

THREADNEEDLE LABS

Beyond the Model

THE NEW ARCHITECTURE OF
ARTIFICIAL INTELLIGENCE



ZIA KARIM AND FARIS KARIM

Free excerpt · Introduction & Chapter 1

About this excerpt

Beyond the Model: The New Architecture of Artificial Intelligence

By Zia Karim and Faris Karim

This free reading sample contains the complete Introduction and Chapter 1, “The Capability Trap”, from the first edition. It introduces the book’s argument: as models become more capable, the frontier moves outward to the architecture of intelligence around them.

The full book develops that argument through adaptive intelligence, evidence, agency, human judgement and institutional control. Its complete contents appear on the next page.

In this excerpt

Introduction	4
Chapter 1: The Capability Trap	13
References cited in this excerpt	22
About the authors and further reading	23

Copyright © 2026 Zia Karim and Faris Karim. First edition, October 2026.

This PDF is provided for personal reading and may be shared unchanged with attribution. All other rights reserved.

Set in Fraunces and Geist. Published by Threadneedle Labs.

In the full book

Five Parts · Eighteen Chapters · One connected argument

Preface

Introduction: The Model Is No Longer the Whole Problem

PART I · AFTER THE MODEL

- 1 The Capability Trap
- 2 Intelligence Is Becoming a System
- 3 The Economics of Thought

PART II · INTELLIGENCE THAT CAN REASON ABOUT ITSELF

- 4 Adaptive Intelligence
- 5 Complexity Must Earn Its Keep
- 6 The Architecture Can Become the Experiment

PART III · WHAT DOES THE MACHINE KNOW?

- 7 The Epistemic Problem
- 8 Evidence Before Confidence
- 9 Reasoning Provenance
- 10 The Unknown Unknown Problem

PART IV · FROM ANSWERS TO ACTION

- 11 Intelligence Becomes Agency
- 12 Agents Need Constitutions
- 13 Bounded Autonomy
- 14 The Right to Interrupt

PART V · INTELLIGENCE INSIDE INSTITUTIONS

- 15 Humans Are Part of the Architecture
- 16 Organisational Intelligence
- 17 Computational Governance
- 18 Institutions Must Retain Sovereignty

Conclusion: The Frontier Moves Outward

References · About the Authors

INTRODUCTION

The Model Is No Longer the Whole Problem

Imagine a bank reviewing a proposed change to a lending policy. An AI system reads the old policy, summarises the proposed text, checks several regulations, compares previous decisions and drafts a recommendation. The model that writes the recommendation may be excellent. Yet almost every consequential question concerns something outside it. Which version of the policy did it read? Were the regulations current? Did the system distinguish a legal obligation from an internal preference? What happened when two sources disagreed? Who authorised the recommendation to become a change in practice? Could an auditor reconstruct the decision six months later?

There is no single model benchmark that answers those questions. They describe an intelligence architecture: the way information enters a system, becomes a claim, supports a decision and eventually changes the world.

For much of the modern AI era the model was the bottleneck. The prevailing strategy was to improve the capacity of the learned component: more data, more computation, better training, better evaluation. That strategy produced remarkable gains. The argument of this book does not require diminishing them. Indeed it depends on them. Weak models cannot support the kinds of systems discussed here. But as capability rises, the marginal value of improvement depends increasingly on what surrounds the model. Better reasoning inside a poor information system can generate more elaborate mistakes. Better tool use inside a poor authority system can generate faster ones.

The phrase *beyond the model* should therefore be read literally. It points to the design work outside the learned component. Some of that work resembles traditional software engineering: interfaces, tests, access control, versioning. Some is less familiar: deciding when a machine has enough evidence to proceed, how much thought a task deserves, whether an architecture should change itself, how a person can intervene without becoming a ceremonial approver. The questions meet in practice. A system cannot sensibly decide how long to reason unless it knows the stakes. It cannot know the stakes without understanding the action it may take. It cannot take that action responsibly without an account of who delegated authority.

The useful unit of analysis is therefore the intelligence system. It observes, retrieves, represents, reasons, selects, acts, evaluates and adapts. Those verbs do not describe a compulsory pipeline. A simple request may need only a model and an answer. A complex request may call for a database, a search tool, a planner, a test harness, a second model and a person. The architecture should make those choices because the task warrants them, rather than because an elaborate design looks impressive.

This is an argument about allocation. Intelligence is scarce even when model access is cheap. Reasoning consumes time and energy. Retrieval consumes attention and can import misleading material. Verification has a cost. Human review has a cost too, including the cost of distracting people with decisions they cannot realistically assess. The rational objective is neither minimum computation nor maximum deliberation. It is the right amount of work, by the right components, for the consequence at hand. A spell-check suggestion and a change to a medical record should not pass through the same decision path.

It is also an argument about knowledge. Language models generate statements in a form that encourages belief. Their training gives them broad regularities and sometimes precise facts, but an articulate answer does not carry its own warrant. A system may be correct by memory, correct by inference, correct because it retrieved a current source, or accidentally correct for the wrong reason. Those cases differ when circumstances change. An answer that cannot say

which assumptions it used is hard to challenge. A recommendation that cannot identify the evidence behind it is hard to govern.

Research on retrieval and long context already gives us a warning. Adding material to a prompt does not ensure that the relevant material will be used. Nelson Liu and colleagues found that, in their tests, models often performed worse when relevant information sat in the middle of a long context than when it appeared near the beginning or end (Liu et al., 2024). Akari Asai and colleagues showed that selective retrieval and critique could improve performance in their experimental setting (Asai et al., 2024). These are particular results, not general guarantees. Together they make the architectural point: the presence, selection and treatment of evidence matter.

We must also distinguish intelligence from agency. A system that drafts an email is different from one that sends it. A system that suggests a code change is different from one that merges it. An agent can have a good understanding of a task and still lack the right to act. Human institutions have long recognised this distinction. A junior analyst may understand a transaction perfectly and still have no signing authority. Artificial intelligence makes the distinction operationally urgent because a model can turn a sentence into a tool call with very little friction.

The right architecture will sometimes allow autonomous action. It will sometimes insist on a human decision. Between those endpoints are many useful arrangements: observe, recommend, prepare, act after approval, act within defined limits, and escalate when conditions change. The boundary should be attached to the action, the evidence, the reversibility and the delegated authority. It should not be inferred from a vague impression that the system is generally trustworthy.

This leads to the human question. A person placed at the end of a machine process is often called a safeguard. Sometimes that is true. Sometimes the person has too little time, context or practical power to do anything but click. Human participation is meaningful only when the work is designed around what people can contribute: purpose, values, situational understanding, responsibility and the ability to reframe a problem. Machines can offer breadth, speed, consist-

ency, calculation and tireless monitoring. Neither list is a moral hierarchy. The design problem is to combine them without confusing apparent fluency with judgement or human presence with control.

At the institutional level, these choices become questions of sovereignty. Organisations will increasingly depend on intelligence they did not train and may not fully inspect. That is not by itself a defect; institutions have always relied on outside expertise. But expertise becomes dangerous when the institution loses control over the evidence, policies, permissions and records through which it acts. An external model may contribute to a decision. It should not silently become the final interpreter of an organisation's obligations or the sole keeper of its memory.

The book follows that widening circle. Part I explains why capability alone has become an inadequate frame and why intelligence must be treated as a system with an economy. Part II examines adaptive architecture: how a system can select methods, resist unnecessary complexity and test changes empirically. Part III asks what machine knowledge means in practice, beginning with uncertainty and ending with the harder problem of missing frames. Part IV follows intelligence across the boundary into action, where permissions, limits and interruption matter. Part V locates the system among people and institutions, where governance must be effective rather than decorative.

Several distinctions will recur. A *model* is a learned component that produces outputs from inputs. An *intelligence system* is the wider arrangement that uses models, information, tools, rules and people to pursue a task. *Evidence* is material that can support or challenge a claim; it is not identical to a citation pasted beside an answer. *Provenance* is an inspectable account of the sources, transformations and decisions that produced an outcome. *Authority* is permission to change state on behalf of someone else. *Sovereignty* is an institution's practical ability to set, inspect and revise the terms on which intelligence affects its decisions. These are working definitions. Later chapters will test their boundaries.

The propositions in the pages ahead have different statuses. There is published evidence that additional reasoning, retrieval and tool use

can improve performance on certain tasks. There is also published evidence of their failure modes. There are engineering claims about what designs should be tested. And there are normative claims about who should remain answerable for consequential action. We will not treat a promising experiment as proof of a universal architecture, nor a governance principle as if it were an empirical discovery.

One can object that future models may absorb much of this architecture. A model might learn to retrieve only when necessary, calibrate its own uncertainty, respect permissions and adapt its reasoning without external scaffolding. Some of that is already happening. Yet the objection has a limit. A model can learn to estimate uncertainty; an organisation still has to decide what uncertainty it will tolerate. A model can learn to propose an action; someone still has to define its authority. A model can explain a conclusion; the institution still needs evidence it can inspect and preserve. Even if many functions move inside the model, the boundary between its capability and the world's authority remains a design choice.

There is a useful historical analogy, though it should not be pushed too far. Early computing was often discussed through the capability of a machine: how much it could calculate and how quickly. As computers became useful in organisations, questions of operating systems, databases, networks, security and human workflow became central. Faster processors remained valuable; they were no longer a complete description of progress. AI is undergoing a related broadening of the engineering problem. The analogy is limited because models deal in language, inference and action under uncertainty, and because their outputs can carry an appearance of judgement. It nevertheless helps explain why success in the core component can reveal a larger architecture.

The book's use of *architecture* is deliberately broad but not vague. It refers to choices that can change an outcome: which information is available, how it is represented, which procedure is used, what checks intervene, who may act and how the result is recorded. If changing a choice can alter accuracy, cost, authority or accountability, it belongs to the architecture. This definition includes technical interfaces and organisational rules. It does not turn every social fact

into a software problem. Rather, it insists that software placed inside an institution will interact with the institution's incentives and responsibilities whether or not designers draw them in the diagram.

Consider two AI assistants that use the same model to review a supplier contract. The first receives a PDF, generates a summary and lists concerns. The second identifies the signed version, compares it with the approved template, links each concern to exact clauses, checks whether the proposed action is within the reviewer's authority and records the disposition of each issue. The second may be more reliable because of its architecture, even though the model is identical. It may also be slower and more costly. The right choice depends on whether the task is an informal preview or a decision that will bind the organisation. Architecture does not imply maximal process; it implies fit between process and consequence.

This fit has an empirical side. A design should be judged by the work it does, not by the sophistication of its description. Adding an agent, a memory layer or a verifier is a hypothesis about improved outcomes. It can be tested against a simpler arrangement. Adding a person is also a hypothesis: their review should prevent a class of mistake or supply judgement the machine cannot legitimately provide. If the person receives no evidence and clicks through under pressure, the architecture has gained a nominal safeguard and little real control. The same scrutiny applies to every component.

The fit also has a normative side. No benchmark can tell an institution what obligations it owes a patient, customer, citizen or employee. A model can help analyse the consequences of a policy, but it cannot derive the legitimacy of a policy from predictive accuracy. The architecture must therefore make space for authorised human decisions about purpose, rights, acceptable risk and redress. This is not an argument for human superiority at every task. It is an argument that some questions concern who is entitled to decide, not only who can predict well.

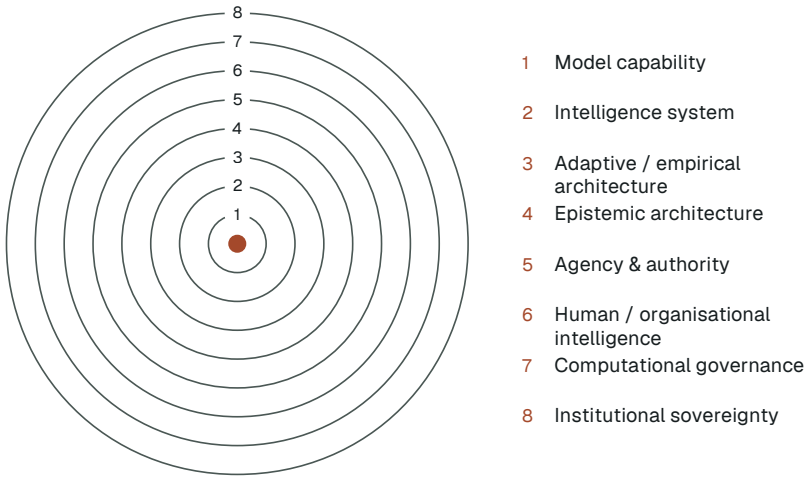
The model's internal opacity is sometimes treated as the decisive obstacle to trustworthy AI. It is an obstacle in some contexts, but it is not the only one. Human experts are also partly opaque. Institutions work with them by defining roles, requiring evidence, preserving re-

cords, reviewing outcomes and allowing challenge. AI can be governed through analogous mechanisms, though its speed, reproducibility and ability to act through software require new precision. We do not need perfect access to every internal computation before we can improve the external conditions under which a system operates. Nor should external controls be used to excuse a model that is too unreliable for the task.

The argument will sometimes sound demanding because it is addressed to consequential use. A low-stakes drafting tool can be valuable with few controls. A system that affects money, health, rights or public decisions deserves a stronger account of evidence and authority. The chapters develop principles that can be applied in proportion. They should not be read as a requirement that every email suggestion pass through a committee. That would be another failure of allocation.

The architecture can fail in ways that make each chapter necessary. A system may route a serious case to a cheap model because it misjudges difficulty. It may retrieve the right document and then omit the exception. It may preserve sources yet lose the chain between source and claim. It may give an agent a narrow mandate but a broad credential. It may escalate to a person who lacks the power to intervene. It may record decisions in a provider-specific format the institution cannot later use. These are not variations of one ‘hallucination’ problem. They call for different methods of design, evaluation and governance.

They also interact. A weak epistemic system can turn bounded autonomy into an illusion because its action limits depend on facts it cannot establish. A strong evidence system can be wasted if the human reviewer sees only the model’s conclusion. An adaptive architecture can improve reasoning while eroding auditability if it changes methods without recording them. Institutional sovereignty can be claimed in policy while practical control resides in tools and records elsewhere. The book’s progression is intended to expose these dependencies. Each layer broadens the meaning of intelligence without making the earlier layer disappear.



The widening architecture of intelligence. Each wider layer retains the model within it.

The argument does not require confidence that models will improve at any particular rate. Some capabilities may advance quickly and others slowly. The architecture question matters wherever models are useful enough to participate in work whose evidence, costs and authority matter. If model progress slows, architecture can help extract more reliable value from existing capability. If it accelerates, the same architecture becomes more urgent because the range of actions available to systems expands. The thesis is about a change in the structure of the problem, not a forecast of release dates or vendors.

There is an important limit to the title. ‘Beyond the model’ does not mean that the model becomes unimportant or that the architecture can be designed independently of its capabilities. A better model may make a simpler workflow possible. A weaker model may require more external checks or may be unsuitable for the task altogether. The surrounding architecture should be designed with empirical knowledge of the model’s strengths and failure modes. The point is to resist treating the model as the whole explanation of an application. The relationship runs in both directions: model capability shapes architecture, and architecture determines what that capability can responsibly accomplish.

The same reciprocity applies to institutions. A technically elegant system can be unusable in an organisation that cannot maintain its sources or respond to escalations. An institution with excellent professional judgement may still struggle if its tools erase provenance and force decisions into the wrong categories. The architecture is a joint arrangement of technology, work and authority. It should be evaluated where these meet. The book's claims become strongest when they lead to concrete changes in that arrangement and weakest when they are reduced to slogans about AI's future.

The central claim is straightforward. As AI models become sufficiently capable, the hardest problems shift from building smarter models to designing the architecture through which intelligence reasons, knows, acts, adapts and remains under human and institutional control. The model is no longer the whole problem. What follows is an attempt to take the rest of the problem seriously.

The Capability Trap

Suppose a team replaces its existing language model with one that scores markedly better on coding benchmarks. The demonstration is persuasive. It reads an issue, writes a patch and explains the change. Yet the production system may improve very little. The agent still edits the wrong branch on occasion. It overlooks the repository's local conventions. It treats a passing unit test as proof that a change is safe to release. Its confident summary reaches a reviewer who has no record of the files it inspected. The model improved; the work did not improve in proportion.

This is the capability trap. Progress in a component is mistaken for progress in the task that component is meant to serve. The trap is especially attractive in AI because model capability is visible and comparatively easy to compare. A benchmark provides a number. An eloquent demonstration provides a story. The quality of the surrounding system is harder to compress. It depends on where a request came from, what context was available, which tools were allowed, how errors were detected and what happened after the answer appeared.

We need to distinguish four related things. *Model intelligence* is the learned component's ability to perform tasks under specified conditions. *System intelligence* is the arrangement's ability to combine models with information, tools and checks. *Task intelligence* is the fit between that arrangement and a particular problem. *Operational intelligence* is the ability to work repeatedly under changing conditions, with real users and consequences. These are not separate substances. They are perspectives on performance. A strong model may improve all four. It does not guarantee any of the last three.

Take a contract review. The model may accurately identify a limitation-of-liability clause in a supplied document. That is model capability. Whether the system selected the signed version rather than an earlier draft is an information problem. Whether it understands which jurisdiction and transaction type matter is a task problem. Whether a lawyer can trace the comparison, correct an error and maintain a consistent process across hundreds of documents is an operational problem. A headline score on legal question answering tells us little about those later questions.

The conventional assumption has a sound core: better models usually make more applications possible. We should be wary of romanticising orchestration as a substitute for competence. A system of mediocre components can become an elaborate way of producing mediocre work. But the opposite mistake is now common. Capability is treated as a universal solvent, expected to dissolve poor context, ambiguous authority, missing data and weak evaluation. It cannot. It can sometimes compensate for them in a demonstration, which makes the underlying defect harder to see.

The effect resembles an old engineering problem. Improvements in one stage of a process reveal bottlenecks elsewhere. Faster computation does not make a slow disk disappear. A more accurate sensor does not make a flawed control policy sound. In AI, stronger models make previously hidden architecture visible. When a model could barely draft a useful answer, the system around it seemed secondary. Once it can write credible answers and operate tools, the quality of the system becomes the limiting factor in a growing number of tasks. That is a contingent claim about mature applications, not a declaration that model research has ended.

The first hidden bottleneck is context. A model can work only with the information it has, plus what it has learned during training. In a live setting, information has versions, owners, expiration dates and conflicting interpretations. The analyst's spreadsheet, the approved policy and the public website may all describe the same product differently. Feeding all three into a long prompt does not decide which is authoritative. Context construction is a decision about relevance,

trust and time. It should be treated as part of the system’s reasoning, not as clerical preparation.

Long context windows make the problem subtler. They permit more material to be supplied, but more material is not automatically more usable. In the tests reported by Liu and colleagues, the position of relevant material affected performance on long-context tasks. This does not imply that every later model behaves in the same way. It does show why a context window measured in tokens is a poor proxy for dependable access to information. Retrieval, ranking, decomposition and explicit source checks can matter as much as nominal capacity.

The second bottleneck is tool use. A model may know how to write code, but changing a repository safely requires a sequence of interactions with state. It must find relevant files, understand tests, inspect the working tree, edit, run checks and notice unexpected changes. Each operation has a failure mode distinct from language generation. The SWE-bench proposal made this point by framing real software issues as work across files and execution environments. The early baseline results in that paper are already historically dated; the durable lesson is in the task definition. Real work asks for coordinated interaction with an environment, not merely a plausible patch printed in a chat window.

Tool use also changes what counts as a mistake. If a model says that a file was changed when it was not, the answer is false. If the system writes to the wrong file, the state is now different. If it merges a flawed patch, others inherit the consequence. The same token generation mechanism may be involved, but the system-level risk is categorically different. A model can be exceptionally capable at selecting a command and still need constraints on where that command may run.

The third bottleneck is evaluation. There is a temptation to define success by whatever can be measured quickly: a benchmark score, a thumbs-up, a passing test. These measures have value, but they answer narrower questions than the application asks. A test suite may not cover the requested behaviour. A user may like a summary because it is confident and brief. A coding benchmark may reward a

patch that passes tests while leaving a security or maintenance problem. Evaluation must begin with the real failure modes of the task, including failures that appear only after the model's output leaves the interface.

Good evaluation is therefore architectural. It asks what the system saw, what it did, what it could have done, and how someone would discover a mistake. One application needs an external fact check. Another needs a compiler. Another needs a second professional to sign a recommendation. Yet another needs a transaction limit. A single model score cannot replace those checks, because the checks exist for different reasons. They test the fit between the system and its operating environment.

The fourth bottleneck is authority. A model may produce excellent advice on a matter it is not authorised to decide. Organisations already separate knowledge from permission. A finance analyst may see that a payment is legitimate but still lack authority to release it. In software, an engineer may write the right fix but lack the right to deploy it alone. AI systems should inherit this separation, not flatten it. The question 'Can the model do this?' is insufficient when the system can alter a ledger, publish a statement or send a message in somebody's name.

Cost is a fifth bottleneck, but it is often discussed too narrowly. Token charges matter. So do latency, energy, infrastructure, review time and the costs of mistakes. A more capable model may reduce re-work enough to justify its price on difficult tasks. It may waste money and time on easy ones. A stronger model may also invite more ambitious automation, creating exposure that did not exist before. The relevant comparison is the cost of the whole process, including the consequences of error and delay.

This complicates the popular idea of a single best model. 'Best' for which task, under which constraints, using what data, with which failure cost? The answer may vary even within one workflow. Extracting a date from a well-formed invoice is different from judging whether a disputed exception is consistent with policy. A portfolio of components can be sensible, but portfolios introduce their own routing and maintenance problems. We should resist turning model plur-

ality into a fashion. The purpose of selection is to improve task outcomes, and the selection mechanism itself has to be tested.

The capability trap is also a management trap. Buying access to a stronger model is a discrete decision. Improving an intelligence system demands attention to records, workflow, responsibility and incentives. Those are less glamorous and harder to delegate. An organisation may therefore keep upgrading the model while leaving the process it serves largely intact. It gains fluent output attached to an old decision structure. Sometimes that is useful. Sometimes it makes the old defects less visible because the new interface looks so competent.

There are limits to the architectural argument. In domains where models remain plainly weak, model improvement is still the most valuable investment. A weak speech recogniser cannot be rescued by an elegant escalation policy if it mishears every other sentence. A poor scientific model cannot be made reliable merely by keeping provenance. Architecture should not become an excuse for accepting inadequate capability. The point is to diagnose the bottleneck at the level of the task rather than assume it always sits in the same place.

We can make that diagnosis concrete. Hold the task distribution fixed and vary one component at a time. Compare the current model with a stronger one under the same information and authority rules. Then hold the model fixed and improve retrieval, verification or tool constraints. Observe not only average accuracy but the tail: the cases where the system is confidently wrong, incurs a costly delay or acts beyond its mandate. Examine whether gains persist when documents change, users phrase requests differently and the environment develops faults. This is ordinary experimental discipline applied to the whole intelligence system.

The strongest version of the capability argument says that sufficiently advanced models will learn to handle the surrounding work themselves. They may. But even then, a system must decide what information they may see, what tools they may use, what counts as evidence and which actions they may take. Those decisions may be delegated in part, but delegation itself is an architectural act. The

larger the model's capability, the more consequential the terms of that delegation become.

One reason the trap persists is that the proof of a component's capability and the proof of a system's reliability have different shapes. A model provider can show a set of difficult examples and a benchmark score. An application team must show what happens across routine work, unusual work, changing data and operational incidents. The second proof is more expensive and less photogenic. It has to include cases in which the right answer is to decline, delay or refer. Those outcomes look unimpressive in a demo, yet may be evidence of a better system.

The pressure to choose a single number can distort development. Suppose two assistants each complete ninety per cent of coding issues. One fails visibly when it cannot find a relevant file. The other silently edits a neighbouring function and reports success. An average completion score may treat them as similar. The second creates a heavier review burden and greater release risk. The institution's objective should reflect the difference. It might measure unsupported claims, unauthorised actions, defects discovered after deployment and time required for a reviewer to understand a patch. These measures are harder to collect, but they are closer to the work.

The trap also affects procurement. A vendor's claim that its model is stronger is a claim about a component under certain tests. A buyer may reasonably care about that claim, but the purchase decision concerns a workflow. Does the product preserve source records? Can the buyer define permissions? Can the system be tested against the buyer's own cases? What happens if the provider changes the model? Does the human reviewer have enough information to challenge the answer? None of these questions can be settled by a leaderboard. They are also not demands for bespoke software in every organisation. They are the ordinary questions one asks before placing a powerful component into a consequential process.

There is a subtle inversion in the relationship between model capability and trust. When models were obviously weak, users checked their work because mistakes were expected. When output becomes fluent and usually useful, checking feels like friction. The risk may

therefore rise even as average model quality improves. This does not mean better models are worse. It means that human and organisational behaviour changes in response to them. A system evaluation has to include that behavioural response. The way an answer is presented can affect whether a person reads its sources, notices uncertainty or approves an action.

Think of a financial analyst receiving an AI-generated comparison of two companies. The model may correctly compute a ratio from numbers it was given. But if one company's figures use a different reporting period, the comparison is misleading. A review focused on arithmetic will pass it. A stronger model may even write a more persuasive explanation of the misleading comparison. The intervention needed is a check on comparability: dates, accounting definitions and source versions. This is a task-specific architectural check. It may be implemented by rules, a model or a person, but its function must be visible in the process.

The same pattern appears in public administration. An assistant may accurately summarise a citizen's application and identify the relevant rule. Yet if the agency's record lacks a recent appeal decision, the recommendation is defective. The model's competence in reading the available material is not the bottleneck. The bottleneck is whether the system knows that the record is incomplete, can seek the missing decision and will avoid acting until it has it. Better summarisation can reduce labour while leaving that failure intact. The question is what evidence the workflow demands before a recommendation acquires authority.

Operational intelligence has a maintenance dimension as well. A system that works on the day it is evaluated may degrade when documents change format, a tool returns a new error code or a policy is revised. Model upgrades can break assumptions in prompts or routing rules. A retrieval index can contain material that was once current and is now obsolete. These are ordinary maintenance problems with unusual epistemic effects: they change what the system appears to know. A mature application needs a way to detect them, assign ownership and correct them. The system should not depend on the

hope that the model will notice every broken surrounding component.

One way to locate the real bottleneck is to compare error classes rather than aggregate success alone. Are failures due to missing information, mistaken interpretation, poor tool execution, weak verification, authority confusion or user misunderstanding? The classification will never be perfect. It can still guide investment. If most costly errors come from stale policy documents, buying a stronger model may be less effective than governing the document store. If most come from genuine reasoning difficulty, model improvement or a deeper reasoning path may be decisive. If most come from tool permissions, neither change addresses the risk.

This analysis can unsettle organisational boundaries. Model teams may own the component while operations teams own the workflow, security teams own permissions and legal teams own policy. The system's intelligence depends on their joint work. An architecture that crosses these boundaries needs named responsibility for the whole. Otherwise each group may report that its component performed as designed while the outcome fails. The claim that intelligence is becoming a system has a managerial consequence: somebody must be accountable for system behaviour across interfaces.

There is one final diagnostic: ask what would have to change for the system to become trustworthy. If the answer is 'a model that reasons better about this class of problem', the bottleneck may indeed be model capability. If the answer is 'a current policy source, a permission check, a reviewable record and a way to stop', the bottleneck lies elsewhere. The distinction is not ideological. It directs work to the layer where improvement can matter. It also prevents teams from expecting a model upgrade to solve a problem whose cause the model never controlled.

The capability trap is easiest to escape when the unit of success is a completed human task rather than a model response. Did the analyst reach a better decision? Did the engineer ship a safe fix? Did the customer receive a correct and timely resolution? These outcomes are harder to measure, but they expose the real value of capability. They also show why a system that asks a clarifying question or re-

fuses an unsupported action can outperform one that always produces a polished answer. Completion is not the same as usefulness.

The diagnosis may change as models improve. A model upgrade can remove a genuine reasoning bottleneck and reveal that source management is now the dominant failure. A better retrieval service can then reveal that approval is the slow point. This is normal systems behaviour. Progress moves the constraint. Teams should therefore revisit their error analysis after major changes instead of assuming the original bottleneck remains. The right architecture is a moving fit between capability and task, not a monument to the conditions under which the application was first built.

There is also a distinction between improving average work and extending the frontier of possible work. A stronger model may enable a scientific analysis that no prior system could attempt. That is real progress even if its first operational architecture is rough. The point is to avoid mistaking the demonstration of a new possibility for a dependable service. Research can celebrate the new capability while engineering asks what conditions would make it repeatable, verifiable and appropriately governed. Both attitudes are needed. Confusing them is what creates the trap.

The useful conclusion is neither a sceptical dismissal of models nor a promise that architecture can cure every weakness. It is a change in the unit of analysis. We should assess the performance of intelligence where it meets a task and an institution. The model matters enormously. It is one part of the result. To understand the rest, we need to look at the system it inhabits.

References cited in this excerpt

Asai, Akari, et al. (2024). “Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection.” ICLR.

Jimenez, Carlos E., et al. (2024). “SWE-bench: Can Language Models Resolve Real-World GitHub Issues?” ICLR.

Liu, Nelson F., et al. (2024). “Lost in the Middle: How Language Models Use Long Contexts.” *Transactions of the Association for Computational Linguistics*, 12, 157–173.

About the Authors

Zia Karim

Zia Karim is an entrepreneur, advisor, author, and technologist. For more than three decades, he has built companies, products, and teams around a recurring question: why do some ambitious ideas become real while others stall?

His work has spanned institutions such as Goldman Sachs and J.P. Morgan, as well as startups, fintech, emerging markets, AI, and organisational design. The constant has been execution: better decisions, better systems, and better ways of turning intent into results.

He also writes about these questions, drawing on what he learns from building companies, products, research programmes, and new institutions.

Faris Karim

Faris Karim brings a background in computer science, artificial intelligence and economics to the study of intelligent systems. He holds a master's degree in computer science and artificial intelligence from the University of Pennsylvania and an undergraduate degree in economics from Vanderbilt University.

His focus is AI: how it works, what it makes possible and how to turn its capabilities into useful systems. In *Beyond the Model*, he brings this technical perspective into conversation with the economic and institutional questions surrounding AI—how intelligence is deployed, how its value is assessed and how its use remains accountable.

Continue the argument

Explore the full book

<https://beyond-the-model-marketing.the-five-year-plan.workers.dev>

Explore Threadneedle Labs research

<https://www.threadneedlelabs.com/research/>