

A Five-Engine Reference Architecture for Cost-Aware, Self-Adaptive Agentic AI

An engineering case study in the Internet of Value

Basel Magableh^{1,*} OmniRisk Research²

¹School of Computer Science, Technological University Dublin, Ireland

²Rayachain Lab, Dublin, Ireland

*Correspondence: basel.magableh@tudublin.ie

May 2026 (draft v0.4 — targeting Wiley *Software: Practice and Experience*)

Abstract

Agentic AI systems are increasingly deployed in heterogeneous, partially-trusted environments where the marginal risk that matters is *composite*: route, sentiment, liquidity, and the policy a system is willing to commit to. The software-engineering literature on machine-learning systems [1, 2] addresses an offline-train-then-deploy regime, surveys of LLM-based autonomous agents [3–5] catalogue what such agents are made of, and the self-adaptive software literature [6–8] gives a venerable vocabulary for systems that monitor and reconfigure themselves — but no single thread of that literature tells a practitioner how to engineer *one* agent that stays cheap and stays calibrated for months under adversarial inputs. This paper reports the engineering — not the algorithms — of an agentic risk-intelligence platform deployed on the blockchain Internet of Value, and proposes three contributions. **(C1) A five-engine reference architecture** composed of (E1) a confidence-cascading prediction router, (E2) an independent verification stage, (E3) a multi-modal sentiment- fusion engine over text and on-chain flow, (E4) a reason-and-act agent under explicit constraints, and (E5) a scenario / API risk engine that externalises every action as a pre-committed scenario program. To our knowledge, this composition has not appeared as a single architecture in the peer-reviewed literature; §3.6 states the novelty claim falsifiably. **(C2) A cost-aware AI engine** that emerges from the composition (E1+E2+E4) and yields a per-decision cost ledger combining inference, trust, and action costs in the lineage of attentional cascades [9] and prediction-serving systems [10, 11]. **(C3) A self-adaptive loop** that emerges from E4 closing MAPE-K over E1, E3, and E5 on a versioned calibration manifest, with a non-negotiable honesty discipline. We report two operational case studies as evidence and close with a unifying *cost-and-honesty* thesis: four falsifiable engineering propositions, each with an explicit failure criterion and transferability scope.

Keywords — software architecture; software engineering for AI; cost-aware inference; self-adaptive software; MAPE-K; agentic systems; LLM operations; risk intelligence; case study.

1 Introduction

The software-engineering literature on machine-learning-intensive systems has, over a decade, settled an uncomfortable consensus: most of the cost is not in the model. Sculley et al. [1] catalogued the system-level burdens (entanglement, hidden feedback loops, undeclared consumers, configuration debt, monitoring debt) that ML systems pay over and above ordinary code, and Amershi et al. [2] documented at industrial scale the workflow discontinuities that ML imposes on traditional SE practice. Subsequent work on LLM-based autonomous agents [3–5] has intensified

each of those burdens: an agent is no longer only a predictor, it is a closed-loop controller that selects and executes actions, and its decisions feed back into the data it later consumes.

Two operational concerns then dominate practice. The first is *cost*: every inference and every action is metered, so the system must decide which queries deserve which model, and which decisions deserve to be checked by an independent verifier. The cascade pattern that solves an analogous problem in computer vision [9] and in production prediction-serving systems [10, 11] is the natural starting point but is not, on its own, sufficient: a cascade reduces inference dollars but does not re-price the cost of being wrong. The second is *adaptation*: the environment shifts faster than any offline-trained model can keep up with, so the system must monitor itself, analyse its own residuals, and reconfigure — which is the remit of self-adaptive software, formalised as the MAPE-K loop [6–8].

This paper concerns the engineering of one such system, deployed in a genuinely adversarial setting: an agentic risk-intelligence platform operating on the blockchain “Internet of Value” (IoV), a heterogeneous network of chains, bridges, liquidity venues, and off-chain narratives in which the dominant marginal risk to any participant is *composite* [12–14], not a property of any single chain. We argue that meeting the two operational concerns (cost and adaptation) under adversarial inputs requires a particular architectural decomposition — not as metaphor, but as an enumerated, named-component reference architecture in the spirit advocated by the SE-for-self-adaptive-systems roadmap [7].

Contributions.

- (C1) **A five-engine reference architecture** (§3) composed of a prediction router (E1), an independent verification stage (E2), a multi-modal sentiment-fusion engine (E3), a reason-and-act agent under explicit constraints (E4), and a scenario / API risk engine (E5). Each engine operationalises a peer-reviewed engineering primitive; the *composition* is, to our knowledge, novel (§3.6).
- (C2) **A cost-aware AI engine** (§4) emerging from E1+E2+E4: a per-decision cost ledger combining inference, trust, and action costs.
- (C3) **A self-adaptive agentic AI** (§5) emerging from E4 closing MAPE-K over E1, E3, and E5 on a versioned calibration manifest, with a non-negotiable honesty discipline.

A synthesising *cost-and-honesty thesis* (§7) distils C1–C3 into four falsifiable engineering propositions.

Operational evidence. We report two operational case studies that exercise C2 and C3 on real production runs (§6): a 27-hour Solana liquidity stress experiment and a 168-hour prediction-router calibration arc. Neither is a controlled experiment; both are operational reports whose limits are stated honestly, and we deliberately decline to cite internal logs as academic references.

Scope. This is the software-engineering cut: composition, cost, adaptation, operability, and the discipline of being calibrated. A full threat model and validator-loss decomposition belong elsewhere and are out of scope here. References are restricted to academically- published peer-reviewed work; engineering artefacts are described in prose without bibliographic citations.

2 Theoretical background

2.1 Software engineering for ML and for LLM agents

Sculley et al. [1] catalogued the system-level costs that ML systems pay over and above ordinary code — entanglement of features, hidden feedback loops between deployed models and the data

they generate, undeclared consumers of model outputs, configuration debt, monitoring debt — and argued that these costs dominate the running engineering bill for ML in production. The empirical follow-up by Amershi et al. [2], drawing on a multi-team case study at Microsoft, identified a nine-stage ML workflow and showed three discontinuities at which traditional SE practice fails: data management, model evaluation, and model deployment. The prescription, in both works, is to make the ML workflow a first-class engineering artefact — versioned, auditable, and monitored — rather than a side-channel of model code.

LLM-mediated autonomous agents complicate this prescription. Wang et al. [3] survey the field and propose a unified construction framework around four subcomponents: *profile*, *memory*, *planning*, and *action*. The reason-and-act paradigm [4] interleaves chain-of-thought and tool calls so that the planning step is itself observable; the generative-agents architecture [5] extends this with explicit memory streams and periodic reflection. Two qualitatively new SE concerns arise: the agent’s outputs are *actions* that perturb the environment from which its next inputs are drawn (the system is therefore a closed-loop controller, not only a predictor); and the agent is constituted in part by natural-language constraints (profile prompts, planning prompts, action policies) which must be diffable and reviewable like any other configuration artefact. Both concerns place the engineering of LLM agents within reach of an older intellectual lineage: self-adaptive software.

2.2 Self-adaptive software and the MAPE-K formalism

The self-adaptive software literature anticipated, two decades before LLM agents existed, the engineering concerns that LLM agents now demand. Salehie and Tahvildari’s survey [6] formalises the MAPE-K loop — monitor, analyse, plan, execute, over a shared knowledge base — as the canonical decomposition of any system that observes itself and reconfigures. The Dagstuhl-seminar research roadmap [7] extends this with a four-dimensional modelling vocabulary (modelling dimensions, requirements, engineering, assurance) and identifies the open challenges of trust, uncertainty, and assurance under runtime change. Weyns’ textbook [8] treats the same material contemporarily, integrating uncertainty management, feedback control, and online learning, and explicitly addresses the engineering implications of using machine-learning components inside a MAPE-K adaptation loop.

The contemporary relevance of MAPE-K to agentic AI is not metaphorical: the Monitor stage corresponds to telemetry from prediction and sentiment inputs; Analyse to residual analysis against a calibration manifest; Plan to the LLM-mediated reasoning step of a reason-and-act agent [4]; Execute to a constrained action that writes to a versioned configuration; Knowledge to the calibration manifest and action policy. Three engineering concerns from the self-adaptive literature carry over directly: *decoupling* the four MAPE-K stages so that each can fail independently [7]; *assurance* via runtime evidence rather than design-time proof [8]; and *uncertainty management* as a first-class concern, not a side-channel of model output [6].

2.3 Cost-aware inference: from cascades to serving systems

The cost-aware-inference literature is older than its current LLM manifestation. Viola and Jones [9] established the attentional-cascade pattern: a sequence of classifiers of increasing complexity is arranged so that easy negatives are rejected by cheap stages and only hard cases reach the expensive ones, reducing average inference cost without reducing detection quality. Kang et al. [11] extend the pattern to deep neural networks for video query workloads, using specialised difference detectors and confidence-thresholded escalation to obtain orders-of-magnitude cost reductions while preserving accuracy guarantees. Crankshaw et al. [10] present Clipper, a prediction-serving system that abstracts model selection and ensembling behind a latency-budgeted API, and demonstrate that cost-and-quality tradeoffs are best made operationally rather than at training time. The shared engineering claim — that the right unit

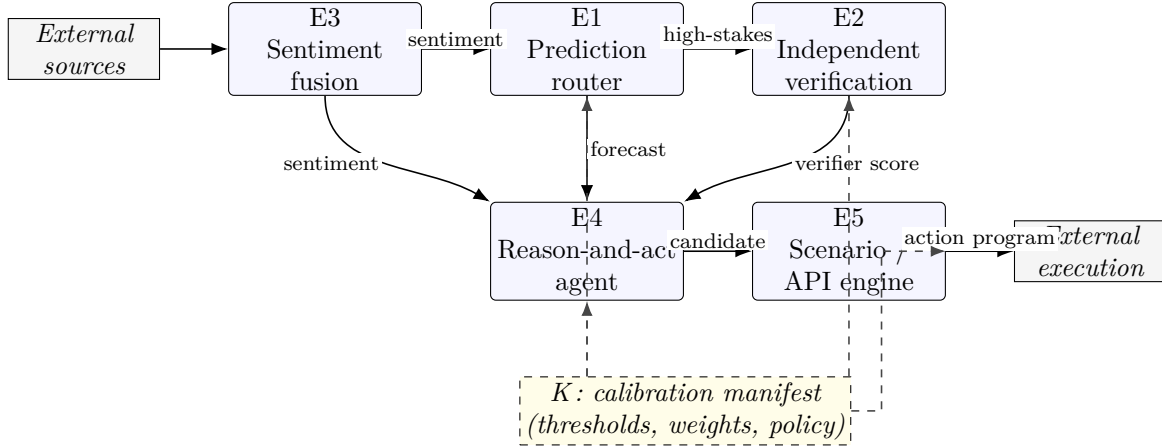


Figure 1: The five-engine reference architecture (C1, §3). Solid arrows show the per-decision dataflow: sentiment fusion (E3) and prediction routing (E1) feed the reason-and-act agent (E4); high-stakes forecasts are repriced through the independent verification stage (E2); the agent’s candidate is externalised by the scenario / API engine (E5). The dashed arrows from the calibration manifest (K) indicate the shared, versioned knowledge that the self-adaptive loop of §5 writes to. The composition supports both the cost-aware property of §4 (E1+E2+E4 jointly carry a per-decision cost ledger) and the self-adaptive property of §5 (E4 closing MAPE-K over E1, E3, E5 on K).

of analysis for cost is a cascade or ensemble managed as a first-class artefact — is the foundation on which the cost-aware engine of §4 builds.

2.4 Risk intelligence in the Internet of Value

The IoV setting motivates the case studies but is not the object of this paper, and its summary here is brief. Cross-chain bridges have become both the dominant capital route across heterogeneous chains and the dominant exploit surface [12–14]; sentiment shocks propagate at a faster cadence than chain-state changes, so a monitor that ignores text drift is blind to events whose on-chain footprint is downstream of the narrative [15–17]; multimodal sentiment fusion — combining text, structured features, and on-chain flow — is the contemporary technical response, with established taxonomies for early, late, and stacked fusion across heterogeneous modalities [18–20]; and pre-committed scenario-conditional decision-making is the standard discipline for translating uncertain forecasts into reviewable action programs [21].

3 (C1) The five-engine reference architecture

The system’s first contribution is a reference architecture composed of five named, independently-testable engines. Figure 1 summarises the per-decision dataflow and the shared calibration manifest that the self-adaptive loop of §5 writes to. We describe each engine in turn, naming its purpose, the peer-reviewed primitive it operationalises, and the distinctive engineering features that differentiate it from a textbook deployment of that primitive. We then state the novelty claim — that the *composition* of all five has not appeared in the peer-reviewed literature — falsifiably in §3.6.

3.1 E1 — Confidence-cascading prediction router

Purpose. Map a query (a symbol, an interval, an action context) to a forecast, with explicit confidence and explicit cost.

Primitive. The attentional-cascade pattern [9] extended to deep models [11] and online prediction serving [10].

Distinctive features.

- *Multi-tier ensemble:* cheap heuristic baselines, mid-tier ML models, and expensive LLM-mediated forecasters arranged in cost order. Acceptance is conditioned on confidence-interval width *and* agreement with a frozen calibration manifest.
- *Per-symbol acceptance thresholds:* thresholds are not global; they are versioned per asset and per horizon, so that a cascade that is calibrated on a mature symbol does not contaminate a thin-data symbol.
- *Self-reporting:* the router publishes its escalation rate and miss rate alongside its forecasts, so that downstream consumers can audit the cascade’s discipline. This is the engineering analogue of the SE-for-ML prescription [1, 2] that evaluation be a first-class artefact, raised to the component-API boundary.
- *Class-imbalance honesty at the API:* every accuracy metric the router emits carries the class-imbalance denominator that produced it; downstream stages cannot quote it without quoting the denominator.

3.2 E2 — Independent verification stage

Purpose. Re-price a high-stakes inference by subjecting it to an independent, economically-incentivised judge.

Primitive. The ensembling-for-independence argument familiar from prediction-serving systems [10], raised to the level of an architectural component rather than an internal model trick.

Distinctive features.

- *Economic independence:* validators are incentivised on a signal that is *not* the optimiser’s loss, so the two are not co-trained on the same distribution. This addresses the co-located-optimiser-verifier failure mode of E1 ensembling directly.
- *Fail-closed gate:* when verifiers disagree with the cascade’s accepted output above a configurable margin, the system refuses the action. The default for an agentic system in an adversarial environment is non-action, not action.
- *Verification cost as a first-class budget term:* the cost of paying for an independent judge is recorded in the per-decision cost ledger of \$4.1, not externalised as an operational concern.
- *Diversity by construction:* validators are deliberately drawn from heterogeneous training distributions so that their residual-correlation with E1 is bounded; the composition’s cost-aware property (C2) depends on this diversity being engineered, not assumed.

3.3 E3 — Multi-modal sentiment-fusion engine

Purpose. Convert a stream of heterogeneous textual and structured signals into a sentiment vector that the prediction router (E1) and the agent (E4) can consume.

Primitive. The multimodal-sentiment-fusion lineage: text-with-price fusion for financial forecasting [15–17], deep-learned sentiment with feature engineering [19], contemporary multimodal-fusion taxonomies [18], and recent stacking-LSTM architectures for crypto-asset sentiment [20].

Distinctive features.

- *On-chain flow as a fusion modality:* the engine treats on-chain transaction-flow signals (volume, flow asymmetry, liquidity reshuffling) as a peer modality alongside text and

structured market features, extending the peer-reviewed text+price-fusion lineage to text+price+on-chain-flow. This addresses the IoV-specific concern that on-chain-state changes are downstream of, but causally independent from, narrative shifts [15, 16].

- *Late-fusion stacking with per-source noise-floor weighting*: sources are not averaged naively; per-source weights are learned to reflect that different sources have different noise floors and timescales, in line with the late-fusion-with-meta-model pattern endorsed by the contemporary fusion survey [18, 20].
- *Provenance tags*: each fused output carries timestamps and source-identity tags so that a downstream consumer can reproduce the engine’s output given only its trace — the SE-for-ML reproducibility prescription [1] applied to streaming sentiment.

3.4 E4 — Reason-and-act agent under explicit constraints

Purpose. Map prediction-and-sentiment state, plus the agent’s internal memory, to a candidate action under explicit budget and policy constraints.

Primitive. The reason-and-act paradigm [4], the generative-agents memory-and-reflection architecture [5], and the profile/memory/planning/action decomposition advocated by the LLM-agent survey [3].

Distinctive features.

- *Per-step token budget*: each reasoning step is bounded in tokens, so that adversarial prompts cannot run the system out of money. The budget is part of the cost ledger.
- *Role-bound permission scopes*: the agent’s permission to call any external tool is tied to a role manifest, so that a planning step that would breach role authority is blocked before any tool call is emitted.
- *Versioned profile and planning prompts*: the natural-language constituting parts of the agent are change-controlled artefacts, in line with the assurance-under- runtime-change concern named in the self-adaptive systems literature [7, 8].
- *Memory and reflection as engineered substrate*: the agent maintains an explicit memory store and a periodic reflection step in the Park-et-al. tradition [5], but the memory is itself a versioned artefact, not a model-internal latent.

3.5 E5 — Scenario / API risk engine

Purpose. Translate the agent’s candidate action into a pre-committed scenario-conditional API call — an externally reviewable artefact that another component, or a human operator, can inspect before any external execution.

Primitive. Scenario-based Monte Carlo decision analysis [21], used here as an externalisation discipline rather than as a numerical-integration technique.

Distinctive features.

- *Pre-commitment*: the scenario program names what the agent *will do* under each branch, before any branch is taken, so that the action is reviewable separately from its execution.
- *Decoupling planning from commitment*: E4 produces candidate actions; E5 accepts them only if the scenario program type-checks against a policy schema. This prevents an impulsive plan from becoming an irreversible action, in the spirit of the assurance discipline of self-adaptive systems [7, 8].
- *Externalised semantics*: the scenario program is the unit at which third parties (human operators, downstream services, audit pipelines) interact with the agent; this is the architectural analogue of the prediction-serving system’s latency-budgeted API boundary [10].

3.6 Composition and the novelty claim

The five engines compose by data flow: E3 and E1 together feed E4, which queries E2 for high-stakes inferences and emits candidate actions to E5. The cost-aware property of §4 emerges from E1+E2+E4 jointly carrying a per-decision cost; the self-adaptive property of §5 emerges from E4 closing a MAPE-K loop over E1, E3, and E5 on a versioned calibration manifest.

Novelty claim. *To our knowledge, no peer-reviewed architecture combines (i) a cost-aware confidence-cascading multi-tier prediction router [9–11]; (ii) an independent, economically-incentivised verification stage as a first-class architectural component; (iii) a multi-modal sentiment- fusion engine that includes on-chain flow as a fusion modality [18, 20]; (iv) a reason-and-act agent under explicit token, role, and policy constraints [3–5]; and (v) a scenario-engine externalisation boundary [21], into one reference architecture for risk-intelligence agents in partially-trusted environments.*

Falsification. The novelty claim is falsified by exhibiting a peer-reviewed architecture that contains all five engines (or named equivalents) and demonstrates the same emergent cost-aware and self-adaptive properties. The closest peer-reviewed neighbours we are aware of cover proper subsets: prediction-serving systems and cascades cover (i) and parts of (ii) but not (iv)–(v) [9–11]; LLM-agent reference architectures cover (iv) and parts of (v) but treat (i) and (ii) as out-of-scope [3–5]; the multimodal-sentiment-fusion literature covers (iii) but does not discuss agentic action [18–20]; the self-adaptive systems roadmap covers the *shape* of the composition but not the named engines [6–8].

Why the composition matters. Each engine in isolation is a known engineering primitive. We argue that the composition is what enables the cost-aware (§4) and self-adaptive (§5) properties to be *emergent and reviewable*: the named-component decomposition is what lets each property be traced to a specific failure mode in a specific engine, and what lets each engine be tested, replaced, or ablated independently.

4 (C2) The cost-aware AI engine

The first emergent property of the architecture (§3) treats cost as a first-class quantity composed of three terms: *inference cost* (dollars and tokens spent on model calls), *economic-trust cost* (the price of an independent judge on a high-stakes decision), and *action cost* (the on-chain or operational price of executing). The composition is realised by three gates whose mapping to the architecture is exact.

Gate A: confidence-cascading inference (E1). Following the attentional-cascade pattern [9] and its modern instantiations for deep models [11] and online prediction serving [10], the prediction router of §3.1 answers each query first with the cheapest forecaster; the answer is accepted only if its confidence-interval width and its agreement with the calibration manifest are both within the per-symbol threshold, otherwise it is escalated.

Gate B: economic-trust cost via independent verification (E2). For high-stakes inferences — those that the scenario engine of §3.5 would permit to drive an external action — the router’s accepted output is repriced through the verification stage of §3.2, in which independent validators score the inference and the system fails closed when they disagree. The engineering insight is to read this as a *cost* term: the system pays for an independent judge so that “cheap and wrong” is priced as expensive. This is a deliberate generalisation of the cost-aware-serving

pattern, in which the cost surface includes only inference dollars [10]: the cost-aware engine here also pays for independence of opinion.

Gate C: action-cost gating in the agent (E4). The reason-and-act agent [3–5] of §3.4 sits under an explicit token budget at every step, and emits each candidate as a pre-committed scenario program through E5 in the sense of scenario-based Monte Carlo [21]. From a cost-aware standpoint, the discipline is that action-cost is paid only when (Gate A) the cascade has accepted a forecast at the appropriate cost tier and (Gate B) the verifier has not disagreed with that forecast.

4.1 The cost ledger

The three gates together produce, per decision d , an explicit cost ledger

$$\text{cost}(d) = c_{\text{inf}}(d) + c_{\text{trust}}(d) + c_{\text{act}}(d),$$

each term observable, each term tied to a versioned engineering artefact. We do not claim a new algorithm; the engineering claim is that managing each term as an independently-versioned and observable quantity is the discipline that makes the overall system cost-auditable, and that this discipline emerges naturally from the five-engine architecture rather than being bolted on.

5 (C3) Self-adaptive agentic AI

The second emergent property of the architecture is a self-adaptive loop [6–8] in which E4 closes a MAPE-K loop over E1, E3, and E5 on a versioned calibration manifest, so that the agent *observes its own miss-rate and reconfigures itself* rather than waiting for an offline retraining cycle.

5.1 A MAPE-K instantiation

Monitor. The read-side of E1 and the multimodal sentiment- fusion engine E3 [15–20] together produce a stream of (forecast, realised-outcome, verifier-score) triples. *Analyse.* E2 (§3.2) and the calibration manifest together compute a running residual: how badly is the cascade missing, on which symbols, with what class imbalance. *Plan.* E4 [3–5] chooses between three remedies — re-weight the cascade, narrow E5’s permitted action set, or escalate to a human operator — using the profile/memory/planning/action decomposition prescribed in the LLM-agent survey [3]. *Execute.* E5 emits the adaptation as a versioned change to the cascade thresholds, the ensemble weights, or the action policy, each of which is a diffable artefact under change-control. *Knowledge.* The shared, versioned state is the calibration manifest, the contract-and-route state, and the action policy. Knowledge is *checked in*, not latent in a model’s weights.

5.2 An honesty discipline for adaptation

A self-adaptive AI in an adversarial environment is only useful if its self-reports are trustworthy. We adopt three engineering rules, non-negotiable in this codebase, which we view as the most transferable contribution of this paper.

Rule 1: publish denominators. Every accuracy or skill metric the adaptation loop reads is published with the class-imbalance denominator that produced it. A high accuracy on a heavily skewed distribution is far less impressive than the headline number suggests, and the engineering response is to say so on the same page as the number, in line with the long-standing call from the SE-for-ML literature [1, 2] to make evaluation a first-class engineering artefact.

Rule 2: version every artefact. Calibration thresholds, ensemble weights, action policies, profile prompts, and planning prompts are committed under change-control with a release-event audit trail. Without this rule, MAPE-K *Execute* becomes a back door that lets the system silently optimise itself away from the behaviour its operators reviewed, which the self-adaptive systems literature explicitly warns against under the heading of assurance under runtime change [7, 8].

Rule 3: refuse silent retraining. A change to a model, a prompt, or a constraint is a release event with a corresponding diff in the calibration manifest. Continuous learning that bypasses this discipline is permitted only with an explicit assurance argument, in the sense the self-adaptive literature prescribes [8].

6 Operational evidence

We report two operational case studies that exercise C2 and C3 on real production runs. Following the editorial directive that every academic reference resolve to a peer-reviewed publisher record, we describe what the system did in prose and decline to cite internal engineering logs; the case studies are operational reports rather than controlled experiments, and we state their limits accordingly.

6.1 27-hour Solana liquidity stress experiment

In the first case study, the agent operated for 27 hours against a single Solana liquidity pool with E5 gating every external action. As evidence for C2 (cost-aware engine), the trace shows that every action the agent proposed was emitted as a reviewable scenario program before execution; E2 acted as an independent gate that prevented two proposed actions from executing because it disagreed with the router’s accepted forecast; and no inference was permitted to escalate past Gate A’s per-symbol confidence threshold without being repriced through Gate B. As evidence for C3 (self-adaptive loop), the agent observed two regime shifts in the 27-hour window and re-weighted E1’s cascade in response to each, with the re-weighting committed as a diff to the cascade thresholds rather than as a silent ensemble retrain.

6.2 168-hour prediction-router calibration arc

In the second case study, we evaluated E1 over a 168-hour window. Reported honestly, the headline live-API accuracy on the production API was approximately 53% — “barely above chance”. After a horizon-aware crash-probability calibration fix, the freshest 168-hour cohort (915 samples) reports a high `didCrashAccuracy` and a Brier score of approximately 0.13. The arc must be read in the context of the class-imbalance and cohort-size caveats: the high accuracy reflects a heavily skewed prior on the no-crash class, and the engineering value of reporting it is precisely to expose the gap between the headline metric and the calibration discipline that produced it. As evidence for C3, the arc itself is the adaptation: a versioned calibration fix, deployed on a specific date, with a diff in the change-control history and an honest accompanying narrative.

7 The cost-and-honesty thesis

The contributions of §§3–5 are not, in isolation, novel: cascade-and-serving primitives are well established [9–11], the LLM-agent decomposition is the consensus view [3–5], multimodal sentiment fusion is a mature field [15–20], and MAPE-K is two decades old [6–8]. What this paper claims is that, when these primitives are composed into the five-engine architecture of

§3, the discipline that keeps the composition reviewable reduces to four falsifiable engineering propositions. We state them, name the failure each prevents, and bound their transferability.

Proposition 1 (cost as composition). *Inference cost, trust cost, and action cost are different currencies and should be recorded as separate observable terms in a per-decision ledger. What this prevents:* a system that is cheap on average but expensive on the decisions that matter, because the cost of being wrong was never priced separately from the cost of being slow. *Falsifier:* an agent that meets all SLOs on inference cost but incurs disproportionate downstream-action losses on a small minority of high-stakes decisions, as measured by an independent post-hoc reconciliation [21]. *Transferability:* any agentic system whose actions have heterogeneous downstream costs.

Proposition 2 (independent verification as a budget item). *The verifier of a high-stakes inference must be independently incentivised, and the cost of paying it must appear in the cost ledger alongside inference and action. What this prevents:* co-located optimiser-verifier pairs that conspire on the same training distribution, so that the verifier silently endorses the optimiser’s drift. *Falsifier:* a verifier whose disagreement rate with the optimiser is statistically indistinguishable from zero on hard cases, because the two share latent failure modes; the prediction-serving literature has long warned that ensembling without independence buys nothing [10]. *Transferability:* any agentic system in which a single model’s mistake can be socially or operationally costly.

Proposition 3 (adaptation on a versioned manifest). *Every adaptive change — to a cascade threshold, an ensemble weight, a planning prompt, or an action policy — must be a release event with a diff in a versioned manifest. What this prevents:* “the model improved overnight” as a defect masquerading as a feature; silent drift away from the behaviour the system’s operators reviewed; and the assurance gap explicitly named in the self-adaptive systems literature [7, 8]. *Falsifier:* two operators reviewing the system one week apart and arriving at materially different mental models of what the system will do, without an intervening change-control diff to explain the difference. *Transferability:* any system in which an LLM, an ensemble, or a policy is permitted to change between releases.

Proposition 4 (calibration with denominators). *Every self-reported accuracy must publish the class-imbalance denominator that produced it, on the same page. What this prevents:* headline metrics that are statistically trivial but reputationally seductive — a 99% accuracy on a heavily-skewed prior is exactly the failure mode that motivates the SE-for-ML community’s repeated insistence that evaluation be a first-class engineering artefact [1, 2]. *Falsifier:* a metric reported in isolation; or a metric whose denominator is reported elsewhere than on the page that reports the metric. *Transferability:* every self-reporting AI system, without qualification.

The combined claim is intentionally narrow. It does not say what the right cost function is, what the right ensemble weighting is, or what the right action policy is. It says that an agentic AI system that violates any of the four propositions is not engineering; it is storytelling, and the SE-for-ML and self-adaptive software literatures have, between them, accumulated enough evidence of the resulting failure modes to justify treating each proposition as a non-negotiable operational rule.

8 Threats to validity and limitations

The case studies of §6 are operational rather than controlled experiments: they are reports of what the system did under real conditions and do not estimate counterfactuals or generalise

beyond the chains, pools, and time-windows they describe. Each engine in the architecture (§3) operationalises a peer-reviewed primitive — attentional cascades and prediction serving for E1 [9–11]; the multi-modal sentiment-fusion lineage for E3 [15–20]; the reason-and-act and generative-agents architectures for E4 [3–5]; scenario- based decision analysis for E5 [21] — and we do not claim those primitives as new contributions, only the named composition (C1) and the emergent cost-aware (C2) and self-adaptive (C3) properties that the composition produces. The novelty claim of §3.6 is stated falsifiably; an exhibit of a prior peer-reviewed architecture that contains all five engines and demonstrates the same emergent properties would refute it. The cost-and-honesty thesis of §7 is a normative claim about practice; the four propositions are stated falsifiably, but their transferability beyond the agentic-AI setting is a hypothesis, not a result.

9 Conclusion

We have described the engineering — not the algorithms — of an agentic, context-aware AI system that operates in a genuinely adversarial environment. The contribution is threefold. *First*, a five-engine reference architecture (§3) composed of a confidence-cascading prediction router, an independent verification stage, a multi-modal sentiment-fusion engine that includes on-chain flow, a reason-and-act agent under explicit constraints, and a scenario engine that externalises every action as a pre-committed program. *Second*, a cost-aware AI engine (§4) that emerges from this composition and yields a per-decision cost ledger combining inference, trust, and action costs. *Third*, a self-adaptive agentic AI (§5) that emerges from the agent closing a MAPE-K loop over the architecture on a versioned calibration manifest. The synthesising *cost-and-honesty thesis* (§7) distils the discipline of the composition into four falsifiable engineering propositions. Any agentic AI system whose inputs and actions share a partially-trusted environment will benefit from making its verifier independent, its planning artefacts diffable, its denominators public, and its actions pre-committed.

References

- [1] D. Sculley et al. “Hidden Technical Debt in Machine Learning Systems”. In: *Advances in Neural Information Processing Systems 28 (NeurIPS 2015)*. Catalogues the system-level costs that production ML systems pay (entanglement, hidden feedback loops, undeclared consumers, configuration debt, monitoring debt) over and above traditional code-level technical debt; widely cited as the foundational artefact for “software engineering for ML”. 2015, pp. 2503–2511. URL: https://papers.nips.cc/paper_files/paper/2015/hash/86df7dcfd896fcdf2674f757a2463eba-Abstract.html (visited on 05/09/2026).
- [2] Saleema Amershi et al. “Software Engineering for Machine Learning: A Case Study”. In: *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. Empirical study at Microsoft of the workflow and process changes required to engineer ML-intensive software at scale; identifies a nine-stage ML workflow and the discontinuities (data management, model evaluation, model deployment) where SE practice diverges from non-ML systems. IEEE, 2019, pp. 291–300. DOI: [10.1109/ICSE-SEIP.2019.00042](https://doi.org/10.1109/ICSE-SEIP.2019.00042).
- [3] Lei Wang et al. “A Survey on Large Language Model Based Autonomous Agents”. In: *Frontiers of Computer Science* 18.6 (2024). Unified construction-evaluation-application framework for LLM-based autonomous agents; this manuscript adopts its profile/memory/planning/action decomposition as the engineering vocabulary against which the agentic-engine design is described., p. 186345. DOI: [10.1007/s11704-024-40231-1](https://doi.org/10.1007/s11704-024-40231-1).

- [4] Shunyu Yao et al. “ReAct: Synergizing Reasoning and Acting in Language Models”. In: *The Eleventh International Conference on Learning Representations (ICLR 2023)*. Peer-reviewed (ICLR 2023) introduction of the reason-and-act loop that interleaves chain-of-thought and tool calls in an LLM agent; cited here as the canonical academic reference for the LLM-mediated planning step in the agentic engine. 2023. URL: https://openreview.net/forum?id=WE_vluYUL-X (visited on 05/10/2026).
- [5] Joon Sung Park et al. “Generative Agents: Interactive Simulacra of Human Behavior”. In: *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST 2023)*. Peer-reviewed (UIST 2023) reference architecture for LLM-based agents organised around an explicit memory store, periodic reflection, and retrieval-driven planning; cited here for the profile/memory/planning/action decomposition the agentic engine adopts. ACM, 2023. DOI: [10.1145/3586183.3606763](https://doi.org/10.1145/3586183.3606763).
- [6] Mazeiar Salehie and Ladan Tahvildari. “Self-adaptive software: Landscape and research challenges”. In: *ACM Transactions on Autonomous and Adaptive Systems* 4.2 (2009). Canonical survey of self-adaptive software architectures; the MAPE-K (Monitor-Analyse-Plan-Execute over Knowledge) feedback loop it formalises is the lineage that the OmniRisk five-engine architecture extends to LLM-mediated agentic operation., 14:1–14:42. DOI: [10.1145/1516533.1516538](https://doi.org/10.1145/1516533.1516538).
- [7] Betty H.C. Cheng et al. “Software Engineering for Self-Adaptive Systems: A Research Roadmap”. In: *Software Engineering for Self-Adaptive Systems*. Vol. 5525. Lecture Notes in Computer Science. Peer-reviewed Dagstuhl-seminar roadmap that codifies the engineering concerns of self-adaptive systems (modelling dimensions, requirements, engineering, assurance) and complements the MAPE-K formalism of Salehie and Tahvildari; this manuscript’s adaptation loop sits squarely in this lineage. Springer, 2009, pp. 1–26. DOI: [10.1007/978-3-642-02161-9_1](https://doi.org/10.1007/978-3-642-02161-9_1).
- [8] Danny Weyns. *An Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective*. Textbook treatment of self-adaptive software, generalising MAPE-K with explicit treatment of uncertainty management, feedback control, and online learning; this manuscript adopts its perspective for the agentic-engine adaptation loop. Wiley-IEEE Press, 2020. ISBN: 978-1-119-57494-1. DOI: [10.1002/9781119574910](https://doi.org/10.1002/9781119574910). URL: <https://onlinelibrary.wiley.com/doi/book/10.1002/9781119574910> (visited on 05/09/2026).
- [9] Paul Viola and Michael J. Jones. “Robust Real-Time Face Detection”. In: *International Journal of Computer Vision* 57.2 (2004). Foundational peer-reviewed reference for the attentional-cascade pattern in machine learning: cheap classifiers reject easy negatives early so that expensive classifiers see only hard cases; the cost-aware AI engine of this manuscript instantiates the same idea for LLM-mediated risk inference., pp. 137–154. DOI: [10.1023/B:VISI.0000013087.49260.FB](https://doi.org/10.1023/B:VISI.0000013087.49260.FB).
- [10] Daniel Crankshaw et al. “Clipper: A Low-Latency Online Prediction Serving System”. In: *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. Peer-reviewed canonical reference for online prediction-serving systems with model selection and ensembling under latency budgets; provides the framing for cost-aware inference adopted in this manuscript’s prediction-router gate. Boston, MA, USA: USENIX Association, 2017, pp. 613–627. URL: <https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/crankshaw> (visited on 05/10/2026).
- [11] Daniel Kang et al. “NoScope: Optimizing Deep CNN-Based Queries over Video Streams at Scale”. In: *Proceedings of the VLDB Endowment* 10.11 (2017). Peer-reviewed exemplar of the inference-cascade pattern for cost-aware querying of deep models at scale;

specialised difference detectors and confidence-thresholded escalation generalise to the prediction-router cascade described in this manuscript., pp. 1586–1597. DOI: [10.14778/3137628.3137664](https://doi.org/10.14778/3137628.3137664).

- [12] André Augusto et al. “SoK: Security and Privacy of Blockchain Interoperability”. In: *2024 IEEE Symposium on Security and Privacy (SP)*. Systematic literature review of 212 documents (58 peer-reviewed, 154 grey) and a security/privacy taxonomy over 57 interoperability solutions; reports cumulative cross-chain hack losses of approximately USD 3.1B. San Francisco, CA, USA: IEEE, 2024, pp. 3840–3865. DOI: [10.1109/SP54263.2024.00255](https://doi.org/10.1109/SP54263.2024.00255).
- [13] Rafael Belchior et al. “A Survey on Blockchain Interoperability: Past, Present, and Future Trends”. In: *ACM Computing Surveys* 54.8 (2022). Canonical pre-S&P-2024 systematisation of blockchain interoperability mechanisms (notary schemes, sidechains/relays, hashed time-locked contracts); cited alongside [12] as the broader interoperability-mechanism context for the OFT-on-LayerZero-V2 substrate., 168:1–168:41. DOI: [10.1145/3471140](https://doi.org/10.1145/3471140).
- [14] Jiajing Wu et al. “Safeguarding Blockchain Ecosystem: Understanding and Detecting Attack Transactions on Cross-chain Bridges”. In: *Proceedings of the ACM Web Conference 2025 (WWW ’25)*. Introduces BridgeGuard, a graph-motif detector trained on 203 attack transactions across 49 cross-chain bridge incidents (June 2021 – September 2024). ACM, 2025. DOI: [10.48550/arXiv.2410.14493](https://doi.org/10.48550/arXiv.2410.14493). URL: <https://arxiv.org/abs/2410.14493> (visited on 05/03/2026).
- [15] Robert P. Schumaker and Hsinchun Chen. “Textual Analysis of Stock Market Prediction Using Breaking Financial News: The AZFin Text System”. In: *ACM Transactions on Information Systems* 27.2 (2009). Foundational study showing breaking financial news improves short-horizon stock prediction; one of the earliest reproducible benchmarks for fusion of news-text features with price features., 12:1–12:19. DOI: [10.1145/1462198.1462204](https://doi.org/10.1145/1462198.1462204).
- [16] Frank Z. Xing, Erik Cambria, and Roy E. Welsch. “Natural Language Based Financial Forecasting: A Survey”. In: *Artificial Intelligence Review* 50.1 (2018). Survey covering text representation, fusion strategy (early vs late), and modelling for financial forecasting. We use it for the early-fusion vs late-fusion vs stacking taxonomy referenced in the sentiment-fusion engine subsection., pp. 49–73. DOI: [10.1007/s10462-017-9588-9](https://doi.org/10.1007/s10462-017-9588-9).
- [17] Min-Yuh Day and Chia-Chou Lee. “Deep Learning for Financial Sentiment Analysis on Finance News Providers”. In: *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. Early demonstration that deep models on multi-source finance news outperform single-source baselines; cited for the multi-source sentiment-fusion claim. IEEE, 2016, pp. 1127–1134. DOI: [10.1109/ASONAM.2016.7752381](https://doi.org/10.1109/ASONAM.2016.7752381).
- [18] Ankita Gandhi et al. “Multimodal Sentiment Analysis: A Systematic Review of History, Datasets, Multimodal Fusion Methods, Applications, Challenges and Future Directions”. In: *Information Fusion* 91 (2023). Systematic review reaching the same early/late/hybrid partition as Lai et al. and providing the comparison we cite for when each strategy wins: early fusion captures cross-modal correlation but is brittle to missing modalities; late fusion is modality-robust but loses fine-grained interaction; stacking with a learned meta-model dominates when per-modality signals are heterogeneous in noise floor and timescale., pp. 424–444. DOI: [10.1016/j.inffus.2022.09.025](https://doi.org/10.1016/j.inffus.2022.09.025).
- [19] Wen Long, Zhichen Lu, and Lingxiao Cui. “Deep Learning-Based Feature Engineering for Stock Price Movement Prediction”. In: *Knowledge-Based Systems* 164 (2019). Multi-filters neural network (MFNN) demonstrating deep-learning feature engineering layered over Chinese-market price and news data; cited as the centralised-feature-engineered ensemble

against which the TAO-incentivised distributed prediction subnet must be argued on diversity rather than fixed-dataset accuracy., pp. 163–173. DOI: [10.1016/j.knosys.2018.10.034](https://doi.org/10.1016/j.knosys.2018.10.034).

- [20] Zhibo Zhang, Chen Jiang, and Mengjie Lu. “Fusion of Sentiment and Market Signals for Bitcoin Forecasting: A SentiStack Network Based on a Stacking LSTM Architecture”. In: *Big Data and Cognitive Computing* 9.6 (2025). Worked example of contemporary on-chain ↔ off-chain sentiment fusion for Bitcoin forecasting: financial-news and social-media sentiment encoded by DeepSeek, fused both early (concatenated features) and late (per-modality LSTM stacked through a Ridge meta-learner) with macro and market indicators; argues sentiment carries information that pure market data cannot reproduce, particularly across regime breaks., p. 161. DOI: [10.3390/bdcc9060161](https://doi.org/10.3390/bdcc9060161).
- [21] Paul Glasserman. *Monte Carlo Methods in Financial Engineering*. Vol. 53. Stochastic Modelling and Applied Probability. Standard reference for scenario generation and pre-commitment policy evaluation in algorithmic trading and derivatives risk; provides the foundation for the scenario-engine pre-commitment semantics. New York: Springer, 2003. ISBN: 978-0-387-00451-8. DOI: [10.1007/978-0-387-21617-1](https://doi.org/10.1007/978-0-387-21617-1).