MERIT

From Pilot to Production: The Engineering Leader's Guide to AI Operationalisation, Platform Reliability, and Enterprise Integration

A guide to the scaling constraints, governance requirements, and integration decisions that determine whether AI systems perform reliably in production

For: AI Engineering Leaders | CTOs and VPs of Engineering | Platform Engineering Leaders
Enterprise Architects | Heads of AI and Data Science

www.meritdata-tech.com

What's inside

- The operationalisation gap: why pilot success and production reliability require different engineering decisions -- and the root causes that account for most failures between them
- The four technical failure modes at the production gate -- and the architectural decisions that resolve monolithic design, unstructured data pipelines, retrieval quality, and governance gaps
- Cognitive workflow automation: the orchestration, confidence gating, exception handling, and human-in-the-loop decisions that determine whether LLM pipelines are production-grade or demo-grade
- Unstructured data at scale: architecture and engineering decisions for NER, entity resolution, and document intelligence pipelines built for production volumes and governed environments
- Production analytics and multimodal processing: the platform and operational decisions behind sentiment analysis, trend forecasting, and Vision Language Model deployment at enterprise scale
- Platform lifecycle management and GenAIOps: decisions around modular AI architecture, model monitoring, inference cost optimisation, component versioning, and integration into enterprise systems
- Two production deployments -- renewable energy and NHS -- evaluated against architecture, operational metrics, and governance
- A readiness diagnostic grouped by architecture, governance, operations, and integration -- for AI engineering leaders evaluating their current programme against the failure modes in this guide

A note before you read this

This guide is written for AI leaders who are past the question of whether to invest in AI. You have already answered that. The question you are now trying to answer is harder: why is the gap between what AI can do in a controlled environment and what it actually does in production so wide -- and what needs to change architecturally to close it.

We will name the specific technical failure modes, the architectural patterns that address them, and the decisions that determine whether an AI programme compounds in value or plateaus at pilot stage. This is written from direct production experience -- not from analyst frameworks or vendor positioning.

CHAPTER 1

MERIT

Everyone Is Running Pilots. Almost Nobody Is Running AI in Production.

The gap between AI investment and AI value is one of the most widely documented and least honestly discussed problems in enterprise technology. Organisations have spent significantly on AI over the past three years. The models have improved dramatically. The tooling has matured. The talent has been hired. And yet the production deployment rate remains stubbornly low.

The problem is not the AI. It is everything around it -- and specifically, the operational realities that a controlled pilot environment never surfaces. In production, latency constraints are real: a model that runs acceptably in a notebook does not necessarily meet the sub-second response requirements of a live operational workflow. Uptime requirements are real: an AI system embedded in a production process needs the same reliability engineering as any other production system -- SLAs, failover, degraded-mode operation, and incident response. Monitoring complexity is real: a model that performs well at launch will drift as the data distribution shifts, and without continuous monitoring of input distributions, output quality, and confidence scores, that drift is invisible until it causes a downstream failure. Inference cost management is real: a model that is economically viable at pilot scale -- processing hundreds of documents per day in a proof of concept -- may be unviable at production scale processing hundreds of thousands, without deliberate architectural decisions around batching, model selection, and tiered processing. These are the operational realities that determine whether AI reaches production. They are engineering problems, not research problems.

The numbers behind the gap

$19.9T in cumulative global economic impact AI is projected to contribute through 2030 (IDC, 2024). The scale of the ambition is not in question -- the gap is between projected value & actual production deployment rates.

80%+ of AI projects fail to reach meaningful production deployment -- twice the failure rate of non-AI technology projects (RAND Corporation, 2024). For engineering leaders, this means the production gap is not an edge case. It is the norm, and the causes are architectural, not algorithmic.

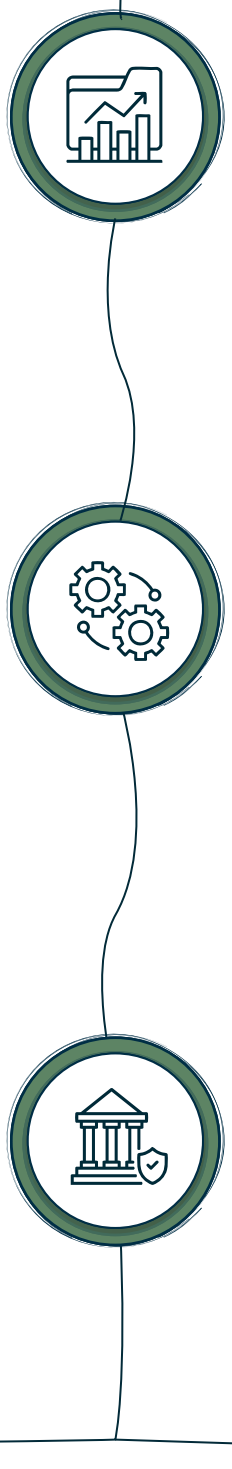
Two-thirds of organisations investing in AI remain stuck in pilot mode, unable to scale across the enterprise (McKinsey, 2025). At that scale, the constraint is not model capability -- it is the absence of the platform engineering, integration, and governance infrastructure that production requires.

48% of AI projects make it past pilot on average, and it takes eight months to go from prototype to production (Gartner, 2024). For CTOs and platform engineering leaders, this means the majority of AI investment is being consumed by the operationalisation gap rather than by the capabilities the business actually needs.

The pattern plays out the same way in organisation after organisation. A proof of concept is built, typically in an isolated environment with clean data and a small team. It performs well. Leadership sees the demo. Investment is approved. Then the team attempts to move it into production -- where the data is messier, the systems are older, the latency requirements are tighter, and the governance requirements are real - and it stalls.

> The gap between AI pilot and AI production is not a gap in model quality. It is a gap in architecture, data infrastructure,integration and governance. These are engineering problems, not research problems.

Why pilots succeed and production fails -- the root causes



The first is data. A pilot is built on a curated dataset assembled for the purpose. Production runs on real data: inconsistent, incomplete, arriving from multiple sources at different latencies, governed by compliance requirements that were not a factor during the proof of concept. The AI that performed well in the pilot encounters data it was not designed for, and degrades. The pipeline feeding it was not built for volume, schema variability, or the quality enforcement that production requires.

The second is integration. A pilot is a standalone system that proves a capability. Production requires that capability to be embedded in an existing workflow -- a CRM, an ERP, an operational data platform, a customer-facing application -- with the latency, reliability, and API contract that integration demands. In most enterprise environments, this means integrating with legacy systems that were not designed to expose clean interfaces, carry unpredictable response times, and require significant middleware work before an AI component can consume them reliably. Most AI systems are not built to be integrated. They are built to be demonstrated. The integration work is treated as a subsequent phase rather than a design constraint, and it is where most production timelines collapse.

The third is governance. In a regulated industry, an AI system that cannot explain its outputs, trace its training data lineage, or demonstrate compliance with data handling requirements cannot be deployed. Governance is not a checkbox added at the end of an AI programme. It is an architectural constraint that has to be designed in from the start -- and almost never is in a pilot context.

The fourth is operational ownership and cost management.

A pilot has a project team. A production AI system needs an operating model: defined ownership of the pipeline, the models, the monitoring, and the incident response process. In most organisations, this ownership is ambiguous at the point of production handover -- the data science team built it, the engineering team was not involved early enough to own it, and the business unit does not have the capability to run it. The system stalls not because it does not work, but because nobody has clear accountability for keeping it working. Alongside ownership, inference cost at production scale is frequently underestimated. A model that is economically viable processing hundreds of documents in a pilot becomes unviable processing hundreds of thousands in production without deliberate architectural decisions around batching, model selection, caching, and tiered processing. Cost management is an engineering problem that needs to be designed for, not discovered after the first production invoice.

MERIT

The Four Technical Failure Modes That Stop AI at the Production Gate

Beyond the strategic gap, there are four specific technical failure modes that account for the majority of AI programmes that stall between pilot and production. The symptoms are recognisable before the root cause is diagnosed: duplicated pipelines built independently by different teams solving the same problem, fragmented experimentation environments that cannot be promoted to production without a rebuild, unmanaged model proliferation where multiple versions of the same capability exist across the estate with no clear ownership or retirement process, and integration work that grows in complexity with each new use case because the underlying architecture was never designed for reuse. These are not signs of poor engineering -- they are signs of an AI programme that has scaled pilot thinking rather than establishing production architecture. Identifying which failure modes apply to your current stack is the first step toward understanding what your operationalisation programme actually needs to address.

Failure mode 01 - Rigid, monolithic AI architecture

Most AI systems built at pilot stage are built as monoliths: a single model, a single pipeline, a single deployment. This is the right approach for a proof of concept -- it is fast, it is simple, and it is sufficient to demonstrate the capability. It is the wrong approach for production, where different use cases require different model types, different latency profiles, and different update frequencies.

The practical indicators of monolithic AI design are specific and recognisable. Components cannot be versioned independently -- updating the entity extraction model requires redeploying the entire system, which means every update carries the risk of the whole platform. Small changes require full redeployments, which creates pressure to batch changes together, which increases deployment risk and slows the rate at which the platform can improve. Environments are coupled -- the development, staging, and production environments share configuration or infrastructure in ways that make promoting a change from one to the next unreliable and manual. Scaling one capability means scaling everything, including components that do not need more resource, which inflates infrastructure cost and operational complexity simultaneously. And when a component fails, the blast radius is the entire system rather than a contained, independently recoverable service.

The production architecture that resolves this is modular and composable: individual AI capabilities -- classification, extraction, summarisation, search, generation -- deployed as independently scalable services with well-defined interfaces between them. Containerisation via Docker and orchestration via Kubernetes are the standard infrastructure pattern for this -- each component runs in its own container with its own resource allocation, scaling policy, and deployment lifecycle. A component that needs to be updated, scaled, or rolled back is handled independently without affecting the rest of the system. API governance -- versioned, documented interfaces with explicit contracts between components -- is what makes this composability durable over time. Without it, services accumulate implicit dependencies that reintroduce the coupling the modular architecture was designed to eliminate.

A new use case is assembled from existing components rather than built from scratch. This is the architectural decision that determines whether an AI platform compounds in value over time or becomes a collection of disconnected, unmaintainable models.

Failure mode 02 - No path from unstructured data to structured output

The data that AI systems are most valuable at processing -- documents, emails, PDFs, images, web content, call transcripts -- is exactly the data that most production systems handle worst. Unstructured and semi-structured data requires a pipeline that can ingest it reliably, extract the relevant information at the right granularity, normalise it into a structured form that downstream systems can consume, and do all of this at the volume and latency that production demands.

The ingestion layer is where most implementations underinvest. Production unstructured data arrives in inconsistent formats -- PDFs with different layouts, images of varying quality, HTML with inconsistent structure, audio transcripts with speaker diarisation errors. A pipeline that handles one format well frequently breaks on the next. Format-specific preprocessing, schema validation at ingestion, and dead letter handling for documents that fail processing are foundational requirements that pilot implementations typically skip.

Multimodal data compounds the challenge. A pipeline processing both text and images -- extracting product attributes from descriptions and photographs simultaneously, for example -- requires separate processing paths for each modality, a merging layer that reconciles outputs from both, and a schema that can represent structured output from heterogeneous inputs consistently. The schema design is as important as the model selection: if the output schema is not defined before the pipeline is built, it will be retrofitted after, and retrofitting schema changes into a live pipeline is expensive.

Metadata management and schema evolution are production concerns that pilots never encounter. In production, the schema of extracted output evolves as new entity types are added, new document formats are processed, and new downstream consumers impose new requirements. Without a governed schema registry and a defined approach to backward-compatible evolution, schema changes break downstream consumers silently. Lineage tracking -- knowing which source documents produced which extracted records, through which pipeline version -- is the governance requirement that makes the output of unstructured data pipelines auditable and correctable. Without it, when extraction quality degrades, there is no systematic way to identify which documents were affected or retrace what happened.

Named Entity Recognition is typically the first extraction component and the one where production failures are most common. A NER system trained on a curated dataset will encounter entity types it has not seen, text formats it was not trained on, and edge cases that require context to resolve. Production NER requires fine-tuned transformer models for high-frequency entity types, LLM-based inference for edge case resolution, and a feedback loop that captures production failures and feeds them back into model improvement. Without all three, accuracy degrades as the production data distribution drifts from the training distribution.

Failure mode 03 - Retrieval that does not actually retrieve the right thing

Retrieval Augmented Generation has become the standard pattern for building AI systems that need to answer questions based on a knowledge base rather than relying solely on a model's parametric knowledge. The concept is straightforward: retrieve relevant context from a document store and provide it to the generation model along with the query. The production implementation is considerably more difficult.

Retrieval quality determines the quality of the entire system. Poor chunking strategy -- splitting documents at arbitrary character limits rather than at semantic boundaries -- produces chunks that contain partial context, making relevant retrieval structurally impossible regardless of the embedding model. Weak embedding models produce similar vectors for semantically different content, returning plausible-looking but wrong chunks. The absence of reranking as a second-pass filter means the generation model receives the highest-similarity chunks rather than the most relevant ones. The generation model then produces a confident, well-written response based on irrelevant context. This is the failure mode that makes RAG systems untrustworthy in production -- they fail quietly, without signalling that the retrieval step went wrong.

How retrieval failures appear in production metrics is important to understand because they are rarely obvious. User-facing symptoms include responses that are factually wrong but structurally coherent, answers that do not address the actual question, and inconsistent responses to similar queries. In monitoring dashboards, retrieval failures show up as declining user acceptance rates, increasing escalation rates to human review, and response latency anomalies where the generation step completes quickly because it is working with insufficient context. None of these directly identify retrieval as the root cause without explicit retrieval instrumentation.

Retrieval evaluation frameworks address this by measuring retrieval quality independently from generation quality. Metrics including context precision -- the proportion of retrieved chunks that are actually relevant -- and context recall -- the proportion of relevant chunks that were successfully retrieved -- need to be tracked as first-class pipeline metrics, not inferred from end-to-end output quality. Tools like RAGAS provide standardised evaluation frameworks for this. Without separate retrieval evaluation, a RAG system that is degrading at the retrieval layer is indistinguishable from one that is degrading at the generation layer until the failure is severe enough to surface in user feedback.

Embedding drift is a production concern that most RAG implementations do not monitor until it causes a visible degradation. As the document corpus evolves -- new documents added, terminology shifts, domain-specific language changes -- the embedding space the retrieval system was built on drifts from the current distribution. Monitoring the statistical distribution of query embeddings and document embeddings over time, and triggering re-embedding and index rebuilding when drift exceeds a defined threshold, is the operational practice that prevents silent retrieval degradation over time.

Agentic workflows extend RAG into something more powerful but introduce additional failure modes around loop termination, tool reliability, and result validation. Building agentic systems that behave predictably in production requires explicit constraints on the action space, deterministic fallback paths when tool calls fail, and human-in-the-loop checkpoints for decisions above a defined confidence threshold.

Failure mode 04 - Governance and auditability as an afterthought

In regulated industries -- healthcare, financial services, energy, insurance -- an AI system that cannot trace its outputs back to the inputs and transformations that produced them cannot be deployed. The governance requirement is not a compliance checkbox that can be added at the end of an AI programme. It is an architectural constraint that determines how the system is designed, how data flows through it, and how outputs are logged and auditable.

The failure mode is not malice -- it is sequence. Teams build the AI capability first and plan to add governance later. Later never comes, because retrofitting traceability and auditability into a system that was not designed for it is as difficult as rebuilding it.

The governance architecture needs four operational components defined before the system is built. A model registry tracks which model version produced which output -- without it, when a model is updated or rolled back, there is no reliable way to identify which historical outputs were produced by which version, which is an audit requirement in most regulated contexts. Lineage tooling traces the path from source data through transformation and ingestion to the inputs a model received at inference time -- OpenLineage-compatible instrumentation embedded in the pipeline from the outset is significantly less expensive than reconstructing lineage after the fact. A policy enforcement layer applies data access, PII handling, and output filtering rules programmatically at the pipeline level rather than relying on application-level controls that break when data moves between systems. Audit workflows define how outputs are sampled, reviewed, and challenged -- including escalation paths for outputs that fall below confidence thresholds or that are flagged by downstream consumers.

Regulatory and standards alignment shapes these requirements concretely. The EU AI Act requires high-risk AI systems to maintain logs enabling post-deployment monitoring and to provide explanations of outputs to affected individuals. GDPR Article 22 governs automated decision-making and requires meaningful human review mechanisms for consequential decisions. ISO 42001 -- the AI management system standard -- provides a framework for documented governance processes that auditors increasingly expect to see. In financial services, SR 11-7 guidance from the Federal Reserve on model risk management requires documented model inventories, validation processes, and ongoing performance monitoring. These are not abstract requirements -- they define the specific artefacts, logs, and processes the governance architecture needs to produce.

MERIT

Cognitive Workflow Automation: Moving LLMs from Demos to Live Operational Pipelines

Cognitive workflow automation -- using LLMs to handle high-volume, judgement-dependent tasks at operational scale -- delivers measurable ROI across a wider range of enterprise workflows than most AI programmes initially target. Document classification, entity extraction, and intelligent routing are the most common entry points. But the same pipeline architecture applies to purchase order validation against supplier contracts, anomaly flagging in supply chain event streams, incident triage in IT operations pipelines, and clinical coding from patient notes. The task structure is consistent across all of them: high volume, rule-bounded but context-dependent, currently absorbing significant manual effort.

The challenge is not demonstrating that an LLM can do these tasks. Every AI team has demonstrated that. The challenge is building a pipeline that does them reliably, at scale, within the latency requirements of an operational workflow, with the orchestration mechanisms, exception handling, and human-in-the-loop processes that production demands.

What production cognitive automation actually requires

Document classification in production is not a single model call. It is a pipeline: document ingestion handling PDF, DOCX, HTML, image-based documents and mixed formats; pre-processing to extract clean text and structure; a primary classification step using a fine-tuned transformer model optimised for the specific taxonomy; a confidence threshold gate that routes low-confidence classifications to a secondary LLM-based reasoning step; and an output validation layer that checks structural consistency before the result is passed to the downstream system.

Orchestration is what holds this pipeline together in production. Each stage needs a defined execution contract -- input schema, output schema, latency budget, and failure behaviour. A pipeline orchestrator manages state between steps, enforces timeouts, and triggers fallback paths when a stage breaches its latency budget or returns an unexpected output. Without orchestration, pipeline failures cascade silently -- a slow upstream step causes a downstream timeout which produces a null output which reaches the downstream system without any signal that something went wrong.

Exception handling needs to be explicit at every stage. When the ingestion layer encounters an unsupported format, it routes to a dead letter queue with a structured error record rather than a silent drop. When the classification model returns a malformed output, the validation layer catches it and triggers a retry or escalation. When a downstream system is unavailable, the pipeline suspends rather than losing the processed output. Each failure path needs to be designed, tested, and monitored before the pipeline goes to production.

The confidence threshold gate is the component most teams skip in a pilot and cannot live without in production. A model that classifies with 95% accuracy on average produces a 5% error rate at enterprise scale -- thousands of misclassified documents or incorrectly routed transactions per day. In a document workflow, low-confidence classifications are routed to a secondary model or human review before reaching the downstream system. In a supply chain workflow, a purchase order validation model that flags low-confidence contract matches for human review prevents incorrect approvals from propagating into procurement systems -- where the cost of a downstream error is significantly higher than the cost of a review. The confidence gate is not a performance optimisation. It is a correctness control.

Human-in-the-loop processes need to be designed as first-class pipeline components. This means a defined review interface presenting the flagged case with the model's reasoning and confidence score, an SLA for review completion built into the workflow's latency contract, a feedback capture mechanism recording the reviewer's decision and reason for any correction, and a process for feeding reviewed cases back into periodic model retraining. The review queue is a designed operating state that maintains output quality at the confidence boundary and continuously improves the model over time.

Entity extraction and routing follow the same pattern. The extraction model identifies entities and their relationships. A normalisation layer maps extracted entities to canonical forms -- resolving variant representations to a single authoritative reference. A routing engine uses the extracted entities and classification to determine the next step in the workflow. Each component needs its own error handling, logging, and monitoring. A cognitive automation pipeline is not one AI system. It is five or six, composed together, each with its own operational contract.

The organisations that have successfully deployed cognitive automation at scale are not the ones with the best models. They are the ones that treated the pipeline around the model with the same engineering rigour as the model itself

MERIT

Unstructured Data at Scale: NER, Document Intelligence and Extraction in Production

The volume of unstructured data in most enterprise environments is growing faster than the capacity to process it manually. The information that matters most for business decisions -- regulatory filings, clinical notes, supplier contracts, product descriptions, customer communications -- is increasingly locked in formats that structured data systems cannot process. NLP-based extraction is the capability that unlocks it. Building it to work reliably in production is the hard part.

Named Entity Recognition in production: Operational architecture beyond the model

NER is one of the foundational capabilities in applied AI, and one where the gap between research performance and production performance is widest. A fine-tuned BERT or RoBERTa model trained on a labelled dataset can achieve high accuracy on the test set from the same distribution. In production, the distribution shifts continuously: new entity types emerge, source formats change, domain-specific terminology evolves. A static NER model degrades over time without an active operational loop.

The production NER architecture combines three layers. A fine-tuned transformer model handles high-frequency, well-defined entity types where training data is abundant and patterns are stable. An LLM-based inference layer handles edge cases and novel entity types the fine-tuned model has not seen. A continuous learning loop captures production annotations and feeds them back into periodic retraining.

The operational concerns that determine whether this architecture holds up in production go beyond model accuracy. Retraining frequency needs to be defined against measured distribution drift -- monthly retraining is common for stable domains, weekly for rapidly evolving ones. Annotation workflows need to be designed to capture corrections from downstream systems and route them efficiently to labelling queues without manual intervention. Monitoring needs to cover throughput -- documents processed per second against SLA -- latency per document by document type, cost per document across the fine-tuned and LLM inference paths, and confidence score distribution over time as a leading indicator of model drift. Accuracy on a held-out test set is a lagging indicator. Distribution shift in confidence scores is the signal that retraining is needed before accuracy visibly degrades.

Entity resolution: Connecting extraction to enterprise data architecture

Extracted entities need to be mapped to canonical references -- drug names to BNF codes, company names to CRM records, product descriptions to taxonomy nodes. This mapping, entity resolution or entity linking, is where most extraction pipelines fail in production.

Sentence embeddings enable semantic similarity-based matching that handles surface form variation -- variant names for the same entity all map to the same embedding cluster. Cosine similarity search against an indexed embedding store returns ranked candidates. Edge cases below the similarity threshold are routed to an LLM-based resolution step.

For enterprise deployments, entity resolution is not an isolated pipeline component -- it is an interface to the organisation's master data management infrastructure. The canonical reference store the resolution pipeline maps to is only as reliable as the MDM system maintaining it. Organisations with fragmented master data -- multiple authoritative sources for the same entity type, inconsistent identifiers across systems -- will find that extraction accuracy is limited not by the NER or embedding models but by the quality of the reference data. Knowledge graph strategies that unify entity references across domains -- connecting a supplier entity in procurement to the same entity in finance and compliance -- extend the resolution capability beyond point-to-point mapping and enable relationship-aware extraction that single-entity resolution cannot provide.

Intelligent document processing at the volumes that matter

When document volume reaches millions, synchronous processing cannot handle the throughput. The production architecture is an asynchronous pipeline: documents queued on ingestion, distributed across a processing fleet, results written to an output store that downstream systems poll or subscribe to. Each processing step -- OCR, layout analysis, text extraction, NER, entity mapping, quality validation -- is a separate service with its own scaling policy and error handling.

Enterprise storage architecture for this pipeline requires object storage for raw and processed documents, a structured output store for extracted entities and quality metadata, and an index layer that supports both exact-match and semantic retrieval against extracted content. Indexing strategy matters for query performance at scale -- full-text indexes for keyword retrieval, vector indexes for semantic search, and composite indexes for filtered queries that combine entity type, source, and time range. Governance requirements -- retention policies, access controls on sensitive document types, lineage tracking from source document to extracted output -- need to be designed into the storage architecture from the outset.

Cost management at document processing scale requires a tiered approach. Not every document warrants VLM or LLM inference -- lightweight classifiers as a first pass route documents to the appropriate processing path, reserving expensive inference for documents where it is genuinely needed. Cost per document needs to be tracked by document type and processing path, with budget thresholds that trigger alerts before inference spend exceeds plan. Autoscaling policies need to be calibrated against queue depth and processing latency rather than CPU utilisation -- a processing fleet that scales on CPU will not respond fast enough to burst ingestion events. Monitoring needs to cover queue depth, processing latency by document type, dead letter queue accumulation rate, and cost per document across processing paths -- these are the operational metrics that determine whether the pipeline is performing within its economic and SLA constraints.

Text and Image Analytics: Sentiment, Forecasting and Vision Language Models in Production

The analytical layer of an AI platform differs from workflow automation in one critical respect: it runs continuously against live data streams rather than processing discrete inputs on demand. Sentiment analysis pipelines ingest customer communications, social content, and regulatory submissions on a scheduled or event-driven basis. Trend forecasting models consume corpus updates as they arrive. Visual attribute extraction processes product image updates as retailers publish them. The production challenges are therefore pipeline challenges as much as model challenges -- continuous data ingestion that handles source variability and schema evolution, monitoring that detects output drift before it affects downstream decisions, and reliability engineering that ensures the pipeline meets its freshness SLA even when upstream sources are unavailable or delayed. A sentiment model that produces accurate outputs when it runs but runs unreliably is not a production asset. It is a liability that generates intermittent trust

Sentiment analysis beyond positive-negative classification

Binary sentiment classification is a solved problem. Production sentiment analysis for enterprise use cases is not. The enterprise use cases that drive real investment are specific: customer operations teams monitoring sentiment across support tickets and call transcripts to identify service failure patterns before they escalate; compliance and legal teams tracking sentiment in regulatory submissions, analyst reports, and news coverage to detect emerging risk signals; product intelligence teams attributing sentiment to specific product attributes across review corpora to inform roadmap decisions. Each of these requires more than a document-level positive or negative score.

The four capabilities that production sentiment analysis requires are aspect-level sentiment, entity-linked sentiment, temporal trend analysis, and anomaly detection. Aspect-level sentiment identifies both the attribute being discussed and its associated sentiment in the same pass -- a customer communication that is positive about delivery speed and negative about product quality needs both signals, not a net sentiment score. Entity-linked sentiment requires the NER pipeline to run first, with sentiment scores attributed to specific identified entities -- knowing that sentiment toward a specific product line is declining is more actionable than knowing that overall brand sentiment is flat. Temporal analysis requires the pipeline to run on a consistent schedule against a consistent data slice, with results stored in a time-series format. Anomaly detection flags sudden shifts in sentiment volume or polarity that warrant investigation -- a spike in negative sentiment about a specific product attribute two days after a release is a signal that needs to reach the product team, not a data point in a weekly report.

Evaluation for production sentiment pipelines needs to go beyond held-out test set accuracy. Aspect extraction precision and recall need to be measured separately from sentiment classification accuracy -- a pipeline that extracts aspects correctly but misattributes sentiment polarity fails differently from one that misses aspects entirely. Consistency evaluation -- whether the pipeline produces the same sentiment scores for semantically equivalent inputs expressed differently -- is a reliability metric that test set accuracy does not capture. Drift detection needs to monitor the distribution of sentiment scores over time, not just model confidence. A genuine shift in customer sentiment looks identical to model drift in the output distribution -- distinguishing between the two requires monitoring both the input data distribution and the model's confidence score distribution simultaneously. When both shift together, it is likely a real-world change. When confidence degrades while input distribution is stable, it is model drift requiring retraining.

Trend forecasting: From pattern recognition to actionable signal

Trend forecasting in production combines two distinct capabilities that are frequently conflated. The first is pattern detection -- identifying topics that are growing in volume, velocity, or sentiment shift across a text corpus. The second is signal interpretation -- determining whether a detected pattern represents a genuine emerging trend or an artefact of data collection changes, seasonal variation, or a one-off event. The engineering challenge is building a pipeline that does both reliably and continuously.

Traditional time-series models -- ARIMA, Prophet, and their variants -- remain the appropriate tool for volume and velocity analysis of topic streams where the signal is numerical and the patterns are relatively stable. These models provide statistically grounded trend decomposition, seasonality adjustment, and confidence intervals that LLMs cannot replicate. Replacing them with LLMs for quantitative time-series analysis overestimates what LLMs are suited for and underestimates the reliability that statistical models provide for this specific task.

LLMs contribute at different points in the pipeline where their strengths are genuinely relevant. Topic modelling using BERTopic -- which applies transformer embeddings and clustering to identify coherent topic clusters across a corpus -- benefits from LLM-based label generation to produce human-readable topic descriptions rather than keyword lists. The synthesis step -- taking a detected trend signal, its supporting evidence, and its temporal context and generating a coherent analyst-grade narrative -- is where LLMs add the most value. A human analyst does not want a list of trending keywords. They want a coherent statement of what is shifting, why it might be significant, what the supporting data shows, and what alternative explanations exist. That synthesis task -- grounded in structured signal from the statistical layer -- is where LLMs augment rather than replace the analytical pipeline.

Anomaly detection flags unusual acceleration or deceleration in specific topic clusters for analyst review. The production requirement is that flagged anomalies include the evidence that triggered the flag -- volume change magnitude, sentiment shift direction, source distribution -- so that analysts can evaluate whether the signal warrants action or represents noise. An anomaly detection system that flags without explanation generates alert fatigue. One that provides structured evidence enables faster and more reliable analyst judgement.

Vision Language Models: From image understanding to structured data

Vision Language Models combine vision encoders with language model architectures to understand and reason about images in natural language. Extracting structured product attributes from fashion images, identifying components in engineering diagrams, reading text from document photographs, classifying satellite imagery -- all problems where VLMs provide a path from unstructured visual data to structured, queryable output.

The production challenges are throughput, cost, and infrastructure management. VLM inference is significantly more expensive per unit than text-only inference. At millions of images per day, processing every image through a VLM is not economically viable. The production architecture uses a tiered approach: lightweight visual classifiers as a first pass route images to the appropriate processing path, reserving VLM inference for cases where richer understanding is genuinely needed. This routing layer is where the most consequential cost and engineering decisions are made.

GPU scheduling for VLM workloads is GPU-memory-bound rather than compute-bound for most enterprise image sizes. Batching strategy needs to be calibrated against the latency SLA -- larger batches reduce cost per image but increase per-image latency. Asynchronous batch workloads with a 24-hour turnaround can tolerate large batches. Near-real-time attribute extraction against live product feeds requires smaller batches with tighter scheduling windows. These two workload types need separate infrastructure configurations and scaling policies.

Cost monitoring needs to track cost per image by processing tier, GPU utilisation against provisioned capacity, and queue depth against throughput rate. Autoscaling policies should be configured against queue depth and processing latency rather than GPU utilisation, which is a lagging indicator at the batch sizes VLM workloads typically require. Budget thresholds that trigger autoscaling or alerting need to be defined before the pipeline goes to production.

MERIT

The Architecture Decisions That Determine Whether AI Scales or Stalls

This chapter addresses the architecture decisions that determine what happens after production -- whether an AI platform compounds in value as new use cases are added, or becomes progressively harder to maintain and extend. These are not model choices. They are infrastructure and design choices with long-term organisational consequences that most AI programmes make implicitly rather than deliberately.

Modular and composable versus monolithic and bespoke

The most consequential architecture decision in an enterprise AI programme is whether individual AI capabilities are built as reusable, composable modules or as bespoke, use-case-specific models. The bespoke approach is faster to start: a model built specifically for one use case can be optimised for that use case and deployed quickly. The composable approach takes longer to establish but changes the economics of every subsequent use case -- assembly and configuration rather than rebuild from scratch.

The organisational conditions that determine whether modular architecture succeeds are as important as the technical design. Without a platform team that owns shared AI modules and treats them as governed, versioned services, composable architecture drifts into a different kind of fragmentation -- modules maintained inconsistently by different teams, with no clear process for versioning, breaking change notification, or retirement. Domain teams consuming modules they did not build need confidence that those modules will be maintained, documented, and supported. That confidence requires a platform team operating with a product mindset rather than a shared codebase mindset.

Reuse governance defines the operational rules that make modularity durable. Which modules are available for reuse, what their SLAs are, how breaking changes are communicated, and what the deprecation process looks like when a module is superseded -- these need to be defined and enforced, not assumed. Without them, modules accumulate undocumented dependencies, version proliferation becomes unmanageable, and the promised reuse benefits erode as each team works around modules they cannot rely on.

Module lifecycle management covers the full lifespan from initial publication through active use to retirement. A module that is no longer fit for purpose but has active consumers cannot simply be switched off. Retirement requires identifying all consumers, communicating the timeline, providing a migration path to the replacement, and confirming cutover before decommission. Organisations that skip this discipline end up maintaining zombie modules indefinitely because nobody has mapped the dependency graph or owns the retirement process.

KIAA -- Merit's modular AI framework -- is built on these principles. Production-ready, configurable modules for entity recognition and mapping, semantic search and retrieval, cognitive workflow automation, real-time extraction, and NLP-powered analytics are governed centrally and composed for each engagement. A new use case is assembled from these modules, configured for the specific domain and data, and deployed against the client's existing infrastructure without requiring a rip-and-replace. The framework's modularity is what allowed Merit to deploy a renewable energy intelligence system in less than 70% of the contracted timeline and to scale fashion data processing to 36 million SKUs daily without increasing operational overhead.

Integration-first versus model-first design

Most AI systems are designed model-first: the model is selected or built, validated in isolation, and integration with existing systems addressed afterward. The consequence is that latency, reliability, and data format requirements surface as constraints after the model is built -- at which point changing the model to satisfy them is expensive and changing the integration to accommodate the model is often technically impossible.

Integration-first design inverts this. The interfaces between the AI system and the existing systems it needs to work with are defined before model selection begins. In enterprise environments, this means defining three integration layers upfront. Event buses -- Kafka, Pub/Sub, EventBridge -- decouple AI systems from the upstream sources and downstream consumers they need to interact with, allowing the AI pipeline to consume and publish without tight coupling to any specific system. API gateways provide the governed interface layer between AI services and consuming applications, handling authentication, versioning, rate limiting, and routing without those concerns leaking into the AI pipeline itself. Middleware layers handle protocol translation, data format normalisation, and retry logic between the AI system and legacy systems that do not expose clean interfaces natively -- a common constraint in enterprise environments where the AI pipeline needs to integrate with systems that predate modern API standards.

Defining these integration patterns first surfaces the requirements that should drive model selection: what latency budget does the API gateway contract impose, what data format does the downstream CRM expect, what reliability guarantee does the event bus provide. A model that cannot meet the latency budget of its integration contract is the wrong model for that context -- and that is a much cheaper discovery to make before the model is built than after.

Real-time by design versus batch-first with real-time bolted on

The difference between an AI system that processes data as it arrives and one that processes it in scheduled batches is not just a latency difference. It is an architectural difference that affects how the system is built, how it is monitored, and how it fails. Building real-time capability into a batch-first system after the fact typically requires rebuilding the pipeline from scratch. Building batch capability into a real-time system is straightforward -- you simply do not consume from the stream continuously.

Many enterprise environments operate hybrid pipelines -- some data flows arrive in real time, others in scheduled batches -- driven by source system constraints, legacy infrastructure limitations, or regulatory requirements. The choice is not always a free one, and presenting real-time as the universal default does not reflect this.

The more useful framing is whether the architecture can accommodate both modes without a rebuild. A real-time capable, event-driven pipeline can support batch consumption as a specific mode without architectural change. A batch-first architecture cannot support real-time requirements without significant rework because latency and failure-handling assumptions are baked into the design.

For new AI systems where latency requirements are not yet fully defined -- the common case early in a programme -- designing for real-time capability while operating in batch mode is lower risk than the reverse. For organisations with established batch infrastructure that cannot be replaced, a hybrid approach is pragmatic: batch ingestion from legacy sources into a streaming layer that the AI pipeline consumes in near real time, decoupling the AI system's latency characteristics from the upstream source's delivery cadence.

CHAPTER 7

Two Production Deployments

The following are from Merit's applied AI work. The three case studies were selected because each one illustrates a different combination of the operationalisation challenges this guide has covered -- and each represents a distinct AI architecture deployed against a real business problem at production scale. Together they cover the spectrum from agentic reasoning to extraction to multimodal processing, and from lightly regulated commercial intelligence to heavily regulated healthcare data. We are describing them at a level of technical detail that gives you an honest picture of what was built and how it was operationalised -- not just what was achieved.

Renewable energy intelligence: Agentic RAG at scale with 100% automation

A US-based media and events company needed to launch a competitive data product covering renewable energy companies in the supply chain within six months, against tight budget constraints. The requirement was continuous, automated identification of renewable energy companies from online sources, extraction of over 20 structured data points per company, and generation of accurate company summaries -- at the accuracy standard required for a commercial intelligence product.

- **Architecture.** An end-to-end automated pipeline combining ScrapeX for web data collection with an agentic RAG architecture built on KIAA. The agentic layer decomposed the data gathering task into four sequential steps -- source identification, company discovery, structured data extraction, and summary generation -- executing each via tool-calling and retrieval. Validation gates between steps caught extraction errors before they propagated downstream, ensuring that malformed or low-confidence outputs were quarantined rather than passed to the next stage. LLMs handled both structured data extraction from unstructured web content and coherent summary generation from extracted data points.

- **Deployment model.** The pipeline ran continuously without human intervention on a defined update cycle. Automated quality monitoring tracked extraction confidence scores, field completion rates, and anomaly signals across the output dataset, flagging exceptions for periodic human review rather than requiring active oversight of routine operations.

- **Reliability and monitoring.** Each pipeline stage had independent error handling and dead letter capture for failed extractions. Monitoring covered extraction accuracy by data field, pipeline throughput against SLA, and anomaly rates in the output dataset. The validation gate architecture meant that reliability was enforced structurally -- errors were contained at the stage where they occurred rather than discovered in the final output.
- **Outcomes.** Delivered in less than 70% of the contracted turnaround time. Dataset captured 2x the record volume anticipated for the MVP. 40% cost savings through KIAA and ScrapeX accelerator platforms. 100% automated ongoing operation with zero human intervention required in routine data gathering and updates.

NHS formularies aggregation: NER, sentence embeddings and LLM inference at 95%+ accuracy across 2.5 million documents

A US-based media and events company needed to launch a competitive data product covering renewable energy companies in the supply chain within six months, against tight budget constraints. The requirement was continuous, automated identification of renewable energy companies from online sources, extraction of over 20 structured data points per company, and generation of accurate company summaries -- at the accuracy standard required for a commercial intelligence product.

- **Architecture.** Data collection via ScrapeX and custom spiders across all 300 NHS sources. A Named Entity Recognition system powered by LLMs with user-defined labels, identifying drug names, dosages, indications, and formulary decisions across heterogeneous document formats. Entity mapping using sentence embeddings and cosine similarity for standard cases, with prompt-based LLM resolution for edge cases where embedding similarity was insufficient. LLM inference for NICE tagging and BNF code alignment. A normalisation pipeline ensuring consistent BNF code mapping and presentation format across all sources.
- **Governance and compliance.** The 95%+ accuracy requirement was not a performance target -- it was a regulated healthcare data standard. The pipeline was designed with this constraint as a primary architectural requirement rather than a post-deployment benchmark. Confidence thresholds were defined per extraction type, with low-confidence outputs routed to validation rather than passed directly to downstream consumers. BNF code mapping was auditable at the record level -- every mapping decision traceable to the source document, the extraction model, and the resolution path taken. The normalisation pipeline enforced presentation consistency as a governance control, ensuring that the same drug appeared identically across all 300 source variations in the output dataset.

- **Outcomes.** Over 95% accuracy across complex, unstructured document formats. Data update turnaround time reduced by 70% through automated workflows replacing manual collection and extraction. Operational costs unchanged despite the volume increase -- automation absorbed the scale without additional headcount. Architecture designed for expansion to new geographies with minimal retraining.

MERIT

Is Your AI Programme Ready for Production? A Diagnostic

These questions are written for AI leaders evaluating their current programme against the failure modes described in this guide. Work through them against what you have actually built and deployed, not what is on the roadmap. The value is in identifying where gaps are concentrated relative to the failure modes most likely to affect your current programme.

For each question, assess against three positions:

- Fully in place,
- Partially in place with known gaps

  or
- Not in place.

The value is in identifying where gaps are concentrated relative to the failure modes most likely to affect your current programme.

Architecture

- **Are your AI capabilities built as composable, independently deployable modules -- or as bespoke, use-case-specific models that share no components?**

  Fully in place means shared modules are governed, versioned, and reused across use cases. Partially in place means some components are shared while others are rebuilt per use case. Not in place means every use case is a full rebuild. The partial state is the most common -- the question is whether the shared components are governed deliberately or accumulating informally. Informal reuse without governance produces a different fragmentation problem over time.

- **Can you deploy, update, and scale individual components of your AI system independently without taking down or redeploying the entire platform?**

  Fully in place means components are containerised, independently deployable, and governed through a CI/CD pipeline with rollback capability. Partially in place means some components are independently deployable while others remain coupled to a shared deployment. Not in place means all changes require a full system redeployment. The partial state requires mapping which coupled components carry the highest change frequency -- those are the ones where deployment coupling is most actively constraining engineering velocity.

Governance

- **Is governance and auditability designed into your AI architecture from the start -- or planned as a phase two activity after the core system is deployed?**

  Fully in place means lineage, output logging, model versioning, and policy enforcement are operational before the system goes to production. Partially in place means some governance controls are in place while others -- typically lineage or output auditability -- are deferred. Not in place means governance is planned but not yet implemented. In regulated industries, the partial state needs to be mapped against specific regulatory requirements to identify which gaps create deployment risk versus which can be resolved post-launch.

- **Do you have confidence threshold gates in your AI pipelines that route low-confidence outputs to secondary models or human review rather than passing them directly to downstream systems?**

  Fully in place means confidence gates are defined, tested, and operational across all production pipelines with defined review SLAs. Partially in place means gates exist on some pipelines but not others -- typically newer pipelines are gated while legacy ones are not. Not in place means all outputs pass directly to downstream systems regardless of confidence. The partial state requires identifying which ungated pipelines carry the highest error cost if low-confidence outputs reach downstream decisions.

Operations

- **Do you have a production NER pipeline with a fine-tuned model for high-frequency entities, an LLM-based fallback for edge cases, and a feedback loop that improves accuracy over time?**

  Fully in place means all three layers are operational with defined retraining schedules and monitored accuracy trends. Partially in place means the fine-tuned model is operational but the feedback loop or LLM fallback is not yet implemented. Not in place means a static model is running without improvement mechanisms. A static NER model in production will have lower accuracy at month six than at launch as the data distribution shifts -- the partial state needs a timeline for closing the gap

- **If you are running RAG, do you have a retrieval evaluation framework that measures whether retrieved chunks are actually relevant -- separately from whether the generated output sounds correct?**

  Fully in place means retrieval precision and recall are tracked as first-class metrics independently of generation quality, with embedding drift monitoring in place. Partially in place means end-to-end output quality is monitored but retrieval quality is not measured separately. Not in place means evaluation is based entirely on whether responses sound correct. RAG systems fail quietly -- a well-written response based on irrelevant context is indistinguishable from a correct one without explicit retrieval evaluation

Integration

- **Is your AI pipeline event-driven and real-time capable by design -- or batch-first with real-time planned as a future enhancement?**

  Fully in place means the pipeline is event-driven and can support both real-time and batch consumption without architectural change. Partially in place means some data paths are real-time while others remain batch-dependent, typically constrained by legacy source systems. Not in place means the pipeline is batch-first throughout. The partial state requires mapping which batch-dependent paths feed time-sensitive AI use cases -- those are the ones where latency constraints will tighten first as the use case matures

Reading the results

- **Is your AI pipeline event-driven and real-time capable by design -- or batch-first with real-time planned as a future enhancement?**

  A pattern of fully in place responses indicates a programme that is operationally mature across the dimensions that most commonly cause production failure. A pattern of partially in place responses -- the most common position for programmes that have moved beyond early pilots -- indicates that prioritisation should be driven by which gaps intersect with the highest-risk failure modes for your specific use cases. A pattern of not in place responses across multiple categories indicates that foundational operationalisation work is a prerequisite before the programme can scale reliably.

CHAPTER 9

How Merit Operationalises AI at Scale

Merit's AI engagements are structured around the full delivery stack - not just the model layer. Production AI depends on governed data pipelines, reliable ingestion infrastructure, and integration architecture that connects AI outputs to the systems and workflows that consume them. In most enterprise environments, some or all of that foundation needs to be built or modernised alongside the AI capability itself. Merit addresses both - data engineering and modernisation where the infrastructure needs to be established, and AI delivery on top of it.

KIAA - Merit's modular AI framework - is the output of building the same AI capabilities repeatedly across energy, healthcare, fashion, automotive, and agricultural intelligence engagements and formalising them into production-ready, configurable modules. When an engagement requires cognitive workflow automation, NER, semantic search, or agentic RAG, we are not building from scratch. We are configuring, extending, and integrating capabilities that have already been validated in production - concentrating engineering effort on the use-case-specific configuration rather than rebuilding foundational infrastructure that should not need to be rebuilt each time.

Traditional AI solutions can be rigid and difficult to scale. KIAA is built for modularity -- allowing businesses to integrate AI into their existing workflows without the rip-and-replace. That is not a marketing claim. It is a design constraint we imposed on ourselves because we had seen what happens when AI is not built that way.

What a Merit AI engagement delivers

- **Cognitive workflow automation.** LLM-powered document classification, entity extraction, content tagging, and intelligent routing -- delivered as production pipelines with orchestration, confidence gating, exception handling, and monitoring built in. Every pipeline is designed for operational handover, not demonstration.
- **NER and entity resolution at scale.** Fine-tuned transformer models for high-frequency entities, LLM-based edge case resolution, and sentence embedding-based entity mapping -- with continuous feedback loops and defined retraining schedules that improve accuracy over time rather than allowing it to degrade as data distributions shift.

- **RAG and agentic workflows.** Retrieval architectures with semantic chunking, embedding-based retrieval, reranking, and retrieval evaluation frameworks that measure chunk relevance independently of generation quality. Agentic systems with constrained action spaces, deterministic fallback paths, and human-in-the-loop checkpoints for decisions above defined confidence thresholds.
- **Production analytics and multimodal processing.** Aspect-level sentiment analysis, BERTopic-based trend detection, LLM-powered synthesis, and VLM-based attribute extraction -- built as continuously running operational pipelines with drift detection, freshness SLAs, and monitoring against live data.
- **Enterprise integration.** AI capabilities delivered against defined integration architecture -- event buses for decoupled data consumption and output publishing, API gateways for governed interfaces between AI services and consuming applications, and middleware layers for legacy system connectivity. Integration requirements are defined before model selection, not addressed as a subsequent phase.
- **Platform lifecycle management and GenAIOps.** Model versioning linked to specific outputs, model registry management, performance monitoring across deployed models, inference cost tracking, and component-level deployment and rollback capability. AI platforms maintained as operational systems with defined ownership, SLAs, and improvement cycles.
- **Secure and governed by design.** ISO 27001 certified. Audit-ready pipelines with full output traceability, lineage from source data to model output, policy enforcement at the pipeline layer, and compliance built in as a design constraint rather than a post-deployment retrofit.
- **60% faster AI deployment** -- measured against comparable build-from-scratch engagements, reflecting the reduction in pipeline build time achieved through KIAA's production-ready modules and ScrapeX accelerators. Typically observed in cognitive workflow automation and NER-based extraction engagements where foundational capabilities do not need to be rebuilt from scratch.
- **80%+ reduction in manual effort** -- measured against the pre-automation baseline for tagging, QA, enrichment, and response workflows within the same organisation. Typically observed in document intelligence, entity extraction, and cognitive workflow automation engagements where high-volume, repetitive human review is replaced by confidence-gated automated pipelines.

- **2 to 5x faster time-to-insight** -- measured as the reduction in elapsed time between data arrival and actionable output reaching decision-makers or downstream systems. The range reflects variation by use case: 2x is typical for analytics and reporting pipelines where batch processing is replaced by near-real-time extraction; 5x is observed in always-on intelligence pipelines where previously manual aggregation is fully automated.
- **100% compliant by design -** Reflects engagements where governance, auditability, and regulatory alignment are built into the pipeline architecture from the outset rather than retrofitted. Applies specifically to regulated industry deployments -- healthcare, financial services, energy -- where audit-ready lineage, model versioning, and policy enforcement at the pipeline layer are contractual or regulatory requirements.

Ready to move from pilot to production?

If your AI programme is generating strong results in controlled environments but stalling on the path to production -- or if you are evaluating whether to build your AI capabilities in-house or partner with a team that has already solved the operationalisation problem at scale -- we are the right people to talk to.

We will tell you honestly where the gaps are in your current architecture, what it will take to close them, and how KIAA can accelerate the parts of that work that do not need to be built from scratch.

About Merit

Merit Data & Technology, part of Merit Group Limited, is a trusted partner in AI-driven data and digital transformation. With over two decades of experience, We deliver scalable, secure and AI-ready automation and data solutions tailored to business needs. **Read More About Us**

www.meritdata-tech.com