CODITATION

MAR 2026

GEN AI: PLAYBOOK

10 Key Practical Lessons on Generative AI — from real deployments across healthcare, logistics, retail, Fin-tech and Mar-tech.

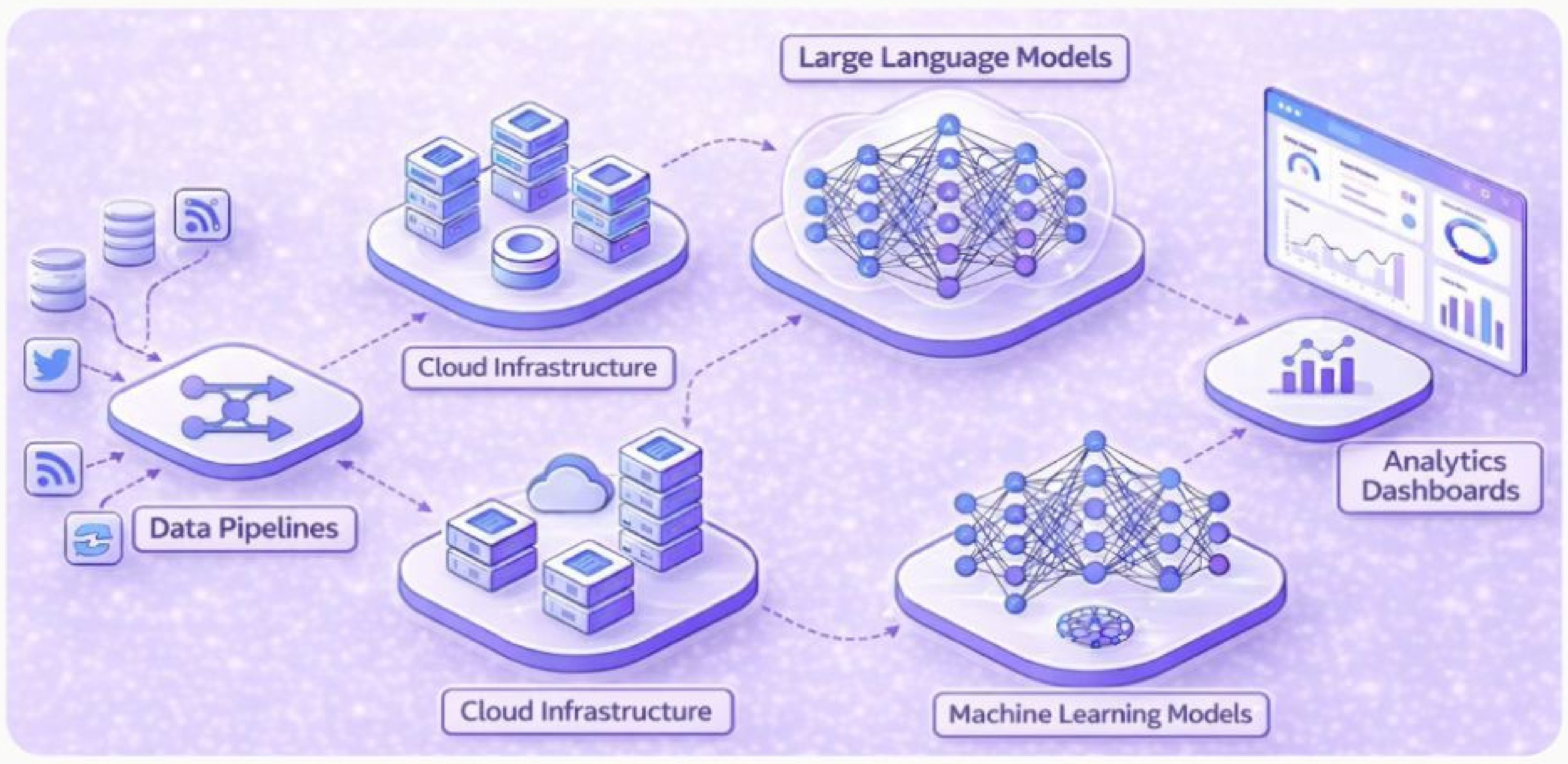
CODITATION

ABOUT US

Coditation

We are a trusted Digital Engineering and Enterprise Solutions partner. We blend engineering expertise, product mindset, technology know-how, and industry experience to help our clients stay ahead of the curve and consistently deliver value to their business. Our offerings and services have enabled global Fortune500 & mid-market enterprises and technology companies to level-up, gain competitive advantage, and successfully undergo transformation.

Building the next generation of digital products



TECHNOLOGY EXPERTISE:

1. Data Platforms
2. AI
3. Machine Learning and Computer Vision
4. SoftOps
5. Scalable Engineering
6. Digital Experience and Apps

Helping build scalable, practical solutions

INDUSTRIES:

1. Logistics
2. Healthcare and Life Sciences
3. FinTech
4. Energy & Industries
5. Digital Commerce
6. HiTech

GEN-AI

"GenAI success isn't primarily a technology problem. It's a problem of selection, design, integration and operational discipline."

Generative AI has grown from "interesting experiment" to "boardroom priority" at a remarkable pace. Engineering teams are eager, budgets are increasing and executives are getting more involved.

But here's what we've also seen: for every GenAI deployment that actually works there are dozens that stall or underdeliver.

// CODITATION'S PERSPECTIVE

At Coditation Systems, we've had the chance to build and ship AI-powered solutions across a wide range of industries from healthcare, financial services, logistics, retail to SaaS platforms.

We've compiled clear patterns — practical lessons about what separates the projects that create real value from those which become expensive science experiments.

This document summarizes those lessons into a practical playbook. It's written for technology leaders, product managers, founders and decision-makers who want to move past the hype and build GenAI solutions that are reliable, cost-effective and genuinely useful.

10 CHAPTERS AT A GLANCE

// INTRODUCTION

THE GEN-AI MOMENT WE'RE ACTUALLY IN

"The real challenge is bridging the gap between technical capability and practical business use, which requires disciplined strategy, architecture, and learning from key implementation lessons."

The first wave of GenAI excitement roughly from 2023 through early 2024 was driven by sheer awe. Large Language Models were able to write poetry, generate code, summarize documents and hold surprisingly coherent conversations.

Organizations rushed to adopt AI without clearly defining the problem they wanted to solve or how they would measure success.

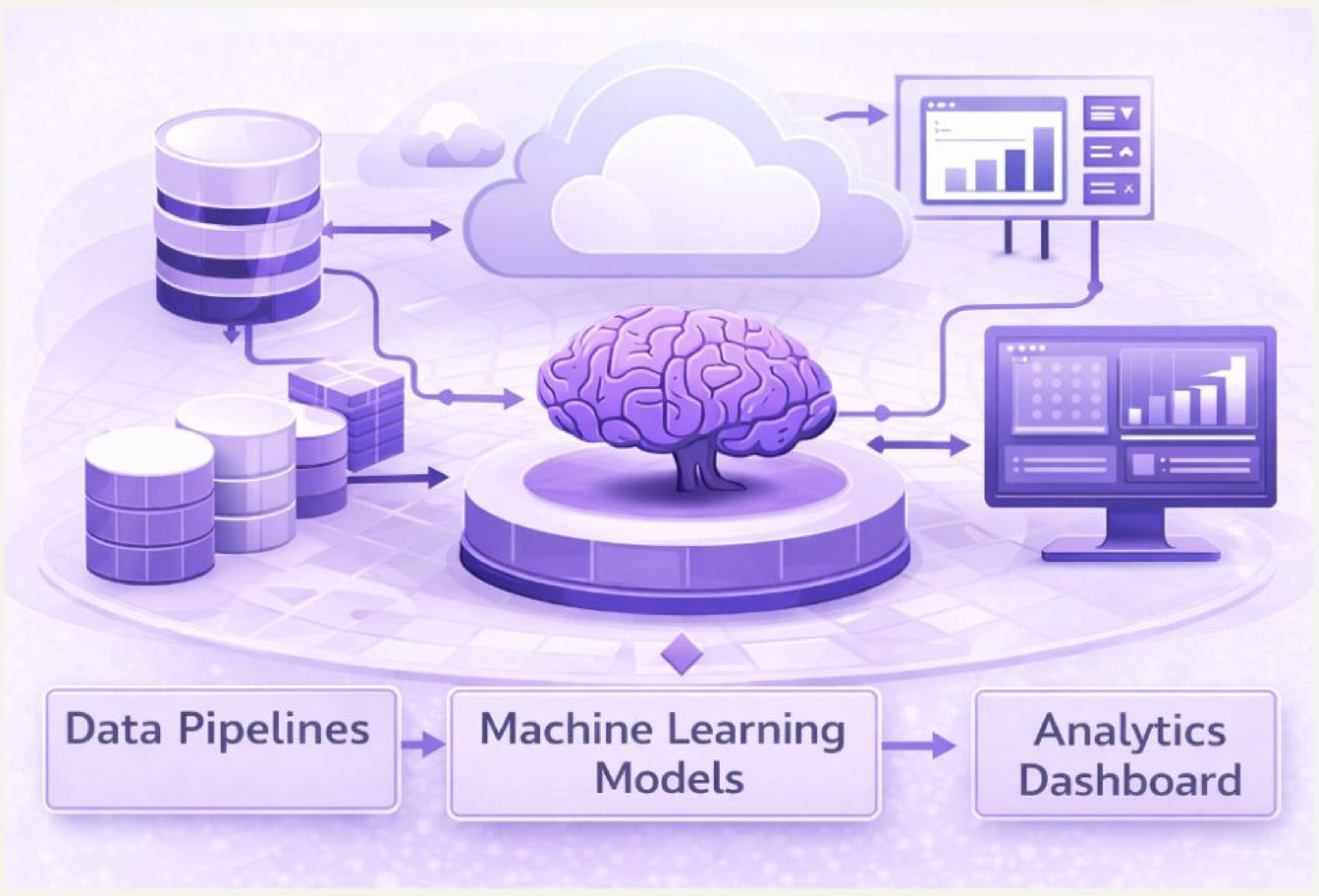
The technology is extraordinary and the models keep getting better at a pace that's genuinely staggering. But the gap between what's technically possible and what's operationally viable in a real business context is very real and bridging it requires discipline.

We've been on both sides of that gap at Coditation. We've built GenAI features that delighted users and delivered immediate ROI and we've also been brought in to rescue projects where the initial approach was fundamentally misaligned with the problem. Honestly, both categories have been equally instructive.

// CODITATION'S PERSPECTIVE

We're in a different phase now but the awe hasn't disappeared completely but it's been tempered by reality. Companies that deployed GenAI early have learned that impressive demos don't automatically turn into production-grade systems. Hallucinations, latency spikes, runaway costs, user adoption challenges and QA headaches have been hurdles to even strong engineering teams.

This playbook is the synthesis of those experiences. A strategic and architectural guide focused on the decisions that matter most and the mistakes that cost the most.

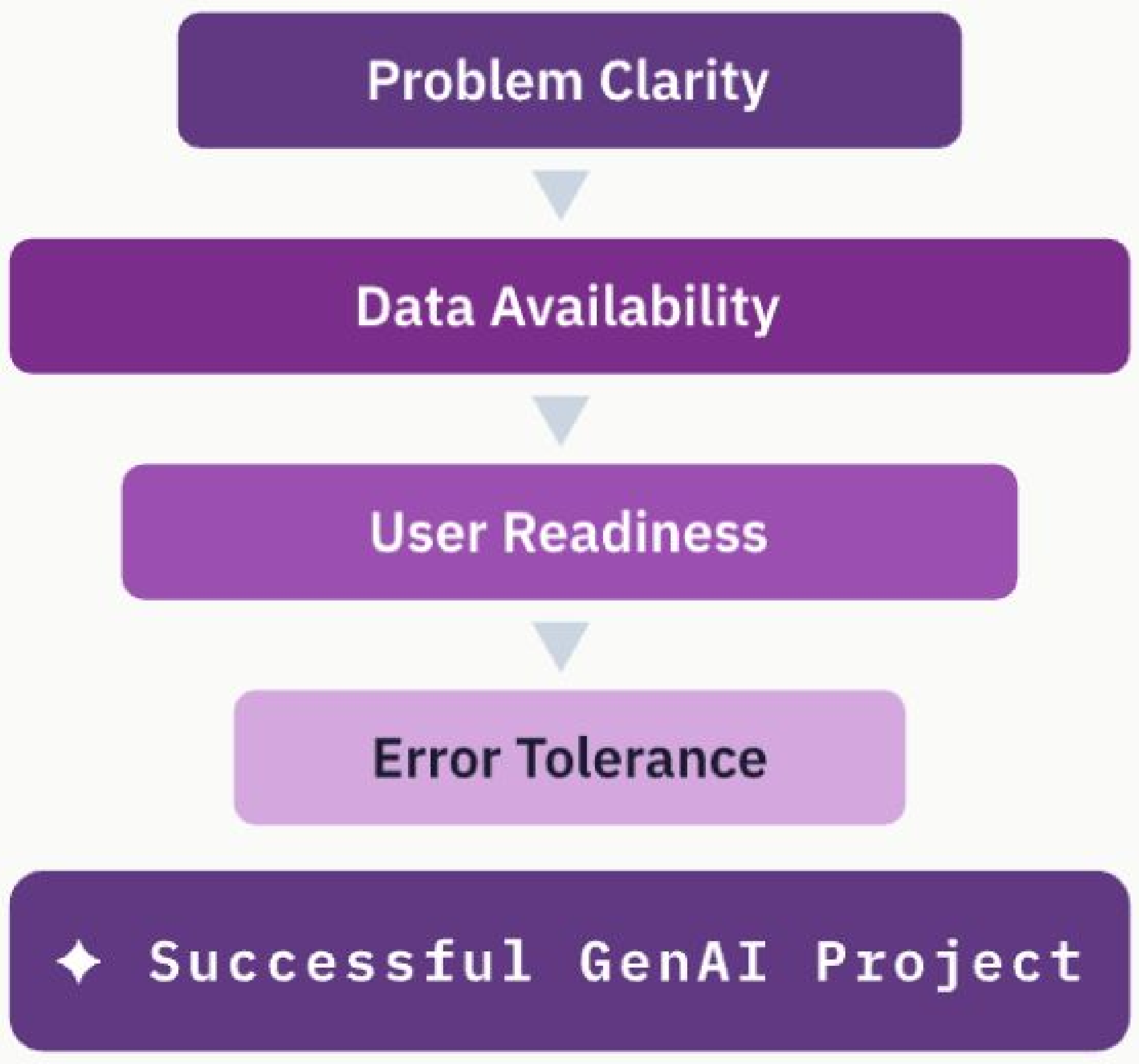


USE-CASE SELECTION: THE SINGLE GREATEST PREDICTOR

// THE CORE INSIGHT

If there's one lesson, I'd want every technology leader to absorb it's this: use-case selection is, by far, the best predictor of whether your project will succeed.

This might sound obvious but it isn't practiced nearly often enough. The temptation in most organizations is to start with the technology and work backward towards the problem. This approach almost always leads to solutions looking for problems. Surely, the resulting projects might be technically interesting but they fail to deliver business value because they were never anchored to a genuine and high impact need.



WHAT MAKES A GOOD GENAI USE CASE?

1. Problem Clarity

Is the problem well-defined? Can you describe success in specific, measurable terms? Vague goals like "make our customer service smarter" are red flags. Specific ones like "reduce ticket resolution time by 30%" are green lights.

2. Error Tolerance

How much does accuracy matter? GenAI thrives where occasional imperfection is acceptable: content drafting, summarization, brainstorming, code scaffolding. It struggles where a single mistake carries serious consequences.

3. Data Availability

GenAI without relevant context is basically a very sophisticated guessing machine. RAG architectures can help but only if the underlying data is accessible, clean and well-structured.

4. User Readiness

A technically perfect GenAI feature that users ignore or don't trust delivers exactly zero value. Understanding user workflows, pain points and attitudes toward AI-assisted tools is essential.

// KEY INSIGHTS — LOGISTICS CLIENT

Mid-size logistics company with 15+ potential use cases. Applied evaluation framework → selected 2 high-scoring use cases → shipped document extraction in 3 months → cut manual processing time by over 60%.

60%
TIME SAVED

A REAL EXAMPLE

Use-Case Selection Done Right

One of our clients, which is a mid-size logistics company, came to us with a broad mandate: "We want to use AI to improve operations." During discovery we identified over fifteen potential use cases spanning from procurement, route optimization, demand forecasting, customer communication to warehouse management.

Instead of trying to tackle everything at once we applied our evaluation framework and selected two use cases: automated extraction and reconciliation of shipping documents (bills of lading, invoices, customs declarations) and intelligent exception handling for delivery anomalies.

The document extraction solution shipped within three months and **cut manual processing time by over 60%.** The client's confidence in GenAI grew and they were far better positioned to tackle the harder problem next.

> "Avoiding these traps requires organizational discipline and honest self-assessment. It takes the courage to say 'not yet' to use cases that aren't ready — even when there's internal pressure to move fast."

FOUR ANTI-PATTERNS TO WATCH FOR

Boil the Ocean
Trying to build an AI-powered everything platform from day one. Scope must be ruthlessly narrow in early phases.

Demo-Driven Development
Picking use cases based on what looks impressive in a presentation rather than what delivers real operational value.

Technology-First Thinking
Starting with a model or framework and trying to shoehorn a business problem into it. Always start with the problem.

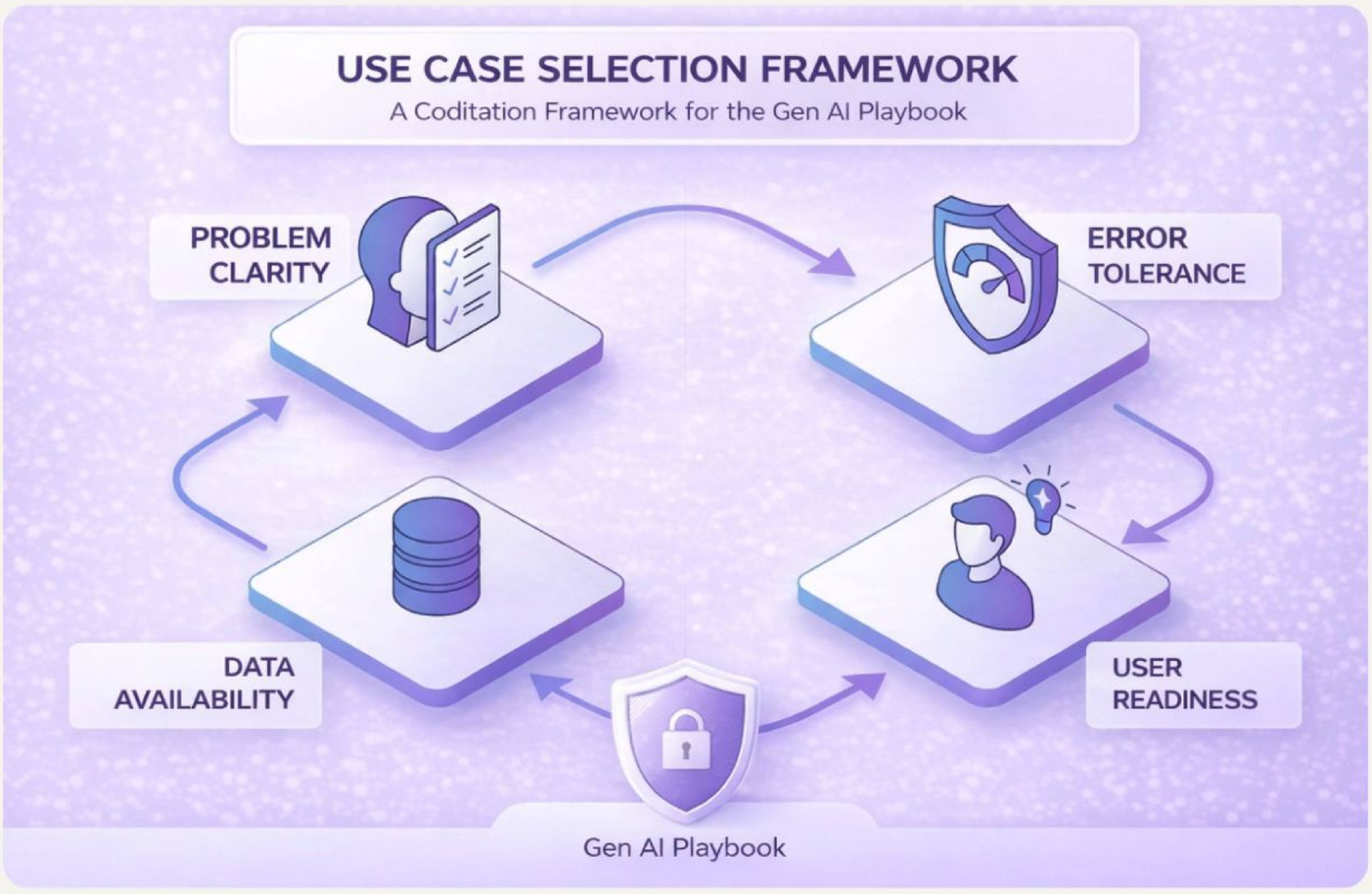
Competitor Copycat
Implementing GenAI features because a competitor announced them, without checking whether the same use case even applies to your situation.

WHAT SCORED HIGH ✓

Document extraction use case: clear problem, moderate error tolerance, abundant historical data, enthusiastic operations team.

WHAT WAS DEFERRED ✗

Exception handling: required real-time decision-making, low error tolerance, fragmented data. Deferred until after first use case was stable and organization had built confidence + infrastructure.

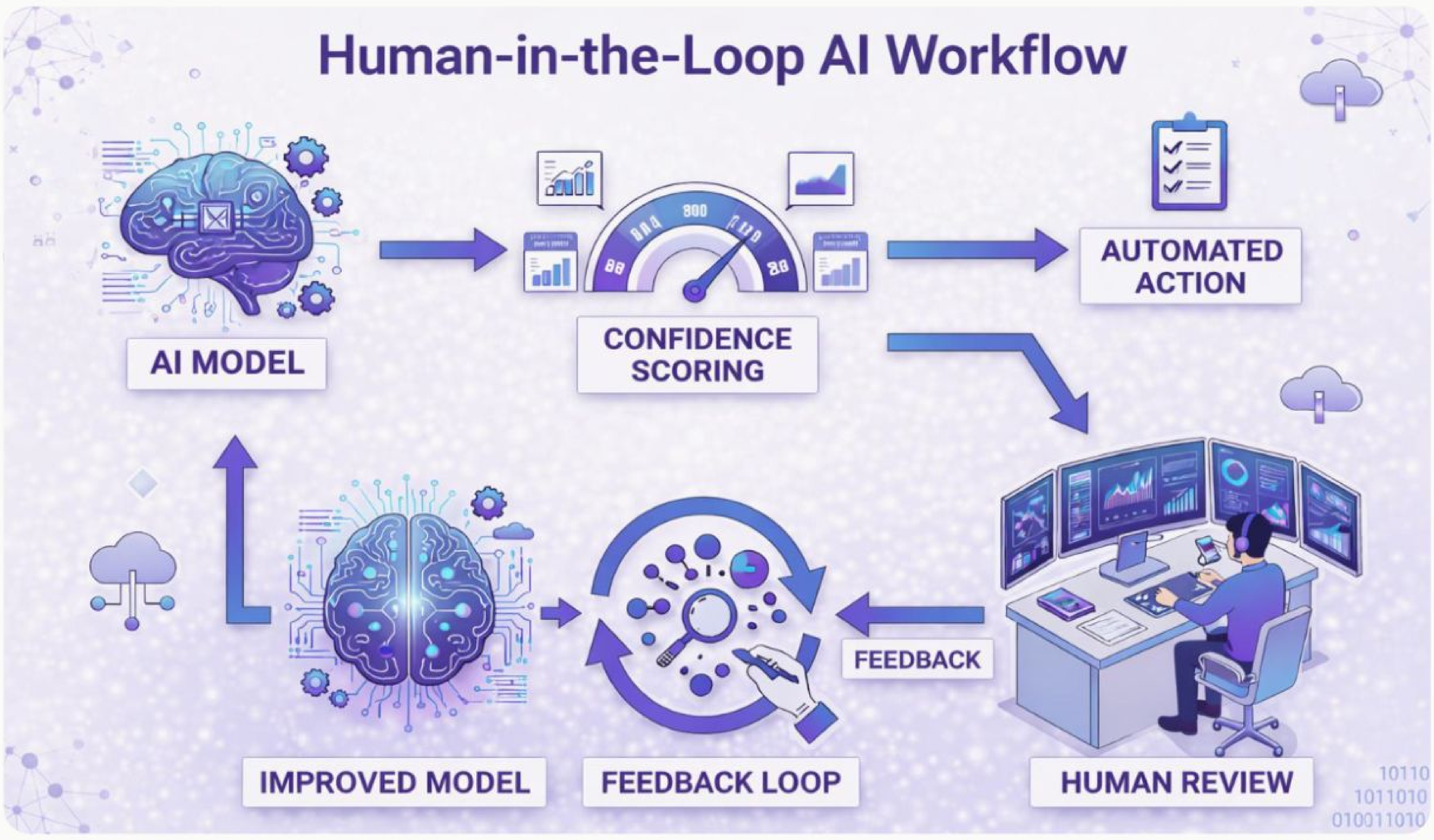


There's a persistent narrative in the GenAI world that we're approaching full autonomy and that AI agents will soon handle entire workflows without any human involvement. The trajectory is definitely promising but the current reality calls for a more measured approach.

Determinism still matters in most enterprise settings where stakeholders need confidence that outcomes are predictable, auditable and controllable. GenAI by its probabilistic nature doesn't inherently provide that. The same prompt can produce different outputs and models hallucinate with remarkable confidence. Edge cases yield surprising and sometimes alarming results.

> "Effective systems design structured human review workflows by using confidence scores, routing only exceptions for review, and capturing human feedback to continuously improve the model."

Human review isn't just advisable in many scenarios but sometimes it's unavoidable. The real question isn't whether to keep humans in the loop but rather how to do it efficiently and that's where UX and workflow design become the difference-makers.



DESIGNING FOR EFFICIENT HUMAN REVIEW

The worst version of human-in-the-loop is what we call the "rubber stamp" model which is the AI generates output and a human is asked to review it with no context, no guidance and no efficient way to give feedback. This approach is slow, frustrating and inevitably degrades. Reviewers either start approving everything without actually looking at it or they reject everything out of caution and neither outcome is useful.

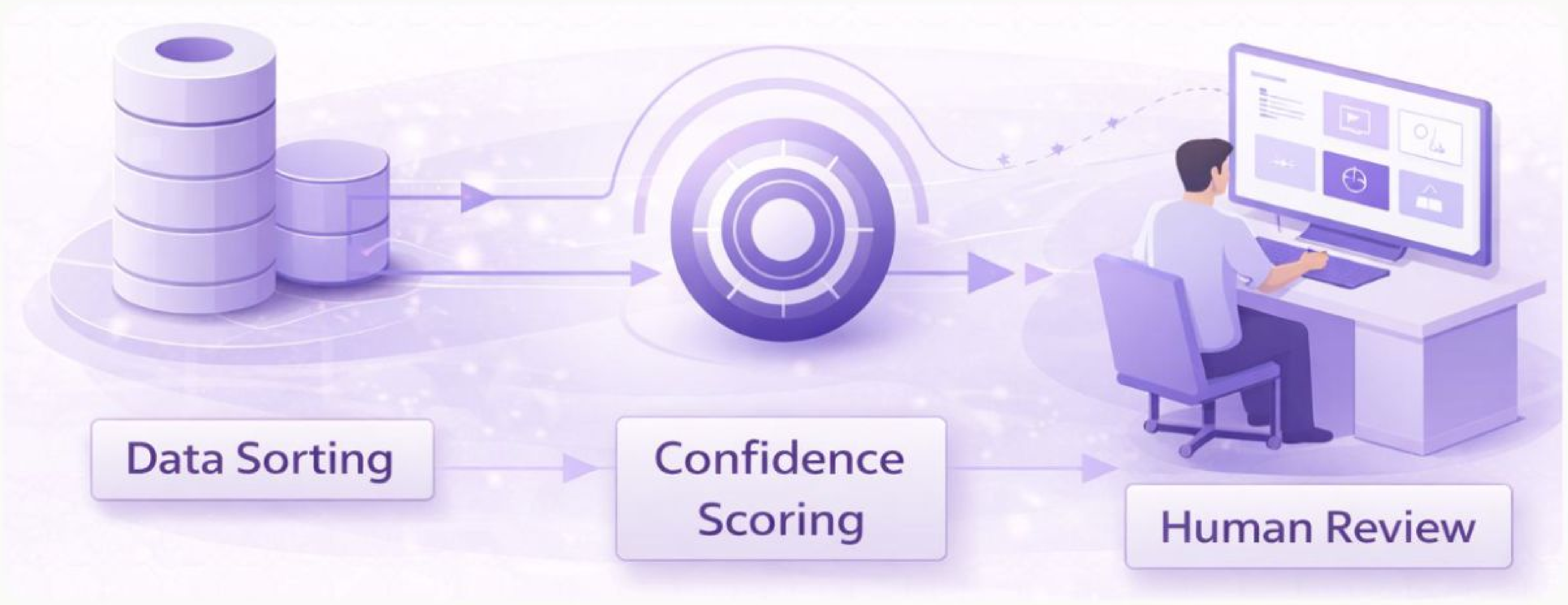
The organizations that win have designed their UX and workflows specifically for efficient and meaningful human review. In practice, that means:

1. **Surface confidence scores** alongside AI outputs so reviewers can focus on the items the model is least sure about and not rubber-stamp everything indiscriminately.
2. **Showing side-by-side comparisons** between AI output and source data enable rapid verification without context switching or hunting for reference materials.
3. **Building exception-based workflows** where only anomalies and edge cases get routed for human review while high-confidence outputs flow through automatically.
4. **Creating feedback loops** that capture human corrections and use them to improve model performance over time, progressively reducing the review burden.

We worked with a healthcare technology company using GenAI to generate clinical documentation summaries from physician-patient encounter notes. Accuracy was critical as errors could affect patient care, billing and regulatory compliance.

The initial approach was straightforward: generate a summary and have the physician review it but the adoption was terrible. Physicians had already pressed for time and found the review process cumbersome. It felt like the AI was adding to work rather than saving it.

> "Human-in-the-loop isn't a limitation to tolerate. When designed well it's a competitive advantage that builds trust, improves quality and drives adoption."



We redesigned the workflow with three changes:

1. **Differential highlighting:** The summary highlighted sections where the model's confidence fell below a defined threshold and drew the physician's eye to precisely the parts that needed scrutiny and high-confidence sections could be scanned quickly.
2. **Structured review interface:** Instead of presenting a wall of free text we broke the summary into discrete components like diagnosis, treatment plan, medications and follow-up instructions that could be individually approved, edited or flagged.
3. **Adaptive thresholds:** As the system collected physician feedback it adjusted confidence thresholds per physician, per specialty and per document type progressively reducing the number of items needing manual review.

The results were striking as **review time per summary dropped by 70%** and the overall accuracy of finalized summaries actually improved because reviewers could direct their cognitive energy where it mattered most.

// KEY INSIGHTS — HEALTHCARE DOCUMENTATION

Redesigned workflow for a healthcare tech company using GenAI for clinical documentation summaries.

Added differential highlighting, structured review interface, and adaptive thresholds per physician.

Result: Review time per summary dropped by **70%** and overall accuracy *improved*.

70%
LESS REVIEW TIME

THE BROADER PRINCIPLE

This example points to something bigger: Gen AI's most impactful applications often aren't about doing something previously impossible. They're about making something previously difficult feel effortless today.

// THE PRINCIPLE

"The real value isn't in AI doing something new — it is in AI *removing friction* from what users already do. When evaluating Gen AI opportunities, always ask: where are our users currently fighting with complex interfaces, steep learning curves or bottlenecks in accessing information? Those are the sweet spots for Gen AI-powered UX improvements that deliver immediate and tangible value."

PATTERNS THAT WORK WELL

→ **Natural language search**	across internal knowledge bases and documentation — removing the need to know the right keyword or folder structure.
→ **Conversational onboarding**	for complex platforms where GenAI walks new users through setup and configuration — eliminating lengthy training programs.
→ **Intelligent form completion**	where GenAI pre-fills fields based on context, history and user intent — cutting manual data entry dramatically.

QUESTIONS
What was our avg delivery time last quarter?
300%
SELF-SERVICE
Traditional BI
Natural Language Query
Instant Insight

ML IS NOT DEAD: THE ENDURING POWER OF TRADITIONAL MODELS

The GenAI excitement has created an unfortunate side effect: people are starting to overlook or flat-out dismiss traditional machine learning. Teams that were building excellent ML solutions a year ago are pivoting entirely to LLM-based architectures — sometimes abandoning approaches that were working perfectly well.

This is a mistake because Machine Learning isn't dead so don't overlook it.

Large Language Models excel at understanding and generating natural language. But there are entire categories of AI applications where traditional approaches remain clearly superior in terms of accuracy, cost, speed and interpretability.

> "Many AI applications — time-series forecasting, computer vision, recommendation systems, tabular data — are still better handled by traditional ML models. LLMs mainly add support features, not replace the core models."

WHERE TRADITIONAL ML STILL WINS

Time-Series Forecasting

Demand prediction, financial modelling, sensor anomaly detection. ARIMA, Prophet, LSTM networks, gradient boosting work extremely well. LLMs can help with interpretation but the core prediction is better handled by purpose-built models.

Computer Vision

Is used in object detection, image classification, quality inspection and medical imaging.
While multimodal LLMs are making impressive progress, production vision systems especially those needing real-time inference, edge deployment or domain-specific accuracy still rely on architectures like YOLO, ResNet, and EfficientNet which typically are fine-tuned on proprietary data.

Recommendation Systems

Includes collaborative filtering, content-based recommendations and hybrid approaches which remain the backbone of personalization across e-commerce, media and advertising. LLMs can add natural language explanations or conversational interfaces on top but the core ranking and scoring still runs on traditional ML.

Tabular Data Problems

For structured data challenges like credit scoring, churn prediction and fraud detection, gradient-boosted trees (XGBoost, LightGBM) are extremely competitive and often outperforming LLM-based approaches while being faster, cheaper and far easier to explain.

"The most effective AI solutions combine traditional ML and GenAI for different roles, as seen in a manufacturing case where an ML model predicted machine failures while GenAI translated those predictions into clear maintenance advice, showing that the key is choosing the right tool for each part of the problem."

A REAL EXAMPLE

ML AND GENAI WORKING TOGETHER

For a manufacturing client, we built a predictive maintenance system that combined traditional ML and GenAI in a complementary way.

The core prediction engine was figuring out which machines were likely to need maintenance in the next 72 hours and ran on a gradientboosted model trained on historical sensor data, maintenance records and environmental factors. It was highly accurate, fast and cheap to run at scale.

// GEN AI COMPONENT

The GenAI component played a completely different role:
"It generated plain-language maintenance advisories for plant operators. When the ML model flagged a machine the GenAI layer pulled relevant context like machine history, similar past incidents, manufacturer recommendations and turned it into a clear and actionable summary that operators could understand without needing to decode raw model outputs."

The ML model excelled at pattern recognition in structured and time-series data and the GenAI layer excelled at synthesizing diverse information into something a human could quickly grasp. Together, they created a solution that was both accurate and accessible and neither piece could have done the other's job well.

// THE STRATEGIC POINT

"The right question is never "Should we use GenAI or ML?" It's "What's the right tool for each part of the problem?" Organizations that maintain strong ML capabilities alongside their GenAI work will have a real advantage. They'll be able to pick the best approach for each use case, avoid the cost and complexity of over-engineering with LLMs and build architectures that play to the strengths of both."

// KEY INSIGHTS — MANUFACTURING PREDICTIVE MAINTENANCE

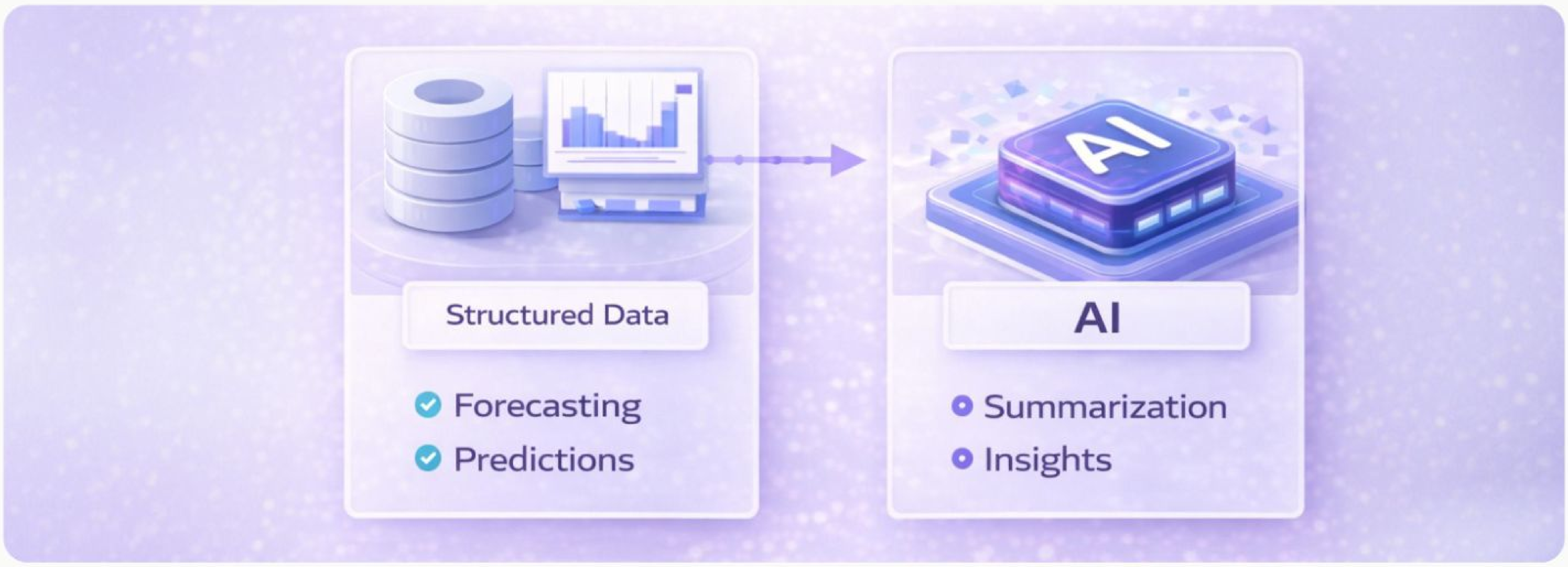
ML COMPONENT

Gradient-boosted model on historical sensor data. Predicts which machines need maintenance in next 72 hours. Highly accurate, fast, cheap at scale.

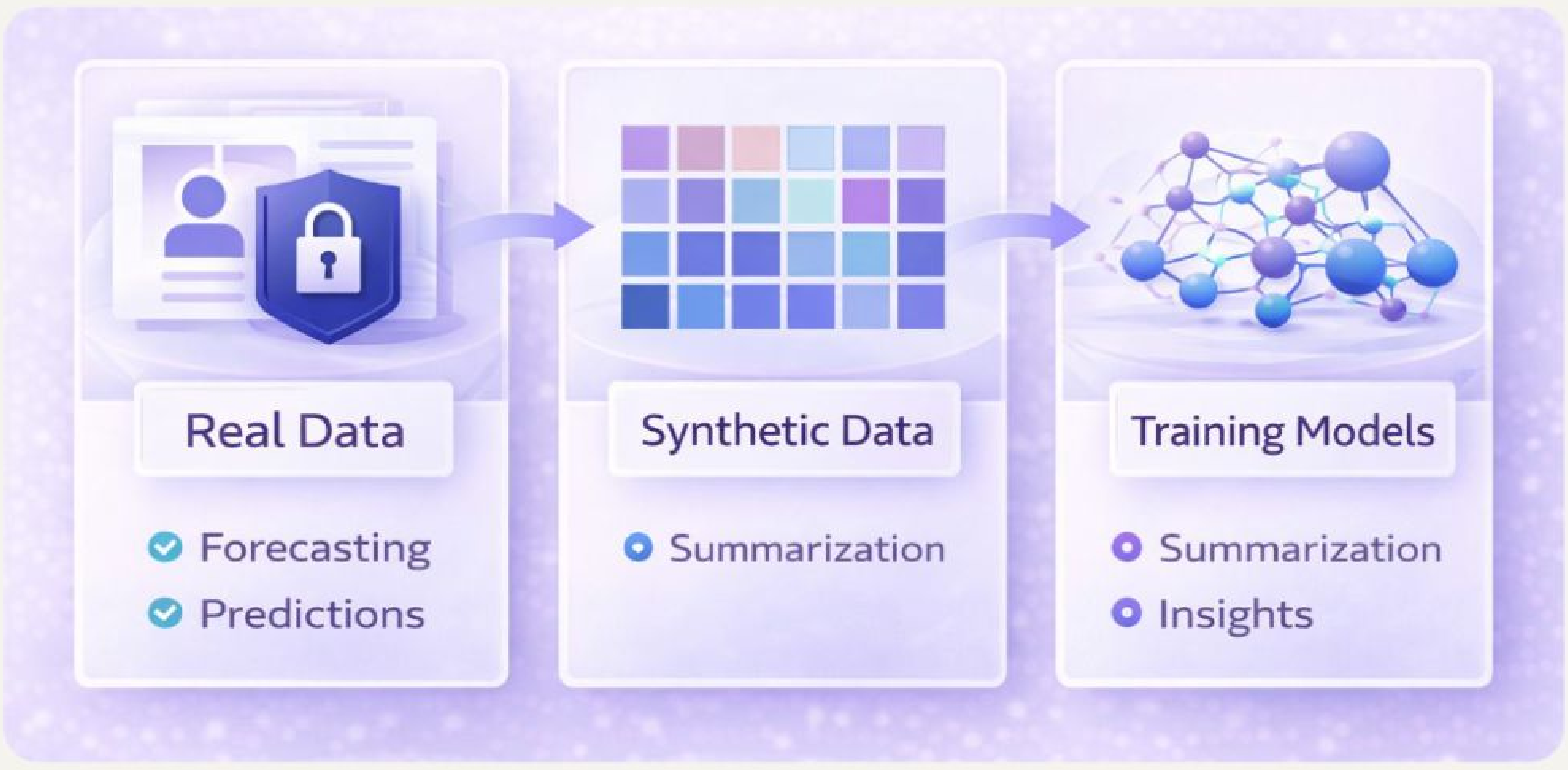
⟷

GENAI COMPONENT

Generates plain-language maintenance advisories for plant operators. Pulls machine history, similar incidents, recommendations. Clear, actionable summaries.



Synthetic data is artificially generated data designed to mimic the statistical properties of real-world data and has become a genuinely powerful tool. It can solve data scarcity problems, enable privacy-compliant model training, boost underrepresented classes and speed up development cycles. In our experience, it's also **a double-edged sword** that requires careful handling.



✓ Where Synthetic Data Shines

Privacy & compliance:

In regulated industries like healthcare and finance access to real data for model training is often locked down by privacy rules (HIPAA, GDPR). Synthetic data that maintains statistical patterns without including personal information can enable development that wouldn't otherwise be possible.

Class imbalance:

Real-world datasets are often highly unbalanced for example fraudulent transactions make up only a small fraction of all transactions. Rare diseases appear in only a small portion of medical records. In these cases synthetic data can augment those minority classes helping models perform better on exactly the cases that matter most.

Rapid prototyping:

When developing a new AI feature, waiting months to gather enough real-world data can stall your momentum. This is where synthetic data offers a practical starting point for development and testing while real data pipelines are still being built.

LLM fine-tuning and evaluation:

Generating synthetic training examples and evaluation sets for fine-tuning has become standard practice. When done carefully it can dramatically cut the cost and time of creating training data.

✗ Where Synthetic Data Can Mislead

Distribution drift:

Synthetic data generators include built-in assumptions about the underlying data distribution. If those assumptions are wrong or if the real-world shifts, then the models trained on synthetic data can fail in subtle and dangerous ways. They look great in testing (against synthetic evaluation data with the same built-in assumptions) but fall apart in production (against messy real-world data).

Reinforcing biases:

If the generation process is based on biased real-world data then synthetic data can amplify those biases rather than correcting them or worse it can create a false sense of security and the teams will think they've addressed the bias problem when they've actually deepened it.

Overfitting to artifacts:

Synthetic data inevitably contains artifacts like the patterns that reflect the generation process itself and not the underlying reality. Models can pick up on these hidden patterns and will be showing impressive results during development but underperforming in real-world deployment.

Model collapse:

When LLMs are trained or fine-tuned on data generated by other LLMs without enough realworld grounding then the models can degrade leading to reduced diversity, amplified errors and flattened performance. This phenomenon is increasing and is called "model collapse, " which is a real and a growing concern.

A REAL EXAMPLE

Enhancing Document AI for Diverse Data Sheets

We worked with a manufacturing client receiving thousands of technical data sheets from different vendors. While these documents looked similar at first glance, every supplier followed a slightly different format. Tables were structured differently, specifications sat in different sections, and the same information often appeared in completely different layouts.

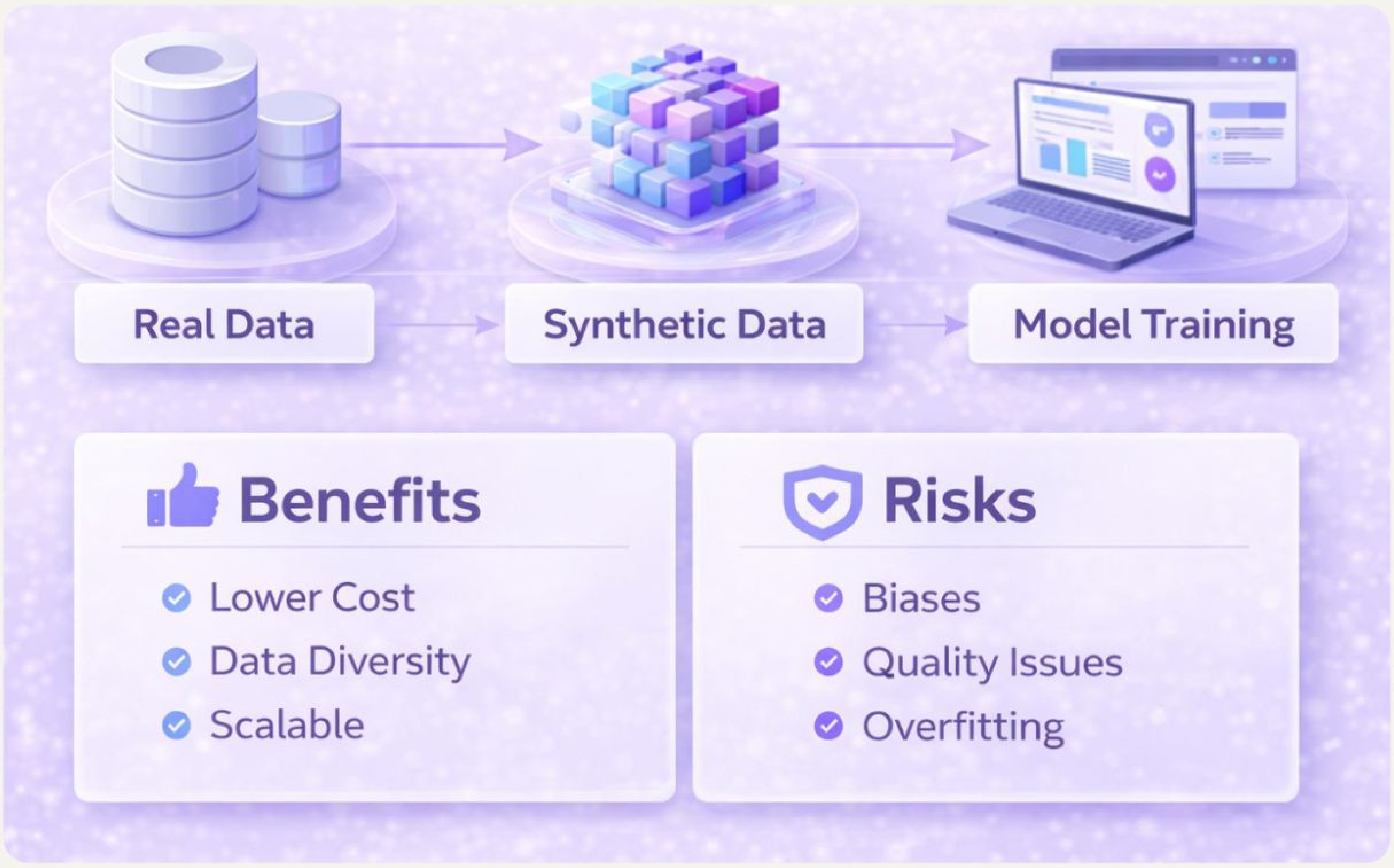
They were using an off-the-shelf Document AI system, but it struggled with unfamiliar formats. Tables broke, values were missed, and manual correction became routine. The problem was straightforward: the model had not seen enough variety. Most of the training data came from a limited set of templates, so it could not handle new layouts.

We addressed this with synthetic data. We created synthetic data sheets that resembled real ones but covered a much wider range of layouts. We changed table structures, shifted sections, varied fonts and spacing, and added imperfections like scan noise and alignment issues. If the model could learn from more variations, it would be less likely to fail on something new.

We did not just generate data and stop there. We continuously tested on real documents, and whenever the model still struggled, we created similar scenarios in the synthetic data to keep learning aligned with real-world challenges. We also varied our generation methods so the model did not become dependent on any single layout style.

Over time, the system became more stable. It started understanding document structure instead of memorizing formats, handled new layouts with fewer errors, and the need for manual corrections dropped significantly.

> “The takeaway for us was that in document parsing, the problem is not just about having more data. It is about having the right kind of variety, and synthetic data makes it possible to create that in a controlled way.”



The mainstream GenAI narrative tends to spotlight the biggest, most powerful models — GPT-4, Claude Opus, Gemini Ultra. These models are highly versatile but also costly to run, slower to respond and often far more powerful than most enterprise use cases require.

Think of it this way: you don't need a Formula 1 car to run your errands. Solution to this is the fine-tuned small language models deployed within well-designed agentic workflows which frequently deliver a **better cost-benefit ratio**.

THE AGENTIC ARCHITECTURE

The real power of small models shows up when you embed them within agentic workflows architectures where multiple specialized components collaborate to handle complex tasks.

Instead of asking one massive model to do everything (understand the request, retrieve data, perform analysis, generate output, format results). An agentic architecture divides the workflow into clear steps with each step handled by the most suitable component such as a fine-tuned model, a piece of deterministic code, a database query and an API call.

// THE BENEFITS ARE:

- **Reliability:** Each piece can be tested, validated and improved independently.
- **Observability:** You can trace exactly where something went wrong in a pipeline instead of debugging a monolithic black box
- **Efficiency:** You only pay (in compute and latency) for the model capability you need at each step.
- **Upgradability:** Individual components can be swapped or upgraded without rebuilding everything.

THE CASE FOR SMALLER MODELS

"Small" is a relative term, usually referring to models with 1 to 13 billion parameters, compared to the hundreds of billions found in frontier models. When fine-tuned on domain specific data for a focused task these smaller models can match or even beat much larger general-purpose models on that particular task.

WHY SMALLER MODELS WIN IN PIPELINES

Deployment Flexibility

Small models operate on-premises, at the edge, or in environments with strict data residency requirements where external API calls aren't possible.

Predictability

Fine-tuned models perform more reliably within their trained domain reducing the need for complex prompting and guardrails.

Cost

Smaller models respond faster (direct UX impact) and cost 10–100× less at inference. The difference compounds dramatically at production scale.

Speed

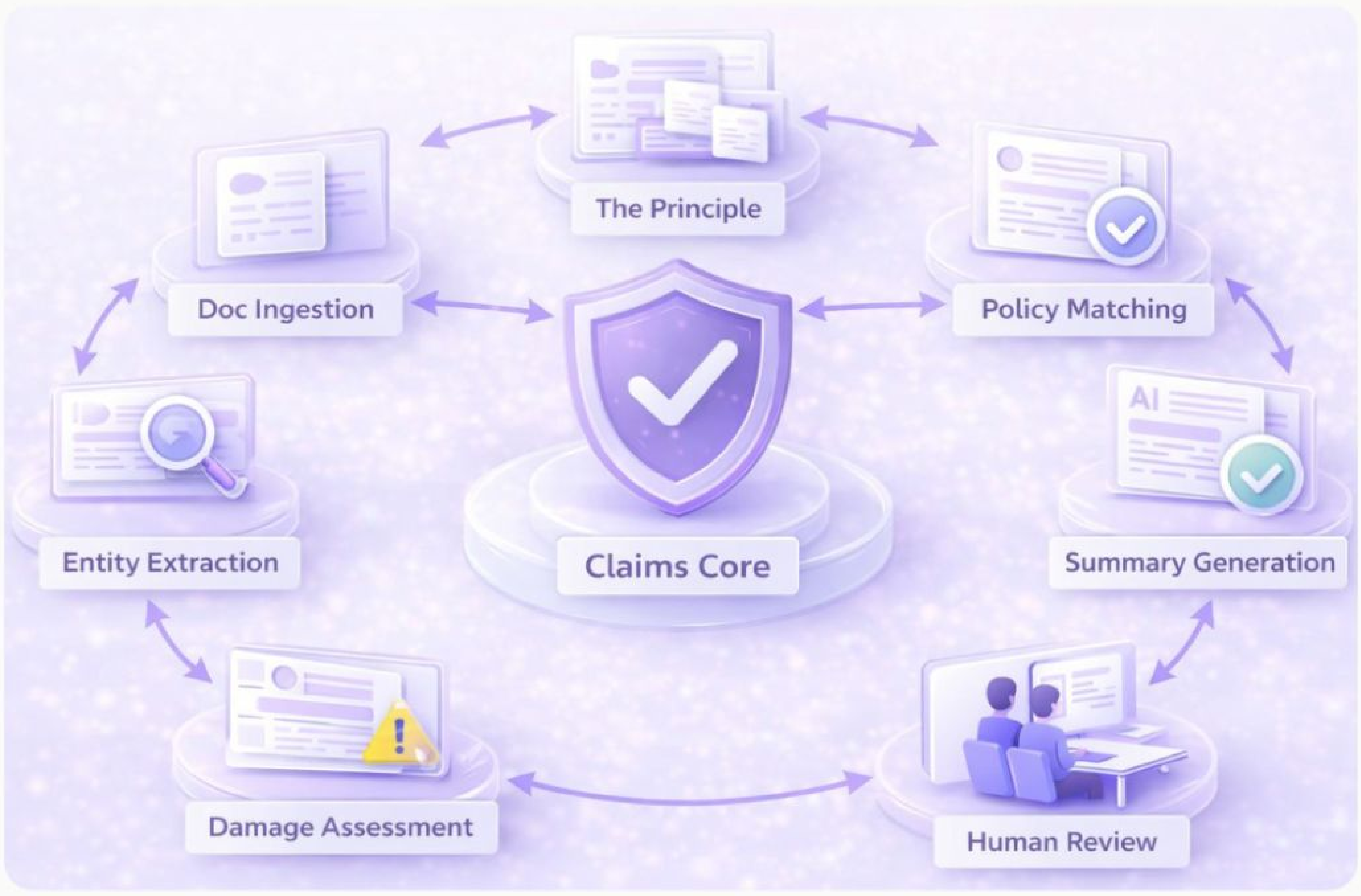
Smaller models respond faster which directly impacts user experience.

THE AGENTIC ARCHITECTURE — INSURANCE CLAIMS EXAMPLE

For an insurance technology client, we built an automated claims processing pipeline using this approach:

01	**Document ingestion & classification** Sorted incoming documents (claim forms, medical reports, police reports, photos) and routed them to appropriate processing streams.
02	**Entity extraction** Extracted structured data such as names, dates, policy numbers, amounts and incident descriptions from each document type.
03	**Policy matching & coverage verification (deterministic code)** Checked extracted data against policy databases to verify coverage, deductibles and limits.
04	**Damage assessment** Evaluated damage from photos and assessed fraud risk based on historical patterns.
05	**Summary generation & recommendation** Generated a clear, human-readable claims summary with a recommendation (approve, flag for review, or deny) followed with supporting rationale.
06	**Human review interface** Flagged claims were presented to adjusters through a dedicated interface including all supporting documents and AI-generated analysis.
07	**The Principle** Use the smallest, most specialized model that can accomplish the task, and let solid code and workflow logic manage the orchestration.

"In an insurance claims automation system, different tasks like document classification, data extraction, policy verification, damage assessment and summary generation were handled through a step-by-step workflow using specialized models and deterministic code, demonstrating that using smaller, task-specific models within structured workflows is more efficient and practical than relying on a single large model."



> "The most successful GenAI implementations we've built don't use chat as the primary interface. Instead, we weave AI capabilities directly into the user's existing workflow."

Let me say this plainly: **pure text-based chat interfaces rarely move the needle in enterprise settings.**

That might seem surprising given that ChatGPT's chat interface is arguably the most successful product launch in tech history. But the enterprise context is fundamentally different from the consumer one and the novelty of conversational AI has largely worn off.

WHY CHAT ALONE FALLS SHORT

Ambiguity
Natural language is inherently imprecise. Users often don't know how to phrase what they want, leading to rounds of clarification that a structured interface would eliminate.

Discoverability
In a chat interface there's no menu, no button, no visual hint about what the system can actually do. Users repeat the same 2–3 familiar queries and overlook everything else.

Efficiency
For repetitive tasks, typing is almost always slower than clicking. "Show me overdue invoices sorted by amount" takes longer to type than clicking two filters and a sort button.

Context Switching
Adding a chat window to an existing application and expecting users to switch from visual to text interaction creates cognitive strain. It disrupts the workflow instead of improving it.

WHAT WORKS INSTEAD

→ **Inline suggestions**	and auto-completions that appear contextually as users work
→ **AI-powered buttons**	"Summarize this document," "Generate a response," "Classify these items" — delivering results within the existing interface
→ **Smart defaults**	and pre-population — Gen AI fills in forms and drafts content based on context, reducing effort without requiring explicit input
→ **Multimodal interfaces**	combining text, visuals, structured data and interactive elements to present AI outputs in the most effective format

A REAL EXAMPLE

EMBEDDED AI VS. CHAT

We built **two versions of an AI-powered contract review tool for a legal tech client.**

Version one featured a chat interface where lawyers could ask questions about contracts like "What are the termination clauses?" or "Identify any non-standard indemnification language. "

Version two was developed based on user feedback from the first version and embedded AI directly into the document review workflow. As lawyers scrolled through a contract the system automatically highlighted clauses of interest also displayed risk ratings in a sidebar, surfaced comparable language from precedent agreements and offered suggested edits for non-standard provisions. All of this happened without the lawyer having to type a single word.

THE LESSON WE LEARNT:

"Meet users where they are, and embed intelligence into existing workflows instead of asking people to adopt a new way of interacting."

// LEGAL TECH CASE's CONCLUSION

Adoption of version two was **more than four times higher.** Lawyers described the embedded experience as a natural extension of how they already worked. In contrast, the chat interface felt like a separate tool they had to remember to open and learn to use.

WHY DETERMINISTIC ORCHESTRATION MATTERS

This principle is in many ways the architectural spine of everything else in this playbook which is: wherever possible, let your code and logic own orchestration and have the LLM do specific and bounded tasks.

The urge to let LLMs take the lead is understandable. Modern models are remarkably capable reasoners and often make the right call. But "often" isn't "always," and in a production system the difference between 95% reliability and 100% reliability can be the difference between a useful tool and a liability.

Predictability

Code runs the same way every time. Deterministic orchestration means you can predict system behavior, reproduce bugs, and ensure business rules are applied consistently.

Auditability

Regulated industries require clear audit trails. A statement like "The LLM decided to route this claim to the fraud team" won't be acceptable but "The fraud score exceeded the configured threshold of 0.85 and triggering routing to the fraud review queue" will be acceptable.

Testability

Deterministic code can be unit tested, integration tested and regression tested using standard engineering practices. In contrast, comprehensively testing decision making by large language models is inherently more challenging.

Cost & Speed

Each decision delegated to a large language model requires an API call or inference pass resulting in additional cost and latency. Decisions that can be handled through conventional programming techniques such as conditional logic, threshold checks, data validation and routing rules should be implemented directly in code.

A NOTE ON LLM-GENERATED CODE :

An important nuance is that the orchestration code itself can be generated by a large language model. Increasingly, organizations use large language models to draft business logic, validation rules and workflow configurations. However, once generated this code is reviewed, tested and deployed as deterministic logic. It does not rely on the large language model during inference time but instead, it executes as conventional code.

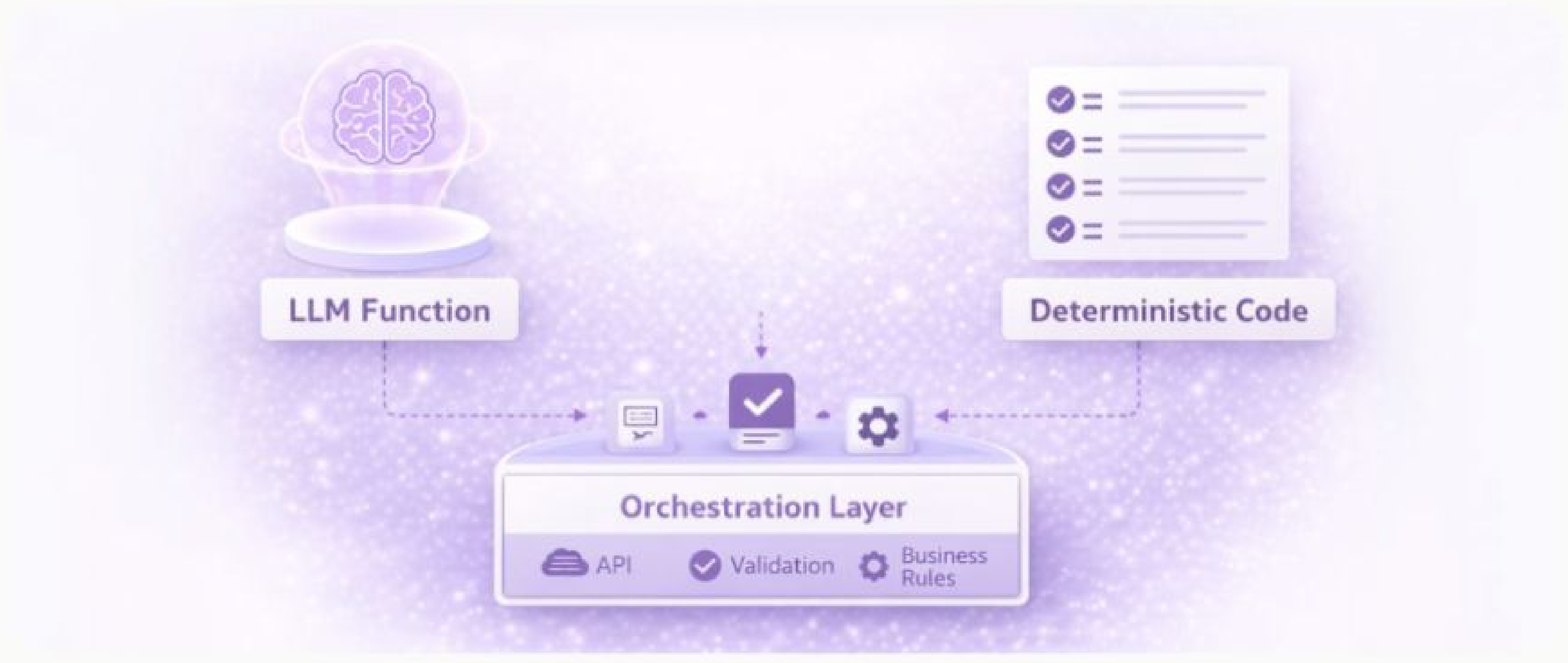
This distinction is subtle but critically important. Leveraging large language models to accelerate development serves as a productivity enhancement. In contrast, relying on large language models to make runtime decisions represents an architectural choice with significant implications for reliability, predictability and operational control.

IN PRACTICE:

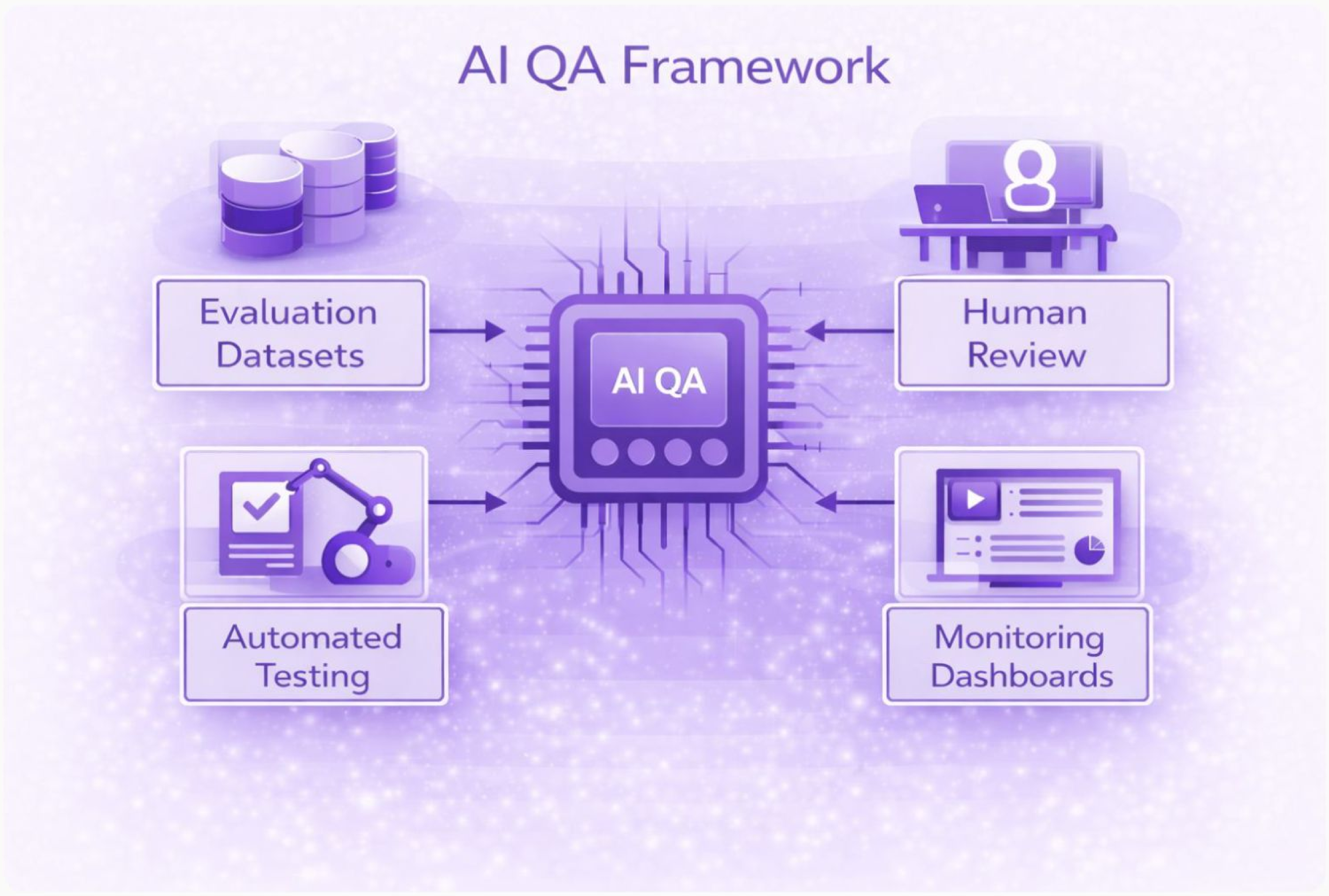
In nearly every production grade generative AI system we have built, across industries, use cases and scales this pattern consistently applies. The orchestration layer remains deterministic while the large language model operates as a specialized component within a broader system architecture. The result is a solution that combines intelligence with reliability and creativity with control.

We encourage every team building generative AI applications to establish a clear distinction between what is determined by conventional code and what is decided by the large language model. Wherever possible, decision making should be handled by the code to ensure greater reliability, predictability and control.

"While LLMs can generate orchestration code during development, the code should be reviewed, tested, and run as deterministic logic in production, ensuring that runtime decisions are handled by reliable code while LLMs act as specialized components within the system."



"Quality assurance for AI is not simply traditional QA under a different label. It demands new tools, revised processes, specialized skills and sustained organizational commitment."



Quality assurance for AI systems differs fundamentally from traditional software QA — and it is an area in which many organizations remain significantly underinvested.

Traditional software quality assurance is grounded in determinism. For example: Given input X the system produces output Y. Test outcomes are binary resulting in either a pass or a fail. Continuous integration and continuous deployment pipelines execute regression test suites to ensure stability. Organizations can rely on decades of mature tools, frameworks and established methodologies to support this process.

Artificial intelligence applications challenge this traditional model. Their outputs are probabilistic rather than strictly deterministic. In many cases, correctness exists on a spectrum rather than as a binary outcome. The same input may produce different outputs across model versions, prompt variations or even separate executions of the same configuration. The range of potential edge cases is effectively limitless.

Moreover, the most serious failures are not system crashes but responses that are subtly incorrect while appearing entirely plausible making them significantly harder to detect and mitigate.

This means you need to pay serious attention to how you QA your AI applications and agents. Invest meaningful time in developing a comprehensive strategy and clear operational guidelines. This should not be treated as a minor consideration or an afterthought; it is a core workstream that warrants dedicated resources, structured planning and sustained leadership attention.

ELEMENTS OF A ROBUST AI QA STRATEGY

→ **Evaluation datasets**	Build and maintain high quality evaluation datasets that accurately represent real production traffic. Version these datasets and update them regularly to reflect evolving use cases and risk profiles. Ensure they include meaningful edge cases and adversarial examples. Evaluation data should be governed and managed with the same level of rigor as applied to training data.
→ **Multi-dimensional scoring**	Avoid relying on a single accuracy metric. Instead, evaluate outputs across multiple dimensions including factual accuracy, completeness, coherence, tone, safety, format compliance and task specific criteria. Develop clear evaluation rubrics that enable consistent and objective assessment by both human reviewers and automated systems.
→ **Automated evaluation pipelines**	Implement systems that automatically run evaluations against established datasets whenever there is a model change, prompt update or system modification. Where appropriate leverage a large language model as a judge approach to assess outputs; however, ensure that these evaluators are themselves validated against human assessments to maintain reliability and alignment.
→ **Regular human evaluation**	Establish regular review cadences in which domain experts assess representative samples of real production outputs. Automate the sampling and presentation processes to ensure efficiency and minimize operational burden while maintaining thorough oversight.
→ **Production monitoring**	Track the quality of artificial intelligence outputs in real time by monitoring user feedback signals such as edits, rejections and regeneration requests alongside automated quality scores and downstream business performance metrics. Establish alerting mechanisms to detect and address quality declines early before they escalate into broader systemic issues.
→ **Prompt regression testing**	Treat prompts with the same discipline applied to software code. Version them systematically and subject them to structured testing. Whenever a prompt is modified make sure to evaluate it against established evaluation datasets and compare the results with the previous version. Even minor prompt changes can produce cascading and non obvious effects across the system.
→ **Red teaming**	Periodically subject your artificial intelligence systems to stress testing with unexpected and unconventional inputs. Evaluate how the system responds to out of domain topics, malformed requests and attempts to probe or manipulate the system prompt. Identifying vulnerabilities proactively is essential as it is far better to uncover weaknesses internally than to have them exposed by users in production.

A REAL EXAMPLE

QA OPERATIONS FOR A GENAI PRODUCT

For a SaaS client that embedded GenAI features across their platform we helped stand up a dedicated AI QA function. This was not added as an afterthought following launch, rather it was established as a parallel workstream from the very beginning.

KEY ELEMENTS

- A curated evaluation dataset comprising more than 3,000 examples across 15 task categories, maintained by a dedicated team and refreshed on a quarterly basis.
- An **automated evaluation pipeline** running nightly, scoring outputs across 7 dimensions using heuristic checks, embedding-based similarity metrics and LLM-based judging.
- **Weekly human review sessions** in which product managers, domain experts and engineers examine a stratified sample of 50 production outputs to identify quality patterns and flag systemic issues.
- A structured **prompt change** management process requiring evaluation results to be reviewed and formally approved before any prompt modification is deployed and effectively applying code review discipline to prompts.
- A **production quality dashboard** which tracks performance metrics over time and correlates them with model and prompt changes providing early detection of potential degradation.

This investment delivered significant returns. The client experienced a measurable reduction in quality related customer complaints, faster resolution of issues and substantially greater confidence when releasing new artificial intelligence features.

// SAAS CLIENT QA PROGRAM

3,000+	15	7
EVAL EXAMPLES	TASK CATEGORIES	QUALITY DIMENSIONS

Nightly automated pipeline + weekly human review sessions. Measurable reduction in quality-related complaints. Greater confidence releasing new AI features.

LATENCY ISN'T JUST A TECHNICAL ISSUE — IT'S A UX DESIGN CHALLENGE THAT NEEDS TO BE ADDRESSED FROM DAY ONE

One final factor that teams consistently underestimate until it escalates into an urgent operational issue is latency.

Large language model inference is not instantaneous. Depending on the model, prompt complexity, output length and underlying infrastructure the response times can range from a few hundred milliseconds to several tens of seconds. In agentic workflows that chain multiple model calls and end to end latency can extend to several minutes.

At the same time, users have been conditioned by decades of web and mobile application experiences to expect near instant responses. Search results typically appear within milliseconds and a button click generally produces a visible change in less than a second. When an artificial intelligence feature takes five to ten seconds even if the output quality is high users often experience friction and frustration which leads to a gradual erosion of trust.

6 STRATEGIES FOR MANAGING LATENCY

1. **Progressive disclosure** — Present results progressively as they become available. Stream text responses incrementally and delivering content token by token to reduce perceived wait time. For multi-step workflows one can display intermediate outputs as each stage completes to provide users with timely feedback and maintain engagement throughout the process.

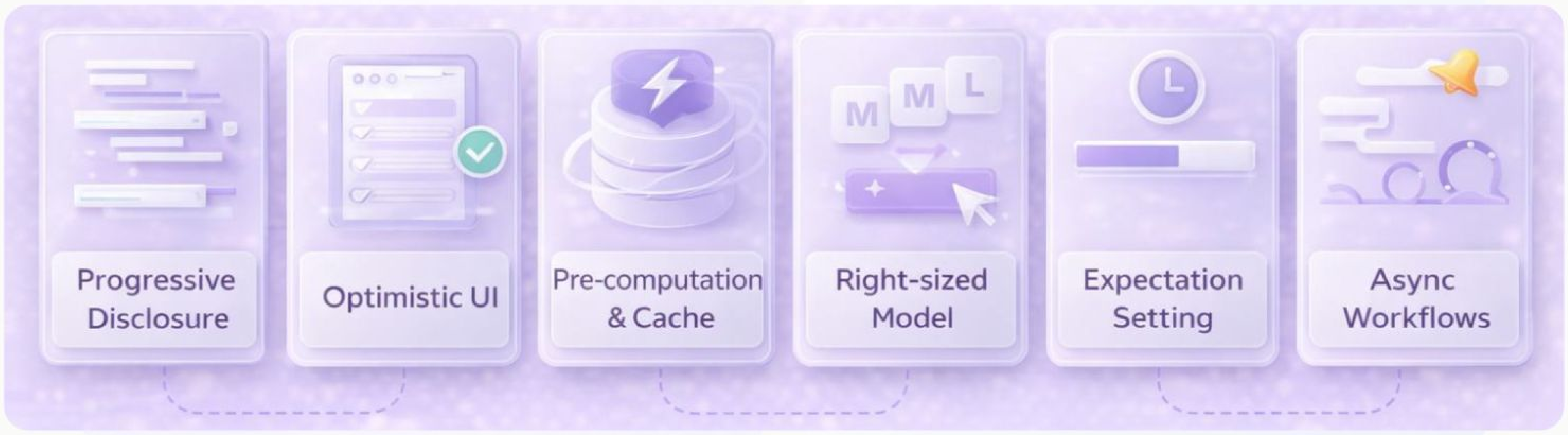
2. **Optimistic UI patterns** — Design workflows that allow users to continue their primary tasks while artificial intelligence processing occurs in the background. Present results as they become available ensuring that core user interactions remain uninterrupted and the overall experience feels responsive and seamless.

3. **Precomputation and caching** — For common queries or highly predictable tasks one must precompute and cache responses in advance. When a small proportion of queries generates the majority of traffic the effective caching strategies can significantly reduce perceived latency for most user interactions and improve overall system responsiveness.

4. **Right-sized model selection** — As discussed in Section 8, smaller models generally offer faster inference times. Select the smallest model that satisfies the quality requirements for each specific task. Additionally, explore optimization techniques such as quantization, batching and infrastructure tuning to further reduce latency and improve overall system performance.

5. **Expectation setting** — When latency cannot be avoided you can communicate it transparently. Use progress indicators, estimated wait times and clear status messages such as "Analyzing 47 documents" to inform users about what is happening. Proactive communication reduces perceived wait time which manages expectations and minimizes frustration.

6. **Asynchronous workflows** — For tasks that require substantial processing time one must enable users to initiate the process and continue with other activities and give a notification once the results are available. This approach is particularly effective for batch processing, report generation and complex analytical workflows.



A REAL EXAMPLE

LATENCY-AWARE DESIGN IN E-COMMERCE

We developed a product description generation system for an e-commerce client. The initial version generated comprehensive descriptions based on product attributes and images however each description required approximately eight to twelve seconds to produce.

For editing a single product, this delay was tolerable. However, for bulk operations in which merchandisers needed to generate hundreds of descriptions the latency was entirely unacceptable.

E-COMMERCE CASE OUTCOMES

80%	2–3s
LATENCY REDUCTION	TIERED MODEL SPEED

WE TACKLED THIS SCENARIO THREE WAYS:

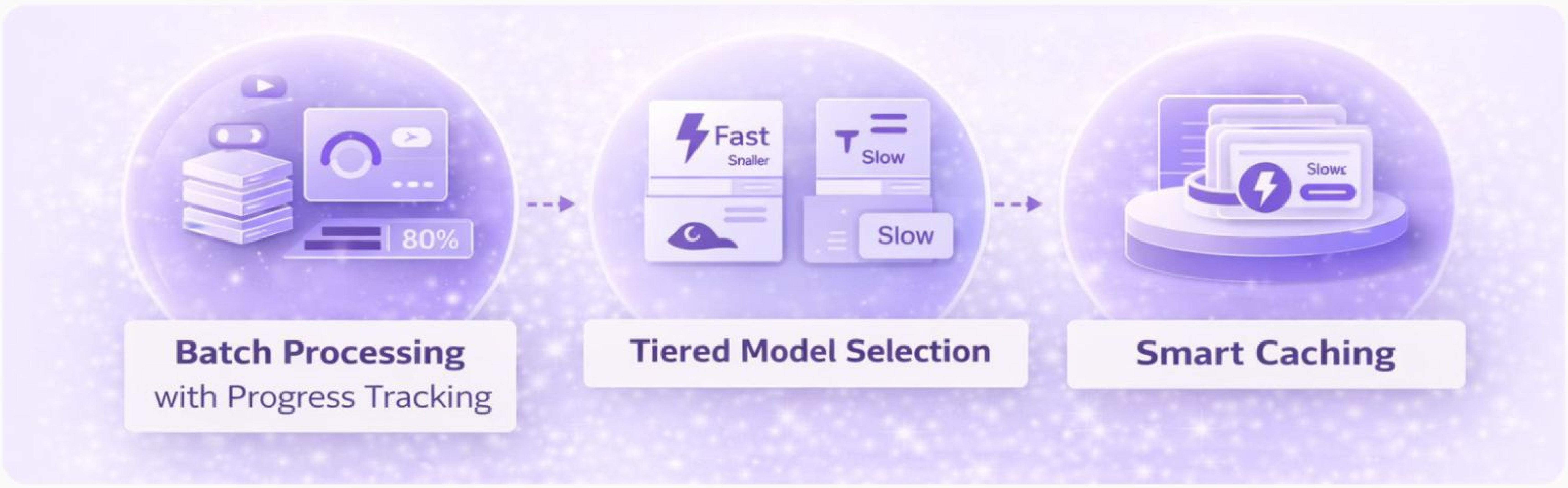
1. **Batch processing with progress tracking** — Bulk processing tasks were executed in the background. Merchandisers were provided with a live dashboard displaying completed, in progress and queued items enabling them to begin reviewing finished descriptions while the remaining items were still being generated.
2. **Tiered model selection** — Initial drafts were generated using a smaller and faster model that produced results in approximately two to three seconds per description. While the outputs were high quality, they were not always fully refined. Merchandisers had the option to selectively route individual descriptions to a larger model for additional polishing when necessary.
3. **Smart caching** — For products with attributes similar to those previously processed the system retrieved and adapted cached descriptions rather than generating new content from scratch which helped in reducing latency for comparable products by nearly 80 %.

// THE CONCLUSION

The result was a system that felt responsive and practical to use even though the underlying content generation process remained inherently time consuming. Following these optimizations the user satisfaction scores increased significantly

THE LESSON WE LEARNT:

Don't treat latency as an afterthought or an infrastructure problem to solve later. Design your UX around the reality of AI inference times from the beginning.



The **ten principles** outlined in this playbook are not theoretical concepts. They are derived from practical experience across dozens of generative AI projects spanning industries, scales, and levels of complexity. They reflect what we have seen succeed at Coditation Systems as well as the approaches that have proven ineffective in real world implementations.

#	PRINCIPLE	CORE MESSAGE
01	Use-case selection	Pick the right problem before you pick the right model.
02	Human-in-the-loop	Design efficient review workflows; don't fight the need for human oversight.
03	New UX	Use GenAI to make existing functionality more accessible, not just bolt on new features.
04	ML is not dead	Use the right tool for each task; traditional ML still excels in many domains.
05	Synthetic data	Powerful but risky; validate relentlessly against real-world data.
06	Small models + agentic	Specialized, fine-tuned models in well-orchestrated pipelines beat monolithic approaches.
07	Beyond chat	Embed AI into workflows; don't lean on chat as the primary interface.
08	Limit LLM decisions	Let deterministic code handle orchestration; use LLMs for specific, bounded tasks.
09	QA for AI	Invest in evaluation infrastructure, processes, and dedicated quality operations.
10	Latency-aware design	Design your UX to accommodate and mitigate AI inference times from day one.

The common thread across all of these principles is **pragmatism.** Generative artificial intelligence is a transformative technology but meaningful transformation does not occur simply by deploying models. It occurs by solving real problems for real people in ways that are reliable, efficient and sustainable over time.

The organizations that will lead in the artificial intelligence era are not necessarily those with the most sophisticated models or the largest research teams. They are the ones that identify the right problems, design effective user experiences, build robust architectures and maintain the discipline required to execute consistently over time.

CODITATION

// GEN AI WITH CODITATION

Gen AI creates value when it's applied with clarity, intent and a strong connection to real business needs. The next step isn't to do everything. It's to start with the right problem.

Coditation works with teams that want to move beyond experimentation and build Gen AI solutions that are practical, scalable and grounded in outcomes. From identifying high-impact use cases to building and refining production-ready systems, the focus is on making AI work in real environments.

The team works across the full AI and machine learning lifecycle: identifying the right problems, designing and building solutions, deploying them, and improving them over time. The approach stays grounded in practical outcomes, ensuring the work connects back to real business value.
If you're thinking about where Gen AI can make a difference in your organization, this is a good place to start.

WHAT WE DO

AI/ML Strategy & Consulting
Use case identification, feasibility assessment, roadmap development and technology selection.

GenAI Application Development
LLM-powered applications, RAG architectures, agentic workflows, model fine tuning and prompt engineering.

Data Engineering
Pipeline design, platform modernization, data quality management and data governance.

Traditional ML
Predictive modelling, computer vision, NLP, time series forecasting and recommendation systems.

MLOps & AI Operations
Model deployment, monitoring, evaluation infrastructure and production operations.

// GENAI SOLUTIONS

Ready to build production-grade and practical Generative AI solutions?
From use-case selection to agentic pipelines & LLM fine-tuning — we help you build GenAI that delivers real ROI.

www.coditation.com

Talk to Us