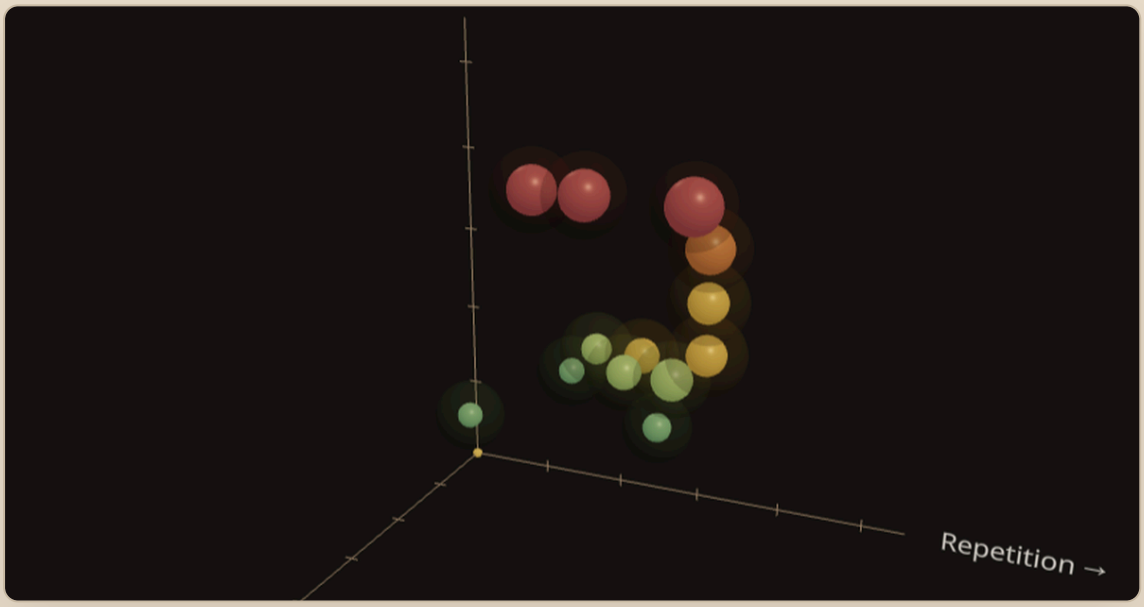
A FRAMEWORK FOR THE AI-TOOL ERA

# The 4D Framework

*An Essay on Choosing the Right Tools in the AI Era*




*THE INTERACTIVE DECISION CUBE · ROIOFAUTOMATION.COM*

Soeren de Vries

APRIL 2026

ABSTRACT

*We live in a moment where hundreds of AI tools promise to transform how we work — and where many of them quietly make things worse. This paper argues that tool selection has stopped being a technology question and has become an economic one. It proposes a funnel of three pre-checks, a four-dimensional decision cube, and a break-even formula to help practitioners choose tools that actually earn their keep. The framework draws on recent empirical work on the "jagged technological frontier" and on the broader debate between AI automation and AI augmentation. It is written for team leads, managers, and hands-on contributors who have to make these calls every week.*

# Executive Summary

Choosing the right tool used to be simple. You had a spreadsheet, a database, maybe an automation platform, and you picked based on what fit the task. That world is gone. The moment AI entered the stack, tool selection turned into something stranger and riskier: a field with thousands of options, shifting every week, where the wrong choice does not merely waste time — it can make a working process measurably worse than before.

This whitepaper proposes a structured way to make that choice. The framework has three layers, in deliberate order:

1. **A funnel of three pre-checks** that filters out projects that should never be built in the first place — *before* anyone debates a tool.
2. **A four-dimensional decision cube** that maps the surviving tasks to the right level of tooling.
3. **A break-even formula** that grounds the final call in economics, not enthusiasm.

A two-mode extension — *Reality* for daily pressure, *Strategic* for deliberate planning — acknowledges that most real decisions happen without the luxury of a whiteboard session. A living tool list with a *hypothesis* layer keeps the framework honest about the fact that the landscape changes faster than any static recommendation can keep up with.

The argument throughout is simple: tools are not neutral. Picking the wrong one is not just inefficient; it is actively harmful in a way that classical tool-selection rarely was. That shift deserves a more rigorous method than gut feeling.

---

# 1. The Problem: Tool Overload Is an Economic Problem

Every week, a new AI tool claims to change everything. Every month, a dozen more quietly disappear. Somewhere in between, teams are left making decisions about which of these tools to adopt, which to ignore, and which to rebuild themselves. Most of them are making those decisions badly — not because they lack skill, but because the question has become genuinely hard.

The traditional framing — *"pick the best tool for the job"* — no longer works, for three reasons.

**First**, the tool set is no longer stable. A workflow you automate in Make this quarter might be built into your CRM next quarter, at which point your automation becomes technical debt you did not plan for. A custom Lovable app that fills a gap today may be obsolete in six weeks, not because it failed, but because the underlying platform caught up. The concept of a "best tool" is losing coherence at the pace at which tools are being released.

**Second**, AI-assisted tools do not fail gracefully. They fail *jagged*. Recent work from Harvard, Wharton, MIT, and BCG — *Navigating the Jagged Technological Frontier* (Dell'Acqua et al., 2026) — documents this empirically. AI improves performance for some tasks and worsens it for others, and these tasks can look almost identical on the surface. Adopting an AI tool without understanding where it sits on that frontier is not a neutral experiment. It is a bet, and the downside can be genuine regression in output quality.

**Third**, automation has a dark twin.

> ***The Silent Regression***
>
> *A process that looks modernised but performs worse than the version it replaced. A human who fills out ten customer forms a week may make one error a month. An AI that fills out the same forms may make one error a week — but produce them faster, with more authority, and with less of the quiet hesitation that flags an unusual case. If nobody measures, the decline is invisible until a customer notices.*
>
> *The silent regression is the failure mode this framework is built to prevent.*

Taken together, these three shifts turn tool selection from a practical craft into an economic judgment. The central question is no longer *"which tool is best?"* but *"given my constraints — frequency, complexity, risk, and the skill of the person who will actually build it — does building with this tool earn back what it costs me, including the risk it introduces?"*

That is the question this framework is built to answer.

---

# 2. Two Narratives: Automation vs. Augmentation

Before the mechanics, a note on framing. In April 2026, Harvard Business Review published a piece by De Neve, Hancock, and Niederhoffer titled *Why Companies That Choose AI Augmentation Over Automation May Win in the Long Run*. The piece captured a divide that had been visible in practice for months.

On one side stood Jack Dorsey, who used AI as a reason to cut nearly half of Block's workforce. *"Intelligence tools have changed what it means to build and run a company,"* he wrote. His bet: AI lets you do what you did, with fewer people.

On the other side stood Micha Kaufman of Fiverr, who told employees bluntly that AI was coming for their jobs, including his own, and that the answer was not to shrink the company but to grow what the re-

maining people could do. His bet: AI frees humans to exercise the kinds of judgment — taste, nuance, strategy — that machines still handle poorly.

The evidence, at least so far, favours the second approach. The 2025 Indeed Workforce Insights Report showed that time saved through AI adoption rarely translates into new, higher-value work. It is more often absorbed by additional volume of the same tasks. Teams become busier without becoming more valuable.

A broader dataset points in the same direction. McKinsey Global Institute's 2025 study *Agents, Robots, and Us* mapped roughly 6,800 skills against their technical automation potential. The central finding: only a small share of work hours — around 11% — is realistically "people-led, AI-untouched". Another small share is "AI-led, people-untouched". The overwhelming majority, close to 72%, sits in the middle, where people and AI genuinely have to work together for the task to get done well. Tool selection, viewed through that lens, is rarely a question of whether humans or machines should handle a task. It is almost always a question of *how* the two are paired.

This framework takes a deliberate side in that debate. It is designed to help teams choose tools that *augment* human capability — not tools that replace humans in places where the human was already doing the job well. The distinction is not ideological; it is economic. Replacing a low-error human with a higher-error AI is a cost increase disguised as a cost reduction. The framework is built to expose that disguise.

---

# 3. The Funnel: Three Gates Before the Cube

The most common mistake in tool selection is starting too deep. Teams begin by debating *how* to build something — Make or Claude Code? Lovable or pure Next.js? — before anyone has asked whether it should be built at all. The funnel corrects this by placing three gates in front of every tool decision. Each gate is a simple yes/no question; each one filters out projects that should never progress further.

Many ideas go in. Only the ones that survive all three checks deserve the effort of actual tool selection. Most of the wrong answers in this whole field are not bad tool choices. They are good tool choices for builds that should never have been started.

### *Gate 1 — Exists?*

Does the solution already exist, off-the-shelf or as a built-in AI feature in a core platform?

This is the cheapest check and the one most often skipped. Before anyone commits time to building a meeting-notes summariser, the question is whether Gmail, Notion, Zoom, or Fathom already handles it. Before building a CRM-data extractor, the question is whether HubSpot or Salesforce ship that feature

today. When the answer is yes, the decision is almost always to use what already exists, even if it is slightly imperfect. The marginal gain from building your own version rarely justifies the marginal cost.

### *Gate 2 — Coming?*

Is your primary platform likely to ship this feature within six months?

Platforms now move at a pace that makes six-month-old custom work obsolete on a regular basis. A Lovable app built during a hackathon to summarise a HubSpot account can become redundant three weeks later when HubSpot releases the same feature natively. The cost of that hackathon is not just the time spent; it is also the maintenance debt that now outlives the useful purpose.

Signals to check: public product roadmaps, beta features, early access programmes, talk tracks at industry conferences, and what the vendor is demonstrating at their own hackathons. None of these are perfect predictors, but together they tell you whether you are about to build something that will be given away for free in a quarter.

### *Gate 3 — Worth it?*

Will the expected value of using the tool exceed the cost of building and maintaining it, including the opportunity cost of what your builder is not doing instead?

This is where the break-even formula earns its keep (see Section 6). Gate 3 has two sides. The first is financial: build-cost in euros against cumulative use-value over the tool's lifetime. The second is opportunity-related: if your automation manager spends four hours on this, they do not spend those four hours on something else. Sometimes the alternative is genuinely more valuable. A framework that ignores opportunity cost ends up recommending builds that maximise local ROI at the expense of a team's broader agenda.

When Gate 3 returns a negative answer, the recommendation is not a cheaper tool — it is usually no tool at all. Some tasks are genuinely best handled by hand.

---

## 4. The Four Dimensions: The Decision Cube

Only when a task has passed all three gates does the four-dimensional cube become relevant. At that point, you know that something *should* be built; what remains is choosing *what kind* of something.

Each of the four dimensions is scored on a 1–5 scale in the user interface, and normalised internally to 1–10 for the break-even calculation.

### *Repetition (Frequency and Scalability)*

How often will this run? Once is a 1. Every hour is a 5. The higher the score, the more attractive automation becomes — and the more that even small per-use savings add up to a real number over time.

### *Complexity (Data Structure and Variability)*

How structured and stable is the work? A straight copy-paste is a 1. A multi-system, edge-case-heavy transformation is a 5. Complexity pushes you up the tool ladder because low-code platforms run out of headroom quickly when the logic branches.

A useful parallel comes from the Fraunhofer IAO's work on the limits of AI automation, which identifies seven protective factors that keep tasks out of reach of current AI: heavy cognitive load, interpersonal interaction, genuine creativity, holistic situational judgement, intuitive high-stakes calls, ethical evaluation, and meaningful discretion. A task that carries any of these tends to score high on this dimension, and should usually be scored high on Risk as well.

### *Risk (Error Tolerance and Compliance)*

What happens when something goes wrong? An internal dashboard is a 1 or 2. A customer-facing contract generator is a 4 or 5. Risk deserves particular attention when AI is involved, because the jagged frontier means the failure modes are not always the ones you expected during testing. Higher risk usually means either more robust tooling, human review loops, or both.

### *Builder Skill (Implementation Capability and Speed)*

Who is actually going to build this, and what do they know? A business user with Excel is a 1 or 2. A full-stack developer is a 5. This dimension is often the one teams get wrong — not because they score themselves inaccurately, but because they build around the skill they *wish* they had. The honest score is what determines whether a recommendation is realistic.

The earlier drafts of this framework called this dimension *Human Talent*. The label was meant kindly but read like a personal verdict. *Builder Skill* names the same thing more sharply: it is about implementation capability, not human worth. It includes the skill itself, the time available to apply it, the experience of having shipped similar things before, and the appetite to maintain what gets built.

*A note on depth:* Builder Skill could be decomposed into a richer compound score — skills, learning capacity, available learning time, team constellation. The framework deliberately keeps it at the same granularity as the other three dimensions. Breaking that symmetry would let this one dimension dominate the recommendation in ways that feel analytical but are actually arbitrary. Teams that want a deeper skill model can add it as an overlay without disturbing the core.

The fourth dimension is the one people argue about most. It seems as if high skill should mean picking the best possible tool, and low skill should mean accepting a cheaper one. In practice, the opposite is often true: high-skill teams sometimes over-engineer, while low-skill teams find pragmatic solutions that scale surprisingly well.

**The best tool is the one the available builder actually knows how to use. Not the one a theoretical expert would recommend.**

---

# 5. The Tool Ladder

The four dimensions map to a ladder of seven levels. Each level has its own sweet spot, its own build cost, and its own failure mode. The goal is not to climb as high as possible — it is to stop at the lowest level that still fits the task.

The earlier version of this ladder mixed categories that did not belong on the same shelf. Cursor, Lovable, and Next.js are not the same kind of thing. The version below separates them more honestly: it groups tools by the *abstraction level a user works at*, not by which company shipped them.

| Level | Category | What it really is | Examples |
|---|---|---|---|
| 0 | Manual | A person and a clear head | A spreadsheet, a printed checklist, judgement |
| 1 | Existing SaaS or native AI feature | Something a vendor already built | Gmail, HubSpot UI, ChatGPT, Claude, Perplexity |
| 2 | Embedded AI inside an existing system | AI inside a tool you already pay for | Notion AI, HubSpot Breeze, Microsoft 365 Copilot |
| 3 | Spreadsheet plus scripts | Code, but only enough to glue cells together | Apps Script, Sheets formulas, light VBA |
| 4 | Automation platform / iPaaS | Visual workflows between systems | Make, Zapier, n8n, Power Automate |
| 5 | Developer-assisted coding | Real code, written with an AI pair | Claude Code, Cursor, GitHub Copilot, Windsurf |
| 6 | Internal app or custom interface | A product, with users and an UI | Lovable, v0, Bolt, Retool, full Next.js builds |

The heuristic is blunt: *go as deep as needed, no deeper*. Each rung up the ladder adds build cost, adds maintenance, and usually adds a little more surface for the jagged frontier to bite. The wins from climbing need to justify all three.

### *A Hidden Modifier: Friction Per Use*

The ladder above describes *what gets built*. It says less about *what it costs the user every time they touch it*. That second cost — the friction per use — quietly determines whether a tool actually accumulates value over time, or whether it sits unused after the first month.

Some of the most valuable tools in modern work live at Level 1 but feel like they belong somewhere else. *Wispr Flow* turns three seconds of speech into a paragraph of typed text; the per-use cost is nearly zero. *Granola* drops meeting notes into the workflow without being summoned. *Raycast* replaces a dozen mouse clicks with a keystroke. These tools are not impressive on the cube. They are impressive in compound: you barely notice them, and yet the time they save accumulates into hours per week.

Friction per use does not need its own cube axis — that would crowd the visual without adding clarity. It belongs in the break-even calculation as a small but persistent drag on Use-Value. The practical rule: when a tool requires more than two or three deliberate actions per use, model the use-cost; when it disappears into the workflow, treat use-cost as effectively zero. The tools that disappear are usually the ones worth standardising on.

---

## 6. The Break-Even Formula

The cube points you to a level. The formula tells you whether that level is worth the build.

```
Break-Even-Uses = Build-Cost / Net-Use-Value-per-Use

where:
    Build-Cost            = Build-Time (h) × Builder-Rate (€/h)
    Use-Value-per-Use     = Time-Saved-per-Use (h) × User-Rate (€/h)
    Risk-Cost-per-Use     = Probability-of-Error × Cost-per-Error
    Net-Use-Value-per-Use = Use-Value-per-Use - Risk-Cost-per-Use
```

A richer form adds maintenance and a safety margin:

```
ROI(n) = n × Net-Use-Value-per-Use - Build-Cost - Maintenance-Cost

Recommendation: build only if
    expected uses over tool lifetime > Break-Even-Uses × 1.5
```

The 1.5 multiplier is a deliberate hedge. Forecasts of future usage are almost always optimistic, and a 50 % safety margin keeps you from building things that look good on paper and fail to earn back their cost in practice.

A small but important note: in earlier versions of this paper, Risk-Cost was deducted twice — once inside Use-Value and once again in the ROI formula. The corrected definition above keeps it clean. Use-Value is the gross gain per use; Risk-Cost is its expected drag; Net-Use-Value is what is left, and that is what the build needs to amortise.

### *Worked Example 1 — The Build That Pays Off*

Customer forms filled by hand: fifteen minutes per form, twenty forms a month, employee rate 30 €/h. An alternative Make automation costs four hours to build at an automation-manager rate of 80 €/h.

```
Build-Cost            = 4 × 80          = 320 €
Use-Value-per-Use     = 0.25 × 30       = 7.50 €
Risk-Cost-per-Use     = 0.50 €  (occasional rework)
Net-Use-Value-per-Use = 7.00 €
Break-Even-Uses       = 320 / 7         = ~46 uses
```

At twenty uses per month, the build pays itself off in about two and a half months. A two-year lifetime yields a return of several thousand euros in reclaimed time. The decision is clear: build.

### *Worked Example 2 — The Build That Dies Three Weeks Later*

A hackathon team builds a Lovable app that summarises HubSpot account data for the sales team. Ten salespeople use it five times a week.

```
Build-Cost            = 6 × 80          = 480 €
Use-Value-per-Use     = 0.1 × 40        = 4.00 €
Risk-Cost-per-Use     = 0.20 €
Net-Use-Value-per-Use = 3.80 €
Break-Even-Uses       = 480 / 3.80      = ~127 uses → about a week
```

Mathematically, the build is a winner. In practice, it was the wrong call. HubSpot released the same feature natively three weeks later. The error was not in the cube, and it was not in the formula; it was in skip-

ping Gate 2. No amount of ROI optimism survives a free alternative shipping shortly after.

The lesson generalises. Break-even economics are necessary, but they are not sufficient. The gates matter more, because they can invalidate the entire calculation.

### *Worked Example 3 — A Sales-Operations Build Worth Doing*

A revenue-operations team wants weekly account-health reports for ninety key accounts. The data lives in HubSpot, Stripe, and a Google Sheet. Today the analyst pulls it manually each Monday — about ninety minutes per week.

```
Manual cost           = 1.5 h × 50 €/h × 52 weeks      ≈ 3 900 €/year
Build option          = Make scenario, 6 h × 80 €/h    = 480 €
Net-Use-Value-per-Use = 1.5 × 50 - ~5 € risk           = ~70 €
Break-Even-Uses       = 480 / 70                       ≈ 7 weeks
```

Less than two months to break-even, on a workflow that runs every week, against data sources with stable APIs. Risk is moderate (the report nudges sales calls, it does not sign contracts). Builder Skill needed: a citizen developer comfortable with Make. The recommendation is unambiguous: build, at Level 4.

### *Worked Example 4 — A Build That Should Not Happen*

An HR generalist asks for an AI tool that drafts personalised candidate-rejection emails by combining the job description, the candidate's CV, and the interviewer's notes. It feels like exactly the kind of work AI was made for: high volume, repetitive, low-creativity. The team is ready to spec it.

But the gates intervene. Gate 1: is this already a feature in their ATS? It will be — the vendor has it on the public roadmap for the next quarter (Gate 2). Gate 3 is where the picture changes entirely: a candidate-facing letter is a customer-trust artefact. The risk is not the time saved; the risk is one tone-deaf phrase that ends up screenshotted and shared. The expected risk-cost per use, even at a low probability of failure, is high enough that Net-Use-Value is borderline negative.

```
Use-Value-per-Use     = 0.15 h × 35 €/h         = 5.25 €
Risk-Cost-per-Use     = 0.5 % × 8 000 €         = 40 €  (one bad letter)
Net-Use-Value-per-Use = 5.25 - 40                ≈ -34 €
```

The math is decisive. The framework recommends manual handling, with a templated draft as a stop-gap, until the ATS feature ships and trust has been validated. The build that *felt* obviously right turned out to be the build that should not happen at all.

### *A Note on Estimation Under Uncertainty*

A fair objection to any break-even formula is that its inputs are hard to measure. Nobody knows exactly what their risk-cost-per-use will be. Nobody has perfect data on how much time an automation will actually save. That is not a reason to abandon the calculation; it is a reason to do it deliberately. When hard numbers are unavailable, a *fermi estimation* — rough bracketing with explicit assumptions — is usually more honest than pretending the question does not apply. Sensible defaults: set risk-cost at roughly 10 % of use-value when no better data exists; model use frequency in three scenarios (optimistic, realistic, pessimistic); treat any break-even that requires heroic assumptions about volume as a warning sign, not a green light.

---

## 7. Two Modes of Operation: Reality vs. Strategic

The framework has two ways of being used. Both are legitimate; the choice depends on the situation.

**Reality Mode** is the default, and it assumes what most working days assume: there is pressure, there is a deadline, and the decision has to be made now. In Reality Mode, the first question is not *what is the optimal tool?* but *what can I actually ship by Friday?* The gates are run quickly, the cube is simplified, and the recommendation favours solutions the current builder already knows. Stop-gaps are explicitly labelled as stop-gaps, with a backlog entry for the eventual proper fix.

**Strategic Mode** is for the decisions where you actually have time. A new workflow for the next quarter. A tooling overhaul for a team. A build-or-buy question at a CIO level. In Strategic Mode, the full framework runs — all three gates, all four dimensions, the complete break-even analysis. Urgency becomes a final override rather than an initial filter. The output is closer to optimum, at the cost of more deliberation.

Most frameworks pretend that every decision deserves the strategic treatment. They do not. In practice, the majority of tool choices happen under pressure, and a framework that cannot handle that pressure gets ignored the moment it would matter most.

---

## 8. The Living Tool List

The principles in Sections 3 through 7 are designed to be stable. The tool names in Section 5 are not. They will age faster than any published framework can keep up with. The solution is to separate the two.

In the accompanying application, the tool ladder lives in a database with three status flags:

- `live` — the tool is in production use, tested, and part of active recommendations.
- `hypothesis` — the tool has been announced or is in closed beta, and the recommendation engine treats it as a conditional: *"If tool X launches in Q3 as announced, it would cover use case Y."*
- `deprecated` — the tool still exists but is no longer recommended, either because a better option has emerged or because the vendor has signalled end-of-life.

Updates run weekly rather than quarterly. At the pace the AI tool market currently moves, quarterly is already a form of neglect. A research inbox feeds candidate tools into the system, where an automated check cross-references them against the live list and flags duplicates, near-duplicates, and genuine novelties for review.

The *hypothesis* layer is what turns this from another static tool recommendation into something more useful. It lets users see not just what to build today, but what to hold off building because the market is about to do it for them. For teams serious about avoiding work that will be obsolete in six weeks, that foresight is worth more than any optimisation of the present moment.

---

# 9. When the Framework Disagrees With You

A framework is only useful if it occasionally tells you something you did not already think. The 4D framework will disagree with common practice in three predictable ways.

**It will often recommend doing less.** When Gate 1 surfaces an existing solution, the answer is to use it, even if your team could build something slightly nicer. When Gate 3 surfaces negative ROI, the answer is to leave the work manual, even when building feels more satisfying. Builders tend to want to build. The framework's job is to ask whether they should.

**It will recommend simpler tools than people expect.** Most teams over-estimate the level at which they should be operating. A task that feels complex is often a 3 or 4 on the tool ladder, not a 5 or 6. Over-tooling is expensive in ways that do not show up until maintenance becomes a burden.

**It will sometimes recommend waiting.** This is the hardest recommendation to accept, especially for teams that have been told their job is to move fast. But if Gate 2 identifies a feature likely to ship from your primary platform within six months, the correct move is often to build a small stop-gap at Level 0 or 1 and let the platform do the rest. The alternative — building at Level 4 or 6 and being obsoleted — is a very expensive lesson.

---

# 10. Limits and Open Questions

No framework is complete. Four limits deserve acknowledgement.

**Scoring is subjective.** Complexity and Risk in particular depend on who is doing the scoring. The framework reduces this by forcing explicit scoring rather than implicit hand-waving, but two honest practitioners may still rate the same task differently. The application addresses this partly through worked examples and partly by letting teams calibrate against their own prior decisions.

**The tool list ages.** This is a structural problem in any recommendation system, not a flaw unique to this framework. The hypothesis layer mitigates it; weekly updates mitigate it further; but nobody should use a tool list that has not been touched for six months and expect it to still be right.

**The break-even formula flattens qualitative factors.** It does not natively model things like team learning, morale, or strategic positioning. A build that loses money on paper can still be worth doing if it teaches a team something genuinely new, or if it opens a capability the organisation needs for a different reason. Practitioners should treat the formula as a baseline, not a verdict.

**The framework addresses rational tool selection, not organisational politics.** Real decisions often get shaped by budget ownership, sponsor preferences, vendor loyalties, or pressure from above. None of that is in the cube, and none of it belongs there. In environments where politics dominates, the framework is most useful as a way of *naming* the gap between the rational choice and the actual one. That gap is sometimes worth closing, sometimes worth accepting, but either way it is worth making explicit. A rational model loses its value the moment it pretends to be the whole picture.

## *A minimum on Governance, Adoption, and Lifecycle*

The cube and the formula are the centre of the framework, not the whole of it. Three further checks belong on every build that passes Gate 3, and they tend to be skipped under pressure. A short version of each is worth carrying around.

*Governance.* Who decides? Who owns the data the tool will touch? Who reviews the privacy and compliance implications before anything goes live? In Germany and the wider EU, that list usually includes the works council and a data-protection officer; in regulated industries it includes more. The framework does not produce these answers, but it should not let a build proceed without them.

*Adoption.* A positive break-even is worthless if the people the tool was built for never use it. The pragmatic test: can the tool be reached inside an existing workflow, or does it require a deliberate detour? Tools that disappear into a workflow get used. Tools that need to be remembered get forgotten.

*Lifecycle.* Every build should ship with a review date. Six months is a reasonable default. Without that date, the tool quietly drifts from solution to debt — exactly the outcome the rest of the framework is de-

signed to prevent.

These three checks are not new ideas; they are well-known across operations and IT. They appear here because the framework, in concentrating on tool depth and economics, can otherwise read as if those depth and economics were the whole story. They are not.

---

# 11. Conclusion

The central claim of this framework is that tool selection in the age of AI is no longer a tooling question. It is a question about economics, risk, and the honest assessment of what your team can actually deliver. The funnel exists because most of the wrong answers come from building things that never should have been built. The cube exists because once a build is justified, the level of tooling still matters. The formula exists because enthusiasm alone is not an economic argument.

Used together, the three layers push practitioners towards tools that *augment* rather than merely *automate* — tools that expand what humans can do without displacing the places where humans were already doing the job well. Deloitte's *2026 Global Human Capital Trends* frame this shift in a phrase that is worth stealing: the move from *human + machine* to *human × machine*. Adding a tool to a team is addition; orchestrating the tool so that people and machines genuinely multiply each other's output is a different kind of arithmetic. Most tool decisions today are still made in the additive register. The decisions that compound — the ones that become a company's quiet, durable advantage — are the multiplicative ones.

The framework will not make every decision obvious. It will, reliably, make the obviously bad decisions harder to justify. In a market this noisy, that is most of the value.

---

# References

De Neve, J.-E., Hancock, J. T., & Niederhoffer, K. (2026). *Why Companies That Choose AI Augmentation Over Automation May Win in the Long Run.* Harvard Business Review, April 2026.

Dell'Acqua, F., McFowland, E., Mollick, E., Lifshitz, H., Kellogg, K. C., Rajendran, S., Krayer, L., Candelon, F., & Lakhani, K. R. (2026). *Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of Artificial Intelligence on Knowledge Worker Productivity and Quality.* Organization Science. https://doi.org/10.1287/orsc.2025.21838

Miedema, E., Waschull, S., & Emmanouilidis, C. (2025). *Towards trustworthy artificial intelligence for decision-making: A lifecycle perspective on knowledge- and data-driven artificial intelligence systems.*

Yee, L., Madgavkar, A., Smit, S., Krivkovich, A., Chui, M., Ramirez, M. J., & Castresana, D. (2025). *Agents, Robots, and Us: Skill Partnerships in the Age of AI.* McKinsey Global Institute, November 2025.

Deloitte Insights (2026). *2026 Global Human Capital Trends: From tensions to tipping points – Choosing the human advantage.*

Fraunhofer IAO (2024). *Schnelltest: Hat dein Job genug Routine für Künstliche Intelligenz?* IAO-Blog. https://blog.iao.fraunhofer.de/schnelltest-hat-dein-job-genug-routine-fuer-kuenstliche-intelligenz/

Indeed Workforce Insights Report (2025). Referenced in De Neve et al., 2026.

---

# About the Author

Soeren de Vries is a Senior Automation Manager focused on AI enablement and process automation. He works at the intersection of revenue operations, service-industry workflows, and AI-driven tooling — the places where theoretical frameworks either earn their keep or fall apart in daily practice. He writes about the economics of automation, the limits of AI in real operations, and how teams can stay pragmatic in a landscape that changes every week.

He lives with his family in Baden-Baden.

Connect on LinkedIn: linkedin.com/in/sören-de-vries-32204157

---