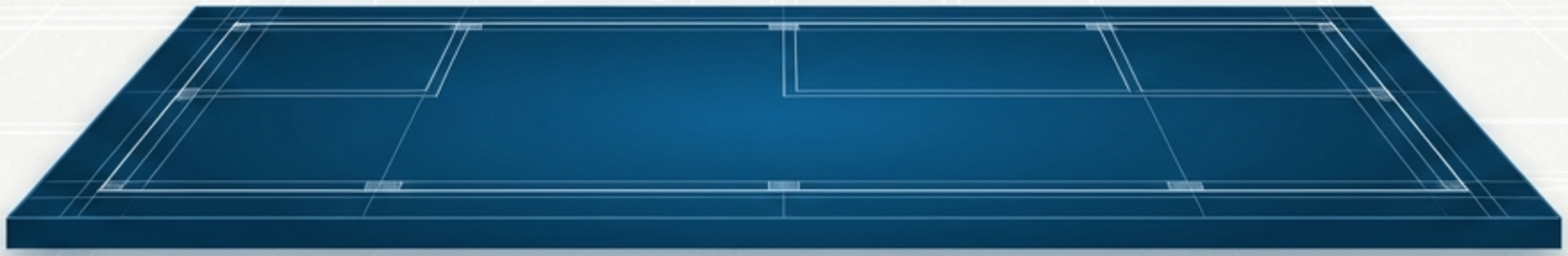
The Blueprint for Modern AI Development

A Strategic Distillation of Chip Huyen's **'AI Engineering'**



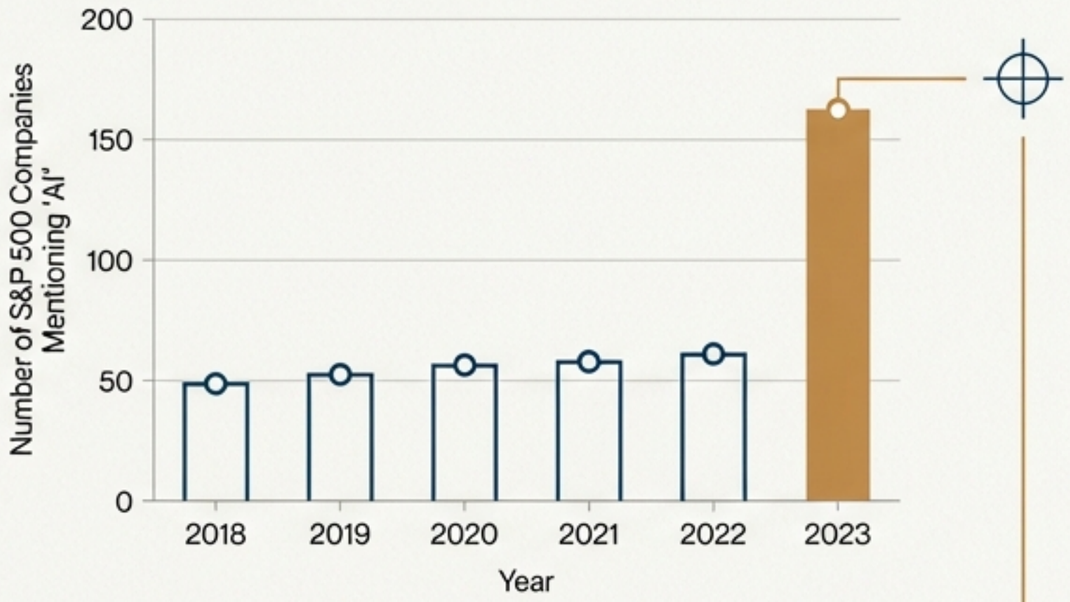
Moving beyond '**prompt hacking**' to establish a **rigorous, systematic engineering discipline** for building **reliable, production-grade applications** with foundation models.

Based on the work of Chip Huyen, author of *Designing Machine Learning Systems* and *AI Engineering*.

The New Reality: An Engineering Discipline Forged by Scale

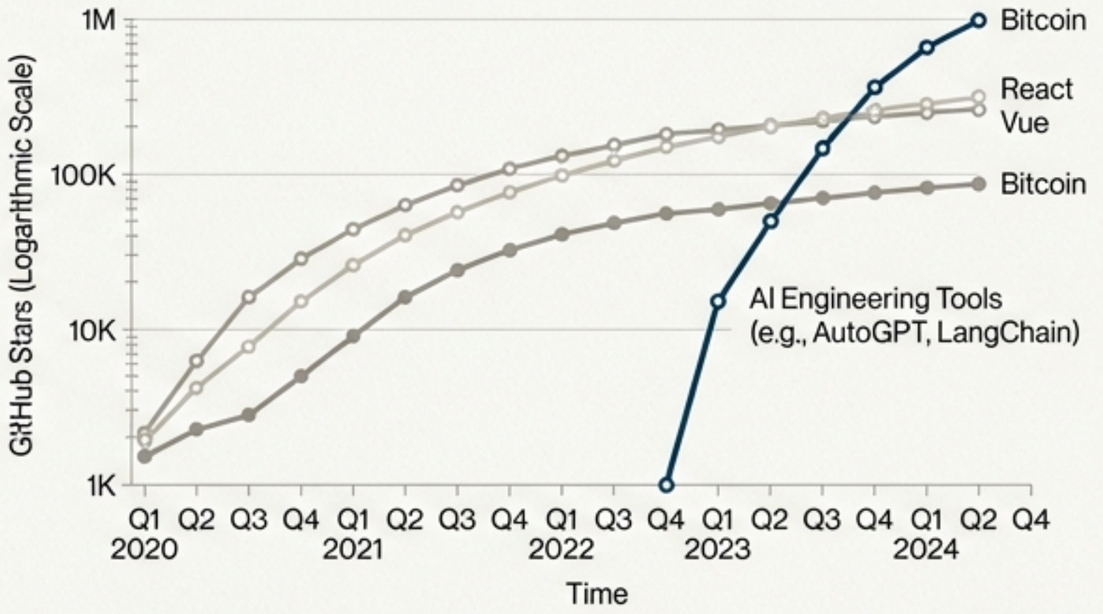
The scaling up of AI models has led to two major consequences: a massive increase in capability, unlocking an explosion of applications, and the rise of 'model as a service,' which changes how we build software.

AI Becomes a Boardroom Imperative



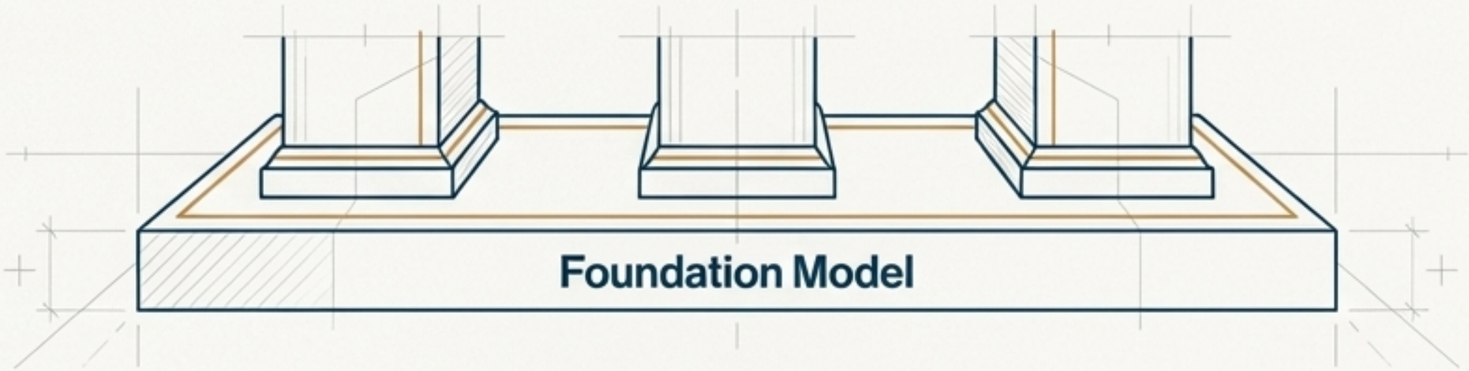
In Q2 2023, one in three S&P 500 companies mentioned AI, a threefold increase from the previous year.

The Fastest-Growing Engineering Ecosystem



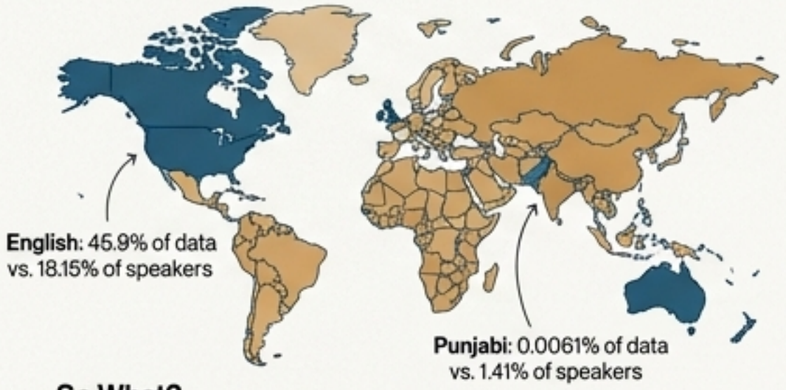
Within two years, open-source AI tools have garnered more GitHub stars than Bitcoin, signalling an unprecedented rate of adoption and innovation.

The Core Material: Understanding the Foundation Model



1. Training Data

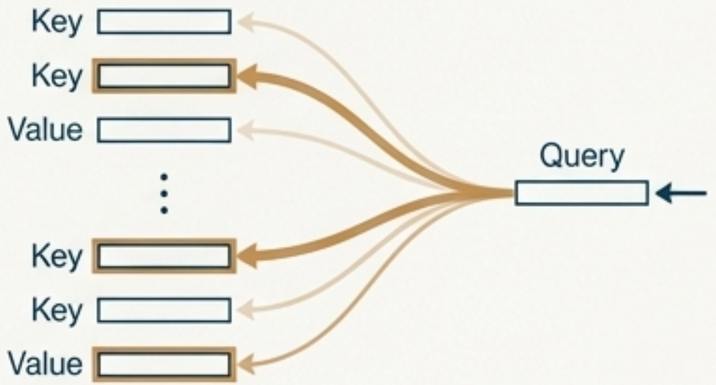
Models are a reflection of their training data. Their capabilities, biases, and limitations are encoded from vast, internet-scale datasets.



So What?
An engineer must scrutinise a model's data origins to understand its strengths (e.g., coding) and weaknesses (e.g., low-resource languages).

2. Transformer Architecture

The Transformer is the dominant architecture, enabling parallel processing of sequences. Its core innovation is the **attention mechanism**.



'Attention' allows the model to weigh the importance of different input tokens when producing the next token, akin to a researcher referencing multiple pages to write a summary.

3. Probabilistic Nature

Models do not 'know' answers; they **sample** them from a probability distribution. The same input can produce different outputs.

- **Logprobs**: Log probabilities measuring model confidence.
- **Temperature**: Controls randomness. Low temp = predictable, high temp = creative.
- **Top-p**: Nucleus sampling; considers the most probable set of tokens.

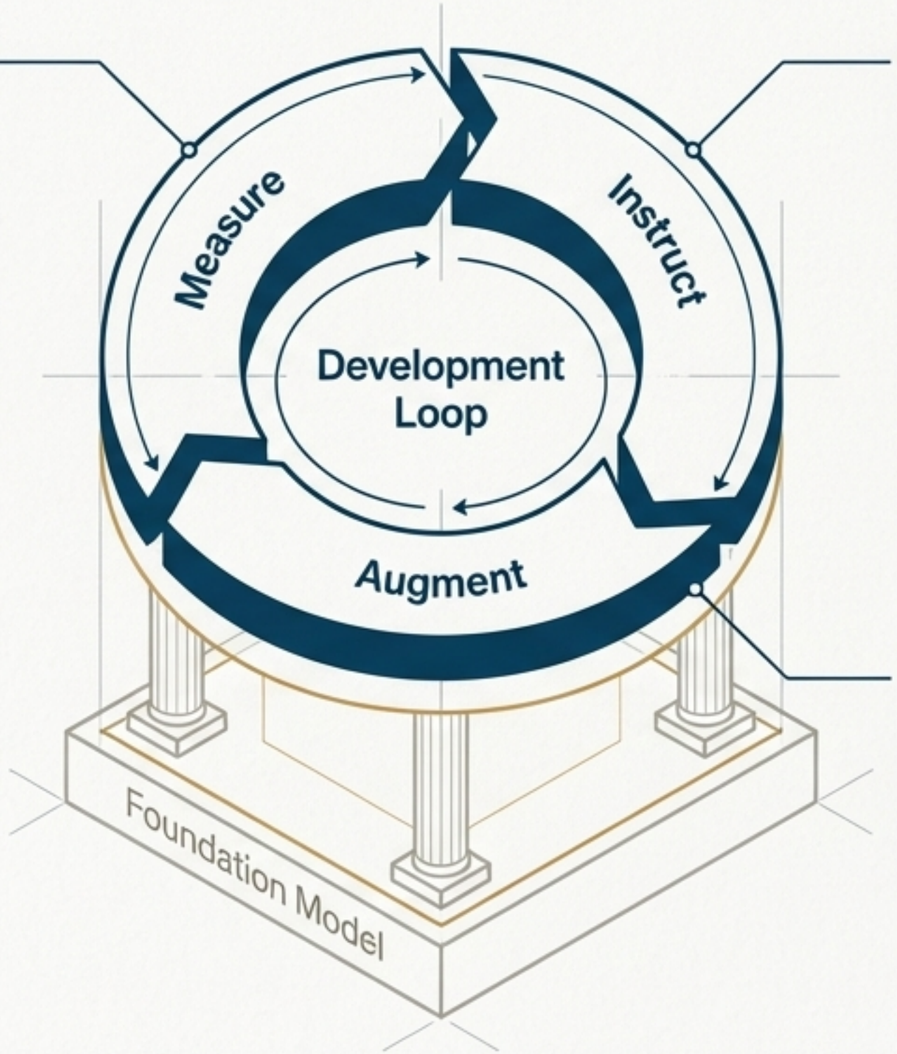
So What?
Engineering for AI means engineering for uncertainty. Managing this probabilistic nature is key to building reliable systems.

The Core Workflow: An Iterative Loop of Instruction, Augmentation, and Measurement

Measure (Rigorous Evaluation)

If you can't measure it, you can't improve it. Evaluation moves from simple accuracy to a multi-faceted assessment of quality, consistency, and safety.

AI as a Judge. Using a powerful model (e.g., GPT-4) to score the outputs of another model based on a detailed rubric. This scales evaluation beyond manual human review.



Instruct (Prompt Engineering)

The art of designing inputs to elicit desired behaviours from the model without changing its weights. This is the primary interface for controlling the model.

"The performance difference between a simple prompt and a well-engineered one can be monumental."

Prompting Unlocks Performance	
Gemini Ultra (5-shot prompt)	**83.7%**
Gemini Ultra (CoT@32 prompt)	**90.04%**

+6.34% on MMLU

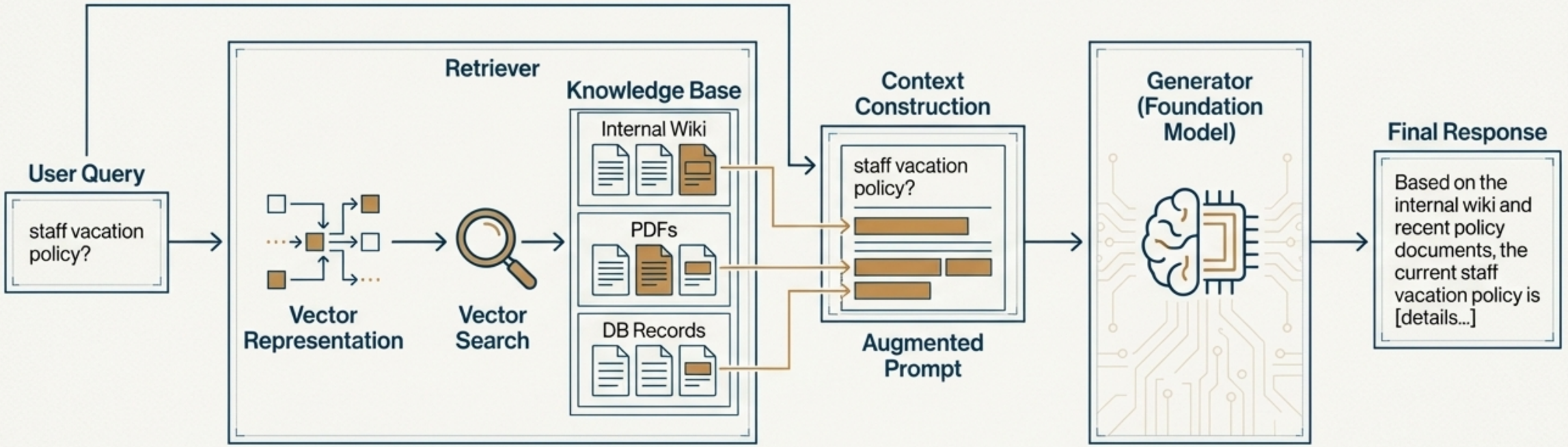
Augment (Context Construction)

Providing the model with external, relevant information at inference time to ground its responses and overcome knowledge limitations.

Retrieval-Augmented Generation (RAG). The system first *retrieves* relevant documents from a knowledge base (e.g., internal wikis, user manuals) and then provides them to the model to *generate* an answer.

Anatomy of Augmentation: The RAG Architecture

RAG is the primary pattern for building knowledge-intensive applications. It separates the model's parametric knowledge (from training) from the external world's source knowledge (provided at runtime).



Key Engineering Decision: The Retriever

- **Term-based Retrieval:** Fast and effective for keyword matching (e.g., BM25). Relies on lexical overlap.
- **Embedding-based Retrieval:** Captures semantic meaning. Finds conceptually related information (e.g., query for "staff vacation policy" finds docs about "employee holidays").
- **Hybrid Approach:** Combining both is often the most robust solution.

Measuring What Matters: The Expanded Scope of Evaluation

Traditional ML evaluation focused on a single model's performance on a narrow task (e.g., classification accuracy). AI Engineering requires evaluating a ***system*** across multiple dimensions.

Domain-Specific Capability

Is the model good at the core task?

e.g., understanding legal contracts, writing SQL

Often measured with benchmarks like HumanEval (for code) or MMLU (for general knowledge).

Generation Capability

Is the output well-written and factually correct?

Fluency: Is it grammatically correct?
Coherence: Is it logical?
Factual Consistency: Does it align with provided context or known facts?

Instruction-Following Capability

Does the model adhere to constraints?

e.g., “respond in JSON format,” “use a formal tone,” “limit response to 100 words”

Crucial for creating structured, reliable outputs for downstream applications.

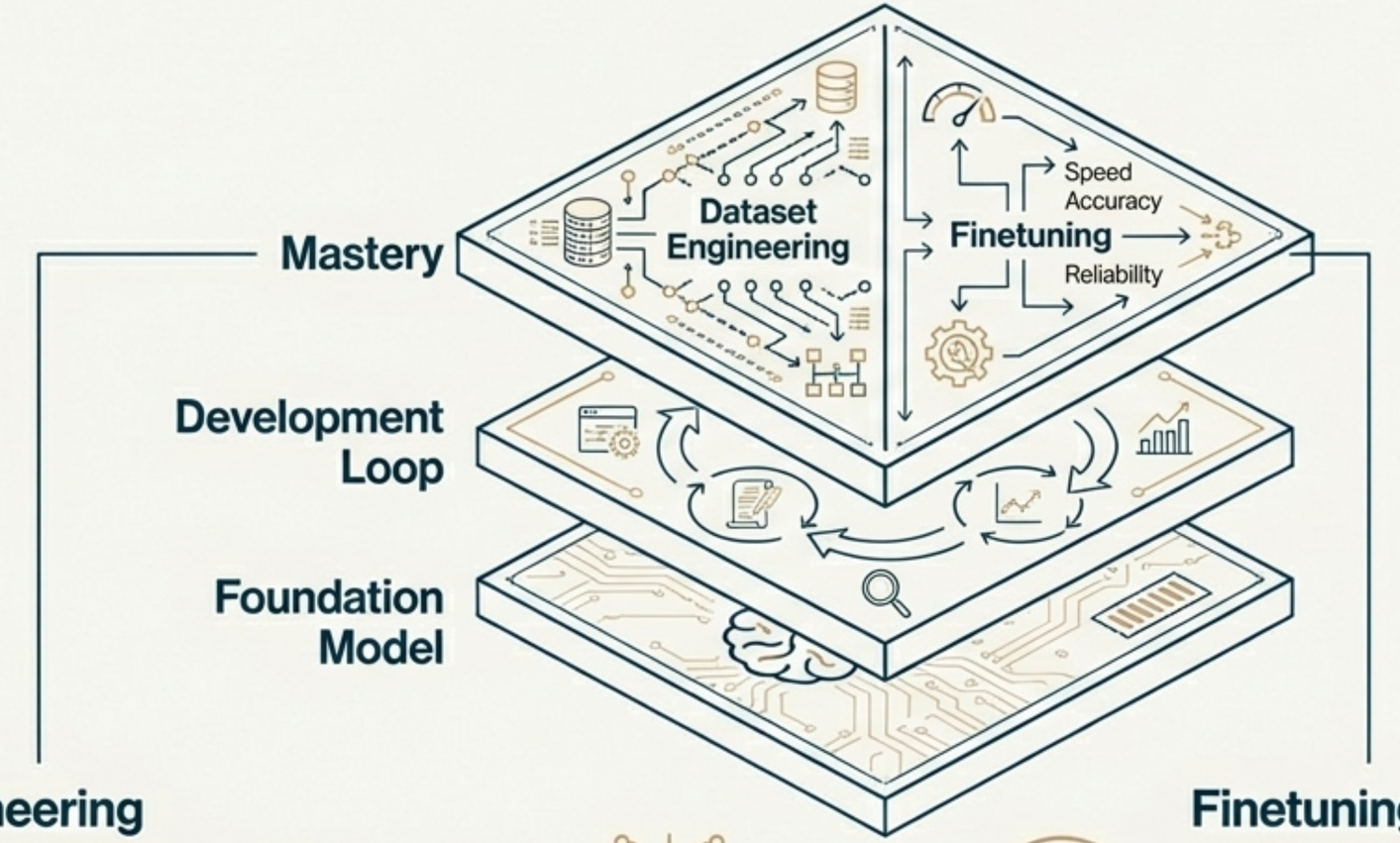
Cost & Latency

How fast and expensive is the system?

Time To First Token (TTFT)
Time Per Output Token (TPOT)
Token Cost (Business-critical metrics for UX and viability)

Mastering the Craft: From Prototype to Differentiated Product

The journey from a working prototype to a production-ready system is where true engineering excellence is demonstrated.



"The journey from 0 to 60 is easy, whereas progressing from 60 to 100 becomes exceedingly challenging."

– Ding et al. (2023), UltraChat paper

Dataset Engineering

Moving beyond prompting to teach the model new skills requires high-quality, curated data. The quality of your data is a primary competitive advantage.

Finetuning

Systematically adapting a general-purpose model to become a specialist for your specific domain or task, improving its performance, reliability, and efficiency.

The Art of the Dataset: Quality Trumps Quantity

A common refrain from practitioners: “Finetuning is easy, but getting data for finetuning is hard.” The effort has shifted from feature engineering to dataset engineering.

The LIMA Paper ("Less Is More for Alignment")

A Llama-65B model finetuned on just **1,000** carefully curated examples was competitive with GPT-4 in **43%** of cases.

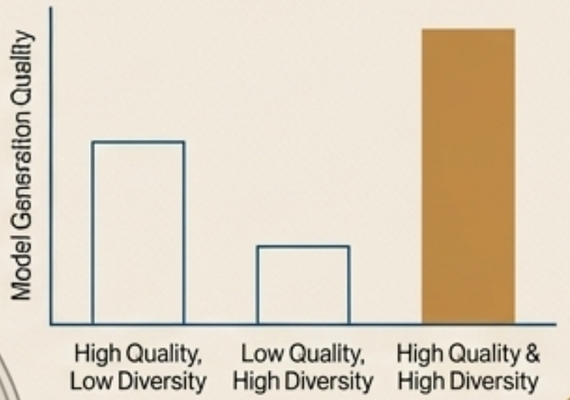
Quality

Data must be relevant, aligned with task requirements, consistent, correctly formatted, unique, and compliant.

Coverage / Diversity

The dataset must be diverse enough to cover the range of inputs and scenarios the model will encounter in production. A lack of diversity leads to a brittle model.

Impact of Data Characteristics

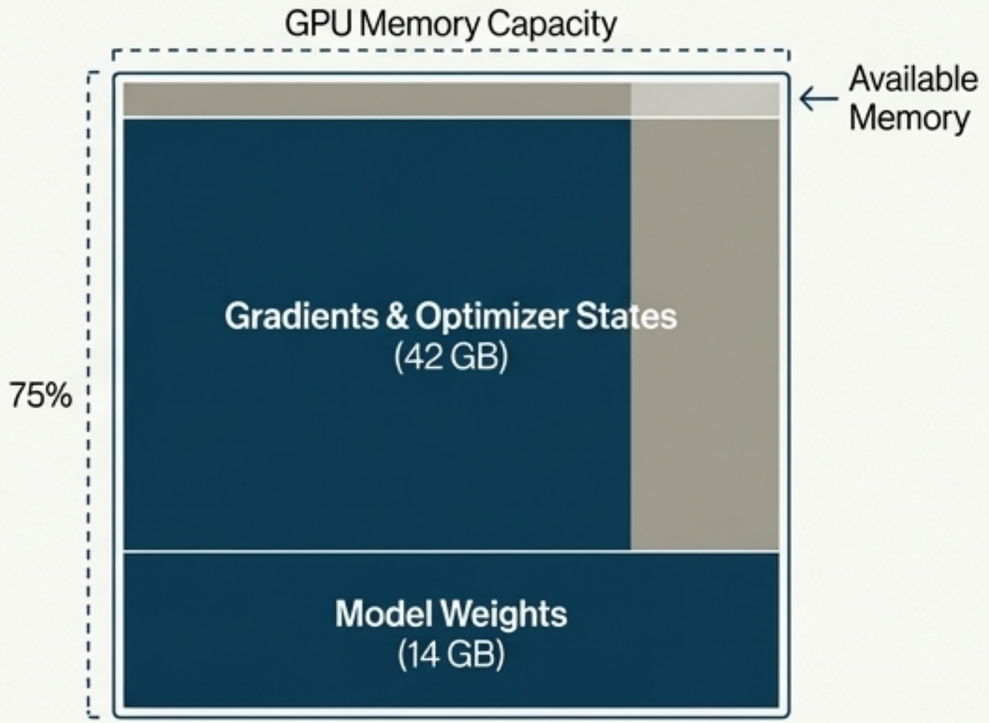


Quantity

How much data is needed depends on the finetuning technique (PEFT requires far less than full finetuning) and task complexity. Start small and scale.

Model Specialisation: The Power of Parameter-Efficient Finetuning (PEFT)

The Challenge with Full Finetuning

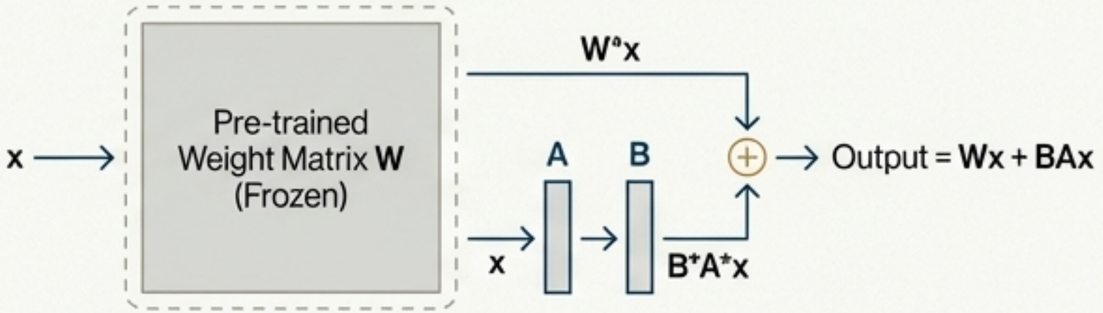


A 7B parameter model requires **~56 GB** of GPU memory for full finetuning, exceeding the capacity of most professional-grade GPUs.

The PEFT Solution

Freeze the original model weights and inject a small number of new, trainable parameters. This achieves strong performance with a fraction of the computational cost.

Dominant Technique: LoRA (Low-Rank Adaptation)



Instead of updating a large weight matrix `W, LoRA learns two small, 'low-rank' matrices `A` and `B` whose product approximates the change needed for `W. Only `A` and `B` are trained.

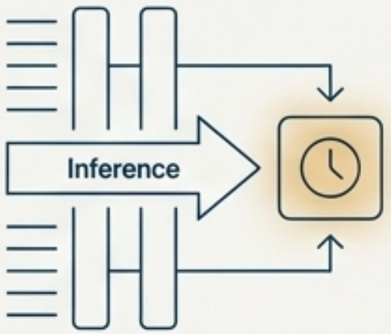
Advanced PEFT: QLoRA

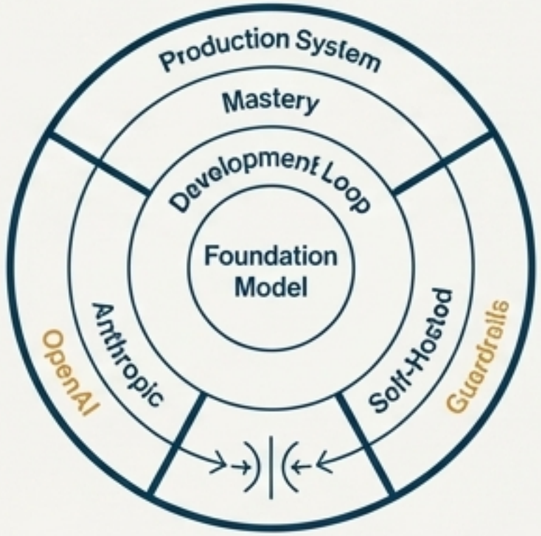
Combines LoRA with aggressive 4-bit quantization and other memory-saving techniques.

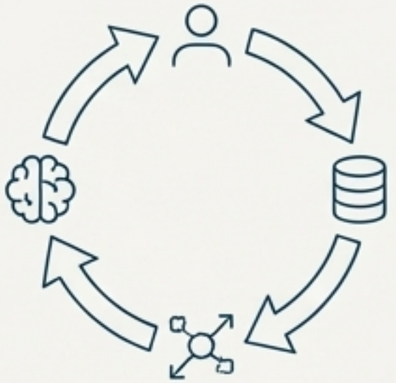
Allows a 65B-parameter model to be finetuned on a single 48 GB GPU.

The Industrialised System: From Model to Production Service

A model is not a product. A production system must serve the model efficiently, reliably, and safely to thousands or millions of users.







Pillar 1: Inference Optimisation

Goal: Make inference faster and cheaper. Latency is a critical user experience factor.

Key Bottleneck: For autoregressive models, decoding is memory bandwidth-bound. Each new token requires reading all model weights from GPU memory.

Crucial Technique: KV Cache. Caching intermediate attention vectors avoids re-computation for every token, drastically speeding up generation.

Pillar 2: System Architecture

Goal: Build a robust, scalable, and secure platform around the model.

Key Component: Model Gateway. A centralised service that routes requests to various models (OpenAI, Anthropic, self-hosted), handling API keys, rate limiting, and logging.

Key Component: Guardrails. Input guardrails scan prompts for PII or injection attacks. Output guardrails check responses for toxicity or hallucinations before they reach the user.

Pillar 3: Feedback Loop

Goal: Enable continuous improvement by systematically capturing and acting upon user interactions.

Key Concept: The most valuable data comes from your own application. A 'data flywheel' leverages user data to create a powerful competitive moat.

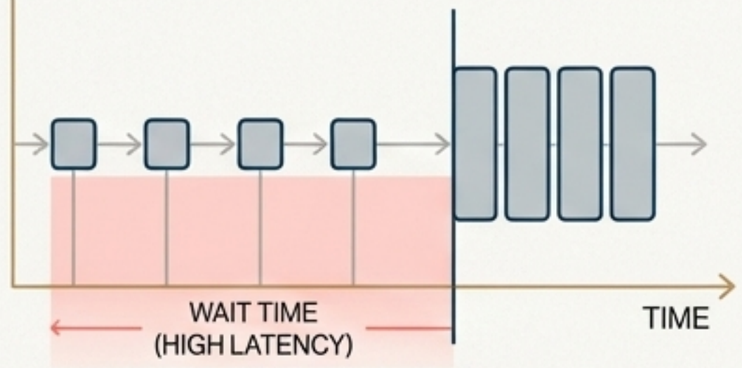
Optimising for Throughput: The Evolution of Batching

Processing requests one by one is inefficient and fails to fully utilise expensive GPU hardware. Batching groups requests to be processed in parallel, dramatically increasing throughput.

Method 1: Static Batching

"The bus waits until it's full."

The server waits for a fixed number of requests before processing.

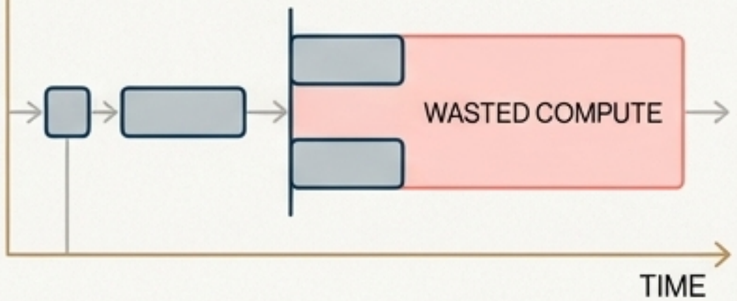


Problem: High latency. The first request is delayed until the last one arrives.

Method 2: Dynamic Batching

'The bus leaves when it's full OR on a schedule."

The server processes when a batch size is met OR a time limit is reached.

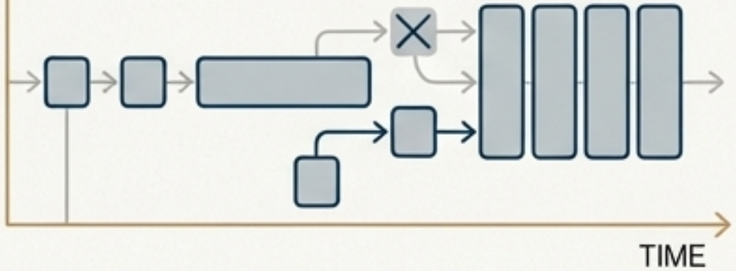


Problem: Inefficient. The entire batch is held up until the *longest* sequence is finished. Compute resources are wasted.

Method 3: Continuous Batching (State-of-the-Art)

"The bus drops off and picks up passengers along the route."

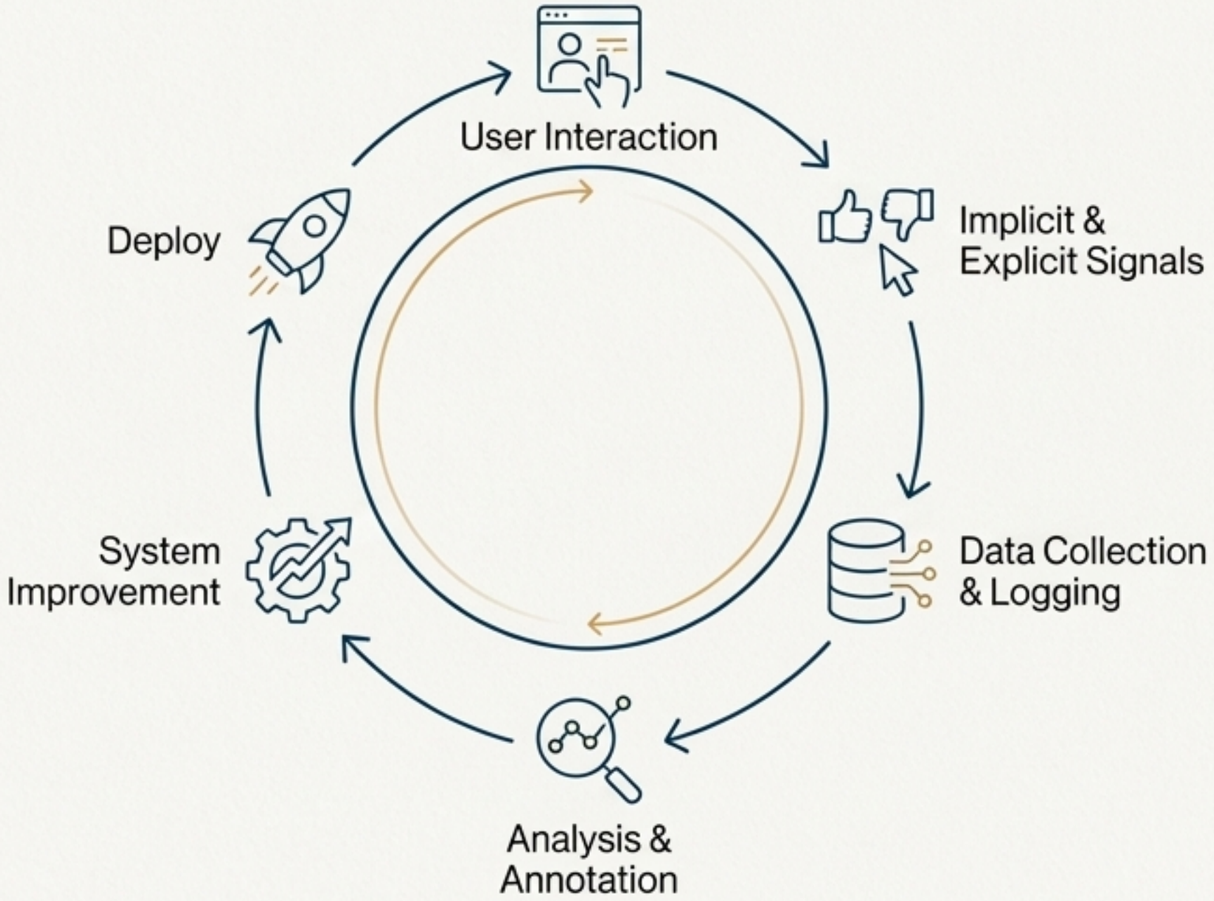
When a request finishes, it is immediately evicted and a new request is added.



Benefit: Eliminates idle time and dramatically improves GPU utilisation and throughput. The standard in modern servers like vLLM.

Closing the Loop: Engineering for Continuous Improvement

The most valuable data comes from your own application. A “data flywheel” leverages data generated by your users to continually improve your product, creating a powerful competitive moat.



Types of Feedback Signals

Explicit Signals

Thumbs up / down

Star ratings

User-written corrections

Choosing one response over another

Implicit Signals

Regeneration: User asks for another response

Copy/Paste: User copies the output (positive signal)

Session Abandonment: User leaves (negative signal)

Editing: User corrects the output (strong signal)

****Warning: Degenerate Feedback Loops.****

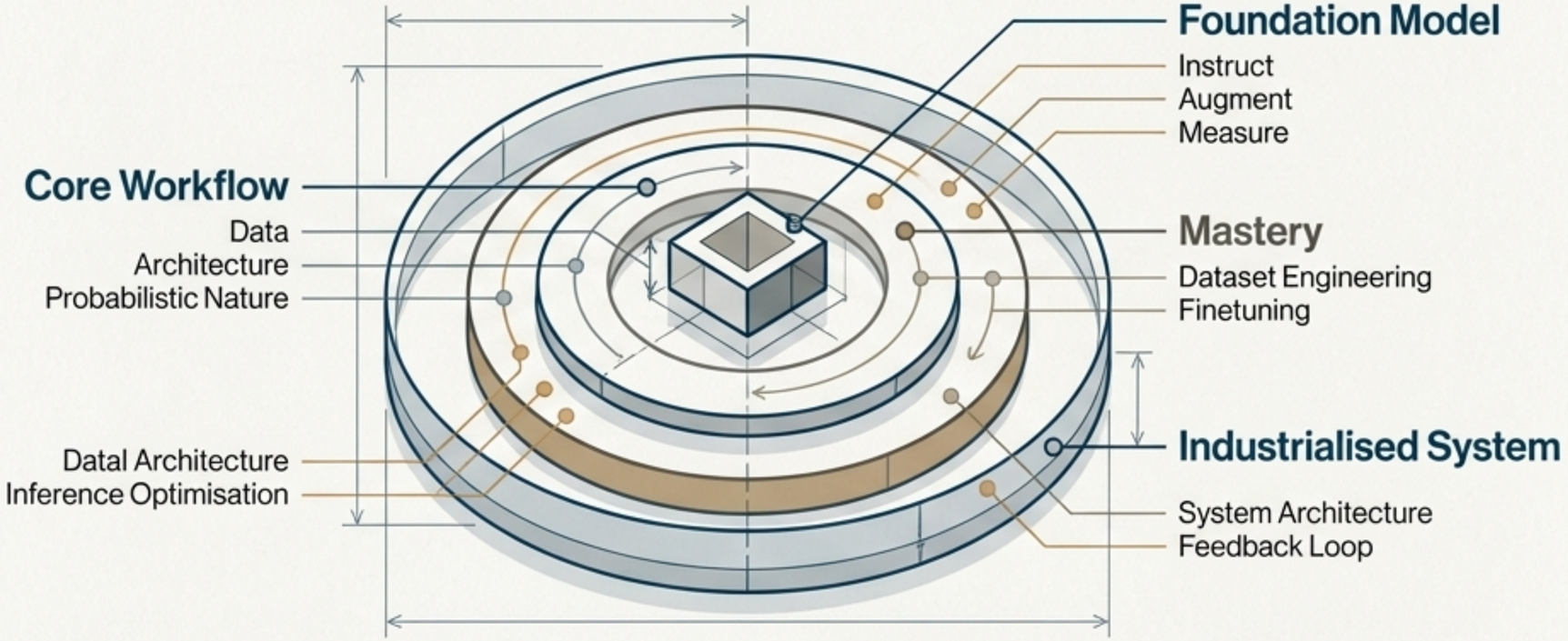
Be aware of feedback amplifying biases. If a model shows users what it thinks they want, it can create filter bubbles and reinforce sycophancy.

The Paradigm Shift: From Traditional ML to AI Engineering

Traditional ML Engineering Neue Haas Grotesk Display Pro	AI Engineering Neue Haas Grotesk Display Pro
Core Unit: Model (Often trained from scratch for a specific task)	**Core Unit: System** (Built around a powerful, pre-trained foundation model)
Primary Data Task: Feature Engineering (Manually crafting numerical inputs from raw data)	**Primary Data Task:** Context Construction (Dynamically providing relevant information via RAG)
Model Improvement: More data, more feature engineering, model retraining	**Model Improvement:** Prompt engineering, finetuning, better retrieval, user feedback loops
Evaluation: Focused on static benchmarks and task-specific metrics (e.g., Accuracy, F1 Score)	**Evaluation:** Multi-faceted, including fluency, factual consistency, instruction-following, safety, and latency
Typical Applications: Classification, regression, prediction on structured data	**Typical Applications:** Generation, summarisation, reasoning on unstructured data

The need for **rigorous evaluation**, **robust system architecture**, **data quality control**, and **deep domain expertise** remains paramount in both disciplines.

The Blueprint for AI Engineering: A Systematic Discipline for Building the Next Generation of Software



Building powerful AI applications is not magic; it is engineering. It requires a structured, multi-layered approach that combines prompt-level creativity with system-level rigour. This blueprint provides the framework for that discipline.

For more resources, visit the book's official GitHub repository: github.com/chiphuyen/aie-book