Governance OS — Toolkit

Introduction

A comprehensive companion to the Governance OS — Operating Manual document.

Usage Contract (READ FIRST)

Purpose

This Toolkit exists to operationalize the Governance OS.

It is **not** explanatory material.
It is **not** optional guidance.
It is **not** a menu.

Each artifact is designed to be used **as written**, without interpretation.

How This Toolkit Is Meant to Be Used

- The **Operating Manual** explains *why* the system exists and *when* each mechanism applies.
- The **Toolkit** defines *what to do*, *how to do it*, and *what gets recorded*.
- The **Playbook** shows examples of the system in motion.

If you are asking *"Should we use this?"* — consult the **Manual**.
If you are asking *"What do I do now?"* — use the **Toolkit**.

Non-Negotiable Rules

1. **No Artifact, No Action**
   If a Toolkit artifact is required and not completed, the action is not allowed.

2. **No Interpretation**
   Artifacts are executed, not debated. If they don't fit reality, the Scope is wrong — not the tool.
3. **Ownership Is Mandatory**
   Every artifact has a named owner. "The team" is not valid.
4. **Everything Produces Exhaust**
   Completed artifacts are retained as decision traceability. This is not documentation overhead — it is audit exhaust by design.
5. **The OS Speaks First**
   The system enforces rules so individuals don't have to.

Failure Modes (Explicit)

- Skipping tools introduces **unowned risk**
- Bypassing gates converts urgency into **technical and organizational debt**
- Informal decisions invalidate downstream protection mechanisms

This Toolkit protects flow by making ambiguity expensive and clarity cheap.

Chapter 1: The Governance Lens

Templates

1. **Scope Card Template**
2. **Scope Charter Template**

Checklists

3. **Definition of Ready (DoR) Checklist**
4. **Definition of Done (DoD) Checklist**

Governance Foundations

5. **Governance OS Role Definitions**
6. **Accountability Quick-Map (New)**

Clarity Frameworks

7. **Boundaries & Dependency Mapping Table**
8. **Ideation Classification Worksheet**
9. **Reuse vs Reinvention Decision Guide**

Evidence & Maturity System (Non-Negotiable)

This section is the operational specification for the Lens scoring system.
It is executed, not interpreted.

Core rule

The Scope Risk Profile is an Evidence Gap Score:

- Lower score = more evidence present = higher maturity
- Higher score = more gaps = lower maturity

Governance OS does not score “quality of writing.”
It scores whether the organization has paid the clarity cost before asking delivery teams to pay the delivery cost.

If the score can be argued, the artifact is incomplete.

Lens Output Block (Mandatory)

This block must exist in every Scope Card and Scope Charter.
If it is missing OR contains invalid evidence (placeholders / non-traceable refs), approval is invalid.

> **Important:** These are **gap counts**, not “risk scores”.
> **0 = no gaps (all evidence tokens present). 5 = maximum gaps (missing all tokens).**

- **F — Feasibility gaps (missing tokens): 0–5** | Evidence refs: ___
- **B — Behavior/Operability gaps (missing tokens): 0–5** | Evidence refs: ___
- **V — Value clarity gaps (missing tokens): 0–5** | Evidence refs: ___
- **D — Dependency volatility gaps (missing tokens): 0–5** | Evidence refs: ___
- **R — Regulatory/Blast radius gaps (missing tokens): 0–5** | Evidence refs: ___
- **N — Novelty/Boundary risk gaps (missing tokens): 0–5** | Evidence refs: ___

GapScore = F + B + V + D + R + N: __ / 30
Maturity Level (derived): L__
Auto-selected Accelerator Path: ___

Deviation (if applicable):

- Skip-Phase Justification link: ___
- Named risk owner: ___

- Approvals/sign-off: ___

Evidence Token Catalogue (Fixed, Non-Negotiable)

Each axis score is computed by counting missing tokens.
Each missing token adds +1 to that axis (max 5).
Token present = Yes/No only. No partial credit.

A token is valid only if:

1. Filled (not empty)
2. Not a placeholder (see blacklist)
3. Owned where required (named accountable owner)
4. Traceable where required (evidence reference)
5. Explicit N/A allowed only as: `N/A — rationale` (silence is invalid)

If a token fails any rule above, it counts as missing.

F — Feasibility (5 tokens)

F1. Reuse vs Reinvent decision completed (with rationale + link)
F2. Non-functional constraints captured (latency/scale/RU/SLA/etc.)
F3. Integration surface listed (systems/contracts touched)
F4. Critical feasibility risks listed (top 3) with mitigations
F5. Feasibility evidence reference exists (prior pattern, spike/POC notes, or equivalent)

B — Behavior / Operability (5 tokens)

B1. Process owner / workflow behavior clarified (how work is actually done)
B2. Failure modes listed (what breaks, how we detect it)
B3. Operational ownership defined (who runs it after release)
B4. Observability intent captured (logs/metrics/alerts where relevant)
B5. Real-world validation plan exists (Pilot success criteria OR explicit alternative test plan)

V — Value Clarity (5 tokens)

V1. Objectives are measurable (KPI baseline → target)
V2. Success criteria observable (metric/event/behavior)
V3. Stakeholders + approval owners named
V4. “Cost of not doing” captured (or explicit assumption)
V5. Scope-to-outcome mapping exists (in-scope items clearly support objectives)

D — Dependency Volatility (5 tokens)

D1. Dependency table completed (owner + needed-by + risk-if-late + fallback)
D2. Boundaries & dependency map completed (at least high-level)
D3. System-of-record decisions recorded where relevant
D4. External team engagement defined (who decides what, and where)
D5. Contract/versioning implications noted (what breaks if behavior changes)

R — Regulatory / Blast Radius (5 tokens)

R1. Data sensitivity clarified (PII/GDPR/retention implications or explicit “none” with rationale)
R2. Security/auth boundary clarified (access model, roles/scopes where relevant)
R3. Audit/traceability requirement clarified (what must be logged, if applicable)
R4. Rollout/risk containment strategy stated (phased vs big bang, mitigation path)
R5. Incident severity expectations clarified (what happens when it fails, who owns response)

N — Novelty / Boundary Risk (5 tokens)

N1. Ideation classification completed (Variant / Extension / Net-New)
N2. Key domain terms defined (avoid semantic drift)
N3. “Do not change” zones identified (what is off-limits)
N4. New semantic objects approved where relevant (SoR owner / Architect)
N5. Open unknowns listed with closure plan (what will be resolved in which phase)

Placeholder Blacklist (Invalid Evidence)

Any field containing one or more of the following is invalid and counts as missing:

- TBD, TBA, TODO
- ?, unknown, unsure, maybe
- later, eventually, we’ll see, to be decided
- depends / it depends (unless dependencies are enumerated with owners + decision date)
- standard / as usual / normal process (unless the process/artifact is referenced)
- same as last time / like we did before (unless prior artifact is linked)
- N/A (without rationale is invalid)

Rule: placeholders are evidence gaps wearing a disguise.

Mandatory Traceability Rule (No ref = no evidence)

Where a field implies external truth, it must include a reference:

- dependency → owner + contract/source

- boundary → system-of-record decision
- constraint → measurement plan or known limit
- risk → mitigation plan or explicit acceptance
- outcome → observable success signal

Rule: No reference = no evidence = token missing.

Valid Reference Rule (Normative)

A “reference” is valid only if it is **resolvable** by a reviewer:

- A URL **or** a unique ID in the system-of-record (ticket ID, PR, decision log entry, contract version, etc.)
- It must point to the **specific** evidence, not a generic space.

Invalid examples (count as missing evidence):

- “Jira”
- “Confluence”
- “Slack thread”
- “See meeting notes”
- “Discussed with X”

Rule: If a reference cannot be opened and verified, it is not evidence.

Maturity Matrix (GapScore → Maturity)

Because the score counts gaps, maturity is inverted by design:

0–6 → L5 (Clear)
7–12 → L4 (Shaped)
13–18 → L3 (Known Gaps)
19–24 → L2 (Sketch)
25–30 → L1 (Fog)

Calibration rule:
Thresholds may be tuned after 3–5 uses, but not per-project and not mid-flight.

Calibration Governance (Normative)

Calibration may only be performed by the **Governance OS Custodian** (or explicitly delegated role).
Any calibration must:

- Be versioned (vX.Y)
- Include rationale + change log
- Take effect only for **new** Scopes after the change date
- Never be changed mid-flight

Rule: If calibration happens without versioning, the organization has exited Governance OS.

Accelerator Auto-Selection (Trigger Rules)

The Accelerator path is auto-selected using **gap triggers** (not opinion).

- If **F ≥ 3** → include **POC** (feasibility must be proven)
- If **B ≥ 3** → include **Pilot** (operability/real-world behavior must be proven)
- If **D ≥ 3** → include **Pilot** (dependency volatility must be confronted in reality, not argued about)
- If **V ≥ 3** → **NO build commitment is allowed yet**
  - You may run **Pilot as a value test** (smallest real validation),
  - but **MVP cannot start** until V is reduced below 3 (objectives + success signals become explicit evidence).
- If **R ≥ 3** or **N ≥ 3** → activate **Shield earlier**
  - boundary + ownership decisions must be recorded **before** widening blast radius

If none trigger → proceed to **MVP/V1** with normal gates.

Deviation Policy (Allowed, but weaponized)

Deviation from the auto-selected path is allowed only as an owned risk decision.

If any phase is skipped, the following is mandatory:

- Skip-Phase Justification (why the phase exists, what is traded)
- Risks introduced by skipping
- Mitigations (or explicit acceptance)
- Named risk owner
- Explicit approvals / sign-off

Rule: skipping phases is purchasing risk. The buyer signs the receipt.

Minimum required sign-off for any deviation (skip/compress phases):

- Sponsor (or delegated mandate owner)
- Scope Lead / Project Lead

- Engineering Lead
- Architect (required if boundaries / data meaning / platform constraints are touched)

If any minimum signer is missing: **deviation is invalid**.

Enforcement Rule: No Lens Output → No Work Starts

If the Lens Output Block is missing or contains invalid evidence:

- Scope is not ready
- Accelerator path may not be selected
- Work may not start beyond discovery

No artifact. No action.

Scope Card – Minimal but Safe Template

A lightweight contract that prevents ambiguity without slowing delivery.

1. Project Snapshot

- Title
- Date
- Owner(s)
- Summary (≤3 sentences):
  - What problem we're solving
  - Who it matters for
  - Why now

2. Objectives

(2–3 bullets, must be measurable — if it can't be measured, it's an assumption)

- O1 — KPI: ___ | Baseline: ___ → Target: ___
- O2 — KPI: ___ | Baseline: ___ → Target: ___

3. Scope

3.1 In Scope

(≤5 bullets — clarity beats completeness)

3.2 Out of Scope

(≤5 bullets — use "Not now" language to avoid emotional friction)

4. Definition of Ready (DoR)

This work **cannot** start until:

- Problem is validated
- Owner(s) named
- Dependencies known
- Stakeholders aligned
- No open ambiguity that blocks execution

5. Definition of Done (DoD)

This work is done when:

- The user can ___ and we can observe it through ___
- Documentation is updated in ___
- Monitoring/alerting is in place if relevant
- All acceptance criteria satisfied

6. Dependencies & Constraints

(Short and binding — every dependency must handshake)

Dependency	What we need	Needed by	Owner	Risk if late

7. Risks & Assumptions

- R1: ___ (Describe the risk → Describe what we will do if it happens)
- R2: ___
- Assumptions (max 3): anything softer than a metric goes here

8. Lens Output Block (Mandatory)

This block must exist in every Scope Card and Scope Charter.
If it is missing OR contains invalid evidence (placeholders / non-traceable refs), approval is invalid.

> **Important:** These are **gap counts**, not “risk scores”.
> **0 = no gaps (all evidence tokens present). 5 = maximum gaps (missing all tokens).**

- **F — Feasibility gaps (missing tokens): 0–5** | Evidence refs: ___
- **B — Behavior/Operability gaps (missing tokens): 0–5** | Evidence refs: ___
- **V — Value clarity gaps (missing tokens): 0–5** | Evidence refs: ___
- **D — Dependency volatility gaps (missing tokens): 0–5** | Evidence refs: ___
- **R — Regulatory/Blast radius gaps (missing tokens): 0–5** | Evidence refs: ___
- **N — Novelty/Boundary risk gaps (missing tokens): 0–5** | Evidence refs: ___

GapScore = F + B + V + D + R + N: __ / 30
Maturity Level (derived): L__
Auto-selected Accelerator Path: ___

Deviation (if applicable):

- Skip-Phase Justification link: ___
- Named risk owner: ___
- Approvals/sign-off: ___

9. Approval (Scope Card)

Approval Mode is determined by Maturity (derived from GapScore).

- **L4–L5 (Shaped/Clear): async approval allowed**
  - **SLA: 48h for Express, 3 days for Standard**
  - **Silence may count as consent only if the approval request is posted in a traceable system-of-record.**
- **L1–L3 (Fog/Sketch/Known Gaps): explicit approval required**
  - **Silence is not consent.**
  - **No explicit sign-off → no start (work is blocked + escalated).**

Non-negotiable: The Lens Output Block must be complete and valid, or approval is invalid.

Silent Consent Protocol (L4–L5 only)

Silence may count as consent **only if** the approval request:

- Is posted in the agreed system-of-record (work item / decision log / RFC repo)
- Includes a direct link to the Scope artifact
- Explicitly tags required approvers
- States the SLA deadline and timezone
- Contains the exact approval question (“Approve scope + Lens Output Block?”)

DMs and side-chats do not count.
If any condition is missing: **silence is not consent**.

Approval vs Evidence (do not confuse them)

Evidence fields do not allow silence. If a token requires info or a reference, silence counts as missing evidence.
Approvals may allow silence only under an explicit SLA and only when approvers are named and the approval channel is the System of Record.

Scope Charter – Single Source of Truth

A deep, cross-functional contract that eliminates ambiguity before delivery begins.

1. Executive Summary

(What we're doing, why we're doing it, what success looks like)

- What we're doing
- Who it serves
- Why now
- Expected outcome
- Success criteria

2. Scope

2.1 In Scope
2.2 Out of Scope

3. Business Case

- Value
- Expected impact
- Cost of *not* doing this

4. Ownership & Governance

(Replace RACI with an honest, founder-style table)

Area	Responsible (R)	Accountable (A)	Consulted (C)	Informed (I)
Business Outcome Ownership	PO	Business Owner	SMEs	Engineering Lead
Technical Solution	Team Lead	Engineering Lead	Architects	PO

Scope Definition & Governance Inputs	PM	PO + Business Owner	Engineering Lead	Stakeholders
Boundary & Dependency Decisions	Engineering Lead	Architecture	Platform Owners	PO
Accelerator Path Selection	Architect + Eng Lead	PO	Business Owner	Team
Sign-Off	PM	Named Owners	QA / Architect	Entire Team

5. Definitions & Boundaries

- Key terms
- Domain boundaries (what objects/data belong where)
- Data truth sources (who owns which system of record)
- Platform constraints
- Versioning or compatibility rules
- “Do not change” zones

6. Dependencies & Constraints

Dependency	What we need	Why we need it	Needed by	Owner	Risk if late	Fallback plan

7. Risks & Assumptions

Real risks, not corporate ones.

8. Acceptance Criteria

Clear, observable, testable.

9. Lens Output & Accelerator Path

9.1 Lens Output Block

This block must be completed. If it is missing or contains placeholders, approval is invalid.

- F — Feasibility: 0–5 | Evidence refs:
- B — Behavior/Operability: 0–5 | Evidence refs:
- V — Value clarity: 0–5 | Evidence refs:
- D — Dependency volatility: 0–5 | Evidence refs:
- R — Regulatory/Blast radius: 0–5 | Evidence refs:
- N — Novelty/Boundary risk: 0–5 | Evidence refs:

GapScore: __ / 30

Maturity Level: L__

Auto-selected Accelerator Path: ___

Deviation (if applicable):

Skip-Phase Justification link: ___

Named risk owner: ___

Approvals/sign-off: ___

9.2 Accelerator Auto-Selection (Trigger Rules)

The Accelerator path is auto-selected using **gap triggers** (not opinion).

- If **F ≥ 3** → include **POC** (feasibility must be proven)
- If **B ≥ 3** → include **Pilot** (operability/real-world behavior must be proven)
- If **D ≥ 3** → include **Pilot** (dependency volatility must be confronted in reality, not argued about)
- If **V ≥ 3** → **NO build commitment is allowed yet**
  - You may run **Pilot as a value test** (smallest real validation),
  - but **MVP cannot start** until V is reduced below 3 (objectives + success signals become explicit evidence).
- If **R ≥ 3** or **N ≥ 3** → activate **Shield earlier**
  - boundary + ownership decisions must be recorded **before** widening blast radius

If none trigger → proceed to **MVP/V1** with normal gates.

10. Change Governance

- How changes will be handled
- When scope is frozen
- When Shield activates

11. Delivery Model

High-level flow referencing the Accelerator.

12. Architectural Overview

1–2 diagrams to support clarity, including:

- Functional high-level flow
- Data touchpoints
- System boundaries
- Upstream/downstream interactions

13. Sign-Off

Rule: Scope Charters require **explicit sign-off**. Silence is not consent.

By signing, approvers confirm they have reviewed and accept:

1. **Scope Contract**
   - In Scope / Out of Scope is locked.
   - Definitions, boundaries, and System-of-Record decisions are recorded.
2. **Lens Enforcement**
   - Lens Output Block is present and valid (no placeholders, traceable refs).
   - GapScore + Maturity Level are accepted as the reality of the current evidence state.
3. **Accelerator Path**
   - The auto-selected path is accepted **OR**
   - A deviation is accepted as a **documented risk decision** (see below).
4. **Change Governance**
   - Change Forum triggers are accepted.
   - “No artifact, no action” is accepted.

Deviation Sign-Off (if applicable)

Deviation from the auto-selected Accelerator path is allowed **only** when all are true:

- Skip-Phase Justification link exists (traceable)
- Named risk owner exists (a real human)
- Deviation is logged in the project’s System of Record (link recorded)

Minimum quorum (non-negotiable):

- Business Owner / Sponsor (accepts business risk)
- Engineering Lead (accepts delivery/technical risk)
- Delivery Owner (PM/PO/Project Lead) (accepts execution accountability)

If quorum is not met: **deviation is invalid** and the OS considers the project out of compliance.

Signature Block

- Business Owner / Sponsor: _____ Date: _____
- Engineering Lead: _____ Date: _____
- Delivery Owner (PM/PO/Project Lead): _____ Date: _____
- (Optional) Architect / Security / Compliance: ____ Date: _____

Recorded at (link): _____

DoR Checklist

Purpose: Ensure no work enters delivery without the minimum clarity required for responsible engineering.

> **If these conditions are not met, the work is not allowed to start. The OS does not compromise on this.**

DoR — Problem & Outcome

- The problem is stated clearly (not a feature request).
- The desired outcome is understood and observable (metric, behavior, or constraint).
- Success criteria are defined and agreed.

DoR — Scope

- In-scope items are explicitly listed.
- Out-of-scope items are explicitly listed.
- No ambiguity remains (“we’ll figure it out later” is banned).

DoR — Ownership

- Sponsor confirmed (mandate + mandate boundaries).
- Scope Lead identified (delivery integrity).
- Product Owner identified (outcome).
- Engineering Lead identified (technical execution).
- Architecture consulted where domain boundaries apply.

DoR — Ideation & Concept Clarity

- Ideation funnel completed (Variant / Extension / New Concept).
- Reuse vs Reinvention decision documented.
- Boundary & dependency scan completed at a high level.

DoR — Constraints & Limitations

- Regulatory / compliance concerns identified.
- Performance / scalability / data constraints identified.
- Dependencies have owners and needed-by dates.

DoR — Feasibility

- High-level technical feasibility validated by Engineering Lead.
- Risks identified and accepted.
- Required resources validated (people, systems, access).

DoR — Approval

- Sponsor acknowledges problem + outcome.
- Scope Lead approves readiness.
- Engineering Lead approves feasibility.
- PO acknowledges scope boundaries.

DoR — Evidence & Maturity (Non-Negotiable)

- Lens Output Block is present in the Scope Card/Charter
- No placeholders exist (TBD/TODO/unknown/etc.)
- Every dependency has an owner and “risk if late”
- System-of-record and boundary decisions are explicitly recorded where relevant
- GapScore is calculated
- Maturity Level is derived
- Accelerator path is auto-selected

> **If any of the above is missing, the work is parked. Time is elastic — clarity is not.**

DoD Checklist

Purpose: Ensure work is complete, stable, and aligned with what was agreed — not “close enough.”

> **DoD is the exit contract. Once these conditions are met, the work is truly finished.**

DoD — Scope Completion

- All in-scope items are delivered.
- No out-of-scope items sneaked in (“nice to have” is not allowed after freeze).
- Acceptance criteria met.

DoD — Quality

- Code meets engineering standards.
- Test coverage meets minimum requirements.
- Functional testing complete.
- Edge cases validated.
- No unresolved P0/P1 defects.

DoD — Architecture & Boundaries

- All architectural constraints respected.
- Data contracts updated and validated.
- Reuse vs reinvention decisions followed.
- No shadow concepts introduced.

DoD — Documentation

- System documentation updated (only what is necessary).
- Runbooks/playbooks updated if affected.
- Knowledge handover completed (TL;DR version included).

DoD — Operational Readiness

- Monitoring added (events, logs, metrics).
- Alerting configured for critical paths.
- Scalability validated if required.
- Rollback or mitigation plan exists.

DoD — Stakeholder Alignment

- PO validates outcome.
- Sponsor acknowledges the delivered value.
- Scope Lead confirms scope integrity.
- Engineering Lead signs off on technical completeness.

DoD — Compliance

- Privacy/security/regulatory checks passed.

- No open compliance deviations.

DoD — Delivery Certification

- “Done” declared by Engineering Lead.
- “Accepted” declared by PO.
- “Closed” declared by Sponsor + Scope Lead.

> **If DoD is not met, delivery is not complete — regardless of sprint demos or stakeholder enthusiasm.**

Governance OS Role Definitions

These are not HR titles.
These are **functional roles** inside the Governance OS.

Product Owner (PO)

Owns the *business outcome*. Defines the “why” and sets success criteria.
Does **not** dictate the technical solution.

Sponsor

Holds the purse, the mandate, and the risk.
Accountable for whether the investment pays off — not for how the work is engineered.

Scope Lead / Project Lead

Guardian of the Scope. Ensures that the work delivered is the work agreed — no more, no less.
Owns delivery integrity, scope protection, and alignment.

Process Owner

Defines how the business workflow actually operates.
No process clarity → no technical clarity.

System-of-Record Owner

Decides what data means and who is allowed to change it.
Guardian of semantics, lineage, and data truth.

Engineering Lead

Owns the technical execution path, solution feasibility, and architectural boundaries.
Accountable for architectural integrity and technical guardrails.

Architect

Guardian of domain clarity, platform reusability, and system boundaries.
Makes the call on reuse vs reinvention, conceptual modeling, and the Accelerator path.

Team Lead

Owns day-to-day technical delivery.
Ensures the team executes the agreed solution correctly and sustainably.

PM (Project Manager)

Owns planning, coordination, risk, flow, and cross-team synchronization.
Does not own scope or solution direction.

SMEs (Subject Matter Experts)

Provide domain expertise and nuance.
Consulted where correctness matters — not decision-makers.

Accountability Quick-Map Sheet

Purpose: Instantly identify who owns what for any incoming work, without re-reading the entire matrix.

This is the **fast, one-screen version**.

Accountability Quick-Map

What needs to be decided?	Who decides it?	Support roles
Why are we doing this? (Outcome)	Sponsor	PO

What are we doing? (Scope)	Scope Lead	PM, PO
How will it be delivered? (Technical shape)	Engineering Lead	Architect, TL
What domain rules apply?	Architect	SoR Owner
What does the data *mean*?	System-of-Record Owner	Data Steward
What is the correct process flow?	Process Owner	SMEs
How does this impact business behavior?	PO	Sponsor
What technical constraints apply?	Engineering Lead	Architect
Do we reuse or reinvent?	Architect	Engineering Lead
Who coordinates execution?	PM	TL
Who protects the scope?	Scope Lead	Sponsor, PO
Who approves final delivery?	Sponsor + Scope Lead	PO, Eng. Lead

---

Quick-Memory Rule (ND-friendly):

> **Sponsor = Why**
>
> **Scope Lead = What**
>
> **Engineering = How**
>
> **Architect = Boundaries**
>
> **PM = Flow**
>
> **Process Owner = Behavior**
>
> **System-of-Record Owner = Meaning**

This becomes second nature after a few uses.

Boundaries & Dependency Mapping Table

Purpose: Reveal the ecosystem impact so teams stop discovering dependencies mid-sprint.

Boundaries & Dependency Mapping Table

Category	Questions to Answer	Fill-In Example
Domain Impact	Which domains are touched? Does this create a new object or extend an existing one?	e.g., “Booking + Catalog”
Upstream Dependencies	What must happen before we can execute? Who owns it? What’s the risk if late?	“Identity Service”
Downstream Dependencies	Who consumes this? What will break if behavior changes?	“Reporting Pipeline”
External Teams	Who must we coordinate with? What decisions require them?	“Payments Team”

System Boundaries	Which systems are allowed/not allowed to change?	“CRM = No schema changes”
Data Contracts	What are the inputs/outputs? Who owns the schema?	“Orders.v3 contract owner: Platform”
Regulatory Constraints	GDPR, PCI, retention, auditability?	“User deletion flows impacted”
Technical Constraints	Latency limits, SLAs, RU caps, scalability?	“API < 200ms required”
Required Decisions	Which boundaries require architectural approval?	“New domain concept?”
Fallback Paths	What’s Plan B if a dependency slips?	“Skip feature X; deliver Y”

Ideation Classification Worksheet

Purpose: Prevent shadow concepts, duplicate work, and invention where extension is sufficient.

This corresponds to the **4-stage funnel**.

Ideation Classification Worksheet

Stage 1 — Spark

Problem statement (not a feature):

→ _____

Stage 2 — Reality Check

Is this:

- Variant
- Extension
- Net-New Concept

Justification:

→ _____

Similarity scan:

- Existing concepts?
- Existing flows?
- Existing data objects?

Notes:

→ _____

Stage 3 — Reuse vs Reinvent

Does reuse cause:

- Cognitive overload
- Technical debt
- Boundary violation
- Architectural inconsistency

If reinventing:

- Why is this *not* an extension?
- What domain boundary requires a new object?

Decision:
Reuse / **Reinvent**

→ _____

Stage 4 — Boundary Map

Ecosystem impacts:

- Systems: _____
- Data contracts: _____
- Upstream: _____
- Downstream: _____

Final Classification Summary (1–2 sentences):

→ _____

Result feeds into:
Scope Card / Scope Charter

Reuse vs Reinvention Decision Guide

Purpose: A blunt instrument for preventing unnecessary invention.

Reuse vs Reinvention Guide

✔ Choose Reuse When:

- The concept is semantically similar
- The workflow is an extension of existing behavior
- Data meaning overlaps by >70%
- The new value is additive, not foundational
- Domain boundaries remain intact
- Extending the existing system is cheaper and cleaner

✔ Choose Reinvention When:

- The concept changes meaning in a way that breaks existing semantics
- Extending would create debt or violate domain rules
- The new object introduces a new lifecycle
- The workflow is distinct enough that retrofitting harms clarity
- Reuse would require more exceptions than rules
- Platform integrity demands separation

> If you need more than 2 exceptions to make reuse fit → it's a reinvention.
> If it fits cleanly in your existing domain language → it's a reuse.

Chapter 2: The Governance Accelerator

The Accelerator toolkit exists to answer one question:

> "Are we moving forward for the right reason — and at the right risk?"

Goal: Move fast *without* collapsing into chaos

Trigger: A Scope Card or Scope Charter has been approved

2.1 Accelerator Phase Overview

Purpose

Select the correct delivery phase based on risk and knowledge.

When to Use

Immediately after the Governance Lens completes.

Inputs Required

- Approved Scope Card or Scope Charter

Execution Steps

1. Review Scope risk profile
2. Select required phases: POC / Pilot / MVP / V1
3. Identify any skipped phases
4. Record rationale if skipping

Recording / Artifact

Accelerator Phase Selection

- Selected phases
- Skipped phases
- Justification
- Owner

Output / Decision

Defined Accelerator path for the initiative.

Failure Mode

Undefined phases lead to blurred purpose and rework.

2.2 Accelerator Gate Checklist

Purpose

Decide whether work may proceed to the next phase.

When to Use

At the end of **every** Accelerator phase.

Inputs Required

- Phase Success Criteria
- Trade-Off Log (if applicable)
- Skip-Phase Justification (if applicable)

Execution Steps

1. Identify phase being exited
2. Open corresponding Gate Checklist
3. Validate each criterion as Pass / Fail
4. Assign owner attestation
5. Make decision

Recording / Artifact

Accelerator Gate Checklist

- Phase
- Evidence links
- Owner
- Decision
- Date

Output / Decision

Proceed / Stay / Reduce Scope / Terminate

Failure Mode

Skipping gates converts unknowns into latent incidents.

2.3 Phase Success Criteria Templates

Purpose

Define objective success before work begins.

When to Use

At phase entry.

Inputs Required

- Scope Card / Charter

Execution Steps

1. Define phase hypothesis
2. Define observable success signals
3. Define explicit failure signals
4. Define ambiguity resolution rule

Recording / Artifact

Phase Success Criteria

- Hypothesis
- Success signals
- Failure signals
- Decision rule

Output / Decision

Binary phase exit criteria.

Failure Mode

Without criteria, phases drift and never end.

2.4 Skip-Phase Justification Form

Purpose

Allow risk-based acceleration without denial.

When to Use

Whenever a phase is skipped.

Inputs Required

- Accelerator Phase Overview
- Risk assessment

Execution Steps

1. Identify skipped phase
2. State why it normally exists
3. Define introduced risks
4. Name risk owner
5. Obtain approvals

Recording / Artifact

Skip-Phase Justification

- Phase skipped
- Rationale
- Risks
- Owner
- Sign-off

Output / Decision

Phase legally skipped with owned risk.

Failure Mode

Unjustified skips externalize risk onto delivery teams.

2.5 Trade-Off Log (A/B/C Options)

Purpose

Make decisions explicit and reversible.

When to Use

Any time scope, estimate, or quality changes.

Inputs Required

- Change proposal

Execution Steps

1. Define decision
2. Define Option A / B / C
3. Record gains and losses per option
4. Select option
5. Assign owner

Recording / Artifact

Trade-Off Log

- Options
- Consequences
- Decision

- Owner
- Date

Output / Decision

Owned decision with visible cost.

Failure Mode

Hidden trade-offs accumulate as silent debt.

2.6 Change Forum Input Sheet

Purpose

Prepare changes for structured decision-making.

When to Use

Before any Change Forum.

Inputs Required

- Active Scope
- Trade-Off Log (if applicable)

Execution Steps

1. Define proposed change
2. Define why now
3. Define impact
4. Define trade-off
5. Assign owner

Recording / Artifact

Change Forum Input Sheet

Output / Decision

Ready-to-decide change proposal.

Failure Mode

Unprepared forums devolve into opinion sessions.

Chapter 3: The Governance Shield

The Shield toolkit exists to answer one question:

> “How do we protect flow without turning people into gatekeepers?”

3.1 Scope Freeze Protocol

Purpose

Make “scope freeze” mean something concrete.

Must contain

- When freeze starts
- When it ends
- What is allowed
- What is explicitly not allowed
- Where new ideas go instead
- Who enforces the freeze

Key sentence (include verbatim)

> “Scope freeze does not prevent change. It prevents invisible change.”

Tone

Clear. Calm. Authority without aggression.

3.2 Noise Shielding Guidelines

Purpose

Define noise *before* people have to argue about it.

Must contain

Two columns:

- Allowed engagement
- Noise

Examples:

- Allowed → Clarifying questions via owner
- Noise → Drive-by opinions mid-sprint

Tone

Non-judgmental. Behavior-based. Specific.

3.3 Change Forum Agenda

Purpose

Keep forums short, decisive, and humane.

Must contain

Agenda sections:

1. Context (2 min)
2. Change proposal
3. Trade-offs
4. Decision
5. Owner & next steps

Hard rule

> No decision = no change.

Tone

Facilitated, not democratic.

3.4 Disruption Logging Sheet

(This one is sneaky-good)

Purpose

Make disruption visible without blame.

Must contain

- Date
- Source of disruption
- Type (scope / priority / dependency / opinion)
- Was Shield active?
- How it was resolved
- Time cost estimate

Why this matters

This becomes **evidence** when patterns repeat.

Tone

Observational. Never accusatory.

3.5 Stakeholder Alignment Contract

Purpose

Make expectations explicit *before* shielding begins.

Must contain

- What engagement looks like
- Where feedback goes
- How change is requested
- What “urgent” actually means
- Acknowledgement section

Important

This is not legal.

It’s social clarity.

Tone

Respectful. Firm. Adult.

3.6 Shield Activation Checklist

Purpose

Remove hesitation from enforcement moments.

Must contain

- Accelerator phase
- Freeze active? (yes/no)
- Change path defined?
- Stop-Work Triggers acknowledged
- Communication sent?
- Owner confirmed?

Key effect

This checklist gives *permission* to protect flow.

Tone

Binary. Simple. Empowering.

Appendix

- Glossary
- Roles
- Example Scenarios
- 1-page OS Summary