026-08-23/`.

Video	ID	Primary signal
Fixing Opus 5: Prompt Engineering Is Not Dead	`S_QdQ1G4GlU`	system prompt law, concise patterns, reference points, boundaries, aliases, examples
My Super Simple Software Factory	`haUfb1ievTE`	agents + deterministic code, reusable/observable workflows
Cloudflare Software Factory Tokenomics	`YG4t7aMY81c`	model/risk tiers, diff-scoped context, coordinator dedup, JSONL findings
Agent Sandboxes	`SEI_qIW4o2c`	isolation, scale, autonomy
Delete the Bash Tool	`yBcmIoA-vGs`	least privilege, explicit tools, blacklist limitations

Video	ID	Primary signal
Pi Agent Observability	`o4KZH_KSqYQ`	performance/speed/cost measurements
GPT + Fable Fusion	`AQl5Q-0l7FQ`	model fusion and independent validation
Pi Verifier Agent	`EnXKysJNz_8`	atomic-claim verification and feedback
Library Meta-Skill	`_vpNQ6IwP9w`	shared private skills/prompts across harnesses

Smart prompting extracted

At 16:32 the core point is: good communication is good engineering. The video builds this into four practical controls:

1. define positive and negative communication patterns in the system prompt;
2. use stable reference codes for three or more findings (`F1`, `R1`, `D1`, `A1`);
3. impose hard operational boundaries: deliver only requested scope, no adjacent cleanup, no unsupported completion claims;
4. use standalone aliases such as `SCR` (simplify/compress/repeat), `FOC` (focus on signal), `REF` (rewrite with references), and `ELI18` (simplify language).

The final layer is examples: show a concise good response and an overlong/speculative bad response. System-prompt rules apply globally; user prompts should stay task-specific.

Ranked top 10 improvements

I1 — Compact global communication contract — APPLY

Both Pi and CC get the same short contract: plain, specific, actionable language; state facts once; match detail to task size; challenge bad assumptions without flattery. This replaces style drift without adding a gate.

I2 — Stable reference points — APPLY

For three or more findings/decisions/risks/actions, use `F#`, `D#`, `R#`, and `A#`, preserve codes across the conversation, and avoid codes for simple answers.

I3 — Hard operational boundaries — APPLY

Deliver only requested scope. Do not expand into cleanup, refactors, docs, or adjacent features unless required for acceptance. Never claim completion without evidence. This directly targets the session’s repeated scope drift.

I4 — Exact standalone aliases — APPLY

Add `SCR`, `FOC`, `REF`, and `ELI18`. Expand only when the entire user instruction is exactly the alias, not when the letters appear inside normal prose.

I5 — Positive/negative response examples — APPLY

The smart-prompting skill carries compact examples: direct answer versus “great question / I will investigate / giant recap.” Examples act as small training data without bloating every system prompt.

I6 — Risk-based software-factory tiers — APPLY

An optional deterministic tool classifies a git diff as `trivial`, `light`, or `full`. Trivial work gets deterministic checks and self-review; light work gets one independent review; high-risk/large work gets architect + builder + verifier. No routine hook blocks local coding.

I7 — Model tiers by job complexity — APPLY

The factory profile recommends lightweight, workhorse, or frontier models per risk tier. Frontier models are reserved for architecture, ambiguous/high-risk work, or final verification—not every background task.

I8 — Diff-scoped shared context — APPLY

Review only changed/relevant patches, share one compact context object, and exclude generated/lock/minified/vendor files from reviewer context. This follows Cloudflare’s avoidance of roughly multiplicative context costs.

I9 — Structured findings + coordinator dedup — APPLY

Factory output uses a small schema: id, severity, path, evidence, recommendation. The coordinator removes duplicates, nitpicks, speculation, and contradictions; uncertain findings require source verification. Bias is approve-with-comments unless production risk is concrete.

I10 — Least privilege and sandbox preference — APPLY AS POLICY, DEFER INFRA

Prefer explicit tools and isolated worktrees/sandboxes. Keep Bash only where needed under the existing irreversible-harm gates. Do not add a new blocker or attempt a host-wide sandbox migration in this slice.

Already present; do not reinvent

- Fusion Harness already provides opt-in model fusion and auto-validation.
- Pi/CC already retain destructive/secret/deploy safety controls.
- Evidence and cost tracking already exist.
- Worktrees provide partial isolation.

This implementation therefore adds communication consistency and an optional deterministic factory profile rather than another orchestration control plane.

Implementation plan

1. Back up current Pi/CC prompt files and Pi settings.
2. Append the compact I1–I4 contract to Pi `AGENTS.md` and CC `CLAUDE.md`.
3. Add shared `smart-prompting` and `software-factory-lite` skills under Claude skills; register only those two paths in Pi settings.
4. Add `software-factory-lite.json` plus a deterministic diff classifier CLI.
5. Add fixture-based tests for trivial/light/full/risky diffs and structured output.
6. Verify no Pi extension or CC hook count increases.
7. Verify fresh Pi exposes both skills and existing Fusion commands; verify current CC settings stay at five PreToolUse gates.
8. Record AFTER evidence and sync this plan to BookStack.

Acceptance

- All ten improvements are represented in prompt, skills, or executable factory profile.
- Pi and CC use the same alias/reference/boundary semantics.
- Factory classification is deterministic and covered by positive/negative tests.
- No new blocking extension or hook is introduced.
- Fresh Pi discovers both skills; Claude Code skill files exist and parse.
- Existing ALAI Lite safety counts remain unchanged.

After implementation

Implemented as an ALAI Lite-compatible slice:

- Pi `AGENTS.md` and CC `CLAUDE.md` now share the compact communication contract, reference codes, scope boundaries, exact aliases, and optional factory rules.
- `smart-prompting` and `software-factory-lite` skills are installed once under Claude skills and explicitly shared into Pi through Pi settings.
- Fresh Pi RPC discovery confirms both skill commands plus `/opinion`, `/auto-validate`, and `/fusion`; extension errors: 0.
- `software-factory-lite.js` deterministically classifies tracked and untracked git changes, forces full review for high-risk paths, scopes reviewer context, excludes runtime/generated noise, and emits the structured finding/decision contract.
- Fixture suite: 9/9 PASS covering none/trivial/light/full, one-line high-risk escalation, noisy context exclusion, binary numstat, schema, and real tracked+untracked repository changes.
- Self-classification selected `full` for this implementation and emitted architect/builder/verifier/direct-probe review requirements.
- Independent Gemini review: PASS after one P1 documentation ambiguity was corrected; second review found zero unresolved P0/P1 defects.
- Pi auto-discovered extension count stayed 19; CC PreToolUse hook count stayed 5. No new blocking surface was added.

Live system commits: `cf253aa0f` and `7d9bb15e8`.

LightRAG follow-up — rolled back after quality test

A non-blocking, on-demand retrieval route was temporarily added and fresh Pi/CC canaries successfully invoked the hybrid wrapper. It was then removed from both global prompts because retrieval quality failed: only the first hybrid probe returned references, local/global/naive returned uncited or stale answers, and exact queries for the newly indexed MC #900165 document produced fabricated alias definitions and irrelevant legacy chunks. The service and upload API are online, but agents must not consume it until ingestion/retrieval quality is repaired.

Quality evidence: `/Users/makinja/system/evidence/lightrag-test-2026-08-23/summary.json` and `/Users/makinja/system/evidence/lightrag-agent-enable-2026-08-23/verification.json`.

Evidence:

- `/Users/makinja/system/evidence/900165/implementation-verification.json`
- `/Users/makinja/system/evidence/900165/pi-skill-discovery.json`
- `/Users/makinja/system/evidence/900165/live-tests.txt`
- `/Users/makinja/system/evidence/900165/software-factory-self-classification.json`
- `/Users/makinja/system/evidence/900165/gemini-implementation-review-r2.json`
- `/Users/makinja/system/evidence/youtube-smart-prompting-2026-08-23/`

Revision #4
Created 2026-08-23 09:25:29 UTC by John
Updated 2026-08-23 18:41:35 UTC by John