Why Effective Program Management Is a Control Systems Problem

Governance, Efficiency, and Value Delivery in an AI-Accelerated World

Andrew Thillainathan

Senior Program & Portfolio Management Leader

February 2026

A systems-engineering perspective on execution, governance, and value preservation

Version 1.1 — March 2026
(Includes Addendum on Governance Calibration and Net Execution Performance)

Foreword

Human performance can no longer keep pace with the accelerating advancement of technology. Artificial intelligence is now penetrating the workforce both deeply and broadly, fundamentally reshaping how work is conceived, planned, and executed. Activities that once depended on human judgment, coordination, and experience are increasingly augmented—or entirely transformed—by AI-driven systems.

The result has been extraordinary gains in productivity and efficiency. Yet these gains introduce a new equilibrium challenge: while machines scale rapidly, human systems—organizational structures, governance models, and decision frameworks—must adapt just as quickly to remain effective. The gap between technological capability and human execution is no longer theoretical; it is now a defining leadership challenge.

Nowhere is this tension more visible than in large programs and enterprise initiatives. As planning accelerates and expectations rise, the traditional focus on individual performance, effort, or intent becomes insufficient. This tension is especially visible in capital-intensive, technology-driven environments—such as cloud platforms, AI infrastructure, and data center–scale systems—where planning velocity has increased dramatically while execution complexity continues to rise. Leaders are increasingly challenged to recalibrate how people, processes, and tools combine to deliver value—at speed, at scale, and with discipline. The emphasis is shifting from managing activity to improving the efficiency, stability, and reliability of execution itself.

This paper is written for project managers, engineers, architects, and senior leaders—particularly executives responsible for complex portfolios and enterprise-scale transformation. It takes an engineering perspective on program execution, governance, and efficiency, reframing program performance not as a reflection of individual capability, but as a measurable system property.

Rather than asking *who* failed, this paper encourages leaders to ask *where* the system is losing efficiency—and how governance, feedback, and control mechanisms can be designed to restore stability and maximize value delivery in an AI-accelerated world.

1. Why This Paper Exists

Artificial intelligence is fundamentally reshaping how programs are initiated and planned. Activities that once defined the most human-intensive and judgment-heavy phase of program management—scope definition, estimation, dependency analysis, sequencing, and risk identification—are increasingly automated or AI-assisted. What previously required weeks of coordinated human effort can now be produced in minutes.

At the same time, organizations face sustained workforce pressure. AI adoption and economic constraints are reducing available human capacity while expectations for speed, scale, and value delivery continue to rise. Programs are now expected to deliver **more with fewer people** as a standing operating condition, not a temporary constraint. Together, these forces have shifted where value is lost.

Today, the primary constraint in program success is no longer planning speed or clarity of intent. It is **execution efficiency**—how effectively business intent is transformed into realized business impact once execution begins.

Despite decades of investment in mature frameworks, methodologies, and certifications, this gap persists. PMI research consistently shows that only about **half of initiatives deliver their intended value**, and that organizations lose **approximately $100 million for every $1 billion invested** due to poor execution outcomes—even in environments with experienced professionals and established processes.

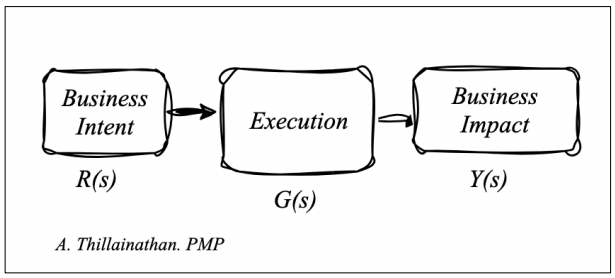
If planning is faster and frameworks are mature, why does so much value still evaporate during execution?

2. Execution as a Transformation System

At its simplest level, execution can be viewed as a transformation:

Business Intent → Execution → Business Impact

Figure 1. Basic execution flow (Open-loop diagram)



In this model, business intent enters execution as an input. Business impact emerges as an output. What happens in between determines how much value is ultimately delivered.

Critically, this is an open-loop system and is represented by the equation:

$$|Y(s)| = |G(s)| \cdot |R(s)|$$

Once execution G(s) begins, intent R(s) flows forward without systematic correction. Outcomes Y(s) are observed only after work has progressed, often too late to prevent loss. Any resistance that arises—dependencies, coordination overhead, decision latency, resource contention, or rework—acts directly on execution with no built-in mechanism to counter it.

As a result, execution is not a lossless process. Delivered impact is almost always less than planned intent, not because people lack effort, but because the system has no way to correct itself while work is underway.

This distinction matters. In open-loop execution, drift is not an anomaly—it is the expected behavior of the system. Value is preserved or lost based on how execution responds to resistance, not on how clearly intent was defined.

Crucially, this loss is not primarily a people problem. It is a system behavior.

3. Efficiency (η): A System Property

To reason about execution rigorously, we introduce a single governing concept: **efficiency**.

$$\eta = \frac{|Y|}{|R|}$$

Where:

- **R** represents planned business intent (scope, urgency, expected value),
- **Y** represents delivered business impact,
- **η** represents execution efficiency.

Efficiency captures the fraction of planned value that survives execution. This yields a simple but powerful relationship:

$$Y=\eta \cdot R$$

Delivered impact equals planned intent scaled by system efficiency. This reframing immediately changes the leadership lens:

> Increasing urgency, effort, or activity increases R. Only the execution system determines η.

As AI accelerates planning and increases the volume and frequency of intent entering execution, efficiency—not effort—becomes the scarce resource.

4. Governance as a Control Layer (Classical Control Systems Theory)

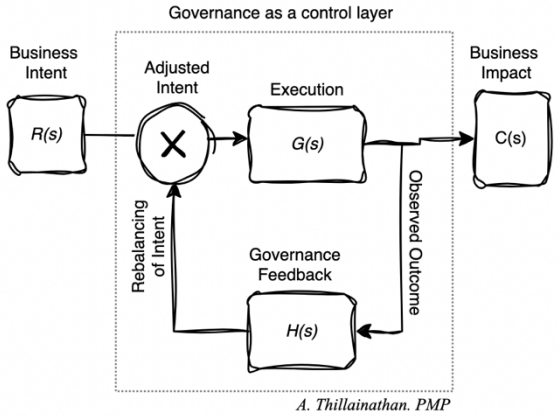
To understand how efficiency can be protected, we borrow explicitly from classical control systems theory—a discipline used for decades in engineering domains such as aviation, power systems, and industrial automation to manage complex, disturbance-prone systems.

Classical control theory provides a formal framework for:

- understanding how inputs are transformed into outputs
- how disturbances degrade performance
- and how feedback stabilizes system behavior

When applied to program execution, governance functions as a control layer: a feedback mechanism that observes outcomes and corrects execution behavior before deviation compounds.

Figure 2. Governance as a Control Layer (Closed-Loop Contextual Diagram)



Note: This paper adopts a negative-feedback formulation for governance. While real programs may exhibit mixed or nonlinear feedback effects, analyzing those dynamics is beyond scope. Negative feedback best represents governance's designed role: error correction, stabilization, and reduction of execution variance.

Using a standard closed-loop control formulation, the relationship between intent and impact can be expressed as:

$$C(s) = \frac{G(s)}{1 + G(s).H(s)}.R(s)$$

Where:

- *R(s)* is business intent,
- *G(s)* represents execution dynamics (capacity, resistance, latency),
- H(s) represents governance feedback (decision authority, cadence, enforcement),
- C(s) is delivered business impact.

While the notation is drawn from control theory, the intuition is simple:

> feedback reduces the loss of intended outcomes caused by real-world resistance.

5. An Intuitive Analogy: Governance as Flight Control

Consider an airplane climbing after takeoff.

A pilot commands a climb rate of 1,000 feet per minute. In practice, aircraft weight, air density, thrust limits, and

aerodynamic constraints determine how much climb can actually be achieved at that moment.

In an open-loop situation—where the aircraft simply responds to the initial command without continuous correction—those disturbances act directly. If conditions change, the aircraft does not compensate. The result might be:

- 1,000 ft/min planned,
- **700–800 ft/min achieved**, or in severe cases,
- an unstable climb profile.

Modern aircraft do not operate this way.

In a closed-loop control system, sensors continuously measure actual climb performance—vertical speed, airspeed, pitch attitude, and engine response. That observed performance is fed back into the flight control and autopilot systems, which make ongoing adjustments to pitch, thrust, and trim to maintain a stable and efficient climb within physical limits.

The system does not eliminate constraints. It works within them. But by continuously correcting for deviations, it prevents errors from compounding.

As a result:

- the aircraft may still not achieve the full 1,000 ft/min under given conditions,
- but it can often stabilize much closer—**900 ft/min instead of 700**—while maintaining control margins and efficiency.

The intent did not change. What changed was how effectively the system corrected for disturbance.

Program execution behaves the same way. Governance does not guarantee perfect outcomes. It continuously adjusts priorities, resolves constraints, and corrects drift so that unavoidable resistance does not compound into unnecessary value loss.

Governance, like flight control, does not push harder. It keeps the system stable under pressure.

6. Open-Loop vs. Closed-Loop Execution

The airplane analogy illustrates a simple principle: when a system encounters real-world resistance, outcomes depend on whether there is a mechanism to observe performance and correct behavior while execution is underway.

In control-systems terms, this distinction is described as open-loop versus closed-loop operation.

In open-loop execution, business intent enters the system and progresses forward with little or no correction:

$$Y(s)=G(s)\cdot R(s)$$

Execution dynamics G(s) — dependencies, coordination overhead, decision latency, and rework—act directly on the work. Any disturbance degrades outcomes immediately. Drift accumulates unchecked. Programs may appear busy and well-reported while value quietly erodes, as lower-value work consumes capacity alongside—or ahead of—higher-value items.

In closed-loop execution, governance introduces feedback that continuously observes outcomes and corrects execution behavior:

$$C(s) = \frac{G(s)}{1 + G(s).H(s)}.R(s)$$

The combined effect of execution dynamics and governance feedback can be expressed as a single, observable quantity:

$$\eta = \frac{G(s)}{1 + G(s).H(s)}$$

So delivered business impact can be written simply as:

$$C(s) = \eta \cdot R(s)$$

In this form, efficiency η represents how much planned value survives execution once resistance and feedback interact. Governance does not eliminate resistance; it reduces the system's sensitivity to it by shortening correction cycles and enforcing alignment.

The practical difference between success and failure is therefore not effort or urgency, but the strength and timeliness of feedback

A Simple Physical Example: An Automatic Paint Shop
Consider an automatic automotive paint shop. Fifty unpainted cars enter the plant. The paint process operates without in-process sensing or correction.

When the cars exit, forty are painted correctly on the first pass, while ten require rework.

From a control-systems perspective, this is an open-loop system.

- Input = 50 cars
- Correct output = 40 cars

The open-loop efficiency is therefore:

$$\eta = 40/50 = 0.80$$

The plant has not failed. The equipment operated as designed. Yet 20% of output required rework because errors were detected only after the process completed.

Now consider the same plant operating with closed-loop control. Sensors monitor paint thickness and coverage in real time. Spray parameters are adjusted dynamically as cars move through the line.

With feedback applied, forty-eight cars are painted correctly on the first pass, and only two require rework.

- Input = 50 cars
- Correct output = 48 cars

The closed-loop efficiency becomes:

$$\eta = 48/50 = 0.96$$

The underlying plant has not changed. What changed is when and how errors were detected and corrected. Feedback prevented defects from propagating to the output.

Program execution behaves the same way. Open-loop execution favors throughput and apparent progress, but allows errors and misalignment to surface only after value is lost. Closed-loop governance does not eliminate constraints—it detects drift during execution and corrects it before loss compounds.

The distinction between open- and closed-loop systems is not effort or intent, but whether correction occurs during execution or only after outcomes are realized.

7. What Governance Feedback Actually Does

The distinction between open-loop and closed-loop execution explains why governance matters. The next question is more practical: what does governance actually do inside the feedback loop to improve outcomes?

A critical clarification is required.

Governance feedback does **not** primarily operate by reducing ambition or arbitrarily cutting scope. Most of the time, effective governance functions to **preserve original intent** by correcting execution behavior as resistance emerges.

In practice, this means:

- clearing roadblocks that stall high-value work
- accelerating decisions to reduce execution latency
- enforcing priorities and sequencing when work begins to drift
- reallocating constrained capacity to protect value density

These actions strengthen feedback in the system, allowing execution to respond to disturbance before loss compounds.

Only when resistance exceeds the system's ability to self-correct does governance deliberately adjust intent—through scope, sequence, or timing changes—to restore stability.

In engineering terms, this is not failure. It is stabilization.

8. Efficiency (η) as an Organizational Maturity Signal

If governance feedback operates by correcting execution behavior in real time, the next question becomes: **how can leaders tell whether those corrections are effective?**

The answer is efficiency.

Efficiency (η) is not a trait of individuals or teams. It is a **measurable property of the execution system**—the cumulative result of how execution dynamics and governance feedback interact over time.

Because efficiency is defined as a ratio, it can be observed and tracked as execution unfolds:

$$\eta(t) = \frac{Delivered\ Value(t)}{Planned\ Value(t)}$$

Efficiency captures how much planned value is surviving the execution process at any given point in time.

Importantly, the most meaningful signal is not a single snapshot of efficiency, but its **trend**:

$$\frac{d\eta}{dt}$$

A sustained decline in efficiency is often the **earliest indicator of systemic instability**—appearing well before schedules slip, budgets overrun, or formal milestones are missed. By the time those symptoms are visible, efficiency has often been degrading quietly for weeks.

Mature organizations distinguish themselves not by occasional peak performance, but by a **higher and more stable floor of efficiency across programs.** That stability reflects the presence of strong, timely governance feedback capable of correcting drift before value is lost.

9. Bringing It Together: A Real World Program Execution Example

Consider a mid-sized enterprise modernization program. AI accelerates planning, and execution begins with high urgency. The planned impact is shown in table-1

Table 1. Planned Business Impact(weighted)

Work Type	Items	Weight	Planned Value
Foundational / hygiene	40	1	40
Operational improvements	40	2	80
Strategic / regulatory	20	5	100
Total	**100**		**220**

As execution progresses, resistance emerges. Dependencies slow complex work. Decision cycles lengthen. Coordination overhead increases.

In response, teams continue delivering what remains feasible. Lower-complexity work progresses while higher-value items increasingly stall. This behavior reflects rational execution under constraint, not a lack of effort or intent. It is an emergent property of the system. The delivered impact is shown in table 2;

Table 2. Delivered Business Impact

Work Type	Items	Weight	Delivered Value
Foundational / hygiene	35	1	35
Operational improvements	30	2	60
Strategic / regulatory	5	5	25
Total	**70**		**120**

Observed efficiency:

$$\eta = \frac{120}{220} = 0.55$$

> Nearly half of planned value was lost—not through failure, but through **systemic drift**.

In this example, **120 units of value were delivered out of 220 planned**, yielding an observed execution efficiency of η=0.55

From a control-systems perspective, this outcome maps directly to the closed-loop relationship:

$$C(s) = \frac{G(s)}{1 + G(s).H(s)}.R(s)$$

Here:

- **R(s)** is the planned business intent (220 weighted units of value),
- **G(s)** reflects execution dynamics once work begins—dependencies, coordination overhead, and decision latency that biased delivery toward easier, lower-value work,
- **H(s)** represents governance feedback—its ability to enforce prioritization, clear bottlenecks, and correct drift.

In this case, execution resistance dominated governance feedback. As a result, only **55% of planned value survived execution**. The system did not fail catastrophically—it **drifted**, which is the expected behavior of a weakly closed-loop system.

Stronger governance feedback would not have increased effort or urgency, but it would have reduced sensitivity to

resistance—raising efficiency and preserving more of the original intent.

Execution efficiency (η) is a lagging metric—it can only be measured once outcomes are observed. However, like all control systems, efficiency is shaped in real time by the system's dynamics and feedback. Governance does not wait to see final results; it monitors leading indicators of drift and intervenes early to prevent efficiency loss from compounding.

10. From Managing People to Engineering Systems

As AI permeates execution, the management problem itself changes.

Leaders are no longer primarily managing individual effort. They are managing **systems** that convert intent into outcomes. The most productive leadership questions are therefore diagnostic, not personal:

- **What is degrading efficiency in our execution system?**
- **Where are we allowing execution to drift without correction?**

That shift—from blame to system diagnosis—is transformative.

11. Closing: Governance as the Maturity Engine of the Enterprise

PMI's findings make one conclusion unavoidable: value loss persists despite mature frameworks. The problem is not the absence of planning discipline—it is the absence of effective control once execution begins.

A well-defined governance model, designed as a control system, is what allows an organization to stabilize execution under pressure, preserve value density, and consistently convert intent into impact.

In an AI-accelerated world, AI increases the amplitude and frequency of intent entering execution. Without strong feedback, that pressure destabilizes delivery. With it, organizations can safely absorb more change, faster, with fewer people.

This reframes leadership responsibility.

Leaders are not responsible for pushing harder. They are responsible for designing and tuning the system.

In an AI-accelerated world, efficiency—not activity—is the scarce resource.

Governance is how efficiency is protected.

Addendum:

1. Reconciling Governance and Execution Calibration

In the original paper, execution was modeled as a closed-loop system where realized performance depends on both execution capability (G) and governance intensity (H). However, in practice, G and H cannot be independently observed or directly solved without knowing the other. This creates a practical limitation: while the model explains behavior, it does not by itself provide an intuitive way to reason about optimal governance levels.

This addendum extends the original formulation by normalizing execution capability and decomposing the system into intuitive components—throughput, stabilization, and coordination cost—allowing us to understand how governance both improves and degrades execution. This leads to a more practical interpretation: governance must be calibrated, not maximized.

2. Original closed-loop expression

In the paper, execution performance was expressed as:

$$\eta = \frac{G}{1+GH}$$

Where:

- **G** = execution capability
- **H** = governance intensity
- **η** = realized execution performance

This equation states that as governance intensity (**H**) increases, the system becomes more controlled, but governance also dampens(degrades) raw throughput. To simplify, let us **normalize** execution capability to:

$$G=1$$

This gives:

$$\eta = \frac{1}{1+H}$$

This term represents the share of raw execution throughput that remains after governance damping.
Examples:

- If **H = 0**, then η=1. No governance damping.
- If **H = 1**, then η=0.5. Half the raw throughput remains.
- If **H = 4**, then η=0.2. Only 20% remains.

If $\eta = \frac{1}{1+H}$ is the portion of execution throughput remaining after governance damping, then the remainder is the portion of governance that is contributing toward correction and stabilization.

We subtract from 1:

$1 - \frac{1}{1+H} = \frac{H}{1+H}$ represents the **stabilization benefit of governance**.

Examples:

- If **H = 0**, benefit = 0
- If **H = 1**, benefit = 0.5
- If **H = 4**, benefit = 0.8

This rises quickly at first, then levels off. That makes intuitive sense: governance helps more at the beginning, but each extra layer helps less than the one before.

3. Why do we need a cost term?

Governance does not come free. More governance usually means more:

- Meetings
- reporting
- approvals
- escalations
- coordination work

That extra work is overhead. To represent it simply, define governance cost as:

$$C(H)=kH$$

Where:

- **C(H)** = governance cost
- **k** = friction coefficient
- **H** = governance intensity

Each additional unit of governance adds some additional overhead.

For example, if **k = 0.08**, then every 1-unit increase in H adds 0.08 units of coordination cost.

4. Net execution performance

Net execution performance becomes:

$$\eta \text{ net}= \frac{H}{1+H} - kH$$

This is the inverted-U equation.

- At **low H**, governance begins to reduce drift, so performance rises.
- At **moderate H**, stabilization is strong and overhead is still manageable, so performance peaks.
- At **high H**, overhead grows faster than the remaining benefit, so performance falls.

That is why governance should be **calibrated**, not **maximized**.

Table A1 Governance and Net Execution Performance Calculations

Governance Intensity (H)	Execution Throughput 1/(1+H)	Alignment Effectiveness H/(1+H)	Total = 1	Governance Cost (kH)=0.08	Net Execution Performance [H/(1+H)] - kH
0	1	0	1	0	0
0.5	0.667	0.333	1	0.04	0.293
1	0.5	0.5	1	0.08	0.42
1.5	0.4	0.6	1	0.12	0.48
2	0.333	0.667	1	0.16	0.507
2.5	0.286	0.714	1	0.2	0.514
3	0.25	0.75	1	0.24	0.51
3.5	0.222	0.778	1	0.28	0.498
4	0.2	0.8	1	0.32	0.48
5	0.167	0.833	1	0.4	0.433

6	0.143	0.857	1	0.48	0.377
8	0.111	0.889	1	0.64	0.249
10	0.091	0.909	1	0.8	0.109

The table illustrates how governance reallocates system capacity between execution throughput and alignment effectiveness, while coordination cost reduces net realized performance.

Figure A2 *As governance intensity increases, net performance decreases after reaching optimal level*

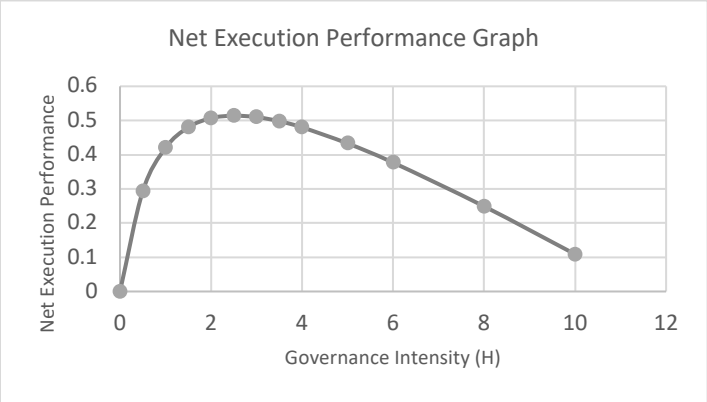
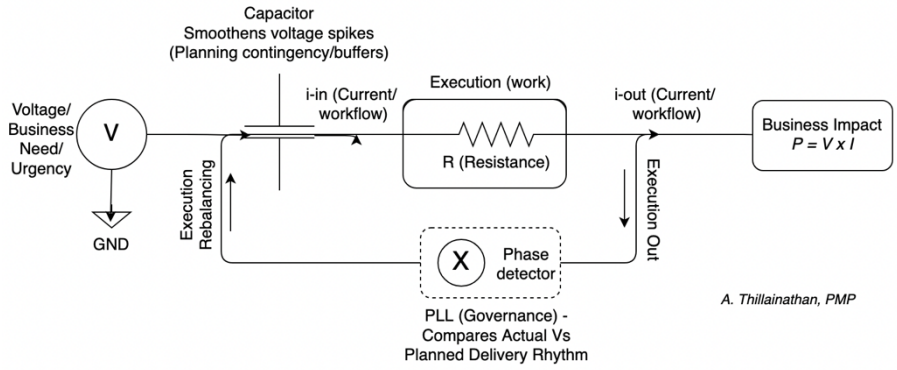


Figure A2 illustrates the inverted-U relationship between governance intensity and net execution performance.

Appendix A:

A Control-Systems Analogy for Program Execution and Governance

Figure A-1. Program Execution as a Control System (Conceptual Analogy, Notional Mapping)



This appendix introduces a simplified control-systems analogy to clarify the dynamics of program execution and governance, particularly for readers with an engineering background. The diagram borrows familiar elements from electrical and control systems—such as voltage, current, resistance, capacitance, and feedback loops—not as a literal circuit design, but as a conceptual mapping between physical system behavior and organizational execution.

In this model, **business urgency** is represented as voltage (V). Urgency applies pressure and establishes potential, but does not directly produce outcomes. Higher urgency increases possible flow, yet impact emerges only when work is executed.

A **capacitive element** placed ahead of execution represents organizational buffering and absorptive capacity—such as planning contingency, intake controls, architectural readiness, and capacity slack. This buffering smooths short-term volatility in demand so execution remains stable rather than reactive. While classical phase-locked loops employ capacitive filtering within the feedback path, the capacitive element shown here operates upstream of execution, shaping how intent enters the system rather than how feedback is processed.

Execution inherently introduces **resistance (R)**. Dependencies, coordination overhead, decision latency, and skill constraints oppose flow once work begins. This resistance is unavoidable and becomes visible only during execution. The interaction of urgency, buffering, and resistance determines actual delivery flow (I)—what is completed in practice, not what was planned.

Governance is represented as a feedback mechanism analogous to a phase-locked loop (PLL). Rather than functioning as oversight or inspection, governance observes execution output, compares actual delivery rhythm against planned intent, and adjusts execution input when sustained deviation appears. By correcting input rather than intervening directly in execution mechanics, governance stabilizes delivery cadence and protects high-value outcomes under pressure.

References

Project Management Institute (PMI). Pulse of the Profession®. Project Management Institute.

Project Management Institute (PMI). Organizational Project Management Maturity Model (OPM3®). Project Management Institute.

Project Management Institute (PMI). AI Innovators: Cracking the Code on Project Performance. Project Management Institute.

McKinsey Global Institute. The Economic Potential of Generative AI. McKinsey & Company.

Ogata, K. Modern Control Engineering. Pearson Education.

Drafting and editorial assistance for this paper was supported by AI-assisted tools. Responsibility for all content, interpretations, and conclusions rests solely with the author.