The Cognitive Decision Loop: A Governed Architecture for Industrial AI Agents Separating Model Reasoning from Agent Control in Governance-Constrained Industrial Operations

Pieter van Schalkwyk[1]
CEO, XMPro
ORCID: 0009-0009-3982-9346
pieter.vanschalkwyk@xmpro.com

Gavin Green[1]
Chief Solutions Officer, XMPro
ORCID: 0009-0007-8315-5559
gavin.green@xmpro.com

[1]XMPro Inc., Dallas, Texas, USA

May 2026

Abstract

Recent reasoning-focused large language models improve performance by allocating additional inference-time computation to multi-step problem solving. While these models are useful for verifiable reasoning tasks, industrial agents require a different architectural focus: governed decision-making under operational constraints. This paper introduces the *Cognitive Decision Loop*, an Observe–Reflect–Plan–Act architecture for industrial AI agents that separates model-internal reasoning from agent control. The loop combines fact-based operational observations, multi-dimensional scored gating (importance, surprise, confidence, source trust), significance-gated reflection, confidence-calibrated conditional planning, multi-agent coordination with consensus over scored signals, provenance-oriented Decision Traces, and bounded actuation. We situate the architecture within three research lineages: operator decision theory, generative-agent architectures, and industrial AI governance. In particular, we argue that Park et al.'s memory–reflection–planning pattern for generative agents [18] can be extended for industrial contexts where believability is less important than traceability, configurability, and bounded action. Using XMPro MAGS as a reference implementation, we argue that large language models should be treated as configurable cognitive services inside a governed decision architecture rather than as the architecture itself. The paper concludes with implications for evaluating industrial agent systems, including control architecture, actuation boundaries, decision-artifact quality, domain-expert configurability, and auditability.

Keywords: industrial AI agents; agent architecture; decision loop; ORPA; OODA; generative agents; large language models; reasoning models; scored gating; predictive processing; governance; provenance; Decision Trace; bounded actuation; multi-agent systems; XMPro MAGS.

1 Introduction

"Reasoning" has become one of the most important and most ambiguous terms in contemporary artificial intelligence. In the language-model literature, the term now often refers to models that allocate additional computation at inference time to improve multi-step problem solving.

These models may generate hidden or visible chains of thought, search through alternatives, self-correct intermediate steps, or use reinforcement learning to improve performance on tasks such as mathematical reasoning, software engineering, and scientific problem solving. This represents a significant advance. Reasoning-focused language models have changed expectations about what large language models can do when the task has a verifiable answer and benefits from additional deliberation.

Industrial agent systems create a different problem. A refinery agent, a mining agent, a production-optimization agent, or a maintenance agent is not merely answering a difficult question. It is participating in a governed decision process. It must decide whether the current operating state warrants attention, whether new facts should trigger reflection, whether reflection should produce a plan, whether a plan creates constraint conflicts, whether other agents or human operators should participate, and whether any intended action is authorized through a configured actuation pathway. These are system-level properties. They cannot be reduced to the depth of a model's internal token-stream deliberation.

The architectural question is whether the agent system has a governed control architecture for deciding what should happen next, separate from whether its model can reason well in isolation.

This paper proposes the Cognitive Decision Loop as one such architecture. The Cognitive Decision Loop is an Observe–Reflect–Plan–Act cycle, or ORPA cycle, designed for industrial agents operating under objectives, constraints, provenance requirements, and bounded action channels. The loop treats language models as cognitive services that assist with interpretation, synthesis, communication, and planning. The architecture is the governed loop in which such reasoning may participate, rather than the model-internal reasoning itself.

The contribution is architectural rather than algorithmic. It draws from three lineages. The first is human decision theory, including OODA, recognition-primed decision-making, dual-process theory, mental models, and predictive-processing accounts of selective belief update. These frameworks converge on the idea that expert decision-making is cyclic, situated, threshold-gated, and action-oriented. The second is the generative-agent computational pattern introduced by Park et al. [18], where believable agent behavior emerges from memory streams, reflection, retrieval, and planning around a large language model. The third is industrial governance: constraints, auditability, provenance, bounded actuation, consensus, and human review.

The novel claim is that memory, reflection, and planning mechanisms need to be re-architected for industrial operations, not introduced as new constructs. In such environments, believable behavior is not the main goal. The main goals are traceable interpretation, explicit objectives, governed planning, multi-agent coordination, and bounded action. A generative social simulation can prioritize believability. An industrial agent must prioritize operational fit, auditability, and configured authority.

The unit of industrial agent design is the governed decision loop. Prompts and model choices sit inside that loop as configurable artifacts.

The remainder of the paper is organized as follows. Section 2 reviews related work in reasoning large language models, generative agents, agent architectures, human decision theory, and governance. Section 3 defines the category error involved in treating reasoning models and industrial decision architectures as the same object. Section 4 introduces the Cognitive Decision Loop, including the scored-gating subsection that controls phase transitions. Section 5 discusses governance and Decision Trace. Section 6 presents XMPro MAGS as a reference implementation, including the use of decision artifacts as units of evaluation. Section 7 compares the pattern with LLM reasoning agents. Sections 8 through 11 discuss implications, limitations, future work, and conclusions.

2 Background and Related Work

2.1 Reasoning Large Language Models

Reasoning-focused large language models extend the capabilities of standard generative language models by allocating additional inference-time computation to tasks requiring multi-step reasoning. The practical mechanisms vary. Some systems use chain-of-thought prompting to elicit intermediate reasoning steps [25]. Others structure search through alternatives, as in Tree of Thoughts [30], or interleave reasoning with action, as in ReAct [29]. More recent reasoning models, including OpenAI's o1 series [17] and DeepSeek-R1 [5], use reinforcement learning and inference-time deliberation to improve performance on complex reasoning tasks.

The shift from immediate answer generation to deliberate inference is significant. It makes language models more useful for mathematics, coding, formal analysis, and other domains where intermediate reasoning can improve final-answer quality. Survey work has framed this shift as movement from more reactive "System 1" behavior toward more deliberate "System 2" behavior in large language models [14].

Reasoning depth, however, is not equivalent to governed agency. A model may deliberate more deeply over a prompt without acquiring a system-level understanding of operational authority, audit requirements, actuation boundaries, or multi-party consensus. Reasoning models improve inference inside the model. Industrial agent systems require explicit architecture around the model.

This distinction becomes especially important when model-internal reasoning is hidden, summarized, or only partially inspectable. For industrial operations, the trace of a model's deliberation is not sufficient. Reviewers need to know which data were admitted as observations, which objectives were active, which constraints were evaluated, which alternatives were rejected, whether a human or other agent was consulted, and how the proposed action was bounded.

2.2 Generative-Agent Computational Pattern

The closest computational lineage for the Cognitive Decision Loop is the generative-agent architecture introduced by Park et al. [18]. In *Generative Agents: Interactive Simulacra of Human Behavior*, Park and colleagues demonstrated that believable agent behavior can emerge when a large language model is embedded within a broader architecture for memory, reflection, retrieval, and planning. Agents maintain a memory stream, assign importance to observations, retrieve relevant memories using recency, importance, and relevance, trigger reflections when accumulated significance crosses a threshold, and use reflections and memories to plan future behavior.

The paper's importance for industrial agent architecture lies in this shift from prompt to system. Park et al. showed that believable agent behavior is not simply a property of a better prompt. It emerges from an architecture around the language model. The model remains important, but the agent's continuity, coherence, and situated behavior depend on memory, retrieval, reflection, and planning.

The Cognitive Decision Loop extends this computational pattern into a different application domain. Park et al. simulated human social behavior in a sandbox environment. The Cognitive Decision Loop adapts the computational pattern, not the application. The industrial setting changes the design requirements. Believability is no longer the primary objective. Instead, the system must handle fact-based operational state, objective functions, utility trade-offs, deontic

constraints, consensus, provenance, human review, and bounded action.

The Cognitive Decision Loop industrializes the memory–reflection–planning pattern by placing it inside a governed operational decision cycle, rather than transposing a social-simulation architecture directly into factories, mines, or utilities. In this setting, reflection is a significance-gated sense-making step over operational facts and domain context, not a device for producing believable social continuity. Planning is a conditional expression of operational intent, not a device for producing plausible daily behavior. Action is bounded by configured channels and review pathways rather than open-ended in a simulated world.

2.3 Agent Architectures and Frameworks

The Cognitive Decision Loop also relates to longer-running work on agent architectures. Belief–Desire–Intention (BDI) agents, for example, represent agents in terms of beliefs about the world, desires or objectives, and intentions that guide action [20, 21]. The BDI tradition is relevant because it distinguishes between world state, goals, and committed courses of action. It also emphasizes that agent behavior depends on architecture, not only on local inference.

Modern LLM-agent frameworks have taken a different path. Systems such as AutoGen, ChatDev, and Voyager and related frameworks often emphasize tool use, role-based multi-agent conversation, workflow orchestration, and task decomposition [19, 24, 26]. Tool-use research such as Toolformer and ReAct provides important mechanisms for combining language models with external actions or environments [22, 29]. The Model Context Protocol [2] and similar ecosystem efforts further standardize access to external tools and context.

These frameworks are useful, but they are usually not designed first around industrial governance. Many are optimized for knowledge work, software development, research assistance, content generation, or agentic task execution. They often treat agents as model-driven tool users or as participants in conversational workflows. The Cognitive Decision Loop emphasizes a different unit of design: a governed operational decision cycle with explicit gates, traceability, multi-agent coordination, and bounded actuation.

This does not make the Cognitive Decision Loop a replacement for LLM-agent frameworks. It makes it a pattern for a different problem class. In knowledge work, the quality of the generated answer or completed task may be the primary concern. In industrial operations, the primary concern is whether an agent's interpretation, planning, and action intent remain within governed operational boundaries.

A related recent framing is the notion of *code as agent harness* [16]. Ning et al. argue that code is increasingly used not only as an output of language models, but as an executable, inspectable, and stateful substrate through which agents reason, act, model environments, receive feedback, and coordinate. Their survey distinguishes model-internal capabilities from harness infrastructure: tools, APIs, sandboxes, memory, validators, permission boundaries, telemetry, and workflows. This distinction is important for the Cognitive Decision Loop because it reinforces the central architectural claim of this paper: reliable agency is not a property of the language model alone, but of the system that surrounds the model and governs how model outputs become stateful action.

The Cognitive Decision Loop can be understood as a parallel harness pattern for industrial operations. In code-centric agent systems, executable programs, tests, repositories, traces, and runtime feedback provide the substrate for verification and adaptation. In industrial agent systems, the corresponding substrate is broader: fact-based observations, operational measures, objective and utility functions, significance and confidence scores, constraints, consensus records, Decision Traces, approval pathways, and bounded action channels. Code may still appear as

tools, actions, simulations, or integration logic, but it is not the primary unit of governance. The primary unit is the governed operational decision artifact.

This difference matters because industrial environments are not merely programmable environments. They are physical, safety-bounded, multi-objective, and authority-constrained. A software agent can often verify progress by running tests or inspecting execution traces. An industrial agent must also evaluate whether an observation is trustworthy, whether a disturbance is significant, whether a proposed plan affects safety, quality, production, maintenance, or economic objectives, whether consensus is required, and whether the action is authorized. The Cognitive Decision Loop therefore extends the harness perspective from executable code toward governed decision architecture.

2.4 Cognitive Science of Expert Decision-Making

The Cognitive Decision Loop is also informed by research on how expert human decision-makers operate under uncertainty. Dual-process theory distinguishes between fast pattern recognition and slower deliberative reasoning [6, 12]. Expert operators rely on both. Most observations do not require extended deliberation. Some observations, however, cross a threshold of surprise, importance, or risk and require slower sense-making.

Recognition-primed decision-making is especially relevant for industrial operations. Klein [13] showed that experienced decision-makers under time pressure often do not compare a full decision tree of options. They recognize a situation as similar to prior experience, mentally simulate a plausible course of action, and proceed if it appears workable. Industrial operators often act in this mode. They do not reason from first principles over every signal. They interpret patterns against experience, procedures, process knowledge, and operating objectives.

Mental-model theory provides another lens. Johnson-Laird [11] argues that human reasoning often involves the construction and manipulation of internal models of possible states. Operators do not merely classify signals. They maintain an internal model of how a process should behave and compare observations against that model. Predictive-processing accounts similarly emphasize the role of prediction error and selective belief update in cognition [4, 7]. This also provides a formal bridge to information-theoretic surprisal: surprise can be treated as prediction error conditioned by context, not merely as novelty in isolation [23].

OODA, developed by Boyd [3], gives a cyclic operational frame: Observe, Orient, Decide, Act. The central insight is not the acronym but the loop. Operational decision-making is continuous, situated, and action-oriented. The Orient phase is especially important because it transforms raw observation into meaning through synthesis against prior models, experience, doctrine, and context.

Across these frameworks, expert decision-making appears as a cycle, not a single-pass deliberation. It is threshold-gated, not continuously active. It is bounded by explicit governance such as training, certification, SOPs, and authorization boundaries. The load-bearing step is the meaning-making step: Orient in OODA, Reflect in ORPA.

2.5 Provenance, Governance, and Industrial Assurance

Industrial agent systems require governance and provenance. The W3C PROV-O ontology provides a vocabulary for representing provenance: entities, activities, agents, usage, attribution, generation, and derivation [1]. It is relevant because industrial agent decisions need not only outputs, but records of how outputs were produced.

AI governance frameworks such as the NIST AI Risk Management Framework and ISO/IEC 42001 emphasize trustworthiness, accountability, governance processes, risk management, and organizational controls [10, 15]. Industrial automation and control system standards such as IEC 62443 emphasize security, system boundaries, and controlled interaction with operational systems [8]. Functional-safety standards such as IEC 61511 may become relevant when agents interact with safety-related processes [9], although the Cognitive Decision Loop should not be treated as a safety-system certification claim.

For the purposes of this paper, these standards provide context rather than a compliance checklist. They reinforce a common architectural requirement: industrial AI agents should produce inspectable records and operate within bounded authority.

3 The Category Error: Reasoning Models vs. Decision Architectures

The category error in many agent discussions is the assumption that a better reasoning model automatically creates a better operational agent. This assumption confuses two levels of analysis.

LLM reasoning is model-internal. It occurs inside a token stream, often with hidden intermediate steps or summarized traces. It can improve answer quality on tasks where the model must solve a difficult problem before responding. Industrial decisioning is system-distributed. It occurs across operational observations, objectives, constraints, plans, teams, tools, traces, and actuation boundaries.

Reasoning models improve inference. Industrial agents require control architecture.

Consider an agent tasked with responding to rising compressor vibration while throughput demand remains high. A reasoning model might analyze possible causes, list maintenance risks, or suggest mitigation options. These capabilities are useful, but they do not by themselves answer the system-level questions. Is the vibration observation admissible and trustworthy? Is it significant relative to the asset profile? Does it cross a reflection threshold? Which prior events should be retrieved? Which objective functions are active? Does the maintenance objective conflict with the production objective? What constraints apply? Does the plan require consensus? Is there an authorized action channel? Should the intent be routed to a human?

These questions are not simply prompts. They are architectural responsibilities.

Figure 1 contrasts the two levels of analysis directly. A reasoning model improves what happens inside a token stream. An industrial decision architecture spans observations, scored gates, reflection, planning, governance, bounded action, and human review, all converging into a Decision Trace.

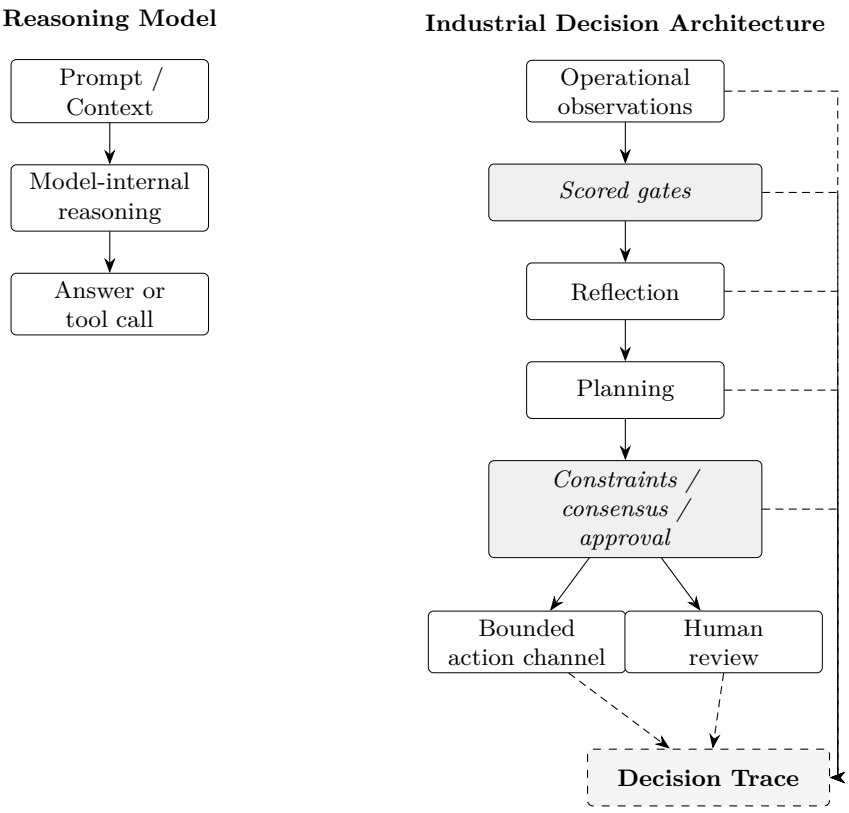

Figure 1: Reasoning model versus industrial decision architecture. The reasoning-model column improves answer quality inside a token stream. The decision-architecture column distributes control across observations, scored gates, reflection, planning, governance, bounded actuation, and human review, with every phase contributing to a Decision Trace.

Industrial agent systems should therefore be evaluated on properties that current reasoning benchmarks rarely measure. Where does the control architecture live? How are observations admitted and scored? How is reflection triggered? How are objectives and utilities represented? Where are constraints evaluated? How is action bounded? What provenance survives review? What happens when no action pathway is authorized?

LLM reasoning happens inside a token stream. Industrial decisioning happens across observations, objectives, constraints, plans, teams, tools, and actuation boundaries.

This argument does not reduce the value of reasoning models. Better models may improve observation interpretation, reflection synthesis, communication, and plan generation. Model improvement does not remove the need for a governed decision loop. The Cognitive Decision Loop provides such a pattern.

4 The Cognitive Decision Loop

The Cognitive Decision Loop is a governed cycle of Observation, Reflection, Planning, and Action. Internally, we describe it as ORPA: Observe, Reflect, Plan, Act. The loop is OODA-inspired but not a direct OODA implementation. The substitution of Reflect for Orient is deliberate. Reflect names the engineering requirement: meaning-making should be explicit, parameterized, and auditable as a decision event, rather than treated as an implicit phase boundary.

The Cognitive Decision Loop is the architecture. The Decision Trace is the audit artifact it

produces.

Figure 2 shows the full ORPA cycle with scored gates between phases. Each gate is configurable, traceable, and produces a decision artifact whether it fires or not.

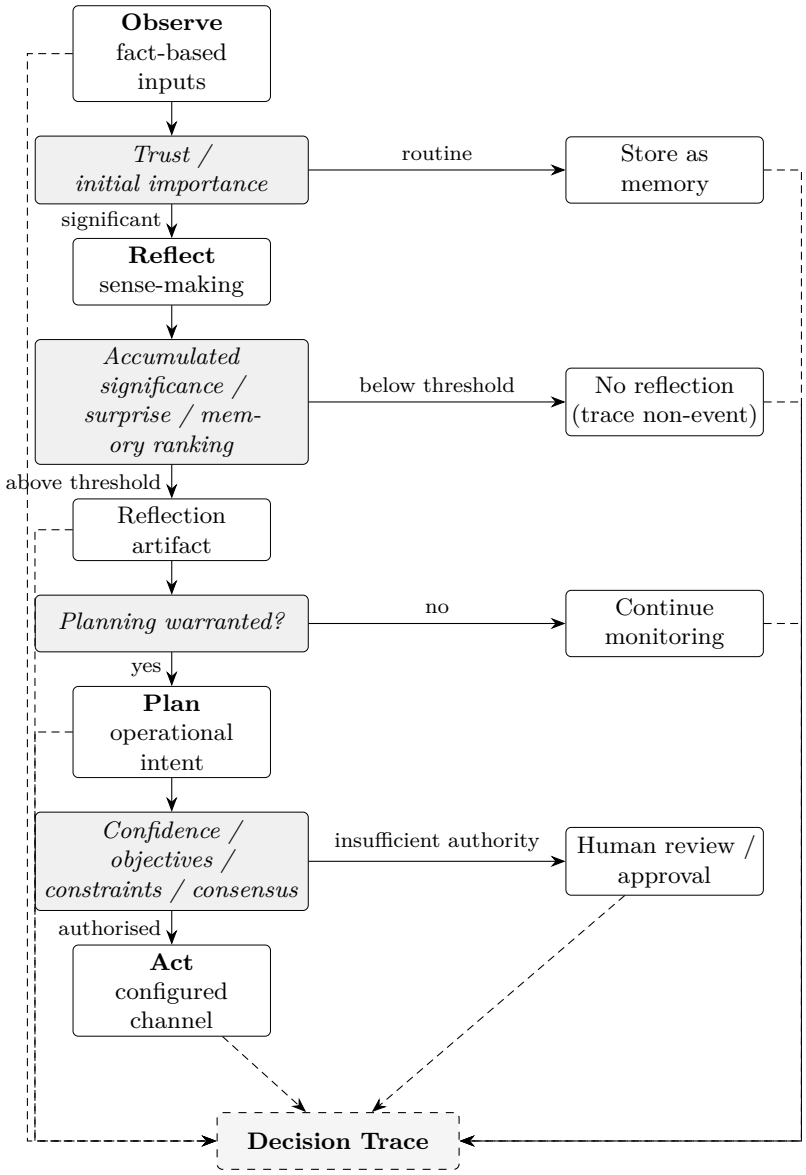

Figure 2: The Cognitive Decision Loop expressed as ORPA, with explicit scored gates between phases. Each gate (italicised) is configurable, and each decision outcome — including non-events such as "no reflection" or "continue monitoring" — contributes a decision artifact to the Decision Trace.

4.1 Observe

The Observe phase receives fact-based operational inputs. These may come from sensors, engineering models, statistical predictions, calculations, digital twins, tools, operational data pipelines, or other agents. The key principle is that the agent is not asked to invent process state. It interprets state that has been supplied through operationally meaningful channels.

This design reduces a common hallucination risk. A maintenance agent should not fabricate a failure probability from language alone. If a failure probability is needed, it should be

computed by an appropriate engineering, statistical, or machine-learning model and supplied as an observation. The agent can then interpret the significance of that fact in context.

Observe inherits part of the Park et al. pattern: observations enter a memory stream and can be scored for importance. The industrial extension is fact-based grounding. In industrial operations, observations should be grounded in operational data, calculations, predictions, and engineering context before they reach the agent.

4.2 Scored Gating: Importance, Surprise, Confidence, and Trust

The Cognitive Decision Loop uses scores as control signals rather than as retrospective annotations alone. In a governed industrial loop, importance, surprise, confidence, and source trust help determine whether an observation remains routine, contributes to accumulated significance, triggers reflection, enters planning, invokes consensus, or requires human review.

These scores have different roles across the loop. During Observe, source trust and initial importance help determine whether incoming facts are reliable and operationally relevant. During Reflect, accumulated significance, surprise, and retrieved-memory ranking determine whether the agent should perform higher-order synthesis. During Plan, confidence and objective-function alignment help determine whether operational intent is sufficiently supported. During Act, confidence, constraint status, consensus outcomes, and action authority determine whether intent can proceed through a configured channel or must be escalated.

Surprise should be understood as context-conditioned prediction error rather than simple novelty. A familiar event in an unusual operating context may be surprising, while a novel observation during an active disturbance may be less surprising than the same observation during stable operation. In this sense, surprise is an operational analogue of Shannon surprisal [23] conditioned on process context, historical activity, and the agent's current role. This connects the scoring layer to predictive-processing accounts of cognition: the agent updates only when deviation from expectation is meaningful enough to matter.

Score composition is itself part of the architecture. Implementations may use independent thresholds, weighted aggregates, or gated cascades in which one score determines whether another evaluation is performed. The important architectural point is that the composition of scores, thresholds, decay parameters, and escalation rules is explicit and configurable rather than hidden inside model behavior. Specific weights and formulas are implementation details under domain-expert control.

Memory decay is also part of the scoring semantics. Park et al. combine recency, importance, and relevance in retrieval. Industrial implementations can extend this pattern by applying configurable temporal decay to importance and surprise, so older memories can remain retrievable while exerting less control pressure unless they remain semantically or operationally relevant. This allows the loop to balance recent disturbances against older but highly diagnostic precedent.

Figure 3 maps the scores to the four ORPA phases. The mapping is the architectural specification of which control signal acts where, and is itself a configurable property of the loop.

4.3 Reflect

Reflection is the loop's meaning-making step. Not every observation produces a reflection. Reflection is triggered when accumulated significance crosses a configured threshold. This threshold-gated behavior matters. It prevents continuous over-deliberation and mirrors operator practice: experienced operators notice many signals, but only some warrant active reflection.

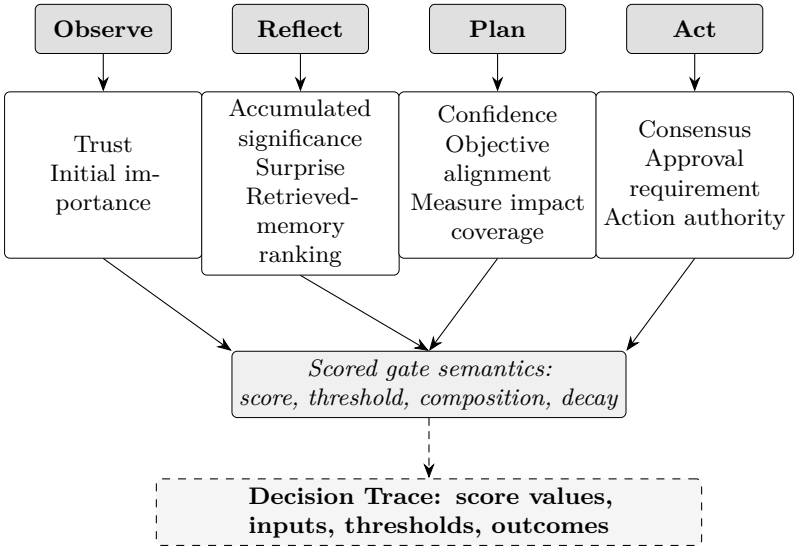

Figure 3: Score semantics across the ORPA cycle. Each phase consumes a distinct set of scores. Trust and initial importance gate the Observe phase, accumulated significance and surprise gate Reflect, confidence and objective alignment gate Plan, and consensus and action authority gate Act. All score values, the inputs that produced them, and the thresholds they were evaluated against are recorded in the Decision Trace.

Reflection retrieves relevant prior memories, which may include observations, prior reflections, summarized memories, procedures, knowledge-base content, and context from other agents. Retrieval should be relevance-scored rather than merely chronological. A recent observation is not necessarily more important than an older but highly relevant precedent.

This phase adapts Park et al.'s generative-agent architecture: importance-scored observations, cumulative significance, threshold-triggered reflection, and relevance-based retrieval. The industrial extension is governance grounding and multi-dimensional scoring. Park et al. use importance and accumulated significance to support believable social behavior. The Cognitive Decision Loop extends this into a scoring substrate in which importance, surprise, confidence, and source trust can act as independent or composite control signals across different phases of the loop.

Reflection triggering should be represented as a decision event, including cases where reflection is not triggered. Candidate memories can be retrieved and ranked using semantic relevance, decayed importance, decayed surprise, memory type, and recency. The agent then accumulates significance over the ranked set until either the configured threshold is reached or the candidate set is exhausted. This makes non-events reviewable: a skipped reflection is still a governed decision that can be inspected later.

Reflection should be shaped by domain knowledge, SOPs, deontic rules, objective context, and expert-configured parameters. Its purpose is to determine what fact-based information means for the current operational state. Behavioral believability is a downstream consideration, not the design driver.

4.4 Plan

Reflections do not always produce plans. Planning is conditional. A reflection may conclude that the current state remains acceptable, that no action is warranted, or that monitoring should continue. This matters because an industrial agent should not treat every reflection as a call to act.

When planning is warranted, the plan expresses operational intent. It may define a goal, tasks, dependencies, constraints, responsible agents, required consensus, or human review. Planning may involve an LLM as a cognitive service, but the LLM should not own the planning architecture or the action boundary. Planning is one function inside a governed loop.

Plan inherits from the generative-agent insight that behavior depends on planning as an architectural module rather than a single prompt response. The industrial extension is objective-driven intent. Plans must be evaluated against objective functions, utility functions, constraints, consensus requirements, and allowed action pathways.

Planning confidence should not be a single model self-rating. In an industrial loop, confidence is better represented as a composite assessment of reasoning quality, evidence strength, consistency with prior decisions, stability under changing context, objective-function alignment, expected measure impacts, and safety or constraint sensitivity. Low confidence does not necessarily mean the plan is wrong; it means the plan has insufficient authority to proceed without additional validation.

4.5 Act

The Act phase is bounded. Plans should become actions only through configured action channels. If no configured action channel exists for a proposed action, the intent should be routed to a human operator or reviewer. The agent runtime itself should not directly actuate simply because a model-generated plan proposes an action.

This is one of the most important distinctions between governed industrial agents and open-ended tool-calling agents. In industrial operations, action authority matters. A human operator may understand what should be done but still lack authority to do it directly. The operator escalates, requests approval, or hands off to an authorized role. The same principle should apply to agents.

Escalation to humans occurs when the decision loop produces operational intent without sufficient authority to execute it autonomously. This may occur because confidence is below a configured threshold, projected impacts affect interdependent team objectives, constraint thresholds are breached, the proposed action is classified as approval-required, or no configured action channel exists. In these cases, the agent converts the decision into a reviewable communication or approval request, preserving the scores, evidence, objective impacts, and constraint context that caused escalation.

Bounded actuation is an architectural property of the runtime, enforced by the dispatch boundary rather than promised by behaviour. The system should make it structurally impossible for an agent to act outside configured pathways.

4.6 Decision Trace

The Decision Trace is the architectural audit concept associated with the loop. It captures the provenance of observations, reflections, planning decisions, constraint evaluations, consensus events, human reviews, and action dispatches. It is what allows a reviewer to reconstruct how an agent moved from operational input to operational intent.

If scores are control signals, they should also be first-class elements of the trace. Score values, the inputs that produced them, the decay or composition rules applied to them, and the thresholds against which they were evaluated should be represented as reviewable provenance. This allows a reviewer to ask not only what the agent decided, but why a gate fired or did not fire.

W3C PROV-O can provide a vocabulary for representing parts of a Decision Trace, especially when records are naturally modeled as activities, entities, and agents. The concept should not be over-specified. Decision Trace is the public architectural concept. Specific implementations may represent different portions of the trace in graph stores, event stores, time-series stores, audit logs, or provenance ontologies.

This flexibility matters. The architectural requirement is that the decision process be inspectable. PROV-O is a strong representation choice, but the concept of Decision Trace should remain portable across implementations.

5 Governance and Traceability

The Cognitive Decision Loop separates governance into two layers: planning-layer governance and actuation-layer governance.

At the planning layer, constraints should be evaluated, conflicts surfaced, and violations recorded. This layer is advisory and auditable. It supports reflection, consensus, escalation, and human review. It should not silently suppress all plans that violate a soft constraint, because some operational situations require making a constrained trade-off explicit.

At the actuation layer, governance is bounded and enforced. Plans become actions only through configured action channels. This layer determines what the agent can actually do.

The distinction mirrors mature industrial control systems. Deviation alarms surface conditions requiring operator judgment. Interlocks enforce hard limits. In agent architecture, planning-layer governance is closer to deviation alarming: it makes conflicts visible. Actuation-layer governance is closer to interlocking: it enforces configured boundaries.

This distinction also clarifies the compliance posture. Industrial agents should not be described as "always compliant." That is a behavioral promise no agent architecture should make without formal proof and operational qualification. Agents can be designed to be bounded, auditable, and consistent in configured workflows.

Bounded means the agent cannot act outside configured action channels. *Auditable* means the decision process leaves records that can be reviewed. *Consistent* means configured thresholds, objectives, and constraints are applied without fatigue or informal drift. These are architectural properties. They do not amount to claims of perfection.

Decision Trace is the audit substrate for this governance model. Bounded actuation controls what the system can do. Decision Trace shows how the system got there. Together, they support incident review, assurance, regulatory response, insurance review, and operational learning.

6 Reference Implementation: XMPro MAGS

XMPro MAGS is a reference implementation of the Cognitive Decision Loop for industrial operations. It is presented here as one implementation of the pattern, not as the only possible implementation. The architectural details that follow are presented as a *reference-implementation disclosure*: public XMPro documentation [27, 28] supports the conceptual framing of MAGS and the Decision Trace pattern, while specific storage-layer and scoring-layer details are described from author access to the implementation rather than as independently verifiable claims.

MAGS implements the loop as a parametric multi-agent architecture. Observations are supplied through operational data pipelines, including XMPro DataStreams, which can carry pre-processed

engineering calculations, statistical predictions, sensor-derived indicators, and other fact-based inputs. Agents are configured with profiles that include significance thresholds, knowledge sources, tools, allowed actions, objective and utility functions, and governance rules.

Parametric configurability means that thresholds, decay parameters, score weights, action channels, approval-required classifications, consensus triggers, and communication routing rules are defined as administrative configuration rather than as runtime model behavior. This distinction matters because the governed behavior of the loop should be inspectable and adjustable by domain experts without requiring a model change.

The multi-agent dimension is important. An industrial operation rarely has a single objective. Maintenance, production, quality, energy, safety, and economics often pull in different directions. A maintenance-optimal action may not be production-optimal. A production-optimal action may not be safety-optimal. A Cognitive Decision Loop implementation must provide a place for these tensions to surface.

In MAGS, teams of agents can share a team objective while individual agents pursue their own objective functions. Consensus can be invoked conditionally when objectives conflict, when constraint conflicts exceed configured thresholds, or when a team-level decision requires coordination. Consensus thresholds are evaluated over participating agents' confidence, objective-impact, and constraint-context signals rather than over free-form dialogue alone. The aim is to structure multi-agent coordination around objectives, utility trade-offs, constraints, records, and review pathways, rather than to have agents persuade one another through unconstrained dialogue.

Decision artifacts are persisted as auditable records. In MAGS, observation and reflection records are represented as PROV-O Decision Traces in a dedicated graph store, while planning, consensus, constraint, and dispatch artifacts are persisted through other audit stores. This distinction matters: the public architectural concept is Decision Trace across ORPA phases, while the implementation may distribute different parts of the trace across storage mechanisms.

6.1 Decision Artifacts as Units of Evaluation

The reference implementation also suggests a different unit of evaluation for industrial agents. Rather than evaluating only final model outputs, reviewers can evaluate decision artifacts: observations, reflection decisions, reflections, planning decisions, plans, consensus records, communications, approval requests, constraint evaluations, and action dispatches. Each artifact can carry normalized scores, contributing evidence, threshold comparisons, model metadata, objective-function context, and links to prior artifacts.

This makes the system inspectable at the level of decision control rather than only at the level of generated text. A recommendation can be evaluated not only for whether it was plausible, but for whether the right observations were admitted, the right memories were retrieved, the right scores were applied, the right gates fired, the right parties were consulted, and the proposed action stayed within configured authority.

Figure 4 shows the artifacts produced across a single loop traversal and the evaluation surface they collectively present.

A practical consequence of the parametric architecture is model-generation portability. When a new language model is adopted, it can be introduced as a service inside the existing decision architecture. The decision architecture is the asset; the model is a component.

As an illustrative scenario, consider a maintenance agent and a production agent receiving an observation that compressor vibration has increased while throughput demand remains high. The

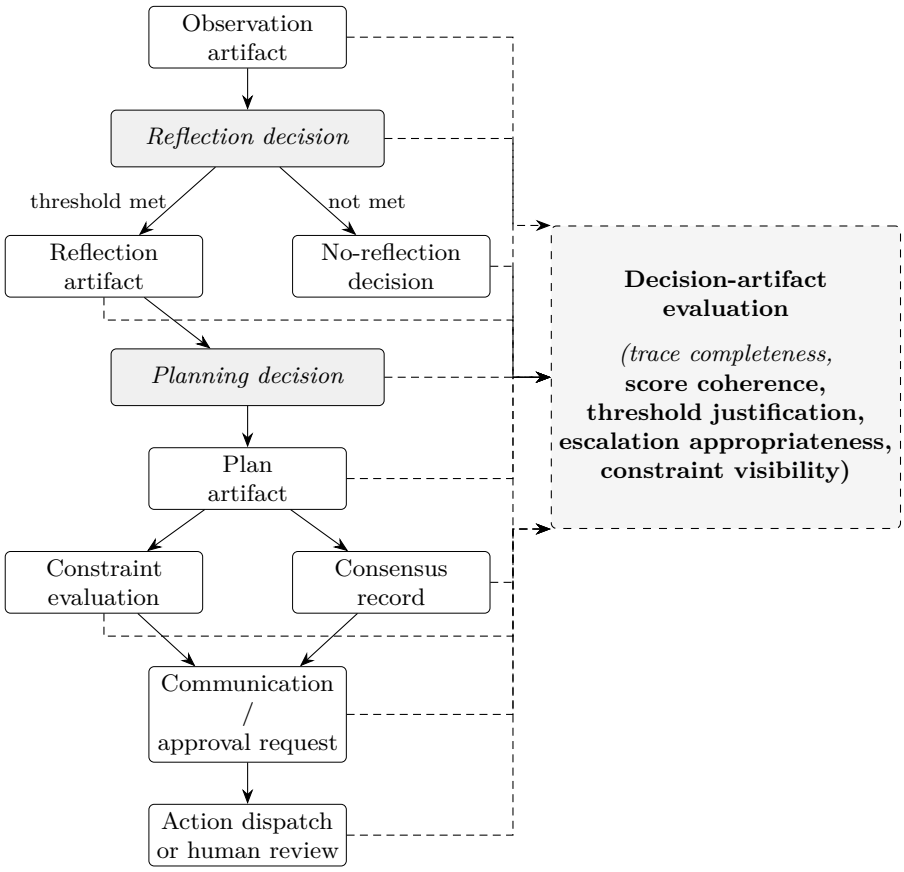

Figure 4: Decision artifacts as units of evaluation. Each phase of the loop produces a typed artifact — observation, reflection decision, reflection, planning decision, plan, constraint evaluation, consensus record, communication or approval request, action dispatch — and the full set forms an evaluable surface independent of the final generated text. Industrial agent benchmarks can score these artifacts on trace completeness, score coherence, threshold justification, escalation appropriateness, and constraint visibility.

maintenance agent reflects against asset-health objectives. The production agent reflects against throughput and quality objectives. The team evaluates trade-offs, records constraint conflicts, generates operational intent, and routes any action through configured action boundaries or human review. This scenario illustrates why the architecture must represent observations, reflection, planning, consensus, traceability, and bounded action as separate concerns.

6.2 Illustrative Scoring Semantics

The reference implementation can be characterised through illustrative scoring patterns that show how importance, surprise, confidence, and escalation are computed, without exposing implementation-specific weights or storage structures. The examples below are schematic. They describe the scoring semantics of the reference implementation rather than its proprietary internals.

Memory retrieval and reflection significance. Each candidate memory in the retrieval window is assigned a score that combines semantic similarity with the current observation, decayed importance, and decayed surprise, modulated by a type weight and a recency factor:

```
memory_score =
  type_weight * recency_factor *
  ( w_similarity * semantic_similarity
  + w_importance * decayed_importance
  + w_surprise   * decayed_surprise )
```

Reflection significance is the sum of memory scores over the ranked candidate set, and the agent triggers reflection only if accumulated significance crosses a configured threshold:

```
reflection_significance = sum( memory_score_i
                               for i in ranked_candidates )

if reflection_significance >= configured_reflection_threshold:
    trigger reflection
else:
    record no-reflection decision
```

Both outcomes — reflection and no-reflection — are recorded as decision artifacts in the Decision Trace. This supports the architectural claim that non-events are themselves reviewable decisions.

Contextual surprise. Surprise is computed as a context-conditioned combination of semantic deviation from prior memories, contextual unexpectedness, and activity rarity:

```
surprise =
  w_pattern  * semantic_deviation
  + w_context * context_unexpectedness
  + w_temporal * activity_rarity
```

A compressor vibration increase may be less surprising during a known process transient and more surprising during otherwise stable operation. Surprise is therefore conditioned by the operational context in which the observation arises, not by the observation alone.

Planning confidence and escalation. Planning confidence is computed as a composite assessment across reasoning quality, evidence strength, historical consistency, plan stability, and objective-function alignment:

```
planning_confidence =
  w_reasoning    * reasoning_quality
  + w_evidence    * evidence_strength
  + w_consistency * historical_consistency
  + w_stability   * plan_stability
  + w_objective   * objective_alignment
```

The composite score gates two governance pathways: consensus invocation and human review.

```
if planning_confidence < consensus_threshold:
    invoke consensus

if planning_confidence < critical_threshold
   or constraint_status == breached
   or action_requires_approval
   or no_authorized_action_channel:
    route to human review
```

Low confidence does not by itself imply that the plan is wrong. It indicates that the plan has insufficient authority to proceed autonomously and must be validated through consensus or human review before any action is dispatched.

The schematic examples above illustrate the scoring semantics of the reference implementation without specifying proprietary weights, storage structures, or implementation code. The contribution is the architecture of scored gating, not the particular numerical configuration.

7 Comparison With LLM Reasoning Agents

The Cognitive Decision Loop does not reject reasoning models. It provides an architecture in which reasoning models can be used safely and productively as cognitive services.

Dimension	**LLM Reasoning Agent**	**Cognitive Decision Loop**
Primary locus of control	Model-internal reasoning or tool loop	Explicit decision architecture
Observation model	Context window and tool results	Fact-based operational state
Reflection trigger	Implicit or prompt-driven	Significance-gated
Gate semantics	Implicit, prompt-driven, or tool-policy driven	Explicit scored thresholds with configurable composition
Planning	Generated plan or tool sequence	Conditional operational intent
Governance	Prompt/tool policy and model behavior	Constraints, consensus, traceability, bounded actuation
Auditability	Conversation/tool logs	Decision Trace
Action boundary	Tool-call permissions	Configured action channels or human review
Configurability	Prompt, model, reasoning effort, tool list	Domain-expert-editable parameters across the decision architecture
Compliance claim	Behavioral, via training and prompting	Architectural, via bounded actuation and auditability

Table 1: Architectural comparison between LLM reasoning agents and the Cognitive Decision Loop. The comparison is descriptive rather than evaluative; the two patterns are appropriate for different problem classes.

The code-as-harness framing [16] helps clarify the distinction. Code-centric agents gain reliability when reasoning and action are externalized into executable artifacts, tests, traces, and shared program state. Industrial agents require the same move away from model-internal reasoning, but the harness substrate changes. In MAGS-style systems, the governing substrate is not

primarily code but scored decision artifacts, operational measures, constraints, objective functions, consensus records, approval pathways, and bounded actuation.

The comparison is not a quality judgment. Reasoning agents excel where the problem is content-rich, the blast radius is bounded, and the primary goal is to produce a high-quality answer or complete a knowledge-work task. Cognitive Decision Loops are appropriate where decisions affect physical or system state under governance constraints, and where the audit surface needs to survive scrutiny by operators, regulators, insurers, and incident reviewers.

The two approaches are complementary. Reasoning models can improve the cognitive services inside a Cognitive Decision Loop. They may improve interpretation, synthesis, communication, planning drafts, and explanation. The model improvement does not, by itself, supply the loop. The industrialization of the generative-agent pattern lies in moving from memory and reflection for behavioral believability to scored, traceable, governed decision control.

7.1 Reasoning Posture as a Per-Agent Configuration

The complementarity between LLM reasoning and the Cognitive Decision Loop has a per-agent dimension that is worth making explicit. The Cognitive Decision Loop architecture itself is uniform across agents: every agent traverses Observe, Reflect, Plan, Act with the same scored gates and the same Decision Trace requirements. What varies between agents is the reasoning posture configured for the LLM services inside the loop, calibrated to the agent's Objective Function, governance posture, and actuation authority.

A virtual-operator agent maintaining a process unit within Standard Operating Procedures and operating limits has tight constraints. Its Objective Function rewards staying within configured envelopes; its action authority is bounded to specific control variables; its reflection draws on SOPs and procedural knowledge. Extended reasoning offers limited benefit for this agent. Its task is to apply known patterns to fact-based observations, not to invent new ones, and additional deliberation depth introduces variability that the actuation-level governance posture is not designed to absorb.

A process-engineering agent looking at the same plant data for efficiency opportunities has a different posture. Its Objective Function rewards finding optimisation opportunities that improve yield, energy efficiency, or quality. Its reflection draws on process engineering principles, historical optimisation campaigns, and counterfactual analysis. Extended reasoning is more clearly valuable here: the agent benefits from deeper chain-of-thought, from exploring multiple causal hypotheses, and from comparing alternative interventions that may not be in any prior playbook. Its action authority is typically bounded to recommendations rather than direct actuation, which makes the higher reasoning budget acceptable because human review or consensus provides the governance gate.

Both agents live inside the same Cognitive Decision Loop architecture and produce the same Decision Trace artifacts. What differs is how their LLM services are configured: which models, with what reasoning-effort budget, against what knowledge bases, under what prompt structure. The choice is parametric and made by subject-matter experts based on the rigour requirements of the Objective Function, the governance requirements of the actuation pathway, and the operational role of the agent in the broader industrial system. Reasoning depth becomes a per-agent design choice rather than a system-wide architectural commitment.

8 Discussion

The Cognitive Decision Loop reframes industrial agent evaluation. Instead of asking only which model reasons best, architects and buyers should ask where the control architecture lives. Is it hidden inside model weights and prompts, or is it represented in explicit decision structures, thresholds, objectives, constraints, traces, and action boundaries?

Fact-based observations reduce the risk of hallucinated operational state. They do not eliminate LLM failure modes, but they shift the agent's role from inventing values to interpreting supplied state. Significance thresholds reduce continuous over-deliberation. Reflection becomes a gated event rather than a constant stream of model output. Conditional planning prevents every interpretation from becoming an action proposal. Bounded actuation ensures that intent and effect remain separated by configured authority.

Decision Trace makes the loop reviewable. It is part of the architecture rather than a logging feature added afterward. Without traceability, an agent's recommendation may be difficult to inspect after the fact. With traceability, reviewers can ask which observations were used, why reflection occurred, which memories were relevant, what constraints were checked, whether consensus was required, and how action was bounded.

The architecture also supports model evolution. Language models will continue to change. Some will reason better, some will be faster, some will be smaller, some will be deployable locally. If the agent architecture is defined primarily by a prompt and a model choice, every model transition becomes an architectural disruption. If the architecture is a governed decision loop, the model can change while the decision architecture remains stable.

The unit of industrial agent design is therefore the governed decision loop. Prompts and model choices are configurable elements inside that loop.

9 Limitations

The Cognitive Decision Loop is an architectural pattern, not a proof of safety. It can support governed behavior, auditability, and bounded actuation, but implementation quality matters. A poorly implemented loop can still fail.

LLM-derived components still inherit LLM failure modes. Observation interpretation, reflection synthesis, communication drafting, planning decisions, and plan generation may still be affected by hallucination, overconfidence, prompt sensitivity, or incomplete grounding. The architecture bounds these risks by keeping LLMs inside explicit control structures, but it does not eliminate them.

Parametric configuration requires domain-expert work. Objectives, utility functions, thresholds, constraints, knowledge sources, allowed actions, and review pathways must be defined and maintained. This is a cost. In governance-constrained industrial operations, it is often the right cost, but it is not appropriate for every use case.

Scoring quality is itself a limitation. Importance, surprise, confidence, trust, decay, and objective-alignment scores can be miscalibrated. A system may overestimate significance, underestimate low-probability risks, or assign confidence poorly relative to ground truth. The architectural response is to make scores, thresholds, and outcomes traceable so that retrospective calibration analysis can improve the configuration over time. Score correctness is assessed through the trace rather than assumed by construction.

Decision Trace quality depends on implementation fidelity. If records are incomplete, inconsistent,

or not linked across phases, the audit value degrades. The architectural concept must therefore be supported by disciplined implementation.

The cognitive-science framing is analogical. The Cognitive Decision Loop is not a claim about biological fidelity, consciousness, or neural implementation. It maps onto useful structural features of expert decision-making without reproducing human cognition.

Finally, the paper does not yet present empirical benchmark results. Future work should compare ORPA-style agents with LLM-only tool agents in controlled industrial scenarios and evaluate decision quality, trace completeness, operator trust, and governance outcomes.

10 Future Work

Several research directions follow from this paper.

First, ORPA-style agents should be evaluated empirically against LLM-only tool agents in operational scenarios. Such evaluations should measure not only task success, but also trace completeness, constraint visibility, intervention quality, and bounded-action behavior.

Second, Decision Trace completeness metrics should be developed. A trace should allow reviewers to reconstruct observations, reflection triggers, retrieved context, planning decisions, constraints, consensus, human review, and action dispatch. The completeness and coherence of such traces can be evaluated independently of model benchmark scores.

Third, human factors studies are needed. Operators, engineers, and supervisors should evaluate whether Decision Traces are understandable, whether recommendations are appropriately calibrated, and whether bounded actuation improves trust without encouraging overreliance.

Fourth, the Cognitive Decision Loop should be mapped into industrial assurance frameworks. This includes alignment with NIST AI RMF, ISO/IEC 42001, IEC 62443, and potentially domain-specific safety-case practices. Such mapping should be careful: the architecture can support assurance, but it should not be treated as certification by itself.

Fifth, formal semantics for objective functions, utility functions, deontic constraints, and action boundaries would strengthen the pattern. A future Decision Trace specification could become an interoperable standard for governed industrial agents.

Sixth, industrial agent benchmarks should evaluate decision-artifact quality. Relevant metrics include trace completeness, score coherence, threshold justification, escalation appropriateness, constraint visibility, and consistency between objective impacts and final recommendations. These metrics would complement task-success benchmarks by measuring whether the decision process itself is governable.

11 Conclusion

Reasoning-focused large language models are a significant capability advance. They improve model-internal inference and can serve as powerful cognitive services inside agent systems. Industrial agents, however, require more than reasoning depth. They require governed decision architectures.

The Cognitive Decision Loop provides one such architecture. It adapts generative-agent memory, reflection, retrieval, and planning patterns into an industrial setting, combines them with insights from operator decision theory, and extends them with governance mechanisms: fact-based

observations, scored gating across phases, objective and utility functions, constraints, consensus, Decision Traces, and bounded actuation.

XMPro MAGS is one reference implementation of this pattern. Its broader significance is that it places large language models inside a governed operational decision loop, with scored, traceable decision artifacts as the unit of evaluation.

Industrial AI agents should therefore be evaluated on decision architecture and actuation boundary, not only on reasoning depth. The relevant questions are architectural: where does control live, what is configurable by domain experts, what is traceable across phases, where is action bounded, and what remains reviewable after the decision?

References

[1] PROV-O: The PROV Ontology. W3C Recommendation, 2013. URL https://www.w3.org/TR/prov-o/.

[2] Anthropic. Introducing the Model Context Protocol. Anthropic announcement, November 2024. URL https://www.anthropic.com/news/model-context-protocol. Accessed May 2026.

[3] John R. Boyd. Destruction and Creation. Unpublished essay, 1976.

[4] Andy Clark. *Surfing Uncertainty: Prediction, Action, and the Embodied Mind*. Oxford University Press, 2016.

[5] DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z. Also available as arXiv:2501.12948 (January 2025).

[6] Jonathan St. B. T. Evans and Keith E. Stanovich. Dual-Process Theories of Higher Cognition: Advancing the Debate. *Perspectives on Psychological Science*, 8(3):223–241, 2013.

[7] Karl Friston. The Free-Energy Principle: A Unified Brain Theory? *Nature Reviews Neuroscience*, 11(2):127–138, 2010.

[8] International Electrotechnical Commission. IEC 62443-3-3:2013 Industrial Communication Networks — Network and System Security — Part 3-3: System Security Requirements and Security Levels. Technical report, IEC, 2013.

[9] International Electrotechnical Commission. IEC 61511-1:2016+A1:2017 Functional Safety — Safety Instrumented Systems for the Process Industry Sector — Part 1: Framework, Definitions, System, Hardware and Application Programming Requirements. Technical report, IEC, 2016.

[10] International Organization for Standardization. ISO/IEC 42001:2023 Information Technology — Artificial Intelligence — Management System. Technical report, ISO, 2023.

[11] Philip N. Johnson-Laird. Mental Models and Human Reasoning. *Proceedings of the National Academy of Sciences*, 107(43):18243–18250, 2010.

[12] Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.

[13] Gary Klein. *Sources of Power: How People Make Decisions*. MIT Press, 1998.

[14] Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, et al. From System 1 to System 2: A Survey of Reasoning Large Language Models. arXiv:2502.17419, February 2025.

[15] National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework (AI RMF 1.0). Technical report, NIST, 2023.

[16] X. Ning et al. Code as Agent Harness. arXiv:2605.18747, May 2026. URL https://arxiv.org/abs/2605.18747v1.

[17] OpenAI. OpenAI o1 System Card. arXiv:2412.16720, December 2024.

[18] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior. arXiv:2304.03442, 2023. UIST '23.

[19] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative Agents for Software Development. arXiv:2307.07924, 2023.

[20] Anand S. Rao and Michael P. Georgeff. Modeling Rational Agents within a BDI-Architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484, 1991.

[21] Anand S. Rao and Michael P. Georgeff. BDI Agents: From Theory to Practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 312–319, 1995.

[22] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. arXiv:2302.04761, 2023.

[23] Claude E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[24] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models. arXiv:2305.16291, 2023.

[25] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903, 2022.

[26] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. arXiv:2308.08155, 2023.

[27] XMPro. Decision Traces for Agentic Operations: Why Agents Need Operational Memory. XMPro Engineering Blog, 2026. URL https://xmpro.com/decision-traces-for-agentic-operations-why-agents-need-operational-memory/. Accessed May 2026.

[28] XMPro. Multi-Agent Generative Systems (MAGS). XMPro product documentation, 2026. URL https://xmpro.com/agentic-ai/multi-agent-generative-systems/. Accessed May 2026.

[29] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629, 2022.

[30] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. arXiv:2305.10601, 2023.