

The Centaur Manifest

A default operating mode for small, AI-augmented delivery units.

Purpose

The Centaur Manifest exists for one reason: to let a small human unit move at machine speed without losing coherence.

In the centaur era, execution is cheap. Drafting, scaffolding, refactoring, exploration, and even broad solution search can be delegated to capable agents. The limiting factors shift. The hard work becomes maintaining shared intent, bounding risk, proving correctness, integrating safely, and learning fast enough to steer.

This document is self-contained. It defines tenets and a recommended way of working that makes those tenets second nature.

Scope and relationship to organization-level frameworks

This manifest is intentionally *unit-scale*: it is optimized for 2-4 humans shipping and learning fast with AI augmentation.

In many organizations, there is also a portfolio/coordination layer whose job is to align multiple teams, allocate attention across competing outcomes, and keep a consistent north-star focus under dependencies and politics. Frameworks like **OBAF** are built for that layer: they are not an AI-augmented, two-person-team runtime. A centaur unit should read OBAF-style artifacts primarily as **upstream intent and coordination constraints** – a way to stay aligned with a broader north-star – without importing heavyweight planning/ceremony into a context where coordination cost is already negligible.

If you are only a single small unit, the Centaur kernel below is usually sufficient. If you become many units with real dependencies, add the minimal composition rules in this manifest (see **Scaling and composition**) and let the org-level layer do what it is good at: outcome portfolio coherence.

The model

A centaur unit is a **small team** (often 2-4 humans) operating with AI augmentation. The unit is defined less by headcount than by a capability mix:

- **Domain expertise** holds the truth about what matters, why it matters, who it matters to, and how to know if it changed.
- **Building expertise** holds the truth about how change is made real: systems design, integration, reliability, security, operability, and proof.

A centaur unit may be one domain expert with one builder, one domain expert with multiple builders, or a builder with deep domain adjacency. The unit remains "*centaur*" so long as intent is explicit, constraints are enforced, and evidence governs direction.

AI is treated as an accelerator of execution and exploration, not as an authority. Human judgment remains responsible for intent, constraints, and proof.

The tenets

1) Outcomes over outputs

Outputs are deliverables; outcomes are observable changes in the world.

2) Evidence is the arbiter

Plans are hypotheses. Confidence yields to signals, observation, and reproducible checks. Signals are *learning instruments*, not performance targets. When a signal becomes a target, it becomes easier to game and less useful as evidence.

3) Constraints are first-class

Security, reliability, cost, compliance, operability, and interfaces/contracts (boundaries over heroics) are part of the outcome, not deferred work.

4) Verification dominates execution

When generation is cheap, proof becomes the bottleneck. Design work so correctness is continuously demonstrable.

5) Automation-first governance

Policies live in pipelines and checks, not in meetings and approvals. Automation must remain *enabling*, not controlling. Automate constraints and proof, not politics.

6) Micro-iterations, small batches

Short cycles reduce error, maximize learning rate, and limit blast radius.

7) One source of truth for intent

A canonical intent artifact defines why, what success means, and what must not break.

8) Coordination is artifact-driven

Async by default. Sync time is reserved for irreducible human alignment. Ownership follows capability and accountability, not titles.

9) Maintain conceptual integrity under speed

Protect shared mental models. Prevent drift as change velocity rises.

10) Capture learning while cheap

Reflection is mandatory, lightweight, and tied to events. Surprises must improve future work.

Recommended implementation

Readiness check (quick diagnostic)

A centaur unit can operate with very little ceremony, but it cannot operate without minimum cultural and technical preconditions. If you answer "no" to many of these, fix the foundation before you blame the method.

- [] **Intent clarity:** Can we state the outcome in one sentence and agree on what would falsify it?
- [] **Evidence access:** Do we have at least one trustworthy feedback loop (telemetry, user contact, ops signals) that updates on a useful cadence?
- [] **Constraint authority:** Do we have the right to define/enforce the constraints that matter (security, reliability, cost ceilings, etc.)?
- [] **Autonomy:** Can the unit change course based on evidence without escalation for every pivot?
- [] **Integration path:** Do we have CI/CD or an equivalent integration mechanism where guardrails can run continuously?
- [] **Definition of done:** Do we treat "done" as *integrated with proof*, not merely implemented?
- [] **Psychological safety:** Can we run blameless micro-reviews where people can speak without punishment?
- [] **Decision latency:** Can we make small decisions quickly without waiting on large meetings?
- [] **Dependency clarity:** If we depend on others, do we have explicit contracts/interfaces and a way to test them?
- [] **Signal hygiene:** Do we review metrics/signals for gaming, vanity, and drift (at least monthly)?

The Centaur kernel

The recommended implementation is an operating system, not a bureaucracy. It minimizes ceremony and maximizes explicit state, because state is what prevents small units (and their agents) from diverging.

The kernel consists of five artifacts and three loops. The artifacts hold intent, hypotheses, verification, flow, and learning. The loops keep the unit aligned with reality.

The artifacts

The centaur unit relies on a small set of artifacts that are always current. They are designed to be read in minutes and updated continuously.

The Intent Card is the single source of truth for "what we are doing and why." It contains the end-state, the purpose, the non-negotiable constraints, and the signals that will tell you whether the outcome is happening. The intent card is not a requirements document. It is a contract with reality: if the signals do not move, the unit changes course.

The Micro-Experiment Note is the unit of progress. It captures one hypothesis and one move. It describes what you believe will happen, how you will know, what would falsify the belief quickly, and what the next action is. It is intentionally small, because large hypotheses are expensive to disprove and invite thrash.

The Guardrail Pack is executable governance. It is the set of tests, checks, policies, and observability expectations that must pass for work to be considered integrated. It is versioned, explicit, and treated as a living system. The guardrail pack is how the unit keeps speed from turning into fragility.

The Flow Board is a minimal representation of work-in-progress. It exists to keep WIP low and to make blocking visible. The board is not a planning tool; it is a flow stabilizer.

The Decision & Learning Log captures the small number of decisions that matter and the lessons that changed the system. It is the memory of the unit. Its purpose is not documentation; it is preventing the unit from relitigating past choices and repeating avoidable failures.

The loops

The unit runs three loops concurrently. The loops are short by design, because long loops are incompatible with centaur iteration velocity.

Loop A: Continuous execution is the default mode. Work is pulled in small slices, executed quickly, and integrated through the guardrail pack. Execution is allowed to be creative. Integration is not. When integration becomes uncertain, the unit treats it as a signal that verification or constraints are underspecified.

Loop B: Daily evidence alignment is a brief, disciplined check-in. The point is not status. The point is to look at signals and decide whether the current hypothesis still deserves belief. This loop is where the humans ensure they still share the same intent and theory of value. If they do not, they update the intent card, not each other.

Loop C: Weekly governance and learning is where the unit improves its own operating system. This is when guardrails are tightened, recurring failure modes are eliminated, and the unit reduces future verification cost by investing in automation. This loop is also where conceptual integrity is protected: if the system is becoming incoherent, this is where boundaries are clarified and contracts are made explicit.

How work actually moves

A centaur unit begins by creating an intent card. The title should be a full intent statement, not a slogan, because intent is easiest to lose when compressed into marketing language. The body of the card makes constraints explicit and ties the work to observable signals. If the unit cannot name a plausible signal, it does not yet understand the outcome.

From the intent card, the unit selects one micro-experiment. This is the smallest move that could produce evidence, reduce uncertainty, or strengthen the verification harness. The micro-experiment is written so it can be falsified. It should be obvious what would make the unit stop and change course. If nothing could falsify the experiment, it is not an experiment; it is a hope.

Execution proceeds as a pull system with low WIP. The unit allows itself to generate quickly, but it does not allow itself to merge quickly without proof. "Done" means integrated through guardrails, not merely implemented. Outcome validation is separate: it is measured through signals, not asserted.

When an event occurs that changes understanding, the unit performs a micro-review immediately. The review is short, factual, and aimed at system improvement. It answers what was expected, what happened, why it likely happened, and what changes now.

Micro-reviews are **blameless** by design:

- no naming/shaming (capture lessons without attribution),
- no interruptions (let each person finish a thought),
- separate facts from explanations,
- always produce a concrete system change (guardrail, constraint, signal, or next experiment). If the review does not change the system, it is theatre.

Default accountabilities

The Centaur Manifest avoids role prescriptions, but small units benefit from explicit default accountabilities. These defaults reduce drift and ensure the two core truths—domain truth and systems truth—remain continuously covered.

Domain accountability focuses on what matters and how to know:

- expresses intent in domain language and stakes,
- defines signals of success and failure,
- gathers evidence from users, stakeholders, or operations,
- clarifies constraints that come from reality rather than preference.

Builder accountability focuses on what must hold and how to prove it:

- maintains architecture and boundaries/contracts,
- builds verification harnesses (tests, checks, observability),
- protects operational fitness (deploy/rollback, failure modes, SLAs),
- manages integration risk under rapid change.

In many units, one person leans domain and another leans builder. In others, the builder set spans multiple people. The important property is not the split; it is that both sets of accountabilities remain owned.

Operating discipline for AI-augmented work

Centaur work assumes a particular posture toward AI: delegate execution aggressively, but express the work in terms that keep the unit aligned and safe.

In practice, the unit communicates intent to agents using four elements: the outcome, the constraints, the acceptance criteria, and the proof obligations. The unit asks agents to propose and implement, but it requires agents to justify changes with evidence and to produce artifacts that make verification cheaper, not harder. Any non-trivial change is incomplete until it includes its verification story.

When an agent is uncertain, the correct behavior is not to guess more confidently. The correct behavior is to narrow the hypothesis, reduce scope, and seek evidence sooner.

Scaling and composition (when there is more than one unit)

The centaur kernel works best when the unit is small and dependencies are minimal. When you have multiple units, the failure mode shifts from *local thrash* to *portfolio incoherence*.

Use these minimal composition rules:

1) North-star intent is upstream and non-prescriptive.

- If the org uses an outcome framework (e.g., OBAF), treat its north-star artifacts as upstream intent.
- Your unit does not "run" the org framework internally; instead, you **map** your Intent Card to it.

2) Supporting intents are explicit.

- Every unit-level Intent Card must name which upstream intent it supports.
- If it supports none, it is a local optimization and should be treated as suspect.

3) Dependencies are contracts, not conversations.

- For each external dependency, define a contract: interfaces, compatibility expectations, ownership, and failure modes.
- Put contract checks into the Guardrail Pack (integration tests, schema/API compatibility, version gates).

4) Decision rights are written down.

- When two units touch the same boundary, clarify ownership and escalation paths in the Decision & Learning Log.

5) Cross-unit sync is a last resort.

- Default to artifact-driven coordination: upstream intent + dependency contract + guardrails.
- Sync time exists to resolve ambiguous intent and irreducible trade-offs, not to do status.

Templates (minimal, readable, and durable)

Intent Card (canonical state)

Use a format that can be read in under two minutes.

Title: a full intent statement (end-state plus purpose)

Upstream intent: what broader outcome/north-star this supports (if any)

Context: the situation as it is, and why it matters now

End-state: the observable change you expect

Purpose: why the change matters (the "so that...")

Constraints: what must not happen; what must hold; boundaries/contracts

Signals: what you will observe if the outcome is happening (avoid vanity)

Quality: what "good" means under real operating conditions

Course of Action (CoA) (*short planning narrative; not a detailed plan*):

- Initially: the first move(s) to create evidence or reduce risk
- Thereafter: the likely follow-on moves if the hypothesis holds
- Finally: how the effort concludes, transitions, or is declared done

Decision points: what evidence causes pivot / persevere / stop

Owner: accountable humans

Dependencies: external contracts that must hold

Timeframe: when meaningful signals should appear

Micro-Experiment Note (one hypothesis, one move)

Hypothesis: what you believe will happen and why

Criteria: what counts as success or failure

Signals: what you will observe (prefer leading indicators + qualitative context)

Falsifier: what would change your mind quickly

Next action: the one move you will do now

Micro-Review (event-triggered learning)

Expectation: what was supposed to happen

Observation: what actually happened

Explanation: why it likely happened (one level deep)

Change: what updates now (guardrail, constraint, signal, next experiment)

Anti-patterns and defenses

Centaur work fails in predictable ways, usually by turning evidence and constraints into performative rituals.

1) Goodhart's Law (signals become targets)

Symptom: teams optimize the metric, not the outcome.

Defenses:

- treat signals as *triangulation*, not a single number;
- pair quantitative metrics with qualitative evidence (user/ops narratives);
- rotate/refresh metrics when behavior starts adapting to the measure;
- make the falsifier explicit on the Intent Card.

2) Vanity metrics (measurement theater)

Symptom: dashboards look good while reality does not change.

Defenses:

- prefer behavior change, system performance, and cost-of-failure signals;
- require each metric to answer: "What decision would this change?"
- include a "kill metric"—a signal that, if it moves the wrong way, forces a stop.

3) Constraint creep (boundaries silently become dogma)

Symptom: "non-negotiables" expand until exploration dies.

Defenses:

- review constraints regularly (monthly/quarterly) for relevance and evidence;
- separate *hard constraints* (law, safety) from *preferences* (habits, legacy);
- encode constraints as tests/checks so they stay explicit and reviewable.

4) Automation-first governance becomes control

Symptom: pipelines encode politics; people stop taking ownership; work slows under the pretense of safety.

Defenses:

- automate proof obligations and boundary checks, not approvals;
- minimize discretionary gates; make exceptions observable and cheap;
- treat every new gate as a hypothesis with a measurable risk it reduces.

5) Micro-experiment random walk

Symptom: constant motion, low cumulative learning, no coherent narrative.

Defenses:

- keep the Course of Action sketch on the Intent Card and update it when evidence changes;
- force explicit decision points (pivot/persevere/stop);
- cap concurrent hypotheses (WIP on learning, not only on work).

6) AI delegation without proof

Symptom: rapid change accrues hidden risk; integration becomes fragile.

Defenses:

- no non-trivial merge without guardrails;
- require agents to produce tests/observability alongside changes;
- treat missing proof as incomplete work, not a future task.

Summary

The Centaur Manifest is a compact doctrine for small units operating at AI speed. It is not a method for writing tickets or running ceremonies. It is a way to keep a fast unit coherent: *outcomes over outputs, evidence over confidence, constraints over optimism, automation over bureaucracy, and learning over repetition.*

Source: <https://pkt.systems/CENTAUR.md>

PDF: <https://pkt.systems/CENTAUR.pdf>