AI IN PRODUCTION

A practitioner's guide to shipping, governing, and scaling reliable AI systems.

www.getspanforge.com

Foreword

There is a gap in the AI industry that nobody talks about openly, but everyone recognises the moment they encounter it.

On one side of the gap: the rapid advance of AI capability. Models that can reason, write, analyse, classify, recommend, predict, and generate at a scale and quality that would have seemed implausible five years ago. Capability that, by any honest measure, is remarkable.

On the other side: the remarkable difficulty of making any of that capability reliably useful in production.

Most AI projects fail. Not because the models are wrong. Not because the teams are incompetent. Not because the technology is immature. They fail because the infrastructure around the model — the organisational understanding, the production contracts, the team structures, the governance frameworks, the operational disciplines — has not kept pace with the capability it is supposed to deliver.

This book is about that gap.

It is not a book about how to build models. There are many excellent books about building models. It is a book about everything that has to be true before, around, and after the model for the model to matter — what production actually means, not as a technical milestone, but as an ongoing operational contract between an engineering team, a business, and the users it serves.

The chapters that follow move through the full lifecycle of an AI system: from the moment someone proposes that AI might solve a business problem, through scoping, problem definition, team design, responsible AI, and the stages of discover, design, build, govern, and scale. Each chapter is grounded in real patterns — the recurring failure modes and the structures that prevent them — drawn from deployments across industries and scales.

Every chapter ends with a failure mode. The patterns of failure in production AI are not exotic or unpredictable. They are consistent, recognisable, and in most cases entirely preventable. Naming them precisely is the first step to not repeating them.

This book is written for engineers, product managers, technical leaders, and business stakeholders who are investing in AI capability and asking, reasonably, why it is so hard to see a return. It is not a research text. It is a field guide.

The technology will keep advancing. The gap closes only when the practice of deploying it advances too.

That is what this book is for.

Introduction

How to Use This Book

This book is structured around a simple premise: the stages at which AI projects succeed or fail are well understood. The problem is not that teams lack information — it is that they lack a shared map to navigate from idea to reliable production operation. This book provides that map.

The chapters follow the natural sequence of an AI project, but they are written to be read independently. A product manager who needs to understand scoping and problem definition can start with Chapters 02 and 03. A technical leader designing a team structure can start with Chapter 04. A compliance professional can go directly to Chapter 05. A leader trying to understand why their programme is not producing production systems should start with Chapter 06.

The Six Chapters

Chapter 01 — What "Production" Actually Means for AI defines the production readiness criteria every AI system must satisfy before it can be considered genuinely operational. It introduces the production contract framework, distinguishes accuracy from reliability, and provides the full production readiness checklist.

Chapter 02 — Scoping AI Projects for Production Success covers the five questions that must be answered before any AI project begins: problem/solution fit, data availability, build vs buy vs API, ROI framing, and kill criteria.

Chapter 03 — Defining the Right AI Problem: The SpanForge Method introduces the SpanForge Problem Statement Canvas — a thirteen-field structured problem definition tool — and the four decision gates that must be cleared before model development begins.

Chapter 04 — Team Design for AI Engineering examines the structural conditions for production success: the data science island failure pattern, the platform vs embedded team decision, the ML engineer vs data scientist distinction, and the on-call question.

Chapter 05 — Responsible AI: Ethics, Governance & Compliance reframes responsible AI as an engineering discipline. It covers bias and fairness, model cards, the EU AI Act, GDPR's right to explanation, and the regulatory landscape.

Chapter 06 — The AI Production Lifecycle introduces the Discover-Design-Build-Govern-Scale lifecycle model. It examines the four critical handoff breakdowns where most projects fail and the structural changes that address them.

What This Book Is Not

This is not a model development tutorial. It does not teach how to train neural networks, tune hyperparameters, or select feature engineering approaches. It assumes you have, or can develop, those capabilities.

This is not a vendor guide. It does not recommend specific platforms, frameworks, or tools. The patterns it describes are applicable across the technology landscape.

This is not a research text. It draws on patterns observed in production deployments — what actually fails, why it fails, and what prevents it failing again.

A Note on Case Studies

Each chapter includes a case study drawn from real deployments. Where identifying details have been withheld, the case is labelled composite — representative of a pattern observed across multiple organisations, not a single fictional scenario. In all cases, the technical and operational details are accurate to the class of failure or success they illustrate.

> *Production AI is not a harder version of research AI. It is a different discipline. This book is about that discipline.*

Contents

CHAPTER 01

What "Production" Actually Means for AI

SLAs, reliability, and the moving target of good enough.

The Word Nobody Defines

Ask ten people on an AI team what "production" means and you will get ten different answers. To the data scientist, it means the model is no longer running locally. To the ML engineer, it means the model is behind an API endpoint. To the product manager, it means users can access the feature. To the business stakeholder, it means results are showing up in dashboards.

All of these answers are right. None of them is complete. And the gaps between them are where AI projects fail.

"Production" for an AI system is not a single threshold. It is a collection of constraints — technical, operational, business, and human — that the system must satisfy simultaneously, on an ongoing basis, in conditions that were not fully anticipated when the system was designed.

This chapter defines those constraints precisely. The single most common avoidable failure in AI production is the discovery — six months into deployment — that engineering, product, and business were operating with different implicit definitions of what success looked like.

What "Production" Actually Means

A production AI system is one that: delivers reliable predictions across its realistic operating envelope; meets defined service level agreements covering latency, availability, throughput, and quality; fails gracefully when encountering out-of-distribution inputs; can be monitored, diagnosed, and improved; has clear ownership; meets its compliance obligations; and continues performing as conditions change.

Notice what is absent: accuracy on a test set. Test accuracy is an input to production readiness, not an equivalent. A model can be highly accurate on a holdout set and fail every criterion above. Production readiness is about the system, not the model.

The Production Readiness Criteria

Before any AI system is deployed, the following criteria should be defined, agreed, and documented. These are not optional. They are the minimum specification for what "working" means.

1. Business Success Definition

What would this system have to do for us to consider it a success? This is a business question. The answer must be expressed in terms of business metrics — revenue impact, cost reduction, error rate reduction — not model metrics. If you cannot answer this question before you start, the project is not ready to start.

2. Service Level Agreements (SLAs)

An SLA for an AI system defines minimum acceptable performance across four dimensions: Availability (what percentage of requests will the system serve without error?), Latency (maximum response time at which percentile?), Throughput (what request volume at peak load?), and Quality (what is the minimum acceptable prediction quality, and how is it measured?). SLAs are contracts — they must be defined before deployment, not derived from observed behaviour afterward.

3. Accuracy vs Reliability: The Right Tradeoff

Accuracy is an aggregate measure of model performance. Reliability is a property of the system over time. The production mindset insists on a complete characterisation of the model's behaviour across the full input distribution it will encounter in production — including the edges, the outliers, and the conditions underrepresented in training.

4. Stakeholder Expectation Setting

AI systems routinely fail to meet expectations not because they perform poorly, but because the expectations were wrong. Expectation-setting is a production readiness prerequisite. Misaligned expectations create pressure to suppress early warning signals, delay rollbacks, and continue running degraded systems.

The Production Readiness Checklist

If any item below is unanswered, the system is not yet ready for production — regardless of model accuracy.

Criteria	Question	Answer Required
Business success	What business outcome must this system produce?	Written definition, agreed with business stakeholder
Availability SLA	What is the minimum acceptable uptime?	Percentage, agreed with consuming team
Latency SLA	What is the maximum acceptable p95 response time?	Milliseconds or seconds, agreed with product
Proxy signal	What real-time signal will indicate quality is holding?	Metric name, normal range, alert threshold
Quality metric	What outcome metric confirms quality over time?	Metric, measurement window, review trigger
Escalation	At what level does the system fall back?	Defined trigger and fallback behaviour
Monitoring	How will we know if the system is degrading?	Alert definitions and response procedures

Criteria	Question	Answer Required
Ownership	Who is accountable in 12 months?	Named team or individual
Rollback	What is the procedure to revert?	Written runbook
Compliance	What legal/regulatory obligations apply?	Audit, sign-off, or documented exemption

FAILURE MODE

Optimising for accuracy. Ignoring reliability.

Pattern: No Baseline — accuracy fills the vacuum left by an absent production contract.

Fix: Add SLAs, business success definition, failure mode characterisation, and monitoring criteria alongside accuracy.

> *In one line: Production AI = Reliable + Observable + Governed + Owned*

Key Takeaways

- "Production" is not a deployment event — it is an ongoing operational contract.
- SLAs must cover availability, latency, throughput, and quality — all four.
- Accuracy and reliability are different properties. Production systems are evaluated on reliability.
- Stakeholder expectations must be formally set before any AI system goes live.
- Production readiness is not a one-time gate — it is a continuously maintained standard.

CHAPTER 02

Scoping AI Projects for Production Success

The feasibility questions nobody asks until it's too late.

The Moment Most Teams Skip

Every AI project has a scoping phase. What most teams actually do is skip it. The excitement of a new capability compresses the question of should we build this? into an assumption. Of course we should. We've already scheduled the kickoff.

This chapter is about what happens in the gap between "we should build an AI system" and "we are ready to start building." That gap contains five questions that determine whether the project ships — or whether it ships and immediately fails.

> **The five scoping questions:**
>
> *1. Does AI actually solve this problem?*
>
> *2. Does the data we need exist, and can we legally use it?*
>
> *3. Should we build this, buy this, or use an API?*
>
> *4. How will we know if this was worth it?*
>
> *5. At what point do we stop?*

Problem / Solution Fit

The first question before any AI project is not "can we build this?" It is: does AI actually solve the problem we have? AI has genuine advantages in problems involving unstructured data, high-volume pattern recognition, personalisation at scale, or prediction in complex variable-rich environments. It rarely has an advantage in problems that are simply under-tooled or under-resourced.

Before committing to an AI approach, answer three questions explicitly: What happens if we don't build this? Is there a simpler solution that would work? Does AI have a meaningful advantage here?

Data Availability Assessment

Most AI projects that fail do not fail because the model was wrong. They fail because the data was unavailable, insufficient, mislabelled, inaccessible in production, or legally unusable. No data. No system. The data availability assessment must happen before the first line of model code.

Question	What you are actually checking
Does the data exist?	Can you point to a specific dataset or source?
Is it sufficient?	Enough quality and volume to meet the performance threshold?
Is it accessible?	Can the engineering team access it?
Is it representative?	Does it reflect the production distribution, including edge cases?
Is it legally usable?	Does it comply with privacy law and consent obligations?

Build vs Buy vs API

Once problem/solution fit is established and data availability is confirmed, the decision is whether to build internally, license an existing solution, or consume a capability through an external API. This is treated as technical when it is actually strategic.

Factor	Build	Buy	API
Time to first output	Months	Weeks	Days
Upfront cost	High	Medium	Low
Ongoing cost	High (ops, retraining)	Medium (licence)	Variable (usage-based)
Customisation	Full	Limited	Prompt engineering only
Data control	Full	Shared with vendor	Shared with provider
Proprietary advantage	Yes (if data advantage)	No	No

Most organisations default to build because it feels most capable. Many should be choosing API or buy, because the capability is commoditised and the cost of internal ownership is not justified by the differentiation it would produce.

ROI Framing

A sound ROI frame has three components: a named business outcome (a specific metric in a dashboard the sponsor looks at); a measurable baseline (the current value); and a minimum threshold (the improvement that justifies the investment at planned cost). The most common ROI error is estimating build cost only, then discovering that operations cost as much over a three-year horizon.

Kill Criteria

Every AI project should launch with a written answer to: At what point do we stop? Without kill criteria, projects that are failing continue to consume resources because stopping requires a decision, and a decision requires a criterion, and nobody defined

one in advance. The absence of kill criteria is what turns a three-month investigation into a two-year sunk cost.

FAILURE MODE

Start building before validating the data.

Pattern: Wrong Problem — excitement compresses scoping into a formality.

Fix: Five questions. Written answers. Evidence-backed. Signed off before sprint one.

> *In one line: Scoping is not the opposite of speed — it is the prerequisite for it.*

Key Takeaways

- AI project selection should include an explicit problem/solution fit test.
- Data availability assessment must cover existence, sufficiency, representativeness, accessibility, and legal usability — all five, in writing.
- The build vs buy vs API decision should be made against differentiation value, data leverage, operational capacity, and risk.
- A sound ROI case names a specific business outcome, establishes a measurable baseline, and defines a minimum threshold.
- Every project should launch with written kill criteria.
- Legal usability of training data is a scoping gate, not a pre-launch checkbox.

CHAPTER 03

Defining the Right AI Problem: The SpanForge Method

Why most AI projects fail before a single line of code is written.

The Trap That Catches Everyone

There is a moment in almost every AI project where a decision gets made that feels inconsequential at the time. Someone describes a business pain. A leader says "we could build a model for that." A room of people nods. The project is named, resourced, and kicked off.

The pain was real. The model, when built, will almost certainly work. And it will almost certainly solve the wrong problem.

The cost is not felt during the build. It is felt nine months later, when a technically correct model that performs exactly as specified fails to move the business metric it was supposed to move — because the model was specified against the symptom, not the cause. This chapter introduces a structured definition method — the SpanForge Problem Statement Canvas — and the decision gates that prevent teams from entering model development with an unvalidated problem definition.

The SpanForge Problem Statement Canvas (D13)

> **Signature Framework**
>
> *The SpanForge Problem Statement Canvas is the primary tool for defining AI problems before any modelling work begins. It is the boundary condition for entering the Build phase.*
>
> *If you cannot define it, you cannot build it.*

The canvas has thirteen fields organised into four zones: Context, Definition, Signal, and Gate. Working through it takes between one and three hours in a facilitated session.

Field	Zone	Required for Go
1. Business domain and context	Context	Background only
2. Stakeholder	Context	Named individual or role

Field	Zone	Required for Go
3. Frequency and scale	Context	Order-of-magnitude estimate
4. Current state	Definition	Specific, observable description
5. Desired state	Definition	Specific, attributable description
6. The specific gap	Definition	Single sentence: current → desired → outcome
7. Root cause hypothesis	Definition	Plausible mechanism for the gap
8. AI advantage statement	Signal	Named from genuine AI advantage conditions
9. The data signal	Signal	Named sources, access status confirmed
10. The success metric	Signal	Business metric with baseline
11. Failure definition	Signal	Observable conditions for degradation
12. Simpler solution assessment	Gate	Documented and explicitly rejected
13. Decision gate	Gate	Explicit go / no-go with reasoning

Decision Gates Before Modelling

Four gates must all be passed before build begins. Three out of four is failure.

> **Gate 1 — Problem Clarity**
> *Can the team state the problem in one sentence, in business terms, without referencing a model?*
>
> *Gate 2 — Data Existence*
> *Does the data exist, is it accessible, sufficient, and legally usable?*
>
> *Gate 3 — Baseline Existence*
> *Does a measurable baseline for the success metric currently exist?*
>
> *Gate 4 — Ownership*
> *Has a specific named person accepted accountability for this project's outcome?*

The Problem Framing Workshop

The canvas is the output of a structured workshop. Duration: 90 to 180 minutes. Attendees must include: the business stakeholder (the only person who can validate the current state, desired state, and success metric); the ML Engineer or Lead Data Scientist; the Data Engineer; and the Product Manager.

Start with Field 4 (current state), not Field 6 (the gap). Work forward to the desired state without referencing solutions. Apply the five whys to the root cause hypothesis. Close with the simpler solution assessment before the decision gate.

FAILURE MODE

Jump to solution. Skip the problem.

Pattern: Wrong Problem — technically capable teams reach for the most powerful available tool before stating the problem.

Fix: The SpanForge Canvas. Thirteen fields. Four zones. One structured session. The cost is ninety minutes. The cost of skipping it is measured in quarters.

In one line: Define the problem before you define the model. The model is easy. The problem is the work.

Key Takeaways

- Problem vs solution confusion is the most common root cause of AI project failure.
- The SpanForge Canvas forces explicit answers in four zones: Context, Definition, Signal, and Gate.
- Zone 2 (Definition) is critical — the current state, desired state, gap, and root cause determine whether AI is the right instrument.
- Four decision gates must be cleared before model development begins — all four, not three out of four.
- The five whys applied to the root cause is the most reliable method for distinguishing AI problems from process problems.

CHAPTER 04

Team Design for AI Engineering

Roles, structures, and the on-call question nobody wants to answer.

The Island Problem

Ask yourself: in your organisation, who owns a model after it ships? In most companies, the honest answer is: nobody quite knows. The data scientist who built it has moved to the next project. The software engineer who integrated it does not understand the model's failure modes. The platform team monitors infrastructure but not predictions. The model is running. Nobody owns it. And that is already failure.

This is not a personnel failure. It is a structural one — the predictable consequence of designing AI teams around the moment of creation rather than the moment of operation. Most AI team structures are optimised for getting to the demo. Production is treated as what happens after the demo succeeds. It is not. Production is the entire point.

The Data Science Island

The most common AI team failure mode has a name: the data science island. A company hires data scientists who sit together in a centralised team and build models. The models are good. And then the handoff happens.

The model eventually ships — transformed enough by the integration process that the data scientist does not fully recognise it. Nobody on the engineering side understands the model's failure modes. Nobody on the data science side is accountable for the production system. This is hand-off theatre. Work is performed. Artefacts are transferred. Ownership is never established.

Platform vs Embedded Teams

There are two primary structural models for AI engineering teams. The platform model centralises AI capability in a single team. It maintains consistent standards and builds shared infrastructure, but creates distance from the business problem and generates ownership ambiguity at deployment. The embedded model places ML engineers inside product teams alongside software engineers and product managers. Each team owns the full stack of its AI systems.

The organisations that ship reliable AI systems at scale have arrived at a hybrid: a platform team that owns infrastructure, tooling, and standards, combined with embedded ML engineers who own the full product cycle from problem definition to production operation.

Dimension	Platform Team	Embedded Team
Primary output	Infrastructure, tooling, standards	Shipped, operating models
Accountability	Shared capability	Production ownership
Staffing profile	Deep MLOps specialists	ML + software engineering generalists
Works across	Whole organisation	One product area
Failure mode	Creates bottlenecks; owns nothing in production	Inconsistent standards; duplicated infrastructure

ML Engineer vs Data Scientist

The data scientist's primary function is to investigate problems in data, develop models, and produce evidence that a proposed approach works. The data scientist is optimised for exploration. The ML engineer's primary function is to take a validated approach and turn it into a production system — reliable, observable, maintainable, and performant. The ML engineer is optimised for operation.

One proves it works. One makes it work. Teams without both, in the right proportions at the right stages, will have gaps.

On-Call for Models

The on-call question is the most honest test of whether an organisation has built a production AI team or a model development team that happens to deploy things. An AI on-call rotation must cover two dimensions: infrastructure health (the system is running and serving requests) and model quality health (predictions are meeting the quality thresholds defined in the production contract).

This requires: a quality signal that can be monitored in near-real-time; a runbook for quality degradation; a defined escalation path; and named ownership. Not a team. Not a channel. A name, and a date.

Hiring for the Production Mindset

The production mindset places reliability, observability, and operational accountability alongside model performance as first-class concerns. Test for it in interview: ask about production failures (not test-set failures); ask who owns the model post-deployment; ask about times they chose a simpler model or delayed a deployment; ask how they would detect a model degrading six months after deployment.

> **FAILURE MODE**
>
> **Build the model. Throw it over the wall. Call it shipped.**

Pattern: Adoption Failure — structural separation between model builders and system operators.

Fix: Embedded ownership, from problem definition to production accountability, is the prerequisite for a system that earns adoption.

In one line: Production AI is a team sport. Organise for the finish line, not the demo.

Key Takeaways

- The data science island is the most common structural cause of AI systems failing to reach or survive production.
- Platform and embedded models are complementary — platform owns infrastructure, embedded teams own production models.
- ML engineers and data scientists are different roles — both required, neither sufficient alone.
- On-call for AI systems covers infrastructure health and model quality health — both require defined signals, runbooks, and named owners.
- Hiring for the production mindset means testing for evidence of a mental map between "model is built" and "model is healthy in production".
- Answer the on-call question before deployment. It is easier then.

CHAPTER 05

Responsible AI: Ethics, Governance & Compliance

Not a checkbox — a production constraint.

The Checkbox Trap

Every organisation building AI systems has a moment where responsible AI enters the conversation. In most organisations, it enters late. The deployment review is scheduled. A question appears on the approval form: has the ethical review been completed? Someone marks yes. The system ships.

That is not responsible AI. That is responsible AI theatre. A form that asks whether ethical review has occurred — without specifying what it should examine, who is qualified to conduct it, or what findings would result in a changed design — produces documentation. Not protection.

Real responsible AI lives in the architecture decisions made at problem definition, the evaluation choices made during model development, the deployment gates applied before launch, and the monitoring systems that run in production. Ethics is not a department. It is a practice.

Bias and Fairness: The Engineering View

Bias in AI systems is not a social failure. It is a technical failure with social consequences. Understanding it as a technical failure makes it tractable — bias can be measured, characterised, and in many cases reduced, if the work happens at the right points in the development lifecycle.

Where Bias Enters

Bias enters AI systems at multiple points: data collection bias (historical decisions reflecting historical discrimination); sampling bias (underrepresented populations receiving worse predictions); label bias (human annotators' implicit assumptions); proxy bias (protected attributes reconstructed through correlated features); and evaluation bias (aggregate metrics masking slice-level failures).

Fairness Metrics: The Incompatibility Problem

There is no single definition of fairness correct for all problems. Several formal fairness metrics exist, and they are mathematically incompatible — it is impossible to satisfy more than one simultaneously when base rates differ across groups.

Metric	Definition	What it prioritises
Demographic parity	Equal positive prediction rates across groups	Equal representation in outcomes
Equal opportunity	Equal true positive rates across groups	Equal benefit for those who qualify
Predictive parity	Equal precision across groups	Equal reliability of positive predictions
Individual fairness	Similar individuals receive similar predictions	Consistency at the individual level
Counterfactual fairness	Prediction unchanged if protected attribute changed	Absence of direct causal discrimination

You are not choosing a metric. You are choosing a value system.

Model Cards

A model card is a standardised documentation artefact describing a model — its intended use, performance characteristics, limitations, and ethical considerations — in a form readable by technical and non-technical stakeholders. A model card that omits slice performance is incomplete by definition. Aggregate performance alone is not a responsible AI disclosure.

> **Model Card: Minimum Viable Structure**
>
> *Model details — Name, version, type, training date, owning team*
>
> *Intended use — Primary use cases and explicit out-of-scope uses*
>
> *Training data — Sources, time period, known limitations*
>
> *Aggregate performance — Primary metrics on test set*
>
> *Slice performance — Metrics disaggregated by demographic and contextual groups*
>
> *Limitations — Known poor-performance conditions and failure modes*
>
> *Ethical considerations — Potential harms, affected groups, fairness metric choice, mitigations*
>
> *Monitoring recommendations — Signals, thresholds, review cadence*

The Regulatory Landscape

EU AI Act

The EU AI Act (in force 2024) introduces a risk-based classification system. High-risk AI includes systems used in employment decisions, credit scoring, healthcare, education, law enforcement, and critical infrastructure. High-risk systems must comply with conformity assessment, technical documentation, human oversight mechanisms, and EU database registration. Teams deploying AI in an EU context — or AI that affects EU residents — must assess whether their system falls in the high-risk category, and design for compliance from the beginning of the project.

GDPR and the Right to Explanation

GDPR Article 22 gives individuals the right not to be subject to solely automated decisions with significant effects. Where automated decisions are permitted, organisations must be able to explain them at the prediction level, not just the model level. Explainability is therefore a regulatory requirement, not a nice-to-have. It must be architected in, not added later.

HIPAA and Health Data

AI systems involving protected health information are subject to HIPAA's Privacy and Security Rules. The consent scope of health data is a scoping gate, not a deployment checkbox. The operational handling of PHI throughout the model lifecycle — training, validation, serving, logging, monitoring — must comply with the Security Rule's technical safeguards.

FAILURE MODE

Build. Evaluate. Ship. Run the ethics review later.

Pattern: Deployment Shock — ethical review deferred until architecture is fixed.

Fix: Earlier questions, not later reviews. Problem definition includes impact assessment. Evaluation criteria include demographic slice thresholds. Deployment gates include bias audit sign-off.

In one line: Ethics is not the last door before shipping — it is the first constraint in the design.

Key Takeaways

- Responsible AI is an engineering discipline, not a compliance activity.
- Bias enters AI systems at multiple points — data collection, sampling, labelling, proxy features, and evaluation. Each must be explicitly examined.
- Fairness metrics are mathematically incompatible; the choice is an ethical and contextual decision that must be documented.
- Slice-level evaluation is the minimum standard — aggregate metrics alone are not sufficient.
- Model cards must include slice-level performance, known failure modes, and ethical considerations.
- The EU AI Act imposes concrete compliance obligations on high-risk AI systems — compliance is a design constraint, not a retrospective certification.
- GDPR's right to explanation creates a technical requirement for prediction-level explainability in automated decision systems.

CHAPTER 06

The AI Production Lifecycle

Discover · Design · Build · Govern · Scale — where projects fail at each stage.

The Map Most Teams Are Missing

Ask an AI team where they are in their project and you will get a confident answer. Ask two teams in the same organisation and you will get different answers — described in different terms, with different implied definitions of what "done" means.

Most AI teams operate without a shared lifecycle model. Data scientists think in experiments. Engineers think in sprints. Product managers think in features. Leaders think in quarters. None of these map cleanly onto the actual stages through which an AI system must travel to reach reliable production operation. The Discover-Design-Build-Govern-Scale lifecycle is that map.

> *DISCOVER → DESIGN → BUILD → GOVERN → SCALESkip a stage, and you pay for it later.*

Stage 1 — Discover

Primary question: Is there a real problem here, and is AI the right instrument to address it?

Discover is the stage most teams spend the least time on and most regret shortchanging. Its output is not a working model — it is clarity: a documented understanding of the problem, the data, the business case, and the conditions under which the project should proceed. Required outputs include: validated problem definition (SpanForge Canvas), data availability assessment, problem/solution fit determination, business case with named outcome and ROI threshold, kill criteria, and go/no-go decision with documented reasoning.

Discover failure mode: teams treat it as a kickoff — a brief alignment meeting before work begins. You cannot discover your way out of a problem you define during Build.

Stage 2 — Design

Primary question: How should this system be built, and what does "working" mean?

Design is the stage of architecture, evaluation criteria, production contract definition, and responsible AI assessment. Its outputs are the decisions that, once committed in Build, become expensive to reverse. Required outputs: production readiness contract (SLAs, business success definition, monitoring plan), system architecture with fallback and escalation paths, evaluation criteria including slice-level thresholds,

responsible AI assessment, build vs buy vs API decision, named production owner, and data governance sign-off.

Design failure mode: teams begin model training before evaluation criteria are defined. Build first, then decide what success looks like — that is drift with good intentions.

Stage 3 — Build

Primary question: Can this system be built to the specification the Design stage produced?

Build is where the outputs of Discover and Design are most sorely tested. Data assumed to exist proves incomplete. Features planned prove unavailable at inference time. The question is whether the team has the artefacts from earlier stages to navigate that friction. Build failure mode: most organisations are stuck in Build — not because they cannot build, but because they never finished deciding. Build expands to fill available time when earlier stages were shallow.

Stage 4 — Govern

Primary question: Is this system safe, accountable, and compliant to operate in production?

Govern is the most commonly skipped, compressed, or rubber-stamped stage. It is the point at which responsible AI assessments are verified, compliance is confirmed, production readiness is evaluated, and the deployment decision is made with explicit, documented reasoning. Govern failure mode: the stage becomes a form, not a review. Governance that produces compliance theatre is worse than no governance — it provides false assurance.

Stage 5 — Scale

Primary question: Can this system be operated reliably, improved continuously, and extended sustainably?

Scale is not about growing traffic. It is about operating the system over time — maintaining quality, managing cost, incorporating improvements. Most teams do not plan for Scale. Deployment is an event. Scale is indefinite. Scale failure mode: the system deploys and is forgotten. Monitoring exists but nobody reviews it. Retraining is deferred. The model doesn't break. It just slowly stops working.

The Handoff Breakdowns

Projects don't fail in stages. They fail between them. Handoff breakdowns occur when a project transitions without required outputs in a documented, agreed, and transferable state.

Handoff	How it fails
Discover → Design	Problem definition lives in people's heads. Data not confirmed. Business case not agreed in writing.
Design → Build	Evaluation criteria too abstract to implement. Fairness requirements stated as principles, not measurable criteria.
Build → Govern	Model card incomplete. Monitoring not set up during Build. Compliance requirements deferred and now structural.
Govern → Scale	Runbook written under pressure. On-call rotation named but not operationally tested. Retraining triggers not implemented.

Why Most Organisations Are Stuck in Build

Most organisations are structurally optimised for Build — they hire for Build skills, fund Build activities, and measure Build outputs. Discover and Design are treated as overhead. Govern is a gate to navigate. Scale is what happens automatically. Three structural changes break this pattern: define stage exit criteria as organisational policy (not guidelines — requirements); invest in Govern and Scale capability as first-class engineering; track lifecycle health metrics alongside project delivery metrics.

FAILURE MODE

Build without a lifecycle. Wonder why nothing ships.

Pattern: Wrong Problem + Deployment Shock — no shared map means no stage gates and no required outputs.

Fix: A shared lifecycle model. And the discipline to use it.

> *In one line: You cannot ship what you cannot navigate — and you cannot navigate without a map.*

Key Takeaways

- The Discover-Design-Build-Govern-Scale lifecycle is a shared mental model that gives teams a common vocabulary for where a project is and what the current stage requires.
- Each stage has required outputs that must exist as documented, agreed artefacts before the next stage begins.
- The four critical handoffs are where most projects fail — breakdowns occur when outputs are informal rather than documented.
- Most organisations are structurally stuck in Build because Discover and Design were shallow and Govern was never formally defined.

– Three structural changes break the Build trap: exit criteria as policy, Govern and Scale investment, and lifecycle health metrics.