

Viability Precedes Authority

The Law of Viability and Viability Engineering

Eliel de Siqueira Valença

February 4, 2026

Abstract

Abstract. We propose the Viability Law as a guiding principle for software engineering in the AI-driven era of abundant code production: value does not come from execution volume, but from the precision of Intention and the sustainability of Design. Building on this postulate, we introduce the IDO paradigm (Intention, Design, and Orchestration) and suggest an effectiveness measure ($E_s = I \times (D + O)$) to make explicit the shift of the bottleneck from code writing to decision-making. Finally, we present the Technical Leadership Matrix (2×2), which structures the tension between time-to-market and technical quality, defines a Convergence Point, and characterizes four operational zones (Excellence, High Performance, Innovation, and Crisis), guiding leadership interventions to preserve viability and prevent entropy.

Executive Summary

The Problem: The Virtual Productivity Paradox

The industrialization of software created a dangerous fetish: confusing motion with progress. In today's landscape—saturated with vanity metrics and an abundance of AI-generated code—excess execution without direction becomes “industrial waste,” accelerating entropy and pushing the system toward the **Crisis Zone**.

The Thesis: Viability Precedes Authority

This essay argues that software development is governed by natural laws that are indifferent to hierarchy. Authority does not solve technical problems; only **Viability Engineering** can ensure that abundant execution is filtered by strong Intention and sustained by resilient Design.

The Strategic Framework

To govern this new era, we propose practical tools for diagnosis and action:

- **IDO Paradigm:** Shifts the human role from writing code to **Intention, Design, and Orchestration**.
- **Effectiveness Formula** ($E_s = I \times (D + O)$): Makes explicit that if Intention (I) is zero, the final value will be zero—regardless of execution speed.

- **Technical Leadership Matrix (2×2):** Defines the **Convergence Point** as the pragmatic target where business and technology stop being adversaries and start generating profit and sustainability.

Implementation: The 4 Pillars of Viability Engineering

Technical leadership must act as the guardian of systemic integrity across four structural fronts:

1. **Clarity:** Mapping reality to dissipate technical fog.
2. **Intentionality:** Aligning every generated byte with the commercial “why.”
3. **Integrity:** Using guardrails and architecture to protect the software's future.
4. **Adaptability:** Promoting continuous evolution to remain competitive.

Conclusion for Decision-Makers

Viability Engineering is not an administrative process, but a shift in governance: from execution as the center to **decision as the center**. Mastering this framework enables an organization to convert AI volume into enduring value, ensuring that software survives its own creation.

1 Contributions

This essay organizes a practical thesis for technical leadership in the post-AI era and proposes a vocabulary to align executive decision-making with engineering reality. Here, *technical leadership* refers to any role with title or influence over engineering or architectural decisions (e.g., tech lead, engineering manager/tech manager, staff/principal, CTO, etc.). The central contributions are:

1. **The Law of Viability:** a formulation that separates *movement* from *progress*, arguing that sustainable value depends on the precision of intention and the sustainability of design.
2. **The Software Value Postulate:** a conceptual model that makes explicit how Abundance of Execution can reduce value when execution is not filtered by intention and design.
3. **The IDO Paradigm (Intention, Design, and Orchestration):** a taxonomy to guide human work when code production becomes a commodity.
4. **Technical Leadership Matrix (2×2):** a diagnostic and communication instrument that makes explicit the trade-off between Time-to-Market (TTM) and Technical Quality / Maturity, defining the Convergence Point and four operational zones.
5. **Viability Engineering:** an operational formulation organized into **4 Pillars** — **Clarity**, **Intentionality**, **Integrity**, and **Adaptability** — to guide technical leadership in the era of abundance.
6. **Operational vectors:** (i) the Stabilization Vector for crisis interventions and (ii) the Maturation Vector to model the accumulation of quality over time, evidencing technical leadership as a necessary force.

Essay status and next steps

This text is initially published as a market-oriented essay: it prioritizes clarity, usefulness, and explanatory power for technically sovereign leadership and executive stakeholders. The equations and vectors presented here should be read as conceptual models; subsequent research aims to operationalize definitions, propose measurable proxies, and test the hypotheses implied by the framework.

As part of this evolution, we are updating the terminology: the term *Viable Value Delivery* (VVD) is being replaced by *Viability Engineering*. Consequently, the *Viable Value Delivery cycle* is being replaced by the *4 Pillars of Viability Engineering*, to more precisely reflect the structural (and not merely procedural) nature of technical leadership work.

As an evolution agenda, future versions should: (i) define metrics for Intention, Design, Abundance of Execution, and Quality; (ii) include case studies and/or retrospective analyses; and (iii) delimit applicability conditions and model limits. These steps prepare the transition from an essay to an article and, potentially, to a master's dissertation.

2 The axiom

2.1 Viability Precedes Authority

Software development is a discipline governed by natural laws that impose insurmountable limits on authority. No matter the title or the budget: technical reality is indifferent to hierarchy.

If a system was not designed to scale, an unlimited credit card will not save it from collapse; it will only finance a more expensive failure. If a capability requires a minimum maturation time, forcing an arbitrary deadline does not bring value forward; it only accelerates chaos and failure.

These examples reveal an inconvenient truth: authority does not solve technical problems. Technical problems require technology — defined here in its purest form: applied knowledge.

The current euphoria around Artificial Intelligence seems to ignore this law. Producing code at high speed without Intention and Design is merely industrializing waste. Code has become an abundant commodity; human sovereignty, therefore, has shifted to the ability to assign meaning and sustainability to that raw material.

Agile development promised to solve the delivery bottleneck, but failed by separating technical authority from strategic decision-making. Part of the blame lies with engineers, who often pursued academic perfection while the market demanded viability. On the other hand, the business pursued immediate profit through unpayable technical debt and exhausted teams.

The result is a broken model. To restore industry sanity, we must transition from “delivery at any cost” to **Viability Engineering**.

Viability is not an implementation detail; it is the prerequisite for the product’s existence. And it can only be achieved if we restore the autonomy of technical leadership — not as a privilege, but as the only safeguard against the obsolescence of the business itself.

3 The Virtual Productivity paradox

The industrialization of software development created a dangerous fetish: the confusion between motion and progress. In today’s landscape, saturated with vanity metrics, excessive code output and high Jira ticket throughput have become indicators of a productivity we call “virtual”. It exists in reports, but it is invisible in the company’s value balance [1, 2].

3.1 The Scale of Waste

Producing what does not need to be built is not merely neutral; it is an act of sabotage against viability. When we spend energy on initiatives misaligned with business Intention, we are not only losing money: we are consuming the opportunity time that should be invested in maintaining and evolving vital infrastructure. Misallocated effort generates an Opportunity Debt that is often unpayable.

3.2 AI: the entropy accelerator

Artificial Intelligence emerges in this scenario as a force multiplier, but without a direction vector. If Intention and Design are not present to filter execution, AI becomes a noise factory. Increasing code output without the protection of technically sovereign leadership does not accelerate value delivery; it accelerates the system’s drift toward the Crisis Zone (Q4). In the era of abundant raw material, excessive purposeless code is the new Industrial Waste of Technology.

3.3 The mirage of instant delivery

The paradox manifests cruelly in the “Friday release that takes down the system on Monday”. In the Virtual Productivity model, that release is counted as success at the end of the week. Under the lens of Viability, however, value that does not sustain over time is nonexistent value — it is merely a cost disguised as progress.

Without the sustainability guaranteed by Design, what we call “delivery” is, in fact, a transfer of risk from engineering to operations.

3.4 The cost of disorientation

Without a Convergence Point between Intention and Viability, development becomes a directionless activity. We spend the brightest talents and the most expensive resources to build software monuments that nobody inhabits or that collapse at the first sign of scale. True productivity is not measured by the volume of raw material processed, but by the precision with which we transform intention into lasting reality.

4 The Law of Viability

At the center of the transition to post-AI engineering, we establish a fundamental postulate that must govern the conduct of all technically sovereign leadership:

“The value of software does not lie in the abundance of its execution, but in the precision of its intention and the sustainability of its design.”

— Valença, Eliel de Siqueira (2026)

$$V_{\text{software}} \propto \frac{P_{\text{Intention}} \cdot S_{\text{Design}}}{E_{\text{Execution Abundance}}} \quad (1)$$

The equation above formalizes the central idea of this section: value is not synonymous with execution volume. The numerator represents *direction quality* (Precision of Intention) and *sustaining capacity* (Sustainability of Design); the denominator represents the *entropy of abundance*, i.e., the systemic cost of producing a lot (and fast) when execution is not filtered by intention and design. Thus, when intention is imprecise (or null), any acceleration of production — including via AI — amplifies waste

and risk; and, without sustainable design, value appears to exist in the short term but turns into operational entropy, transferring risk from engineering to operations.

4.1 The commoditization of raw material

Historically, code was treated as the final artifact of our work. Due to its intrinsic complexity, it took the form of a product when, in fact, it was always only a means to an end. Artificial Intelligence removed the veil of that illusion. By making code writing fast, cheap, and virtually inexhaustible, AI transformed code into what it truly is: commodity raw material.

4.2 The bottleneck shift

Code is no longer the production bottleneck, but its abundance revealed the true and far more dangerous bottleneck: decision-making cognitive capacity. Infinite execution without a clear north is not productivity; it is systemic noise. The modern leadership challenge is no longer “how to write”, but “what should exist” and “how to ensure it keeps existing”.

4.3 Governing by viability

Guiding development through the Law of Viability means accepting that accelerating for acceleration’s sake is, by definition, not viable. The role of technical leadership is to be the guardian of this balance: to ensure execution is fast enough for the business, but safe enough for the product’s survival.

Real value is not in how much code a team can “spit out” per sprint, but in how little code it needs to materialize a powerful and sustainable intention.

5 The IDO Paradigm

To define viability, we use the IDO Paradigm, which proposes three taxonomies for organizing human work in the AI era: Intention, Design, and Orchestration. These values will be used in the Software Effectiveness formula.

Without the IDO pillars, software development becomes a game of chance with company resources.

Intention is what separates useful functionality from technical waste: it is the surgical precision of understanding the “why” before authorizing the “how”.

Design is the protective membrane: it ensures that execution speed does not break the minimum quality line, turning today’s speed into tomorrow’s collapse. It enables safe industrial-scale code production.

Orchestration is the human activity of conducting AI agents: the person no longer as executor, but as the operator of the automation system.

With that, we can compute the effectiveness of software.

5.1 Software effectiveness

Software effectiveness can be calculated as follows:

$$E_s = I \times (D + O) \quad (2)$$

Note that if Intention is zero, the quantity or quality of Design and the output produced by Orchestration will be zero. As the Law of Viability states, value lies in the precision of Intention and the sustainability of Design.

6 The Technical Leadership Matrix

The Technical Leadership Matrix is a bidimensional (2×2) model designed to map the boundaries of viability and the cost of technical failure. It does not merely organize chaos: it reveals the invisible tensions between market need and code sustainability.

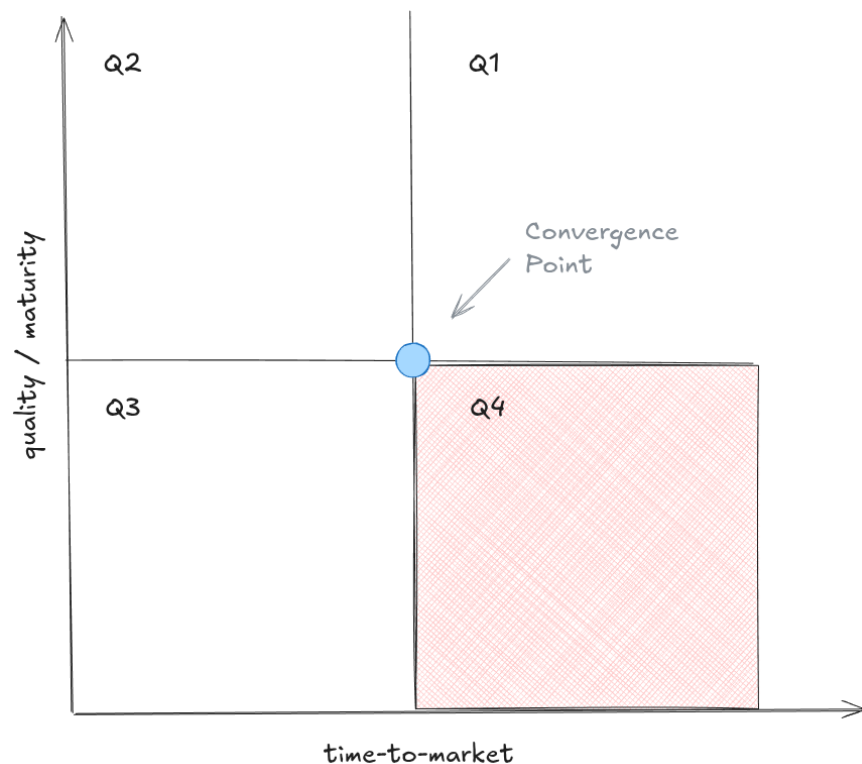


Figure 1: Technical Leadership Matrix.

6.1 Axes: the orthogonality of viability

For a matrix to be analytically useful, its variables must be independent. In our structure:

- **X Axis (Time-to-Market):** represents the opportunity window and business direction. It is the force that pushes the product into the world.
- **Y Axis (Technical Quality / Maturity):** represents systemic integrity and robustness. It is the force that ensures the product remains in the world.

First definition. Time without intention does not generate maturity. Maturity is not a passive byproduct of time; it is the result of deliberate effort by technical leadership to ensure the product evolves toward minimum levels of viability.

6.2 Boundaries: cut lines

The matrix stops being abstract when we draw the boundaries of the compromise:

- **Vertical line (Business Limit):** delimits the minimum Time-to-Market for delivery to generate perceived value. This pressure point is defined by the business.
- **Horizontal line (Minimum Quality Floor):** establishes the minimum quality and maturity threshold below which the system becomes non-viable, whether due to maintenance cost or operational risk. This “floor” is defined by technical leadership.

6.3 The utopia

Unlike classic managerial matrices, the upper end of Q1 (Maximum Quality + Maximum Time) is not the target. The blind pursuit of the top is what distanced engineering from the business and isolated technical directorates.

Technical perfection has an opportunity cost: while engineering polished the code, the market changed. Therefore, we establish a new axiom for the modern leader:

“Excess quality becomes margin for the competition.”

— Valença, Eliel de Siqueira (2026)

A company may have capital to finance perfection, but this strategy remains valid only until a competitor appears delivering “good enough” for a fraction of the price and time.

Pragmatism, therefore, is the search for the Convergence Point — the place where technical effort and business need meet to generate profit, not merely code. It is the target.

Q1 lies above the Convergence Point and is, therefore, a strategic target reserved for precision engineering. Some teams and companies may need to operate at higher levels of quality and maturity (e.g., deep tech, R&D, high-risk software). However, within those same organizations, other teams need the pragmatic view of the Convergence Point to sustain an efficient business model; thus, Q1 is a quadrant of strategic exception.

6.4 The Convergence Point

The intersection of the horizontal (Minimum Quality) and vertical (Time-to-Market) lines is not a mere graphical accident; it is where plans become possible. It is the exact coordinate of viability: the Convergence Point.

In this space, the two forces — business and technology — abandon idealism and yield pragmatically. The business accepts that there is no “zero deadline” without sacrificing product survival, and technology accepts that “infinite perfection” is the fastest path to commercial irrelevance.

6.5 The intentional work of leadership

Unlike other indicators that may fluctuate by chance, the Convergence Point is not reached by inertia. As established, time is indifferent to maturity. Reaching this target and, more importantly, evolving with it requires intentional work by technical leadership.

Moving the Convergence Point means:

- continuously mapping where the cut lines should be;
- negotiating viability when external pressures try to push the point outside the safety zone;
- ensuring that technical execution does not merely “deliver code”, but fulfills the maturity requirements needed for that specific moment in the product life cycle.

It is at this point — and only at this point — that technology and business stop being adversaries in a zero-sum game and become allies in a structure of sustainable growth.

6.6 Quadrants

By crossing the cut lines, we define four possible states for development. While the first three are operational states with distinct purposes, the fourth represents the system’s pathology: the failure of technical sovereignty.

Q1: Excellence Zone. Profile: High Quality / High TTM. **Where it lives:** R&D departments, deep tech, critical domains (Healthcare, Aerospace). **Reality:** this is the long-term investment quadrant. It operates above the Convergence Point because error risk outweighs time cost. However, for most companies, the top of this quadrant is a utopia. Keeping an entire organization here is financially unsustainable and commercially dangerous.

Q2: High Performance. Profile: High Quality / Low TTM. **Where it lives:** surgical teams, rapid response units, interoperability missions. **Reality:** this is the state of maximum flow, with autonomy and independence to decide. However, it is an unstable equilibrium: in the long run, lack of challenges or constant pressure for speed with perfection can lead to demotivation or burnout.



Figure 2: Quadrants and operational zones of the Technical Leadership Matrix.

Q3: Innovation Zone. Profile: Low Quality (maturity under construction) / Low TTM. **Where it lives:** MVPs, proofs of concept, early product phases via Scrum. **Reality:** here, technical debt is a financial instrument: a debt intentionally taken to buy market time. The role of the technical leader is to ensure the team does not “fall in love” with the debt.

Q4: Crisis Zone. Profile: Low Quality / High TTM. **Where it lives:** unmanaged legacy projects, teams suffocated by bureaucratic processes, technical failure. **Reality:** this is the state of maximum entropy. We spend time, but maturity does not evolve. Business intention is lost and design collapses. AI, in this scenario, acts as a disaster accelerator.

6.7 Recovery diagnosis

Attempting to exit the crisis via Q1 (more documentation, meetings, and slow processes) is a fatal mistake. The way out of crisis is diagonal: it requires the management shock and the autonomy of Q2 (High Performance) to stabilize the system before any other evolution.

$$F_{\text{Stabilization}} = \Delta V_{\text{focus}} + \Delta Q_{\text{critical}} \quad (3)$$

This is done through the surgical action of Q2, seeking greater Intention precision. Instead of first pursuing the quality jump, we pursue the efficiency jump with deep pragmatism and autonomy, eliminating process noise that prevents sovereign technical execution. At this stage, intervention by high-performance technical leaders is necessary to pull the system out of Crisis Zone entropy.

7 The role of technical leadership

Technical maturity is not a passive byproduct of time: it is the accumulated result of sustained quality over time [3]. If left to inertia, the organization tends to trade robustness for speed, and the system “slides” into states where quality does not grow (or even regresses). Therefore, technical leadership acts as a necessary force: it creates the conditions (methods, standards, guardrails, platform, and risk governance) to preserve the minimum quality floor and, when possible, raise it.

Formally, we can model the *Maturation Vector* as the accumulation of technical quality over the interval $[t_0, t_1]$:

$$R_{\text{Maturation}}(t_0, t_1) = \int_{t_0}^{t_1} Q(t) dt \quad (4)$$

Here, $Q(t)$ represents the effective level of system quality/maturity (or, alternatively, its marginal maturity contribution) at each instant. The role of technical leadership is to keep $Q(t)$ above the viability floor and guide investments that increase its accumulated area; without this force, Execution Abundance (especially with AI) may increase output volume without increasing sustained value.

8 Viability Engineering

In the era of augmented intelligence, code writing is no longer the epicenter of technical competence. Code has become an abundant *commodity* and, without direction, an entropy accelerator. **Viability Engineering**, therefore, emerges as the discipline of ensuring software integrity and longevity through the curation of Intention and the orchestration of Design.

Human sovereignty shifts from execution to strategic decision-making: it is not about how much code we can generate, but about how little code we need to materialize a powerful intention. Without an engineering discipline that directs these efforts, organizations succumb to the **Virtual Productivity paradox**: excess motion that masks the absence of real progress.

8.1 The Four Pillars of Viability Engineering

1. The Clarity Pillar (Mapping)

- **Function:** Dissipate the technical “fog of war”.
- **Delivery:** A map of systemic reality. Without an accurate diagnosis of debts, risks, and AI limits, any direction becomes an act of sabotage against viability.
- **Effect:** Psychological and technical safety.

2. The Intentionality Pillar (Alignment)

- **Function:** Ensure every generated byte has a commercial “why”.
- **Delivery:** The fusion of profit and code. This is where technical leadership and product converge to eliminate systemic noise.
- **Effect:** Sense of purpose and effectiveness.

3. The Integrity Pillar (Protection)

- **Function:** Shield the system against collapse and obsolescence.
- **Delivery:** Architectural *guardrails* and design sustainability. It is the membrane that protects the software’s future from the present’s reckless speed.
- **Effect:** Operational confidence.

4. The Adaptability Pillar (Evolution)

- **Function:** Prevent the system and the team from becoming static monuments.
- **Delivery:** Continuous evolution and antifragility. It ensures that, as maturity increases, the Convergence Point shifts toward higher levels of efficiency.
- **Effect:** Longevity and competitiveness.

9 Transition proposal

The transition to **Viability Engineering** is not an administrative process; it is a change in governance: from execution as the center to *decision* as the center. In practice, the **4 Pillars** operate as the mechanism that governs the variables of Software Effectiveness ($E_s = I \times (D + O)$), preserving value when execution (via AI) becomes abundant.

1. **Clarity:** makes the system’s reality explicit (risks, debt, constraints, and AI limits), reducing noise and creating the conditions for consistent decisions.
2. **Intentionality:** acts directly on Intention (I), ensuring the multiplier in the equation is not zero due to lack of purpose, priority, or business coherence.
3. **Integrity:** sustains Design (D) through *guardrails*, architecture, and risk governance, preventing apparent speed from turning into operational entropy.
4. **Adaptability:** increases Orchestration efficiency (O) and the capacity for continuous change, so the system can evolve without collapsing under its own accumulation.

The goal of the transition is simple: ensure Execution Abundance does not amplify waste, but is filtered by Intention and sustained by Design — converting volume into viability.

10 Conclusion

The post-AI era requires a new leadership archetype. If code has become an inexhaustible commodity, technical authority can no longer emanate from the ability to “do”, but from the wisdom to “make viable”.

The Technical Leadership Matrix provides the map for this transition, taking software development out of the territory of luck and placing it under the domain of Intention and Design.

By mastering this framework, technical leadership regains its intellectual sovereignty. It stops being the execution bottleneck and becomes the guardian of systemic integrity.

Ultimately, the value of software is not in how fast it is written, but in how precisely it materializes a powerful intention and how capable it is of surviving its own creation.

Viability, therefore, is not the end of the journey, but the fundamental axiom that allows the business to exist, compete, and evolve in a world of growing complexity.

References

- [1] Frederick P. Brooks. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, anniversary edition, 1995. Original: 1975.
- [2] Tom DeMarco and Timothy Lister. *Peopleware: Productive Projects and Teams*. Dorset House, 1987.
- [3] M. M. Lehman. On understanding laws, evolution, and conservation in the large-program life cycle. *Journal of Systems and Software*, 1(3):213–221, 1980.