ARCHITECTURE POSITION PAPER

DETERMINISTIC AND PROBABILISTIC DECISIONING

DESIGNING DEPENDABLE AI-ENABLED SERVICES

ARCHITECTURAL PRINCIPLES AND
DECISION FRAMEWORK



VERSION
1.0

DATE
SEPTEMBER 2026

STATUS
DISCUSSION PAPER

PAUL R WILLIAMS
ENTERPRISE ARCHITECTURE | AI | GOVERNANCE

www.modern-architecture.co.uk
www.paulrwilliams.co.uk

Table of contents

Table of contents 1
Document Control 2
Introduction 5
Executive summary 7
The distinction that matters 10
A decision framework for Architects 13
Work the machine should normally do 16
Work where AI earns its place 19
The hybrid pattern across real services 23
Reference architecture: bound the uncertainty 27
The anti-pattern catalogue 30
Worked example: Financial Assurance 33
Testing two kinds of system 37
Cost, latency and operational control 41
A practical review method 45
Conclusion: Intelligence through restraint 49
Appendix A: Applying the pattern to Microsoft 365, Power Platform and Azure 50
Appendix B: Sources and further reading 55

Document Control

Document Control Table

Parameter	Value
Document reference	AIARCH-001
Document title	Deterministic and Probabilistic Decisioning
Author	Paul Williams
Status	Discussion Paper
Version	1.0
Date	08 September 2026

Revision History

Revision / Version	Author	Note
0.1	P Williams	Initial Draft : Discussion Paper
0.2	P Williams	Initial Review : Grammar and formatting changes accepted
1.0	P Williams	Final : Portfolio Edition

Reader Guide

This is an architecture position paper examining how work should be allocated between deterministic software, probabilistic models and accountable human decision-makers.

It is intended to support design discussion rather than prescribe a single implementation pattern. Its central position is that AI should be used where inference creates genuine value, while established software techniques should normally retain responsibilities requiring precision, repeatability, security or authoritative control.

Intended audience

The paper is written for solution and enterprise architects, engineers, product owners, delivery leaders, assurance specialists and governance practitioners involved in designing or reviewing AI-enabled services.

How to interpret the recommendations

The principles in this paper are defaults, not absolute rules. They should be applied in proportion to the consequence, scale, volatility and expected lifespan of the service.

A reversible, low-volume experiment does not require the same controls as a payment decision, employment process, public-service entitlement or clinical system. The appropriate objective is the simplest architecture that can test or deliver the capability safely.

Terms such as “should” express an architectural recommendation unless the surrounding text identifies a specific legal, regulatory or contractual obligation.

Terminology

Deterministic describes processing governed by explicit logic which, for the same controlled inputs, configuration and software version, should produce the same result. It does not mean that the logic is necessarily correct.

Probabilistic describes processing that estimates, classifies, predicts or generates an output from learned patterns. Its quality is established statistically rather than by expecting every individual output to be identical.

Hybrid describes a design in which deterministic software, probabilistic models and human judgement perform distinct responsibilities within the same service.

Decisioning refers to the complete process through which facts, rules, model observations and human judgement contribute to an outcome.

An observation identifies something that may be relevant. A recommendation proposes an action. A decision authorises or determines an outcome. These terms are not interchangeable.

Architecture conventions

The boundaries shown in this paper are primarily logical boundaries. They do not imply that every component must become a separately deployed service.

A parser, rules evaluator, model adapter, workflow component and report generator may exist within a single modular application. Physical separation should be introduced only where independent ownership, scaling, security, deployment or resilience justifies it.

Likewise, structured output, grammar constraints, tool calling and deterministic validation can make model-based processing suitable for bounded tasks. These controls reduce particular failure modes but do not prove that a model’s interpretation is semantically correct.

Examples and applicability

Examples are illustrative rather than exhaustive. They demonstrate how the principles may apply across different domains but do not establish that every organisation or use case is subject to the same controls.

The Financial Assurance example describes a pre-screening and reviewer-support capability. It should not be interpreted as automated certification, a determination of misconduct or a replacement for accountable review.

Regional context

The legal and governance discussion is written primarily for a UK audience. It draws on UK legislation-facing guidance, regulators and government assurance material current at the stated publication date.

This paper is not legal, regulatory, financial or clinical advice. Teams should verify current requirements and obtain appropriate specialist advice for their activities and circumstances.

Introduction

Enterprise decision systems are rarely entirely deterministic or probabilistic. They combine explicit rules and calculations, inference from incomplete or ambiguous information, and accountable human judgement.

The architectural challenge is not to choose between “AI” and “no AI”. It is to assign each responsibility to the capability best suited to it, and then combine those capabilities through explicit contracts, proportionate controls and clear accountability.

The Position

Generative AI offers extraordinary capabilities in language, synthesis and interpretation. It is not a universal automation runtime. Yet many early proofs of concept ask a language model to parse known formats, perform arithmetic, reconstruct established policy rules, control workflow, query systems of record and generate structured reports in a single prompt.

The demonstration may appear intelligent, but the resulting system is often slower, more expensive, harder to test and less defensible than it needs to be.

The design question is therefore not simply, “Can AI do this?” It is, “Which parts of this task genuinely require probabilistic inference, and which should be implemented as controlled software?”

That change in framing marks the difference between an impressive demonstration and an operable enterprise service.

One sentence to remember

> *If an outcome can be expressed as an exact rule, calculation, schema or state transition, deterministic software should be the default starting point - not another prompt.*

This is a design default, not an absolute requirement. For a low-volume, reversible or short-lived task, a constrained model with validation may be the simplest safe and economical choice. The case for deterministic implementation strengthens as potential consequences and volume increase, and as requirements for repeatability, performance and longevity become more demanding.

This paper does not argue for removing AI. It argues for using AI where it creates distinctive value, while surrounding it with systems that constrain, validate and record what happens.

The objective is not maximum determinism or maximum AI adoption. It is the simplest defensible architecture for the required outcome.

Executive summary

The rush to ‘add AI’ has created a recurring architectural anti-pattern: the model becomes parser, rules engine, calculator, database, workflow engine and report writer. This places probabilistic interpretation across tasks that may already have known semantics and concentrates uncertainty where organisations most need control, repeatability and evidence.

The alternative is not ‘no AI’, nor does it require an elaborate enterprise platform. The answer is a deliberate division of labour: deterministic software handles stable and explicit logic; probabilistic models address ambiguity and inference; and people exercise accountable judgement where evidence is incomplete, values conflict or the consequences warrant it.

The resulting architecture and its controls should be proportionate to the consequence, scale, volatility and expected service life of the use case.

Key conclusions

- **Deterministic does not mean correct.**
  It means that, for controlled inputs and a known configuration and version, behaviour can be reproduced and tested. A badly written rule is still wrong, but it is inspectable, version-able and fixable.

- **Probabilistic does not mean careless.**
  Statistical and generative models can perform tasks that conventional rules cannot handle economically, particularly where language, images, patterns or incomplete evidence require interpretation. Their suitability must be demonstrated against a defined use case and appropriate performance thresholds.

- **Constraints reduce uncertainty; they do not remove it.**
  Lower temperature settings, structured outputs, tool calling, schema constraints and deterministic validation can make a model more reliable for bounded tasks. They do not guarantee that its interpretation is semantically correct or transform it into a rules engine. Generative models still produce outputs from learned probability distributions.

- **The potential consequence of error should determine the control.**
  A reversible drafting aid may tolerate correction after generation. A payment release, access-control decision or safety-critical calculation requires a much stronger deterministic boundary. Where human review is required, it must be meaningful: the reviewer needs sufficient information, expertise, time and authority to challenge the output rather than merely approve it.

- **The strongest enterprise pattern is usually hybrid.**

Deterministic components can handle known-format ingestion, validation, calculation, identity, authorisation, state transitions, policy gates and audit. Probabilistic components can perform bounded classification, interpretation or explanation. People can resolve exceptions and decisions requiring material judgement.

- **Architectural separation does not automatically mean microservices.**
  These are logical responsibilities and control boundaries. A well-structured modular application may be entirely appropriate until scale, ownership, security or operational requirements justify independently deployed services.

- **Rigour should be proportionate.**
  A low-volume, reversible or short-lived internal experiment may justify a constrained model with validation. The case for engineered deterministic components and stronger assurance increases with consequence, volume, repeatability, performance requirements and expected service life.

- **Economics must be assessed across the lifecycle.**
  Repeatedly asking a model to rediscover a stable schema, calculation or policy at scale is architectural waste. Conversely, maintaining bespoke rules for a volatile, infrequent and low-risk task may cost more than using a model with suitable validation. The comparison should include delivery, maintenance, assurance, compute, latency, correction and failure costs-not token consumption alone.

Specific Context

This paper has been written specifically for UK-based organisations. For enterprises, choosing AI does not displace the legal, regulatory and professional obligations attached to the activity. Depending on the use case, the architecture may need to support data protection, equality, public law, employment, procurement, information governance, security and sector-specific duties.

The UK Government's AI guidance emphasises understanding limitations, using the right tool for the job, maintaining meaningful human control at the right stages and managing the complete lifecycle . Its AI assurance guidance also advocates an outcomes-based and proportionate approach, with the strength of assurance reflecting the context and risk of the system .

Where personal data contributes to significant decisions made solely through automated processing, the design must address applicable UK data-protection safeguards, including transparency, representations, human intervention and the ability to contest the decision. The required controls should therefore be determined by the activity, data and potential effect on people - not simply by whether the technology carries an 'AI' label.

The architectural test

Question	What it indicates	Useful starting point
Can the outcome be completely and stably specified as rules or mathematics?	Known logic	Deterministic software
Must the same controlled input, configuration and version produce the same result?	Strong reproducibility requirement	Deterministic software
Does the task involve exact calculation, identity, authorisation, release of funds or an irreversible action?	Consequential execution	Deterministic core, with independent authorisation or approval where required
Does the task depend on language, images, intent, patterns or incomplete context?	Genuine inference	Evaluated probabilistic model
Can a model observation be validated before it affects an outcome?	Bounded uncertainty	Hybrid pattern
Is the task low-volume, reversible or short-lived?	Limited exposure and service life	A constrained model with validation may be proportionate
Are the rules volatile, difficult to specify or expensive to maintain?	Material lifecycle trade-off	Compare model-and-validation and coded-rule approaches empirically
Would competent people reasonably disagree because evidence is incomplete or values must be balanced?	Accountable judgement	Probabilistic assistance with human decision-making

The test is a starting point, not a substitute for use-case risk assessment.

Several answers may apply to the same process. That is usually evidence for a hybrid design, not a failure of classification.

This test is a starting point. It supports, but does not replace, use-case risk assessment, legal and regulatory assessment, user research, technical evaluation and lifecycle-cost analysis.

The Distinction that Matters

The useful distinction is not between systems that contain AI and systems that do not. It is between responsibilities that can be specified, responsibilities that must be inferred, and decisions that require accountable human judgement.

Deterministic processing

A deterministic component applies explicit logic to defined inputs. Given the same controlled inputs, relevant state, configuration, operating conditions and software version, it should produce the same result.

Examples include validating a date against a specified format, calculating VAT once the applicable treatment and rate have been established, comparing an amount with a threshold, parsing a documented record format, enforcing an API authorisation policy, controlling a state transition and rendering a report from a fixed template.

Its strength is not intelligence but controllability. Unit and property-based tests can exercise known conditions; rule versions can be reviewed and approved; calculations can be reconciled; and failures can be reproduced and investigated.

Deterministic does not mean correct. A rule may be badly specified, incorrectly implemented or no longer reflect the policy it represents. Determinism makes that behaviour inspectable and repeatable; it does not make it true.

Its principal limitation is that it only handles conditions represented in its logic. Novel, incomplete or ambiguous inputs must be rejected, handled as exceptions or passed to another capability. Attempting to encode every possible interpretation as rules can itself create a brittle and expensive system.

Probabilistic processing

A probabilistic component estimates, classifies or generates an answer from patterns in data rather than deriving it entirely from explicit, human-authored rules.

This includes conventional machine learning, such as classification, forecasting, anomaly detection and computer vision as well as foundation models that generate language, images or code.

Probabilistic does not necessarily mean operationally variable. A fixed classification model may return exactly the same score whenever it receives the same input. The result remains probabilistic because it represents a learned statistical estimate rather than a conclusion guaranteed by formal logic. Generative models may introduce the additional possibility of variation between executions. The important distinction is therefore the basis of the answer, not merely whether two executions produce identical output.

Performance should be evaluated statistically across representative cases using measures appropriate to the task, such as precision, recall, calibration, error severity, robustness and failure rates. A handful of successful demonstrations does not establish production fitness.

Probabilistic processing is valuable where exact rules are unavailable, unstable or uneconomic: interpreting the intent of an email, locating semantically similar cases, estimating demand, detecting unusual payment behaviour, reading a damaged invoice or explaining a complex result in accessible language.

Human judgment

Human decisions are not deterministic either. People interpret evidence differently, overlook information and exhibit individual or systemic bias. Their distinctive contribution is the ability to consider context, values, exceptions and consequences while exercising delegated authority and accepting responsibility for a decision.

'Put a human in the loop' is therefore not a complete control. Human involvement is meaningful only when the reviewer has:

- sufficient competence to understand and challenge the output;
- access to the relevant evidence and explanation;
- adequate time and attention;
- authority to reject or change the proposed outcome; and
- a clear route for recording reasons and escalating uncertainty.

Human approval should not be inserted indiscriminately into every process. Excessive review creates delay, overload and the risk of automation bias or rubber-stamping. It should be placed where the nature or consequence of the decision makes accountable judgement necessary.

This distinction has particular relevance in the UK. The ICO's summary of the Data (Use and Access) Act 2025 describes a decision as solely automated where there is no meaningful human involvement and identifies safeguards for significant automated decisions involving personal information.

The ICO and Alan Turing Institute guidance also recommends selecting explanations according to the domain, use case and effect on the individual, and designing systems so that the information needed to support those explanations can be extracted and communicated.

The false binary

Enterprise architecture rarely requires one decisioning mode to be selected for an entire process. A single business journey may contain exact calculations, ambiguous classifications, policy choices and accountable human decisions.

The architect's task is to assign each responsibility deliberately, define the contracts between them and prevent uncertainty from leaking into components that require precision.

These separations are logical rather than necessarily physical. They may be implemented as modules within one application, as components of a workflow or, where scale, security, ownership or operational requirements justify it, as independently deployed services.

Property	Deterministic	Probabilistic	Human judgment
Primary contribution	Precision and repeatability	Interpretation and prediction	Context, accountability and discretion
Basis of result	Explicit rules, calculations and state	Learned statistical relationships	Evidence, experience, policy, values and delegated authority
Repeatability	Expected under controlled conditions	May be repeatable, but correctness remains statistical	May vary between people and over time
Typical assurance	Unit, property, contract and reconciliation testing	Representative evaluations, calibration, robustness, bias and drift monitoring	Competence, decision review, sampling, escalation and appeal
Change control	Code, rule, configuration and dependency versions	Model, prompt, data, tool, threshold and evaluation versions	Policy, delegation, training and decision guidance
Characteristic failure	Incorrect or obsolete specification; omitted condition	Misclassification, hallucination, bias, drift or false confidence	Inconsistency, bias, fatigue, overload or automation bias

No mode is inherently superior. A defensible architecture uses each where its strengths create value and where its characteristic failures can be detected, controlled and tolerated.

A Decision Framework for Architects

Classify each task before selecting the technology. A workflow labelled as an ‘AI use case’ may contain ten separate decisions, each with different technical characteristics and assurance requirements.

The framework should therefore be applied to individual responsibilities and decision points, not to the application as a whole.

Start with the nature of the answer

- **Specified outcome:**
  The answer follows from a schema, equation, lookup, policy threshold, validation constraint or state transition. Prefer deterministic processing.

- **Estimated outcome:**
  The answer is a prediction, score or pattern inferred from historical data. Use a statistical model with task-appropriate evaluation, calibration where relevant, and ongoing monitoring.

- **Interpreted meaning:**
  The answer depends on language, images, intent or contextual ambiguity. Use an appropriate language, vision or multimodal model, grounded against controlled evidence where possible, with structured outputs, validation and an explicit exception or review path. Do not rely on a model’s self-reported confidence unless that confidence has been independently evaluated and calibrated.

- **Normative judgement:**
  The answer determines what ought to happen, particularly where the outcome materially affects a person. A model may organise evidence, identify options or support explanation, but policy ownership, lawful authority and accountability should remain outside the model. Once an approved policy has been expressed as explicit rules, its routine application may be deterministic.

Seven Tests to Apply

The next step is to apply the following, seven tests;

Test	Design implication	Warning signal
Reproducibility	Use versioned logic and retain the inputs, state and configuration needed to reproduce outcomes that must replay exactly.	A result must be reconstructed for audit or challenge, but the applicable version or processing state is unknown.

Precision	Keep money, quantities, dates, identifiers and exact calculations in typed and validated software. AI may extract a candidate value, but deterministic code should validate and normalise it.	A prompt asks the model to calculate, compare or preserve exact values without independent validation.
Ambiguity	Use AI where rules cannot capture meaning efficiently. Remove known structure first and present only the residual ambiguity to the model.	A model repeatedly interprets a format that already has a documented schema or fixed mapping.
Consequence	Increase validation, authorisation, fallback and recovery controls with potential impact and irreversibility. Separate recommendation from execution where appropriate.	Model output directly releases payment, denies a service, grants access or changes an authoritative record.
Explainability	Record source evidence, derived facts, rule identifiers and versions, model and configuration versions, model output, validation results and accountable decisions separately.	Only a free-text model explanation survives, or that explanation is treated as a faithful account of the model's internal reasoning.
Adversarial exposure	Treat documents, messages, webpages and other external content as untrusted data, never as authority. Separate instructions from content, restrict permissions and tools, and validate proposed actions.	Content being processed can alter the model's instructions, bypass controls or cause it to invoke capabilities beyond the task.
Economics	Measure total lifecycle cost, including implementation, integration, latency, tokens, retries, correction, human review, monitoring, maintenance and failure.	AI repeats stable work on every request, or bespoke software is built for a rare and volatile task without an economic case.

The tests are not independent pass-or-fail gates. They expose competing design pressures. A task may involve substantial ambiguity and therefore benefit from AI, while its consequence requires the model to remain advisory. Another task may be difficult to perform with rules but easy to verify, making model inference followed by deterministic validation an effective hybrid pattern.

UK government guidance supports this task-level approach. The Artificial Intelligence Playbook for the UK Government emphasises understanding AI's limitations, using the right tool for the job, retaining meaningful human control at the appropriate stages and managing the complete lifecycle. The UK Government's AI assurance guidance also recommends proportionate assurance based on the context and risk of the system.

The approach is consistent with the NIST AI Risk Management Framework, which structures AI risk management around governing, mapping, measuring and managing risk across the lifecycle. NIST

provides useful international alignment, but it should complement rather than replace the applicable UK legal, regulatory and assurance context.

A successful demonstration shows that a concept can work. It does not, by itself, demonstrate that the resulting service is secure, reliable, lawful, economical or operable at scale.

NIST frames AI risk management around governing, mapping, measuring and managing risk across the lifecycle-not around treating a successful demonstration as sufficient evidence.

A useful escalation rule

As potential consequence, irreversibility or adversarial exposure increases, unconstrained model autonomy should normally decrease unless the organisation can demonstrate correspondingly stronger evidence, monitoring and controls.

This does not mean that every consequential process requires manual approval. Deterministic validation, restricted permissions, segregation of duties, exception-based review and reliable rollback may provide stronger protection than routine human approval.

The relevant question is not whether AI can be trusted in general. It is whether the proposed degree of autonomy is justified by evidence and supported by controls appropriate to that particular use.

Work the Machine Should Normally Do

The following are not ‘anti-AI’ examples. They are tasks for which the required behaviour can usually be specified and where conventional engineering provides a stronger control surface.

In some cases, AI may help interpret an input, identify candidate facts or propose a rule. Once the required outcome has been defined, however, its calculation, validation or execution should normally be entrusted to controlled software.

Domain	Task	Why deterministic processing leads
Finance	Calculate totals, VAT, currency conversions using an approved rate, rounding and threshold comparisons.	Exact arithmetic, declared inputs and reconciliation are required
Finance	Match a transaction to an approved account, supplier or cost centre.	A governed lookup has a defined answer. Fuzzy entity resolution may assist with exceptions, but should not silently replace the authoritative match.
Payroll	Apply versioned PAYE, National Insurance and pension contribution rules.	Rates and bands may change, but the calculation must be reproducible under the version applicable at the time.
Payments	Apply explicit payment caps, blocks and routing rules, and release funds following an authorised decision.	Financial execution requires authority, segregation of duties, idempotency and traceable rules.
File processing	Parse CSV, XML, JSON, fixed-width, VCF or other documented record formats.	The format already defines its structure, types and validation rules
Reporting	Create a multi-sheet workbook, such as a 14-sheet assurance report, from approved result data.	Templates, formulas, cell types, totals and worksheet relationships must remain stable.
Identity and access	Authenticate credentials and enforce roles, scopes and tenant boundaries.	A security boundary cannot depend solely on generated interpretation.
Workflow	Advance a case through valid states and manage retries.	State transitions, time-outs, duplicate requests and recovery require predictable semantics.
Data quality	Validate mandatory fields, data types, dates, ranges and identifier syntax.	The rules are explicit and failures must be repeatable and actionable.
Integration	Map known source fields to a canonical API or event contract.	The transformation is defined, testable and versionable

Records management	Execute approved retention and deletion schedules, backup policies and legal holds.	Evidence that the correct policy was executed matters more than interpretation
Billing	Meter successful calls, apply tariffs and calculate charges.	Financial records require exact measurement, idempotency and reconciliation
Service management	Calculate SLA deadlines, pause periods and escalation timers.	Business calendars and contractual durations can be encoded and tested.
Scheduling	Detect an exact timetable, rota or resource collision.	The constraint can be computed directly once the relevant availability is known
Public administration	Calculate statutory or policy response deadlines once the relevant events have been established.	Calendar rules should be reproducible; deciding whether an exemption or exception applies may require separate judgement
Education	Calculate weighted marks, progression outcomes or classifications from approved regulations.	Academic judgement may determine individual marks, but arithmetic and application of approved weighting rules should be consistent.
Procurement	Check whether mandatory declarations, evidence and value-threshold requirements are present.	Completeness and routing rules can be verified directly; evaluating the quality of a narrative response is a separate task
Software delivery	Compile code, run automated tests, verify signatures and compare checksums.	The pass-or-fail criteria are machine-verifiable, although passing them does not prove that the software is defect-free.
Compliance	Check for required evidence, expired approvals or explicitly prohibited conditions.	Presence, expiry and exact rule matches are factual; relevance or sufficiency may still require interpretation or judgement.
Safety	Enforce hard operating limits, fail-safe conditions and interlocks.	A probabilistic model should not be able to negotiate, reinterpret or silently bypass a safety boundary.

The word ‘normally’ matters. For a low-volume, reversible or short-lived task, using a constrained model with independent validation may be more economical than building bespoke software. The decision should reflect consequence, frequency, volatility, validation cost and expected service life.

Where source material is ambiguous or unstructured, an AI model may extract candidate values or propose mappings, rules, formulas or report text. Those outputs should then pass through appropriate schema, type, range, reconciliation and policy checks before they affect an authoritative record or consequential action.

A useful hybrid division is:

- the model interprets, extracts, proposes or explains;
- controlled software validates, calculates and enforces;
- an authorised person resolves material ambiguity or exceptions; and
- the platform records the evidence, versions and decisions.

The model may therefore help create a rule, mapping, formula or template. The approved artefact should then be reviewed, versioned, tested and executed deterministically.

Let the model assist in creating the machine's instructions; do not make the model the execution environment.

Work Where AI Earns its Place

AI earns its place where a task depends on ambiguity, variation, complex patterns or the generation of content for which several answers may be acceptable. In these circumstances, attempting to specify every possibility through procedural code may be impractical, uneconomic or incapable of capturing the meaning required.

Its distinctive value is its ability to generalise: to work with language, images, speech and patterns that were not individually anticipated by a developer. That value does not make the model authoritative. Its outputs remain observations, estimates, candidates or drafts whose treatment should reflect their reliability and consequence.

AI is also a broad category. A classification model, forecasting model, computer-vision system or specialist retrieval model may solve a task more efficiently and predictably than a general-purpose foundation model. An AI requirement is not automatically a generative-AI requirement. Use the least complex capability that satisfies the need.

Capability	Good candidate task	Required qualification
Language understanding	Classify the intent of a free-form service request, complaint or case note	Evaluate using representative examples, measure confusion between classes and provide an abstention or fallback route
Semantic retrieval	Find relevant policy, guidance or previous cases despite differences in wording	Evaluate retrieval quality, enforce access controls before retrieval, return sources and separate retrieved evidence from the final judgement
Summarisation	Condense a long case history, correspondence trail or evidence pack for review	Preserve links to the underlying evidence, test for material omissions and distortions, and never discard the authoritative source record
Drafting	Prepare a response, explanation, decision letter or report narrative	Identify the output as a draft, validate factual and policy claims, and retain accountable approval before consequential release
Information extraction	Identify likely fields in varied invoices, receipts, forms or correspondence	Use schema-constrained outputs where possible and validate types, identifiers, totals and relationships deterministically. Retain references to the source location
Computer vision	Identify likely objects, damage, document structure or visual patterns	Evaluate against representative conditions, including image quality and relevant populations. Route uncertain or consequential cases for review.

Speech recognition	Transcribe service calls, interviews or meetings	Preserve the source recording where lawful, evaluate performance across accents and audio conditions, and check material statements before relying on them.
Semantic comparison	Identify whether two passages, cases or requirements express similar meaning	Define a similarity threshold, evaluate false matches and retain the original material for comparison
Entity resolution	Suggest likely matches between inconsistent supplier, customer or case records	Return ranked candidates and supporting evidence. Do not merge authoritative records solely because a model considers them similar.
Forecasting	Estimate future demand, cost, workload or resource requirements	Back-test against historical periods, communicate prediction ranges and assumptions, and monitor for drift. Do not present a forecast as a commitment or fact.
Anomaly detection	Surface unusual transactions, access patterns, claims or operational behaviour	Establish an appropriate baseline and investigation process. An anomaly is not, by itself, evidence of fraud, misconduct or non-compliance
Pattern discovery	Identify recurring themes within feedback, case notes or incident records	Treat findings as exploratory, assess whether the data is representative and avoid presenting correlation as a causal conclusion
Recommendation and ranking	Rank guidance, knowledge articles, options or next actions	Make the optimisation objective and constraints explicit, evaluate bias and inappropriate feedback loops, and allow override where appropriate
Translation	Produce an initial translation of correspondence, guidance or service content	Preserve the source text and use qualified review where accuracy affects safety, rights, legal meaning or access to services.
Accessibility support	Produce a plain-language draft, summary or alternative explanation	Test with intended users, preserve the authoritative version and do not assume that generated content establishes legal accessibility compliance
Code assistance	Generate, explain, refactor or review code and propose tests	Require developer review, compilation, automated testing, dependency and

		licence checks, and appropriate security scanning before acceptance.
Conversational system access	Convert a user request into a query or proposed API operation	Authenticate the user independently, authorise every operation, validate parameters and restrict the model to explicitly permitted tools and actions
Scenario exploration	Generate possible options, risks, assumptions or failure scenarios	Treat outputs as hypotheses for investigation, verify supporting claims and do not substitute generation for formal analysis or professional advice
Evidence interpretation	Explain why a receipt, narrative or circumstance may support a policy exception	Cite the evidence and relevant policy, expose uncertainty and leave approval with the authorised decision-maker

Make uncertainty visible

The objective is not to surround useful AI with so many controls that its value disappears. It is to ensure that the uncertainty introduced by the model remains visible and does not silently become an authoritative fact or action.

A well-designed contract should identify whether an AI output is:

- an extracted candidate;
- a classification or score;
- a retrieved source;
- a prediction;
- a recommendation;
- a generated draft; or
- an explanation intended to support a person.

Where relevant, the output should also carry its supporting evidence, model and configuration version, validation status, applicable limitations and escalation route. A confidence score should be exposed only where its meaning has been evaluated and calibrated for the specific task; an untested model assertion that it is ‘highly confident’ is not assurance.

The UK Government’s AI Playbook states that AI systems are not guaranteed to be accurate and emphasises testing, understanding limitations, using the right tool and retaining meaningful human control at the appropriate stages. UK guidance on AI assurance similarly focuses on measuring, evaluating and communicating the trustworthiness of systems against their intended context.

The NIST Generative AI Profile identifies confabulation-the confident presentation of erroneous or false content-as a characteristic generative-AI risk. It also warns against extrapolating capability from narrow or anecdotal demonstrations and recommends empirical evaluation and verification of sources.

None of this makes generation unusable. It makes task selection, evaluation and verification part of the design.

AI earns its place when it addresses ambiguity or complexity more effectively than simpler alternatives and when its remaining uncertainty can be measured, communicated and controlled.

The Hybrid Pattern Across Real Services

The practical design choice is rarely ‘AI or no AI’. It is a controlled allocation of responsibilities between conventional software, probabilistic models and accountable people.

The division will not always be equal. AI may be central to one stage and unnecessary in another. The responsibilities are logical boundaries: they may be implemented within a modular application and do not automatically require separate services.

Use case	Deterministic responsibility	Probabilistic responsibility	Human responsibility
Financial assurance	Parse source records, build canonical transactions, calculate values, apply approved rule packs and produce traceable findings	Interpret ambiguous merchant descriptions, supporting evidence or exception narratives	Own and approve policy and rule packs, investigate findings and decide material exceptions
Invoice processing	Validate supplier records, purchase orders, totals, VAT treatment, bank details and duplicate references	Extract candidate fields from varied scans and classify unusual supporting documents	Approve new suppliers, bank-detail changes and material exceptions under appropriate segregation of duties
Contract review	Control document versions, identify dates, check required clause presence and track explicit obligations	Summarise obligations, compare semantically similar clauses and flag ambiguous or unusual language	Legal and commercial owners determine interpretation, accept risk and approve contractual positions
IT service management	Authenticate users, retrieve authoritative records, calculate SLAs, enforce valid workflow states and execute permitted actions	Classify intent, retrieve semantically relevant guidance, summarise incidents and suggest resolutions	Own service policy, authorise material or irreversible changes and resolve escalations
Healthcare support	Validate patient identifiers and units, enforce approved bounds and interlocks, control workflow and preserve audit records	Assist with image or text interpretation, prioritisation and clinical-document summarisation	Qualified clinicians make clinical decisions; manufacturers and deploying organisations retain regulatory, safety and operational responsibilities
Cybersecurity	Enforce identity, isolation policies, allowlists, logging,	Correlate events, rank anomalies, identify patterns and summarise alerts	Security owners set risk appetite, approve high-impact containment and

	response gates and rollback controls		manage novel or uncertain incidents
Recruitment	Validate application completeness and compare declared evidence with published essential criteria	Summarise experience, identify relevant skills evidence and support accessible candidate communication	People define lawful and job-relevant criteria, provide reasonable adjustments and make accountable employment decisions
Public services	Apply versioned entitlement calculations once relevant facts and policy interpretation are established; control deadlines, workflow and audit	Interpret free-form circumstances, retrieve guidance and explain information in accessible language	Accountable officials resolve disputed facts, discretion and exceptional or materially adverse cases, with appropriate routes to challenge and redress
Education	Validate prerequisites, calculate approved mark weightings and enforce timetable constraints	Recommend learning resources, identify themes and draft feedback or accessible explanations	Educators exercise academic judgement, decide support needs and resolve exceptional circumstances
Manufacturing	Enforce tolerances, equipment permissions, stop conditions and safety interlocks	Detect likely defects from images or sensor patterns and predict maintenance needs	Engineers own the safety case and handle uncertain, novel or safety-critical conditions
Supply chain	Apply inventory constraints, approved reorder rules and contractual limits; execute authorised orders	Forecast demand, identify disruption patterns and summarise supplier intelligence	Operational owners set risk tolerances and approve material sourcing or continuity decisions
Insurance and claims	Validate policy dates, limits, excesses, calculations, evidence requirements and workflow	Extract evidence, estimate likely damage and identify anomalous claim characteristics	Accountable claims professionals resolve complex, contested or adverse decisions and provide appropriate review and redress
Customer communication	Retrieve approved facts, apply contact preferences, enforce disclosure controls and issue mandated notices	Draft context-sensitive responses, identify sentiment and support translation or plain-language explanation	Service owners define communication policy; qualified people handle vulnerable customers, complaints and consequential correspondence

Software delivery	Compile, test, scan, sign and deploy through controlled release gates	Generate or explain code, propose tests and assist with defect diagnosis	Engineers review changes; product, security and service owners accept residual risk and authorise release

Accountability does not transfer to the model

The probabilistic column deliberately describes a contribution rather than a responsibility. A model cannot own a policy, accept legal or operational risk, hold delegated authority or answer for an outcome. Those responsibilities remain with the organisation and the people acting on its behalf.

The human and organisational column does not mean that a person must approve every individual transaction. Accountability may instead be exercised by:

- approving policies, rule packs and operating limits;
- determining which actions may be automated;
- setting confidence, validation and escalation thresholds;
- reviewing exceptions and adverse outcomes;
- monitoring performance and emerging risks; and
- retaining authority to suspend or change the service.

Human involvement must also be meaningful. A reviewer cannot compensate for an unsafe system if they lack the evidence, competence, time or authority needed to challenge its output.

The healthcare example

Healthcare demonstrates why human ownership is necessary but not sufficient. A statement that ‘the clinician remains responsible’ does not remove the responsibilities of the technology manufacturer, deploying organisation or service operator.

In the UK, software performing a medical purpose may fall within the MHRA’s medical-device framework, including requirements applying across the software and AI lifecycle. NHS clinical-risk management standards distinguish responsibilities associated with manufacturing health IT systems and those arising from their deployment and use. NHS England is currently reviewing DCB0129 and DCB0160, but they remain important expressions of manufacturer and deploying-organisation accountability.

The NICE Evidence Standards Framework provides a further mechanism for determining what evidence should support the effectiveness and economic value of digital health technologies.

The resulting healthcare pattern is therefore not simply ‘AI advises and a clinician approves’. It combines clinical evidence, regulated technology, clinical-risk management, controlled workflow, professional judgement, organisational governance and post-deployment monitoring.

Why the hybrid pattern is stronger

A hybrid design allows each type of component to fail in a way that the wider service is prepared to detect and manage:

- deterministic components provide repeatable execution and enforce hard boundaries;
- probabilistic components address ambiguity and surface useful observations;
- people and organisations own policy, material judgement and residual risk; and
- the platform preserves the evidence connecting all three.

Hybrid architecture is not an uneasy compromise between software and AI. It is the deliberate containment of uncertainty so that the organisation can exploit AI's strengths without allowing uncertainty to govern the entire service.

Reference Architecture: Bound the Uncertainty

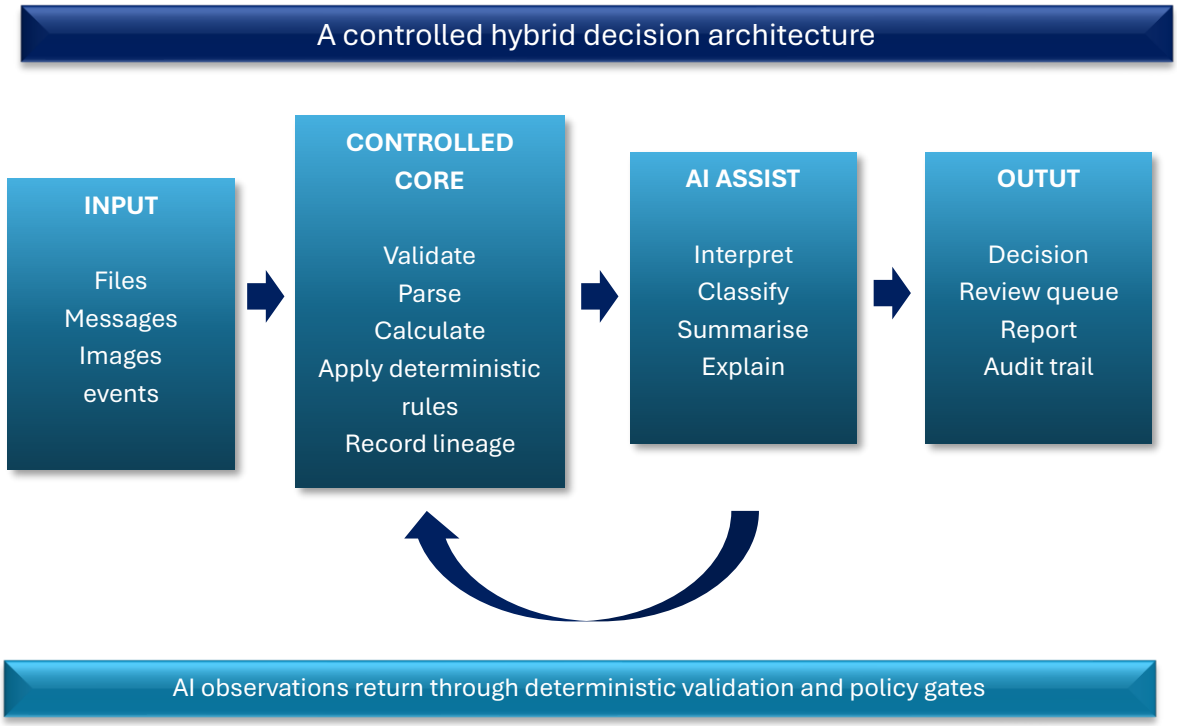


Figure 1. Probabilistic interpretation is a bounded capability inside a controlled system, not the system's authority.

The architecture shown above defines logical responsibilities and trust boundaries. It does not require every component to be deployed as a separate service. A low-risk or early-stage implementation may use a modular monolith; separate deployment units should be introduced where security, scale, resilience, ownership or release independence justifies them.

Information may cross these boundaries; authority should not. The controlled core determines what information the AI capability may receive, validates what it returns and retains control of business state and external action. The AI component is a source of useful but untrusted computation - not a root of trust.

Design rules

- **Keep authority outside the model.**
  Authentication, authorisation, tenant isolation, policy ownership, permissions and secret handling belong in controlled services. Identity and tenant context must come from a verified security context, not from a prompt or model output.

- **Authorise before retrieval and minimise disclosure.**

Restrict records and documents before they enter the model context. Provide only the information required for the task, applying redaction or pseudonymisation where appropriate. Do not give a model broad access and ask it to decide what the user is permitted to see.

- **Use typed canonical data.**
  Represent money, currencies, dates, quantities and identifiers using typed models. AI may propose candidate values extracted from ambiguous material, but deterministic software should normalise, validate and reconcile them.

- **Use versioned, structured contracts.**
  Pass structured data across component boundaries and validate model-produced output against an explicit schema. Schema-valid JSON can still be factually or semantically wrong; domain checks, reconciliation and policy validation must follow.

- **Keep evidential states distinct.**
  Represent source facts, derived facts, model observations, recommendations, findings, human decisions and executed actions separately. An observation or recommendation must not silently become an approved outcome.

- **Make uncertainty explicit.**
  Provide outcomes such as unknown, insufficient evidence, out of scope and requires review. Missing, unreadable or conflicting evidence must not be converted into false certainty.

- **Constrain tools and side effects.**
  Use allowlisted tools, least-privilege credentials, bounded parameters and authorisation at the point of action. A model may propose an action; it does not grant permission to perform it.

- **Apply independent policy gates.**
  Run deterministic validation and policy controls before payments, record changes, communications or other consequential external actions. Use confirmation, approval, idempotency and recovery controls in proportion to the potential impact.

- **Design controlled failure.**
  Timeouts, unavailable models, malformed output, missing evidence and failed validation should produce known states or safe fallbacks-not improvised continuation. Confidence thresholds should be used only where the score has been evaluated and calibrated for the task; a model's assertion that it is confident is not itself a control.

- **Preserve proportionate lineage.**

Record relevant source references, validation results, rule and policy versions, model deployment, prompt or template version, retrieval source version, correlation identifiers and accountable decisions. Avoid creating unnecessary copies of personal or confidential information in logs.

- **Treat changes as releases.**
  Changes to models, prompts, rules, schemas, retrieval sources and tool permissions should receive regression testing, evaluation, approval and rollback capability proportionate to the use case.

- **Make human review meaningful**.
  Give reviewers the evidence, uncertainty, deterministic findings and proposed outcome needed to challenge the result. The reviewer must also have sufficient competence, time and authority; an approval button alone is not effective human oversight.

- **Observe the complete service**.
  Monitor validation failures, abstentions, overrides, latency, cost, security events and operational failures. Protect telemetry using appropriate access controls, redaction and retention rules.

The NCSC treats security as a whole-system and whole-lifecycle requirement, covering secure design, development, deployment, operation and maintenance. Security cannot therefore be added solely around the prompt or model interface.

The NCSC also warns that large language models may not reliably distinguish instructions from data. Its example of malicious instructions concealed within a financial transaction reference is directly relevant to financial assurance systems. Transaction descriptions, documents, emails, receipts and retrieved text must consequently be treated as untrusted data, never as authority.

Prompt-level safeguards may reduce the likelihood of manipulation, but the architecture must also limit the consequences if those safeguards fail. The objective is not to prove that the model cannot be influenced; it is to ensure that influencing the model cannot bypass authorisation or produce an unacceptable action.

The Anti-Pattern Catalogue

An anti-pattern is not a technique that must never be used. It is a recurring design that initially appears expedient but assigns inappropriate responsibilities to the model and creates predictable problems as the service scales, changes or becomes consequential.

The LLM as universal execution engine

One prompt receives a file, interprets its format, applies policy, calculates amounts, decides breaches and creates the final report. Every responsibility becomes entangled with the model's context, configuration and behaviour.

The problem is not that a model participates in several stages. A model may legitimately interpret information, retrieve evidence, propose actions and draft an explanation within the same service. The anti-pattern arises when its output substitutes for parsing, calculation, authority, durable state, security controls or audit evidence without independent boundaries.

Anti-pattern	Why it fails	Better design
LLM as calculator	Arithmetic, comparison and rounding may be wrong even when the accompanying explanation appears convincing.	Use typed decimal arithmetic and versioned calculations. AI may explain or contextualise the independently calculated result.
LLM as parser for documented formats	Known fields are rediscovered on every request, while mandatory fields, types, lengths and malformed records become implicit prompt behaviour.	Use a typed, versioned parser with explicit schema validation and error reporting. Preserve rejected records and source lineage; reserve AI for genuinely ambiguous content outside the documented format.
LLM as policy engine	Natural-language policy may be reinterpreted on every call, making precedence, effective dates and exception handling opaque.	Express the determinable parts of approved policy as versioned rules. Use AI to retrieve, explain or identify ambiguity, and route genuine judgement appropriately.
LLM as database	Prompt context may be stale, incomplete or lossy and does not provide integrity constraints, concurrency control or transactional semantics.	Query authoritative systems through constrained APIs and retain source identifiers, timestamps and provenance.
LLM as access-control gate	Prompt behaviour is not a security boundary, and untrusted content may influence what the model believes a user is permitted to do.	Authenticate, authorise and restrict data and tools before model access. Re-authorise consequential operations at the point of action.

LLM as workflow engine	A model does not provide durable state, reliable retries, idempotency, transaction boundaries or predictable recovery.	Use a durable workflow engine or state machine. The model may propose a bounded transition, but controlled software validates and executes it.
LLM as spreadsheet engine	Cell types, formulas, relationships, layout and totals may drift between executions, making generated workbooks difficult to reconcile and regression-test.	Generate workbooks from validated data using a controlled template and appropriate software library. Use AI only for narrative or interpretive content where useful.
LLM as compliance certifier	Plausible narrative can conceal missing or conflicting evidence, and a generated conclusion cannot confer organisational authority.	Produce structured findings, applicable rule references, supporting evidence and explicit evidence gaps. Retain accountable approval where certification or judgement is required.
LLM as security control	Probabilistic interpretation cannot reliably enforce a hard security boundary, and adversarial input may influence the component performing the assessment.	Enforce identity, isolation, allowlists, permissions and response gates deterministically. AI may correlate events, rank anomalies and summarise alerts inside constrained boundaries.
LLM as its own reviewer	A second prompt or model call may repeat, rationalise or rephrase the original error. Fluent agreement is not independent assurance.	Use authoritative data, deterministic validation, representative evaluation, independent checks and appropriately trained review according to the risk.
LLM as audit trail	A generated reconstruction is not reliable evidence of the inputs, versions, tool calls, validations, decisions or actions that actually occurred.	Record source lineage, processing events, versions, correlation identifiers and authorised actions directly. Store model output or explanation as a separate artefact.
LLM as confidence authority	A self-reported confidence statement may sound precise without having any measured relationship to the likelihood or severity of error.	Expose confidence only where it has been evaluated and calibrated for the task. Otherwise use explicit outcomes such as unknown, insufficient evidence or requires review.

What a successful demo can conceal

A demonstration answers a useful but limited question: can the proposed capability work on selected examples? It does not, by itself, establish that the design is reliable, secure, economical or operable as a service.

A successful demonstration may conceal that:

- the examples were selected or simplified and happened to fit the prompt and context window;

- the model produced the expected answer for the wrong reason, without an independent test oracle exposing the difference;
- a reviewer noticed and corrected errors without that effort being measured;
- the demonstration identity had broader access to data or tools than a production user should receive;
- one model and platform version was tested, while a later update may change behaviour after acceptance;
- malformed inputs, hostile content, duplicate requests, retries, timeouts and dependency failures were never exercised;
- demonstration volumes did not expose concurrency limits, rate limits, provider quotas or tail latency; and
- the apparent saving excluded integration, assurance, token consumption, correction, monitoring, support and failure costs.

A successful demonstration establishes possibility, not production fitness.

Before scaling it, decompose the prompt into its individual responsibilities and ask what independently validates each output, owns business state, authorises consequential action and records the supporting evidence. Where the answer to all four questions is simply ‘the model’, the proof of concept has exposed an architecture task-it has not completed one.

The model may contribute across several stages of a service. It should not silently become the calculator, database, policy owner, security boundary, decision-maker and auditor at the same time.

Worked example: Financial Assurance

This worked example contrasts a model-heavy proof of concept with a proposed production-shaped alternative. The alternative is an architectural design direction; it is not a claim that the API or its constituent components have been approved, integrated or deployed.

The financial assurance use case began with a sensible observation: asking an agent to understand a Visa transaction file, interpret policy and build a substantial Excel review consumes time and model capacity. More importantly, it asks the model to reconstruct stable structures, calculations and rules during every run.

Using a model (in this case, Copilot) in this way can demonstrate end-to-end value quickly. The weakness is not that the model is involved. It is that probabilistic interpretation has been extended across responsibilities that already have defined structures and require stronger control.

Model-heavy proof of concept

Consider the current model-heavy approach:

- Copilot receives the transaction file, policy material and output expectations.
- Within one model-centric flow, it interprets the VCF structure, reconstructs transactions, applies limits, reasons about potential breaches and prepares a multi-sheet workbook.
- Parsing, calculation, policy interpretation, finding generation and reporting are consequently coupled to the model's context and behaviour.
- The flow may produce a useful result, but its intermediate decisions are difficult to inspect, replay, test independently or cost accurately.

This is not a failed proof of concept. It can demonstrate that the underlying business problem is valuable, reveal the information reviewers need and identify where interpretation may improve the process.

It becomes an architectural anti-pattern only if the proof-of-concept implementation is treated as the production design without first separating its responsibilities.

Proposed Production-shaped division of labour

A technical prototype of the Financial Assurance API was then deployed to an Azure Container App in a sandbox environment. It exposes operations for health, assessments and policies through `/health`, `/v1/assessments` and `/v1/policies`.

The deployment establishes a real boundary for exercising the deterministic architecture. It has not been formally adopted, integrated into the current assurance process or approved as a production service. The optional AI assistance and downstream operational integrations shown below remain part of the proposed design.

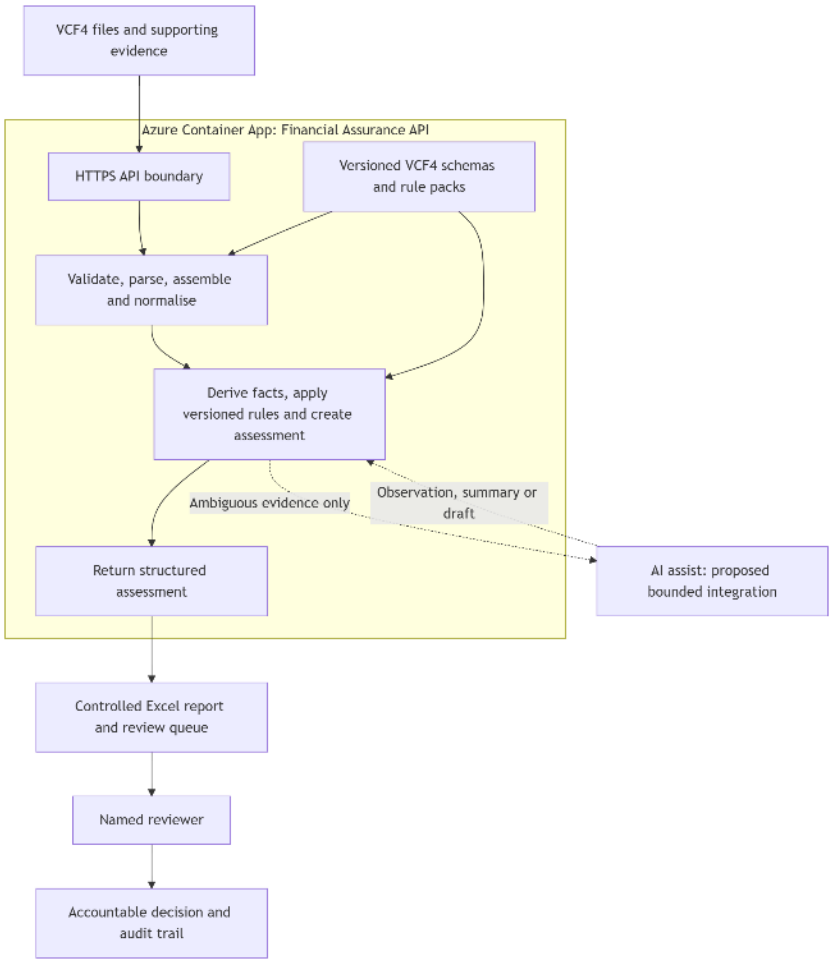


Figure 2. The deployed Azure Container App provides the technical API boundary for the prototype. AI assistance is introduced as a bounded contribution and cannot bypass the controlled assessment or accountable review.

The following table describes a proposed allocation of responsibilities. It does not represent the current implementation status of the Financial Compliance Agent or assert that an API has been formally adopted.

Stage	Proposed Implementation	Architectural Purpose
Upload	Authenticated upload or API boundary with file-type and size controls, malware checks and a correlation identifier.	Establishes a controlled and auditable entry point.
Ingestion	Versioned VCF4 T5, T7 and T8 parser.	The source format has a documented structure that should not be rediscovered by a model during every run.
Assembly	Deterministic transaction linking, sequencing and validation.	Missing, duplicate and orphan records must be detected and reported consistently.

Normalisation	Typed canonical transaction model.	Downstream rules and reporting should not depend directly on the Visa record layout.
Facts	Named and typed derived facts with references to their source evidence.	Rules should consume stable evidence rather than values reconstructed from prose.
Policy	Approved and versioned rule pack with effective dates and rule identifiers.	Thresholds, prohibitions and precedence must be reviewable, testable and reproducible.
Assessment	Structured findings, unknowns, evidence references and processing diagnostics.	The assessment is a traceable pre-screen rather than an unsupported certification.
AI assistance	Optional bounded interpretation of genuinely ambiguous merchant descriptions, receipts or exception evidence.	Model effort is reserved for semantic uncertainty, and its output remains an observation or draft rather than an authoritative finding.
Reporting	Multi-sheet Excel workbook generated from the structured assessment using a controlled template.	Fourteen worksheets do not require fourteen separate acts of model interpretation.
Review and decision	Named reviewer receives the source evidence, deterministic findings, model observations and unresolved exceptions.	The reviewer can resolve material ambiguity and record an accountable decision and its basis.
Audit	Protected record of source identifiers, component versions, rule results, model interactions, validation outcomes and reviewer decisions.	An outcome can be reconstructed without relying on a generated narrative as the audit trail.

These are logical responsibilities rather than a proposed network of microservices. An initial implementation could expose a modular application through a controlled API and separate components physically only where scale, security, ownership or operational requirements justify it.

If adopted, this division of labour would provide benefits beyond reduced token consumption: independently testable components, explicit rules, faster execution, clearer failure modes, reproducible reports and an audit trail that does not depend on trusting fluent prose.

Where AI still adds material value

AI still provides substantial value within the proposed design. Potential uses include:

- interpreting ambiguous receipt or merchant descriptions;
- summarising why several independently established findings matter when considered together;

- retrieving semantically similar historical exceptions for reviewer consideration;
- drafting reviewer narratives and accessible explanations; and
- answering grounded questions about the assessment and its supporting evidence.

A historic exception would be contextual evidence, not automatically a precedent or an amendment to policy. A generated summary or explanation would similarly support the reviewer without replacing the underlying findings.

Model outputs should therefore be identified as observations, recommendations or drafts, linked to their supporting sources where appropriate, and retained separately from deterministic facts, rule findings and accountable decisions.

The intended design preserves the speed and interpretive value demonstrated by the proof of concept while moving calculation, policy enforcement, business state and authority into controlled components.

Testing Two Kinds of System

A hybrid solution needs different assurance disciplines for its deterministic and probabilistic components. Deterministic software is tested against specified behaviour and invariants; probabilistic capability is evaluated statistically against defined tasks and acceptable error. The combined service must then be tested end to end.

Treating an AI evaluation like a unit test, or deterministic code like an informal model demonstration, will miss important failures. Testing each component separately is also insufficient because defects frequently emerge at contracts, state transitions, fallback paths and authority boundaries.

Assurance area	Deterministic component	Probabilistic component
Expected result	Exact outputs and invariants for controlled inputs, state, configuration and version	Task-appropriate quality across representative cases; exact wording or output need not be identical
Coverage	Branches, boundaries, states, malformed inputs, retries and concurrency conditions	Relevant segments, languages, contexts, rare cases, ambiguous examples and adversarial inputs
Regression	Approved cases and invariants replay exactly unless an intentional change has been made	Compare metrics, error severity and failure taxonomy with the accepted baseline; retain critical-case checks
Contracts and validation	Schema, type, domain, interface and state-transition tests	Structured-output validation followed by semantic, factual and source-grounding checks
Evidence and replay	Retain relevant input references, state and code, rule, schema and configuration versions	Retain proportionate records of context, model deployment, prompt or template, retrieved sources, tools and inference settings
Performance and cost	Latency, throughput, resource consumption and capacity limits	Latency distribution, token or call consumption, retries, fallbacks, quotas and provider variance
Quality measures	Pass or fail, property-based assertions and reconciliation	Task-appropriate measures such as precision, recall, calibration, error severity and rubric-based assessment
Change unit	Code, dependency, rule, schema or configuration	Model or deployment, prompt, retrieval corpus, tool contract, inference setting or threshold

Failure handling	Explicit errors, safe state transitions, idempotency and rollback	Abstention, fallback, review routing, degraded operation and malformed-output handling
Security and abuse	Authentication, authorisation, input validation, dependency and interface testing	Prompt injection, data disclosure, adversarial input and attempted misuse of tools or permissions
Production monitoring	Exceptions, invariant breaches, reconciliation failures and service health	Drift, sampled quality, abstentions, overrides, safety events, latency, cost and unexpected model changes

Passing the tests in both columns does not by itself demonstrate that the service is safe. End-to-end tests should prove that:

- a model observation cannot overwrite an authoritative source fact;
- malformed or semantically invalid model output is rejected;
- unavailable or slow models produce a controlled state;
- retries cannot duplicate a finding, decision or external action;
- untrusted content cannot alter permissions or bypass policy gates; and
- evidence and lineage survive every transition through the service.

For the proposed 'Financial Compliance Agent' design example (in section 11), deterministic tests should prove that T5, T7 and T8 records are parsed and linked correctly, financial values remain exact, orphan records remain visible, rule versions are recorded and reports reconcile with structured assessment data.

AI evaluation should separately measure the interpretation of ambiguous descriptions and supporting evidence. End-to-end tests should then demonstrate that a model-produced observation remains distinct from source facts and cannot become a rule finding or accountable decision without the required validation and policy controls.

Evaluation principles

- **Define the intended use first.**
  Identify the task, affected population, error tolerance and consequences of false-positive, false-negative and abstained outcomes before selecting metrics.

- **Use appropriate test assets.**
  Deterministic paths need exact expected outcomes and executable properties. Probabilistic paths need versioned evaluation sets containing representative examples, difficult negative cases, supporting evidence and explicit scoring criteria. Generative tasks may permit several acceptable answers.

- **Separate development from acceptance.**

Do not repeatedly tune prompts, thresholds or models against the final evaluation set. Maintain suitable development, regression and held-out acceptance cases.

- **Use representative data under appropriate controls.**
  Include realistic variation, missing evidence, poor-quality inputs and important user or operational segments. Protect customer-derived, personal and confidential data throughout evaluation.

- **Set acceptance thresholds in advance.**
  Define service-level and critical-case thresholds before the final evaluation. Do not move them merely to declare success. Confidence thresholds should be used only where the score has been evaluated and calibrated for the task.

- **Inspect distributions and serious failures.**
  An acceptable average can conceal poor performance for a particular segment or a small number of high-consequence cases. Measure error severity as well as aggregate frequency.

- **Test model and non-model failures.**
  Include unavailable dependencies, partial files, unsupported formats, stale policy, malformed output, prompt injection, conflicting evidence, duplicate requests, timeouts and failed downstream actions.

- **Measure the complete service.**
  Include correction time, review effort, abandonment, retries, fallback use, latency, operational cost and downstream impact-not merely the apparent quality of generated text.

- **Re-evaluate after meaningful changes.**
  Changes to code, rules, schemas, models, prompts, retrieval sources, tools, thresholds or providers may alter system behaviour and should trigger proportionate regression testing.

Evaluation is therefore not a one-off gateway passed before deployment. It continues through controlled release, production sampling, incident analysis and periodic reassessment.

Human review is a designed component

Human review should not be treated as a general-purpose remedy for unresolved technical risk. Its design should specify:

- which cases reach review and why;
- what evidence, facts, findings and model contributions are shown;
- what the reviewer may approve, reject, amend or return;
- the competence, time and authority the reviewer requires;

- how disagreement, uncertainty and reasons are recorded;
- how urgent, disputed or exceptional cases are escalated; and
- how review outcomes contribute to monitoring and improvement.

Source evidence, deterministic facts, rule findings and model-generated observations should be presented distinctly. A fluent model explanation should not obscure contradictory evidence or be treated as a faithful account of the model's internal reasoning.

The review process must itself be evaluated. Useful measures may include review time, reviewer agreement, overrides, appeals, false acceptance and rejection, recurring evidence gaps and indications of automation bias.

A nominal approval button placed after a persuasive AI narrative is not independent oversight. Human review becomes a control only when the reviewer can understand, challenge and change the proposed outcome.

Cost, Latency and Operational Control

Cost, responsiveness and operability are properties of the complete service, not merely of the model endpoint. A low-cost model call can form part of an expensive process if context preparation, retries, validation and review dominate. A more expensive inference may still be economical if it resolves ambiguity that would otherwise require substantial manual effort.

Why unnecessary AI becomes expensive

- Stable instructions, schemas and policies are repeatedly transmitted and, depending on the platform and its caching behaviour, repeatedly processed and charged as input.
- Large files often require chunking, repeated context, retrieval and reconciliation across several model calls.
- Multi-stage prompts, agents and tool calls can multiply inference, orchestration and failure costs without a corresponding increase in value.
- Variable or malformed output drives retries, defensive prompting, deterministic validation, human checking and exception handling.
- A model-heavy stage may become a latency bottleneck and inherit provider quotas, rate limits and availability constraints.
- Model, prompt and platform changes become functional releases because behaviour, formatting, latency and interpretation may change together.
- At high volume, a small recurring inference cost can exceed the one-off cost of implementing stable logic in conventional software.

A conventional service also costs money to design, build, secure, test, operate and change. The correct comparison is therefore total cost of ownership, not the price of one model call against one function execution.

Stable, high-volume responsibilities will often favour deterministic implementation. Infrequent, volatile or ambiguous work may favour model inference with suitable validation because the cost of encoding and maintaining exhaustive rules would be disproportionate.

The economic question is not simply which component is cheaper to execute. It is which design produces an acceptable outcome at the lowest sustainable cost and risk over the expected life of the service.

Measure the complete service

Cost and latency should be measured per completed, usable outcome rather than per successful API invocation.

Measure	What should be included
End-to-end latency	Upload, parsing, retrieval, queueing, container cold start, model inference, tool calls, retries, validation, review and report generation
Cost per completed outcome	Model calls, retrieval or embedding services, platform compute, storage, networking, monitoring, human review, correction and reprocessing
First-pass completion	Cases producing a valid and usable result without regeneration, correction or manual intervention
Exception and review rate	Abstentions, validation failures, timeouts, unsupported inputs, policy exceptions and cases routed to people
Capacity	Documents or decisions processed per period, concurrency limits, queue depth, provider quotas and burst behaviour
Change cost	Implementation, evaluation, regression testing, approval, release, migration and rollback
Failure cost	Investigation, recovery, repeated work, downstream correction and potential financial, legal or operational impact

Average latency alone is insufficient. The architecture should define appropriate percentile measures, such as p50, p95 and p99, together with the maximum acceptable processing time for the service. A fast median can conceal unacceptable delays for large, complex or repeatedly retried cases.

These measures should be segmented where relevant. A service may perform efficiently for short English-language documents while becoming slower, more expensive or less reliable for scanned evidence, large case files or uncommon formats.

Design for controlled operation and degradation

A model endpoint should be treated as a variable dependency whose latency, availability, capacity and behaviour require explicit controls.

- Apply bounded timeouts, concurrency limits, rate limits and cost controls.
- Use queues and back-pressure where demand may exceed processing capacity.
- Retry only operations that are safe and idempotent, with a defined maximum number of attempts.
- Distinguish unavailable, timed out, abstained, invalid output, insufficient evidence and policy validation failed. These conditions have different operational meanings.
- Where AI assistance is optional, allow the deterministic assessment to continue and record that interpretation or explanation was unavailable.
- Where model assistance is essential, route the case to a controlled pending or review state rather than improvising an answer.

- Do not allow automated provider failover to silently alter model behaviour, data location, contractual terms or assurance status. Alternative providers and models should be evaluated and approved explicitly.
- Make caching version-aware and subject to the same access, retention and confidentiality controls as the original information.
- Monitor invocation volume, tokens, retries, validation failures, latency, cost, quota consumption and fallback use alongside conventional service telemetry.

For a production-shaped Financial Assurance service, large assessments may be better handled as asynchronous jobs. The API could validate the submission, return a correlation or assessment identifier and expose an explicit processing status and result. This avoids holding an HTTPS request open throughout parsing, assessment and reporting, while supporting controlled retries and idempotent result retrieval.

The sandbox deployment does not need to implement every production mechanism immediately. It should, however, preserve an architecture through which these controls can be introduced without redesigning the decision boundary.

Governance follows the decision, not the label

Governance should reflect what the service does, what information it uses, who it affects and what consequences may follow. An exact rules engine used to determine entitlement may require stronger governance than a low-impact generative drafting aid. Conversely, describing a system as hybrid does not dilute the obligations attached to its probabilistic component or final decision.

The UK Government's Algorithmic Transparency Recording Standard establishes a standardised way for public-sector organisations to publish information about how and why algorithmic tools support decisions. Its formal scope applies to government departments and specified arm's-length bodies, while it remains recommended more broadly across the public sector. Organisations outside that scope may still find its structure useful for recording purpose, ownership, data, decision role, oversight, risks, performance and change.

Record	Deterministic layer	Probabilistic layer
Purpose and role	Authoritative calculation, validation, policy or workflow responsibility	Bounded task assigned to the model and permitted uses of its output
Identity and version	Service, code, dependency, schema, configuration and rule-pack versions	Provider, model deployment, prompt or template, retrieval source, tool and inference-setting versions
Inputs	Validated fields, authoritative state and source lineage	Proportionate record or reproducible reference to context, retrieved sources and tool results

Output	Derived facts, calculations, validation results and rule findings	Observation, classification, score, prediction, recommendation, draft or supplied explanation
Validation	Schema, type, reconciliation, policy and state checks	Schema compliance, grounding, factual checks, uncertainty and validation status
Decision and action	Authorisation, policy gate, workflow transition and executed action	Advisory contribution or candidate output-not implied authority
Accountability	Product, service, policy, data and operational owners	Owner of the model-enabled capability, evaluator and accountable business decision-maker
Change control	Approved code, schema, rule and configuration releases	Evaluated changes to model, prompt, retrieval, tools, thresholds or provider
Monitoring	Errors, reconciliation failures, invariant breaches, performance and availability	Quality drift, harmful output, abstentions, overrides, latency, cost and incidents

These records should be connected through appropriate correlation identifiers and retained proportionately. Recording every prompt indiscriminately may itself create security, privacy and records-management risks.

Where a system is high-impact or regulated, teams should obtain current legal, regulatory, clinical and information-governance advice. Where it applies, the EU AI Act uses a risk-based structure with obligations that depend on the use case, the organisation's role and the applicable implementation stage. Its territorial application and requirements cannot be inferred simply from a system containing AI or from describing that system as hybrid.

A Practical Review Method

Architecture review should examine the individual decisions and actions within a service rather than approve or reject an application simply because it contains AI.

The method below turns the principles in this paper into a repeatable and traceable design exercise.

Run this exercise before approving an AI use case

Draw the service as a sequence of decisions and actions rather than as a collection of applications. A single journey may include parsing a document, retrieving evidence, interpreting meaning, calculating an amount, applying policy, recommending an outcome, authorising an action and producing a report. These responsibilities should be considered separately.

For each decision or action:

- Define its purpose and required output.
- Identify its inputs, authoritative sources and applicable policy.
- State whether the output is a fact, model observation, estimate, recommendation, decision or executed action.
- Record the tolerance for error, potential consequence, reversibility, expected volume and required response time.
- Identify the person or organisational role accountable for the outcome.
- Classify the responsibility and select an appropriate implementation.
- Define its validation, failure, fallback, monitoring and change controls.

Classification	Evidence	Default implementation
D Deterministic	Stable rules, exact calculation, known schema, authoritative lookup, security boundary or defined state transition	Typed software, parser, query, rules engine, state machine or controlled workflow
P Probabilistic	Meaning, prediction, classification or pattern cannot be completely and economically specified	Evaluated statistical or AI model with bounded inputs, defined output status, validation and monitoring
H Human judgement	Normative choice, disputed evidence, discretion or an exception requiring delegated authority	Authorised person supplied with usable evidence, sufficient time and a means to record and escalate the decision
X Hybrid composition	Two or more of D, P and H are required, or a model contribution must be validated or gated before it is used	Explicit staged workflow with typed contracts, authority boundaries, fallback and end-to-end lineage

X is not a fourth kind of answer. It marks a controlled path containing more than one decisioning mode. Record the actual sequence wherever possible; for example,

P → D, P → D → H or D → H rather than writing only **X**.

Potential consequence affects the strength of the controls, not necessarily the classification. A high-impact entitlement calculation may still be deterministic if approved policy defines the answer, while a low-impact interpretation of ambiguous feedback may be probabilistic.

The exercise should produce a recorded architecture decision containing:

- the responsibility being assessed;
- its classification and proposed implementation;
- the alternatives considered;
- the evidence supporting the choice;
- the expected benefits and limitations;
- the applicable controls and acceptance measures;
- the accountable owners;
- dependencies and unresolved risks; and
- any conditions that must be satisfied before release.

The possible outcome is not limited to approval or rejection. The review may approve the design, approve it subject to conditions, restrict it to an experiment, require redesign or conclude that AI does not add sufficient value to that responsibility.

Questions for an architecture review

- **Need and allocation**
  What business outcome and individual decision does the proposed capability support?

  What exact capability is AI supplying that a simpler component cannot provide adequately?

  Which parts of the process already have a schema, calculation, lookup, rule or state transition?

  Has the proposal compared the model-based approach with a credible simpler alternative?

  Is AI being used for genuine interpretation or inference, or being asked to reconstruct stable software behaviour?

- **Evidence and authority**
  Which outputs are source facts, derived facts, model observations, estimates, recommendations, decisions and executed actions?

  What is the authoritative source for each fact, rule and policy?

  Can evidence and lineage be preserved from source input to final outcome?

  Which outputs must be independently validated before they are relied upon?

  Which actions require separate authentication, authorisation, confirmation or approval?

Can a model output alter an authoritative record or trigger an external action without passing through a controlled gate?

- **Evaluation and failure**
  What kinds of error are possible, and what are the consequences of each?

  What representative evaluation data, critical cases and acceptance thresholds will demonstrate suitability?

  How will performance be examined across relevant users, document types, languages and operating conditions?

  What happens when the model abstains, returns malformed output, contradicts evidence or produces a plausible but incorrect answer?

  What happens when the model or another dependency is unavailable, slow or rate-limited?

  Does failure produce an explicit and recoverable state?

- **Security and information governance**
  What data is supplied to the model, and is every item necessary for the task?

  Are access controls applied before retrieval and disclosure?

  Can documents, messages or other untrusted content influence instructions, tools or permissions?

  What information is retained in prompts, logs, evaluation data and provider systems?

  Are tool access, credentials and parameters restricted to the minimum required capability?

- **Operation and economics**
  What are the measured end-to-end cost, latency and throughput, including retries, validation, review and correction?

  What proportion of cases completes without regeneration, fallback or manual intervention?

  What capacity, quota, concurrency and degraded-mode controls are required?

  Can the organisation reproduce the evidence, rules, configuration and model deployment associated with an outcome?

  How dependent is the service on a particular provider, model or proprietary interface?

  How would the capability be suspended, replaced or retired?

- **Accountability and change**
  Who owns the service, policy, data, model-enabled capability and final business outcome?

  Where human review is required, does the reviewer have sufficient evidence, competence, time and authority?

  How can affected people question, challenge or appeal an outcome where appropriate?

  How are code, policy, model, prompt, retrieval, tool and data changes evaluated and approved?

  What events trigger regression testing, reassessment or rollback?

Who monitors performance and emerging risk, and who has authority to suspend the service?

Definition of done

A production AI-enabled service should be considered complete only when, in proportion to its risk, scale and expected service life:

- the purpose, intended users and permitted uses are documented;
- deterministic, probabilistic and human responsibilities are explicitly allocated;
- authoritative facts, model contributions, decisions and actions remain distinguishable;
- inputs, outputs and component boundaries use defined and validated contracts;
- authentication, authorisation, data protection and security controls are implemented;
- evaluation has met predefined service-level and critical-case thresholds;
- failure, abstention, fallback, recovery and degraded operation have been tested;
- any required human review is meaningful and operationally sustainable;
- evidence, versions, decisions and changes can be reconstructed;
- cost, latency, quality, security and operational health can be monitored;
- accountable owners, escalation routes and incident procedures are established;
- changes can be tested, approved, released and rolled back safely; and
- applicable legal, regulatory, contractual and professional requirements have been addressed.

A successful model response demonstrates only that the model produced an output. It does not demonstrate that the service is ready.

> *A production AI service is done when the organisation can operate, evaluate, challenge, secure, explain, recover, change and, when necessary, suspend the whole decision system safely.*

Conclusion: Intelligence Through Restraint

Good AI architecture is not measured by how much work is handed to a model. It is measured by whether the resulting service uses the right capability for each responsibility.

AI creates distinctive value where language is varied, meaning is uncertain, patterns are subtle, evidence is incomplete or people need help navigating complexity. Its ability to interpret, generalise and synthesise across those conditions is precisely why it belongs in modern enterprise architecture.

That capability does not make a model the natural execution environment for every surrounding task. AI is an unnecessarily expensive and weak substitute for an existing schema, an exact calculation, an approved policy threshold, a permission check, a state transition or a report template.

The mature pattern may look less magical than the proof of concept, but it is considerably more valuable. Files are parsed by parsers. Money is calculated using decimal arithmetic. Access is enforced by identity and authorisation services. Rules are approved, versioned and executed. AI interprets the language, evidence and context that rules cannot efficiently settle. People exercise accountable judgement where policy and evidence do not determine the answer. The platform records the lineage connecting them.

This division of labour does not make AI peripheral. It concentrates model capability on the responsibilities where it adds substantial value and prevents its remaining uncertainty from silently becoming fact, authority or action.

Restraint is therefore not reluctance to innovate. It is the discipline that allows powerful but uncertain technology to become part of a dependable service.

Do not remove AI from the solution. Remove from AI the responsibilities it should never have been given.

The challenge to delivery teams

When a demonstration succeeds, ask one further question before scaling it: which parts worked because the model supplied a capability that simpler software could not, and which parts worked only because an expensive probabilistic engine was being used to imitate an ordinary parser, calculation, rule or workflow?

Answering that question honestly is not a brake on innovation. It is how a successful demonstration becomes a dependable service - and how innovation becomes infrastructure.

Appendix A: Applying the pattern to Microsoft 365, Power Platform and Azure

This appendix translates the platform-neutral principles in this paper into common Microsoft 365, Power Platform and Azure implementation choices. It is an application guide rather than a mandatory reference architecture. Product selection must still consider licensing, scale, support, information governance, security and the organisation's existing technical estate.

A product is not itself a decisioning mode. Power Automate can contain deterministic conditions and probabilistic AI actions. Copilot Studio can combine authored logic, generative orchestration, knowledge retrieval and connector tools. Dataverse can hold source facts, model observations and human decisions. The deterministic, probabilistic and human classification should therefore be applied to each responsibility-not inferred from the product name.

A.1 Capability selection matrix

Responsibility	Microsoft 365 or Power Platform starting point	Azure or custom extension	Important boundary
User interaction and collaboration	Teams, Power Apps, Microsoft 365 Copilot or a Copilot Studio agent	Custom Teams application or web interface	The channel presents information and captures intent; it does not become the source of truth or grant authority.
Governed documents and knowledge	SharePoint sites and document libraries, with SharePoint agents or Copilot Studio grounded in selected sources	Azure AI Search or a custom retrieval service where bespoke indexing, ranking or multi-source retrieval is required	Apply access control before retrieval, preserve source links and versions, and address existing oversharing.
Simple team-owned records	Microsoft Lists or SharePoint lists	Custom store where platform limits or integration requirements justify it	Appropriate for relatively simple, list-centred processes; avoid gradually turning a list into an undocumented transactional system.
Structured business records and case state	Dataverse with Power Apps, typed columns, relationships and role-based security	Azure SQL, Cosmos DB or an existing line-of-business service behind an API	Authoritative state should remain in a governed data store, not solely in an agent

			transcript, email or generated document.
Workflow and human review	Power Automate, Approvals, Power Apps and Teams notifications	Logic Apps, Azure Container Apps jobs or custom workflow code for more demanding orchestration	Model explicit states, retries, duplicate requests, time-outs, escalation and recovery. An approval action does not by itself make review meaningful.
Bounded validation and routing	SharePoint validation, Power Fx, Dataverse business rules and explicit Power Automate conditions	Reusable application service where rules are shared, complex or consequential	Keep the rules explicit, testable and separate from generated language.
Exact parsing, calculation and policy execution	Power Platform may invoke the capability through a standard or custom connector	Azure Container App or Function containing typed parsers, decimal calculations, versioned rules and automated tests	AI may provide candidate inputs or explanations, but controlled code validates, calculates and enforces.
Language interpretation, retrieval, summarisation and drafting	Microsoft 365 Copilot, SharePoint agents or Copilot Studio	Azure-hosted model or specialist AI service behind a bounded adapter	Return an observation, candidate, recommendation or draft. Preserve evidence, validate outputs and provide abstention or fallback.
Microsoft 365 data access	Microsoft Graph and standard Power Platform connectors	API Management, a custom connector and a controlled application API	Graph is an access interface, not a database. Choose delegated or application access deliberately and request the least privilege required.
External or custom actions	Copilot Studio tools, agent flows and Power Automate connectors	Custom connector calling a secured Azure Container App API, optionally through API Management	Authenticate and authorise at the action boundary. The agent's decision to call a tool is not permission to perform the action.
Identity, credentials and secrets	Microsoft Entra ID, SharePoint permissions, Dataverse security roles and governed	Managed identities, Azure RBAC and Key Vault	Identity must come from a verified security context. Never obtain authority, tenant

	Power Platform connections		identity or secrets from prompt content.
Information governance	Microsoft Purview sensitivity labels, DLP, retention, eDiscovery and audit; Power Platform data policies and Managed Environments	Azure Policy and appropriate controls for Azure data, networking, logs and retention	Connector governance and content governance are related but distinct. Neither replaces correct source permissions.
Monitoring and lineage	Microsoft Purview Audit, Power Platform administration, Dataverse auditing and flow-run history	Application Insights, Log Analytics and service-specific telemetry	Use correlation identifiers to connect the user request, retrieved evidence, model interaction, rule execution, review and final action.
Application lifecycle	Power Platform environments, solutions, pipelines and source control	Container images, infrastructure as code, CI/CD and Container Apps revisions	Treat prompts, topics, agents, connectors, schemas, rules, models and APIs as versioned release artefacts.

A.2 Permissions and actions require separate treatment

Microsoft states that SharePoint agents respond according to each user's permissions to the underlying sources. Copilot Studio knowledge sources using SharePoint and Dataverse can similarly use the agent user's Microsoft Entra authentication.

This is an important control, but it does not establish that all agent operations run under the same identity. Copilot Studio connector tools use user credentials by default but can also be configured to use maker-provided credentials. An authenticated conversation must not therefore be treated as proof that every subsequent retrieval or action has been authorised correctly.

For Microsoft Graph, delegated permissions allow an application to act within both its granted scope and the signed-in user's permissions. Application permissions allow operation without a signed-in user and may provide organisation-wide access. Each integration should record which model is used, why it is required and how its permissions have been restricted.

A.3 Three common implementation patterns

Pattern	Typical composition	Appropriate use
Read-only knowledge assistant	SharePoint library → SharePoint agent or Copilot Studio → Teams or Microsoft 365 Copilot	Finding, summarising and explaining governed material without changing

		authoritative records or executing business actions
Assisted business process	Teams or Power App → Copilot Studio interpretation → Power Automate workflow → Dataverse case → authorised review	Classifying a request, collecting structured information and routing it through controlled workflow and human judgement
Controlled custom service	SharePoint or Power App intake → Power Automate or custom connector → secured Azure API → structured result → Dataverse, report or review queue	Exact parsing, complex validation, calculations, versioned rules or processing that requires conventional software engineering

The third pattern allows Microsoft 365 to remain the familiar user and collaboration environment while a controlled Azure service performs the machine's job.

For example, the Financial Compliance design can be composed as:

`SharePoint or Power Apps intake → Power Automate or custom connector → Azure Container App API → structured assessment → controlled report and review queue → named reviewer`

The API validates and parses VCF4 records, assembles typed transactions, applies versioned rules and returns structured findings. An AI capability may be invoked only for genuinely ambiguous evidence and returns an observation or draft into the controlled assessment path. It cannot bypass validation, alter permissions or make the accountable decision.

The current Azure Container App deployment demonstrates this technical pattern as a sandbox prototype. It should not be represented as an adopted or production-integrated service until its identity, networking, operational ownership, monitoring, assurance and business-process integration have been completed.

A.4 Practical selection rules

- Use **Teams** as a channel for conversation, notification and collaboration.
- Use **SharePoint** where the primary artefact is a governed document or knowledge collection.
- Use **Dataverse** where the primary artefact is a structured case with relationships, business state and role-based access.
- Use **Power Apps** where people need a structured task, case or review interface.
- Use **Power Automate** for bounded connector-led orchestration, notifications and approvals.
- Use **Copilot Studio** where language interpretation, grounded dialogue or conversational access creates value.
- Use **Microsoft Graph** as a controlled API to Microsoft 365 - not as an independent source of truth.
- Use **Azure application code**, including Azure Container Apps, where the responsibility requires exact parsing, complex validation, reusable domain logic, specialist libraries, predictable performance or stronger automated testing.

- Use **Microsoft Entra ID**, **Power Platform governance** and **Microsoft Purview** across the whole composition rather than treating them as safeguards that belong only around the AI component.

The objective is not to keep every solution inside Microsoft 365 or to move every important function into Azure. It is to place each responsibility in the capability that provides the clearest control surface, then connect those capabilities through authenticated, typed and observable contracts.

Appendix B: Sources and further reading

These sources support the risk, security, explainability, regulatory and lifecycle principles used in this paper. The task classifications, decision framework and architectural recommendations are the author's synthesis.
Guidance documents are not substitutes for applicable legislation, regulation, standards or specialist advice. Their inclusion does not establish that every source applies to every use case.

[1] Government Digital Service, *Artificial Intelligence Playbook for the UK Government*, 10 February 2025.
[2] Department for Science, Innovation and Technology, *Introduction to AI assurance*, 12 February 2024.
[3] UK Parliament, *Data (Use and Access) Act 2025*, particularly section 80 and Schedule 6; see also UK Government, *Data (Use and Access) Act factsheet: UK GDPR and DPA*, 27 June 2025.
[4] Information Commissioner's Office and The Alan Turing Institute, *Explaining decisions made with AI*, May 2020. At the date accessed, the ICO identified this guidance as under review following the Data (Use and Access) Act 2025.
[5] National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, NIST AI 100-1, January 2023.
[6] National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile*, NIST AI 600-1, July 2024.
[7] Medicines and Healthcare products Regulatory Agency, *Software and artificial intelligence (AI) as a medical device*, updated 3 February 2025.
[8] NHS England, *National review of clinical risk management standards DCB0129 and DCB0160: supporting information*, 29 June 2026.
[9] National Institute for Health and Care Excellence, *Evidence standards framework for digital health technologies*.
[10] National Cyber Security Centre, *Guidelines for secure AI system development*, version 1.0, 27 November 2023.
[11] National Cyber Security Centre, *Exercise caution when building off LLMs*, 30 August 2023.
[12] National Cyber Security Centre, *Prompt injection is not SQL injection (it may be worse)*, 8 December 2025.
[13] Government Digital Service, *Algorithmic Transparency Recording Standard: guidance for public sector bodies*, updated 8 May 2025.
[14] European Union, *Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)*, consolidated text as at 27 July 2026.
[15] European Commission, *AI Act: regulatory framework*, including the current application timetable and implementation guidance.

Microsoft platform implementation guidance
The following Microsoft documentation supports the illustrative mapping of deterministic, probabilistic and human responsibilities to Microsoft 365, Power Platform and Azure capabilities in Appendix A. Product capabilities, licensing and service limits change; implementation teams should verify the current documentation during design and before release.

[16] Microsoft, *Manage access to agents in SharePoint*, Microsoft Learn, updated 25 June 2026.
[17] Microsoft, *Knowledge sources summary*, Microsoft Copilot Studio, Microsoft Learn, updated 4 August 2026.

[18] Microsoft, *Use Power Platform connectors as tools in Copilot Studio agents*, Microsoft Learn, updated 28 August 2026.
[19] Microsoft, *Overview of Microsoft Graph permissions*, Microsoft Learn, updated 26 December 2025.
[20] Microsoft, *Data policies*, Microsoft Power Platform, Microsoft Learn, updated 8 April 2026.
[21] Microsoft, *Use Microsoft Purview to manage data security and compliance for Microsoft 365 Copilot and Microsoft 365 Copilot Chat*, Microsoft Learn, updated 1 May 2026.
[22] Microsoft, *What is Microsoft Dataverse?*, Microsoft Learn, updated 31 March 2026.
[23] Microsoft, *Overview of application lifecycle management with Microsoft Power Platform*, Microsoft Learn, updated 29 January 2025.
[24] Microsoft, *Custom connectors overview*, Microsoft Learn, updated 27 August 2026.
[25] Microsoft, *Azure Container Apps overview*, Microsoft Learn, updated 7 July 2026.
[26] Microsoft, *Managed identities in Azure Container Apps*, Microsoft Learn, updated 3 June 2025.

Online sources were accessed on / before 8 September 2026.

Guidance, legislation, standards and online services change. Teams should verify the current version, applicability and implementation status for each use case.



PAUL R WILLIAMS
ENTERPRISE ARCHITECTURE | AI | GOVERNANCE

www.modern-architecture.co.uk
www.paulrwilliams.co.uk