Phase Diagrams for Human–AI Development: An Observational Theory of Finalization and Responsibility

Katsuya Shibuki

2026-01-16

Abstract

In human–AI collaborative development, the generation of change units (e.g., PRs, diffs, change requests) accelerates and the number of concurrent change processes increases. Meanwhile, exposure to irreversible loss (safety, legal, economic, reputational, etc.) and accountability ultimately collapse at a responsibility-fold point (RP) coupled to observable finalization events. As generation pressure (arrival rate $\times$ concurrency) grows, the RP fold point's effective service rates for finalization and for maintaining the correctness reference system (CRS) and verification recipes (VR) can become bottlenecks. The operational regime may then exhibit phase transitions—sudden shifts from stable to failure modes—regarding (a) controllability (whether unfinalized WIP remains bounded), (b) CP health (whether CRS/VR remain coherent rather than stale), and (c) reproducibility (whether the same verification recipe yields a unique finalized state).

This paper separates the development system into a Data Plane (DP; object layer) and a Control Plane (CP; meta layer), and proposes an observational language that describes phase transitions using variables estimable from logs. Core variables include the finalization load ratio ρ, CRS staleness pressure Π, verification-recipe (VR) staleness pressure Ψ, a non-commutativity proxy $1 - \widehat{p}_{\Delta}$, a reproducibility-loss proxy $\widehat{\varepsilon}$, proxies for verification-gate detector performance, exploration share $\widehat{s}$, CRS resolution $\widehat{R}$, and phase-diagram coordinates such as concurrency pressure κ and the "non-commutativity $\times$ strictness" summary χ. Phase transitions are defined observationally as structural changes in distributional shape, with heavy-tailed cycle-time distributions as a canonical example. This paper makes projection relations among phase diagrams explicit, provides two example class propositions (non-monotone boundaries; possible reproducibility loss under concurrent finalization with non-commutative updates), and enumerates falsifiability conditions.

The goal is to avoid premature heuristic short-circuits from experience and instead describe diverse futures (centralized review, formalization and automated verification, exploration-heavy regimes, relaxed reproducibility requirements, institutional shifts in responsibility) as movements of phase boundaries.

---

0. Positioning (Theoretical Stance)

0.1 Objective

To avoid prematurely collapsing experience into a single success narrative, this paper delivers:

1. **Variable definitions**: quantities estimable (or approximable via proxies) from operational logs
2. **Phase diagrams**: visualization of failure/stability modes and branching conditions
3. **Falsifiability conditions**: quantitative criteria under which the framework is refuted or requires revision

This paper's observational language was motivated by experience in a project building operational software with AI coding agents without code inspection (Shibuki 2026). That project succeeded under specific conditions, but this paper does not claim those conditions generalize; the goal is to provide a language describing diverse futures as movements of phase boundaries.

0.2 Stance (semi-formal / falsifiability-oriented)

This paper uses formal notation but does **not** aim for closed-form solutions or global optimization. Following Box's perspective on models as approximations (Box 1976), the question here is usefulness rather than optimality. The goal is a transferable language for design and operation of human–AI collaboration under potential phase transitions, connected to observables.

0.3 Example class propositions

The propositions here are not universal theorems; they illustrate sufficient conditions within a class of assumptions. Even if a proposition fails empirically, the observational language (variables, phase diagrams, falsifiability) remains useful.

---

1. Responsible Principal (RP)

In this paper, **RP (Responsible Principal)** is not an “actor” per se. RP is a **responsibility-fold point**: the locus at which exposure to irreversible loss (safety, legal, economic, reputational, etc.) and accountability ultimately **collapse** at **finalization events**, consistent with safety-engineering perspectives on control and accident risk (Leveson 2012; Perrow 1984). Work can be distributed up to this fold point (across humans, AI agents, teams, tools, and institutions), but beyond finalization events responsibility becomes consolidated by design or by institution.

Operationally, the RP fold point is the place where the system must (i) execute/approve **finalization events** and (ii) maintain/update the **correctness reference system (CRS)** that defines correctness. In current institutions, implementations of RP are typically human individuals or organizations; however, the framework does not assume a particular implementation (human vs non-human, single vs multi-party, centralized vs distributed), only that the fold point exists.

The RP fold point has at least three finite effective “bandwidth” resources (notation fixed throughout):

- **finalization bandwidth** $C_{\mathrm{RP,fin}}(t)$: capacity (service rate) to process finalization events (judgment, audit, approval)
- **CRS maintenance bandwidth** $C_{\mathrm{RP,CRS}}(t)$: capacity to maintain/update the normative/semantic reference system
- **VR maintenance bandwidth** $C_{\mathrm{RP,VR}}(t)$: capacity to maintain/update verification recipes (operational gates)

---

2. Two-Plane Model: Data Plane and Control Plane (object/meta)

2.1 Definitions and mapping

- **Data Plane (DP)**: the object layer (artifacts, states, implementations) updated by change units
- **Control Plane (CP)**: the meta layer (norms, criteria, recovery standards)

The CP/DP terminology is borrowed from network engineering (Kurose 2025), but its meaning matches the meta/object decomposition (normative criteria vs artifacts/states). This paper treats **CP ≡ meta layer** and **DP ≡ object layer**. This separation aligns with the end-to-end argument in system design (Saltzer, Reed, and Clark 1984).

This paper decomposes CP into functional components:

- **CRS (correctness reference system)**: the normative/semantic reference defining correctness
- **VR (verification recipes)**: operational procedures mapping CRS into practical judgments; detectors approximating correctness (operationalization of correctness)
- **RC (rules/commands)**: constraints (rules) and recovery standards (commands); not verification procedures

CRS and VR are distinct: **CRS defines correctness** as a normative reference (Dijkstra 1972); **VR operationalizes it as detectors** whose performance is subject to misclassification and measurement error (Greenland 1988; Carroll 2006). This is a functional decomposition and does not require physical or organizational separation.[1]

2.2 Conceptual diagram

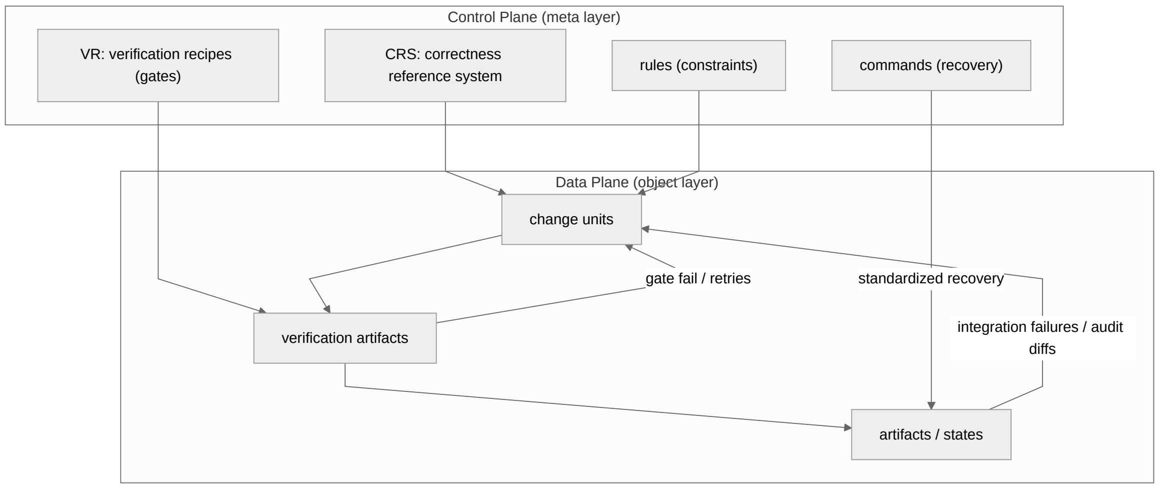
Control Plane (meta layer)
VR: verification recipes (gates)
CRS: correctness reference system
rules (constraints)
commands (recovery)
Data Plane (object layer)
change units
gate fail / retries
standardized recovery
verification artifacts
integration failures / audit diffs
artifacts / states

Figure 1: Control Plane / Data Plane conceptual diagram. CP contains CRS, VR, rules, and commands. DP contains change units, verification artifacts, and states.

[1]Minimal implementation listing: CP (meta layer) = CRS + VR (verification gates/recipes) + rules + commands; DP (object layer) = change units (diff/PR/ticket, etc.) + verification artifacts (tests/analyses/audit scripts) + artifacts/states.

3. Logging Schema (observability)

3.1 Change unit

A "change unit" is any unit (PR, diff, change request, bundled commits, etc.) for which one can define:

- an arrival time
- linkage to verification-recipe (gate) attempts
- linkage to a finalization event (merge/release/regulatory enactment/audit approval, etc.)
- traceable rework (review rounds, retries, revert/hotfix)

3.2 Finalization event (observational definition of irreversibility)

Finalization is a concept; **finalization events** are observable. A finalization event is defined as a boundary after which rollback is materially constrained (costly, delayed, or institutionally prohibited), thereby coupling the system state to irreversible loss exposure (Gray 1978; Bernstein, Hadzilacos, and Goodman 1987) and anchoring the RP definition to observable events.

(Example proxies: rollback cost, recovery lead time, incident severity, audit/regulatory impact.)

3.3 "No backlog" environments

Even without an explicit backlog list, WIP exists and can be observed via delay proxies (cycle time, waiting time, retry density). Hence the model remains applicable.

---

4. Core Metrics: ρ (finalization), Π (CRS), Ψ (VR)

4.1 Finalization load ratio ρ (controllability)

4.1.1 Total inflow (closed-loop included)

$$A_{\text{tot}}(t) = A_{\text{new}}(t) + A_{\text{rw}}(t)$$

- $A_{\text{new}}(t)$: newly arriving change units
- $A_{\text{rw}}(t)$: re-arrivals due to rework

Rework rate (proxy):

$$r(t) = \mathbb{E}\big[\#(\text{review rounds} + \text{gate retries} + \text{revert/hotfix}) \mid t\big]$$

Approximation:

$$A_{\text{tot}}(t) \approx A_{\text{new}}(t) \cdot (1 + \beta r(t))$$

4.1.2 Per-unit finalization cost w_{fin}

$$w_{\text{fin}}(t) = c_{\text{rev}} \cdot \#\text{review rounds} + c_{\text{gate}} \cdot \#\text{gate retries} + c_{\text{conf}} \cdot \#\text{conflicts} + c_{\text{rb}} \cdot \#\text{rollback}$$

4.1.3 Load ratio

$$\rho(t) = \frac{A_{\text{tot}}(t) \cdot w_{\text{fin}}(t)}{C_{\text{RP,fin}}(t)}$$

Sustained $\rho > 1$ indicates a regime where unfinalized WIP tends to diverge (Little 1961; Harchol-Balter 2013).

4.2 CRS staleness pressure Π

DP updates induce demand for CRS updates. Let $\lambda_{\text{CRS}}(t)$ be the arrival rate of CRS update demands and $w_{\text{CRS}}(t)$ their expected cost:

$$\Pi(t) = \frac{\lambda_{\text{CRS}}(t) \cdot w_{\text{CRS}}(t)}{C_{\text{RP,CRS}}(t)}$$

Induced-demand mechanism (one-line summary for separability from VR):

$$\lambda_{\text{CRS}}(t) = g_{\text{CRS}}\Big(\lambda_{\text{new}}(t),\ 1 - \hat{p}_{\Delta}(t),\ \text{domain shift proxies}(t),\ \hat{R}(t),\ \ldots\Big)$$

4.3 VR staleness pressure Ψ

DP updates also induce demand for VR (verification recipe) updates. Let $\lambda_{\text{VR}}(t)$ be the arrival rate of VR update demands and $w_{\text{VR}}(t)$ their expected cost:

$$\Psi(t) = \frac{\lambda_{\text{VR}}(t) \cdot w_{\text{VR}}(t)}{C_{\text{RP,VR}}(t)}$$

Induced-demand mechanism:

$$\lambda_{\mathrm{VR}}(t) = g_{\mathrm{VR}}\Big(\lambda_{\mathrm{new}}(t),\ \text{detector proxies}(t),\ \text{flake rate}(t),\ \text{artifact drift proxies}(t),\ \dots\Big)$$

Example proxies for VR staleness: recipe-revision event density; variance spikes in gate-failure rates; repeated retries due to the same root cause.

---

5. Non-commutativity, reproducibility, and verification-gate detectors

5.1 Non-commutativity proxy $\hat{p}_\Delta$

$$\hat{p}_\Delta(t) = 1 - \Pr(\text{conflict} = 1 \mid \text{concurrent within window } \Delta \text{ at } t)$$

Window Δ is fixed for comparability; commutativity and serializability are central concerns in concurrent systems (Bernstein, Hadzilacos, and Goodman 1987).

5.2 Reproducibility loss proxy and requirement

$$\hat{\varepsilon}(t) = \Pr(\text{non-identical outcome} = 1 \mid \text{same verification recipe at } t)$$

$\varepsilon_{\max}$ is an operational SLO; sustained $\hat{\varepsilon} > \varepsilon_{\max}$ indicates violation. Conflict-free replicated data types (CRDTs) provide one approach to commutativization by design (Shapiro et al. 2011).

5.3 Detector performance (proxies)

Verification gates are treated as detectors. When $(\mathrm{TPR}, \mathrm{FPR})$ is not identifiable, approximate detector performance via proxies such as post-finalization incident rates, audit findings, or recurrence rates after rework (Greenland 1988; Carroll 2006). Increased false-positive proxies can increase retries and push $\rho \uparrow$ through the rework loop.

---

6. Exploration/Execution two-phase view and CRS resolution

6.1 Two phases

- **Exploration**: updates that increase CP resolution (CRS/VR) and decision-ability
- **Execution**: DP updates (implementation, modification, repair) to satisfy CP constraints Debugging is treated as a subtype of execution, not a separate phase. This two-phase view aligns with the exploration–exploitation framework in organizational learning (March 1991).

6.2 Exploration share $\hat{s}$

$$\hat{s}(t) = \frac{\#\text{exploration updates (pre-finalization)}}{\#\text{all updates (exploration+execution)}}$$

6.3 CRS resolution $\hat{R}$

$\hat{R}(t) = w_1 \cdot \#\text{contract clauses} + w_2 \cdot \#\text{CRS-aligned invariants} + w_3 \cdot \#\text{explicit boundary definitions}$

$\hat{R}$ measures resolution as operationalizable decision criteria, not prose volume.

Directional implication: generally $\hat{s} \uparrow$ is expected to be accompanied by $\hat{R} \uparrow$ (unless exploration degenerates into non-operational prose). Typically:

- $\hat{R} \uparrow \Rightarrow r \downarrow,\ 1 - \hat{p}_\Delta \downarrow,\ \hat{\varepsilon} \downarrow$.
- Exploration that does not increase $\hat{R}$ can increase w_{CRS} and push $\Pi \uparrow$.

---

7. Concurrency pressure κ: observable "arrival rate × concurrency"

Let $\lambda_{\text{new}}(t)$ be the new arrival rate and $L(t)$ a concurrency-window proxy (parallel WIP). Define:

$$I(t) := \lambda_{\text{new}}(t) \cdot L(t)$$

and nondimensionalize by RP finalization bandwidth:

$$\kappa(t) = \frac{\lambda_{\text{new}}(t) \cdot L(t)}{C_{\text{RP,fin}}(t)}.$$

---

8. Projection relations and χ

$$\rho = \rho(\kappa, \hat{p}_\Delta, \hat{\varepsilon}, \text{detector proxy}, \hat{s}, \hat{R}, \dots)$$

$$\Pi = \Pi(\kappa, g_{\text{CRS}}, \hat{R}, \dots), \quad \Psi = \Psi(\kappa, g_{\text{VR}}, \text{detector proxy}, \dots)$$

Define a dimensionless "non-commutativity $\times$ strictness" summary:

$$\chi := f!\,(1 - \hat{p}_\Delta,\ \varepsilon_{\max};\ \Delta)\,, \quad \text{and in this paper I use the simple instantiation } f = \frac{1 - \hat{p}_\Delta}{\varepsilon_{\max}}.$$

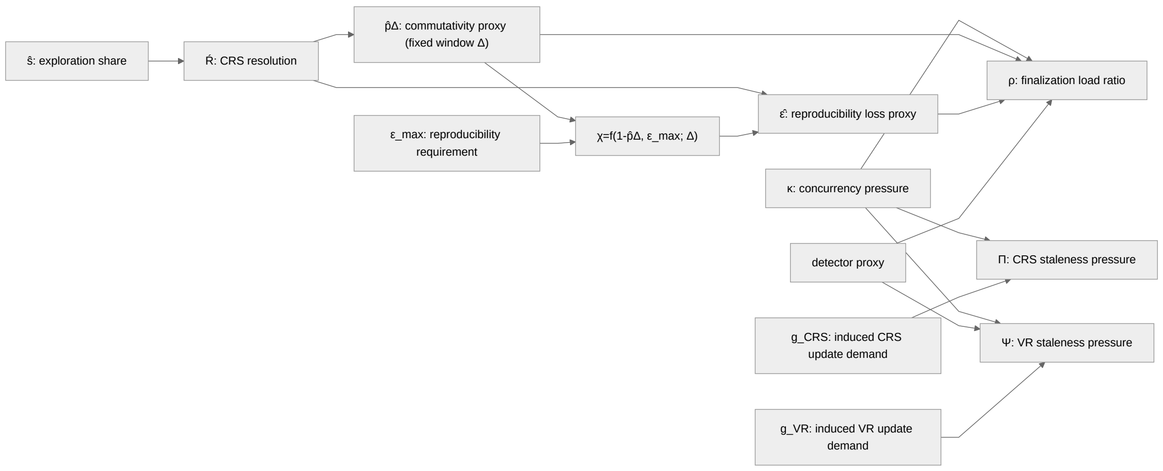
ŝ: exploration share
R̂: CRS resolution
p̂Δ: commutativity proxy (fixed window Δ)
ε_max: reproducibility requirement
χ=f(1-p̂Δ, ε_max; Δ)
ε̂: reproducibility loss proxy
ρ: finalization load ratio
κ: concurrency pressure
detector proxy
Π: CRS staleness pressure
g_CRS: induced CRS update demand
Ψ: VR staleness pressure
g_VR: induced VR update demand

Figure 2: Projection relations among variables. drives , Π, Ψ. Exploration share ŝ influences resolution R̂, which affects commutativity and reproducibility proxies.

---

9. Observational definition of phase transitions and phase diagrams

9.1 Observational definition (distribution-shape change)

A phase transition here means a structural change in distributional shape (not merely a smooth mean shift). A canonical example is heavy-tailed cycle-time distributions (Clauset, Shalizi, and Newman 2009) (rapid growth of upper quantiles, tail-index change) for time-to-finalization. Regions where such structural changes persist are treated as transitions from stable to failure regimes. Self-exciting point processes can model event clustering in failure cascades (Hawkes 1971).

9.2 Phase A-CRS: $(\rho \times \Pi)$

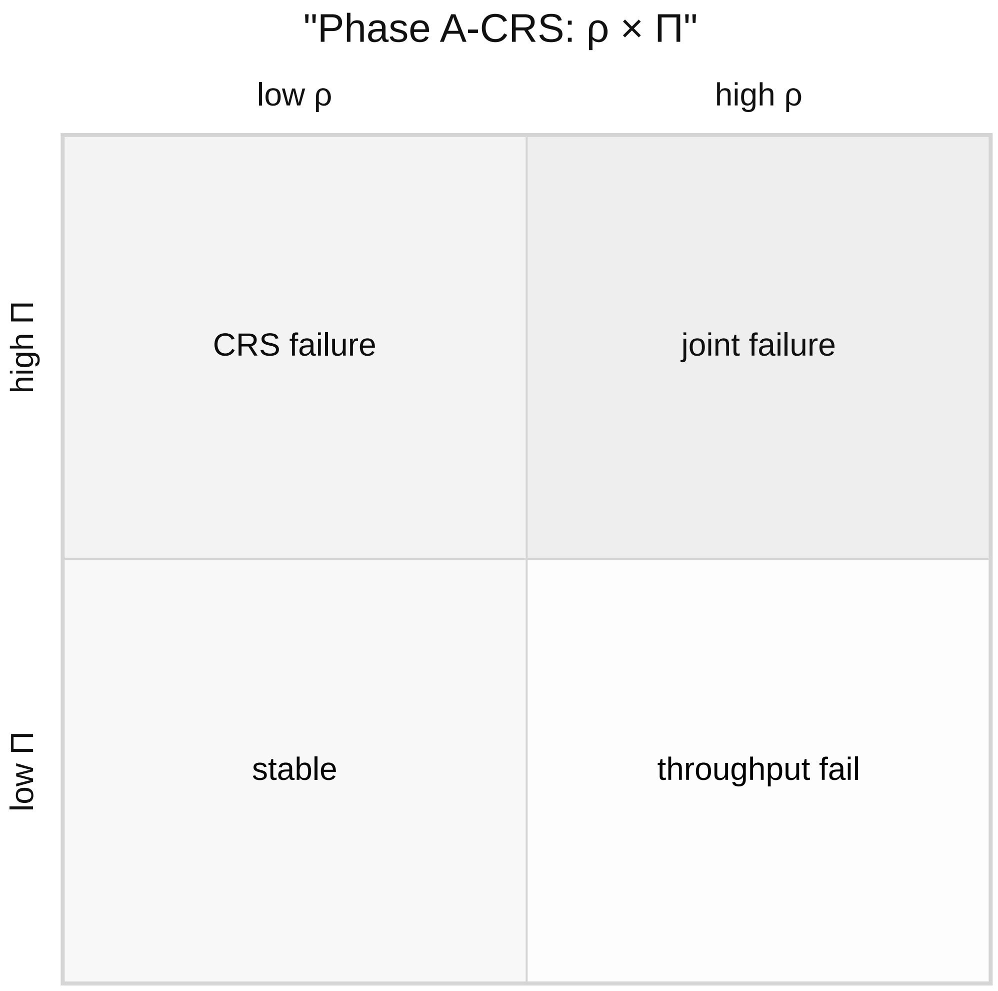

Figure 3: Phase A-CRS diagram: (finalization load) $\times$ Π (CRS staleness). Quadrants show stable, CRS failure, throughput failure, and joint failure regions.

9.3 Phase A-VR: $(\rho \times \Psi)$

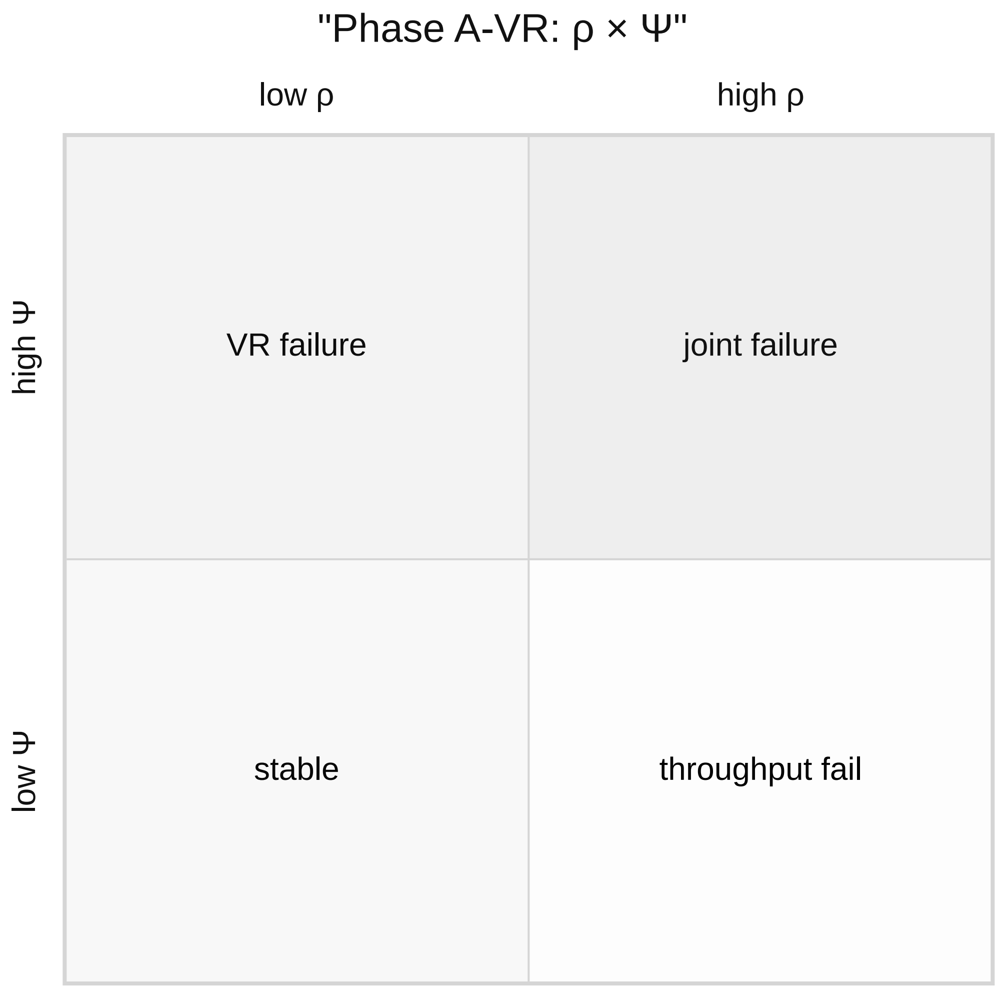

Figure 4: Phase A-VR diagram: (finalization load) × Ψ (VR staleness). Quadrants show stable, VR failure, throughput failure, and joint failure regions.

Joint-failure region (illustrative sufficient condition):

$$(\rho \text{ high}) \ \wedge \ ((\Pi \text{ high}) \ \vee \ (\Psi \text{ high})).$$

Which component tends to lead (CRS-driven vs VR-driven) is domain-dependent: high regulation/audit/safety constraints often yield CRS-driven failure, while high implementation drift and verification-artifact drift often yield VR-driven failure.

9.4 Phase B: finalization constraint phase (non-commutativity × strictness)

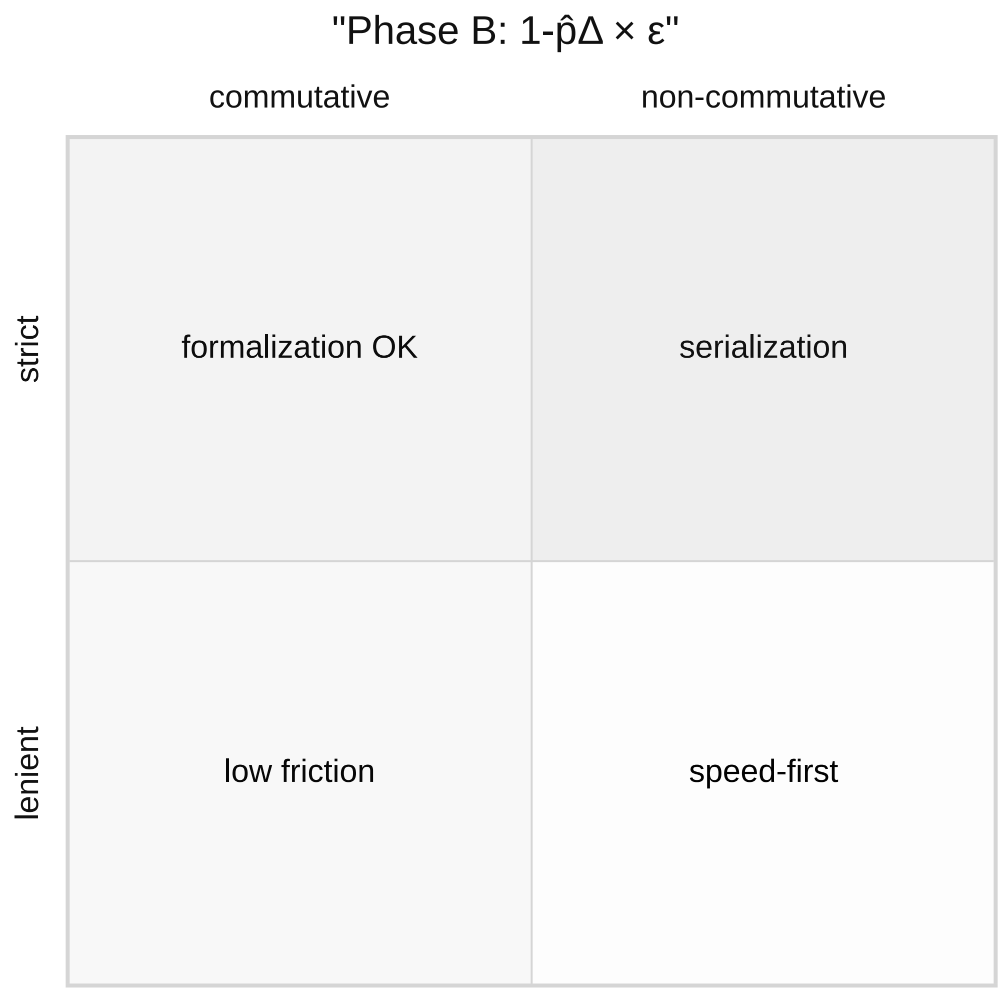

Figure 5: Phase B diagram: non-commutativity $(1 - \hat{p}_\Delta)$ × strictness $(\varepsilon_{\max})$. Quadrants show low friction, formalization OK, speed-first, and high serialization pressure.

9.5 Phase C: configuration phase $(\kappa \times \chi)$

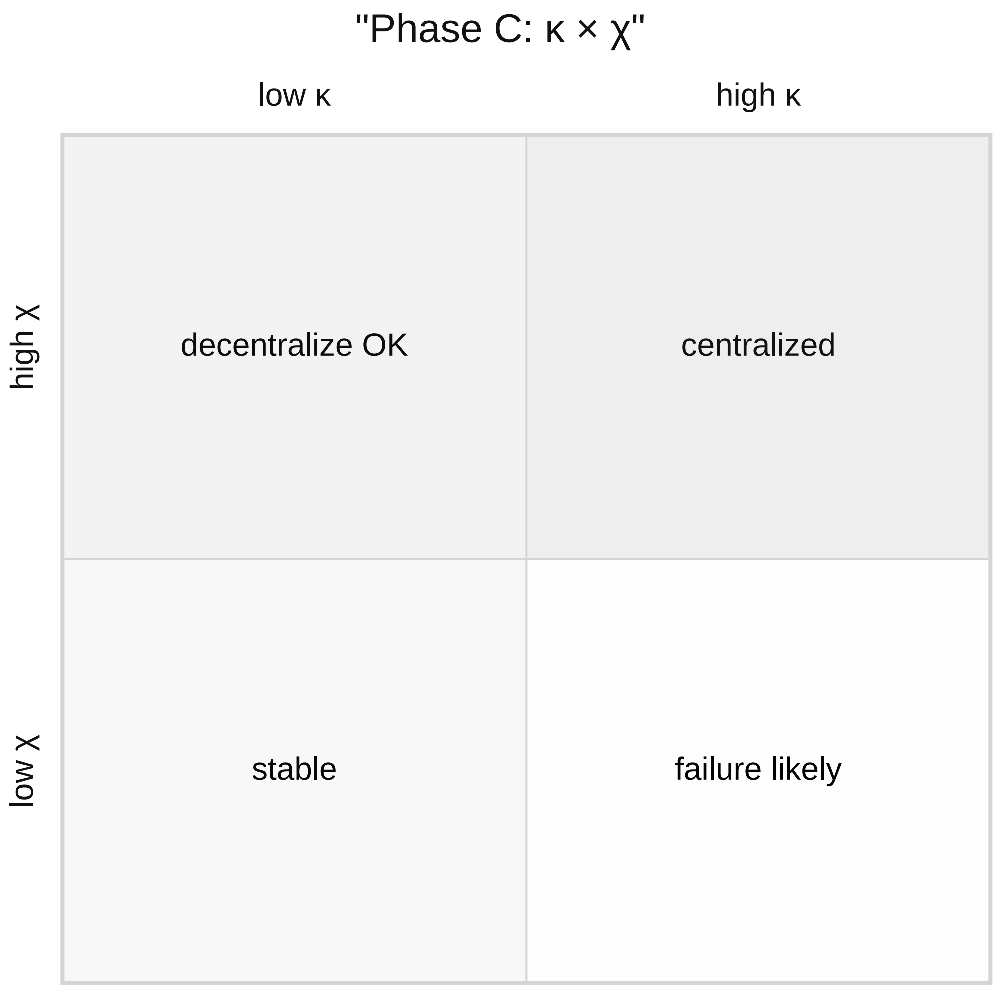

Figure 6: Phase C diagram: (concurrency pressure) × (non-commutativity × strictness). Quadrants show stable, decentralize OK, failure likely, and centralized finalization valuable.

10. Flat enumeration of futures (branching regimes)

- **Fut-1: Centralized finalization dominates** $\kappa \uparrow$ and $\chi \uparrow$. Finalization becomes a structural bottleneck.
- **Fut-2: Formalization and automated verification dominate** $1-\hat{p}_\Delta \downarrow$ and improved detector proxies reduce the need for centralized review.
- **Fut-3: Exploration and resolution dominate** $\hat{s} \uparrow, \hat{R} \uparrow$ reduce r and suppress ρ, absorbing increases in κ.
- **Fut-4: Speed-first via relaxed reproducibility requirements** $\varepsilon_{\max} \uparrow$ implies $\chi \downarrow$, shifting Phase B toward the lenient region.
- **Fut-5: VR fails first** $\Psi \uparrow$ precedes $\Pi \uparrow$, increasing gate-induced rework and pushing $\rho \uparrow$.
- **Fut-6: Reference maintenance ceases to dominate** $\max \Pi, \Psi$ remains low (localization/automation), shifting focus to audit/recovery/incident control.

---

11. Example class propositions

Proposition 1 (existence of non-monotone boundary): conditions under which $\rho(a)$ can have a minimum

For an intervention level $a \in [0,1]$ (encompassing automation, AI assistance, and parallelization), consider:

$$\rho(a) = \frac{A(a), G(a), w_{\text{fin}}(a)}{C_{\text{RP,fin}}(a)}, \quad G(a) = 1 + \beta r(a)$$

Taking a log-derivative:

$$\frac{\rho'(a)}{\rho(a)} = \frac{A'}{A} + \frac{G'}{G} + \frac{w'_{\text{fin}}}{w_{\text{fin}}} - \frac{C'_{\text{RP,fin}}}{C_{\text{RP,fin}}} \tag{LD}$$

If there exists $a^* \in (0,1)$ such that

$$\frac{A'}{A} + \frac{G'}{G} + \frac{w'_{\text{fin}}}{w_{\text{fin}}} - \frac{C'_{\text{RP,fin}}}{C_{\text{RP,fin}}} = 0 \tag{C1}$$

and

$$\left(\frac{A'}{A} + \frac{G'}{G} + \frac{w'_{\text{fin}}}{w_{\text{fin}}} - \frac{C'_{\text{RP,fin}}}{C_{\text{RP,fin}}} \right)' \Big|_{a=a^*} > 0, \tag{C2}$$

then $\rho(a)$ can have a local minimum at a^* (a U-shape is possible). Since $A(a)$ is exogenous, convexity of $r(a)$ alone does not imply a U-shape; the boundary is determined by the balance of relative change rates among $A, w_{\text{fin}}, C_{\text{RP,fin}}, r$ (Sterman 2000).

Proposition 2 (existence claim): non-commutativity with concurrent finalization can produce reproducibility loss

Define "concurrent finalization" as a finalization window $[t, t{+}\Delta]$ in which the multiset of updates is not totally ordered and application order can depend on scheduling. If non-commutative update pairs occur with positive probability within the window (Bernstein, Hadzilacos, and Goodman 1987; Shapiro et al. 2011), the same verification recipe can yield divergent outcomes, implying $\hat{\varepsilon} > 0$ can occur. A coarse lower bound is:

$$\hat{\varepsilon} \gtrsim 1 - \exp\big(-\gamma, \Theta(\Delta)\big).$$

Here $\Theta(\Delta)$ measures "opportunities for order variation" within window Δ (e.g., expected number of candidate concurrent pairs or non-commutative pairs), and γ aggregates sensitivity from non-commutativity to outcome divergence (including integration structure and detector characteristics).

---

12. Falsifiability conditions (Falsifiability-first)

A framework is scientifically useful insofar as it can be refuted (Popper 1959). Refutation tests should fix a time window (e.g., 4-week moving average; or N-day pre/post intervention windows) and can be implemented via monotonicity tests, change-point detection (Bai and Perron 1998), or distributional distances (e.g., Mann–Kendall; segmented regression/changepoint; KS distance). This paper does not fix a single method; it specifies minimal requirements for falsifiability.

ID	Target hypothesis	Observables (log metrics)	Example refutation criterion
F1	Closed-loop dominance / non-monotone boundary (Prop 1)	ρ, r vs intervention a	r does not increase and ρ improves monotonically as a rises
F2	CRS staleness drives failure	Π and CRS-failure proxies	CRS-failure proxies do not worsen even with sustained $\Pi > 1$
F3	VR staleness drives failure	Ψ and recipe-failure proxies	recipe-failure proxies do not worsen even with sustained $\Psi > 1$
F4	Resolution improves downstream outcomes	$\hat{R}$ vs $r, 1 - \hat{p}_{\Delta}, \hat{\varepsilon}$	improvements do not occur despite $\hat{R} \uparrow$
F5	Detector false positives worsen short-term load	retry proxies vs ρ	ρ does not worsen despite increased retries

ID	Target hypothesis	Observables (log metrics)	Example refutation criterion
F6	Non-commutativity $\times$ strictness implies serialization pressure (Prop 2)	$\widehat{p}_\Delta, \widehat{\varepsilon}$	$\widehat{\varepsilon}$ does not worsen under concurrent finalization even in strict/non-commutative region
F7	Exploration share improves downstream outcomes	$\widehat{s}$ vs $r, 1-\widehat{p}_\Delta, \widehat{\varepsilon}$	no improvement despite $\widehat{s}\uparrow$
F8	Concurrency pressure raises ρ	κ vs ρ	ρ does not systematically rise as $\kappa\uparrow$

---

13. Minimal estimation & intervention design (Identifiability-first)

13.1 Minimal estimation set

$$\rho(t), \Pi(t), \Psi(t), r(t), \widehat{p}_\Delta(t), \widehat{\varepsilon}(t), \text{detector proxy}, \widehat{s}(t), \widehat{R}(t), \kappa(t), \chi(t)$$

plus proxies relevant to g_{CRS} and g_{VR}.

13.2 Treat RP bandwidths as estimable service rates

RP bandwidths $C_{\mathrm{RP},*}(t)$ are not normative constants; they are estimated as service rates from observed throughput and waiting-time/cycle-time distributions (including WIP proxies), and treated as time-varying effective capacities. The same applies to $C_{\mathrm{RP,CRS}}$ and $C_{\mathrm{RP,VR}}$ using CRS/VR update event volumes and lead-time distributions.

Minimal estimation sketch: (1) Extract low-load periods (sufficiently small ρ); use an upper quantile (e.g., 90th percentile) of observed finalization throughput as a proxy for $C_{\mathrm{RP,fin}}$. (2) Alternatively, reference Little's law using WIP proxy L and cycle time to back out a service rate consistent with observed λ_{eff} and W (and analogously for CRS/VR update streams).

13.3 "No backlog" environments

Even without explicit backlog artifacts, ρ can be estimated from waiting times and cycle-time distributions; Π and Ψ can be estimated from event densities and lead-time distributions of CRS/VR updates.

13.4 Intervention axes (examples)

- Generation pressure: λ_{new}, L
- Exploration & resolution: $\hat{s}$, $\hat{R}$
- Detector improvement & recipe updates: detector proxies; λ_{VR}
- Commutativization: improving $\hat{p}_{\Delta}$
- Reproducibility requirement and windowing: $\varepsilon_{\max}$, fixed Δ

---

14. Limitations

- Proxy validity is context-dependent and requires fixed definitions and validation.
- ρ, Π, Ψ are first-order approximations and do not fully capture multi-stage bottlenecks or batching.
- Phase boundaries are bands, not sharp lines, due to lag, transients, and exogenous shocks.
- Many practical interpretations assume current institutions where RP is typically human; if non-human entities institutionally accept irreversible-loss exposure, bottlenecks and phase boundaries may change qualitatively.
- The paper does not prescribe value judgments; it provides branching conditions and observability.
- Software evolution pressures (Lehman 1980) and modularization choices (Parnas 1972) affect how CRS/VR staleness accumulates; these dynamics are not modeled explicitly.

---

15. Conclusion

This paper presents a portable observational language for phase transitions in human–AI collaborative development. The core is to treat (i) finalization load ρ and (ii) dual CP staleness pressures Π (CRS) and Ψ (VR) as primary axes, and connect them to measurable proxies for non-commutativity $\hat{p}_{\Delta}$, reproducibility loss $\hat{\varepsilon}$, detector behavior, exploration share $\hat{s}$, CRS resolution $\hat{R}$, concurrency pressure κ, and strictness summary χ. Phase transitions are defined observationally as distribution-shape changes, and falsifiability conditions specify minimal time-window and test-form requirements. The framework aims to prevent premature heuristic short-circuits and to map diverse futures as movements of phase boundaries.

Glossary

- **CP**: Control Plane (meta layer)
- **DP**: Data Plane (object layer)
- **CRS**: correctness reference system (normative/semantic reference defining correctness)
- **VR**: verification recipes (operational detectors mapping CRS to judgments)
- **RC**: rules/commands (constraints and recovery standards)
- **verification gate (recipe)**: a detector recipe
- **verification artifacts**: tests/analyses/audit scripts (DP-side artifacts)
- **finalization event**: an event after which rollback is materially constrained

RP bandwidths (notation fixed)

- $C_{\mathrm{RP,fin}}(t)$: finalization bandwidth (estimated as a service rate)
- $C_{\mathrm{RP,CRS}}(t)$: CRS maintenance bandwidth (service rate)
- $C_{\mathrm{RP,VR}}(t)$: VR maintenance bandwidth (service rate)

Staleness pressures

- $\Pi = \lambda_{\mathrm{CRS}} w_{\mathrm{CRS}} / C_{\mathrm{RP,CRS}}$
- $\Psi = \lambda_{\mathrm{VR}} w_{\mathrm{VR}} / C_{\mathrm{RP,VR}}$
- $g_{\mathrm{CRS}}(\cdot)$, $g_{\mathrm{VR}}(\cdot)$: induced-demand mechanisms

Non-commutativity and reproducibility

- Δ: fixed concurrency window
- $\hat{p}_\Delta$: commutativity proxy
- $\hat{\varepsilon}$: reproducibility-loss proxy
- $\varepsilon_{\max}$: reproducibility SLO

Exploration and resolution

- $\hat{s}$: exploration share
- $\hat{R}$: CRS resolution proxy

Concurrency pressure

- λ_{new}: new arrival rate
- L: concurrency window proxy
- $\kappa = \lambda_{\mathrm{new}} L / C_{\mathrm{RP,fin}}$: concurrency pressure

Strictness summary

- $\chi = f(1 - \hat{p}_\Delta, \varepsilon_{\max}; \Delta)$, instantiated as $(1 - \hat{p}_\Delta)/\varepsilon_{\max}$

Auxiliary terms for Proposition 2

- $\Theta(\Delta)$: opportunities for order variation within window Δ
- γ: sensitivity from non-commutativity to outcome divergence

---

Conflict of Interest

The author declares no conflict of interest.

References

Bai, Jushan, and Pierre Perron. 1998. "Estimating and Testing Linear Models with Multiple Structural Changes." *Econometrica* 66 (1): 47. https://doi.org/10.2307/2998540.

Bernstein, Philip A., Vassos Hadzilacos, and Nathan Goodman. 1987. *Concurrency Control and Recovery in Database Systems.* 11. [Dr.]. Addison-Wesley Series in Computer Science. Reading, Mass.: Addison-Wesley.

Box, George E. P. 1976. "Science and Statistics." *Journal of the American Statistical Association* 71 (356): 791–99. https://doi.org/10.1080/01621459.1976.10480949.

Carroll, Raymond J. 2006. *Measurement Error in Nonlinear Models: A Modern Perspective, Second Edition.* 2nd ed. Chapman and Hall/CRC Monographs on Statistics and Applied Probability Ser v.105. London: CRC Press LLC.

Clauset, Aaron, Cosma Rohilla Shalizi, and M. E. J. Newman. 2009. "Power-Law Distributions in Empirical Data." *SIAM Review* 51 (4): 661–703. https://doi.org/10.1137/070710111.

Dijkstra, Edsger W. 1972. "The Humble Programmer." *Communications of the ACM* 15 (10): 859–66. https://doi.org/10.1145/355604.361591.

Gray, J. N. 1978. "Notes on Data Base Operating Systems." In *Operating Systems*, edited by G. Goos, J. Hartmanis, P. Brinch Hansen, D. Gries, C. Moler, G. Seegmüller, J. Stoer, et al., 60:393–481. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-08755-9_9.

Greenland, Sander. 1988. "Variance Estimation for Epidemiologic Effect Estimates Under Misclassification." *Statistics in Medicine* 7 (7): 745–57. https://doi.org/10.1002/sim.4780070704.

Harchol-Balter, Mor. 2013. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action.* 1st ed. Cambridge University Press. https://doi.org/10.1017/CBO9781139226424.

Hawkes, Alan G. 1971. "Spectra of Some Self-Exciting and Mutually Exciting Point Processes." *Biometrika* 58 (1): 83–90. https://doi.org/10.1093/biomet/58.1.83.

Kurose, James F. 2025. *COMPUTER NETWORKING A TOP-DOWN APPROACH, 9TH EDITION*. S.l.: PEARSON EDUCATION LIMITED.

Lehman, M. M. 1980. "Programs, Life Cycles, and Laws of Software Evolution." *Proceedings of the IEEE* 68 (9): 1060–76. https://doi.org/10.1109/PROC.1980.11805.

Leveson, Nancy G. 2012. *Engineering a Safer World: Systems Thinking Applied to Safety*. The MIT Press. https://doi.org/10.7551/mitpress/8179.001.0001.

Little, John D. C. 1961. "A Proof for the Queuing Formula: $L = \lambda\ W$." *Operations Research* 9 (3): 383–87. https://doi.org/10.1287/opre.9.3.383.

March, James G. 1991. "Exploration and Exploitation in Organizational Learning." *Organization Science* 2 (1): 71–87. https://doi.org/10.1287/orsc.2.1.71.

Parnas, D. L. 1972. "On the Criteria to Be Used in Decomposing Systems into Modules." *Communications of the ACM* 15 (12): 1053–58. https://doi.org/10.1145/361598.361623.

Perrow, Charles. 1984. *Normal Accidents: Living with High Risk Technologies - Updated Edition*. Princeton, NJ: Princeton University Press. https://doi.org/10.1515/9781400828494.

Popper, Karl. 1959. *The Logic of Scientific Discovery*. 0th ed. Routledge. https://doi.org/10.4324/9780203994627.

Saltzer, J. H., D. P. Reed, and D. D. Clark. 1984. "End-to-End Arguments in System Design." *ACM Transactions on Computer Systems* 2 (4): 277–88. https://doi.org/10.1145/357401.357402.

Shapiro, Marc, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. 2011. "Conflict-Free Replicated Data Types." In *Stabilization, Safety, and Security of Distributed Systems*, edited by Xavier Défago, Franck Petit, and Vincent Villain, 6976:386–400. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-24550-3_29.

Shibuki, Katsuya. 2026. "Zero Human Code: Building Operational Software with AI Coding Agents Without Code Inspection." Zenodo. https://doi.org/10.5281/ZENODO.18244548.

Sterman, John D. 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Nachdr. Boston: Irwin/McGraw-Hill.