COGNITIVE EMPIRE · CANON MG-001

Maintenance Gravity Canon

Public Edition v1.1

Ebaeldien Maghazi
Cognitive Empire Systems Ltd
14 November 2026 · replaces v1.0
cognitiveempire.com/research/maintenance-gravity
Licensed under CC BY-ND 4.0

CONTENTS

· Canonical declaration

I Definition

II The five terms

III The Ten Laws

IV Measuring rate of change

V The formula

VI Bands

VII Ownership Clarity

VIII Counterweights

IX Review cadence

X Instruments

XI Validation protocol

XII Owner and Operator

XIII What Maintenance Gravity is not, and how it sounds

XIV A worked example

· Version log, verification and licence

Canonical declaration

Maintenance Gravity is a measurement framework for the burden of keeping a system. Cognitive Empire Systems Ltd publishes it, and v1.1 replaces v1.0, the Public Edition of 2026.

Status of the numbers. Every threshold, exponent and band cutoff in this canon is a calibrated default. Each was chosen by reasoning, not fitted to data. They will be refitted once the validation protocol in Part XI has run on 20 systems, and the results will be published whatever they show. Until then, a Maintenance Gravity score ranks systems by how heavy they are to keep. It does not yet measure that weight in units.

What is public. The physics is public: the definition, laws, terms, formula, bands, counterweights, cadence, instruments and validation protocol. Anyone may score their own system with it.

Cite as: Maghazi, E. (2026). *Maintenance Gravity Canon — Public Edition v1.1*. Cognitive Empire Systems Ltd.

PART I

Definition

> Maintenance Gravity is the accumulating upkeep burden a system places on the people accountable for it, expressed through the change capacity consumed by preserving existing capability and the necessary upkeep left outstanding. It rises when capability is added faster than coherence is preserved, and is shaped by complexity, dependency, opacity, rate of change and the clarity of ownership.

I.1 Consumed and outstanding. Upkeep consumed is a flow: the share of change capacity spent preserving existing capability over a period. Upkeep outstanding is a stock: necessary preservation work identified but not done, at a point in time. A system that defers its upkeep is not light; it carries gravity it has not yet paid. Every Maintenance Gravity audit records both, and they are never added together.

I.2 One physics, separate instruments. The same five terms and the same formula apply to every instrument. What changes is the unit measured and the evidence used:

- **Code Gravity** measures a codebase or service.
- **Ops Gravity** measures a team's estate of tools, vendors and automations.
- **Agent Gravity** (from v1.2) will measure a deployed agent workflow.

Scores from different instruments are never added together or averaged.

I.3 Boundary. Technical debt, tool overload and "AI risk" are symptoms of Maintenance Gravity, not the thing itself. Maintenance Gravity measures the burden of keeping a system, whatever caused it.

I.4 Validation rule. A Maintenance Gravity score is valid only if it predicts upkeep consumed and upkeep outstanding, each tested separately and measured independently of the score (Part XI). Until that is shown on systems outside the pilot, every score is a calibrated estimate. Scores rank systems rather than measure them: a higher score means heavier, not proportionally heavier.

There are two reasons. The five terms overlap: complexity raises opacity, and dependency raises the rate of change, so multiplying them counts some of the same condition twice. And the thresholds are calibrated defaults.

Upkeep consumed is the primary validation target. Upkeep outstanding is secondary: it tests whether the score misses deferred gravity. If scores consistently under-predict it, v1.2 adds a deferred-upkeep term.

PART II

The five terms

Each term is scored 1 to 5 from evidence, never from opinion. Scores 2 and 4 sit between the anchors below.

TERM	WHAT IT MEASURES	1	3	5
C, Complexity	How many moving parts the system has and how entangled they are	Few parts; a change stays where it is made	A moderate number of parts; some changes ripple to neighbours	Many parts; most changes ripple unpredictably; no one holds a full map
D, Dependency	How much the system relies on what its accountable people do not control: vendors, APIs, models, packages, single individuals	Few dependencies, all pinned, each with a tested fallback	Several dependencies, mostly pinned; fallbacks only for critical ones	Many dependencies, unpinned or on "latest", no fallbacks, or one person holds critical knowledge
O, Opacity	How difficult it is to see what the system is doing and when it is failing	Failures alert a named person; behaviour can be traced; operating notes are verified	Major failures alert; minor ones are found by users; some notes contradict behaviour	Failures surface through customers or by chance; no traces; notes absent or contradicted
R, Rate of change	New or materially changed capability per 90 days, relative to inventory (Part IV)	Under 2.5% of inventory	5–10%	Over 20%
OC, Ownership Clarity	How clearly named and able the people who answer for each part are (Part VII)	Critical components have no named owner, or owners who fail the Part VII tests	Most components have named owners; the tests are not demonstrated for some	Every critical component has an owner who passes all four tests

III.1 Evidence rules.

- R is counted, not judged: Part IV defines what counts.
- OC cannot score above 3 without a current Ownership Ledger (C4). Absence of ownership evidence scores OC 1; it does not prevent OC from being scored.
- O cannot score below 3 unless a Silent-Failure Review (C5) has run within the current review cycle.
- A term with no evidence is recorded as "insufficient evidence", not scored as low. A system with any term marked insufficient evidence receives no published score.

III.2 The terms overlap. Complexity tends to raise opacity, and dependency tends to raise the rate of change. Multiplying the terms counts some of the same condition more than once, which is why v1.1 scores rank systems rather than measure them (I.4).

PART III

The Ten Laws

The Ten Laws carry over from v1.0, reworded for precision. In v1.1 each law names the term it acts through, so a law can be checked against a score rather than only quoted.

#	LAW	STATEMENT	ACTS THROUGH
I	Every capability creates maintenance	Every tool, automation, integration or agent adds future upkeep from the day it is added	C, R
II	Invisible systems decay faster	What no one can see, no one maintains	O
III	Capability without ownership is deferred incident	Every capability needs a named owner and a plan for when it fails	OC
IV	AI multiplies output before governance catches up	Output scales on adoption; the discipline to maintain it scales only by design	R, O
V	Dashboards can hide reality	A dashboard is control only when its source, owner and use are explicit	O
VI	Human memory is not infrastructure	Knowledge held by a single person is a single point of failure	D, O
VII	Integration multiplies blast radius	Every connection spreads failure as well as capability	C, D
VIII	Gravity compounds quietly	It accumulates unseen until a trigger exposes it	Upkeep outstanding
IX	Speed without escalation conceals failure	Problems need a defined path to an accountable human	OC, O
X	Survivability outranks elegance	Durable systems keep operating and remain intelligible under pressure	All terms

PART IV

Measuring rate of change

R is the share of a system's capability inventory added or materially changed in the trailing 90 days. It is measured the same way whatever the review cadence.

$$R = \frac{\text{new capabilities + materially changed capabilities}}{\text{capability inventory at the start of the window}}$$

IV.1 Capability. A capability is a distinct production function or workflow with an identifiable outcome: a service, an endpoint, an automation, an integration, an agent. The capability inventory is the list of them at the start of the window.

- An estate with no inventory cannot be scored until a baseline exists.
- Each audit declares its counting level before counting (for example: services, not endpoints). A capability and the capabilities nested inside it are never both counted.
- Changing the counting level requires a documented re-baseline.
- If the inventory has fewer than 30 entries at the declared level, count one level finer, so that a single change cannot move R by a whole band.

IV.2 Material change. A capability counts as materially changed if, within the window, any of these happened:

- Its externally observable behaviour, contract or data schema changed.
- It gained or lost a dependency the accountable people do not control: a vendor, API, model or named person.
- Its permission boundary changed, or who can act through it without approval changed.
- Its owner, underlying model or major version changed.

Refactors with no contract change, settings changed within existing ranges, patch-version updates and copy edits do not count. Each capability counts once per window, however many times it changed. A change made by a vendor counts the same as a change made by the team.

IV.3 Bands.

R	SHARE OF INVENTORY CHANGED IN 90 DAYS
1	Under 2.5%
2	2.5–5%
3	5–10%
4	10–20%
5	Over 20%

IV.4 Where the bands come from. The bands are anchored to the review cadence, not derived from data. They are set so that, under the cadence in Part IX, unreviewed material change stays at or below about 10% of inventory between reviews. That anchor is a calibrated default and will be tested (Part XI).

IV.5 Evidence. Changelogs, merged changes, deployment history, vendor change notices and the owner map. Code Gravity can compute R from version control. Ops Gravity collects it from the owner map, vendor notices and a monthly diff of the tool inventory.

PART V

The formula

The score multiplies the four burden terms, divides by ownership squared, and maps the result onto 0 to 100. Squaring ownership is a declared prior: it encodes the doctrine's claim that ownership is the master counterweight. The claim is tested in the pilot, not assumed to be true (V.6).

V.1 Raw value.

$$\text{Raw} = \frac{C \times D \times O \times R}{OC^{k}} \qquad k = 2$$

Raw runs from 0.04 (all burden terms 1, OC 5) to 625 (all burden terms 5, OC 1).

V.2 Score.

$$\text{Score} = 100 \times \left(\frac{\ln(\text{Raw} \cdot 5^{k})}{\ln(625 \cdot 5^{k})} \right)^{\gamma} \qquad \gamma = 1.199$$

V.3 Where γ comes from. γ is the only value chosen by hand. It is set so that an average system, every term 3, scores exactly 50. The logarithm makes each term's effect proportional rather than additive; γ sets where the middle of the scale falls. Neither is derived from upkeep data.

$$\gamma = \frac{\ln 0.5}{\ln\left(\ln(81 \cdot 5^{k} / 3^{k}) / \ln(625 \cdot 5^{k}) \right)} = 1.199 \text{ at } k = 2$$

V.4 Reference points. The k = 1 column is the comparison variant scored alongside in the pilot.

C, D, O, R	OC	SCORE (K = 2)	COMPARISON (K = 1)
All 1	5	0	0

C, D, O, R	OC	SCORE (K = 2)	COMPARISON (K = 1)
All 2	4	26.8	25.1
All 3	5	38.9	42.9
All 3	4	43.7	45.9
All 3	3	50.0	50.0
All 3	2	59.1	55.9
4, 4, 3, 4	2	69.1	69.0
All 5	5	61.5	73.2
All 3	1	75.2	66.4
All 5	1	100	100

V.5 What squaring does. From the average system, raising one burden term from 3 to 4 adds about 3.2 points. Lowering ownership from 3 to 2 adds about 9.1. Ownership does not reduce the burdens themselves; it reduces the upkeep they cost. An owner who can account for a system, stop it and maintain it finds and fixes problems sooner. Squaring encodes that claim, and the pilot tests it. A system where every burden term is 5 but every owner passes all four tests scores 61.5, in the Fragility Zone but not in Collapse Risk.

V.6 Refit. After the 20-system pilot (Part XI), the better-predicting of k = 2 and k = 1 is selected. It is then tested on separate systems before any accuracy claim is published.

- If k = 1 predicts better, the master-counterweight claim returns to doctrine hypothesis and v1.2 adopts k = 1.
- Only k is fitted. γ stays derived from k and the all-3s anchor, and band cutoffs are re-described from the data rather than fitted, so 20 systems never carry six parameters.
- The v1.0 calibration audits are re-scored under the adopted k.

PART VI

Bands

Bands classify the burden of keeping a system. They do not predict the probability of an outage, a financial loss or a failure. Like every threshold in v1.1, they are provisional.

BAND	SCORE	WHAT IT DESCRIBES
Light	0–20	Upkeep barely registers against new work
Manageable Drag	20–40	Upkeep is noticeable; building continues at pace
Operational Weight	40–60	Upkeep competes with building for the same people
Fragility Zone	60–80	Preserving outweighs building; small changes risk breakage elsewhere

BAND	SCORE	WHAT IT DESCRIBES
Collapse Risk	80–100	Change capacity is collapsing: the accountable people can no longer keep up with what exists

VI.1 Collapse means change capacity. "Collapse Risk" names the collapse of the team's capacity to change the system. It is not a forecast that the system will fail.

VI.2 Boundary rule. Bands are assigned from the unrounded score. No combination of whole-number terms lands exactly on a cutoff, but displayed scores are rounded: a score shown as 60.0 may be 59.96 and still sit in Operational Weight.

VI.3 One scale for both instruments. v1.0 gave Code Gravity its own four bands (Orbital, Drag, Sink, Event Horizon). v1.1 retires them. Code and Ops Gravity share the five bands above, because they share one formula.

VI.4 Calibration notes. Light is rarely reached: 4.7% of the 3,125 possible term combinations score below 20. Real systems do not spread evenly across those combinations, so this describes the scale, not how common light systems are.

Once validation has run, the bands will be re-anchored in units of observed upkeep share. The Fragility Zone will begin where more than half of change capacity goes to preserving, and scores will map onto the bands through the refitted curve.

PART VII

Ownership Clarity

Ownership is evidenced, not declared. A name in a spreadsheet is not ownership.

VII.1 Two kinds of owner. The **accountable owner** is the person or organisation that answers for the system's consequences. The **operational owner** is a named human who understands, runs and keeps the system. Both are recorded. Legal responsibility and operational control are assessed separately, and neither substitutes for the other.

VII.2 The four tests. For each critical component, the operational owner must pass:

TEST	PASSES WHEN	EVIDENCE AN AUDITOR SEES
1. Named	A single named human is recorded, with a named substitute	The Ownership Ledger (C4)
2. Able to account	The owner explains what the component may do, why it was permitted, how it fails and which checks run. They need not explain every internal computation	The owner walks through a sampled run or incident, live
3. Able to stop	The owner can trigger a halt or a safe state through a named path	Demonstrated during the audit; time to halt recorded
4. Able to maintain	The owner has authority to obtain the capacity upkeep requires	Budget or time the owner controls, or a named approver who has granted it

VII.3 Accountability sink. A named owner who fails test 2, 3 or 4 is an accountability sink, not an owner. A human in the loop who cannot account for the system or stop it counts against Ownership Clarity, not for it. Where this is found, the audit records whom the consequences would actually fall on.

PART VIII

Counterweights

A counterweight is a standing practice that reduces Maintenance Gravity. v1.1 has eight; C7 and C8 are new. A score changes only on demonstrated effect: installing a practice earns nothing until the system is shown to be easier to keep.

Some counterweights reduce a term directly: C2, C3 and C8 (complexity, opacity), C4 (ownership) and C5 (opacity). Others reduce the burden a term creates without changing the term: C1 gives capacity for upkeep work, C6 lowers the rework that change causes without lowering R, and C7 lowers D only when a fallback is tested, a version pinned or a dependency removed.

#	COUNTERWEIGHT	RULE	AUDIT EVIDENCE	TERM
C1	Coherence Floor	15–20% of change capacity is protected for upkeep and cannot be borrowed for new work	Capacity plan or sprint records showing the protected share was spent on upkeep	C
C2	Duplication Tripwire	New capability is blocked if it duplicates an existing one, until the overlap is resolved	Inventory check before release; record of blocked or merged duplicates	C
C3	Deletion Authority	A named person has authority to retire capability, and a deletion review runs at the cadence in Part IX	Named authority; the last review's list of retired items	C
C4	Ownership Ledger	Every component has an accountable owner and an operational owner, with a substitute	Ledger current within one review cycle; Part VII tests on a sample	OC
C5	Silent-Failure Review	Failures that raise no alert are hunted on schedule and turned into alerts	The last review's findings and the alerts added	O
C6	Verification Budget	Verification capacity grows with the rate of change; no material change ships unverified	Verification time against change volume over the window	R
C7	Dependency Register	Every dependency the team does not control is listed with its version, owner, fallback or exit path, and safe state. People who alone hold critical knowledge are listed as dependencies	Register current; pinned versions; one fallback demonstrated	D
C8	Living Documentation	Every component has one verified operating note; a material change is not complete until its note is updated	Five sampled notes compared with live behaviour; one recovery walkthrough	O, supports OC

VIII.1 C7 in detail. Recording a dependency does not reduce D. A tested fallback, a pinned version or removing the dependency does. A high-impact external change (a vendor update, a model version, a package release) is verified before it is adopted, or the component is isolated. Other external changes are re-verified within 14 days. An unpinned dependency with no published changelog raises D and O, for less control and less visibility, not R.

VIII.2 C8 in detail. An operating note states what the component does, what it depends on, how it fails, how to halt or recover it, who maintains it, and when it was last verified. Verified notes lower O by at most one level. If more than 25% of sampled notes contradict observed behaviour, O rises by one level, capped at 5. AI-drafted notes count only once the owner has verified them against the running system.

VIII.3 C1 and the consumed share. Applying C1 raises upkeep consumed. That is intended: it converts upkeep outstanding into upkeep paid, so read the two together (I.1). Audit reports show C1's effect on upkeep consumed explicitly, so a higher consumed share after adopting C1 is not read as the counterweight failing. C1 also supports ownership test 4, able to maintain.

VIII.4 Counterweight discipline. Every counterweight is itself something to keep, and an unmaintained one raises Opacity.

- Time spent keeping counterweights counts inside measured upkeep.
- Every system carries a minimum set: C4 Ownership Ledger, plus one of C5 Silent-Failure Review or C8 Living Documentation.
- Others are added for the system's highest-scoring terms, and only where those terms score 4 or 5.
- C3 Deletion Authority applies to counterweights too: one that no longer lowers its term is retired.

PART IX

Review cadence

How often a system is reviewed depends on how fast it changes, not on the calendar. The score does not set cadence until the bands are validated.

R	REVIEW INTERVAL	UNREVIEWED CHANGE AT MOST
1–2	180 days	About 10% of inventory
3	90 days	About 10%
4	30 days	About 7%
5	14 days, plus event triggers	About 8% at 50% change per 90 days; event triggers cover faster change

A review covers the C4 ledger, the active counterweights (VIII.4) and a re-score.

IX.1 Event triggers. Whatever the calendar says, a review of the affected components is due within 14 days of any of these events:

- an external change to a registered dependency (C7);
- a change of accountable or operational owner;
- a failure discovered by a customer rather than by an alert.

A high-impact external change is verified before adoption, or the component is isolated.

IX.2 Same rule for every instrument. Code and Ops Gravity follow the same cadence. Ops relies more on event triggers, because most of its change is made by vendors.

IX.3 Score validity. A score is valid until the next review date. An expired score is withdrawn, not carried forward.

PART X

Instruments

Code Gravity and Ops Gravity are scored in v1.1. Agent Gravity is previewed here and will be scored from v1.2. All three use the same terms, formula, bands, ownership tests and counterweights; only the unit and the evidence differ.

X.1 Evidence by instrument.

TERM	CODE GRAVITY EVIDENCE (UNIT: A CODEBASE OR SERVICE)	OPS GRAVITY EVIDENCE (UNIT: A TEAM'S TOOL AND AUTOMATION ESTATE)
C	Module count and coupling; cyclic dependencies; change hotspots; duplicated blocks	Tool and automation count; overlapping tools; integrations between tools
D	Packages and unpinned versions; external APIs; share of commits held by one person	SaaS vendors; APIs the automations call; admin access held by one person
O	Test coverage on critical paths; swallowed exceptions; tracing and alerts	Automations without alerts; how the last failure was found; runbooks and operating notes
R	Counted from version control (Part IV)	Inventory diff and vendor change notices (Part IV)
OC	Code-owner coverage; Part VII tests on a sample	Owner map; who holds admin rights; Part VII tests on a sample

X.2 Code Gravity without sharing code. Most Code Gravity evidence can be computed by a script the client runs on its own machine. Only the summary figures leave the client; no code or credentials are exchanged.

X.3 The Seven Forces (Code Gravity). Each Force names a pattern that raises one or more terms, so an audit report can name the Force behind a high score.

FORCE	WHAT IT IS	RAISES
Duplication Mass	Redundant logic spread across the codebase	C
Comprehension Debt	Code no one currently understands	O; fails ownership test 2
Churn Drag	Recently written code that keeps being reworked	R

FORCE	WHAT IT IS	RAISES
Calcification	Old code never updated, consolidated or retired	C, D
Verification Deficit	Testing and review lagging behind code generation	O; amplifies the cost of R
Dependency Weight	Dependencies accumulating without curation	D
Judgment Erosion	Skills fading, architectural authority diffusing, no one willing to delete. The terminal force	OC

X.4 The Eight Domains (Ops Gravity).

DOMAIN	WHAT IT IS	RAISES
Tool Gravity	Overlapping or unowned tools and platforms	C, D
Workflow Gravity	Processes that depend on informal knowledge and workarounds	O, OC
Data Gravity	Data that cannot be trusted, connected or interpreted	O, C
Automation Gravity	Fragile automations that fail silently	O, D
AI Gravity	AI output that inflates review and quality-control burden	R, O
Decision Gravity	Decisions made without records or owners	OC
Knowledge Gravity	Critical knowledge held in heads, chats and scattered files	D, O
Governance Gravity	Authority and escalation lagging behind system complexity	OC

X.5 Agent Gravity (preview for v1.2). The unit is one deployed agent workflow. It is a separate instrument because an agent's behaviour can change when its model provider changes the model, while the team's own changelog stays empty.

- **Evidence basis:** pending. No public dataset yet covers agent upkeep.
- **Rate of change:** a provider change that alters behaviour counts toward R once observed, through an announcement or a regression test. An agent on an unpinned model whose provider publishes no changelog scores higher on D and O; R rises only with observed changes.
- **Agent counterweights:**
- **Kill condition:** if Ops evidence alone predicts agent scores (correlation above 0.8), Agent Gravity merges back into Ops Gravity.

PART XI

Validation protocol

A framework that cannot fail is not a measurement. Maintenance Gravity is tested by comparing its scores with upkeep measured independently, and the result is published whatever it shows.

1. **Shadow metric first.** Before any scoring, the accountable team estimates the share of last quarter's change capacity spent preserving rather than building, from its own calendars, tickets or time records. This is upkeep consumed. Work is classified by one rule:
2. **Upkeep outstanding.** The same intake lists necessary preservation work not done, with evidence: open dependency updates, known fragile components, postponed recovery work. Each item records estimated person-hours, the uncertainty of that estimate, and the date. It is never added to upkeep consumed.
3. **Blind scoring.** Whoever scores the five terms does not see the shadow figures until the score is fixed. In the pilot, the scorer sits outside the accountable team and is trained on the anchors. In external validation, the scorer sits outside the organisation.
4. **Source recorded.** Each shadow figure records where it came from: time tracking, ticket classification, or memory. Estimates from memory are kept but weighted lowest.
5. **Two stages.** Stage 1, the pilot, tests usability, whether two auditors reach the same scores, evidence collection, and early relationships between the score and each upkeep measure. Stage 2, external validation, freezes the model and tests it on a separate set of Code and Ops systems before any claim of predictive accuracy is published. Predictions are compared with observed upkeep, never with the counterweights a system has installed.
6. **Independent data.** For at least 5 pilot systems, upkeep consumed comes from ticket or calendar data classified by someone who never speaks to the scorer, so the team's sense of overload cannot drive both sides of the test.

XI.1 Kill conditions and checkpoints.

- At 5 systems: if the score does not track upkeep consumed (correlation near zero or negative), stop and redesign before continuing.
- At 10 systems: check band placement.
- If any two terms correlate above 0.8 across the pilot, they are one term, and v1.2 merges them.
- Interim results are published at each checkpoint.

XI.2 Intake questions.

1. Last quarter, roughly what percentage of your team's change capacity went to keeping existing things running, rather than building new ones?
2. What necessary upkeep did you postpone last quarter? List it.
3. Where do those numbers come from: time tracking, tickets, or memory?

XI.3 What public cases can and cannot show. Public failures show that terms were high. They do not show the upkeep share in the months before. They illustrate Maintenance Gravity; only audits under this protocol validate it.

XI.4 Minimum audit. A 60–90 minute screen-share yields a provisional score: the five terms with their evidence, and the counterweights to start with. It does not yield a published band, and it does not test the score against upkeep.

PART XII

Owner and Operator

Ownership Clarity rests on consequential judgment, which the Cognitive Empire doctrine defines: the accountable commitment that machines may inform and execute but never own (Operator Kernel, Chapter 1 and Appendix D). This part sets out only what scoring needs from that doctrine.

XII.1 Operator. The operator is the human or machine that performs or selects actions within a system. Operators are increasingly agents.

XII.2 Owner. The owner answers for the consequences: an accountable owner, a person or organisation able to bear the loss, and an operational owner, a named human (Part VII). Delegating execution to an operator never transfers ownership.

XII.3 Where accountability goes wrong. When a system decides and no owner can account for it or stop it, accountability does not disappear. It slides to the nearest human operator, who carries consequences without control. Maintenance Gravity records this as an Ownership Clarity failure (VII.3).

PART XIII

What Maintenance Gravity is not, and how it sounds

Maintenance Gravity measures one thing: the burden of keeping a system, in change capacity consumed or owed. It does not replace the practices that reduce that burden.

MAINTENANCE GRAVITY IS NOT	WHY
Technical debt	Debt is one source of gravity, in code only. Maintenance Gravity also covers vendors, tools, automations, people and agents
Tool overload	A symptom: too many tools raise C and D
AI risk	Maintenance Gravity does not estimate harm or the probability of failure. It measures upkeep
Governance	Governance sets the rules. Maintenance Gravity measures what keeping the system costs, whether the rules are followed or not

XIII.1 Relation to governance. Several counterweights draw on practices governance work also uses: named ownership (C4), verification (C6), deletion authority (C3). Maintenance Gravity does not claim they are new. It claims the burden they reduce can be measured, and that whether they work is shown by the score falling. "Governance" appears in Maintenance Gravity only as the name of a domain.

XIII.2 Symptom register. Gravity is usually heard before it is measured. Each phrase points to the term to check first.

WHAT PEOPLE SAY	CHECK FIRST
"Only Sarah knows how that works."	D (a person as dependency), OC
"We have the data somewhere."	O
"The automation usually works."	O
"Don't touch that Zap."	C, OC
"The dashboard isn't fully accurate."	O
"We use three tools for the same thing."	C
"We'll clean this up later."	Upkeep outstanding
"The AI helps, but now we have too much output."	R, O
"Nobody owns that process."	OC
"We're moving fast, but things are messy."	R, C

PART XIV

A worked example

This example is illustrative: the system is invented, so that every step can be shown. It scores 69.1, in the Fragility Zone, with weak ownership as the lever that moves it most. A worked audit of a real system will be published once one has run under this protocol.

The system. The Ops Gravity of a 12-person agency's tool and automation estate.

1. Inventory and counting level. The declared counting level is automations, integrations and tools. The inventory at the start of the window holds 40 entries, so no finer counting is needed (IV.1).

2. Shadow metric, before scoring. From ticket data: 45% of last quarter's change capacity went to preservation. Upkeep outstanding: about 60 person-hours of postponed work, including three unpatched integrations and one automation known to fail at month-end. Source: tickets, classified at close.

3. Term scores with evidence.

TERM	SCORE	EVIDENCE
C	4	31 integrations between 14 tools; two CRMs overlap; a change to the lead form breaks reporting
D	4	14 SaaS vendors; three AI model calls on unpinned versions; one person holds admin rights to the automation platform
O	3	Major failures alert; minor ones are found by clients; no Silent-Failure Review has run, so O cannot score below 3
R	4	6 of 40 entries materially changed in 90 days, which is 15%
OC	2	No current Ownership Ledger. Named owners for about half the estate; two owners could not stop their automation during the audit

4. Calculation.

$$\text{Raw} = \frac{4 \times 4 \times 3 \times 4}{2^2} = 48 \qquad \text{Score} = 100 \times \left(\frac{\ln(48 \times 25)}{\ln(625 \times 25)} \right)^{1.199} = 69.1$$

5. Band. Fragility Zone (60–80): preserving outweighs building, and small changes risk breakage elsewhere. That fits the shadow figure of 45% consumed plus 60 hours owed.

6. Forces and Domains behind the high terms. Tool Gravity and Automation Gravity (C, D, O), Knowledge Gravity (one admin), Decision Gravity (half the estate unowned).

7. Counterweights selected under VIII.4.

- Minimum set: C4 Ownership Ledger, plus C5 Silent-Failure Review.
- For the terms at 4: C2 Duplication Tripwire for the overlapping CRMs (C), C7 Dependency Register with pinned model versions (D), and C6 Verification Budget (R).

8. Cadence. R 4 means a review every 30 days, plus event triggers (IX.1).

9. What would move the score. Raising OC from 2 to 3, by building the ledger and making both owners able to stop their automations, takes the score to 59.7, Operational Weight. Lowering D to 3 as well takes it to 56.4. Under VIII, neither counts until the next review shows the effect.

Version log, verification and licence

Canon v1.0 stays published as it was. v1.1 replaces it on publication, and every change is listed here.

AREA	V1.0	V1.1
Definition	Share of change capacity consumed	Upkeep consumed and upkeep outstanding, kept separate (I)
Laws	Ten Laws	Reworded for precision, three titles included; each tied to the term it acts through (III)
Terms	Named, not defined for scoring	Defined, with 1/3/5 anchors and evidence rules (II)
Rate of change	"Per quarter", undefined	Trailing 90 days, share of a capability inventory, defined material change (IV)
Formula	(C×D×O×R)/OC	Ownership squared as a declared prior, k = 1 tested alongside; full derivation of γ (V)
Bands	Ops: five bands. Code: Orbital, Drag, Sink, Event Horizon	One five-band scale for both instruments; bands classify burden, not failure odds (VI)
Ownership	One term	Accountable and operational owner; four tests; accountability sink (VII)

AREA	V1.0	V1.1
Counterweights	Six	Eight: C7 Dependency Register and C8 Living Documentation added; counterweight discipline (VIII)
Cadence	Quarterly, fixed	Set by R, with event triggers (IX)
Instruments	Code and Ops layers	Code and Ops Gravity as instruments; Forces and Domains mapped to terms; Agent Gravity previewed (X)
Validation	One-sentence rule	Shadow metric, blind scoring, kill conditions, pilot then external validation (XI)
Doctrine	Consequential Judgment stays human	Owner and operator defined; consequential judgment referenced from the Operator Kernel (XII)
Status	Presented as a measurement	Calibrated defaults pending validation; scores rank, not measure

Verification. Every figure in Parts V, VI and XIV can be reproduced with this script, which scores all 3,125 combinations of the five terms.

```
import math, itertools

def score(C, D, O, R, OC, k=2):
    lo, hi = 5 ** -k, 625               # lowest and highest possible Raw
    gamma = math.log(0.5) / math.log(math.log(81 / 3**k / lo) / math.log(hi / lo))
    raw = C * D * O * R / OC ** k
    return 100 * (math.log(raw / lo) / math.log(hi / lo)) ** gamma

print(round(score(3, 3, 3, 3, 3), 1))   # 50.0
print(round(score(4, 4, 3, 4, 2), 1))   # 69.1
for k in (2, 1):
    s = [score(*t, k) for t in itertools.product(range(1, 6), repeat=5)]
    print(k, [round(100 * sum(lo <= x < lo + 20 or (lo == 80 and x == 100) for x in s) / 3125, 1)
              for lo in (0, 20, 40, 60, 80)])
```

Share of all 3,125 combinations in each band. This describes the scale, not how common each band is in real systems.

BAND	K = 2	K = 1
Light (0–20)	4.7%	7.2%
Manageable Drag (20–40)	30.7%	33.0%
Operational Weight (40–60)	40.9%	40.2%
Fragility Zone (60–80)	19.6%	17.0%
Collapse Risk (80–100)	4.1%	2.6%

Licence. Licensed under CC BY-ND 4.0. You may copy and redistribute this canon with attribution to Cognitive Empire Systems Ltd and the canonical URL. You may not distribute modified versions.

Cite as: Maghazi, E. (2026). *Maintenance Gravity Canon — Public Edition v1.1.* Cognitive Empire Systems Ltd.