

BEYOND THE PLATFORM POSTURE

Build, Buy, or Orchestrate: The Wrong Question About Enterprise AI

How the platform-posture debate lets organisations defer every decision that actually matters

Giovanni Leonardi

July 2024



Build, Buy, or Orchestrate: The Wrong Question About Enterprise AI

Transformation · No. 385

Originally written July 2024 | Re-edited and re-rendered for the WhereBeyond Edition, 2025

"It is a data-and-evaluation difficulty wearing a platform costume."

1. The meeting where the real question never gets asked

By the middle of this year the scene has hardened into a ritual. A steering committee convenes to settle "the AI platform strategy." Three options sit on the slide: build, buy, or orchestrate. Someone from architecture argues for building — control, differentiation, no dependence on a vendor's roadmap. Someone from the commercial side argues for buying — speed, support, a licence and a helpline instead of a hiring problem. And then, almost inevitably, the room settles on the third option, because orchestrate sounds like the grown-up synthesis of the other two: take the best models from whoever happens to have them this quarter, wrap them in a layer of your own, keep your options open while the field is still moving. The committee feels sophisticated. A decision has been recorded. Everyone goes to lunch.

I have sat in enough of these rooms to say plainly what I have come to believe: the meeting that chooses between build, buy, and orchestrate is usually deciding almost nothing. It is choosing a posture and mistaking it for a strategy. The three words describe how an organisation intends to feel about its suppliers. They say nothing about where its advantage will actually come from, what it will genuinely put into production rather than demonstrate, or which of the hard problems it has just quietly agreed to own. Those are the only questions that matter, and the build-buy-orchestrate frame is very good at ensuring none of them gets asked.

2. The difficulty does not disappear; it moves

Strip away the vocabulary and each of the three options is a decision about where to place the same fixed quantum of difficulty. Choosing a posture does not reduce the difficulty. It relocates it — usually somewhere less visible, which is precisely why the choice feels like progress.

- Build relocates the difficulty into talent and time. The organisation that fine-tunes its own models or writes its own applications against raw model APIs is not wrong to want control, but it has taken on an engineering and evaluation burden that does not show up on the decision slide. The models it builds against will be superseded within months; the scaffolding it writes will need rewriting.

- Buy relocates the difficulty into fit and dependence. Licensing a copilot or an AI feature bolted onto software you already run is genuinely the fastest way to put something in front of users. But you have accepted someone else's definition of the problem, someone else's data boundaries, and someone else's release schedule — and the moment your requirement diverges from their roadmap, you are stuck waiting.
- Orchestrate relocates the difficulty into the seams. This is the one the room least wants to examine, because it is dressed as the answer that avoids the other two traps. It does not. It simply moves the hard part into the integration layer, the retrieval pipeline, the evaluation harness, and the governance of a system whose behaviour now depends on components you do not control and cannot freeze.

Posture	What it promises	Where the difficulty actually lands
Build	Control and differentiation	Scarce talent; models and scaffolding that go stale in months
Buy	Speed and support	Vendor's definition of the problem; you wait on their roadmap
Orchestrate	Flexibility; best-of-breed	The seams — data, retrieval, evaluation, and governance you still own

None of this makes any of the three wrong. It makes the choice between them, taken at the level of an enterprise-wide platform posture, close to meaningless — because the difficulty each one relocates is the same difficulty, and it is not a platform difficulty at all. It is a data-and-evaluation difficulty wearing a platform costume.

3. Why "orchestrate" became the answer everyone reaches for

It is worth being fair to the case for orchestration, because it is a strong one and I do not want to knock down a straw version of it. The honest argument runs like this: the model layer this year is moving faster than any procurement cycle can track. A model that leads on a benchmark in the spring is matched by three others by the summer, and prices per token keep falling. In that environment, hard-wiring an organisation to a single provider — whether by building deeply against one model's quirks or by buying one vendor's suite — is a genuine risk. An orchestration layer that treats models as interchangeable, swappable behind a stable internal interface, is a rational hedge against a market that has not settled. That reasoning is correct as far as it goes, and anyone who dismisses it has not been paying attention to how quickly the ground is shifting.

But the argument proves less than it appears to. Keeping the model layer swappable is worth doing; it is also the easy part, and it is not where the risk concentrates. Swapping one capable general model for another is, increasingly, a configuration change. What does not swap out — what an organisation is left holding no matter which model sits behind the interface — is the retrieval pipeline that feeds the model its own proprietary context, the evaluation regime that tells it whether the answers are good enough to trust, and the accountability for what the system does when it is wrong. Orchestration keeps the cheap thing flexible and quietly leaves the expensive things exactly where they were. It is a hedge against the risk that has already largely commoditised itself, and it does nothing

about the risks that have not.

The model layer is the part of the stack most likely to become a commodity, and therefore the part least worth building a strategy around. The strategy lives in the data you can bring to a model and your ability to tell, rigorously, whether its output is good enough to act on.

4. A worked example, invented but true to life

Consider a composite that will be familiar to anyone who has run one of these programmes this year. A large services organisation wants to give its front-line operations staff an assistant that answers questions from a sprawling body of internal policy, product, and procedural documentation — the sort of knowledge that currently lives in the heads of a few long-serving people and in several thousand pages nobody reads until something goes wrong.

The steering committee runs the ritual. Build is rejected as too slow and too talent-hungry. Buy is tempting — a general enterprise copilot is already licensed — but it cannot see the proprietary documentation in any depth, and its answers, while fluent, are confidently wrong often enough to be dangerous in a regulated setting. So the programme chooses to orchestrate: a capable third-party model, a retrieval layer over the document store, a thin application on top. On paper, the mature choice.

Then the real work begins, and none of it is the work the posture debate was about. The documents turn out to be inconsistent, undated, and in several places contradictory — three versions of the same policy, and no reliable signal as to which is current. Retrieval returns plausible passages that are subtly out of date. The team discovers it has no agreed way to measure whether an answer is correct, only whether it reads well. A first pilot answers roughly seven questions in ten to a standard the business will accept — impressive in a demonstration, unacceptable when the other three carry regulatory consequences. Getting from seven-in-ten to something closer to nineteen-in-twenty consumes the next several months, and almost every hour of it is spent on the corpus and the evaluation harness — curating documents, resolving contradictions, writing a test set of real questions with known good answers, measuring, and correcting. Not one of those hours is spent on the choice between build, buy, and orchestrate. That choice, the one the committee agonised over, turned out to be the least consequential decision in the entire programme.

The lesson is not that orchestration was the wrong posture. It is that the posture was never the point. The value and the risk both lived in the proprietary corpus and in the discipline of evaluation — and those would have been the deciding factors whichever of the three words the committee had written on its slide.

5. The decision that is actually worth making

If the platform posture is the wrong altitude, what is the right one? The useful decisions are not taken once, for the enterprise, in a steering committee. They are taken per capability, and they turn on a single question the build-buy-orchestrate frame never asks: where does our advantage actually come from here?

1. Start from defensibility, not from the stack. For each use case, ask what a competent competitor with the same off-the-shelf models could not easily replicate. Almost always the answer is your proprietary data and the specific workflow it sits inside — never the model itself. Anchor every subsequent choice to that.
2. Buy the undifferentiated. Where a capability is genuinely generic — drafting, summarising, coding assistance, meeting notes — buy it, take the vendor's roadmap, and spend none of your scarce talent there. Refusing to buy the commodity because you want "control" is how organisations run out of the people they needed for the parts that mattered.
3. Invest where your data is the moat. Where the advantage is your proprietary knowledge and process, put your effort into the corpus, the retrieval, and the evaluation — regardless of whether the surrounding assembly is called building or orchestrating. This is where the months go and where the return lives.
4. Make evaluation a first-class deliverable, not an afterthought. If you cannot measure whether an answer is good enough to act on, you have not built a system; you have built a demonstration. The test set is as much a part of the asset as the application.
5. Keep the model layer swappable, and stop congratulating yourself for it. Do it — it is cheap insurance — but recognise it for the routine hygiene it is, not the strategy.

None of these is a posture. Each is a judgement about a specific capability, and taken together they will usually produce a portfolio that builds a little, buys a great deal, and orchestrates in the places where proprietary data meets a fast-moving model layer. The organisation ends up doing all three — but because each decision was anchored to where its advantage actually lives, not because it declared an allegiance in a meeting.

6. The honest account

The uncomfortable truth beneath the build-buy-orchestrate debate is that most of the durable value available from this technology this year is unglamorous. It is in cleaning up the corpus nobody wanted to own, in writing the evaluation set nobody budgeted for, in the patient work of turning a fluent demonstration into a system that is right often enough to be trusted with consequences. The platform posture is a way of feeling strategic without doing any of that — a decision that can be taken in ninety minutes, before lunch, by people who will never touch the corpus.

“Choose the posture and you have chosen how to feel about your vendors. Choose where your data is the moat, and where evaluation must be ruthless, and you have chosen something that will still matter after the current crop of models has been forgotten.”

The next time the three options appear on a slide, the most valuable thing a leader can do is refuse the question as posed. Not build, buy, or orchestrate — but which capabilities carry our advantage, what will it take to make them trustworthy, and where are we merely renting the commodity? Answer those, and the posture takes care of itself. Answer only the posture, and you will spend the year discovering, expensively, that you deferred every decision that counted.