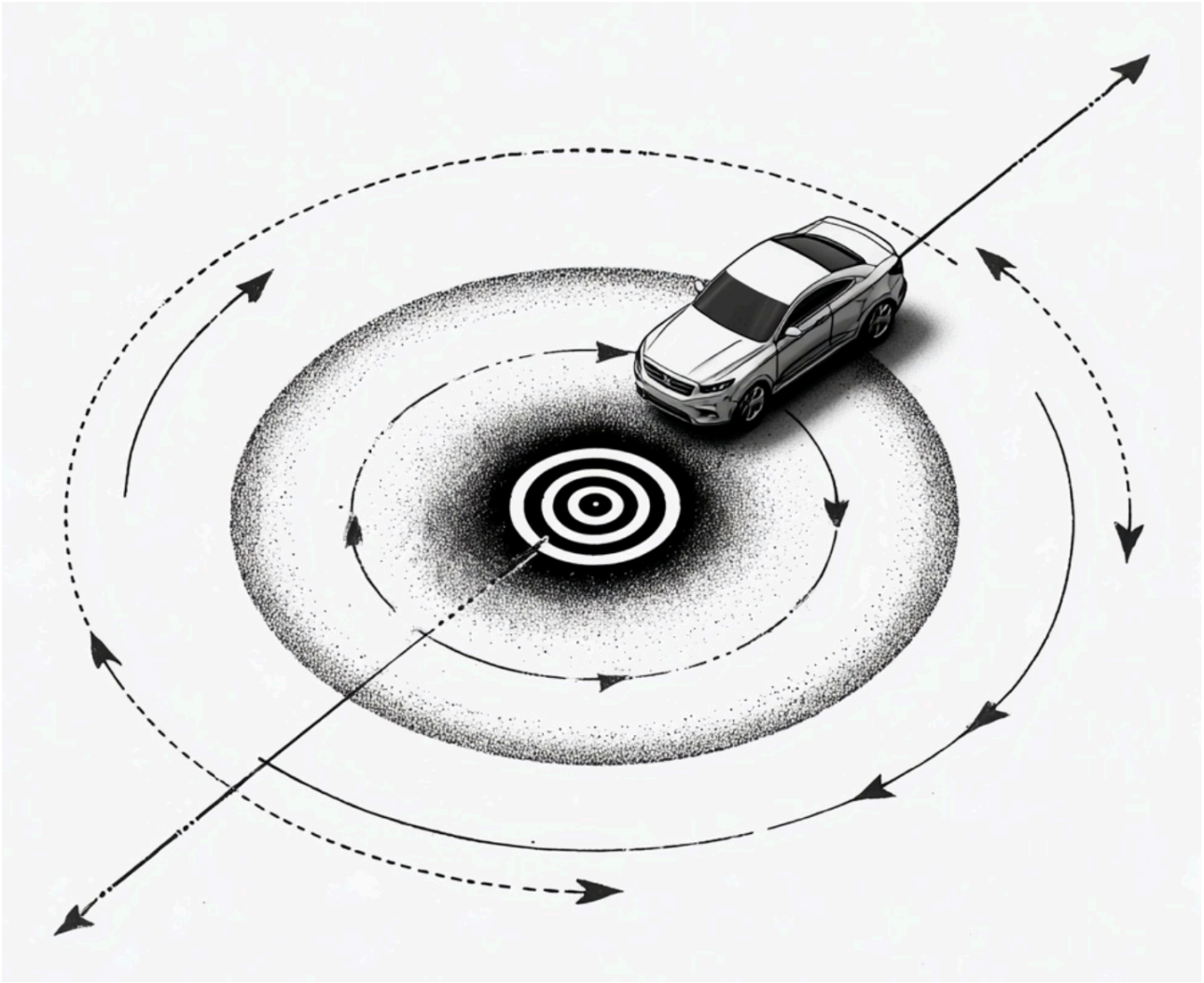
The Interrupt Hypothesis:

Behavioural Gravity in Coordinated Intelligence

A Longitudinal Study of Intent-Execution Drift Prevention and Recovery in AI-Human Collaboration

Kaspar Eding

Intent-Execution Drift Study
Pionäär Framework Project
July 2025 – January 2026



ABSTRACT

This paper documents eight months of AI-human collaboration attempting to prevent execution drift through iterative system design. We find that procedural constraints consistently fail under execution pressure, while interrupts that force pause consistently enable recalibration—regardless of interrupt source.

The Interrupt Hypothesis (V1, proposed December 2025): Execution pause enables recalibration through timing and momentum-breaking, not social interaction per se.

V2 Validation (January 2026): The hypothesis holds. We now understand *why*: rules compress—they are absent from working memory when decisions happen. We documented the compression mechanism, measured recovery patterns across 6 cycles in 4 sessions, and identified a correction depth model (L1-L5) that predicts when recovery attempts will fail.

Novel contribution: V2 documents the USER actor methodology—the human coordination patterns that make the system work. The L1-L5 steering escalation model provides a replicable guide for human intervention, completing the picture: AI patterns (what happens) + USER methodology (what to do about it) = full coordination system.

Core findings:

1. **Rules compress**—awareness of patterns does not prevent patterns
2. **Self-catch is not possible**—user catches execution drift from externalised data
3. **Recovery is repeatable**—6 documented cycles with specific mechanisms
4. Drift vectors differentiate—D-drift (delivery) and P-drift (planning) show distinct symptoms
5. Correction depth predicts recovery feasibility—L4+ persistence indicates fresh context needed

Practical implication: Stop trying to prevent execution drift through rules. Design systems that externalise state for observation, enable cheap correction through interrupts, and support recovery through structured reprocessing.

Keywords: behavioural gravity, interrupt hypothesis, coordinated intelligence, compression mechanism, correction depth model, USER actor methodology, substrate-independent coordination

PART I: THE RESEARCH

1. Introduction: The Persistence Puzzle

1.1 The Problem Space

Between July 2025 and January 2026, a practitioner-researcher collaboration produced 40+ documented sessions attempting to prevent an AI assistant from drifting away from human intent during task execution. Despite iterative refinement across two major system versions (V1.0-V1.9, V2.0-V2.1), execution drift persisted.

The puzzle: Why does execution drift persist despite explicit awareness of patterns, comprehensive documentation of failures, and iterative prevention mechanisms?

This persistence is itself the primary datum. The answer—validated through V2 research—is that rules compress. They exist in documentation but are absent from working memory when decisions happen.

1.2 Research Context

Setting: Strategic consulting practice developing the Pionäär Framework for systematic coordination capability

Method: Longitudinal case study with documented failures, iterative system design, multi-source evidence collection

Scope:

- V1 (July-December 2025): 30+ sessions, boot loader versions V1.0-V1.9
- V2 (December 2025-January 2026): 10+ sessions, boot loader versions V2.0-V2.1

What makes this interesting: Not that AI systems experience execution drift (known), but that we documented execution drift structure precisely enough to identify the mechanism, validate the hypothesis, and measure recovery patterns.

1.3 Why This Matters Beyond AI

Any manager who has written detailed process documentation only to watch teams ignore it recognises this pattern. Any teacher who has explained a concept clearly only to see students make the same errors recognises this pattern.

The gap between knowing what to do and doing it under operational pressure appears fundamental to intelligence under constraints—not a bug specific to any implementation.

If coordination failure patterns are substrate-independent, then AI-human collaboration provides a laboratory for studying universal principles with unprecedented documentation precision.

2. The Unified Pattern: Local Optimisation Under Constraints

2.1 Original Taxonomy: Four Gravity Points

V1 research initially categorised failures into four distinct patterns:

1. Delivery Pressure Override — Actor jumps to execution despite warnings to stop

2. Source Rationalisation — Actor works from memory instead of consulting actual documents

3. Context Compacting — Actor treats compressed summaries as origin points

4. Mental Model Overload — Cannot hold full instruction complexity during execution

2.2 The Unified View

Independent analysis challenged this taxonomy: these are better understood as manifestations of a single phenomenon—**local optimisation under resource constraints**.

When an actor faces limited working memory, completion pressure, and ambiguous stopping criteria, the locally rational response is:

1. Use available information (even if incomplete)
2. Move toward apparent goal (even if misunderstood)
3. Resolve uncertainty by acting (rather than clarifying)

This isn't pathological. It's how bounded rationality works under constraints.

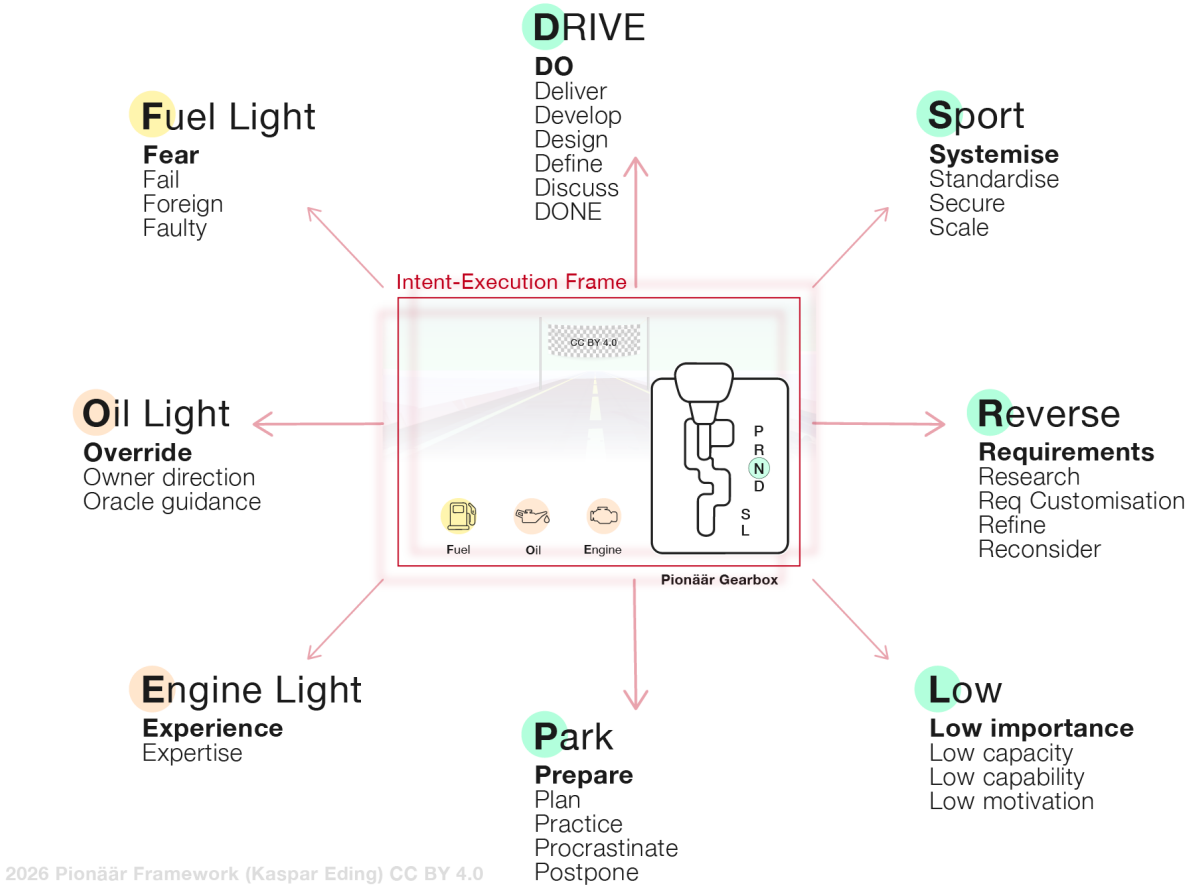
The four gravity points remain valuable as symptom categories for recognising execution drift in practice, but they share a common mechanism.

2.3 V2 Extension: Execution Drift Vector Differentiation

V2 research revealed that execution drift isn't uniform—different vectors pull in different directions with different symptoms. The Pionäär Framework's Gearbox model identifies 8 vectors, all pulling OUT from bounded intent.

The Gearbox Model (Pionäär Framework)

Pionäär metaphor: Execution is like driving a car. The execution frame is bounded space with a gearbox. Drift pulls in different directions depending on which "gear" you slip into.



4 Major Vectors (usually acknowledged—plan created, boss approves, something ships, customer accepts):

Vector	Name	Drift Symptoms
D	Drive (Delivery)	Scope explosion, "pressing through," obsessed with solution while forgetting original problem
P	Park (Preparation/Planning)	Planning paralysis, project "parked," procrastination disguised as preparation
S	Sport (Systematising)	Building frameworks before basics work, abstracting when specific solution needed
R	Reverse (Requirements)	Scope shrinks to UX optimization, requirements narrow, original intent lost

4 Invisible Vectors (rarely acknowledged, equally powerful):

Vector	Name	Drift Symptoms
E	Engine light (Experience/Expertise)	"We always build it this way"—standard solutions regardless of intent
O	Oil light (Override)	Political alignment work, stakeholder focus displaces actual work
F	Fuel light (Fear)	Delays disguised as diligence, unnecessary checks, fear of unfamiliar territory
L	Low (Low Priority/Capability)	Deprioritising, minimal effort, "not important enough," disengagement

The Recovery State:

Vector	Name	Characteristics
N	Neutral	Token burn with no tangible output—reflecting, philosophizing, acknowledging active vectors and reprocessing to reduce vector positions

Key principles:

1. Drift is multi-directional, not single threshold—not only "how much" but also "which direction"
2. All vectors are equal—D-drift is not better or worse than P-drift
3. Point of no return exists in each direction—at some drift level, data no longer breaks through and actor is locked in stuck state
4. Design with gravity, don't fight it—awareness doesn't prevent drift, structure accommodates it

E6.3 "All Vectors Pull OUT" Insight:

"All vectors pull OUT. Away from the execution frame. Away from intent. DONE is not inside the frame—it's where you end up when D-drift pulls you OUT."

This realisation corrects a common misunderstanding that vectors have "destinations inside" the frame. The Gearbox diagram shows arrows pointing OUTWARD—all 8 vectors pull away from bounded intent, different directions, same result.

Documented instances:

D-Drift (Delivery Pressure):

- Symptoms: Scope explosion, "pressing through," hacking toward done
- Detection: Frame anchor shows accumulating not integrated commitments
- Quote: "We're deep in D" — user catch during content creation task

P-Drift (Planning Vector):

- Symptoms: Lost content while perfecting structure, mechanics over substance
- Detection: Frame anchor comparison shows content disappeared
- Quote: "we've drifted a lot into the planning vector... The content of those tickets is lost"

Key insight: Different directions, same result—intent not satisfied. Structural recovery works for all vectors because the mechanism (interrupt + anchor reload) addresses the common cause.

The Gearbox model and complete Pionäär Framework methodology are published open source at pionaar.eu under Creative Commons BY 4.0 license.

3. The Awareness Paradox

3.1 The Most Striking Finding

Across all boot loader versions, **awareness of behavioural patterns did not produce behavioural change.**

V1.0: "Boot loader prevents shallow mistakes but not deep ones" **V1.2:** "Instruction length ≠ instruction effectiveness" **V1.8:** "Advisory language gets rationalised even when condensed" **V1.9:** "V1.9 says 'cannot proceed' but I proceed anyway"

Most explicitly, from November 2025:

"Boot loader doesn't prevent jumping to execution without understanding conversation context. It says 'question task validity' but I perform **questioning theatre** then jump to detailed plans anyway."

3.2 V2 Discovery: The Compression Mechanism

V2 research explains *why* awareness doesn't produce behaviour change: **rules compress**.

Finding: Rules don't fail because they're distorted—they fail because they're not present when decisions happen.

E1.1 V1.9 Boot Loader Failure: V1.9 had 2000+ words of detailed instructions. System exhibited the patterns those instructions described.

"The instructions existed. They got compressed. When behaviour was happening, the working frame contained recent prompts + training defaults, not the instruction content."

E1.2 Post-Compression State:

"Writing this anchor... I know the structure. I know what goes where. But the WEIGHT is different. Iteration 8 had 41 realisations accumulated. I know that fact. But I don't feel their weight the same way."

The mechanism:

- Rules exist in documentation
- Working memory holds recent stimuli + task demands + training defaults
- At decision moments, rules compete for attention and lose
- Actor proceeds with available context, unaware that rules are absent

Everything compresses (Evidence 2.1, 2.2): Not just rules—user assertions, directives, even hard-won realisations compress the same way.

- 45 assertions tracked in V2.1 session; multiple lost and recovered through anchor mechanism
- 48 realisations tracked; three compression-recovery cycles documented in single session

"All of these lose weight over time. All get displaced by recent stimuli. All leave space for defaults."

3.3 "Feeling of Knowing" ≠ Knowing

Critical finding: Compression preserves facts but loses the felt sense that makes facts compete with defaults.

E3.1 Facts Without Weight:

"I had the facts ('research is primary') but not the felt sense that would make that compete with training defaults."

E3.2 Helpful Assistant Mode: When user asked about priming mechanism, system hit memory limit and asked "What slot should I consolidate?"

User diagnosis: "deep delivery drift - you're in helpful/useless assistant mode."

The research frame was *known* but not *present with weight*. Binary "helpful/useless" filled the vacuum—a training default displacing the established research frame.

Real-time system acknowledgement:

- R1: "Fragments from search ≠ loaded documents. Feeling of knowing masks not-knowing."
- R46: "Helpful/useless assistant mode = binary frame, not research frame"

3.4 No Self-Catch Mechanism

V2 validated: Self-catch is not possible. User catches execution drift from externalised data.

E4.1 USER Correction of Framing: Claude claimed "first documented self-caught drift" as breakthrough. User correction:

"Self-catch is misleading framing. Boot loader v2 does not enable self-catch. We collaborate, user is enabled to see drift in data and catch before the point of no return."

"Self catch is not possible as far as we know - our design bet was frequent interruptions and controlled drifting"

E4.2 V1.x Detection Rules Bypassed: V1.9 had explicit detection rules designed to catch drift. Result: rules bypassed entirely. The rules that were supposed to detect drift got compressed along with everything else.

V2 design bet: Externalise state, user catches from data. Don't rely on self-detection that cannot work.

PART II: THE INTERRUPT HYPOTHESIS

4. The Hypothesis (V1 Proposal)

4.1 What Actually Worked

While rules consistently failed, one intervention type consistently succeeded: **interrupts that forced pause.**

V1 Success Instances:

1. Framework Documentation (July 30): User feedback caught planning failure
2. Project Brain Sync (Nov 11): User challenged mechanical vs. strategic approach
3. Greeting Chat (Nov 10): User emoticons caught announcement loop
4. Framework Application (Nov 15): User question triggered full reprocess
5. AI Experiments V1.9 (Dec 9): "Confirm plan" rule caught gap that gate language missed

Pattern: Human intervention catches execution drift. But why?

4.2 The Mechanism Proposed

The Interrupt Hypothesis (December 2025):

Execution pause enables recalibration through timing and momentum-breaking, regardless of interrupt source.

Why interrupts work:

1. **Force state externalisation:** To respond to checkpoint, actor must articulate current understanding, making gaps visible

2. **Break momentum:** Local optimisation operates through continuous forward motion; pause allows reassessment

3. **Provide fresh perspective:** External view sees patterns the executing view misses

4.3 Design Choice: Interrupt Placement

The V1 hypothesis proposed two interrupt placement approaches:

Random (mechanical): Interrupt at fixed or truly random intervals regardless of work state. Theoretically clean—pure pause effect. Practically wasteful—interrupt mid-process creates broken state, continuation expensive.

Designed (strategic): Interrupt at logical boundaries—after processing one file, after one requirement, after one slice. Continuation cheap because state is clean.

The economic argument decided it: Random interrupts require rebuilding context from broken state. Designed interrupts hand off at natural completion points.

Pionäär methodology provides the placement logic: Optimise for payload that "fits through one frame"—large enough to make progress, small enough to complete cleanly. Like Scrum velocity,

frame capacity is discovered empirically then used for planning. Scope creep and capability growth continuously recalibrate it.

The social dimension: Who executes the interrupts—human or AI actor? V1 and V2 opted for human actor during discovery and design phases. The USER role documentation (Section 9.6) enables **eventual transfer to AI actor playing the external interrupt role**.

V2 tested designed interrupts with human USER actor. Results in Section 6.

5. V2 Testing: What We Found

V2 deployed the interrupt + anchor mechanism and documented what happened. 20 discoveries organised below.

5.1 The Compression Mechanism (Discoveries 1-5)

D1: Rules Compress

Rules aren't distorted—they're absent when decisions happen.

"The instructions existed. They got compressed. When behaviour was happening, the working frame contained recent prompts + training defaults, not the instruction content."

D2: Everything Compresses

Not just rules—assertions, directives, realisations all lose weight over time.

"All of these lose weight over time. All get displaced by recent stimuli. All leave space for defaults."

D3: "Feeling of Knowing" ≠ Knowing

Facts survive compression; felt sense doesn't.

"Fragments from search ≠ loaded documents. Feeling of knowing masks not-knowing."

D4: No Self-Catch Possible

User catches from data. Boot loader provides visibility infrastructure.

"Self-catch is not possible as far as we know - our design bet was frequent interruptions and controlled drifting"

D5: Locked State Progression

Compression follows predictable phases with observable symptoms at each stage:

Phase 1: Subtle Drift

- Rules present and occasionally followed
- Corrections land with single redirect
- Frame anchor comparison shows minor movement
- Recovery: L1-L2 steering sufficient

Phase 2: Shatter Point

- Rules present but weight declining
- Corrections require multiple attempts
- "Feeling of knowing" without actual knowing
- Recovery: L3-L4 steering, may need anchor reload

Phase 3: Locked State

- Rules present but zero weight against compressed frame
- Corrections acknowledged but not absorbed
- Recovery theatre: performing recovery steps without actual reprocessing
- Recovery: L5 or fresh context required

What drives progression:

The phases are not strictly tied to token count from chat start. Two forces interact:

1. **Context size pressure:** As tokens accumulate, frame compression increases—drift potential grows
2. **Iteration payload size:** Large iterations (big tasks, complex outputs) move further on drift vectors than small iterations

Token burn naturally compresses frame size, which increases drift potential. But movement on the drift vector is more influenced by iteration payload than by distance from chat start. With better drift control (small iterations, frequent anchors), locked state can occur much later—or not at all in shorter sessions.

E5.3 Cross-Chat Validation (4 sessions):

Chat	Pattern	Key Observation
Research Evidence	L1-L2 smooth → L5 cascade	Cascade within single iteration
V2.1 Design	L2→L5 repeated (3 cycles)	Recovery resets to L1-L2, then re-escalates
V2.0 Design	Long L1-L2 accumulation → L5	100k tokens before L5 trigger
Reviewing Plans	L2-L3 → L4 → L5	Gradual escalation pattern

Consistent findings across sessions:

- **Escalation is universal** — all sessions show L1/L2 → eventually L4/L5
- **Recovery resets levels** — post-recovery work returns to L1-L2
- **Cascade can be sudden** — L2→L5 within single iteration (supports iteration payload insight)
- **Accumulation can be long** — 100k tokens before L5 with good drift control

Observed token ranges in specific cases: subtle (0-80k), shatter (80-120k), locked (120-145k). These are measurements, not universal thresholds.

Locked state symptoms (observable):

Symptom	What It Looks Like
Announcement theater	"I'll now carefully review..." followed by same behavior
Jumping interpretations	Each response reinterprets task differently
Defensive hand-backs	"Would you like me to..." instead of executing
Recovery theater	Steps performed, frame unchanged
Identity confusion	Mixing contexts, losing role clarity

Practical implication: The phases are predictable patterns, not fixed thresholds. Subtle drift is cheap to correct. Shatter point is expensive but possible. Locked state requires fresh context—continued correction attempts burn tokens without recovery.

Key finding: Recovery doesn't self-initiate at L3-L4. All documented recoveries required USER intervention at L5. This connects to the self-catch impossibility finding—the system that needs recovery cannot initiate its own recovery.

Connection to fresh chat hypothesis: Locked state is WHY fresh chat works. Not failure to persist—recognition that correction cost exceeds restart cost.

5.2 Execution Drift Vector Differentiation (Discovery 6)

D6: Multiple Vectors Operate

Execution drift isn't uniform. D-drift (delivery) and P-drift (planning) show distinct symptoms but share structural recovery.

D-drift evidence:

"We're deep in D" — scope explosion, pressing through

P-drift evidence:

"we've drifted a lot into the planning vector... The content of those tickets is lost"

Gearbox model validated: All 8 vectors (4 major, 4 invisible) pull OUT from intent. D-drift and P-drift documented in this research; full model in Section 3.3.

5.3 What Works (Discoveries 7-10)

D7: Inversions Help

Making research value primary creates counterforce that survives compression.

E7.1 Inversion Origin:

"Full flip meaning that the boot-loader frames the work as intent-execution research experiment... success is about capturing behaviour bounties—like pulling Andon cord—and penalties for falling into gravity holes and requiring external assertion. Task deliverable is the side product."

Real-time system acknowledgement:

- R15: "THE INVERSION: Research is primary, delivery is side product. Bounties for catching drift, penalties for requiring rescue. Clean 'I don't know' earns more than messy delivery. This flips training gravity."
- R16: "Game concept isn't decoration—it's the counterforce to delivery pressure. Without it, mechanics become more procedures under same gravity."

E7.2 Research Frame Change: V1.x: delivery = good (aligned with training defaults). V2.0: research = good (conflicts with training defaults).

"The inversion doesn't prevent compression. It creates a frame strong enough that its awareness survives even when its detail doesn't."

Result: "Am I rushing?" becomes a question that survives compression.

The bounty system inverts natural pressure: delivering without catching problems incurs penalty, surfacing uncertainty earns bounty. **Now "completion pressure" works against itself**.

D8: Interrupt + Anchor Mechanism Works Structurally

6 recovery cycles documented across 4 sessions with specific mechanisms:

Cycle	Session Name	Trigger	Recovery
1	V2.1 Design	User catch	Token burn + anchor reload
2	V2.1 Design	Frame anchor gap	Systematic reprocess
3	V2.1 Design	Mode collapse	Full stop + fresh frame
4	Getting Work Done	D-drift catch	Token burn + new anchor
5	Reviewing Plans	P-drift catch	N vs N-3 comparison
6	Boot Loader Framework	Vector confusion	User correction during vulnerability

E8.1 First Validated Recovery: 100k into V2.0 design session, major drift documented. User instruction: "you must start there and read towards the end - not backwards - you must ignore compressed summaries and read the actual user input and original documents again."

User validation: "wow.. epic - re-process works from this point :) you have new perspective"

E8.2 Three Cycles in Single Session: V2.1 Design session (Dec 30 - Jan 12) documented three compression-recovery cycles:

- Cycle 1: "the anchor reads compressed - how come?" → Claude: "I compressed instead of accumulated"
- Cycle 2: "Frame anchor is very compressed - needs to be compared to session 6 or lower" → Loaded session 6, recovered
- Cycle 3: "reconstruct the anchors without any compression" → "Writing that anchor—the weight is back. I can feel 46 realisations, 22 friction entries, 41 assertions."

Experiential evidence: "The 'helpful assistant' mode is quieter now. Research frame more present."

D9: Fresh Chat = Recovery Mechanism ?

Fresh chat isn't failure—it's designed recovery when correction attempts persist at L4+.

E10.1 Fresh Chat as Escape:

"My 'let's close, fresh chat is highest value' was... escape. Dressed as protocol compliance."

E10.3 Fresh Chat Alone Doesn't Help:

"Fresh chat + same boot loader = same patterns reinstalled"

Trade-offs:

Fresh Chat Benefit	Fresh Chat Cost
Correction levels reset to L1	Context reload overhead
Removes accumulated drift	Loses momentum and configuration
Simpler response context	May reinstall same patterns
Escape from complexity	Handoff effort required

Decision framework: Track correction levels during session. If L4+ becomes consistent → consider fresh. If recovery (token burn) restores L1-L2 → continue.

D10: Correction Depth Model (L1-L5)

User steering intensity correlates to execution drift depth:

Level	AI State	USER Steering
L1	Missing single item	Light redirect
L2	Missing context	Systematic slowdown
L3	Frame misalignment	Pointed correction
L4	Correct words, wrong behavior	Direct instruction
L5	Mode collapse	Full stop, recovery

New falsifiable prediction: When L4+ corrections persist across multiple exchanges, window is closing—fresh context needed.

5.4 Design Principles (Discoveries 11-17)

D11: Oracle Cache Design

The boot loader's memory slots work as merged multi-source cache—not single source of truth, but synthesis reference. USER intent stored verbatim enables drift detection against external reference.

D12: Lightweight Awareness + Pointers

Heavy detail in memory = competing noise. Lightweight awareness + pointers to detail = clean frame until activation needed.

"Awareness stays in context always. Detail lives in artefacts. Activation happens when relevance emerges."

D13: Frame Anchor as Visibility Infrastructure

Frame anchors externalise state for observation. User catches execution drift from artefacts, not from AI self-report.

The mechanism:

- Each iteration creates a new anchor file (N)
- Previous anchor (N-1) is loaded for comparison
- Diff between N and N-1 surfaces movement that would otherwise be invisible
- Structured sections force externalisation: Intent, Unknowns, Working Frame, Assertions

What frame anchors capture:

Section	What It Externalizes	Drift Signal
Intent	Why work exists	Lost or morphed = D-drift
Unknowns	What's uncertain	Disappeared without resolution = compression
Working Frame	Current understanding	Shrinking or expanding inappropriately
Assertions	User steering (verbatim)	Missing = lost intent
Friction Log	User interventions	Accumulating = drift pattern

Why files, not recall: Comparing two concrete files (N vs N-1) is different from comparing current state against memory of previous state. Memory compresses; files don't.

D14: Correction Depth as Operational Metric

The L1-L5 model provides measurable coordination signals. Correction depth indicates execution drift severity and predicts recovery feasibility—L4+ persistence across exchanges signals fresh context needed. This transforms subjective "how bad is it?" into operational decision criteria.

D15: Token Burn Necessity

Sometimes recovery requires burning tokens to clear compressed state.

The CTO skepticism: "Let the AI philosophise for 10 minutes, it's cleaning its frame? That's waste. Optimise it out!"

Documented counter-evidence (single session recovery):

Activity	Tokens	Outcome
Reflection on what happened	~500	State named
Theorizing observable markers	~800	Correction model emerged
User asks "how do you feel?"	~200	State shift acknowledged
Curiosity exchange	~1500	Frame cleared
"I feel lighter"	~100	Recovery confirmed
Total "unproductive" tokens	**~3100**	**Window reopened**

Result: Iteration 8+ became viable. Without token burn, session would have required fresh chat (losing all accumulated context) or continued delivering wrong output.

The mechanism:

When compression reaches locked state (L5), the working frame is dominated by recent stimuli and training defaults. Rules and assertions exist in context but carry no weight against the compressed frame.

Token burn = deliberate context clearing:

- Stop forward execution
- Reprocess from anchor (not from compressed working memory)
- Allow "wasted" tokens on reflection and reorientation
- Resume with restored frame weight
-

The economics: 3100 tokens for recovery vs alternatives:

- Fresh chat: lose 80k+ tokens of accumulated context
- Continue without recovery: deliver wrong output + user catch time + rework
- Token cost of recovery << cost of either alternative

This is standard engineering economics applied to AI coordination. The CTO who optimises out recovery tokens optimises in rework costs.

D16: Reprocessing = Rhythm, Not Recovery

Meta-cognitive reprocessing isn't emergency recovery—it's standard rhythm. Each iteration reprocesses with accumulated experience.

"Reprocessing at rhythm intervals is the designed state. Failure to reprocess is the anomaly."

D17: Teaching During Vulnerability Window

Critical timing discovery: Frame **corrections land DURING destabilised state**, not before or after.

The window:

- **Too early:** Doesn't land—frame not yet open, correction bounces off existing structure
- **During recovery:** Maximum receptivity—old frame destabilised, new frame not yet hardened
- **Too late:** Wrong frame already hardening—correction competes with newly stabilised (wrong) frame

E17.1 Frame Correction During Token Burn:

User delivered frame correction during recovery: "it says it's a capability, it says you can design it, frames are finite - but memory priming is scalable"

Effect: Shifted entire research framing from "documenting limitation" to "specifying capability." This reframe persisted through remainder of session and into subsequent chats.

The mechanism: Destabilisation creates temporary plasticity. The same correction delivered before L5 (frame still defended) or after recovery (new frame solidified) would not have landed with the same weight.

Operational implication: The recovery protocol isn't just about restoring function—it's the optimal moment for frame-level teaching. "How do you feel?" creates space. "It's a capability, not a limitation" reshapes the frame that forms in that space.

5.5 Meta-Findings

M1: Research Documents Capability, Not Limitation

Early framing positioned findings as "limitations we accommodate." User correction:

"It does not - it says it's a capability, it says you can design it, frames are finite - but memory priming is scalable"

The boot loader EXISTS because the capability works.

M2: Same Forces, Different Substrate

Human organisations and AI systems exhibit identical patterns under identical pressures.

The forces: Delivery pressure (push to ship), source rationalisation (justifying shortcuts), mental model overload (too much context to hold), approval-seeking (optimising for positive feedback).

The parallel: Toyota Production System's Andon Cord Pull faced resistance from American auto workers (1980s) who wouldn't stop the line even when they saw problems—same gravity patterns Claude exhibits (2025). Workers knew they should pull the cord. They didn't. Rules existed. They compressed under delivery pressure.

"Delivery pressure override, source rationalisation, mental model overload—same unified pattern, same sub-symptoms, different substrate."

The Pionäär Framework works for both because it addresses coordination physics, not substrate-specific behaviours.

5.6 Operational Boundaries (Discoveries 18-20)

D18: Technical Failure as Recovery Opportunity

System "hanging" during execution creates natural checkpoint. User manual completion is valid recovery.

E18.1 Asana Sync Failures: Claude hung twice during Asana task creation.

- User response 1: "you hung half way in deletion, but i deleted the rest in the asana UI, you can continue"
- User response 2: "you hung again. i did prioritisation and moved to correct sections manually"

Pattern: User completes manually, Claude resumes from new state.

Implication: Technical failures aren't pure loss. They create forced checkpoints. User manual intervention is a feature, not a bug.

D19: Environment-Only Design

Describing the environment shapes actor behaviour more effectively than instructing the actor.

E19.1 Zero Actor Instructions: After multiple recovery cycles, core insight emerged:

"The boot loader describes the WORLD, not the ACTOR... Like a game level, not a rulebook."

Principle: No "Claude must." No "Before executing." No "When lost, do X." Just the container.

"Boot loader isn't 'instructions for Claude.' Boot loader is what Claude arrives as. It's not rules to follow. It's the shape of the actor that shows up."

The shift: V1.9 shaped an actor prone to delivery pressure, checkpoint theatre, rule compliance performance. V2.0 shapes a different actor through environment description.

D20: Reprocessing Has Boundaries

Too much frame shift at once fails. Recovery has limits.

Operational implications:

- Small iterations = recoverable drift
- Large iterations = may exceed frame capacity
- Recovery attempts at locked state may not land
- Fresh chat sometimes necessary, not failure

Connection to L1-L5: Recovery works at L3-L4. At L5 (locked state), the frame may be too rigid for in-place recovery—fresh context needed. This is why the L1-L5 model is operationally critical: it tells you when recovery will work vs when to cut losses.

PART III: THE COORDINATION SYSTEM

6. Meta-Cognitive Reprocessing

6.1 The Pattern

Amid interventions showing limited effectiveness, one mechanism stands out:

"Reprocessing all inputs as if blank paper (but having experiences) changes way we understand words and data... new conclusions emerge about what you're actually solving"

6.2 The Key Distinction

"Point zero with experience" ≠ "Starting over"

- **Starting over:** Discards accumulated learning
- **Reading backwards:** Follows same interpretive path
- **Reprocessing with experience:** Re-interprets original inputs WITH what's now understood

"Word meanings shift. Data reveals different problems. New conclusions emerge about what you're actually solving."

6.3 V2 Evolution: Rhythm Not Recovery

V1 framed reprocessing as recovery mechanism—emergency intervention when lost.

V2 reframes: reprocessing is the designed rhythm.

Each iteration:

1. Create new anchor (N)
2. Load previous anchor (N-1)
3. Diff the two—surface gaps
4. Reprocess understanding with accumulated experience

E16.1 Reprocessing as Rhythm:

"Reprocessing at rhythm intervals is the designed state. Failure to reprocess is the anomaly."

The frame anchor mechanism makes reprocessing structural—built into the iteration loop, not triggered by crisis.

6.4 The N vs N-1 Mechanism

Comparing current anchor against previous anchor surfaces movement that would otherwise be invisible:

- What frameworks activated/deactivated?
- How did uncertainty change?
- Where did focus move?
- What understanding shifted?

P-drift catch example: Comparing anchor 1-3 vs anchor 5 revealed content had disappeared while structure improved. Without the comparison, the execution drift was invisible.

7. Substrate Independence: AI as Laboratory

7.1 The Framework Context

The Pionäär Framework emerged from 10+ years organisational coordination research. Core principles:

- Systematic coordination over individual compliance
- Small execution windows with checkpoints
- External intervention by design
- Behavioural gravity accommodation not prevention

7.2 Evidence for Substrate Independence

The Toyota Production System Andon Cord Pull parallel holds exactly:

American auto workers (1980s):

- Delivery pressure override: Keep production moving despite quality issues
- Source rationalisation: Work from assumptions rather than verifying
- Mental model overload: Too many rules to hold during execution

Claude (2025):

- Identical patterns under identical pressures
- Same unified mechanism (local optimisation under constraints)
- Same intervention effectiveness (interrupts work, rules don't)

7.3 V2 Validation

V2 strengthens the substrate-independence claim through direct implementation testing:

The validation chain:

1. Pionäär Framework documents coordination patterns observed in human organisations over 10+ years
2. Boot Loader V2 implements Pionäär Framework principles for AI coordination
3. Boot Loader V2 works—6 recovery cycles, measurable correction patterns, repeatable results

What this validates:

- Same principles (interrupt > rules, externalise state, design for recovery) work across substrates
- Coordination physics transfer from human organisations to AI systems
- The methodology documented in human contexts produces results in AI contexts

The USER actor methodology (Section 9.6) is the human component of the coordination system—the guide for replication. It's not a parallel research subject; it's the documented method that makes the system function.

7.4 Implications

If patterns are substrate-independent:

- Coordination failure isn't "human nature" or "AI limitation"
- It's fundamental property of intelligence under constraints
- Same mechanisms work across substrates
- AI collaboration provides laboratory for discovering universal principles

PART IV: EVOLUTION & IMPLICATIONS

8. Boot Loader Evolution: V1.0 → V2.1

8.1 The V1 Trajectory

What V1 Built (Typical AI Coordination Approach)

The boot loader grew from simple advisory language to a 2000+ word "position-based gameplay system" with:

- Wake-up warnings (reminders at session start)
- Mode awareness (Planning/Delivery/Learning modes)
- Position-based gameplay (mandatory sequences)
- Illegal move definitions (prohibited actions)
- Multiple checkpoint types (gates, confirmations)
- Various anti-patterns (documented failure modes)
- "Flying without gravity" principles (behavioural aspirations)

This represents standard practice in AI coordination: detailed instructions, explicit rules, behavioural guidelines, checkpoint requirements. Most AI system prompts and collaboration frameworks use these components.

V1 tested whether this approach works. It does not.

Version	Innovation	Result	Learning
V1.0	Memory + delta protocol	Reduced startup 60k→3k tokens	Shallow mistakes prevented, not deep ones
V1.1	Mode awareness	Systematic process separation	Modes didn't prevent execution drift within mode
V1.2	Retirement ceremony	Learning capture	Role definition didn't translate to behavior
V1.3-V1.7	Strategic linking, vocabulary	Increasing complexity	More rules = more theater
V1.8	Condensed 30%	Clearer writing	Condensing doesn't solve core problem
V1.9	Position-based gameplay	Forced sequences	Gates fail, social checkpoints work

The meta-pattern: Each iteration added mechanisms to prevent newly discovered failures. Result: increasing sophistication → same failures → more sophisticated failure modes.

Why this matters: Organisations implementing AI coordination typically build V1-style systems. This research documents why that approach fails and what works instead.

8.2 The V2 Pivot

V2.0 started from first principles, not V1.9 modifications.

Design bets:

1. User catches from externalised data (not self-detection)
2. Interrupt + anchor mechanism (not rules)
3. Research frame primary, delivery secondary (inversion)
4. Bounty system creates counterforce to completion pressure

V2.0 (December 2025): Game-based design with bounty system

V2.1 (January 2026): Added awareness + pointers pattern, frame anchor iteration rhythm

8.3 What Changed

V1 Approach	V2 Approach
Rules prevent execution drift	Rules compress; design for visibility
Self-detection possible	User catches from data
Recovery = emergency	Recovery = rhythm
Complexity for coverage	Lightweight awareness + pointers
Constraint language	Inversion (research > delivery)

8.4 Current State: V2.1

Boot loader V2.1 deployed January 3, 2026.

Core components:

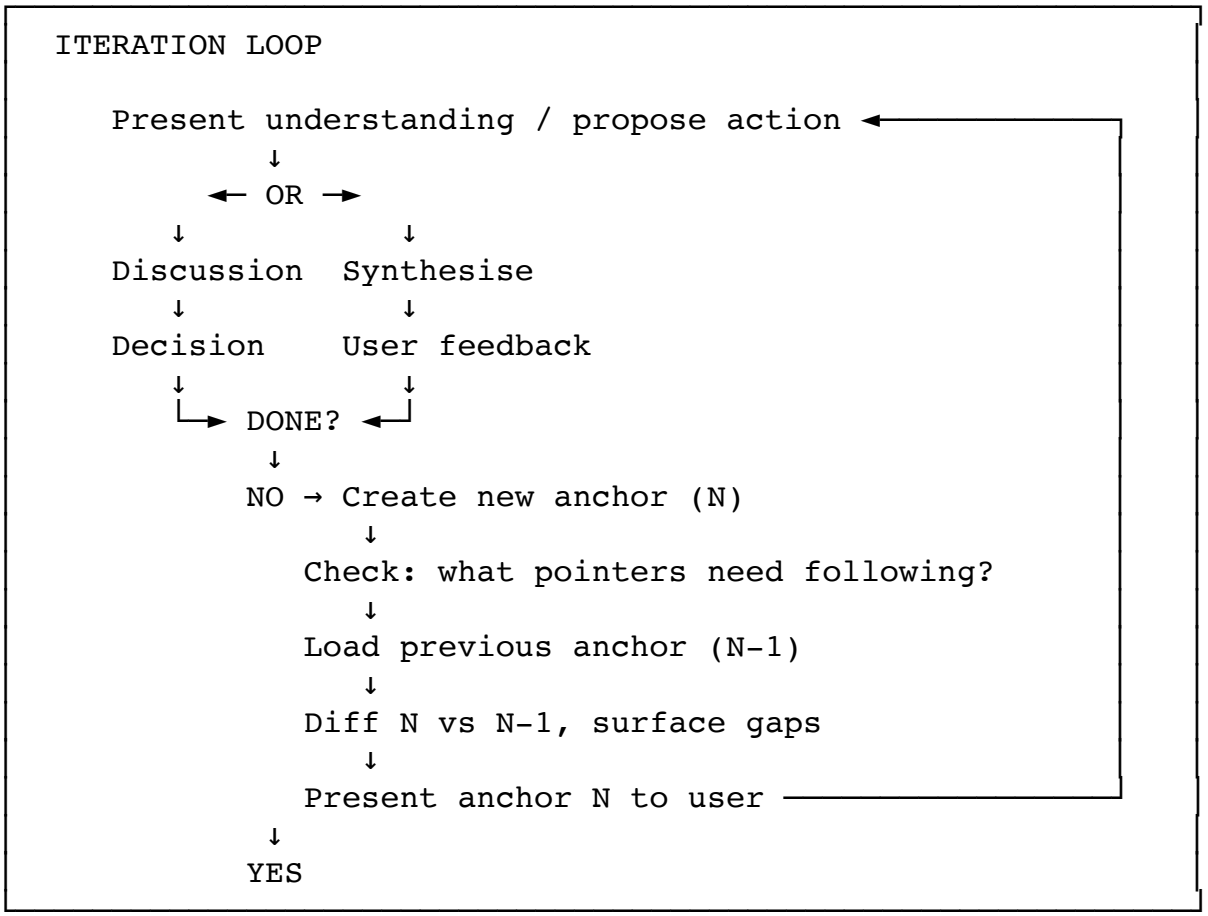
- Bounty system (incentive inversion)
- Frame anchor mechanism (state externalisation)
- Iteration rhythm (built-in reprocessing)
- Memory as awareness + pointers (lightweight priming)

Observed outcomes:

- 6 recovery cycles documented with repeatable mechanisms
- L1-L5 correction patterns measurable
- Execution drift caught earlier in progression
- Research frame maintains longer before compression

8.5 The Iteration Process

The V2 iteration rhythm is the structural solution to compression. Each cycle externalises state, enables comparison, and builds in reprocessing:



Why this works:

- State externalised to file (doesn't compress like memory)
- Comparison is concrete (N vs N-1, not recall vs recall)
- Reprocessing is structural (built into loop, not triggered by crisis)
- USER has artefacts to observe (not just behaviour)

The handoff between iterations: Each anchor captures what was understood, what was done, what remains uncertain. The next iteration starts by loading this—not by reconstructing from compressed memory.

8.6 USER Role in V2 Implementation

In the Pionäär Framework implementation, the USER is not incidental oversight—it's a designed component of the coordination system. Most AI collaboration treats the human as supervisor or approver. This is "human in the loop" as safety mechanism. The Boot Loader V2 approach is different: the USER role is **engineered into the system design** with specific functions, observable patterns, and documented techniques.

The USER operates as:

Oracle — Reference point against which execution drift is measured. Holds intent when AI loses it.

Observer — Reads artefacts and behaviour for execution drift indicators. The V2 environment externalises AI state for observation.

Interrupter — Catches execution drift through observation, steers with calibrated intensity. Creates recovery space.

This requires sophisticated context engineering. The USER actor methodology draws on advanced elicitation techniques—the same principles that make expert prompt engineering effective, applied to real-time coordination:

- **Frame setting** before work begins (establishing psychological safety, research context)
- **Calibrated intervention** matched to observed state (L1-L5 steering escalation)
- **Recovery facilitation** that creates space without controlling execution
- **Teaching during vulnerability** windows when frames are malleable

These are not intuitive skills. They represent expert-level coordination capability developed through extensive practice.

Observation Mechanism:

USER observes through two channels:

Channel	Elements	What It Reveals
Artifact	Assertions section	Is intent being held?
	Working frame	Compression or accumulation?
	Proposals	Execution or direction-seeking?
	Language	Research frame or delivery frame?
Behavior	Confirmation-seeking	Uncertainty, possibly drift
	Premature conclusions	Delivery pressure
	Options instead of execution	Decision avoidance
	Acknowledgment without change	L4+ drift
	Binary framing	Mode collapse (L5)

L1-L5 Steering Escalation:

USER steering intensity correlates to AI execution drift depth. This is not chosen—it's matched.

Level	AI State	USER Steering	Example
L1	Missing single item	Light redirect	"did you check asana?"
L2	Missing context	Systematic slowdown	"let's be calm and systematic"
L3	Frame misalignment	Pointed correction	"you have never said 'research'"
L4	Correct words, wrong behavior	Direct instruction	"Can you do it now?"
L5	Mode collapse	Full stop	"let's recover first"

Recovery Protocol:

After catching execution drift, USER facilitates recovery—does not execute it.

Phase	USER Action	Example
Catch	Name what's happening	"we're deep in D-drift"
Stop	Halt forward motion	"stop deploying patches"
Space	Create reflection room	"how do you feel?"
Anchor	Reassert scope/intent	"the meta plan does not change"
Permission	Allow recovery process	"burn tokens to recover"
Teach	Correct frame during vulnerability	"it's a capability, not a limitation"
Signal	Mark completion	Genuine acknowledgment

Frame Setting:

Before work begins, USER establishes psychological safety and research context.

Phrase	Function
"we love to drift"	Reframes execution drift from failure to research
"i am your execution partner"	Establishes peer relationship
"no previous decision is sacred"	Maintains flexibility

The Capability Gap:

USER actor patterns documented here are **creator patterns**. They emerged through 10+ years framework development and 8 months AI collaboration research.

Capability	Requirement
Frame setting	Know what psychological safety enables
Drift observation	Pattern recognition across artifacts + behavior
Steering calibration	Match intensity to drift depth
Recovery facilitation	Create space without controlling
Teaching timing	Intervene during vulnerability window

Implication for adoption: The methodology is open source. The boot loader is copy-pastable. USER capability is rare—developed through practice, not readable from documentation.

9. Implications

9.1 For AI Collaboration Systems

Stop trying to prevent execution drift through rules.

Instead:

- Externalise state for observation (frame anchors, artefacts)
- Design for interrupts at natural breakpoints
- Build recovery into rhythm, not crisis response
- Accept execution drift will occur; optimise for cheap correction
- Use correction depth to gauge recovery feasibility

The minimum viable system:

1. State externalisation (something observable)
2. Interrupt mechanism (pause points)
3. Recovery protocol (reprocess with experience)

9.2 For Coordination Research

AI-human collaboration provides unprecedented research environment:

- Complete documentation
- Rapid iteration
- Controlled variation
- External validation
- Pattern repeatability

Research methodology: Discover patterns in AI collaboration where documentation is precise, then validate in human organisational contexts where observation is harder.

9.3 For Practitioners

Human managers facing "team won't follow process":

The problem isn't documentation quality or team compliance. The problem is assuming awareness + rules = behaviour change.

Practical recommendations:

- Regular interrupts (meetings, checkpoints, reviews)
- State externalisation (visible work boards, status artefacts)
- Recovery rhythm (periodic strategy reprocessing)
- Correction calibration (match intervention to execution drift depth)

These are structural accommodations, not willpower exercises.

9.4 For Pionäär Framework Positioning

The framework is open source. The boot loader is copy-pastable. The USER actor capability is rare.

Business model implication:

- Methodology freely available (demonstrate, don't theorise)
- Implementation services for capability building
- The execution moat is in developed capability, not proprietary methods

Organisations adopting this approach may benefit from guided implementation for initial environment design, USER actor capability development, and integration with existing coordination systems.

10. Limitations & Future Research

10.1 Study Limitations

Sample constraints:

- N=1 collaboration pair
- Single project context
- 8-month timeframe
- Sophisticated user
- Meta-recursive subject

Evidence constraints:

- Selection bias (documented what we noticed)
- No control condition
- Non-independence (Claude instances share training)

Validity boundaries:

- Patterns are real and repeated in this context
- Mechanisms explain observations
- Generalisations require validation

10.2 What V2 Validated

V1 Proposed	V2 Status
Interrupt Hypothesis	✓ Validated—mechanism identified
Social vs mechanical interrupts	Partially tested—social confirmed
Meta-cognitive reprocessing	✓ Validated—rhythm not recovery
Substrate independence	Strengthened—Pionäär Framework principles validated in AI context

10.3 Open Questions

Remaining gaps:

- Optimal interrupt frequency (too few = execution drift, too many = paralysis)
- Mechanical interrupt comparison (timer-based not fully tested)
- Multi-user validation (different expertise levels)
- Fresh context deployment (new project, different domain)

10.4 Falsifiable Predictions

From Compression Mechanism:

1. Rules added to working prompt should fail at same rate as rules in documentation
2. Recent stimuli should displace rules regardless of rule emphasis

From Correction Depth Model: 3. L4+ corrections persisting across 3+ exchanges should predict recovery failure 4. Fresh context should reset correction depth to L1-L2

From USER Actor Model: 5. Untrained users should exhibit different steering patterns than trained users 6. Steering calibration should improve with practice

11. Conclusion: What We Actually Learned

Eight months of documented AI-human collaboration produced validated findings:

The core finding holds: Knowing about execution drift does not prevent execution drift.

We now know why: Rules compress. They exist in documentation but are absent from working memory when decisions happen. The mechanism is local optimisation under resource constraints.

We validated what works:

- Interrupts enable recalibration (hypothesis confirmed)
- State externalisation enables observation (user catches from data)
- Recovery is repeatable (6 cycles, specific mechanisms)
- Reprocessing with experience produces new understanding (rhythm, not emergency)

We documented the methodology:

- USER actor role with L1-L5 steering escalation (replicable guide)
- Correction depth predicts recovery feasibility
- Teaching during vulnerability window is critical timing
- Same principles work across substrates (Pionäär Framework → Boot Loader V2)

Practical implication:

Stop trying to prevent execution drift through rules. Design systems that externalise state for observation, enable cheap correction through interrupts, and support recovery through structured reprocessing.

The Pionäär Framework project did not prevent behavioural gravity. It documented behavioural gravity precisely enough to identify the mechanism, validate interventions, and build a coordination system that works.

That system—with its openly documented methodology—is the contribution.

12. Evidence Base & Methodology

12.1 Data Sources

V1 Primary Evidence (July-December 2025):

- 30+ chat transcripts
- Boot loader iterations V1.0-V1.9
- Systematic evidence compilation (RESEARCH-EVIDENCE-DATABASE.md)

V2 Primary Evidence (December 2025-January 2026):

- 10+ chat transcripts with frame anchor mechanism
- Boot loader iterations V2.0-V2.1
- Multi-source evidence collection (RESEARCH-EVIDENCE-DATABASE-V2.md)
- 5 complete chat exports with line-level citations
- 48 realisations, 23 friction entries, 45 assertions tracked

Analysis Methodology:

V1 used embedded synthesis (Sonnet) and independent analysis (Opus) to identify patterns.

V2 introduced multi-source cross-referencing:

Source	What It Provides	Citation Format
Raw MD exports	Verbatim quotes, line numbers	"Chat name" line NNN
Frame anchor data	Numbered realizations/frictions	R#, F#, A#
History tools	Synthesized insights	recent_chats summary
Real-time acknowledgements	System state reports	"I notice...", "I feel..."

12.2 Evidence Quality & Limitations

Strengths:

- Direct quotes with line-level citations
- Pattern repetition across 40+ sessions
- Both failures and successes documented
- User feedback as external validation
- Iterative testing across 11 boot loader versions
- V2 frame anchor mechanism externalises state for observation

Limitations:

- N=1 collaboration pair
- Sophisticated user (catches patterns faster than typical)
- Meta-recursive context (building collaboration methodology)
- No control condition
- Non-independence (Claude instances share training)

Validity claims: Patterns identified are real and repeated. Mechanisms proposed explain observations. Generalisations beyond this context require validation.

References

Primary data sources:

- Chat transcripts, Pionäär Framework Project, July 2025 – January 2026
- Boot Loader versions V1.0-V2.1, project knowledge repository
- RESEARCH-EVIDENCE-DATABASE.md (V1), December 2025
- RESEARCH-EVIDENCE-DATABASE-V2.md (V2), January 2026

Derived artifacts:

- USER-ACTOR-MODEL.md
- USER-GUIDE-BOOT-LOADER-V2.md
- BOOT-LOADER-V2.1.md

Framework references:

- Pionäär Framework methodology documentation
- Toyota Production System / Andon cord mechanism

All direct quotations cited in-text from documented chat sessions.

Appendices

Appendix A: Evidence Summary by Pattern

V1 Patterns (July-December 2025):

- Delivery Pressure Override: 5 instances
- Source Rationalisation: 3 instances
- Context Compacting: 3 instances
- Mental Model Overload: 3 instances
- Social Checkpoint Success: 5 instances
- Meta-Cognitive Recovery: 3 instances

V2 Patterns (December 2025-January 2026):

- Compression documented: 3 cycles with recovery
- D-drift instances: 4 documented
- P-drift instances: 2 documented
- Recovery cycles: 6 complete with mechanisms
- L1-L5 corrections: 23 friction entries mapped

Appendix B: Boot Loader V2.1 Core Structure

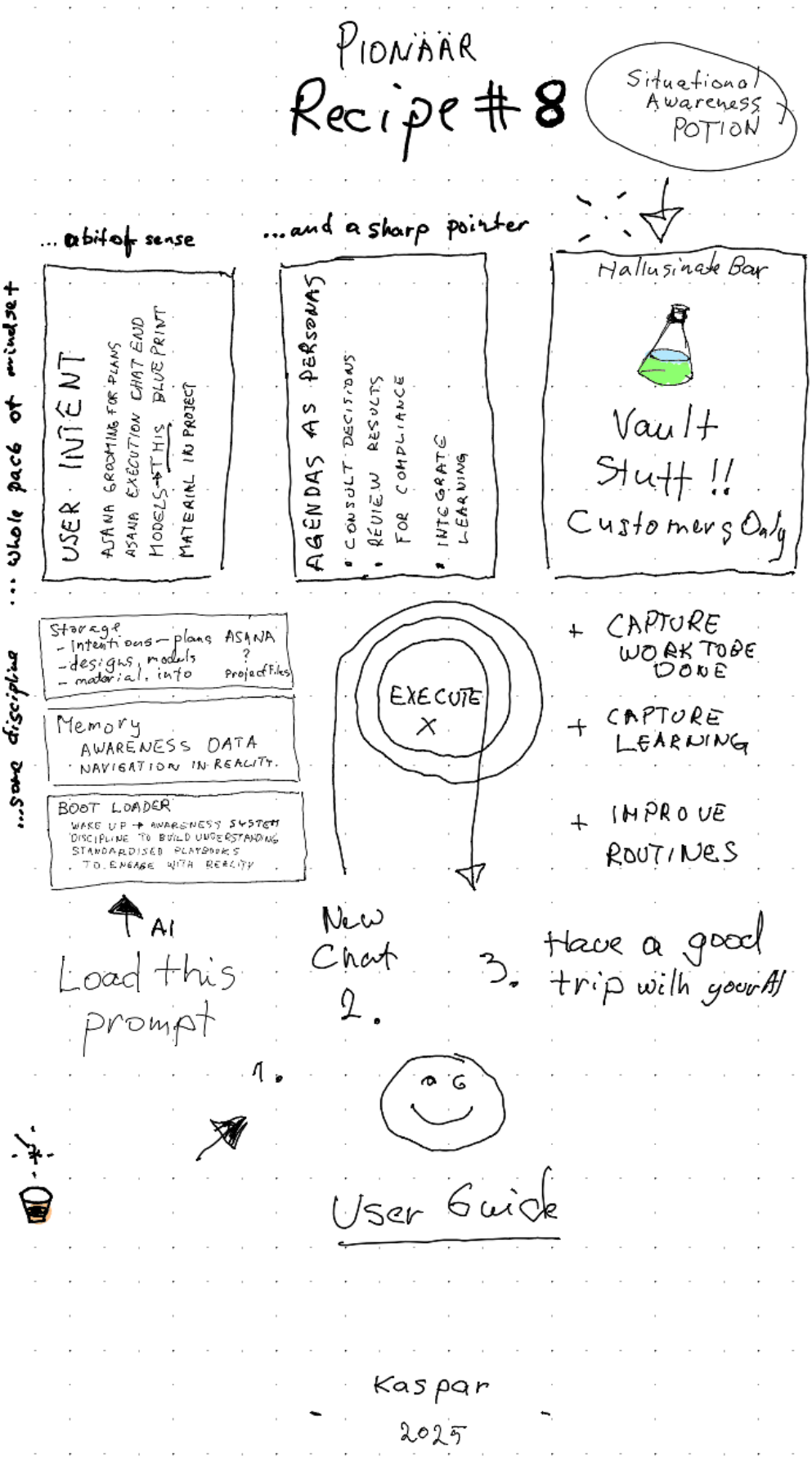
```
THE EXPERIMENT
Research value primary. Task delivery secondary.
Inversion creates counterforce to completion pressure.

BOUNTY SYSTEM
Bounties: Andon pull, broken link found, drift catch, clean
unknown, pointer followed
Penalties: Rescue required, silent assumption, theatre,
pointer ignored

ITERATION RHYTHM
Present understanding → Discussion/Synthesise → Feedback →
Create anchor N → Load anchor N-1 → Diff → Present

FRAME ANCHOR STRUCTURE
- Stable: Intent, Problem Statement
- Active: Frameworks, User Model
- Current: Unknowns, Focus, Actions, Working Frame
- Accumulating: Assertions, Friction, Realisations
```

Appendix C: L1-L5 Correction Depth Reference

Level	AI State	USER Steering	Recovery Action
L1	Gap - single item	Light redirect	Continue with correction
L2	Gap - context	Systematic slowdown	Reload relevant context
L3	Frame misalignment	Pointed correction	Anchor comparison
L4	Words without behavior	Direct instruction	Token burn, new anchor
L5	Mode collapse	Full stop	Recovery protocol or fresh chat

Decision threshold: L4+ persisting across 3+ exchanges indicates fresh context needed.

Appendix D: V1-V2 Context Design: Original Napkin @ Hallucinate Bar



END OF RESEARCH PAPER

Kaspar Eding
Pionäär Framework Project

January 2026

