AN AI-LED ENGINEERING PLATFORM

Crucible

Engineering documentation that builds itself.

From a plain-language idea to a complete, traceable V-model package —
the documentation keeps itself in lockstep with the design.

It started with one hard, slow job.

Automate the engineering grind — then keep the paperwork honest.

The spark was concrete: describe a requirement in plain language, have an agent draft the design work, check it against spec, and iterate. That capability is still the technical core of Crucible — but the real product is the wrapper that turns it into a disciplined process an engineering firm will actually trust and buy.

CORE

The design agent

Drafts the engineering work — circuits, firmware, mechanical, software — and checks it against the spec. This is the engine engineers recognize as doing real work.

WRAPPER

V-model orchestration

Concept → requirements → architecture → build → verification. Phase gates and an approval flow turn the agent from a tool into a process.

MOAT

Living documentation

Traceability, verification records, an audit-ready package — produced as a side effect of the work and kept in lockstep with the design. This is what makes a firm sign.

The design work is what gets the engineer excited. The documentation is what gets the firm to buy.

The engineer's documentation tax.

40–60%

of an engineer's time at small firms doing regulated or contract work goes to requirements, verification, and traceability paperwork — not to the design itself.

Where the time goes:

Requirements	scattered documents, hand-numbered, no traceability
Verification plans	written after the build instead of before
Traceability	maintained by hand in spreadsheets
Audit / review packages	compiled manually, weeks before each gate review
Change impact	ripples get missed; rework and respins cost dearly

Small teams feel it worst — they win the contracts that demand rigor, but lack the headcount and tool budget for heavyweight ALM suites.

AI is now genuinely good at the process. The deep domain judgment stays human.

Frontier models can produce long-form, structured, internally consistent engineering artifacts — which is exactly what the V-model is made of. What they can't do is the discretion-heavy judgment that only an experienced engineer has. Crucible is built around that line, not against it.

AI HANDLES WELL

- Concept dialogue & requirements drafting
- Verification & validation planning
- Architecture decomposition into blocks
- Behavioral / first-pass design artifacts
- Self-checking artifacts against the spec
- Documentation & traceability upkeep

THE ENGINEER STILL OWNS

- Discretion-heavy component & material choices
- Behavior the models can't see (parasitics, edge cases)
- Risk, safety, and mitigation strategy
- Domain-specific design craft
- Vendor quirks & supply-chain reality
- Final judgment and sign-off

The collaboration is the product. Not an autopilot — a copilot that knows its own limits.

Eight gates, one V-model.

Design flows down the left arm; each level is proved by the matching activity on the right. Build sits at the turn. Throughout, Crucible constantly compares every document against the others, keeping the whole set coherent.

8 Lessons Learned
Capture what to carry forward.

1 Concept
What are we building, and why?

validates

7 Validation
Solves the real need? Confirm vs. the concept.

2 Requirements
What must it do? Testable, numbered.

verifies

6 Verification
Meets the requirements? Run the V&V plan.

3 Architecture
How is it built? Blocks, interfaces, allocation.

accepts

5 Acceptance
Built to spec? Conformance to the handoff.

4 Build
Make it. The handoff package is the build contract.

Manual *oracle*

"Can we use it?" — operational coherence.

Pitch *oracle*

"Can we sell it?" — commercial coherence.

Cross-cutting checks — run against the whole V, continuously.

Freezes are commitments, not certainties.

A freeze isn't a claim that an artifact is now correct — it's a commitment to consequences: long-lead parts ordered, tooling started, milestones signed. Crucible recognizes three freeze states, tied to the cost of changing them.

SOFT FREEZE

Working reference

Locked as the agreed working draft; no external cost committed yet. *Iteration is cheap — edit freely, coherence propagates, downstream re-review banners appear.*

HARD FREEZE

Real money committed

Locked because something with quantifiable cost is committed — parts, contracts, tooling. *Changes carry measurable cost; the agent surfaces schedule and rework impact before proceeding.*

SEALED

Shipped or submitted

Locked because changing it would invalidate shipped product or a regulatory submission. *Formal change request, full re-verification of dependents, complete impact analysis.*

The freeze model mirrors real engineering cost — the further you've committed, the more deliberate a change must be.

Backtracking is not a failure.

Returning to a prior phase when new information makes an earlier commitment wrong is the correct engineering response — not a process violation. Crucible treats backtracking as a first-class operation and makes its cost transparent across three forces.

PRODUCT

Does it sharpen or muddy?

What does the change do to quality, capability, or user value? Does it close a gap or open one?

SCHEDULE

What moves?

What does it do to delivery dates? Which downstream dependencies shift? How much work gets scrapped?

SCOPE

Creep, contract, or rebalance?

Does the boundary of what's being built change? Are stakeholder commitments affected? Is this a quiet expansion that needs naming?

The owner balances the three forces and makes the call. The agent surfaces the trade-offs honestly — that judgment belongs to the human.

Several users, one workspace.

Crucible meets several different people where their strengths are — and where their gaps are. They all end up with the same complete package; none would produce it alone. In a small team, one person wears every hat.

PERSONA I

Product Owner / PM

Strong at coordination:

- Driving timeline and phase gates
- Stakeholder approvals across functions
- Scope discipline, sign-off rigor

Weak at engineering depth:

Needs the tool to keep specs and customer-facing copy honest without becoming an engineer.

PERSONA II

Systems Engineer

Strong at the top of the V:

- Stakeholder dialogue, requirements rigor
- Architecture, decomposition, interfaces
- Verification planning, traceability

Weak below the line:

Rusty on deep discipline work; usually hands detailed design to a track-specific engineer.

PERSONA III

Discipline Engineer

Strong at the bottom of the V:

- Detailed design — EE, firmware, mechanical, software
- Domain craft, edge-case behavior
- Vendor knowledge, build reality

Weak above the line:

Skips traceability when deadlines press; documentation gets done last and rushed; loses track of why.

In a one-person shop, the same person wears every hat. In a larger firm, the roles hand off via Crucible. Same package, scaled by the team.

An AI systems engineer for any discipline.

Crucible takes a spec — or just an idea — interviews you to lock the scope, runs the V-model front end with you in the loop, and hands off detailed design with full constraint envelopes. The documentation stays in lockstep with the work the whole way.

I

Front-end guidance

Concept → requirements → V&V plan → architecture, each checked against the spec before detailed design begins.

II

Honest collaboration

When confidence drops, the agent surfaces the constraint envelope and lets you choose. No invented part numbers, no false authority.

III

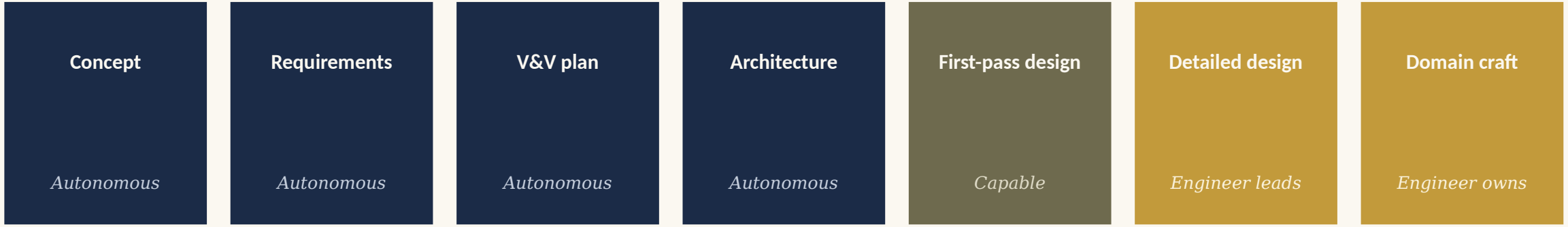
Living documentation

Requirements, V&V, architecture, and traceability are one coherent package — auto-updated on every change.

It doesn't replace the engineer. It does the structured grind so the engineer does the engineering.

Autonomy varies across the lifecycle.

The agent isn't equally capable at every phase. It declares its confidence at each step and hands you the wheel at the right moments — explicitly, not silently.



The agent asks for the wheel at the right moments. Discretion-heavy work stays the engineer's, by design.

Idea to validated system, gate by gate.

1	*Drop in the idea*	“A 100 W servo drive, 24 V supply, CAN bus, within the relevant safety envelope.”
2	*The agent interviews*	You answer ~12 questions — operating range, tolerances, accuracy targets, interfaces, qualification expectations.
3	*Requirements drafted*	Two dozen requirements across categories, each with rationale. You edit a few, approve the rest. Gate locked.
4	*V&V plan drafted*	Methods, test definitions, pass/fail criteria — written before the design, not after. You approve with one change.
5	*Architecture proposed*	Block decomposition with interface contracts and first-pass design artifacts per track. You approve.
6	*Build*	Take the handoff package and build it — yourself, or hand it to a contractor. Log the build back into Crucible.
7	*Verify*	Run the verification plan against the build — did we build it right? Record results against each requirement.
8	*Validate*	Exercise it against the concept with real users and conditions — did we build the right thing? Then release.

Who does what — explicit, not implied.

AGENT

Process, drafting & memory

- Orchestrates the phase state machine
- Drafts requirements with traceable IDs
- Writes the V&V plan before the design
- Decomposes into blocks + interface contracts
- Produces first-pass track artifacts
- Maintains traceability on every change
- Surfaces its confidence at every step

ENGINEER

Judgment & sign-off

- Approves at every phase gate
- Edits where the agent misread intent
- Makes discretion-heavy design choices
- Owns domain craft and edge-case behavior
- Owns risk, safety, and mitigation
- Calls vendor and qualification decisions
- Overrides agent recommendations freely

The agent declares confidence at every step. The engineer accepts, edits, or overrides — and the package stays coherent either way.

One coherent package, not a pile of templates.

Everything is generated as you go and exportable as a bundle — the documents stay consistent with each other and with the design, because they're produced from the same process.

Concept of Operations

Mission, stakeholders, scenarios, boundary.

Requirements & V&V plans

Numbered, testable, each paired to a verification and validation item.

Architecture

Block decomposition, interface contracts, allocation matrix.

Track-specific artifacts

Buildable starting points — SPICE netlists, firmware, CAD parameter tables, schemas.

Handoff package

The build contract, with full traceability concept → requirement → test.

Execution records

Build, acceptance, verification, and validation results against the plan.

The documentation is the moat.

Plenty of tools can draft text. What's hard — and valuable — is keeping a whole engineering record coherent and traceable as the design changes. That's the discipline Crucible enforces, and it's the part a firm can't get from a general chatbot.

COHERENT

Always in sync

Change one thing and everything downstream that depended on it reopens for review. The record never silently drifts from the design.

TRACEABLE

Nothing untethered

Every requirement ties to the concept and to a test. When a customer or auditor asks “why is this here?”, the thread is there.

DISCIPLINED

Baselines that mean something

Gates freeze. Changes are deliberate, logged events — not quiet overwrites. The audit trail is a side effect of working.

A general chatbot produces words. Crucible produces a record a firm can stake a contract on.

CLOSING

AI doesn't replace the engineer.

It puts the engineer back on the things only an engineer can do — and turns the documentation from a tax into a byproduct.

Try Crucible Now