Tri-Contour Dynamics of Limited Resource Allocation (TC)

A system model that describes how systems allocate limited resources across three competing functional demands: Survival, Reproduction, and Evolution.

Viktor Jevdokimov, Vilnius, Lithuania

Table of contents

1. INTRODUCTION

1.1 Tri-Contour Dynamics of Limited Resource Allocation – Model and Theory

> ⚠️ **ATTENTION: You may be viewing a downloaded version.**
> The living, latest version of this documentation is always available online: TC Official Documentation

2. THEORY

2.1 The Model

2.1.1 Overview

Tri-Contour Dynamics of Limited Resource Allocation describes how systems allocate limited resources across three competing functional demands: **Survival**, **Reproduction**, and **Evolution**. System behavior — its posture, trajectory, and eventual fate — is treated as an outcome of this allocation under environmental pressure, shaped by cross-cutting mediators and constrained by scarcity.

The model's primary application domain is human systems where practices, frameworks, or diagnostics are built: personal effectiveness, project delivery, team dynamics, organizational design. It provides a structural foundation that can operate independently or integrate with other models and frameworks.

The model is theoretically general — its structural logic holds across biological, technical, socio-economic, and cosmological systems where scarcity, competing demands, and non-static behavior are present. However, its development originates from the practice of understanding and improving human dynamics in software delivery, and its applied focus remains there.

2.1.2 The Three Functional Contours

The three contours represent primary functional intents — not claims that every system activity has only one effect. A single action may produce consequences across multiple contours. The contours remain analytically distinct at the level of primary function.

Survival

Survival is the contour responsible for preserving the current viability and continuity of the system. Its functional role is to maintain existence, protect integrity, absorb disruption, and keep the system operational within tolerable bounds.

Reproduction

Reproduction is the contour responsible for copying, scaling, extending, or propagating the system's existing form, patterns, or outputs. Its functional role is to increase presence, reach, quantity, or continuity through multiplication of what already works.

Evolution

Evolution is the contour responsible for changing the system's form, behavior, structure, or capabilities. Its functional role is to adapt, learn, restructure, or transform — producing something the system was not previously capable of.

2.1.3 Resource and Scarcity

A resource is any limited input that can be allocated among competing system functions. Resources may be physical, temporal, energetic, informational, economic, social, institutional, or symbolic. The model does not require resources to be of a single type — only that scarcity and trade-offs exist at the relevant level of analysis.

Scarcity is the driver of the model. When resources are unlimited, allocation produces no trade-offs, contours do not compete, and the model has no explanatory value. The model becomes relevant precisely where allocation to one contour reduces what is available to others.

Allocation is the central observable. The model does not ask what a system intends or values — it asks where resources go. Among the resource types listed above, time has a specific role: it is the medium through which all allocation is enacted. Every act of allocation requires duration, and the pattern of allocation sustained over time produces cumulative effects — debt, potential, overhead — that shape the system's trajectory. The temporal dynamics of allocation are treated in System Metabolism.

2.1.4 Environment

The environment is the set of external conditions that shape resource availability, constraints, pressures, and system viability. The environment may be stable or volatile, permissive or hostile, abundant or scarce. It influences both what the system can do and what it must do to continue.

The environment is not static. Changes in environment alter the viability of existing allocation patterns and may force reallocation, whether the system chooses it or not.

2.1.5 Mediators

A mediator is any cross-cutting factor that shapes allocation, signaling, constraint, coordination, or feedback across the three contours — without itself being a fourth contour.

Mediators include factors such as governance, trust, legitimacy, power, information structure, and institutional rules. They are not peripheral — in complex human systems, mediators are often decisive. The model treats them as cross-cutting because they modify how the three contours operate rather than replacing the contours themselves.

The distinction matters: adding mediators as a fourth contour would break the model's allocation logic. Mediators do not compete for the same resource pool as the contours — they shape how that pool is divided.

2.1.6 Applicability

The model applies when three conditions hold:

- the system has limited resources,
- those resources face competing demands,
- and the system exhibits non-static behavior over time.

When all three conditions are present, the model can describe the system's allocation posture, diagnose distortion, and inform intervention. When any condition is absent — resources are effectively unlimited, demands do not compete, or the system is static — the model does not apply.

2.1.7 What the Model Does Not Do

The model is descriptive first, diagnostic second, and prescriptive only when context, boundaries, and mediating conditions are made explicit.

It does not assume moral value, political orientation, or a preferred outcome. No contour balance is inherently superior to another. A system that allocates heavily to Survival is not worse than one that allocates heavily to Evolution — it is different, and the consequences of that allocation are what the model describes.

The model is not a governance framework, a self-organization theory, or a complexity theory. It does not prescribe management structure, explain emergence, or model network behavior. It describes allocation dynamics and their consequences for system viability.

2.1.8 Theoretical Context

The model was developed independently, originating from applied work in software delivery and organizational dynamics. It was not derived from or built upon any existing theory.

Subsequent analysis identified structural correspondences with established theoretical traditions — most notably Life History Theory in biology, which models trade-offs between survival, reproduction, and growth under resource constraints, and General Systems Theory, which provides the foundational vocabulary of systems, boundaries, and environments. These correspondences serve as external validation of the model's structural logic, not as derivation.

The model is differentiated from viable system theories (Beer's VSM, Schwarz's VST) by its central question: where viable system theories ask how governance or self-organization should be structured, Tri-Contour asks where resources go and what happens as a result. Resource allocation is explicit and central, not implicit. Normativity is absent by design, not embedded.

2.2 System Structure

2.2.1 System Elements

The model defined in the previous chapter — three contours, limited resources, environment, mediators — describes *what* is being allocated and *why*. This chapter describes *what the system is made of* structurally: the elements through which contour dynamics operate.

The model operates through five elements arranged in three groups:

- **Substrate:** Code
- **Structure:** Boundary, Gate
- **Dynamics:** Signal, Receiver

Code is foundational — it is not at the same level as the other four elements. It is the substrate that determines how each of the other four operates in a given system instance. Boundary and Gate define the system's shape and permeability. Signal and Receiver carry and interpret information about system state.

These five elements are the infrastructure of contour allocation. Without them, the model can name contours and state that resources are allocated — but cannot explain how allocation is structurally enacted, how information about contour state travels, or why the same pressure produces different responses in different systems.

2.2.2 Substrate: Code

Code is the operating logic of a system. It defines what the system is, how it distinguishes itself from its environment, what it admits and rejects, what signals it can generate and interpret, and how it responds to pressure.

Code precedes and conditions all other elements:

- Boundary uses Code to distinguish inside from outside.
- Gate uses Code to determine what is permitted to cross.
- Signal is legible only within a shared Code context.
- Receiver is configured by Code to parse specific signal formats.

Code is not the same as narrative. Narrative is the expressed story — how Code is read, performed, and communicated by actors in a system. Code is the underlying logic being executed. The distinction matters: a system may change its narrative without changing its Code. When this occurs, the expressed story diverges from actual operating behavior — the system says one thing and does another. Durable system change requires change at the Code level, not only at the narrative level.

In human systems, Code manifests as shared beliefs about what the system is, what it exists to do, and what constitutes legitimate action within it. Code is not written down in any single document — it is distributed across the system's actors, practices, and institutional memory. It may be partially explicit (policy, charter, founding principles) and partially tacit (unwritten rules, inherited assumptions, cultural defaults).

Code is rewritable. The conditions and mechanisms of Code rewriting are defined in the dynamics layer of the model.

2.2.3 Structure: Boundary and Gate

Boundary

Boundary defines inside and outside — what belongs to the system and what does not. It uses Code as its reference: what Code defines as self is inside the Boundary; what Code does not recognize is outside.

Boundary is not necessarily physical or spatial. In human systems, Boundary may be organizational (who is a member), contractual (who is party to an agreement), informational (what data is internal), or identity-based (what the system considers part of itself).

Boundary determines the scope of contour allocation. Resources inside the Boundary are subject to the system's allocation logic. Resources outside the Boundary are part of the environment. Boundary persistence is not passive — it requires continuous resource expenditure. A Boundary that appears static is actively maintained, and that maintenance is itself an allocation cost, a dynamic treated in detail in System Metabolism.

Gate

Gate controls flow across Boundary. It governs what enters the system, what exits, and what is blocked — including resources, signals, actors, and information.

Gate is not binary. A Gate may be fully open, fully closed, or selectively permeable — admitting some flows while blocking others. Gate selectivity is configured by Code: what the system's operating logic defines as admissible passes through; what it defines as inadmissible is rejected.

Gate behavior shapes contour dynamics directly. A Gate that blocks signals about Evolution-relevant change prevents that information from reaching the system's allocation logic. A Gate that admits resources selectively may channel them toward one contour over others. Gate configuration is therefore a structural determinant of allocation posture — not merely a boundary mechanism.

2.2.4 Dynamics: Signal and Receiver

Signal

Signal transmits state-change information within the system or across its Boundary. A Signal carries information about contour state — distortion, pressure, opportunity, need — from one part of the system to another, or from the environment into the system.

A Signal is not inherently meaningful. Its meaning depends on the Code context within which it is received. The same Signal — the same information, in the same format — may produce a response in one system and no response in another, depending on whether the receiving system's Code makes that Signal legible.

Receiver

A Receiver is an actor or subsystem configured by Code to detect, parse, and respond to specific signals. A Receiver has three properties:

Threshold is the minimum signal strength required to trigger a response. Signals below Threshold are present but produce no action — the Receiver does not activate.

Legibility Range is the set of signal formats the Receiver can parse, as defined by Code. Signals outside the Legibility Range are not blocked by a Gate or suppressed by insufficient strength — they are simply not recognized. The Receiver has no category for them. The Signal lands and is inert.

Contour Orientation determines which contours the Receiver is primed to monitor. A Receiver oriented toward Survival detects signals relevant to system viability and continuity. A Receiver oriented toward Evolution detects signals relevant to change and adaptation. Contour Orientation shapes what the Receiver is looking for, which in turn shapes what information reaches the system's allocation logic.

Legibility

Legibility is not an element. It is a compatibility condition between Signal and Receiver — specifically, whether the Signal format falls within the Receiver's Legibility Range as defined by shared Code.

When a Signal is legible to a Receiver, it can be parsed and may produce a response (subject to Threshold). When a Signal is not legible, it produces no response — not because it was blocked, not because it was too weak, but because the Receiver cannot interpret it.

This distinction has diagnostic value. A system that fails to respond to a well-formed Signal may be doing so for three structurally different reasons: the Gate blocked the Signal (access failure), the Signal was below Threshold (strength failure), or the Receiver lacks the Legibility Range for that Signal format (interpretation failure). Each requires a different intervention.

2.2.5 Contour Tensions

The three contours compete for the same limited resource pool. This competition produces three structural tensions, each describing a trade-off between two contours:

Survival ↔ Evolution: stability versus change. Resources allocated to preserving current viability are not available for transformation. Resources allocated to transformation reduce the system's capacity to maintain its current state.

Survival ↔ Reproduction: integrity versus scale. Resources allocated to protecting system quality and coherence are not available for expanding quantity or reach. Resources allocated to scaling reduce the system's capacity to maintain the integrity of what it already does.

Reproduction ↔ Evolution: copying versus changing. Resources allocated to propagating the current form are not available for developing a new one. Resources allocated to developing new capabilities reduce the system's capacity to replicate what currently works.

These tensions are structural — they arise from scarcity, not from conflict between actors. Even a well-governed system with aligned actors experiences these tensions, because allocation to one contour necessarily reduces what is available to others.

Tensions are not problems to solve. They are permanent features of any system operating under the model's applicability conditions. The model does not prescribe how tensions should be resolved — it describes the consequences of different resolution patterns.

2.2.6 Contour Boundary Conditions

The three contours are analytically distinct at the level of primary functional intent. However, real-world activities frequently produce effects across multiple contours. Classifying an activity as primarily serving one contour requires identifying its primary intent — the functional demand it was undertaken to address.

Primary intent is the classification principle. An activity that scales existing output is Reproduction even if it incidentally improves process quality (a Survival effect). An activity that restructures a capability is Evolution even if it incidentally produces new output (a Reproduction effect).

Where primary intent is ambiguous, classification should be deferred until sufficient evidence is available. Premature classification produces interpretive drift. Detailed classification rules and discipline belong in the application layer of the model.

2.2.7 Mediator Architecture

Mediators, introduced in the previous chapter as cross-cutting factors that shape allocation, operate through the element infrastructure defined above.

Mediators shape allocation by operating on elements:

- A mediator may configure or reconfigure **Gates** — determining what flows are admitted or blocked. Governance structures, for instance, control which signals reach decision-makers and which resources are released.
- A mediator may modify **Signal** propagation — amplifying, attenuating, or distorting signals as they travel through the system. Trust, for instance, determines whether a signal is treated as credible or dismissed.
- A mediator may influence **Receiver** orientation — shaping which contours the system's actors are primed to monitor. Power structures, for instance, may orient the system's Receivers predominantly toward Survival signals, reducing sensitivity to Evolution signals.
- A mediator may, over time, contribute to **Code** change — altering the system's operating logic. Institutional rules, accumulated precedent, and legitimacy structures all participate in shaping what the system treats as its foundational logic.

Mediators do not allocate resources directly. They shape the conditions under which allocation occurs. This is why they are cross-cutting rather than a fourth contour: they operate on the allocation mechanism, not within it.

Mediator Configuration Properties

The preceding section defines what mediators do — operate on elements to shape allocation. This section defines how mediators differ structurally from one another. Two mediators may operate on the same element (both configuring Gates, for instance) and yet produce different allocation outcomes because their structural configuration differs. The configuration properties defined here determine the character of a mediator's effect, not merely its presence or absence.

Three configuration properties are defined.

Selectivity describes the pattern by which a mediator differentiates what it admits, amplifies, or blocks. Selectivity is not a scalar (more or less selective). It has a shape — the structural pattern that determines which flows, signals, or actors are treated differently.

A mediator with sharp selectivity draws clear distinctions: what passes the mediator's criteria is fully admitted, what does not is fully blocked, and the boundary between the two is narrow. Sharp selectivity produces concentrated, hierarchical allocation outcomes — resources flow into well-defined channels with clear jurisdictional boundaries. In human systems, a governance structure with many nested approval layers, strict jurisdictional boundaries, and binary pass/fail criteria exhibits sharp selectivity.

A mediator with smooth selectivity draws gradual distinctions: flows experience varying degrees of facilitation or resistance rather than binary admission or rejection, and the transition between facilitated and resisted is wide. Smooth selectivity produces distributed, diffuse allocation outcomes — resources flow along broad channels without sharp concentration points. In human systems, a governance structure with broad guidelines, few checkpoints, and gradient-based criteria exhibits smooth selectivity.

The distinction between sharp and smooth selectivity is not a value judgment. Sharp selectivity produces efficient local concentration — resources reach their destination with minimal diffusion. Smooth selectivity produces broad distribution — resources reach more destinations but with less concentration at any one. Which configuration is appropriate depends on the system's environmental conditions and contour demands, not on an inherent superiority of one pattern over the other.

Resolution describes the minimum scale at which a mediator operates. Every mediator has a characteristic scale below which it does not differentiate — flows, signals, or actors below this scale pass through the mediator's influence without being shaped by it.

Resolution defines a floor. Above the floor, the mediator's selectivity pattern applies — flows are admitted, blocked, amplified, or attenuated according to the mediator's configuration. Below the floor, the mediator is structurally absent — not because it has been removed, but because the flows at that scale are too fine-grained for the mediator's configuration to act on.

In human systems, resolution manifests as the granularity at which governance, trust, or power structures operate. An executive governance structure may have high resolution at the division level (shaping allocation across major units) and no resolution at the team level (individual team decisions pass below the governance floor). Trust may operate at high resolution between named individuals (shaping specific signal credibility) and low resolution at the institutional level (general institutional trust applying uniformly). The mediator exists at both scales, but its configuration only differentiates at one.

Resolution has diagnostic value. When a mediator appears ineffective — when it does not shape allocation as expected — the cause may be that the relevant flows operate below the mediator's resolution floor. The mediator is present but the activity is too fine-grained for it to act on. This is structurally distinct from a mediator being absent (not present at all) or blocked (present but overridden by a competing mediator). Each requires a different intervention: absence requires introducing the mediator, blocking requires addressing the competing mediator, and resolution mismatch requires either raising the mediator's resolution or aggregating the fine-grained flows to a scale the mediator can act on.

Temporal behavior describes whether a mediator's configuration is static or dynamic over the time horizons relevant to the system's metabolism.

A mediator with static temporal behavior maintains a constant configuration within the relevant time horizon. Its selectivity pattern and resolution do not change as the system operates. The Gate configurations it produces are stable — flows experience the same mediator influence at the beginning and end of the observation period.

A mediator with dynamic temporal behavior changes its configuration within the relevant time horizon. Its selectivity pattern, resolution, or both shift as the system operates. The Gate configurations it produces are time-dependent — flows experience different mediator influence at different points in the system's metabolic cycle.

Dynamic temporal behavior may be periodic (the configuration oscillates between states at a characteristic frequency), monotonic (the configuration shifts in one direction over time), or state-dependent (the configuration changes in response to specific system conditions). Each pattern produces different allocation consequences. Periodic mediator behavior produces rhythmic, pulsatile allocation patterns — resources flow differently at different phases of the cycle. Monotonic mediator behavior produces trending allocation patterns — the system's allocation posture drifts as the mediator reconfigures. State-dependent mediator behavior produces conditional allocation patterns — the system operates under one regime until a condition is met, then shifts to another.

The distinction between static and dynamic mediators is relative to the observation time horizon. A mediator that appears static at quarterly resolution may be dynamic at daily resolution. This is an instance of the Metabolic Frame Dependence defined in System Metabolism — the observed temporal behavior of a mediator depends on the temporal resolution of the observer's frame. Declaring a mediator's temporal behavior requires declaring the time horizon within which that behavior is assessed.

Configuration and Allocation Consequences

The three configuration properties interact to produce the mediator's total effect on allocation. A mediator with sharp selectivity, high resolution, and static temporal behavior produces concentrated, fine-grained, stable allocation channeling. A mediator with smooth selectivity, low resolution, and dynamic temporal behavior produces distributed, coarse-grained, shifting allocation patterns.

Two systems identical in their contour demands, resource pools, and element infrastructure but different in mediator configuration will produce different allocation outcomes. This is the structural claim: mediator configuration is a determinant of allocation posture, independent of contour demand and resource availability. When diagnosis identifies allocation distortion, the cause may lie not in the contours or resources but in the mediator configuration that shapes how resources flow between them.

The specific configurations that produce specific allocation consequences are domain-dependent. Identifying which selectivity pattern, resolution level, and temporal behavior are operative in a given system is part of the diagnostic process and should be declared explicitly, following the same discipline required for proxy selection and comparability conditions.

2.2.8 Environment–System Interface

The environment, defined in the previous chapter as external conditions shaping the system, interacts with the system through Boundary and Gate.

Environmental pressure reaches the system's allocation logic through a structural path: pressure changes conditions at the Boundary → Gate admits or blocks the resulting signals and resource flows → admitted signals reach Receivers → Receivers parse signals according to Code and contour orientation → parsed signals inform allocation decisions.

This path is not instantaneous or lossless. At each stage, information may be delayed, filtered, distorted, or blocked. Gate selectivity may prevent environmental signals from entering. Receiver Legibility Range may prevent entered signals from being parsed. Code may define certain environmental conditions as irrelevant.

The consequence is that a system's response to environmental change is not determined by the change itself — it is determined by how much of that change survives the path from Boundary to allocation decision. A system with open Gates, broad Receiver Legibility, and adaptive Code will respond to environmental shifts that a system with closed Gates, narrow Legibility, and rigid Code will not detect.

2.3 System Dynamics

The previous chapters defined what the model describes and what the system is made of. This chapter defines what happens when the system operates — how allocation behaves under pressure, and how the system's own structure changes over time.

System Dynamics are divided into two categories:

Allocation Dynamics govern how resources move between contours under environmental pressure and internal tension. These mechanisms operate within the system's existing structure — they do not change what the system is, they change where resources go.

Structural Dynamics govern how the system's own architecture changes — its Code, its Boundaries, its Gate configuration. These mechanisms alter the infrastructure through which allocation operates. When structural dynamics succeed, the system that emerges is not the same system with different allocation — it is a different system.

The distinction matters because the two categories require different interventions. Allocation problems are addressed by changing resource distribution. Structural problems are addressed by changing the system's operating logic, boundaries, or permeability. Misidentifying which category a problem belongs to produces interventions that fail for structural reasons.

Each mechanism defined below is introduced at the level necessary to establish its role in the model. Detailed treatment of individual mechanisms — typologies, formal properties, failure modes — belongs in dedicated companion documents.

2.3.1 Allocation Dynamics

Allocation dynamics describe the mechanisms through which contour balance shifts, distortion is masked, and allocation decisions are reinforced or corrected over time. Six mechanisms are defined.

Displacement

Displacement is the mechanism by which one contour's allocation is reduced to serve another under pressure.

When resource availability declines relative to demand — or when demand on one contour intensifies — the system cannot maintain its current allocation across all three contours. Something must receive less. Displacement is what determines which contour loses resources and which gains them.

Displacement is triggered by the gap between available resources and required allocation, not by absolute scarcity. A system with abundant resources can experience displacement if demands grow faster than supply. A system with scarce resources may avoid displacement if demands are proportionally modest.

Displacement order — which contour is reduced first — is context-dependent, not universal. The order depends on what the system's Code treats as most expendable under pressure. In most human systems, Evolution is displaced first because its returns are distant and uncertain, and its absence produces no immediate consequence. Reproduction is typically displaced second, and Survival last — because Survival failure threatens the system's existence directly. However, a system whose Code prioritizes transformation over scale would displace Reproduction before Evolution. Displacement order is a property of the specific system's Code, not a universal law.

Displacement is gradual by default. Allocation shifts incrementally as pressure builds. However, displacement can appear sudden to observers when the system compensates — masking the ongoing shift through buffer expenditure. In such cases, displacement has been occurring invisibly. When compensation margin is exhausted, the accumulated displacement becomes visible all at once. The displacement was gradual; its appearance was sudden. This connection between Displacement and Compensation is one of the model's central explanatory claims.

Displacement produces contour distortion — a state where allocation deviates from the system's prior balance. Distortion is not inherently pathological. A system may displace deliberately, accepting reduced allocation to one contour in order to address a temporary pressure on another. Distortion becomes problematic when it persists without a restoration trajectory, or when the system is unaware that displacement is occurring.

Compensation

Compensation is the mechanism by which a system maintains observable viability under contour distortion through the expenditure of a finite buffer that is not part of the system's primary allocation logic.

Compensation arises when three conditions hold simultaneously: allocation between contours has deviated from viable balance, the system does not directly correct the deviation, and the gap between expected and actual contour output is absorbed by a buffer.

Compensation has a three-element structure. A **trigger** — an observable gap between expected and actual contour output. A **buffer** — a finite resource or capacity that absorbs the gap. A **signal** — evidence that buffer expenditure is occurring.

Compensation can be classified by intent. **Adaptive Compensation** is intentional, bounded, and accompanied by a plan to restore balance — the system knows it is compensating and when it will stop. **Structural Compensation** is unintentional, unbounded, and proceeds without a restoration plan — the system compensates because it has no alternative.

Compensation is not inherently pathological. It is an adaptive mechanism with a finite operational horizon. It becomes pathological when buffer expenditure persists without correction, consuming capacity that cannot be replenished.

Without Compensation, the model can describe distortion but predicts that distortion produces visible consequences immediately. This does not match observed system behavior. Compensation explains the temporal gap between the onset of contour distortion and the emergence of observable failure — why systems hold longer than their allocation profile predicts, and why failure often appears sudden to external observers.

The temporal dynamics of compensation — how buffers accumulate, how they deplete under sustained load, what curve shapes govern their exhaustion, and how different buffer types produce different temporal profiles — are formalized in System Metabolism. Compensation as defined here is the mechanism; its behavior over time is an accumulation dynamic.

Detailed treatment — buffer typology, compensation margin, dynamics, failure modes, and formal propositions — belongs in the dedicated Compensation chapter.

Contour Saturation

Contour Saturation is the state where one contour's allocation reaches the system's physical limit, eliminating the allocation space for the other two contours entirely.

Saturation is structurally distinct from dominance. Under dominance, one contour receives a disproportionate share of resources while the others receive reduced but nonzero allocation. The system is imbalanced but retains degrees of freedom — reallocation remains structurally possible. Under Saturation, the dominant contour has consumed the entire allocation space. The other contours receive not merely less, but nothing — their allocation is not reduced, it is structurally impossible. The system has no remaining degrees of freedom within which to reallocate.

Saturation is the terminal state of displacement. It is what displacement becomes when the pressure driving it is absolute — when the demand on one contour is not merely greater than the others, but has absorbed the system's full resource capacity. In this sense, Saturation is not a separate mechanism from displacement but the boundary condition that displacement reaches under extreme pressure.

Three conditions define Saturation:

Allocation collapse — the total available resource is bound to a single contour. This is not a matter of priority or preference — it is a physical state in which the system's resources are structurally committed and unavailable for redistribution.

Demand persistence on displaced contours — the functional demands that Reproduction and Evolution represent do not disappear because their allocation has been eliminated. They persist as unmet structural requirements. The gap between zero allocation and nonzero demand is the maximum displacement the model can describe.

Compensation impossibility — no buffer exists that can absorb the gap between zero allocation and the persistent demand on the displaced contours. Compensation requires a finite resource outside the primary allocation logic. At Saturation, no such resource remains — the saturating contour has absorbed everything that could serve as a buffer.

The consequence of Saturation is that the system's allocation logic ceases to function. Allocation logic operates by distributing resources among competing demands — it requires a space of possible distributions. When that space collapses to a single point (all resources to one contour), the logic has no decisions left to make. The system is not making a bad allocation — it is no longer allocating at all.

Saturation produces a characteristic exit dynamic. Because the displaced contours' demands persist at nonzero levels while their allocation is zero, the displacement pressure does not resolve — it accumulates without bound. No compensation buffer can absorb it. No feedback mechanism can redirect it, because there are no resources to redirect. The system reaches a state where the internal pressure from unmet demand exceeds the structural capacity of the system to contain it.

The exit from Saturation is not reallocation — reallocation requires degrees of freedom that Saturation has eliminated. The exit is system reconstitution: the current system's Boundary fails, its resources disperse, and a new system forms under different allocation conditions.

Reconstitution dynamics — what persists across this transition, what resets, and what determines the new system's initial allocation — are defined in System Course.

Saturation is rare in human systems because resource pools are diverse enough and boundaries permeable enough that total allocation collapse is difficult to achieve. A human organization can always, in principle, reallocate some fraction of attention, time, or effort — the allocation space does not fully collapse. Saturation is more characteristic of physical systems where resource pools are singular and conservation laws are absolute — where the entire available resource can, under sufficient pressure, be bound to a single functional demand with no remainder.

The model's theoretical generality claim — that its structural logic holds across systems where scarcity, competing demands, and non-static behavior are present — requires that it can describe what happens at the boundary conditions of its own logic. Saturation is that description. It is the state where the model's central mechanism (allocation among competing contours) reaches its limit, and the model must hand off to a different dynamic (reconstitution) to describe what follows.

Feedback

Feedback is the mechanism by which allocation outcomes modify future allocation decisions.

Every allocation decision produces consequences — changes in contour output, environmental response, signal generation. Those consequences travel back through the system's element infrastructure — as Signals reaching Receivers — and inform subsequent allocation. Feedback is this return path: from outcome back to decision.

Feedback operates in two modes.

Reinforcing feedback amplifies the current allocation direction. A system that displaces Evolution and experiences short-term stability receives a signal that the displacement was viable — reinforcing further displacement. Reinforcing feedback accelerates existing trends. It can accelerate beneficial trends (a system investing in Evolution that receives positive environmental response) or destructive ones (a system depleting its Survival resources while Reproduction metrics remain strong).

Balancing feedback counteracts the current allocation direction. A system that displaces Evolution and subsequently loses adaptive capacity receives a signal that the displacement has consequences — creating pressure to restore Evolution allocation. Balancing feedback dampens oscillation and promotes return toward a prior balance.

Whether feedback reaches the system's allocation logic depends on the element infrastructure defined in the previous chapter. Feedback signals must cross Gates, match Receiver Legibility Range, exceed Receiver Threshold, and be parsed within the system's Code context. A system may generate feedback signals that never reach its own allocation logic — because Gates block them, because Receivers are not configured to parse them, or because Code defines them as irrelevant.

This structural dependency explains why some systems self-correct and others do not. Self-correction requires not only that balancing feedback exists, but that the system's element infrastructure permits it to reach allocation decisions. A system with functioning feedback loops and open signal paths is structurally capable of self-correction. A system with blocked or distorted feedback paths is not — regardless of whether balancing signals are being generated.

The most consequential reinforcing feedback pattern in the model — the loop between accumulating Evolution debt and increasing Survival overhead, in which capability degradation drives higher maintenance effort which further reduces Evolution investment — is formalized as an accumulation interaction in System Metabolism. Feedback as defined here is the mechanism; its sustained operation over time produces the accumulation dynamics that determine system trajectory.

Endogenous Phase Transition

Endogenous Phase Transition is displacement triggered not by external pressure or environmental change, but by the system's internal state crossing a critical threshold at which the current allocation pattern becomes unstable.

Displacement, as defined above, is triggered by a gap between available resources and required allocation — typically produced by environmental change, demand intensification, or resource reduction. These are exogenous triggers: something outside the system's current allocation pattern changes, and the system responds.

Endogenous Phase Transition describes a different causal path. The system's environment may be stable. Demands may be constant. Resources may be unchanged. But the system's internal accumulation — debt, potential, overhead, or structural complexity — reaches a threshold at which the current allocation pattern can no longer sustain itself. The transition is triggered by what has accumulated inside, not by what has changed outside.

Three conditions define an Endogenous Phase Transition:

Accumulation threshold — a quantitative change in the system's internal state has reached a critical level. The threshold is not a single value but a boundary in the system's state space beyond which the current allocation regime is no longer self-sustaining. Below the threshold, the system's allocation pattern is stable under its current conditions. Above it, the same pattern becomes structurally unstable — small perturbations that would previously have been absorbed now trigger cascading reallocation.

Absence of external trigger — the transition is not caused by environmental change, demand shift, or resource shock. These may coincide with the transition — and when they do, the external event is typically credited as the cause. But the structural claim is that the transition would have occurred without the external event, because the internal state had already crossed the threshold. The external event, if present, is a catalyst for a transition that was already structurally inevitable.

Discontinuous allocation shift — the resulting displacement is not gradual. It is a rapid, qualitative change in allocation pattern — a shift from one allocation regime to a different one. The system does not smoothly adjust; it transitions. Before the threshold, incremental internal changes produce no visible allocation effect. After the threshold, a small additional increment produces a large allocation shift. The transition is discontinuous in the same sense that a phase change in physics is discontinuous — the underlying variable (temperature, accumulation) changes continuously, but the system's macro-state (liquid/gas, allocation regime) changes abruptly.

Endogenous Phase Transition has diagnostic value because it explains a class of system failures and transformations that appear to have no external cause. When a system that has been stable under constant conditions suddenly shifts its allocation pattern, the standard diagnostic — what changed in the environment? — may find nothing. The change was internal. Accumulated debt crossed a self-sustaining threshold. Accumulated complexity exceeded coordination capacity. Accumulated overhead reached a regime boundary. The system transitioned because its own metabolic state made the prior regime untenable, not because anything pushed it.

The relationship to other mechanisms is specific. Displacement describes what happens (allocation shifts between contours). Compensation may mask the accumulation that is approaching the threshold. Feedback may accelerate or delay the approach. The Endogenous Phase Transition describes the trigger condition — the moment when accumulated internal state converts from a quantitative change to a qualitative one.

The temporal dynamics of Endogenous Phase Transitions — what accumulation curves produce them, what threshold shapes are characteristic, and how the transition point can be estimated — are treated in System Metabolism. The mechanism is defined here; its temporal behavior is an accumulation dynamic.

Mediator Dynamics

Mediator dynamics describe how mediators — the cross-cutting factors defined in the model and architecturally positioned in the previous chapter — actively shape allocation mechanisms in motion.

Mediators do not merely exist as structural features. They participate in allocation dynamics by amplifying, blocking, or redirecting the other five mechanisms.

A mediator may **amplify** a mechanism — accelerating displacement, extending compensation, strengthening feedback, lowering the threshold for Endogenous Phase Transition, or hastening the approach to Contour Saturation. Trust, for instance, amplifies feedback: in a high-trust system, signals from actors experiencing displacement are more likely to be received, parsed, and acted upon. In a low-trust system, the same signals are discounted or ignored.

A mediator may **block** a mechanism — preventing displacement from reaching a contour, shielding a buffer from exhaustion signals, or interrupting a feedback loop. Power structures, for instance, may block feedback that would threaten the current allocation pattern — protecting the positions of actors who benefit from the existing distribution. A mediator may also raise the threshold for Endogenous Phase Transition — institutional resilience, for instance, may allow higher internal accumulation before the allocation regime becomes unstable.

A mediator may **redirect** a mechanism — changing which contour is displaced, which buffer absorbs distortion, or which feedback signal reaches the allocation logic. Governance, for instance, may redirect displacement from one contour to another by imposing allocation rules that override what the system's actors would choose under pressure.

Mediators can conflict. When two or more mediators push in opposing directions — one amplifying Evolution feedback while another blocks it — the system experiences mediator tension. Mediator tension is distinct from contour tension: contour tension arises from resource scarcity, mediator tension arises from competing cross-cutting forces shaping how that scarcity is managed.

The consequences of mediator dynamics depend on mediator strength, persistence, and alignment. Aligned mediators produce coherent allocation behavior. Conflicting mediators produce erratic or contradictory allocation — the system appears to pursue incompatible strategies simultaneously. Detailed treatment of mediator interaction patterns belongs in a dedicated chapter.

Mediator Reciprocity

The preceding section describes mediators acting on mechanisms — amplifying, blocking, or redirecting displacement, compensation, feedback, and the other allocation dynamics. The relationship is not one-directional. Allocation mechanisms act back on mediators, modifying their configuration properties over time.

Mediator Reciprocity is the structural claim that sustained allocation patterns alter the mediators that shape them. A mediator that shapes allocation is itself shaped by the allocation outcomes it produces. The relationship is bidirectional: mediator configuration → allocation pattern → mediator reconfiguration.

Reciprocity operates through the mediator's configuration properties defined in System Structure — selectivity, resolution, and temporal behavior. Sustained allocation outcomes modify these properties, not the mediator's existence. The mediator persists, but its structural character changes.

Three reciprocal pathways are defined.

Allocation-driven selectivity shift. Sustained displacement in one direction alters the mediator's selectivity pattern over time. A mediator that initially shapes allocation with smooth selectivity may sharpen under sustained Survival-dominant distortion — as the system's allocation logic increasingly treats resource flows as binary (critical/non-critical), the mediator's discrimination pattern narrows to match. Conversely, a mediator operating with sharp selectivity under balanced allocation may soften if the system enters a period of exploratory Evolution investment — the governance structures that previously enforced binary approval may gradually shift toward continuous influence as the system's Code incorporates experimentation as legitimate.

The mechanism is Code-mediated. The mediator's selectivity is configured by Code (as defined in System Structure). Sustained allocation patterns produce Code drift (as defined in Structural Dynamics). Code drift modifies the configuration under which the mediator operates. The mediator's selectivity shifts not because the mediator itself "decides" to change, but because the Code substrate it operates on has drifted.

Allocation-driven resolution shift. Sustained allocation patterns may alter the scale at which a mediator differentiates. Under sustained overhead ratchet (defined in System Metabolism), governance mediators often lose fine resolution — the elevated Survival allocation produces reporting, approval, and compliance structures that operate at aggregate scale, while individual-level or team-level resource flows become unmediated. The mediator's resolution coarsens as a consequence of the allocation pattern it is embedded in.

The reverse also occurs. A system that sustains balanced allocation with active Evolution investment may develop finer mediator resolution over time — trust, for instance, may differentiate at increasingly specific levels as actors accumulate shared experience across diverse contexts.

Allocation-driven temporal shift. The temporal behavior of a mediator — whether it is static or dynamic — may change as a consequence of sustained allocation patterns. A mediator that was static under balanced allocation may become dynamic under sustained distortion. Trust, for instance, may oscillate under chronic Survival-dominant allocation — periodically restored by crisis-response solidarity, then eroded by the ongoing displacement of Evolution and Reproduction investment. The oscillation is not inherent to trust as a mediator — it is produced by the allocation pattern that trust is embedded in.

Reciprocity produces a characteristic diagnostic challenge. When a system's allocation behavior changes, the change may originate in the allocation dynamics (a new displacement pressure, a compensation buffer exhausting, a feedback loop activating) or in the mediator configuration (selectivity shifting, resolution coarsening, temporal behavior changing). Because the two are reciprocally linked, the causal direction may be ambiguous. Diagnostic discipline requires identifying whether the allocation change preceded and caused the mediator reconfiguration, or whether the mediator reconfiguration preceded and caused the allocation change. The temporal sequence is the primary diagnostic tool: what changed first determines the causal direction. When both appear to change simultaneously, the system is likely in an active reciprocal loop — the allocation pattern is modifying the mediator that is modifying the allocation pattern. Intervening on either side of the loop can arrest it, but the choice of intervention target determines what changes and what persists.

Reciprocity is the structural mechanism that explains why mediators in long-lived systems rarely maintain their original configuration. Governance, trust, power, legitimacy — all are shaped by the allocation history of the system they operate within. A mediator's current configuration is a product of both its initial design and the cumulative allocation patterns it has experienced. Diagnosing a mediator's configuration without accounting for its allocation history produces a snapshot that misses the trajectory.

2.3.2 Structural Dynamics

Structural dynamics describe mechanisms that change the system itself — its Code, its Boundaries, its Gate configuration. Unlike allocation dynamics, which operate within existing structure, structural dynamics alter the infrastructure through which allocation operates.

Structural change is distinct from allocation change. A system that shifts resources from Survival to Evolution has changed its allocation. A system that rewrites its Code to redefine what constitutes Survival has changed its structure. The first produces a different posture within the same system. The second produces a different system.

Code Rewrite

Code rewrite is the mechanism by which a system's operating logic changes — deliberately or through emergent drift.

Code can change through two paths. **Drift** is gradual, unintentional change accumulating through repeated small departures from existing Code — each individually insignificant, collectively transformative. Drift is the default path. Most systems undergo Code change through drift rather than deliberate intervention. **Deliberate rewrite** is intentional change targeting specific elements of the system's operating logic. Deliberate rewrite is rarer and more difficult than drift, because the system's existing structure tends to resist it.

Deliberate Code rewrite requires four conditions to succeed:

Targeting — the actor or process attempting the rewrite must identify which element of Code is to be changed. Without precise targeting, change activity occurs without contacting the operating logic — the system experiences disruption but its Code remains intact.

Vector — the rewrite must have a delivery mechanism capable of reaching the Code. In human systems, vectors include trusted actors whose signals cross Survival-contour Gates, crises that bypass normal gatekeeping, or external events that carry new logic inside the system's Boundary.

Suppression — the system's resistance to Code change must be sufficiently reduced for the rewrite to take hold. Systems actively resist Code change — existing Code configures Gates and Receivers to reject signals that threaten it. Successful rewrite requires that this resistance is temporarily weakened, whether through legitimacy of the change agent, severity of crisis, or exhaustion of existing Code's credibility.

Expression — the rewritten Code must be active in the system's operating context. A Code change that is formally adopted but never enacted — the system says the new thing but runs the old logic — has not achieved expression. Expression requires that the system's Receivers, Gates, and Boundaries actually operate according to the new Code, not merely acknowledge it.

Failure at any condition produces a characteristic outcome. Without targeting: activity without structural contact. Without vector: the rewrite never reaches inside the Boundary. Without suppression: the rewrite is rejected and the change agent may be expelled. Without expression: the rewrite is nominal — new narrative, old Code.

Code–Narrative Divergence

Code and Narrative may diverge. When they do, the system's expressed story — what it says about itself, what it communicates to its environment — no longer reflects its operating logic.

Divergence is not rare. It is a common system state, particularly in human organizations where narrative is more easily changed than Code. A system may adopt new values, publish new principles, or announce new priorities — all narrative changes — without its Code changing. The system's actual allocation behavior, Gate configuration, and Receiver orientation remain unchanged. The narrative is new; the behavior is old.

Divergence has diagnostic value. A system whose narrative and Code are aligned produces behavior consistent with its stated intent. A system whose narrative and Code have diverged produces behavior that contradicts its stated intent — and the contradiction is often visible to actors inside the system before it is visible to observers outside.

Divergence is not inherently pathological. A system may deliberately adopt narrative that leads its Code — stating aspirational intent before the operating logic has caught up. This is coherent if the system is actively pursuing Code rewrite toward the stated narrative. It becomes pathological when the divergence is permanent and unacknowledged — the system sustains a false story about itself without any trajectory toward alignment.

The temporal dynamics of divergence — why it occurs so readily and why it persists — have a structural explanation. Narrative operates at Signal speed: it can propagate through the system nearly instantaneously. Code changes at metabolic rates: constrained by the actual transformation of operating logic, practices, and institutional behavior. The speed difference between Narrative and Code is not incidental — it is a consequence of their different relationships to time. This temporal relationship, and its diagnostic implications, is treated in System Metabolism.

Boundary Dissolution

Boundary dissolution is a system state in which environmental stress exceeds Boundary enforcement capacity, temporarily suspending normal Gate control.

Under normal conditions, Boundary and Gate regulate what enters and exits the system. Gate selectivity — configured by Code — determines which signals, resources, and actors are admitted. This selectivity is part of the system's structural integrity.

Under sufficient stress, this enforcement breaks down. Gates that normally block certain signals or actors become permeable. Flows that would normally be filtered or rejected enter the system without passing through the usual selection process. The system's interior becomes temporarily accessible to influences that existing Code would normally exclude.

Boundary dissolution is not failure. It is a transient state with a characteristic window. During the window, structural change that would otherwise be blocked becomes possible — including Code rewrite that could not satisfy the suppression condition under normal enforcement. The window is brief. When stress subsides or the system adapts, enforcement reconstitutes — often more aggressively than before, as the system's Code incorporates the dissolution experience into its threat model.

Boundary dissolution is triggered by systemic stress that exceeds the system's enforcement capacity — not by local pressure or incremental change. In human systems, triggers include existential competitive threat, organizational crisis, leadership collapse, or external shocks that invalidate the assumptions underlying current Gate configuration.

The outcome of boundary dissolution depends on what enters during the window. If actors carrying viable Code alternatives gain access, the system may undergo productive structural change. If no coherent alternative is available, dissolution produces disruption without transformation — the system restores its prior Code with additional defensive reinforcement.

2.4 System Metabolism

The previous chapters defined what the system is made of and how it behaves under pressure — how allocation shifts, how structure changes, how mechanisms operate. This chapter defines what happens when those mechanisms operate over time — how allocation produces accumulation, how accumulation produces trajectory, and how the temporal behavior of contour allocation becomes the primary observable for diagnosing system state.

System Dynamics describes mechanisms — what can happen. System Course describes trajectory — where the system goes. This chapter occupies the space between them: what accumulates, at what rate, and with what shape. Without this layer, the model can describe mechanisms and name trajectories but cannot explain why the same mechanism produces different trajectories in different systems, or why systems with identical structures at a single point in time produce divergent outcomes over time.

2.4.1 Time

Time is not a model element. It is a universal constraint that the model inherits from physical reality.

Every metabolic act — every transformation of resources across or within contours — requires nonzero duration. A signal cannot propagate instantaneously. A gate cannot open and close without interval. A boundary cannot be maintained without continuous expenditure. No element of the model operates outside of time. This constraint is not a theoretical claim — it is a physical fact that applies to biological, organizational, and social systems alike.

The model does not define time. It uses time in a specific way: contour allocation over duration is the measurable substrate of system metabolism. Two systems identical in structure and dynamics but different in their allocation history over time are different systems with different predicted trajectories. The temporal record of allocation is not supplementary — it is constitutive of system state.

Time operates in the model at two levels simultaneously. Continuous time is the background within which all system activity occurs. It flows regardless of whether anything changes. A system sitting in unchanged allocation for a year has experienced a year of continuous time — and that duration has consequences, because the environment has changed even if the system has not. Event-driven time marks the discrete moments when metabolic state changes — a contour allocation shifts, a conversion occurs, a synchronization event happens. Continuous time answers when something occurs or how long a state persists. Event-driven time answers what changed and in what sequence.

Both levels are always present. Accumulation dynamics — debt, potential, overhead — operate in continuous time. They grow or decay whether or not discrete events occur. Conversion dynamics — the transition from accumulated potential to productive output — are event-driven. They happen at specific moments, triggered by specific conditions.

Time and the Element Infrastructure

All elements defined in System Structure operate within time, but their relationship to time differs.

Code is the element most directly subject to temporal dynamics. Code accumulates — drift adds incremental change over time. Code degrades — operating logic that was viable under prior conditions becomes progressively less viable as conditions change. Code transforms — rewrite produces discontinuous change at specific moments. The temporal state of Code — what has accumulated, what has degraded, what has been rewritten and when — constitutes the system's metabolic record.

Narrative, defined in System Dynamics as the expressed story of Code, operates at a different temporal speed. Narrative can change nearly instantaneously — a new strategy announcement, a revised mission statement, a leadership speech. Code changes at metabolic rates — constrained by the actual transformation of operating logic, practices, and institutional behavior. The gap between Narrative speed and Code speed is itself a diagnostic indicator. When the gap is large — Narrative has changed but Code has not — the system is producing divergence, not transformation. When Narrative and Code change at comparable rates, transformation is genuine.

Boundary requires continuous maintenance — it persists only because the system expends resources to sustain it. This maintenance is a metabolic cost. A boundary that appears static is actively sustained; the time invested in that sustenance is Survival-contour allocation.

Time and Contour Behavior

The three contours impose distinct temporal properties on any activity they govern. This is not a claim that activities inherently have fixed temporal properties — it is a claim that the contour an activity serves shapes its temporal behavior.

Survival imposes urgency and non-deferrability. An activity serving Survival operates under the shortest tolerable cycle — production incidents are resolved immediately, payroll is processed without delay, infrastructure is maintained before failure. Survival-contour time is characterized by the inability to postpone without immediate consequence.

Reproduction imposes cyclicality and predictable throughput. An activity serving Reproduction operates within a recognizable rhythm — delivery cadences, release cycles, hiring rounds, planning periods. The period scales with the complexity of what is being reproduced, but the cycle shape is consistent.

Evolution imposes latency and stochastic return. An activity serving Evolution operates without a guaranteed timeline or deterministic outcome — learning accumulates over unpredictable durations, experiments produce results that may or may not convert to capability, and the interval between investment and return is characteristically longer than for the other two contours.

These temporal properties have diagnostic value. An activity's actual contour can be identified by observing its temporal behavior, independent of how it is labeled. A "learning initiative" that operates on tight deadlines with predictable deliverables exhibits Reproduction-contour temporal properties — it is Reproduction, regardless of its name. A code review that proceeds without time pressure and explores unfamiliar architectural patterns exhibits Evolution-contour temporal properties — it is Evolution investment, regardless of whether it appears on a delivery timeline.

Metabolic Frame Dependence

The preceding sections establish time as the medium through which allocation is enacted and accumulation occurs. They treat time as uniform — a shared background within which all observers and all parts of the system experience the same duration. This is a sufficient assumption for most applications of the model. It is not a universal one.

Metabolic Frame Dependence is the condition where the observed metabolic rate of a system varies depending on the observer's structural relationship to the system. The same system, undergoing the same allocation dynamics, presents different apparent metabolic rates to observers positioned differently relative to its structure.

Frame dependence arises from two sources.

Structural position determines what metabolic activity is visible to an observer. An observer positioned at the system's Boundary sees flows crossing in and out — resource acquisition, output delivery, environmental signal exchange. An observer positioned at the system's allocation logic sees contour trade-offs, displacement decisions, feedback processing. An observer positioned at a single element — one Gate, one Receiver, one subsystem — sees the metabolic activity that passes through that element. Each position produces a different apparent metabolic rate, because each position makes different subsets of the system's total metabolic activity observable.

This is not a measurement limitation that better instrumentation could resolve. It is a structural property of observation. A system's total metabolic activity is distributed across its elements, and no single observation point captures all of it. The metabolic rate measured from any given position is a real measurement of a real subset of the system's metabolism — but it is a subset, not the whole.

Temporal resolution determines what metabolic patterns are visible to an observer. An observer sampling system state at quarterly intervals sees a different metabolic signature than an observer sampling daily. The quarterly observer detects slow accumulation dynamics — overhead ratcheting, debt growth, potential decay — but cannot resolve the event-driven dynamics that occur between samples. The daily observer detects operational allocation patterns — Survival urgency cycles, Reproduction throughput rhythms — but may miss the slow accumulation trends that are only visible at longer intervals. Both observers are measuring real metabolic activity. Neither has the complete picture.

The combination of structural position and temporal resolution defines an observer's metabolic frame. The same system observed from different frames presents different apparent metabolic states — and both observations are accurate within their frame.

Frame dependence has diagnostic consequences. When two observers of the same system disagree about its metabolic state — one perceiving health while the other perceives stress — the disagreement may not indicate that one observer is wrong. It may indicate that each observer's frame reveals different aspects of the system's metabolism. A board observing quarterly financial metrics may perceive Flow. An engineering team observing daily delivery dynamics may perceive Starvation. Both are correct within their frame. The system's actual metabolic state includes both observations — the financial stability seen at quarterly resolution and the capability erosion seen at daily resolution.

The model's diagnostic claims — about accumulation, about metabolic signatures, about trajectory prediction — are frame-dependent claims unless the observation frame is declared. An assertion that a system is in Starvation is an assertion from a specific structural position at a specific temporal

resolution. The same system observed from a different frame may present a different signature. This does not make the diagnosis invalid — it makes the diagnosis positional. Diagnostic discipline requires declaring the frame: where the observation is positioned, at what temporal resolution, and what subset of metabolic activity that frame makes visible. Without frame declaration, two valid diagnoses of the same system may appear contradictory when they are in fact complementary.

The model's primary application domain — human organizations — typically operates within frames that are close enough to be treated as shared. Actors within the same organization experience approximately the same temporal flow and have access to overlapping subsets of metabolic activity. Frame dependence becomes consequential when the observation gap between frames is large: board versus engineering, headquarters versus field, investor versus operator. In these cases, frame dependence explains disagreements about system state that cannot be resolved by better data — they can only be resolved by integrating observations across frames.

The model does not prescribe a privileged frame. No observation position is inherently superior to another. Each frame reveals real metabolic activity that other frames may not capture. What the model requires is frame declaration — making explicit which position and which resolution an observation uses, so that frame-dependent observations can be compared, integrated, or recognized as complementary rather than contradictory.

2.4.2 Metabolism

Metabolism is the process by which a system sustains itself through the continuous transformation of inputs across contours over time. The term is adopted from biology, where metabolism denotes the set of life-sustaining reactions through which organisms convert energy, build structure, and respond to their environments. The adoption is not metaphorical — it identifies the same structural phenomenon at a different scale: a bounded system maintaining a non-equilibrium state through ongoing resource transformation.

A system with no metabolic activity — no resource transformation across contours over time — is inert. It is not a living system in the model's terms. The model's applicability conditions (limited resources, competing demands, non-static behavior) are metabolic conditions: they describe a system that is actively transforming resources under contour competition.

Metabolism operates on Code, not on Narrative. The metabolic state of a system is determined by what is actually being transformed — what resources are actually allocated to which contours, what capabilities are actually being built or degraded, what buffers are actually being consumed. Narrative may describe a different metabolic state than the one that exists. Measurement of metabolism requires observation of Code-level activity, not narrative-level claims.

2.4.3 Accumulation

The central claim of this chapter is that contour allocation over time produces accumulation — quantities that grow, decay, or transform as a consequence of sustained allocation patterns. Accumulation is what connects mechanism to trajectory: the mechanisms defined in System Dynamics produce allocation patterns; those patterns, sustained over time, produce accumulations; those accumulations determine which trajectories defined in System Course are available.

Three types of accumulation are defined. They follow different dynamics, are measured in different units, and interact with each other in ways that produce compound effects.

Debt

Debt is the accumulation produced by sustained under-allocation to a contour. When a contour receives less than what is required to maintain its functional capacity, debt accumulates — the gap between what the system can do and what its environment requires grows over time.

Debt is not an abstraction. It manifests as concrete capability loss: skills that atrophy, tools that become obsolete, relationships that decay, infrastructure that degrades, institutional knowledge that dissipates.

Evolution debt accumulates when the system under-invests in adaptation, learning, and capability change. Its characteristic curve is approximately exponential with a perception delay. Three mechanisms drive the acceleration:

Environmental change compounds. The environment does not change at a constant rate — new technologies enable further technologies, competitors who have adapted create additional pressure, regulatory and market conditions shift in response to shifts. The distance the system must travel to close the gap grows faster than linearly because the destination is itself moving.

Sensing capacity degrades. The longer a system operates without Evolution investment, the less capable it becomes of recognizing what it is missing. The receivers and actors that would detect the gap — people with current knowledge, processes that surface external developments, gate configurations that admit environmental signals — atrophy through the same under-allocation that produced the debt. The system progressively underestimates its own debt.

Compensation buffers exhaust. Early Evolution debt is often masked by existing buffers — experienced individuals who carry institutional capability, established tools that continue to function, client relationships that tolerate declining adaptive capacity. These buffers are finite. As they deplete, the debt that was always present becomes suddenly visible. The debt was accumulating gradually; its appearance is sudden. This dynamic — gradual accumulation, sudden visibility — is structurally identical to the compensation mechanism defined in System Dynamics, operating at the temporal scale.

The combined effect is a curve that appears flat for an extended period (while buffers mask and sensing degrades) and then rises steeply (when buffers exhaust and the accumulated gap becomes undeniable). From inside the system, this trajectory is experienced as a sudden crisis. From outside, the curve was climbing throughout.

The unit of Evolution debt is adaptation cost — the effort required to close the gap between current capability and environmental demand, measured from the system's current position. Adaptation cost captures the compounding dynamic: as infrastructure degrades and sensing capacity diminishes, the cost of closing the gap grows faster than the gap itself.

Survival debt accumulates when the system under-invests in maintaining its current viability — deferring infrastructure maintenance, under-resourcing operational functions, neglecting the sustenance of existing commitments. Its characteristic curve is approximately linear in accumulation but step-function in consequence.

Each unit of deferred maintenance adds roughly comparable additional debt. Unlike Evolution debt, Survival debt does not characteristically accelerate through compounding — a month of deferred server maintenance adds approximately the same risk as the previous month. However, the consequences of Survival debt are not proportional — they are threshold-based. The system functions adequately until a threshold is crossed, then fails abruptly. A server runs until it doesn't. Staff tolerate conditions until they don't. Compliance holds until it doesn't.

The unit of Survival debt is failure proximity — the distance between current accumulated debt and the nearest consequence threshold.

Reproduction debt accumulates when the system suspends productive output for an extended period. The machinery of Reproduction — delivery coordination, release discipline, quality judgment under time pressure, trade-off intuition — degrades through disuse. Its characteristic curve is logarithmic decay of readiness: initial pause produces minimal impact, with each additional unit of inactivity adding diminishing additional degradation.

Reproduction debt does not produce the catastrophic failure associated with Survival debt or the exponential divergence of Evolution debt. It produces increasing restart cost — the system becomes progressively rustier, and resuming productive output requires rebuilding coordination and rhythm from a degraded base.

The unit of Reproduction debt is restart cost — the effort required to resume productive output at a level comparable to the system's prior capacity.

Potential

Potential is the accumulation produced by Evolution-contour investment that has not yet converted to Reproduction-contour output. Learning that has occurred but not been applied. Capability that has been developed but not deployed. Adaptive capacity that exists but has not been activated by a Reproduction context.

Potential accumulates in steps, not continuously. Each significant learning event, exposure to a new domain, or capability development effort creates a discrete increment of potential. Between events, potential does not grow — it persists or decays.

Potential has a half-life. Unconverted potential does not persist indefinitely. Deep structural understanding — foundational models, architectural intuition, systemic insight — decays slowly. Specific technical fluency — tool-specific knowledge, implementation patterns, domain-current awareness — decays faster. The half-life varies by depth: the more foundational the potential, the longer it persists without activation.

Conversion of potential to productive output is discontinuous. Potential does not flow gradually into Reproduction — it converts when a threshold is crossed. The threshold has two components:

Accumulated potential — sufficient learning, capability, or adaptive capacity must exist to be activated.

Context readiness — a Reproduction-contour opportunity must exist that matches the stored capability. Context readiness is external to the unit that holds the potential. A person who has learned deeply cannot force conversion — they require a project, a role, a problem that needs what they have

built. An organization that has invested in new capabilities cannot force their deployment — it requires a market condition, a client need, or an internal demand that activates the investment.

Conversion, when it occurs, is characteristically disproportionate to the apparent input. The system invests over an extended period with no visible output change, then produces a rapid capability shift that appears sudden to observers. The investment was gradual; the conversion is discontinuous. This is the temporal inverse of the compensation-exhaustion pattern: where compensation masks gradual debt until sudden visibility, potential masks gradual investment until sudden conversion.

The unit of potential is conversion readiness — how much of the stored capability can be activated given an appropriate context.

Overhead

Overhead is the allocation share claimed by Survival-contour activity. It is not inherently pathological — every system requires Survival allocation to maintain viability. Overhead becomes consequential when it grows beyond what Survival functionally requires, crowding out Reproduction and Evolution allocation.

Overhead does not grow linearly under stress. It exhibits threshold-regime dynamics. Below a stress threshold, Survival allocation is proportional to the threat and reversible when the threat subsides. Above the threshold, the system shifts to a new allocation regime — Survival claims a larger fixed share that persists even after the stressor diminishes.

This persistence is a ratchet effect. Once Survival overhead increases beyond a threshold, it does not automatically return to baseline. In biological systems, chronic stress recalibrates the hormonal stress response to a new baseline that does not self-correct. In human organizations, crisis-era processes — additional reporting requirements, tighter approval chains, expanded compliance checks — persist as permanent overhead long after the crisis that justified them has passed. The overhead becomes structural.

Returning to a lower overhead regime requires active, deliberate intervention — not merely the removal of the original stressor. The system must dismantle the overhead mechanisms that were built during the stress period. This dismantling is itself resisted, because existing Code has incorporated the elevated overhead as normal operating logic.

The unit of overhead is allocation share — the fraction of total system resources claimed by Survival-contour activity. This is directly measurable through time allocation, budget distribution, and attention patterns.

2.4.4 Accumulation Interactions

The three accumulation types do not operate independently. They interact through reinforcing and masking dynamics that produce compound effects.

The Reinforcing Loop: Debt and Overhead

Evolution debt and Survival overhead form a reinforcing loop that is the most dangerous compound dynamic in the model.

As Evolution debt grows, the system's capability falls behind its environment's demands. Maintaining existing commitments with degrading capability requires more effort — more workarounds, more manual intervention, more crisis management. This additional effort is Survival-contour allocation. Overhead increases.

As overhead increases, the allocation available for Evolution decreases further. Less Evolution investment accelerates debt accumulation. More debt drives more overhead. The loop feeds itself.

This reinforcing loop is the structural mechanism underlying the degradation sequence defined in System Course. Distortion (initial displacement of Evolution) produces stress (accumulating debt generates overhead) produces degradation (the loop accelerates, consuming the system's capacity). The degradation sequence describes the trajectory; the debt-overhead loop describes the engine that drives it.

Masking: Potential as Debt Buffer

Accumulated potential can mask growing debt. A system with capable, experienced actors — people who have invested in Evolution historically — can sustain performance levels that its current allocation does not support. The actors' stored potential absorbs the gap, functioning as a compensation buffer.

This masking is structurally identical to the compensation mechanism defined in System Dynamics, with potential serving as the buffer. It is finite. The actors' potential is consumed — applied to maintaining current output rather than converted to new capability. When these actors depart or their potential decays, the masked debt becomes visible.

This pattern — a system sustaining itself by metabolizing its own accumulated potential as Survival fuel — is analogous to biological catabolism under starvation, where the organism breaks down its own structural tissue for energy. The system survives longer but becomes progressively less viable.

2.4.5 Metabolic Signatures

The accumulation dynamics defined above — debt, potential, overhead, and their interactions — produce characteristic observable patterns when sustained over time. These patterns recur across domains and are recognizable without formal quantitative measurement. They are defined here as metabolic signatures.

A metabolic signature is a recognizable configuration of contour dominance and accumulation profile that recurs across systems and predicts a characteristic trajectory. Signatures are identified through two axes: which contour dominates allocation (or whether allocation is balanced), and which accumulation type is primary (what is growing, decaying, or being consumed).

Contour dominance alone does not uniquely identify a signature. Two systems may both be Survival-dominant but exhibit structurally different metabolic states depending on what is accumulating beneath the surface. The accumulation profile is required as a secondary axis to complete the identification.

Six signatures are defined below. These represent the most common configurations arising from the model's accumulation dynamics. They are not exhaustive — additional signatures can be constructed by extending the same two-axis logic with additional dimensions, as described in the Application layer of the model.

Starvation

Contour profile: Survival-dominant. Accumulation profile: Evolution debt accumulating without buffer.

The system produces using only existing capability. No resources are allocated to adaptation, learning, or capability change. Output is sustained through the repetition of known patterns.

Starvation is not a temporary crisis response — it is a sustained metabolic regime in which Evolution-contour allocation has ceased without a restoration trajectory. The system is not deferring Evolution investment — it has stopped making it.

The characteristic trajectory is the debt-overhead reinforcing loop defined in Accumulation Interactions. Evolution debt grows exponentially (compounding environment, degrading sensing capacity, exhausting buffers). Overhead increases as maintaining commitments with degrading capability requires more effort. The loop accelerates toward degradation and breakdown.

Observable indicators: delivery conversations reference only known solutions; technical decisions default to precedent without awareness that alternatives exist; no latency-type activity is visible — no one is doing anything whose return is uncertain or distant; when environmental change demands adaptation, the system's response is slow and expensive because no adaptive infrastructure exists.

Borrowed Life

Contour profile: Survival-dominant. Accumulation profile: Evolution debt masked by consumption of accumulated potential.

The system sustains output levels that its current allocation cannot support. The gap between actual allocation and observable output is absorbed by specific actors whose historically accumulated potential functions as a compensation buffer. The system is Survival-dominant in its allocation but does not appear to be — because the buffer actors maintain output quality that exceeds what the system's current investment would produce.

Borrowed Life is structurally distinct from Starvation. In Starvation, Evolution debt is visible in degrading output. In Borrowed Life, the same debt is invisible because stored potential masks it. The distinction matters because the diagnostic path is different: Starvation is identifiable from output quality; Borrowed Life requires identifying the buffer actors and recognizing that the system's viability depends on their finite, non-replenished capacity.

The characteristic trajectory is apparent stability followed by sudden transition. While buffer actors are present, the system appears functional. When they depart — or when their potential is fully consumed — the accumulated debt becomes visible simultaneously, producing a rapid shift to visible

Starvation or direct entry to degradation. The transition speed is the diagnostic marker: a system that degrades suddenly after specific departures was in Borrowed Life, not in health.

Observable indicators: a small number of individuals are disproportionately referenced in decisions and problem-solving; knowledge is concentrated rather than distributed; output quality varies in patterns that map to individual involvement; succession planning is absent or produces anxiety when raised; the buffer actors themselves exhibit stress signals disproportionate to their formal role scope.

Fermentation

Contour profile: Evolution-loaded. Accumulation profile: potential accumulated without conversion context.

The system has invested in Evolution — learning has occurred, capability has been developed, adaptive capacity has been built. But the investment has not converted to Reproduction-contour output because no Reproduction context exists that matches the accumulated potential.

Fermentation is not pathological by default. It is a pre-conversion state that may be productive — potential will convert when context arrives — or may be wasteful — potential will decay without ever converting. The signature describes the metabolic state, not the outcome.

The characteristic trajectory has two exits. Conversion: a Reproduction-contour opportunity appears that matches accumulated potential, producing rapid and disproportionate output — the investment was gradual, the conversion is sudden. Decay: no context appears within the potential's half-life, and the system transitions toward reduced metabolic activity as accumulated learning degrades without activation.

The critical variable is context readiness, which is external to the fermenting system. The system cannot force conversion — it can only sustain potential until a matching context arrives or accept decay if none does.

Observable indicators: actors describe high engagement with learning and frustration with application; prototypes and experiments are abundant, shipped products are scarce; conversations are rich in possibility and thin on commitment; no delivery rhythm is present — activity is internally motivated and open-ended; the system feels intellectually alive but organizationally inert.

Flow

Contour profile: balanced — no single contour dominant. Accumulation profile: debts minimal, feedback functional.

All three contours receive allocation sufficient to maintain their functional capacity. Evolution investment converts to Reproduction output within recognizable cycles. Survival maintenance is adequate without crowding out the other contours. Feedback loops are open — signals about all three contours reach the system's allocation logic.

Flow is a specific metabolic state, not a generic description of health. It requires active conditions: functional feedback across all contours, Evolution investment that converts regularly, Survival maintenance that remains proportional, and allocation trade-offs that are explicit and managed. These conditions are demanding. Most systems are not in Flow most of the time.

The characteristic trajectory is sustained viability with adaptive capacity. Flow does not prevent degradation — it provides the adaptive capacity to respond to pressure without immediately entering the debt-overhead reinforcing loop. The most common exit from Flow is through environmental shock that forces Survival-dominant allocation, displacing Evolution first. A system that was in Flow before a shock has more capacity to recover than a system that was already in Starvation or Borrowed Life.

Observable indicators: Evolution investment is visible in Reproduction output within the same operational cycle — learning translates to improved delivery in real time; actors describe challenge without overwhelm; feedback operates bidirectionally — Survival concerns are heard, Reproduction commitments are realistic, Evolution investment is protected but accountable; the temporal signature is layered — Survival maintenance rhythms, Reproduction delivery cycles, and Evolution investment latency coexist without one crowding out the others; trade-offs are explicit and discussed, not hidden or denied.

Proliferation

Contour profile: Reproduction-dominant. Accumulation profile: Survival debt and Evolution debt accumulating simultaneously.

The system scales its existing form — expanding reach, increasing quantity, replicating patterns — without maintaining what it has or developing new capability. Growth proceeds without proportional investment in integrity or adaptation.

Proliferation maps to Reproduction-Dominant Distortion defined in System Course. The metabolic dimension adds what is accumulating beneath the growth: dual debt in both Survival and Evolution contours. The distortion pattern describes the posture; the metabolic signature describes the cost that is building.

The characteristic trajectory is deceptively stable. Growth metrics reinforce the allocation pattern through the feedback loop described in System Course — visible success metrics encourage further Reproduction allocation. Survival debt accumulates linearly until a threshold. Evolution debt accumulates exponentially. When either threshold is crossed — an infrastructure failure, a quality crisis, a market shift that demands adaptation the system cannot produce — the accumulated dual debt surfaces across multiple dimensions simultaneously. The system that was growing enters degradation or breakdown rapidly because it has no adaptive capacity and its maintenance infrastructure has been neglected.

Observable indicators: growth metrics dominate all reporting and conversation — headcount, revenue, market share, output volume; quality signals are present but deferred — "we'll address that after we scale"; new units replicate existing patterns including their weaknesses — no differentiation, no local adaptation; the temporal signature is accelerating Reproduction cyclicality without corresponding Survival maintenance rhythms or Evolution latency; infrastructure complaints increase but are treated as operational noise.

Churn

Contour profile: Evolution-dominant. Accumulation profile: no stable potential — each capability investment is replaced before it matures.

The system transforms constantly. Architecture changes, methodology shifts, strategic pivots, technology replacements — all occur at high frequency. But no investment is sustained long enough to stabilize into usable capability. Potential does not accumulate because each Evolution cycle resets rather than builds on the previous one.

Churn is structurally distinct from Fermentation. In Fermentation, potential accumulates and awaits a conversion context. In Churn, potential does not accumulate because no investment persists long enough to mature. The system is Evolution-dominant but Evolution-unproductive.

The characteristic trajectory is chronic underperformance without acute failure. Survival is adequate — the system does not collapse. Reproduction is intermittent and below the system's potential. Evolution activity is high-energy but low-yield. The system can sustain Churn for extended periods because it does not accumulate debt in the catastrophic patterns of Starvation or Proliferation. What it accumulates is opportunity cost — the growing gap between what the system could have achieved with sustained investment and what it actually achieved through constant reset.

The exit from Churn requires a Code-level intervention: the system's operating logic must be rewritten to value consolidation and sustained investment. This is difficult because Churn-culture actors often have high personal Evolution-contour orientation and interpret consolidation as stagnation.

Observable indicators: architecture, technology stack, or methodology changes with high frequency — each change justified by compelling reasoning, none sustained long enough to prove or disprove its value; actors describe fatigue rather than excitement about change; institutional knowledge is thin because nothing persists long enough to become institutional; the temporal signature is characterized by absence of both Reproduction cyclicality and Evolution latency — activity is intense but each cycle resets; the system cannot answer "what have we learned?" because each learning is overwritten by the next initiative.

2.4.6 Open Questions

This chapter establishes the foundational dynamics of accumulation and their metabolic signatures. Several areas require further development and are acknowledged as open.

Additional signatures. The six signatures defined above are the most common configurations. Additional signatures can be constructed from the same two-axis logic — contour dominance and accumulation profile — extended with additional dimensions such as relational context, temporal oscillation patterns, or metabolic intensity. The construction of domain-specific signatures belongs in the Application layer.

Quantification. The accumulation curves defined in this chapter have specified shapes (exponential, linear, logarithmic, threshold-regime) but not specified parameters. The shapes hold cross-domain; the rates are domain-specific. Establishing measurement approaches for specific application domains — particularly human organizations — is required for the model to move from qualitative diagnosis to quantitative prediction.

Signature transitions. The signatures describe metabolic states. The transitions between signatures — how a system moves from Flow to Starvation, or from Fermentation to Flow — are implied by the trajectory descriptions but not formalized as transition dynamics. Whether predictable transition sequences exist, and under what conditions transitions are reversible, requires further development.

Endogenous Phase Transition thresholds. The Endogenous Phase Transition mechanism defined in System Dynamics identifies internal accumulation thresholds as triggers for discontinuous allocation shifts. The accumulation dynamics defined in this chapter — debt curves, overhead ratchets, potential decay — are the substrates that approach those thresholds. Formalizing which accumulation profiles produce Endogenous Phase Transitions, what threshold shapes are characteristic for each accumulation type, and whether transition thresholds can be estimated from observable accumulation rates, requires further development. The connection is structural: the mechanism is defined in System Dynamics; the accumulation that drives it is defined here; the formalized relationship between them remains open.

2.5 System Course

The previous chapters defined what the system is made of, how it behaves under pressure, and what accumulates when those behaviors are sustained over time. This chapter defines the trajectories that result — how systems degrade, how they fail, and how they recover or transform.

System course is not predetermined. A system under stress may correct and return to its prior balance. It may reorganize around a new balance. It may transform into a structurally different system. Or it may collapse. The model does not predict which outcome occurs — it describes the structural conditions that make each outcome more or less available.

The degradation sequence defined in this chapter is a diagnostic tool, not a prophecy. It names the stages a system passes through under sustained distortion, and at each stage identifies the structural choices available — correction, adaptation, transformation, or continued deterioration. System Metabolism provides the accumulation dynamics that drive progression through these stages — what is building beneath the surface at each point, and what determines how rapidly the system moves from one stage to the next.

2.5.1 Distortion

Distortion is the state where allocation between contours deviates from the system's prior balance. It is the entry point for all subsequent system course trajectories.

Distortion is produced by displacement — one contour receiving less allocation to serve another, as defined in System Dynamics. Distortion is not inherently pathological. A system may distort deliberately, accepting temporary imbalance to address a specific pressure. Distortion becomes consequential when it persists without a restoration trajectory.

At the metabolic level, distortion is the point at which debt begins to accumulate in the under-allocated contour. The accumulation is below observable threshold — the system's output has not visibly changed. But the temporal record has begun: the gap between current capability and environmental demand is growing, even if no actor in the system can yet detect it.

At distortion, the system's structural choice is: **correct the allocation or continue**. If corrected — through reallocation, environmental change, or deliberate intervention — the system returns to its prior balance or establishes a new one. If uncorrected, distortion produces stress.

2.5.2 Stress

Stress is the state where sustained distortion begins to produce observable tension within the system. The system is still functional, but the consequences of imbalanced allocation are accumulating.

Stress manifests through the element infrastructure defined in System Structure. Signals about contour imbalance increase in frequency and intensity. Receivers oriented toward the under-allocated contour activate more often. Actors within the system begin to experience trade-offs as recurring rather than occasional.

At the metabolic level, debt accumulation has crossed a detection threshold — the system's signals now carry information about contour imbalance. Overhead may begin to increase as maintaining existing commitments with reduced capability requires additional effort. The debt-overhead reinforcing loop defined in System Metabolism may be initiating, though its effects are not yet dominant.

Stress may be masked by compensation — buffer expenditure absorbing the gap between expected and actual contour output. When compensation is active, the system may appear less stressed than it is. The stress is present; its visibility is reduced. Accumulated potential in experienced actors may function as a compensation buffer, producing the Borrowed Life metabolic signature — apparent stability sustained by finite, non-replenished capacity.

At stress, the system's structural choice is: **reallocate or compensate**. Reallocation addresses the underlying distortion. Compensation defers its consequences. Both are viable responses — but they lead to different trajectories. Reallocation can resolve stress. Compensation sustains it at the cost of buffer margin.

2.5.3 Degradation

Degradation is the state where sustained stress has begun to reduce the system's functional capacity. The system is not merely imbalanced — it is losing capability in the under-allocated contour.

The distinction between stress and degradation is: stress produces tension within a functioning system; degradation produces loss of function. A stressed system can still perform across all contours, though not at its prior level. A degrading system is losing the ability to perform in one or more contours.

At the metabolic level, the debt-overhead reinforcing loop defined in System Metabolism is active. Accumulated debt is producing measurable capability loss, and maintaining commitments with degrading capability drives Survival overhead higher, which further reduces the allocation available for the under-invested contour. The loop accelerates itself. If potential-as-buffer masking was present during the stress stage, it may be approaching exhaustion — the actors whose stored potential was absorbing the gap are departing or depleting.

Degradation is often the first stage visible to external observers. Internal actors may have experienced stress for an extended period — making trade-off decisions, watching signals escalate, feeling compensation pressure. External observers see degradation when output quality, system coherence, or adaptive capacity visibly decline.

At degradation, the system's structural choice is: **restructure or persist**. Restructuring means reorganizing allocation, potentially changing the system's operating priorities, to arrest the loss of function. Persisting means continuing the current pattern, which leads to further capacity loss. Restructuring at this stage is more costly than reallocation at the stress stage — it requires changing patterns that have become established during the stress period.

2.5.4 Breakdown

Breakdown is the state where the system can no longer sustain its prior contour balance under any reallocation. The capacity lost during degradation cannot be restored through redistribution of existing resources.

Breakdown is a threshold — the point where return to the prior state ceases to be available. Before breakdown, the system can recover its earlier balance through sufficient reallocation. After breakdown, the prior balance is structurally unavailable. The resources, capabilities, or conditions that sustained it have been consumed, lost, or irreversibly altered.

At the metabolic level, accumulated debt has exceeded the system's recovery capacity. The reinforcing loop has consumed the system's adaptive margin — the overhead required to maintain Survival under degraded capability now structurally prevents the reallocation that recovery would require. Overhead has ratcheted to a new regime and will not return to its prior level without active dismantling.

Breakdown does not mean the system ceases to function. It means the system cannot function as it did before. The system that continues past breakdown is operating under a different set of constraints than the system that entered the degradation sequence.

At breakdown, the system's structural choice is: **transform or deteriorate further**. Transformation means establishing a new contour balance — different from the prior one, but viable under current conditions. This may involve Code rewrite, boundary redefinition, or fundamental reallocation of priorities. Deterioration means continuing without establishing a new viable balance, which leads toward failure.

2.5.5 Failure

Failure is the state where the system can no longer maintain viability across its contour functions. The system is unable to sustain Survival, Reproduction, and Evolution at levels sufficient to continue operating as a coherent system.

Failure does not require that all contours cease simultaneously. A system may fail because one contour has collapsed entirely while others continue — but the loss of that contour makes the system as a whole non-viable. A system that cannot evolve in a changing environment will eventually fail to survive, even if Survival allocation is currently adequate.

At failure, the system's structural choice is: **reconstitute under new terms or collapse**. Reconstitution means the system's components, actors, or resources reorganize into a new system — with new Code, new Boundaries, potentially new contour priorities. The new system is not a continuation of the old one — it is a successor that inherits some of the old system's elements. Collapse means the system ceases to function as a system. Its resources disperse, its Boundary dissolves, and its contour allocation ceases.

2.5.6 Collapse

Collapse is the terminal state. The system ceases to exist as a bounded entity with contour allocation. Resources that were inside the Boundary return to the environment. Actors that were part of the system become independent or join other systems. The system's Code may persist as cultural memory, institutional precedent, or inherited assumption — but the system itself no longer operates.

Collapse is rare as an endpoint. Most systems exit the degradation sequence before reaching collapse — through correction, restructuring, or transformation at earlier stages. Collapse typically requires that all available choices at prior stages were either unavailable or not taken, and that the system's environment did not provide conditions for reconstitution.

2.5.7 Recovery, Transformation, and Reconstitution

The degradation sequence is not a one-way path. At each stage, the system has structural choices that can arrest the progression, reverse it, or redirect it.

Recovery is the return to a prior or comparable contour balance. Recovery is available at distortion, stress, and degradation — the stages where the system's prior capacity has not yet been irreversibly lost. Recovery becomes progressively more costly at each stage: correcting distortion requires reallocation; resolving stress may require reallocation plus unwinding compensation patterns; arresting degradation may require restructuring established patterns.

The progressive cost of recovery is driven by accumulation dynamics defined in System Metabolism. At each stage, the system carries more accumulated debt, has consumed more potential as buffer, and may have ratcheted Survival overhead to a higher regime. These accumulations do not reverse automatically when the decision to recover is made — they must be actively addressed. Debt must be closed through renewed investment. Consumed potential must be rebuilt. Ratcheted overhead must be deliberately dismantled. The later the stage, the greater the accumulated burden and the more costly the recovery.

Rebalancing is the establishment of a new contour balance that differs from the prior one but is viable under current conditions. Rebalancing is available at any stage, including breakdown. A system that cannot return to its prior balance may stabilize around a different one — allocating differently across contours, operating at a different scale, or serving different functions. Rebalancing is not failure — it is adaptation to changed conditions.

Transformation is structural change to the system itself — Code rewrite, Boundary redefinition, fundamental change to the system's operating logic. Transformation is distinct from recovery (which restores the prior system) and from rebalancing (which adjusts allocation within the existing system). Transformation produces a different system — one that shares elements with its predecessor but operates on different Code.

Transformation is most available during breakdown and boundary dissolution — states where the system's prior structure has already lost its hold and resistance to structural change is reduced. Transformation during earlier stages is possible but faces greater resistance, because the system's existing Code, Gates, and Receivers are still configured to reject structural change.

Reconstitution is the formation of a new system from the dispersed elements of a collapsed one. Reconstitution is available only after collapse — it is the exit path that the other three responses cannot provide, because at collapse the system itself no longer exists as an entity capable of recovering, rebalancing, or transforming. What reconstitutes is not the system but its elements — resources, actors, and persisting Code fragments — reorganizing into a new bounded system under new allocation conditions.

Reconstitution is structurally distinct from transformation. Transformation changes the system from within — the system persists as a bounded entity throughout the process, even as its Code, Boundaries, and allocation logic change. Reconstitution occurs after the Boundary has dissolved and the system has ceased to operate. There is no continuity of system identity. The successor system inherits elements from its predecessor but is a new system, not a modified version of the old one.

Reconstitution has three structural components:

Code persistence — some portion of the collapsed system's Code survives as an organizing principle for the successor. Not all Code persists. Code that was specific to the prior system's allocation pattern — its particular Gate configurations, its particular Receiver orientations, its particular Boundary definitions — typically does not survive, because these were products of the system's specific structure. Code that was foundational — the operating logic that defined what the system fundamentally was, what it recognized as legitimate, what rules governed its internal dynamics — may persist, because this level of Code is not dependent on a specific allocation pattern. It is the logic that any allocation pattern within the system operated on. What persists is determined by Code depth: shallow Code (specific practices, particular configurations) disperses with the system. Deep Code (foundational logic, governing rules, structural principles) may survive as inherited organizing logic in the successor. The persistence of deep

Code is not automatic — it requires that the dispersed elements carry it and that the reconstitution context activates it. Deep Code that is carried but never activated in a successor system decays over time, following the half-life dynamics defined for potential in System Metabolism.

Allocation reset — the successor system begins with a new allocation pattern. It does not inherit the collapsed system's contour posture. This is structurally necessary: the collapsed system's allocation pattern was the proximate cause of its trajectory through the degradation sequence. If the successor inherited that pattern, it would inherit the trajectory. Reconstitution produces a viable successor precisely because the allocation constraints of the prior system do not carry forward. The successor faces its own environmental conditions and establishes its own contour balance — potentially a very different one from what its predecessor sustained.

Reconstitution context — the conditions under which elements reassemble determine what kind of successor system forms. The same set of dispersed elements can produce different successor systems depending on what environmental conditions exist at the moment of reconstitution, which Code fragments are activated, and what new elements from the environment are incorporated during reassembly. Reconstitution is not deterministic — the same collapse can produce different successors under different conditions.

Reconstitution connects to Contour Saturation as defined in System Dynamics. When a system reaches Saturation — one contour consuming the entire allocation space, the other two displaced to zero with demands persisting at nonzero levels — the exit path is not reallocation (no degrees of freedom remain) but system reconstitution. Saturation drives the system through failure and collapse at a rate determined by how rapidly the unbounded displacement pressure exceeds the system's structural containment capacity. What follows is reconstitution or terminal collapse, depending on whether persisting Code and dispersed elements find a viable reconstitution context.

When reconstitution occurs repeatedly — a system collapses, reconstitutes, traverses a trajectory, collapses again, and reconstitutes again — a cyclic pattern may emerge. The model does not claim that cycles are universal or inevitable. It claims that when they occur, each cycle is governed by persisting deep Code but begins with a reset allocation pattern. Across cycles, the persisting Code itself is subject to drift — the same mechanism defined for Code change in System Dynamics. Each collapse-reconstitution event is a potential site for Code modification: the conditions of collapse may alter which Code fragments persist, and the reconstitution context may activate Code in a modified form. Over multiple cycles, this drift may produce systematic change in the successor systems — each governed by recognizably similar but not identical deep Code. Whether cyclic reconstitution produces convergent drift (successors becoming progressively more similar), divergent drift (successors becoming progressively more different), or stable replication (no systematic drift) depends on the specific dynamics of the collapse-reconstitution interface and the environmental conditions across cycles. Formalizing these dynamics — transition conditions, drift rates, cycle stability — is an open question for further development.

The relationship between failure and transformation: a system that fails is not necessarily a system that dies. Failure means the current system configuration is no longer viable. What follows depends on whether the system's components can reorganize. If they can — under new Code, with new Boundaries — a successor system emerges. If they cannot, collapse follows. Many of the systems that appear to have survived catastrophic failure have actually transformed — the entity persists, but the system operating within it is structurally new.

2.5.8 Distortion Patterns

When displacement persists and one contour chronically receives disproportionate allocation, the system develops a characteristic distortion pattern. Three patterns correspond to the three contours.

These patterns describe the allocation posture — which contour dominates and what structural consequences follow. The metabolic consequences of sustained distortion — what accumulates beneath the posture, at what rate, and with what observable indicators — are defined as metabolic signatures in System Metabolism. Distortion patterns and metabolic signatures are complementary: the pattern names the posture; the signature names what is building as a result.

Survival-Dominant Distortion

The system allocates disproportionately to preserving current viability at the expense of scaling and adaptation. Resources flow to protection, control, stability, and continuity. Reproduction and Evolution are chronically under-allocated.

Structural consequences: the system becomes rigid. Its Gate configuration tightens — fewer signals are admitted, fewer flows cross the Boundary. Its Receivers narrow toward Survival-relevant signals, reducing sensitivity to Evolution and Reproduction signals. Its Code reinforces existing operating logic and resists rewrite. The system survives but does not grow or change.

This pattern is self-reinforcing through feedback: rigidity reduces adaptive capacity, which increases perceived threat, which reinforces Survival allocation.

Reproduction-Dominant Distortion

The system allocates disproportionately to scaling, copying, and propagating its existing form at the expense of integrity and adaptation. Resources flow to growth, replication, and expansion. Survival and Evolution are chronically under-allocated.

Structural consequences: the system grows but becomes brittle. Quality degrades as scale increases without proportional investment in integrity. Weaknesses in the existing form are propagated rather than corrected. The system's Code prioritizes throughput over coherence. Adaptation is deferred because changing the form would slow replication.

This pattern is self-reinforcing through feedback: growth produces visible success metrics, which reinforces Reproduction allocation, which further defers investment in the Survival and Evolution contours that would address the growing structural fragility.

Evolution-Dominant Distortion

The system allocates disproportionately to transformation, experimentation, and capability change at the expense of stability and scale. Resources flow to learning, restructuring, and innovation. Survival and Reproduction are chronically under-allocated.

Structural consequences: the system changes constantly but does not stabilize or scale. Each new capability is abandoned before it matures. Operational reliability declines because continuity receives insufficient investment. The system's Code favors novelty over consolidation. What works is not preserved or propagated — it is replaced by the next experiment.

This pattern is self-reinforcing through feedback: constant change produces a sense of progress, which reinforces Evolution allocation, which further defers the Survival and Reproduction investment needed to consolidate and sustain what has been developed.

2.5.9 Element Failure Modes

System elements defined in System Structure can fail individually. Each element has characteristic failure modes — ways in which it ceases to perform its structural function.

Code Failure

Corruption — Code becomes internally inconsistent. Different parts of the system operate on contradictory logic. Boundary, Gate, Signal, and Receiver are configured by conflicting instructions, producing incoherent system behavior.

Drift — Code changes gradually and unintentionally, departing from the logic that made the system viable without any deliberate decision to change. The system slowly becomes something it did not intend to become.

Conflict — multiple Codes coexist within the same system Boundary. This occurs when systems merge, when subunits develop independent operating logic, or when Code rewrite is partial — changing some elements while others retain the prior Code.

Boundary Failure

Impermeability — Boundary blocks all flow in both directions. The system is isolated from its environment. No signals enter, no resources flow, no feedback reaches the allocation logic. The system operates on stale information and depleting internal resources.

Dissolution — Boundary ceases to distinguish inside from outside. The system loses its identity as a bounded entity. Resources, actors, and signals flow freely in all directions without structural control.

Misidentification — Boundary misclassifies inside as outside or outside as inside. The system treats its own productive elements as threats (expelling value-generating actors) or treats external threats as internal resources (admitting destructive flows).

Gate Failure

Permanent open — Gate admits all flows without selection. The system is flooded with signals, resources, and actors that its Receivers and allocation logic cannot process.

Permanent closed — Gate blocks all flows. The system is cut off from specific resource or signal channels regardless of their relevance or value.

Miscalibration — Gate admits what should be blocked and blocks what should be admitted. Selection criteria no longer match the system's actual needs. This is distinct from Boundary misidentification — the Boundary correctly defines inside and outside, but the Gate applies wrong criteria to flows crossing it.

Signal Failure

Noise — Signal channel carries information that does not correspond to actual system state. Receivers activate on false signals, producing allocation responses to non-existent conditions.

Attenuation — Signal weakens as it travels through the system, arriving at Receivers below Threshold. The information exists but does not trigger response.

Distortion — Signal is altered during transmission, arriving at Receivers with changed content. The Receiver responds, but to information that does not match the original state-change event.

Flooding — Signal volume exceeds Receiver processing capacity. Relevant signals are lost in volume. The system cannot distinguish important state-change information from routine transmission.

Receiver Failure

Threshold miscalibration — Receiver triggers at inappropriate signal strength. Too low: the Receiver responds to noise, producing constant false activation. Too high: the Receiver fails to respond to genuine signals, producing blindness to real state changes.

Legibility narrowing — Receiver's legibility range contracts, reducing the set of signal formats it can parse. Signals that were previously legible become inert. The system progressively loses the ability to interpret information from parts of its own operation or environment.

Orientation lock-in — Receiver becomes fixed on a single contour orientation, losing sensitivity to signals from other contours. A Receiver locked on Survival signals ceases to detect Evolution-relevant information, regardless of signal strength or format.

2.5.10 Compound Failure Patterns

Individual element failures frequently co-occur and interact, producing compound patterns with distinct characteristics. Three patterns are identified from case analysis. Additional patterns may exist — this set is not claimed as exhaustive.

Self-Attack

The system's own protective mechanisms target productive internal elements. Boundary misidentification (classifying inside as outside) combines with Gate miscalibration (admitting action against internal elements) and Receiver orientation lock-in (monitoring only for threat signals).

The system optimizes against itself — expelling actors, suppressing signals, or dismantling capabilities that are structurally valuable but that the system's distorted Code identifies as threats. The more productive the targeted element, the more aggressively the pattern escalates.

Unregulated Replication

A system function escapes the constraints that normally regulate its scale. Reproduction-contour activity proceeds without Survival-contour accountability or Evolution-contour input. Gate failure (permanent open on the replication channel) combines with Receiver orientation lock-in (monitoring only Reproduction metrics) and feedback distortion (growth signals amplified, integrity signals attenuated).

The system scales its existing form without constraint — propagating current patterns including their weaknesses. Growth metrics remain strong while structural fragility accumulates beneath them.

Signal Isolation

Information stops propagating through the system. Gate failure (permanent closed on signal channels) combines with Signal attenuation and Receiver legibility narrowing. The system's upper and lower levels, or its internal and boundary-facing functions, lose the ability to communicate.

Actors generating signals about system state — distortion, stress, degradation — find that their signals do not reach the allocation logic. Actors making allocation decisions operate on stale or absent information. The system fragments into locally functional but systemically disconnected regions.

2.6 Multi-Unit Behavior

The previous chapters defined the model for a single unit — its structure, dynamics, metabolism, and course. This chapter extends the same logic to systems composed of multiple units operating within a shared boundary.

The model applies to single units of any size. A person, a team, an organization, a country — each can be analyzed as a single unit with its own contour allocation, element infrastructure, and system course. Multi-unit analysis becomes necessary when the analyst needs to understand the relationships between units, not just the units themselves.

This chapter follows the same progression as the single-unit theory:

- **Unit Composition** extends The Model — defining how units nest and compose.
- **Inter-Unit Relations and Shared State** extends System Structure — defining how units connect and what the enclosing system looks like.
- **Multi-Unit Dynamics** extends System Dynamics — defining how allocation mechanisms operate across unit boundaries.
- **Multi-Unit Metabolism** extends System Metabolism — defining how accumulation dynamics operate across unit boundaries and how metabolic signatures interact between units.
- **Multi-Unit Course** extends System Course — defining how composed systems degrade, recover, and transform.

The model itself does not change at the multi-unit level. The contours are the same. The elements are the same. The dynamics are the same. What changes is that the system under analysis contains internal boundaries — and what happens at those boundaries determines whether the composed system is coherent, strained, or failing.

2.6.1 Unit Composition

Any unit can be decomposed into sub-units or composed with other units into an enclosing system. A family is composed of individual members. An organization is composed of teams. A country is composed of institutions, communities, and individual citizens. Each level of composition is a valid unit of analysis under the model.

The choice of unit boundary is analytical — it is made by the observer for a specific diagnostic purpose, not discovered as a property of the system. The same system can be analyzed as a single unit (what is its overall contour posture?) or as a multi-unit composition (how do its sub-units relate to each other?). Both analyses are valid. They answer different questions.

One unit's environment may be another unit's interior. When a team is the unit of analysis, the organization is its environment. When the organization is the unit of analysis, the same team is an internal sub-unit. This is not a contradiction — it is a consequence of boundary choice. The model's definitions (environment, boundary, gate) apply relative to the declared unit boundary, not in absolute terms.

Scale transitions — the point at which individual unit tracking yields to aggregate treatment — are driven by analytical purpose, not by a fixed threshold. An analyst tracking a three-person team may treat each member as a unit. An analyst tracking a thousand-person organization will not. The decision to aggregate is a judgment about what level of decomposition serves the diagnostic question. The model does not prescribe a correct level — it requires that the chosen level be declared explicitly.

2.6.2 Inter-Unit Relations and Shared State

Coupling

Units within a shared boundary may be coupled — dependent on each other's contour output for their own viability. Coupling describes the degree and nature of this dependency.

Tight coupling means one unit's contour output is a direct input to another unit's allocation. A team that depends on another team's deliverables for its own Reproduction is tightly coupled to that team. Disruption in one unit propagates directly to the other.

Loose coupling means units share a boundary but do not depend directly on each other's contour output. They draw from the same resource pool and operate under the same environmental conditions, but one unit's allocation does not directly constrain the other's. Disruption in one unit affects the other indirectly — through shared resource availability or environmental change — not through direct dependency.

Coupling is not uniform across contours. Two units may be tightly coupled on Reproduction (dependent on each other's output) but loosely coupled on Evolution (pursuing independent adaptation paths). The coupling profile — which contours are coupled and how tightly — shapes how dynamics propagate between units.

Synchronization

Synchronization describes whether units' contour postures are compatible within the shared boundary.

Synchronized units have contour postures that are mutually sustainable. Their respective allocations to Survival, Reproduction, and Evolution do not create structural conflict. This does not require identical postures — it requires compatibility. One unit may emphasize Reproduction while another emphasizes Survival, and the pairing may be viable if their outputs complement each other.

Desynchronized units have contour postures that create structural conflict. Their respective allocations work against each other. One unit's Reproduction emphasis may consume resources that another unit requires for Survival. One unit's Evolution efforts may destabilize conditions that another unit depends on for continuity.

Synchronization is not a binary state. Units may be synchronized on some contours and desynchronized on others. The degree and pattern of synchronization determines the enclosing system's structural viability.

Synchronization also has a temporal dimension. Units do not maintain static contour postures — their allocations shift over time in response to local pressures, environmental change, and internal dynamics. Synchronization is therefore not a one-time achievement but a process that requires periodic recalibration. The temporal dynamics of synchronization — how desynchronization accumulates between recalibration events, and what determines the minimum viable resynchronization frequency — are defined in Multi-Unit Metabolism.

Shared State

The enclosing system — the composed whole — has a state that is not an average of its units' states. Shared state is emergent. It is produced by the interaction of local contour postures, synchronization quality, mediator conditions, and compensation patterns operating across units.

A shared system state can be characterized as:

Coherent — local states are synchronized, mediators are functional, and the enclosing system operates as an integrated whole. Contour allocation at the shared level reflects the combined postures of its units without significant internal friction.

Tense but viable — local states contain tensions or partial desynchronization, but the system sustains viability through explicit trade-offs that are acknowledged and periodically renegotiated. The tension is managed, not hidden.

Compensated — local states conflict, but one or more units absorb the resulting friction through buffer expenditure. The enclosing system appears viable, but viability depends on hidden compensation. This state is structurally identical to single-unit compensation — finite, dependent on margin, and subject to exhaustion.

Distorted — local states are desynchronized and the enclosing system's allocation has deviated from viable balance. Compensation may be active but insufficient to mask the distortion. The enclosing system's contour output does not match what its combined resources should produce.

Degrading — the enclosing system is losing functional capacity. Desynchronization and distortion are consuming the system's resources faster than they are being replenished. Compensation margin is shrinking. Inter-unit relations are deteriorating.

Failing — the enclosing system can no longer sustain viability across its contour functions. Local units may still be individually viable, but the composed whole is not. The system's inter-unit relations have broken down to the point where the shared boundary no longer contains a functioning system.

Shared state is not determined by any single unit, no matter how dominant. A common diagnostic error is to treat the most visible, powerful, or vocal unit's state as the shared state. Shared state assessment requires examining the full composition — all participating units, their coupling, their synchronization, and the mediators operating across them.

2.6.3 Multi-Unit Dynamics

The allocation dynamics defined in System Dynamics — displacement, compensation, feedback, mediator dynamics — operate across unit boundaries within a composed system.

Cross-Unit Displacement

Displacement within a composed system can propagate across units. When the enclosing system faces resource pressure, displacement may not be distributed evenly — some units absorb more displacement than others, depending on their coupling, their position in the power structure, and the mediators operating across units.

A unit with less structural power may be displaced on behalf of a unit with more — its contour allocation reduced to sustain another unit's balance. This cross-unit displacement follows the same logic as single-unit displacement (context-dependent order, triggered by gap between available and required) but the context now includes inter-unit power relations and coupling.

Cross-Unit Compensation

One unit may compensate for another — absorbing a contour gap that originates in a different unit. The compensating unit expends its own buffer to sustain the enclosing system's observable viability.

Cross-unit compensation is structurally identical to single-unit compensation: it has a trigger (gap in one unit's contour output), a buffer (the compensating unit's resources or capacity), and a signal (evidence of buffer expenditure in the compensating unit). The difference is that the trigger and the buffer are in different units.

This creates a diagnostic challenge. The unit generating the distortion may show no signs of stress — its gap is being absorbed elsewhere. The unit providing compensation may show stress signals that appear disproportionate to its own allocation posture — because it is carrying load that originates outside its boundary. Without multi-unit analysis, the compensating unit appears to have an internal problem. With multi-unit analysis, the structural source of the distortion becomes visible.

Cross-Unit Feedback

Feedback signals generated by one unit may reach another unit's Receivers — crossing internal boundaries within the composed system. Whether these signals reach their destination depends on the same element infrastructure defined for single units: Gates must admit them, Receivers must have the Legibility Range to parse them, and Code must define them as relevant.

Cross-unit feedback can be reinforcing or balancing, following the same logic as single-unit feedback. A unit that displaces Evolution and produces short-term output gains may generate reinforcing signals that reach other units' Receivers, encouraging them to follow the same displacement pattern. Conversely, a unit that experiences degradation may generate balancing signals that reach other units, prompting them to adjust their own allocation to compensate or correct.

The effectiveness of cross-unit feedback depends on the inter-unit Gate configuration. In composed systems with open inter-unit signal paths, feedback propagates and the system can self-correct at the multi-unit level. In composed systems where inter-unit Gates are closed or narrowly configured, feedback does not propagate — each unit operates on local information only, and system-level distortion accumulates without triggering a system-level response.

Mediators Across Units

Mediators operating at the enclosing system level shape inter-unit dynamics in the same ways they shape single-unit dynamics: amplifying, blocking, or redirecting displacement, compensation, and feedback across unit boundaries.

Governance structures determine which units receive priority in resource allocation. Trust between units determines whether cross-unit signals are received as credible or dismissed. Power structures determine which unit's displacement is absorbed by which other unit. Legitimacy determines whether cross-unit compensation is sustained voluntarily or enforced.

Mediator alignment at the multi-unit level has the same structural consequence as at the single-unit level: aligned mediators produce coherent inter-unit behavior; conflicting mediators produce erratic or contradictory dynamics across units.

2.6.4 Multi-Unit Metabolism

System Metabolism defines how allocation over time produces accumulation — debt, potential, overhead — within a single unit. At the multi-unit level, the same accumulation types operate, but two additional dynamics emerge: accumulation across unit boundaries, and the interaction of metabolic signatures between units.

The metabolic dynamics of multi-unit systems differ structurally depending on the relationship between units. Two types of relationship produce distinct metabolic behavior:

Parallel units occupy the same level within a shared boundary — two teams in an organization, two departments in a company, two siblings in a family. Their metabolic relationship is symmetric: neither unit has structural authority over the other's allocation. They compete for shared resources and may synchronize or desynchronize, but neither can impose its metabolism on the other.

Serial units occupy different levels in an enclosure relationship — a team inside an organization, an acquired company inside a parent, an individual inside a team. Their metabolic relationship is asymmetric: the containing unit has structural authority over the embedded unit's allocation parameters. The container sets resource envelopes, imposes regulatory mechanisms, and defines Gate configurations that constrain the embedded unit's metabolic range.

This distinction matters because the primary metabolic dynamic differs by relationship type. Parallel units' central metabolic dynamic is desynchronization. Serial units' central metabolic dynamic is equilibration.

Desynchronization Debt

Desynchronization debt is the accumulation produced by sustained divergence between parallel units' contour postures without recalibration. It manifests as increasing coordination cost — the additional effort required to produce joint output from units whose allocation logics have diverged.

When parallel units are synchronized, their respective contour postures complement each other. Handoffs are smooth, expectations align, and trade-off decisions are mutually legible. When they desynchronize, each interaction requires additional translation, negotiation, and friction absorption.

Desynchronization debt compounds rather than accumulating linearly. Three mechanisms drive the compounding:

Mutual legibility degrades. As units' contour postures diverge, their Codes drift apart. Each unit progressively loses the ability to interpret the other's signals. Urgency signals from a Survival-dominant unit are read as overreaction by a balanced unit. Evolution investment by one unit is read as self-indulgence by a Starvation-signature unit. Each misreading further narrows the Receiver Legibility Range each unit maintains for the other's signal format, reducing the probability that the next signal will be received accurately.

Shared practices erode. Parallel units that were once synchronized typically share practices, tools, conventions, and coordination rituals. As their metabolic postures diverge, these shared practices become friction points rather than coordination mechanisms. A shared cadence that works when both units have compatible Reproduction rhythms serves neither when one unit shifts to Survival-dominant urgency and the other to Evolution-dominant exploration. Each abandoned shared practice removes a resynchronization mechanism, making the next recalibration harder.

Mediator alignment weakens. Trust, governance, and legitimacy between parallel units depend on mutual predictability. As postures diverge, predictability decreases. Trust erodes through incomprehension rather than betrayal. Governance mechanisms that assumed compatible postures produce inconsistent outcomes. Weakened mediators reduce the system's capacity to coordinate resynchronization, which extends the interval between recalibrations, which increases debt further.

The unit of desynchronization debt is coordination cost — the additional effort required to produce joint output compared to the effort required when units were synchronized. This is observable through alignment time in meetings, handoff failure rate, rework from mismatched assumptions, escalation frequency, and the reported experience of increasing difficulty in inter-unit collaboration.

Minimum Viable Resynchronization Frequency

Parallel units must recalibrate their contour postures periodically to prevent desynchronization debt from exceeding recovery capacity. The minimum viable frequency is determined by three variables:

Coupling tightness — how much joint output depends on coordination. Tightly coupled units accumulate desynchronization debt faster because every interaction exposes the divergence. Loosely coupled units can tolerate longer intervals because fewer interactions surface the divergence.

Divergence rate — how fast units' postures are moving apart. Units under similar environmental pressure with similar Code diverge slowly. Units under different pressures or with different Code diverge rapidly. Divergence rate is not a constant — it can accelerate when environmental conditions push units in opposite directions simultaneously.

Buffer capacity — how much desynchronization can be absorbed before it surfaces as friction. Coordinator roles, shared rituals with tolerance for variation, and actors who are fluent in both units' metabolic languages all function as buffers. These buffers are finite, following the same compensation logic as single-unit buffers.

The structural relationship is: minimum frequency must increase with coupling tightness and divergence rate, and can decrease with buffer capacity.

Desynchronization debt becomes unrecoverable when the coordination cost of resynchronization exceeds the benefit of remaining within the shared boundary. At this point, selective decoupling — defined in Multi-Unit Course — becomes more viable than resynchronization. The practical signal of this threshold: actors in both units begin reporting that independent operation would be easier than continued coordination.

Desynchronization debt exhibits a ratchet effect parallel to single-unit Survival overhead. Each desynchronization episode leaves residual coordination overhead — shared practices that were abandoned are not automatically rebuilt, trust that eroded through incomprehension does not instantly restore. This means each episode leaves the system slightly less resilient to the next divergence event.

Metabolic Equilibration

Metabolic equilibration is the process by which an embedded unit's metabolic signature converges toward its containing unit's metabolic baseline under sustained asymmetric metabolic authority. It is the defining metabolic dynamic of serial relationships.

The embedded unit's metabolic state follows an approximately exponential decay toward the container's baseline. The convergence operates through four concurrent processes:

Regulatory imposition. The containing unit's governance mechanisms — process standardization, reporting requirements, approval chains — impose the container's operating model on the embedded unit. This directly shifts allocation: Survival overhead increases as the embedded unit spends more time on compliance and process adherence. The container's Gate configurations restrict the embedded unit's allocation discretion.

Resource envelope constraint. The container determines the embedded unit's resource allocation — budgets, headcount, tool access, time. The container's own metabolic logic shapes what it considers reasonable allocation. Evolution-contour investment that the embedded unit's original metabolism would have funded becomes unjustified under the container's allocation logic.

Selective attrition. Actors whose personal metabolic needs do not match the new regime depart. The departure is selective — Evolution-oriented actors leave first because they have the highest sensitivity to contour mismatch and the most external alternatives. Each departure removes a carrier of the original metabolic signature and shifts the remaining population's aggregate metabolic expectation closer to the container's baseline. Attrition is not a side effect of equilibration — it is the equilibration mechanism operating at the population level.

Code drift. The embedded unit's Code gradually shifts through accumulated small departures. New actors are socialized into the container's norms. Decisions are made using the container's criteria. Practices that distinguished the embedded unit are replaced with container-standard alternatives. This is the Code drift mechanism defined in System Dynamics, operating specifically in the direction of the container's Code.

The convergence rate is modulated by drag forces that slow equilibration:

Legacy relationships — clients, partners, or external commitments formed under the embedded unit's original metabolism that explicitly demand the original metabolic signature. These create external pressure to maintain the original allocation pattern and attenuate as relationships end or are reassigned.

Residual Code strength — if the embedded unit's original Code is deeply held and institutionally embedded, it resists drift and regulatory imposition. Code strength decays over time through attrition and drift, but strong Code can significantly slow convergence in early phases.

Container's integration attention — if the container does not actively impose its model, convergence slows. Regulatory imposition and resource envelope constraint are weak under benign neglect. Convergence still occurs through attrition and drift, but at a lower rate. When the container turns attention to integration, the convergence rate increases abruptly.

Evolution floor — the container's own Evolution allocation level sets the asymptote. If the container permits some Evolution investment, the embedded unit converges to a state that includes some Evolution — not at the original level, but above zero. A higher floor means less total metabolic distance to travel.

Irreversibility

Equilibration becomes irreversible when the embedded unit can no longer recover its original metabolic signature even under restored allocation autonomy. Three mechanisms drive irreversibility in sequence:

Population thinning is the leading process. The actors who carried the original metabolic signature depart through selective attrition. At some point, the remaining original-metabolism actors are too few to sustain the original Code. The threshold depends on how distributed or concentrated the original Code was — a system where the original metabolism was carried by the entire culture needs a larger surviving population than one where it was concentrated in a few founders.

Code rewrite completion lags behind population thinning. As carriers depart and container-standard practices accumulate, the infrastructure that supported the original metabolism is replaced. Gate configurations, Receiver orientations, and allocation decision criteria all operate on the container's Code.

Potential exhaustion is the terminal process. All accumulated potential from the original metabolism has been consumed as buffer or has decayed through half-life. No stored capability from the original investment remains to seed recovery.

Effective irreversibility occurs at the population threshold — the earliest of the three. Once the carrier population drops below the minimum needed to sustain the original Code, recovery requires not just reallocation but recruitment of new actors who carry compatible metabolic expectations. This is building a new system, not recovering the old one — the distinction between recovery and transformation defined in System Course.

The diagnostic test for irreversibility proximity: if the container's regulatory imposition were removed and the embedded unit were given full allocation autonomy, could it re-express its original metabolic signature with its current population? If yes — equilibration is reversible. If partially — the system is in a critical zone where some aspects can be recovered and others are lost. If no — irreversibility has been crossed.

Arrested Equilibration

Equilibration can arrest — stabilizing at a point between the embedded unit's original metabolism and the container's baseline — under specific conditions:

Persistent external demand for the original signature. If legacy clients or external relationships continuously require the embedded unit's original metabolic output, and the container values these relationships enough to protect them, the drag force does not attenuate. The embedded unit must maintain some degree of its original metabolism to serve these relationships.

Structural separation within the container. If the embedded unit retains a degree of operational autonomy — its own Gate configurations, some independent allocation discretion — the container's regulatory imposition is attenuated. This is a deliberate architectural choice by the container.

Metabolic complementarity encoded in Code. If the container's Code includes a commitment to maintaining the embedded unit's distinct metabolism — because the container recognizes that the distinct metabolism produces something the container's own metabolism cannot — then the container actively limits its own convergence pressure.

The first two conditions are circumstantial and inherently fragile. Legacy clients eventually churn. Operational autonomy erodes through organizational inertia and alignment initiatives. These conditions slow equilibration but do not stop it permanently.

The third condition is the only potentially durable arrest mechanism — because it is encoded in the container's Code rather than dependent on external circumstances. But it requires the container's Code to simultaneously hold standardization as a value and exception as a policy. This is a Code complexity that most Survival-dominant containers eventually resolve by eliminating the exception, because the exception creates management overhead, reporting inconsistency, and perceived unfairness — all Survival-contour costs.

Arrested equilibration is therefore typically a metastable state — real and potentially long-lasting, but vulnerable to perturbation. Any disruption to the sustaining conditions can restart convergence from the arrested point.

Measurement Artifact

Metabolic equilibration produces a characteristic measurement artifact: lagging indicators improve as the system converges toward its new baseline. Retention rates stabilize because the remaining population is metabolically compatible with the new regime. Delivery predictability improves because the system operates within the container's standard patterns. Satisfaction metrics may rise because the actors who would report dissatisfaction have departed.

These improving indicators do not reflect system health in the original sense. They reflect metabolic homogeneity — the system has shed the variation that produced the dissatisfaction signals. The system is stable. It is not the system that was acquired.

The leading indicator of equilibration is contour mismatch — the divergence between what the system's actors expect in terms of contour allocation and what the system actually delivers. Contour mismatch signals precede population change. Population change precedes lagging indicator stabilization. The measurement sequence is: mismatch → attrition → metric normalization.

Signature Interaction

When units with different metabolic signatures are coupled, the interaction produces structurally specific dynamics that depend on the relationship type, coupling tightness, and inter-unit Gate configuration.

In serial relationships, signature interaction follows the equilibration dynamic defined above — the containing unit's signature subsumes the embedded unit's. This process is named **Digestion**: the metabolic process by which a containing unit's metabolism subsumes an embedded unit's distinct metabolic signature, converging the embedded unit toward the container's metabolic baseline.

In parallel relationships, four interaction patterns are identified. These are not exhaustive — other pairings may produce additional patterns. They are the most common configurations arising from the model's dynamics.

Metabolic parasitism. One unit's metabolic deficit is partially offset by extracting value from a coupled unit's surplus. The surplus unit bears the cost — its own overhead increases from supporting a unit that cannot match its metabolic rhythm. The deficit unit's debt growth slows but does not stop, because it is consuming the other unit's potential rather than building its own. The surplus unit's metabolic signature degrades over time through the additional load.

Failed complementarity. Units hold complementary metabolic resources — one has the need, the other has the supply — but inter-unit Gate configuration prevents transfer. Both units degrade independently despite the combined system holding the resources to address both deficits. The structural waste is that the system contains both the problem and the solution but cannot connect them.

Metabolic entrenchment. Parallel units with the same pathological signature reinforce each other through cross-unit feedback. The shared metabolic state creates a norm. Actors in either unit who exhibit contour behavior inconsistent with the shared signature are perceived as deviating, and the system's cross-unit feedback mechanisms suppress the deviation. The enclosing system's metabolic state becomes more rigid than the sum of its parts — degradation accelerates beyond what either unit's individual signature would predict, because mutual reinforcement suppresses the balancing feedback that might otherwise trigger correction.

Metabolic disruption. One unit's metabolic instability imposes Survival-contour costs on a coupled unit without any corresponding benefit. The stable unit's overhead increases from adapting to the unstable unit's volatility. The stable unit's signature degrades — typically toward Starvation — not through its own allocation choices but through externally imposed coordination costs. The unstable unit is often unaware of the impact because its own activity-level metrics appear healthy.

Three variables determine which pattern emerges: coupling tightness (how much each unit's output depends on the other's), signature complementarity (whether the units' metabolic states could theoretically benefit from exchange), and inter-unit Gate configuration (whether resources, signals, and potential can actually flow between units).

Cross-Boundary Detection

When one unit's metabolic state shifts, coupled units detect the shift through a four-stage chain, each stage introducing latency:

Internal-to-output latency — how long before the shifting unit's metabolic change affects its cross-boundary output. This is determined by the shifting unit's compensation mechanisms. If buffers are strong — experienced actors masking debt, established processes carrying Reproduction momentum — the internal shift produces no output change for an extended period. This latency equals the shifting unit's compensation margin duration.

Output-to-signal latency — how long before the output change produces a signal strong enough to reach the boundary between units. Some output changes are immediately visible — a deliverable is late, a capability is no longer offered. Others are gradual — average quality drifts, response time increases, innovation declines. The shape of the underlying accumulation curve matters: Evolution debt produces output change that is flat for an extended period then steep, which means the signal, when it arrives, is sudden and large rather than gradual and proportional.

Signal-to-detection latency — how long before the signal crosses the inter-unit Gate, reaches the detecting unit's Receivers, and exceeds Threshold. This is determined entirely by the detecting unit's element infrastructure: its Gate configuration (which signal types are admitted), its Receiver Threshold (how much change is required to activate), and its Receiver Legibility Range (whether it can parse the specific type of signal the shifting unit's degradation produces).

Detection-to-recognition latency — how long before the detecting unit interprets the detected signal as evidence of systemic metabolic shift rather than noise or an isolated incident. This is the detecting unit's Code-level processing. Most systems require multiple detections before reclassifying a partner's state.

The detecting unit's own metabolic state shapes detection speed. A unit in Flow — with balanced allocation, open Receivers, and broad Legibility Range — detects partner degradation early, through quality and capability signals. A unit in Starvation — with narrowed Receivers and Survival-dominant Gate configuration — detects partner degradation late, only when it manifests as delivery failure. A unit in Proliferation — monitoring primarily volume metrics — detects latest of all, only when volumetric output declines.

Detection latency is therefore a property of the unit pair, not of the shifting unit alone. The same metabolic shift produces different detection timings depending on the detecting unit's metabolic state.

This has a structural consequence for equilibration dynamics: an embedded unit's highest-quality partners — those most likely to be in Flow, because they chose the embedded unit for quality reasons — detect equilibration earliest and exit first. The remaining partners are progressively less capable of detecting the shift, not because the shift has slowed, but because their own metabolic state renders them insensitive to it. The partner portfolio degrades in parallel with the embedded unit's metabolism.

2.6.5 Multi-Unit Course

Composed systems follow the same course progression as single units — distortion, stress, degradation, breakdown, failure, collapse — with structural differences arising from the presence of multiple units within the shared boundary.

Local Viability with System-Level Incoherence

The defining characteristic of multi-unit failure is that individual units may remain locally viable while the composed system fails. Each unit maintains its own contour allocation, its own element infrastructure, and its own internal coherence — but the inter-unit relations that make the composed system function have broken down.

This is possible because units can sustain themselves locally as long as they have access to resources and a viable environment — even when the enclosing system is failing. The enclosing system's failure manifests not as unit-level collapse but as loss of inter-unit synchronization, breakdown of shared state, and inability to produce system-level output that depends on inter-unit coordination.

Failure Cascade

Failure can propagate across coupled units. When one unit's contour output is an input to another unit's allocation, failure in the first unit disrupts the second. If the second unit is coupled to a third, the disruption propagates further.

Cascade speed depends on coupling tightness. Tightly coupled units propagate failure rapidly — disruption in one produces immediate pressure on the next. Loosely coupled units propagate failure slowly or not at all — the disrupted unit's failure affects the shared resource pool but does not directly constrain other units.

Cascade can be arrested by decoupling — reducing the dependency between the failing unit and other units — or by compensation at the inter-unit level, where another unit absorbs the gap. Arrest through compensation is temporary, subject to the same margin constraints as single-unit compensation.

Multi-Unit Compensation Masking

Cross-unit compensation can mask system-level distortion in the same way single-unit compensation masks contour-level distortion. When one unit compensates for another, the enclosing system's observable output may appear viable — but viability depends on buffer expenditure in the compensating unit.

This produces the same observer-detection delay described in System Course: actors within the compensating unit experience the trade-off directly, while observers of the enclosing system see only the sustained output. When the compensating unit's margin is exhausted, the system-level distortion becomes visible — appearing sudden to observers who were not monitoring inter-unit compensation.

Recovery, Rebalancing, and Transformation

Composed systems exit the degradation sequence through the same structural choices available to single units, with additional options arising from multi-unit composition.

Unit-level recovery — individual units correct their own distortion, and inter-unit synchronization restores itself as local postures realign.

Inter-unit rebalancing — the composed system establishes new coupling patterns, synchronization arrangements, or resource-sharing agreements that produce a viable shared state different from the prior one.

Selective decoupling — units that cannot synchronize are separated, reducing the composed system's scope but restoring viability within the reduced boundary. The enclosing system becomes smaller but coherent.

System-level transformation — the enclosing system's Code is rewritten, producing new inter-unit relations, coupling patterns, and shared state. The composed system that emerges is structurally new — different from its predecessor in how its units relate, not just in how they allocate.

Unit replacement — a failing unit is removed and replaced by a new unit, or a unit's internal structure is rebuilt while its position in the composed system is preserved. The enclosing system sustains continuity while individual components change.

The availability of these options depends on the same factors as single-unit course: stage of degradation, compensation margin remaining, mediator conditions, and whether boundary dissolution has opened a window for structural change.

2.7 Application

The previous chapters defined the theory — what the model is, what systems are made of, how they behave, how they degrade and recover, and how composed systems operate. This chapter defines how the theory is applied: the discipline required to use the model on real systems without misapplication.

Application is not a separate theory. It is the procedural bridge between the model's concepts and their use in diagnosis, intervention, and comparison. Every concept defined in the theory chapters can be misapplied — through imprecise classification, unmeasured estimation, undeclared boundaries, or intervention mismatched to diagnosis. Application defines the constraints that prevent this.

2.7.1 Classification

Classification is the act of assigning a contour label — Survival, Reproduction, or Evolution — to a system activity, resource flow, or allocation decision.

Primary Intent Rule

Classification is based on primary functional intent — the contour demand the activity was undertaken to address. An activity that scales existing output is Reproduction, even if it incidentally improves process reliability (a Survival effect). An activity that restructures a capability is Evolution, even if it incidentally produces new output (a Reproduction effect).

Secondary effects do not change classification. They are recorded separately. An activity classified as Reproduction with a secondary Survival effect is not reclassified as Survival — it remains Reproduction with a noted secondary effect.

Ambiguous Cases

When primary intent is unclear, classification should be deferred until sufficient evidence is available. Premature classification produces interpretive drift — the analyst assigns a label based on partial information, and subsequent analysis is shaped by that label rather than by the system's actual behavior.

The most common source of ambiguity is the Survival–Reproduction boundary in organizational systems. Activities that maintain delivery capacity (Survival) and activities that scale delivery output (Reproduction) can appear identical from the outside. The distinction lies in intent: is the system preserving what it has, or expanding it? When the answer is unclear, the analyst should declare the ambiguity rather than force a classification.

Classification Discipline

Every classification should be accompanied by:

- the declared unit boundary (what system is being analyzed),
- the time horizon (over what period is primary intent assessed),
- the confidence level (how certain the classification is),
- and any secondary effects noted.

Classification without these declarations is undisciplined and should be treated as provisional.

2.7.2 Diagnosis

Diagnosis is the structured assessment of a system's state using the model's concepts. It is not freeform analysis — it follows a sequence that ensures all relevant structural features are examined.

Diagnostic Sequence

Declare the boundary. What system is being analyzed? What is inside the boundary, what is outside? If multi-unit, which units are included and at what level of decomposition?

Identify contour posture. What is the system's current allocation across Survival, Reproduction, and Evolution? Which contour appears dominant, which appears under-allocated? Is the current posture deliberate or the result of displacement?

Identify active dynamics. Which mechanisms from System Dynamics are currently operating? Is displacement occurring — and if so, which contour is being displaced and by what? Is compensation active — and if so, what type and through what buffer? Is feedback reinforcing the current posture or counteracting it? Are mediators amplifying, blocking, or redirecting any of these dynamics?

Assess system course stage. Where is the system in the progression defined in System Course? Is it in distortion, stress, degradation, breakdown, or failure? What structural choices are available at the current stage?

Examine element infrastructure. Are any system elements exhibiting failure modes? Is Code coherent or drifting? Are Gates appropriately calibrated? Are Signals reaching Receivers? Are Receivers configured to parse the signals the system needs to respond to?

Identify compound patterns. Are any compound failure patterns (self-attack, unregulated replication, signal isolation) present?

If multi-unit: assess inter-unit relations. What is the coupling profile? Are units synchronized or desynchronized? What is the shared state type? Is cross-unit compensation occurring?

State the diagnosis. Name the system's state, the dynamics producing it, and the structural choices available. Declare confidence level and note what evidence would change the diagnosis.

Diagnostic Discipline

A diagnosis is only as reliable as its boundary declaration and evidence base. Two constraints apply:

Boundary consistency. The analyst must maintain the same boundary throughout the diagnosis. Shifting between local and shared boundaries without explicit notice produces incoherent diagnosis — mixing unit-level and system-level observations as if they describe the same thing.

Evidence over inference. The diagnosis should be grounded in observable indicators — allocation patterns, signal behavior, gate configuration, output metrics — not in assumed intent or projected outcomes. Where direct evidence is unavailable, the analyst should declare the inference and its basis.

2.7.3 Metrics and Proxies

The model's concepts — contour posture, compensation margin, displacement pressure, feedback direction — are not directly measurable in most systems. They must be estimated through observable indicators.

Proxies

A proxy is an observable quantity that correlates with a model concept without measuring it directly. Proxies are estimates, not measurements. They are useful when direct measurement is impossible and misleading when treated as precise.

Each model concept requires its own proxy set, selected for the specific system under analysis. No universal proxy set exists — what indicates Survival-dominant allocation in one system may indicate something different in another. Proxy selection is part of the diagnostic process and should be declared explicitly.

Proxy Discipline

Every proxy should be accompanied by:

- the model concept it estimates,
- the direction of the relationship (does an increase in the proxy indicate an increase or decrease in the concept?),
- the confidence level of the correlation,
- and known conditions under which the proxy becomes unreliable.

Proxies without these declarations are undisciplined and may produce false confidence in diagnosis.

2.7.4 Intervention

Intervention is action taken to change a system's contour posture, dynamics, or course. The model informs intervention by connecting diagnosis to intervention class — what type of action is structurally appropriate for what type of diagnosed state.

Intervention Classes

Reallocation — shifting resources between contours. Appropriate when the diagnosis is distortion caused by displacement and the system has not yet reached degradation. The system's structure is sound; its allocation is imbalanced.

Compensation management — making compensation visible, monitoring margin, establishing exit conditions. Appropriate when the diagnosis identifies active compensation — particularly structural compensation without awareness or exit conditions. The intervention target is not the compensation itself but the underlying distortion it masks.

Feedback intervention — opening blocked feedback paths, dampening reinforcing loops, strengthening balancing loops. Appropriate when the diagnosis identifies feedback dysfunction — reinforcing loops accelerating distortion, or balancing signals failing to reach allocation logic due to Gate or Receiver failure.

Mediator intervention — changing the cross-cutting factors that shape allocation dynamics. Appropriate when the diagnosis identifies mediator dysfunction — mediators blocking necessary signals, amplifying destructive dynamics, or conflicting with each other. Mediator intervention is indirect — it changes the conditions under which allocation occurs, not the allocation itself.

Structural intervention — changing the system's elements: Code rewrite, Boundary redefinition, Gate reconfiguration, Receiver reorientation. Appropriate when the diagnosis identifies element failure or when the system's current structure cannot support viable allocation under any reallocation. Structural intervention is the most consequential class — it changes the system itself, not just its allocation.

Decoupling — reducing inter-unit dependency. Appropriate in multi-unit systems where failure is cascading through tight coupling and cannot be arrested through other intervention classes. Decoupling reduces the composed system's scope but arrests propagation.

Intervention Discipline

Match intervention to diagnosis. Each intervention class is appropriate for a specific diagnostic state. Applying structural intervention to an allocation problem is excessive. Applying reallocation to an element failure is insufficient. The mapping between diagnosis and intervention class is the core discipline of intervention design.

Intervene on the cause, not the symptom. Compensation is a symptom of displacement. Feedback dysfunction is a symptom of element failure. Intervening on symptoms (managing compensation without addressing displacement, fixing feedback without fixing the Gate that blocks it) produces temporary relief and structural persistence of the underlying problem.

Declare the expected effect. Every intervention should be accompanied by a stated expectation: what contour posture change, what dynamic change, or what element change the intervention is intended to produce. Without a declared expectation, the intervention cannot be evaluated — success and failure are indistinguishable.

Monitor for unintended displacement. Every intervention that changes allocation on one contour affects the others. An intervention that restores Evolution allocation may displace Survival or Reproduction. The analyst should anticipate which contour will absorb the reallocation and monitor for secondary distortion.

2.7.5 Comparability

Comparison is the act of relating two or more system states — across time, across units, or across cases. Not all comparisons are valid. Comparability defines when comparison is legitimate and when it produces false equivalence.

Comparability Conditions

Two system states are comparable when:

- they share the same unit boundary definition (or the boundary difference is explicitly declared),
- they are assessed at the same level of decomposition,
- the time horizons are compatible,
- and the environmental conditions are either the same or the difference is accounted for.

When any condition is not met, comparison is invalid unless the analyst explicitly declares the difference and accounts for its effect on the comparison.

Common Comparability Errors

Boundary mismatch — comparing a team-level diagnosis with an organization-level diagnosis as if they describe the same thing.

Decomposition mismatch — comparing a single-unit analysis of one system with a multi-unit analysis of another.

Temporal mismatch — comparing a system's state at two points in time without accounting for environmental change between them.

Environmental mismatch — comparing two systems operating under different environmental conditions as if their contour postures are equally available to both.

2.7.6 Use Constraints

The model can be misapplied. Use constraints define the boundaries of legitimate application — what the model can and cannot support.

The Model Does Not Prescribe Outcomes

No contour balance is inherently correct. The model describes the consequences of allocation patterns — it does not rank them. An analyst who uses the model to argue that a system "should" allocate more to Evolution is making a normative claim that the model does not support. The model can say what will happen if Evolution remains under-allocated. It cannot say that this outcome is wrong.

The Model Requires Declared Context

Every application of the model requires explicit declaration of: unit boundary, time horizon, environmental conditions, and confidence level. Application without these declarations is undisciplined and may produce conclusions that do not hold under different boundary or horizon assumptions.

The Model Does Not Replace Domain Expertise

The model provides a structural lens for diagnosis and intervention. It does not replace the domain knowledge needed to interpret what contour allocation means in a specific context, what proxies are valid for a specific system, or what intervention is feasible under specific constraints. The model structures thinking — it does not substitute for it.

The Model Is Falsifiable

The model makes structural claims that can be tested. If a system with limited resources, competing demands, and non-static behavior does not exhibit contour tension — the model's applicability claim is falsified for that case. If displacement does not produce distortion — the dynamics claim is falsified. The model invites testing and expects to be revised where evidence warrants it.

3. REFERENCE

3.1 About the Author

Viktor Jevdokimov, Vilnius, Lithuania — Creator of 3in3.dev, HCS, CDS, and 3SF

Viktor Jevdokimov is a software engineering leader, systems thinker, and framework designer with over 30 years of experience in software product delivery, modernization, and team alignment.

He is the creator of the **Human Cooperation System (HCS)**, **Commitment Design System (CDS)**, and the **3-in-3 SDLC Framework (3SF)**, and founder of the **3in3.dev** initiative — an independent platform dedicated to advancing collaboration and alignment between **Client**, **Vendor**, and **Product** ecosystems. Together, **HCS**, **CDS**, and **3SF** form a coherent system stack: HCS explains *how cooperation works*, CDS governs *what is being committed*, and 3SF structures *how work is delivered over time*.

Professional Background

- Began career supporting distributed banking software on DOS and Windows, developing a deep appreciation for troubleshooting and system design.
- Progressed through roles of **developer**, **architect**, **delivery lead**, and **practice lead**, working with international clients on modernization and cloud migration initiatives.
- Specializes in **Client–Vendor relationship design**, **commitment formation**, and **delivery system diagnostics**.
- Advocates for *"Context before Method"* and *"Trust before Control"* as guiding principles of effective collaboration.

Creative and Personal Work

Beyond software, Viktor is an **active musician and live sound engineer**, performing and mixing with the *Great Things* cover band.
He approaches both sound and systems with the same mindset: striving for **clarity, balance, and authenticity**.

About 3in3.dev

3in3.dev is an independent research and publishing initiative founded by Viktor Jevdokimov.
It consolidates his experience and experimentation into open frameworks that help organizations improve how they **engage, deliver, and measure value** across collaborative ecosystems.

3in3.dev publishes:

- The **Human Cooperation System (HCS)** — creates the conditions for cooperation.
- The **Commitment Design System (CDS)** — makes commitments explicit and survivable.
- The **3-in-3 SDLC Framework (3SF)** — turns those commitments into delivery and value.
- Supporting tools, templates, and learning materials under an open license.

> "These systems aren't about control — they're about clarity, trust, and the shared intent that makes collaboration work."
> — Viktor J., Creator of 3in3.dev
