

The AI System Model: Why Your System—Not the Model—Decides Who Wins with AI

Nicholas Beaudoin, Caltech CTME



Boardrooms across the country are issuing the same mandate: become an AI-driven organization or fall behind. These sweeping edicts typically land on technical leaders: CIOs, CTOs, and CDOs who have built careers in technology, but not in AI. Those that sit closest to data are within the blast radius, effectively made into de facto AI leaders with little background in producing “AI.” What none of them were handed along with the title is a way to think about the problem.

However smart and experienced these new AI leaders are, they are at the mercy of vendors. They sit through pitch after pitch, each one promising AI value, personalized to the company, delivered at scale, with returns that arrive almost immediately. Every pitch reinforces the same message: we have cracked how to deliver that value, and your company is already behind. Without a background in data science, machine learning, and AI, these technical leaders are at an information disadvantage, relying on their own technical acuity rather than any means to disentangle the vendor claims.

The AI System Model

Throughout the 2010s and early 2020s, companies that successfully scaled machine learning (ML) programs focused on the systems that live around the model. The learnings from these applications, referred to as Machine Learning Operations (MLOps), were seen as a continuation of the DevOps movement of the 2000s. We approach LLM-based systems in the same light rather than selecting the most powerful LLM on the market, and hoping it performs miracles in our enterprise. The methodology below comes directly from our popular course *AI for Executives: Building a Rapid AI Blueprint*.

The central thesis of the AI System Model is that every layer depends on the ones around it. Data and context feed the model through repeatable retrieval, and the model's behavior follows from what the business feeds it rather than from how powerful the model is on its own. The application layer carries those outputs back to users at the moment they need them, aligned to how the work gets done. Governance runs through all of it, setting what data the model can see, what the system is allowed to do, and who answers when it is wrong. Skip any one of these layers and the system breaks.

See Figure 1.

Why We Wrote This

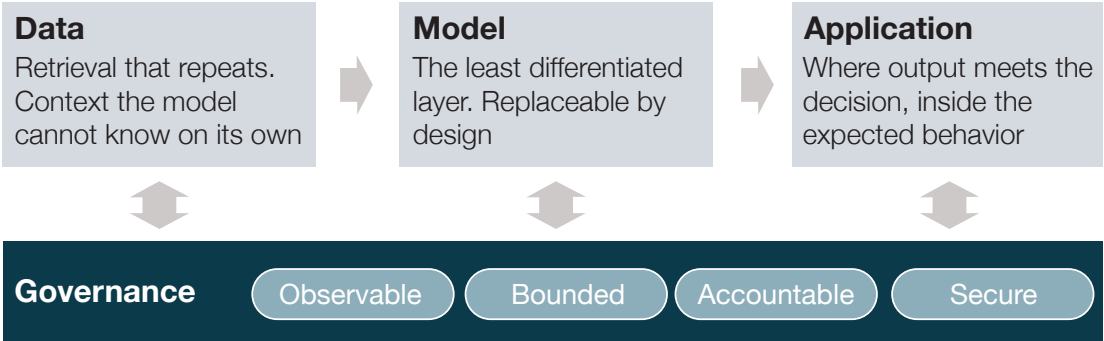
Caltech CTME, the Center for Technology and Management Education, is the professional and executive education arm of Caltech. We teach working professionals and their organizations across industries, both through open-enrollment courses and customized executive education built around a single company's problem. In all of it, we focus on repeatable methodologies rather than the leading model benchmark or flashy new product making headlines.

This paper comes out of that teaching, but not only from the classroom. Its author spends the rest of his time as a systems integrator and solutions architect at a boutique AI consultancy, building the kinds of systems described here. Several of the examples that follow come from that work, or from a co-author's, with client details removed. What follows was learned from systems that shipped and had to keep working, which is a harder test than a demo.

*Our course **AI for Executives: Building a Rapid AI Blueprint** was developed as a response to the numerous Fortune 500 companies we have worked with who have struggled to identify a practical path for AI adoption. It comprehensively draws from a deep case study bench of lessons in successful implementations of AI, and unsuccessful approaches that ended up in the dust bin. While the world of AI moves rapidly and most organizations are holding summits on the “Art of the Possible,” this course works on the premise of the “Art of Reality.”*

Figure 1
The AI System Model

The AI System Model



Data Layer

“A team of PhD-level experts in your pocket.” That was how Sam Altman introduced GPT-5 in the summer of 2025, unveiling what Open AI billed as the most capable model then available. Benchmarks followed, and news about the model’s ability to shift on its own between fast answers and deeper reasoning.

Quickly after OpenAI’s release, Anthropic upgraded their Sonnet and Opus models to great fanfare from the developer community. Leaders responsible for AI adoption took notice and the calls for accessing frontier lab models went through the roof.

Corporate obsession over high-performing models, based on popular benchmarks, often drives the AI adoption narrative. “If only our organization had access to GPT, Fable or Mythos, and so on, then we would be able to truly become an AI-first company.” The common misunderstanding is that the model is driving the performance of an AI system. Most organizations see the power of AI represented by these frontier lab models. Reality, however, is

that the model means much less than we think in an AI system. Most organizations lack data pipelines that allow for their organizational knowledge to be fed into the model. This is what drives model performance and behavior at the organizational level—not a subscription to the most recent Sam Altman or Dario Amodei pronouncement.

Picture the best operator your strongest competitor has, thirty years in the field, poached and starting Monday. They know the work cold and nothing about how you run. Hand them an RFP and they cannot answer it in your company’s voice, drawing on positions the firm has held for years. Ask them to work the sales pipeline and they do not know how it feeds downstream hiring. They can speak to anything their career taught them and almost nothing about your company, which is the instant ROI you were counting on and will not get on day one.

By analogy, this is how many technical leaders and AI evangelists see the world of AI. Purchase the subscription, get the cutting-edge model, and let it bring ROI. But organizations need to give those

models context. This context is the policies, procedures, plans, historical data pulls, and positions that the firm has held over the years— the justifications for a new position based on the experience of someone who has been there for 20 years. Not a benchmark breaking model.

As the AI vendor hype-train cascades upon these technical leaders, they find themselves under more pressure. But the answer has been looking them in the face since they began working at the company. If they can get the data into a format that the model can leverage, then the real gains and ROI will emerge. For many, that is easier said than done. The data is scattered and inconsistent, and the processes producing it are broken in ways no model will fix; instrumenting them as-is just automates the mess faster. That cleanup is the first project, and doing it well is what makes everything downstream possible. **That data, once it is useable, is the moat that the school of Altman has been preaching. They've just been looking in the wrong direction.**

Model Layer

The first thing to understand about models, particularly LLMs, is that they are a frozen form of knowledge. They can only recall what they were trained on, up to a fixed cutoff. They are generalists by default and will not know anything about your organization. With external tool calls, like the ability to search the internet, LLMs can be given agentic capabilities to call tools on your behalf and provide more updated information.

However, this runs against the instincts of most technical leaders, who came up on deterministic systems where the same query returns the same row every time, and a difference in output signals that something is broken. With a model, variation in the output is at the core of its purpose, which is precisely why a model can never be trusted to run unattended in the way a database or a reporting pipeline can. Feeding it the right context narrows the odds of a wrong answer considerably, but it

never drives them to zero, and that residual uncertainty is what the entire governance layer exists to manage.

Instead, technical leaders should treat the model as a commodity, one that is highly variable on the market. As frontier labs rush new models to market, your current champion model that you built your AI system around will seem dated in months. Instead, focus on building AI systems that are decoupled and not tied to one specific model. See every model as one that ships with an expiration date. If you build the system so that the model can be replaced, updated or refined, your AI system will be an upgradable moat, rather than a withering cost center.

Application Layer

The application layer represents when the user interacts with the system. From the screen they are presented with, the voice or chat exchange they speak into and read back, the handoff where one agent passes work to the next without a person watching, and the API call another system fires on their behalf. These are seen from two angles. The technical angle is what makes the system reachable and reliable. The behavioral angle is whether anyone trusts it enough to use it under real conditions. A system can be flawless on the first and still fail on the second, which is where most deployments quietly come apart.

Technical

The technical angle covers how the system is reached and used. One important caveat: the one using it is not always a human. With the rise of agentic capabilities, AI systems are responsible for acting on the user's behalf. Much of the ability to act depends on the context that was fed into the model and what the model decides to do with that information. This is where the application layer comes into play. Users interact with an AI system in various ways through prompt windows, strokes of the keyboard and voice or visual commands. The AI system presents information to users at

various stages for feedback.

The recent rise of agentic capabilities gives the Application layer new meaning. Many of the agent-based AI systems remove human interaction in favor of agency; giving responsibility to the system to make decisions and remove the human from the mundane. While agents provide a one-task service, agentic systems combine agent workflows in an orchestration of connections, all relying on each other. How these connections operate, when hand-offs of information move from one agent to another, is where the Application layer thrives.

Behavioral

Modern technical leaders are required to drive AI adoption. In fact, the mandates from Boards and C-Suite executives to “do more with less” are landing at a staggering pace. Rather than repeating this banal phrase, technical leaders should think long and hard about how the employees in their organization will use the AI systems they integrate into workflows. The key questions relevant here are primarily behavioral. For example, how do we train employees without overburdening their already busy schedules? How do we enforce compliance? What if users offload too much of their work and their cognitive capabilities decline on the job, resulting in AI slop in the workplace (i.e., AI workslop)?

None of these questions are technical, which is exactly why they get missed. When a deployment fails at this layer, the cause is usually a person: someone who doesn't trust the output, or trusts it too much, or works around the system entirely and keeps doing the job the way they always have. A vendor demo will never show you that failure mode.

So the behavioral questions deserve as much weight as the technical ones. Take training. Most of it should not be a feature tour; it should teach people when to lean on the system and when to check its work. Better yet, remove the need for training where you can by putting the AI inside a workflow people already run instead of handing them one more tool to open. Compliance works the same

way. Anything you have to police by hand will slip the first week everyone gets busy, so build the rule into the system's guardrails, what we called the Bounded bucket, rather than relying on good intentions. As for workslop—the fear that people offload so much judgment that quality erodes—the guard against it is *ownership*. Put a named person on the hook for the output and the incentive to think does not disappear. A system can run flawlessly and still fail if no one uses it well, and that is the part that decides whether the engineering mattered at all.

Governance Layer

There is a certain tension that we have felt throughout the previous three layers. As we are building the AI System, we are also designing rules of the road: what data can be fed into the model and what can't; what level of agency we want in our AI system; what the system should have access to and who should access it; who is responsible for making those decisions and what should happen when the AI system doesn't achieve its goal. These are all core questions that lead us to our final layer.

This Governance layer is broken down into four distinct buckets to ensure the AI system is Observable, Bounded, Accountable and Secure. The acronym “OBAS” helps us simplify a traditionally confusing enforcement dance.

Depending on your background, one bucket will tend to rise above the others. For the CISO, the Secure bucket will be the first barrier to a successful AI deployment. For the Data Officer, it will be Bounded to ensure that only those with a need-to-know should have access to information in the AI system. For the COO, the Accountable bucket will lead their thinking to ensure the AI system is owned by the right internal division and leader. Regardless of where you fall, these four buckets drive each layer of the AI System Model.

Observable

As AI systems get more complex and integrated at various levels of the enterprise, it gets more difficult to look under the hood for errors and irregularities. When deploying dashboards, analysts could look at the SQL code to determine why data warehouse queries were not working. In traditional machine learning (deep learning aside), data scientists and ML engineers could look at the values of models through feature importance and find which features (categories of information) were driving the system's performance. With the advent of LLMs, looking under the hood relies on looking at the data coming in and going out. As organizations adopt agents to act on their behalf, observability becomes paramount. The potential for downstream harm grows as agentic orchestration becomes more complex.

There are a few ways technical leaders can address observability in AI systems, particularly those that rely on LLMs for operations and agentic behavior. Giving access to engineers to diagnose the traces that LLMs and agents produce is paramount. Similar to how the technical community diagnoses a broken API connection giving a 400 warning, the same can be done for agents. Traces are similar to an API trace with an input and output, but far more complicated. Agentic behavior is derived on inputs from tool calls, retrieved context, applied prompts and various sub-agents spun up to run parallel processes. Looking at a trace of different spans is akin to trying to untangle a complicated knot.

The AI market is awash in observability platforms, from Arize.ai to open-source MLflow on Databricks. But creating alert systems is the responsibility of technical leaders to ensure that the right information is being provided to their teams at the right time. Too little, you miss the signal, while too much results in a burned-out team with alert fatigue. There is no technical right answer. Perhaps the answer lies in coordinating between technical leadership and their

engineers interacting with the AI systems.

Bounded

Most Fortune 500 companies have clear policies on what information is allowed to be used with AI and what is not. This is a narrative that has accelerated in recent years as the conversation has moved from “are you training on my data” to “are you training on my internal usage patterns?” What we allow the system to do is frequently described as AI system guardrails. Building around these guardrails is the practice of Harness Engineering. (Don't worry if you haven't heard about it; it has only been around for about 5 minutes). Whatever your definition of bounded is, ensuring that the AI system follows your organizational policies and procedures is paramount. When building AI systems, we frequently bound its abilities based on pre-defined requirements. No Personal Identifiable Information (PII) can be ingested or exposed to the Application layer. No secrets or system tokens can be put into the Model layer. The agent can only call whitelisted tools or MCP servers. Token limits are put on agents that run in loops to avoid a monthly model bill that drives your procurement team up the wall. These are all examples of bounded behavior that can operate at either the Data, Model or Application layer.

Accountable

Who is responsible when the system is wrong? Often, we find that an organization's brand is tied to the results its chatbots give. But LLMs no longer stay in the chatbot. They increasingly drive higher-impact decisions, from loan approvals and healthcare coverage to denial of care and resume screening. As the stakes rise, assigning accountability for the system's output becomes imperative.

The starting rule is one people nod at and then ignore: a model cannot be accountable, and neither can a vendor. Accountability lands on a person, and that person has to be named. Every AI system needs an owner who answers for its outputs,

the way a process owner answers for a report that goes out with the wrong numbers. So when the system denies a claim it should have approved, the answer to “who is responsible” should already be written down somewhere, not worked out in a hurry after the fact.

From there, scale the human oversight to the stakes. A tool that drafts marketing copy can run with someone checking it afterward. A system that approves loans, adjudicates care, or screens resumes cannot; those calls need a person in the loop before anything takes effect, plus a way to undo the decision when it turns out wrong. And none of this works unless the Observable bucket is doing its job, because you can’t hold anyone accountable for a decision you can’t reconstruct. Without the trace, “the system got it wrong” is just an argument; with it, you have a record of what the model saw, what it did, and who approved it. One last point on vendors: buying the model does not offload the accountability. The brand on the screen is still yours, and so is the apology when it fails.

Secure

For many readers, the Secure bucket is the de facto priority. Security touches every element of the AI System Model, from encryption of data at rest and in transit, to the guardrails placed before and after injection into the model, to how users access the application. Furthermore, the area of LLM security is frequently changing. As organizations attempt to react to the shifting landscape of LLM deployment security, the reality is it is becoming more difficult to assess. It starts with the data itself. Data Loss Prevention strategies, PII reduction, secret tracking should all be embedded in the security guardrails. As for the model layer, knowing how to deal with prompt injections and jailbreaking of performance is critical. Finally, the output of the model should have protections against outputting protected information.

Together, the Governance layers of Observable, Bounded, Accountable and Secure work as four questions asked at every layer of the system, from the data that enters it to the output it returns, and asked again each time the system changes.

Activating Reality

What these leaders were never handed, along with the title, was a way to think about the problem. That is what the AI System Model gives them. The point was never to chase the smartest model or wait on the next one. The point is the system itself, because the system is the one thing a vendor can’t sell you and a competitor can’t copy.

Start with data and context; that is where your real advantage sits, and most organizations already own it without realizing it. Treat the model as the commodity it is, and build so you can swap it when a better one lands, because one will. The application layer is where trust is won or lost, since a real employee has to use the system when the pressure is on, not in a controlled demo. And governance runs underneath all of it, relying on four buckets you keep asking at every layer: Is it **Observable, Bounded, Accountable, Secure**? Drop any one and the system starts to fail, slowly enough that you might not notice until it already has.

This is what we mean by the Art of Reality. Not the conference-keynote version about the art of the possible, but the ordinary work behind it: getting your organizational knowledge into a shape the model can use, deciding what the system is and isn’t allowed to do, and naming the person who answers when it gets something wrong. The organizations that come out ahead this decade won’t be the ones that bought the biggest model. They will be the ones that built the better system around it. The moat was never the model. It was everything the company already had, sitting unused.



ABOUT CALTECH CTME

Caltech's Center for Technology and Management Education (Caltech CTME) educates the scientists and engineers already at work in industry, the people responsible for systems that have to function.

Our clients are organizations whose competitive position is decided by engineering: semiconductor manufacturers working at atomic tolerances, equipment makers turning machines into autonomous and electrified platforms, capital markets firms running on quantitative systems, and the aerospace, energy, and government programs that have always turned on technical execution. What they share is a workforce of technical professionals and a need to keep innovating faster than their industry is changing.

Caltech CTME is part of the California Institute of Technology, an institution built on physics, computation, and applied mathematics. NASA's Jet Propulsion Laboratory is a division of Caltech, and among the most demanding technical enterprises in the world. That heritage shapes what Caltech CTME teaches. Systems engineering here is a way of thinking rather than a method alone, taught alongside the practices that move an innovation pipeline from idea to deployment.

Programs are taught by people who have done the work, practitioners from industry alongside researchers from across the Caltech community. They are delivered two ways: custom programs designed for a single organization, and open-enrollment courses that individuals and teams can join directly.

© Caltech 2026. All rights reserved.

To request permission to reprint, contact execed@caltech.edu. For current programs and updates, visit ctme.caltech.edu or follow Caltech CTME on LinkedIn.