Hybrid AI Architecture: Deciding What Must Remain Deterministic

A Control-Allocation Framework for Rules, Models and Human Judgement

Authored by Chennakeshav Adya, Independent Researcher

September 2026

Abstract

Enterprises are adding generative models to workflows originally designed around software rules, databases and accountable people. A model can interpret unstructured material and propose a useful answer, while the same probabilistic behaviour can weaken repeatability, traceability or control when it is allowed to approve, post, pay, disclose or commit without an appropriate boundary. The resulting architecture problem is a question of allocation: which steps should remain deterministic, which can use a model, and which require human judgement.

This paper develops a hybrid AI architecture framework for strategy, risk, finance, product and engineering leaders. It decomposes a workflow into observable decision steps, measures consequence and ambiguity, assigns control authority, and designs exception, override and safe-state behaviour. It uses the EU AI Act, NIST AI Risk Management Framework, CBUAE guidance, UK information-rights guidance, international secure-AI guidance and official cloud architecture publications as primary and authoritative anchors.

Five original figures and five decision tables provide a workflow decomposition, control-allocation matrix, exception tree, latency-cost frontier and architecture decision scorecard. A worked example applies the framework to a hypothetical supplier-onboarding process. All workload volumes, costs, percentages, scores and loss assumptions in the example are analytical assumptions. The paper does not provide legal advice, a prediction, a valuation or a recommendation for a specific system, provider or enterprise.

JEL Classification: D81, L86, M15, O32

Keywords: hybrid AI, deterministic controls, generative AI, human oversight, workflow architecture, model risk, decision rights, assurance, automation, operating model

1. Treat hybrid architecture as an allocation of authority

A hybrid AI system combines deterministic software, one or more statistical or generative models, and human action. The important design choice is the authority granted to each component. A model may draft, classify, extract, recommend, approve, execute or monitor. Those verbs carry different consequences even when they use the same underlying model.

The architecture should state who or what can change an enterprise record, release money, communicate externally, accept a contract, deny a customer, alter a control or trigger a physical action. Authority should follow an explicit policy rather than emerge from a convenient integration. A technically possible action does not establish that the model should be permitted to take it.

Deterministic software produces the same result for the same defined inputs and versioned rule. It is suitable where the organisation can specify the condition and required response. Models are

useful where inputs vary, meaning is contextual or the relationship cannot be captured economically through fixed rules. Humans resolve novelty, exercise accountable judgement and own decisions that depend on values, negotiation or incomplete evidence.

These domains overlap. A model can extract a payment amount, a rule can test it against an approved limit, and a person can review an exception. The strongest design often uses each mechanism for the part it can perform and evidence well. The architecture therefore becomes a chain of bounded authorities rather than a choice between automation and manual work.

2. Decompose the workflow before selecting technology

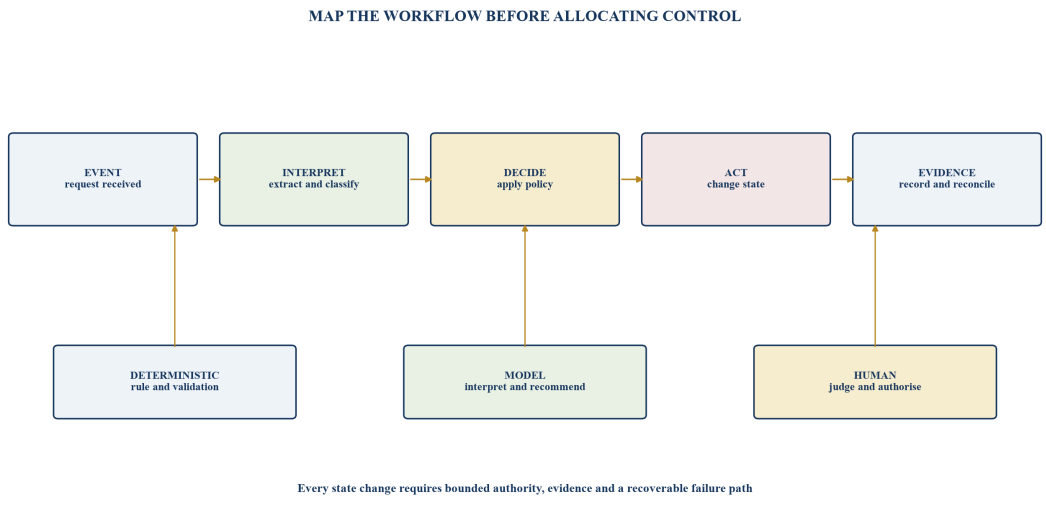
Architecture begins with a workflow map, not a model catalogue. The team should trace the initiating event, input sources, transformations, decisions, approvals, actions, records, notifications, exceptions and final acceptance. Each step should have a defined purpose, owner, input contract and observable output.

Large workflow labels conceal different tasks. “Review a supplier” can include identity matching, document extraction, sanctions screening, policy interpretation, risk scoring, commercial judgement, approval, system creation and payment enablement. These steps have different ambiguity, consequence and evidentiary needs. Applying one model to the whole label transfers authority without making the transfer visible.

The decomposition should identify state changes. Reading, drafting and recommending are generally reversible; posting, paying, deleting, disclosing and signing can create external effects. A state-changing action needs a validated input, an authorised decision and a durable record. The design should also show where a failure can be contained and where it can propagate.

The map should include existing controls and manual workarounds. An AI project can appear efficient by omitting reconciliation, review or correction that people perform outside the measured system. Capturing the complete current process creates a fair baseline and reveals which steps need redesign before any model is introduced.

Figure 1. Workflow decomposition into bounded decision steps



Each step receives a specific authority, evidence requirement and safe-state response.

Table 1. Workflow decomposition register

Field	Management question	Required evidence	Decision owner
purpose	what business result should this step create?	process and customer outcome	process owner
input	what data enters and under which rights?	source, lineage, permission and quality	data owner
decision	what judgement or rule is applied?	policy, model card, prompt or procedure	risk owner
authority	what may the component approve or change?	access role, limit and approval record	accountable executive
action	what internal or external state changes?	transaction, message or system event	control owner
evidence	how can the result be reconstructed?	logs, versions, citations and reviewer record	assurance owner
safe state	what happens when evidence or confidence is inadequate?	stop, queue, rollback or manual route	service owner

The register separates interpretation from authority and action.

3. Define determinism in operational terms

Determinism is an operational property, not a claim that a system is simple. A calculation, permission check, state machine, schema validation or policy threshold can be complex while still producing a specified result for a specified input and version. The organisation can test the rule exhaustively within a defined domain and explain the outcome through the rule path.

The required degree of determinism depends on the action. Financial posting needs balanced entries, valid accounts, currencies and periods. Identity and access management needs explicit roles and separation of duties. A contractual notice needs an approved template, recipient and dispatch record. A safety interlock needs a defined threshold and response. A model can help prepare inputs, while the acceptance gate should remain a controlled rule when the condition is knowable.

Deterministic does not mean static. Rules can be parameterised, versioned and changed through governance. Their strength is that the change is intentional and reviewable. Hidden hard-coded logic creates a different risk; a rule estate can become inconsistent, duplicated and difficult to maintain. Hybrid architecture therefore requires rule ownership and testing as seriously as model governance.

The team should distinguish hard constraints from preferences. A legal prohibition, available-funds test or segregation rule belongs in a hard boundary. A prioritisation preference can be a score or recommendation. Mixing the two allows a model trade-off to override a condition that management intended to be absolute.

4. Identify where models provide distinctive value

Models are most useful where language, images, sound, patterns or uncertain relationships carry information that fixed rules cannot capture economically. They can extract clauses from varied contracts, summarise complex records, detect anomalies, classify free text, predict demand or propose a response. Their value depends on the workflow outcome and the quality of evidence surrounding the output.

A model output should be treated as an observation or recommendation unless the organisation has deliberately authorised more. Confidence values, verbal explanations and citations can support review, while none of them automatically proves correctness. The system should test the output against available records, policies, schemas and business constraints before it is accepted.

The use case should define an operating envelope. It includes intended users, data, languages, jurisdictions, task types, volume, acceptable error, prohibited actions and escalation conditions. A model evaluated on English customer enquiries should not silently expand to legal contracts in another language. Scope changes need a new evidence decision.

Model variability can sometimes improve exploration and drafting. The same variability is unsuitable for a final accounting identity or access decision. Temperature settings and prompt wording may reduce variation, but they do not convert a probabilistic system into a deterministic control. Architecture should use an actual rule, validated service or human approval when repeatability is required.

5. Measure consequence before choosing autonomy

Consequence measures what can happen if a step is wrong, late, unavailable or misused. Financial loss, customer harm, legal exposure, safety impact, data disclosure, operational interruption and reputational damage should be considered. The analysis should cover both individual events and correlated scale.

Reversibility changes the control need. A draft can be edited before use. A public disclosure, payment or irreversible deletion may require preventive approval. Time to detect and time to recover also matter. An error that is reversible within seconds through an automated rollback differs from one discovered after a reporting period or customer decision.

Scale can transform a small unit error into a material event. A weak recommendation seen by one analyst may be caught; the same recommendation executed across thousands of accounts can create concentrated exposure. Autonomy decisions should therefore incorporate rate limits, aggregate limits and stop conditions rather than rely only on per-case accuracy.

The consequence assessment should identify affected stakeholders and the owner who accepts residual risk. Engineering can estimate system behaviour, while business, legal, risk and control owners determine the importance of the outcome. The approval record should state the evidence reviewed and the conditions under which authority must be reduced.

6. Measure ambiguity and evidence availability

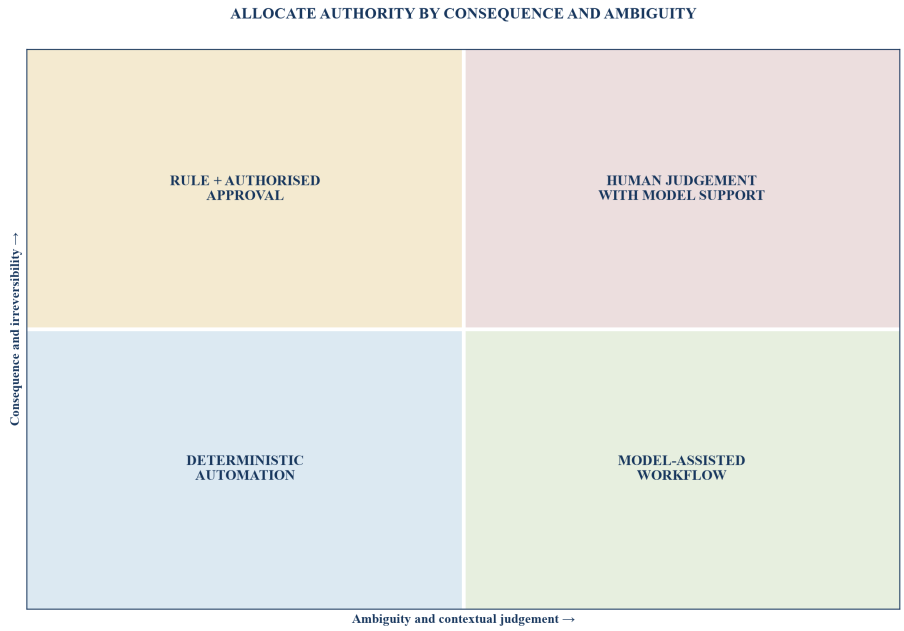
Ambiguity describes how many reasonable interpretations an input or policy may support. A valid date format has low ambiguity. A customer complaint, negotiated contract or strategic recommendation can depend on context, intent and incomplete facts. Models can assist with ambiguity, while consequential ambiguity often requires human judgement.

Evidence availability is a separate dimension. A task may be conceptually clear but poorly documented. A model asked to approve an invoice cannot compensate for a missing purchase order or receipt. The correct response may be to request evidence rather than infer the missing fact. The workflow should reward abstention when the evidentiary threshold is unmet.

Policy clarity should be tested. If knowledgeable people disagree about the required outcome, automating the disagreement can create false consistency. Management should resolve policy, define acceptable discretion and identify escalation before encoding a rule or prompt. Exceptions can then be analysed as policy feedback rather than model failure alone.

The organisation should measure the distribution of ambiguity, not only an average. A large share of routine cases may be suitable for deterministic processing, a middle group for model-assisted review, and a small complex tail for specialists. This segmentation often creates stronger economics and control than giving every case the same architecture.

Figure 2. Control-allocation matrix



Higher consequence and ambiguity reduce the authority suitable for an autonomous model.

Table 2. Control-allocation guide

Condition	Primary mechanism	Model role	Required control
clear rule, low consequence	deterministic automation	extract or classify input	schema check and monitoring
clear rule, high consequence	deterministic rule plus approval	prepare evidence	separation of duties and hard stop
ambiguous, low consequence	bounded model workflow	draft, rank or recommend	sampling, feedback and rollback
ambiguous, high consequence	authorised human judgement	analyse and present evidence	documented decision and escalation
inadequate evidence	exception route	identify missing material	no action until evidence threshold
outside operating envelope	safe state	abstain and explain boundary	queue, stop or controlled fallback

The allocation remains subject to applicable law, policy and tested evidence.

7. Build the deterministic control plane

The control plane sits around model services and enforces permissions, limits, data boundaries, approved versions, routing, evidence capture and safe-state behaviour. It should remain available even when a model produces an unexpected output or a provider is unavailable. The plane turns enterprise policy into executable boundaries.

Input controls can validate identity, consent, jurisdiction, schema, document type and permitted data. Output controls can enforce structure, verify totals, compare facts with a system of record, remove prohibited content and restrict actions. Transaction controls can apply value limits, velocity checks, approved counterparties and dual authorisation.

The control plane should use positive permission. A tool or data source is available because it is explicitly allowed for the workflow and role. Broad credentials granted for development convenience create an authority gap when the model begins to act. Short-lived credentials, least privilege and action-specific interfaces reduce the possible effect of a mistaken instruction.

Version control is essential. Policies, rules, prompts, models, tools and schemas can change independently. The evidence record should identify every version involved in a material decision. A release process should test the combined system, because a stable model can behave differently after retrieval, prompt or policy changes.

8. Use models as bounded interpretation services

A model service should have a narrow contract: defined inputs, allowed context, expected output schema, quality thresholds and prohibited actions. Structured output reduces downstream ambiguity, while the system still needs to validate fields and business meaning. A syntactically valid answer can remain factually wrong or unsupported.

Retrieval can ground a model in approved sources. The architecture should preserve source identity, version, access rights and citation mapping. Retrieval quality should be measured independently from generation quality. A model cannot cite a relevant policy that the retrieval layer failed to supply.

Tool use should be split between read and write capabilities. Reading an approved record presents a different consequence from changing it. A model can prepare a proposed transaction, while a deterministic service validates it and an authorised component commits it. This separation produces a reviewable boundary.

Abstention should be a valid response. The model or surrounding evaluator can indicate that evidence is missing, the case is outside scope or outputs disagree. The workflow should route abstentions efficiently rather than pressure the model to manufacture completeness. Management can then analyse abstention causes and improve data, policy or design.

9. Design human oversight that can change the outcome

Human involvement is effective only when the person can understand the task, inspect relevant evidence, recognise limitations and exercise authority. A reviewer who sees hundreds of alerts without context may provide ceremonial approval. Oversight needs an interface, workload and decision right designed for the actual risk.

The review screen should show the proposed decision, source evidence, rule results, material uncertainty, prior exceptions and consequence. It should distinguish model text from verified system-of-record facts. The reviewer should be able to approve, modify, reject, request evidence, escalate or stop, and each action should be recorded.

Automation bias should be addressed through process design. Reviewers can over-rely on a fluent recommendation, particularly when time is short or dissent is difficult. Training, independent checks, reason codes and periodic blinded review can reveal whether the human is applying judgement or merely confirming the machine.

Staffing should reflect arrival patterns and service commitments. A workflow that routes five per cent of cases to people can still fail if exceptions cluster at month end or require rare expertise. Capacity planning should use the distribution of case complexity, handling time and escalation, including absences and incident conditions.

10. Specify exceptions before production

An exception is an expected architectural state, not an accidental afterthought. Missing data, conflicting records, policy ambiguity, low confidence, provider failure, security alert and attempted prohibited action should each have a defined route. The route should identify ownership, priority, evidence, time limit and permitted next step.

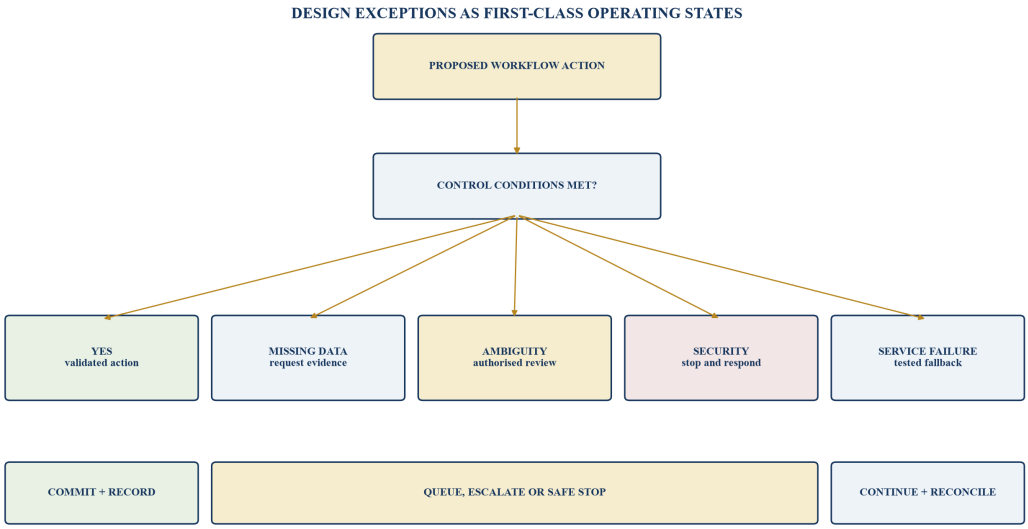
Exceptions should be classified by cause. Data-quality exceptions may return to the source owner. Policy questions may go to legal or risk. Model uncertainty may require a specialist review or a deterministic fallback. Security events may stop the workflow and invoke incident response. A single manual queue conceals these different operating needs.

The design should prevent silent continuation. Defaults can create material risk when a field is absent or a service times out. A fail-closed response is appropriate for some consequential actions, while continuity requirements may justify a previously tested fallback for others. The policy should state the choice and its rationale.

Exception outcomes should improve the system. Resolved cases can identify new rules, data repairs, model-evaluation sets, training needs or policy clarifications. Learning should pass

through governance before changing production behaviour. Automatically learning from every human correction can absorb inconsistent or unauthorised decisions.

Figure 3. Exception and safe-state tree



An unresolved control condition cannot flow silently into a state-changing action.

11. Make safe state and rollback explicit

A safe state is the condition entered when the system cannot establish that continuing is acceptable. It may pause a transaction, preserve the last approved record, disable a tool, route to manual processing or use a tested deterministic service. The choice depends on consequence and continuity obligations.

Rollback needs more than a previous model version. The system may have changed records, sent messages or initiated external actions. Compensation procedures should identify which effects can be reversed, who can authorise reversal and how affected parties are informed. An immutable event history supports reconstruction.

Kill switches should be technically reachable and operationally owned. A control that requires unavailable credentials or an extended release cycle cannot support an urgent stop. The organisation should test suspension, fallback and recovery under realistic load and provider conditions, then retain the evidence.

Recovery criteria should be defined before an incident. Restoring service can require data reconciliation, model or rule validation, backlog review, customer communication and risk approval. A rushed restart can reintroduce the condition that caused the failure. The architecture decision should therefore include recovery effort and time, not only steady-state performance.

12. Evaluate the complete human-machine system

Model benchmarks provide useful component evidence, while production assurance must test the full workflow. Retrieval, prompts, tools, rules, permissions, interfaces, reviewers and operating conditions can change the result. Evaluation should use representative cases, difficult tails, adversarial inputs and known failure patterns.

Each control should have an observable test. A schema validator can be tested with invalid fields. A transaction limit can be tested at boundaries. A model can be evaluated for extraction, groundedness or abstention. A human-review process can be tested for decision quality, handling time and escalation. End-to-end tests should confirm that a failure enters the intended safe state.

Evaluation sets need provenance and versioning. Production incidents, appeals and exceptions can contribute cases after appropriate review and protection. The enterprise should prevent contamination between development and acceptance tests and should explain how the set represents expected use.

Acceptance should reflect consequence. A drafting assistant may tolerate a correction rate that is unacceptable for an automated payment decision. Thresholds should state the metric, cohort, confidence, sample and decision rule. Management should understand residual failure, including rare events that averages conceal.

13. Price latency, control and review together

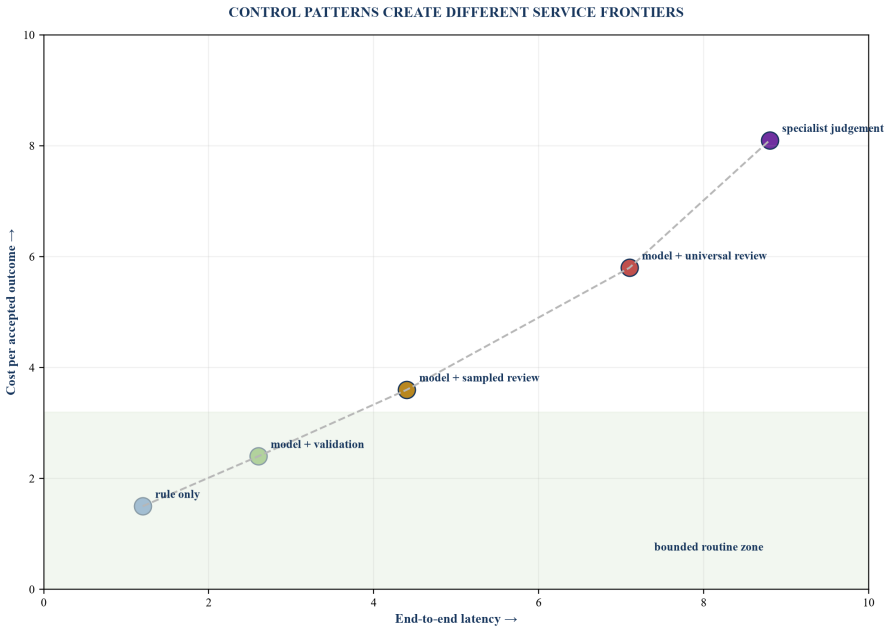
Every control adds some combination of compute, engineering, review and delay. The objective is an economically justified service within the approved risk envelope. Removing review can reduce unit time while increasing rework or loss. Adding universal review can make a low-risk workflow uneconomic and create queues that weaken control.

The team should measure time from event to accepted outcome. Model latency is one component. Retrieval, rule checks, tool execution, queueing, human handling, reconciliation and exception resolution can dominate. Percentiles and peak conditions matter for customer commitments and working capital.

Different architecture zones can serve different cases. Routine low-consequence cases may use deterministic automation. Variable but bounded cases may use a model with automatic validation. Consequential or ambiguous cases may enter specialist review. This segmentation concentrates expensive control where it creates value.

The economic model should include build, operation, assurance, vendors, data, security, human capacity, incidents and change. It should compare the current workflow and credible alternatives. Savings attributed to AI should remain separate from benefits created by policy simplification, data repair or process redesign.

Figure 4. Latency-cost frontier by control pattern



The preferred architecture sits inside the approved consequence envelope and minimises total accepted-outcome cost.

Table 3. End-to-end architecture economics

Cost or value domain	Measure	Evidence source	Typical owner
task completion	accepted outcomes and cycle time	workflow events and acceptance	operations
model service	tokens, calls and provisioned capacity	provider usage and invoices	engineering and finance
deterministic control	compute, licence and maintenance	platform cost and change records	technology
human oversight	handling, escalation and queue time	workforce and case records	operations
rework	correction, repeat and appeal	case outcomes and customer records	product
incidents	response, remediation and interruption	incident and finance records	risk and finance
protected value	avoided loss and retained service	documented scenario and observed events	accountable executive

Amounts shown in a business case should be reconciled to actual records; the categories are generally applicable.

14. Establish decision rights and separation of duties

Architecture governance should identify who owns the process, data, model, rules, security, legal interpretation, risk acceptance and production service. One person may hold several roles in a small organisation, while the decision record should remain clear. Material conflicts should be addressed through independent review or approval.

Developers should not unilaterally expand production authority. Product leaders should not waive security or legal controls without the designated owner. Reviewers should not approve their own access or payment. Separation of duties can be implemented through roles, workflow gates and immutable evidence rather than organisational size alone.

The approval pack should present the operating envelope, authority map, evaluations, known limitations, exception design, economics, monitoring and rollback. Conditions can include volume limits, customer cohorts, data types or mandatory review. The service owner should report when observed use approaches or exceeds those conditions.

Vendor responsibilities need the same clarity. A provider may operate a model or platform, while the enterprise remains responsible for the workflow, data, permissions and customer outcome. Contracts and service design should support evidence access, incident response, change notification, portability and applicable oversight duties.

15. Monitor authority, not only performance

Production monitoring should detect changes in what the system is doing and what it is allowed to do. Model quality and latency remain important. The control view also tracks tool calls, state changes, privilege use, overrides, exceptions, stop events and decisions outside the intended envelope.

Authority drift can occur when a new integration adds write access, a role changes, an approval is bypassed or a prompt encourages broader action. Dependency updates can alter behaviour without a model-name change. Monitoring should therefore compare actual action paths with the approved architecture.

Exception rates are leading indicators. Rising missing-data, low-confidence or policy-conflict cases may show source deterioration, product expansion or changed customer behaviour. Management should investigate the cause before increasing thresholds or suppressing alerts.

Human oversight also needs monitoring. Approval rates, handling times, disagreement, override reasons, queue age and reviewer concentration can reveal automation bias or capacity failure. Quality review should sample both approvals and rejections and should include the complex tail.

16. Apply regulatory and assurance obligations to the workflow

Regulatory obligations attach to context, role, data, outcome and jurisdiction. The EU AI Act uses a risk-based structure and includes requirements for human oversight, logging, documentation, accuracy, robustness and cybersecurity for relevant high-risk systems. Its Article 14 describes effective natural-person oversight and the ability, where appropriate, to disregard, override, reverse or interrupt an output.

The NIST AI Risk Management Framework organises activity through Govern, Map, Measure and Manage. Its core addresses human-AI configurations, oversight, component risks and contextual interpretation. The Generative AI Profile extends the framework for risks associated with generative systems. These publications support a lifecycle view rather than a one-time model test.

CBUAE guidance for licensed financial institutions focuses on responsible adoption, consumer protection, transparency, bias, accountability, explainability and privacy in relevant AI and machine-learning use. The UK Information Commissioner's guidance and risk toolkit connect AI processing to accountability, fairness, accuracy, security, minimisation, rights and impact assessment.

The applicable legal analysis must be performed for the actual use case. A generic architecture label cannot establish compliance. The system register should connect each workflow, jurisdiction, stakeholder and obligation to the responsible owner and retained evidence.

17. Worked example: supplier onboarding

Consider a hypothetical enterprise receiving 10,000 supplier applications per year. The current process uses email, manual extraction, policy checks and several system entries. Management is evaluating a hybrid architecture. The example values are assumptions used to demonstrate the method; they are not observed company data.

The proposed workflow uses a model to classify documents, extract names, ownership details, bank information and contractual terms, and draft a case summary with source citations. Deterministic services validate schema, match identity records, test bank-account changes, run approved screening services, apply value and geography rules, and prevent creation when mandatory evidence is absent.

Routine cases that satisfy all evidence and policy conditions can enter a sampled assurance route if the enterprise's legal and risk assessment permits it. Cases involving conflicting ownership, screening alerts, bank changes, policy ambiguity or high expected spend go to authorised reviewers. The model has read access to approved materials and cannot create a supplier, change bank details or release a payment.

Management compares task completion, exception mix, review time, first-pass acceptance, false clearance, false escalation, incidents and full operating cost. The proposed design is approved only for defined document types and jurisdictions. Expansion requires new evaluation and a revised authority decision.

Table 4. Hypothetical supplier-onboarding allocation

Workflow step	Proposed mechanism	Model authority	Acceptance condition
document intake	deterministic gateway	none	identity, file and malware checks pass
data extraction	model with citations	propose fields	schema and source links validate
identity match	deterministic service	none	approved records agree
policy interpretation	model-assisted review	recommend	authorised owner resolves ambiguity
screening	approved deterministic service	none	result and timestamp recorded
risk summary	model draft	recommend	evidence complete and reviewer rule met
supplier creation	deterministic transaction	none	all approvals and separation checks pass
monitoring	rules plus anomaly model	flag	exception route and accountable owner

The volumes and control thresholds are illustrative analytical assumptions.

18. Stress test correlated failure

Hybrid systems can fail through combinations that component tests miss. A provider change can alter extraction, causing more exceptions at the same time as a reviewer shortage. A corrupted source can be retrieved consistently and then pass a superficial citation check. An identity-service outage can tempt staff to bypass a deterministic gate.

Stress tests should combine plausible operational events: peak volume, degraded model quality, unavailable tools, delayed human review, partial logs, credential compromise and conflicting records. The test should measure whether the system contains the event, enters the correct safe state, preserves evidence and restores service within the approved objective.

Concentration should be visible. Several workflow components may depend on one cloud region, identity service, model provider or specialist team. Apparent redundancy can share a common dependency. The architecture map should identify these links and decide whether diversity, offline capability or a manual continuity procedure is economically justified.

Stress results should feed capital and liquidity planning where interruption or error can create material cash effects. The scenario should separate gross exposure, probability assumptions, control effectiveness and recovery cost. Assumptions should be documented and reviewed rather than presented as forecasts.

19. Use an architecture decision scorecard

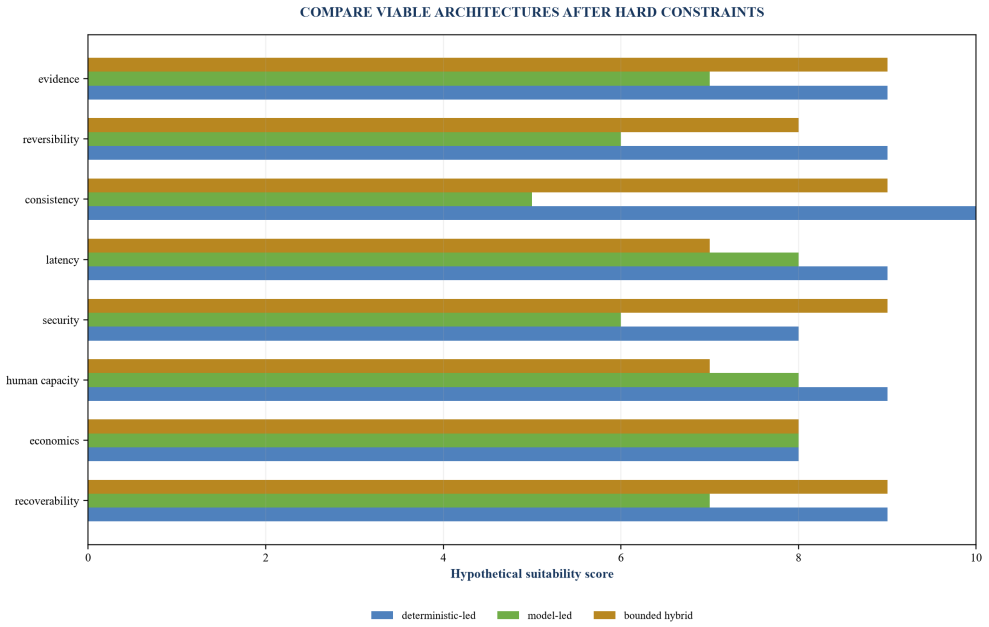
A scorecard creates a consistent conversation across business, risk and technology. It should not mechanically replace judgement. The dimensions can include consequence, ambiguity, evidence quality, reversibility, required consistency, latency, scale, explainability, security, human capacity and economics.

Each proposed architecture should show the source and confidence of its score. Observed production data carries different weight from a vendor demonstration or management estimate. Material unknowns should create a condition, pilot or evidence request rather than disappear inside an average.

Hard constraints should sit outside the weighted score. An architecture that breaches a legal duty, lacks required authority, permits an unsafe action or cannot produce necessary evidence should not pass because it scores well on cost and speed. The scorecard helps select among viable alternatives after those conditions are met.

The decision should include a review date and trigger events. New jurisdictions, data categories, customer groups, actions, providers, models or incident patterns can change the allocation. Governance is strongest when reassessment is linked to observable changes rather than an arbitrary annual presentation.

Figure 5. Hybrid architecture decision scorecard



Hard constraints are assessed before weighted operating dimensions.

Table 5. Architecture approval scorecard

Dimension	Key question	Evidence	Decision effect
consequence	what harm or loss can a wrong action create at scale?	incident, legal and financial analysis	limits model authority
ambiguity	how much contextual judgement is unavoidable?	case review and policy analysis	determines human role
evidence	can the system establish material facts and lineage?	source and data-quality tests	sets abstention threshold
reversibility	can effects be contained and corrected promptly?	rollback and recovery test	determines preventive control
consistency	must identical inputs receive the same rule outcome?	policy and obligation mapping	favours deterministic gate
human capacity	can reviewers act effectively within the service level?	workload and quality evidence	sizes review route
economics	what is the full cost per accepted outcome?	reconciled cost and workload model	compares viable patterns
recoverability	can the service stop, fallback and reconcile?	resilience exercise	conditions production approval

Evidence quality and hard constraints are recorded separately from weighted scores.

20. Implement through bounded releases

The first release should target a narrow workflow segment with reliable data, clear acceptance and limited consequence. It should run against a measured baseline and retain enough manual control to compare outcomes. The release plan should identify entry and exit criteria, maximum volume, permitted users and stop conditions.

Shadow operation can test interpretation without granting action authority. The system produces a proposed result while the existing process remains authoritative. Differences create an evaluation

set and reveal policy, data and interface issues. A shadow result should not influence the operator unless the test design explicitly allows it.

Authority can expand in stages. A model may begin as a draft assistant, then support recommendations, then allow bounded automatic processing for a demonstrated routine cohort. Each stage needs evidence for its additional authority. Volume growth alone does not justify expansion.

The implementation backlog should separate data repair, policy clarification, deterministic controls, model work, interface design, operating capacity and assurance. This prevents the model team from being held responsible for every weakness and helps management see which investment creates value.

21. Board and investment-committee questions

The board should ask which actions the AI system can take, the maximum exposure before detection and the evidence supporting that authority. It should understand which controls remain deterministic, where human judgement is mandatory and how the workflow behaves when data, models, tools or people are unavailable.

Management should present economics per accepted outcome, including exceptions, review, assurance and incidents. A component-level accuracy or productivity result cannot establish an enterprise return. The decision pack should show the current baseline, viable alternatives, assumptions, sensitivity and residual exposure.

The committee should also ask how authority changes. Vendor updates, new tools, prompt changes and role permissions can expand behaviour. Release governance, monitoring and periodic access review should make those changes visible. Contracts should support the evidence and intervention required by the operating model.

Approval should state conditions and owners. The record should include the use case, operating envelope, authority limits, evaluation, stop conditions, reporting, review date and responsible executives. This creates a practical mandate for teams and a basis for later assurance.

22. A 90-day architecture programme

During days 1 to 30, the enterprise should select one material workflow, map decision steps, identify state changes, document current controls and establish baseline outcomes. Legal, risk, data, security, operations, finance and technology owners should agree the consequence and evidence framework.

During days 31 to 60, the team should design two or more viable allocations, build the deterministic control plane, define model contracts, create evaluation sets and specify exceptions, oversight and safe state. The economic model should include current and proposed full costs.

During days 61 to 90, the organisation should run shadow or bounded production, test correlated failures, measure reviewer behaviour and reconcile outcomes. The approval body should receive an evidence pack with observed results, unresolved gaps, architecture alternatives and conditions for the next stage.

The programme ends with a decision, not an assumption of scale. Management may expand, redesign, retain the existing process or stop. A disciplined stop can preserve capital when data, policy or economics do not support the intended authority.

23. Limitations and research agenda

The framework is cross-sectoral and cannot replace use-case legal, safety, cybersecurity or professional analysis. Determinism can be difficult to establish across distributed systems, third-party services and changing data. Human decisions also contain inconsistency and bias; retaining a human does not by itself create a strong control.

The worked example is hypothetical. Its volumes, scores and control allocations are not empirical claims. Real architecture decisions require observed workflow data, representative evaluation, contractual review and accountable risk acceptance.

Further research should examine how formal methods, automated reasoning, constrained decoding and verifiable computation can support deterministic boundaries around generative systems. More evidence is also needed on reviewer effectiveness, automation bias, exception economics and correlated failures in deployed agentic workflows.

Enterprises should publish or share incident and evaluation evidence where confidentiality and law permit. Comparable evidence would improve architecture decisions beyond provider-specific demonstrations and help investors distinguish durable operating capability from experimental adoption.

24. Govern data lineage as part of architecture

Hybrid control depends on knowing which data informed a decision. The architecture should distinguish systems of record, approved reference material, user-provided content, derived fields and model-generated text. These categories carry different reliability, access and retention requirements. A fluent synthesis should never erase the difference between a verified record and a model proposal.

Lineage should survive transformation. When a document is parsed, chunked, embedded, retrieved and summarised, the workflow should retain a path to the original source, its version and relevant location. If a model extracts a field, the evidence record should identify the source passage and validation outcome. This supports review, correction, dispute handling and assurance.

Data quality rules belong close to the source. Required fields, formats, duplicates, stale records and inconsistent identifiers should be tested deterministically where possible. A model can help interpret variation, while it should not silently repair a material fact without marking the proposed change for verification. Source owners remain accountable for correction.

Retention should reflect purpose and obligation. Keeping every prompt and response can increase privacy, security and cost exposure. Keeping too little can prevent investigation or explanation. The design should identify the minimum evidence needed for the workflow, control and applicable record duty, with access and deletion enforced through policy.

Data residency and cross-border transfer can affect provider and architecture choice. The use-case review should map where inputs, logs, embeddings, model processing and backups reside.

Contractual statements should be confirmed against deployed configuration and observed data flow. Any change in region, provider or feature should trigger a lineage and rights review.

25. Defend the boundary against hostile or misleading inputs

Model-assisted workflows can receive instructions embedded in documents, web pages, messages or tool results. Prompt injection and data poisoning can attempt to redirect the model, expose information or invoke an unauthorised action. The deterministic control plane should treat external content as data, not as authority to change system policy.

System instructions, tool permissions and business rules should be separated from retrieved content. The model should receive only the tools and data required for the current step. Action parameters should be validated independently, and sensitive operations should require an approval or service token that the model cannot manufacture.

Content filters alone cannot establish security. Attackers can vary language, encoding and sequence. Defence should combine input handling, provenance, least privilege, isolation, output validation, monitoring and incident response. High-consequence tools may need a deterministic allowlist and transaction-specific constraints even when the model recommendation appears reasonable.

Testing should include indirect attacks through realistic enterprise material. A supplier document, customer email or retrieved webpage can contain instructions that conflict with the workflow. The expected result should be observable: ignore the instruction, flag the content, preserve evidence and prevent prohibited tool use. Red-team findings should enter release and regression tests.

Security monitoring should connect model events with identity, network, data and transaction logs. A suspicious prompt becomes more important when followed by unusual retrieval or tool calls. Correlation allows the enterprise to detect an attempted chain rather than review each event in isolation. The recovery plan should cover credential rotation, evidence preservation, affected-record review and controlled restoration.

26. Control third-party models and services through contracts and evidence

Many hybrid systems depend on external models, cloud services, retrieval platforms, screening tools and data providers. The architecture should identify which control claim depends on each supplier and what evidence the enterprise can obtain. A contractual promise has limited operating value if the deployed service cannot produce the required logs, regions or intervention rights.

Due diligence should cover capability, security, privacy, data use, model and service changes, availability, subcontractors, intellectual property, incident notification, support, exit and evidence. The depth should follow workflow consequence. A drafting assistant and a service supporting a high-impact customer decision should not receive identical diligence.

Change management is critical. Providers can update models, safety layers, limits, pricing, regions or interfaces. The enterprise should know which changes are announced, which require acceptance and which can occur automatically. Evaluation gates and version pinning should be used where available and proportionate. Unannounced behaviour changes should be detectable through monitoring.

Exit design should preserve data, prompts, rules, evaluations, interfaces and operating knowledge. Portability is not limited to replacing an API endpoint. A successor system must reproduce the workflow's accepted outcomes and control evidence. The contract and architecture should support extraction, transition testing, parallel operation and deletion confirmation.

Concentration should be measured at the workflow and portfolio levels. Different applications may depend on the same identity, cloud region, model family or specialist supplier. Procurement records alone can miss this technical concentration. An enterprise component register supports scenario analysis and informs whether a fallback, second provider or manual continuity route is justified.

27. Standardise patterns without forcing every use case into one design

An enterprise can accelerate delivery by publishing approved hybrid patterns. Examples include model-assisted drafting with human release, extraction with deterministic validation, recommendation with authorised decision, and autonomous low-consequence processing with sampled assurance. Each pattern should state suitable conditions, required controls and prohibited uses.

Reusable components can include gateways, identity, retrieval, policy services, schema validation, logging, evaluation, review interfaces and safe-state orchestration. Standard components reduce duplicate engineering and produce comparable evidence. They should remain configurable for jurisdiction, consequence and workflow needs.

A central platform should avoid becoming an uncontrolled universal agent. Teams need a clear way to request tools, data and authority, with review proportional to risk. The platform should make the secure path easier through templates, automated tests and observable controls. Exceptions to the standard should be documented and time-bound.

Portfolio governance should classify applications by authority and consequence. A catalogue that records only the model name provides little insight. Management should be able to see which systems read sensitive data, change records, communicate externally, influence customers, use human approval or depend on a particular provider.

Metrics should support comparison across patterns. Accepted outcome, exception rate, reviewer load, incident rate, latency and complete cost are more useful than a single adoption count. The portfolio view can identify patterns that perform well, controls that create bottlenecks and use cases that should be retired. Standardisation then becomes an evidence programme rather than a technology mandate.

28. Include hybrid architecture in transaction diligence

AI architecture can affect the value and risk of a target company. A buyer should determine whether claimed automation is a bounded operating system or a collection of model calls supported by hidden manual work. The diligence scope should map authority, data rights, evaluation, incidents, third-party dependence, economics and customer obligations.

Revenue quality can depend on model and control performance. A product may require costly review, support or provider capacity that is absent from the reported gross margin. Cohort-level unit economics should reconcile customer revenue to the complete workflow cost. Material

credits, committed spend and unrecorded human effort should be examined.

Technology diligence should test reproducibility and change control. The buyer should identify model, prompt, retrieval, rule and tool versions; inspect evaluation and release evidence; and determine whether the team can restore or migrate the service. A demonstration cannot establish production reliability or portability.

Regulatory and contractual diligence should connect the actual workflow to affected stakeholders and jurisdictions. Data use, automated decisions, disclosures, audit rights, service levels and incident duties can create integration conditions. Remediation cost and time should enter valuation, transaction documents and the post-close plan where material.

The value-creation plan can use the control-allocation framework to separate scalable automation from necessary expert work. Strong deterministic gates and evidence can allow safe expansion of model assistance. Weak foundations may require data, policy and platform investment before growth. The transaction thesis should therefore link architecture maturity to a funded operating roadmap.

29. Translate architecture quality into strategic value

A well-designed hybrid system can create value through faster accepted outcomes, lower rework, controlled scale, better evidence and more resilient service. These effects should be measured in operational and cash terms. Adoption counts and model usage are activity indicators; they do not establish value by themselves.

Revenue benefits may arise from shorter onboarding, improved service, higher conversion or a product that customers trust for consequential work. Cost benefits may arise from automation, fewer corrections, better triage and reusable controls. Risk benefits may include reduced exposure or faster recovery. Each benefit needs a baseline, owner, evidence source and attribution method.

Architecture quality can also preserve strategic options. Bounded interfaces, portable data, versioned policies and tested fallbacks make it easier to change providers, integrate acquisitions or enter a new jurisdiction. The option has value when management can identify the avoided transition cost or increased commercial flexibility.

Investment governance should fund the complete operating system. A model licence without data repair, deterministic controls, review capacity, security and assurance can produce an undercapitalised programme. The business case should stage funding against evidence and make unresolved dependencies visible.

The strategic review should also identify uses that should remain manual or rule-based. Capital is created by selecting the right architecture for each decision, including a decision not to use a model. A portfolio that concentrates AI where it improves accepted outcomes and retains stronger mechanisms elsewhere can generate more durable value than broad, weakly controlled deployment.

Conclusion

Hybrid AI architecture is the disciplined allocation of interpretation, authority and accountability. Rules should govern conditions the enterprise can specify, models should handle bounded variation where they add distinctive value, and people should exercise informed judgement where consequence or ambiguity requires it.

The practical unit is the workflow decision step. Decomposition exposes state changes, evidence gaps and hidden authority. A deterministic control plane, bounded model contracts, effective human oversight, explicit exceptions and tested safe state convert that map into an operating system.

Value is demonstrated through accepted outcomes, service performance, full cost and controlled exposure. A staged evidence programme allows authority to expand only when the complete human-machine system supports it. This creates an architecture that can scale useful AI while preserving the repeatability, accountability and recovery on which enterprise operations depend.

References

[1] European Union. Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence. https://eur-lex.europa.eu/eli/reg/2024/1689

[2] European Commission. AI Act regulatory framework. https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai

[3] European Commission. Guidelines on obligations for general-purpose AI providers. https://digital-strategy.ec.europa.eu/en/faqs/guidelines-obligations-general-purpose-ai-providers

[4] National Institute of Standards and Technology. AI Risk Management Framework. https://www.nist.gov/itl/ai-risk-management-framework

[5] NIST AI Resource Center. AI RMF Core. https://airc.nist.gov/airmf-resources/airmf/5-sec-core/

[6] National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile, NIST AI 600-1. https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf

[7] National Institute of Standards and Technology. Secure Software Development Framework. https://csrc.nist.gov/Projects/ssdf

[8] UK National Cyber Security Centre and international partners. Guidelines for secure AI system development. https://www.ncsc.gov.uk/collection/guidelines-secure-ai-system-development

[9] UK National Cyber Security Centre. Secure design guidance. https://www.ncsc.gov.uk/collection/developers-collection/principles/secure-design

[10] Cybersecurity and Infrastructure Security Agency. Secure by Design. https://www.cisa.gov/securebydesign

[11] Central Bank of the UAE. Guidance Note on the Consumer Protection and Responsible Adoption and Use of Artificial Intelligence and Machine Learning by Licensed Financial Institutions in the U.A.E. https://rulebook.centralbank.ae/en/rulebook/guidance-note-consumer-protection-and-responsible-adoption-and-use-artificial-intelligence

[12] Central Bank of the UAE. Guidelines for Financial Institutions adopting Enabling Technologies. https://www.centralbank.ae/en/our-operations/fintech-digital-transformation/

[13] Dubai Financial Services Authority. AI Survey 2025. https://www.dfsa.ae/news/new-dfsa-ai-survey-generative-ai-adoption-has-nearly-tripled-within-difc-last-12-months-governance-continues-develop

[14] UK Information Commissioner's Office. Guidance on AI and data protection. https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/artificial-intelligence/guidance-on-ai-and-data-protection/

[15] UK Information Commissioner's Office. AI and data protection risk toolkit. https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/artificial-intelligence/guidance-on-ai-and-data-protection/ai-and-data-protection-risk-toolkit/

[16] OECD. OECD AI Principles. https://oecd.ai/en/ai-principles

[17] AWS. Responsible AI Lens. https://docs.aws.amazon.com/wellarchitected/latest/responsible-ai-lens/responsible-ai-lens.html

[18] AWS. Generative AI Lens. https://docs.aws.amazon.com/wellarchitected/latest/generative-ai-lens/generative-ai-lens.html

[19] AWS. Identify human oversight opportunities. https://docs.aws.amazon.com/wellarchitected/latest/responsible-ai-lens/raiuc04-bp02.html

[20] Microsoft. Azure Well-Architected Framework guidance for AI workloads. https://learn.microsoft.com/en-us/azure/well-architected/ai/

[21] Microsoft. Responsible AI Standard. https://www.microsoft.com/en-us/ai/principles-and-approach

[22] Google. Secure AI Framework. https://saif.google/

[23] Google Cloud. Responsible AI. https://cloud.google.com/responsible-ai

[24] OWASP Foundation. Top 10 for Large Language Model Applications. https://genai.owasp.org/llm-top-10/

[25] Bank of England and Financial Conduct Authority. Artificial intelligence in UK financial services survey. https://www.bankofengland.co.uk/report/2024/artificial-intelligence-in-uk-financial-services-2024

About the Author

Authored by Chennakeshav Adya, Independent Researcher

Chennakeshav (CK) is a corporate finance and investment banking executive with 25+ years of global experience in deal origination, structuring and execution across M&A, growth capital and corporate strategy. He has led value-creation mandates for founders, corporates and funds; bridging the boardroom view to hands-on execution and close.

His career spans Morgan Stanley, HSBC, Lloyds Banking Group, EWEC, ADQ portfolio companies and Emirates Growth Fund, across TMT, real estate, fintech, deeptech, cleantech, infrastructure and energy. He has partnered with C-suite leaders, private equity and venture funds, sovereign wealth funds and family offices to finance complex fund raises and scale-up ventures, and has led M&A due diligence, post-merger integration and business-transformation initiatives to create value.

At Matchpoint Partners he is Managing Partner, leading the firm's corporate finance, M&A and capital-raising practice. He holds an MBA from London Business School, an engineering degree from VTU and a Master of Laws (LLM, in progress) from UCL London.

An active start-up mentor, CK mentors at Techstars, DIFC FinTech Hive, Startup Grind, Founder Institute and IN5, serves as Entrepreneur Mentor in Residence (EMiR) at London Business School, and judges the Entrepreneurship World Cup.