

Article

A Hybrid BDI + RAG Multi-Agent Architecture: Plan-Based Reasoning over Multimodal Retrieval

Halil Yesil ^{1,2} , Baris Tekin Tezel ^{2,*}  and Moharram Challenger ³ ¹ Siskon Industrial Automation and Software, Izmir 35390, Turkey; halilyesilceng@gmail.com² Department of Computer Science, Dokuz Eylul University, Izmir 35390, Turkey³ Department of Computer Science, University of Antwerp and Flanders Make, 2020 Antwerp, Belgium; moharram.challenger@uantwerpen.be

* Correspondence: baris.tezel@deu.edu.tr

Abstract

Retrieval-Augmented Generation (RAG) gives language models access to external text and image collections, but it does not make the decisions taken on that content reproducible. Classical Belief–Desire–Intention (BDI) agents provide explicit plans and traceable decisions, although they normally expect beliefs in symbolic form. We connect these capabilities in a hybrid architecture in which RAG updates agent beliefs while BDI plans remain responsible for reasoning and coordination. The plans are implemented in AgentSpeak on JASON/JADE, agents communicate through FIPA-ACL, and probabilistic models are accessed as computational services through the Model Context Protocol (MCP). The orchestrator routes messages but does not derive authorization decisions. Domain agents and the fusion agent produce those decisions through plan execution. We instantiated it for multimodal access control over 167 research posters and compared it with AutoGen and LangGraph baselines on an identical service backend driven by one locally served model, qwen3.5:9b, so that only the control layer differs. Eight campaigns each vary one setting: pipeline length, tool declaration order, the prerequisite and terminal hints, the sampling seed and the fusion policy. They total 4800 baseline runs, with the plan-based arm measured in the two campaigns that have a plan-library counterpart, 300 runs each. Under a favorable configuration, both baselines schedule the workflow correctly and no verdict changes between repeats, yet only 52% and 86% of queries reproduce the same sequence. Their valid-schedule rate falls to 32.0% and 28.0% at nine agents, and presentational edits to the tool declarations move the ordering and, in one arm, the verdict. The plan-based arm returns 100.0% valid schedules and a single sequence in both campaigns.

Keywords: Belief–Desire–Intention agents; Retrieval-Augmented Generation; hybrid architecture; deterministic reasoning; Model Context Protocol; multi-agent systems



Academic Editor: Paolo Renna

Received: 20 July 2026

Revised: 3 September 2026

Accepted: 7 September 2026

Published: 30 September 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

RAG is commonly used to answer questions over domain collections by retrieving relevant material before an LLM generates a response [1]. Retrieval improves grounding, but it does not change how the model generates or how an LLM-based workflow selects its next action. Repeating the same request can therefore alter the answer, an extracted fact, or the path taken through a multi-agent workflow. That variability is often acceptable in exploratory question answering. It becomes harder to accept when a system grants access, applies a policy, or must later explain why a particular action was taken.

BDI agents address a different part of this problem. Their beliefs, goals, and plan-selection rules are explicit, allowing a decision to be related to the plan and beliefs that produced it [2]. In return, a BDI agent normally expects the relevant beliefs to have already been represented symbolically. This assumption is restrictive for repositories in which evidence is embedded in prose, images, charts, or visual metadata.

Our study asks whether retrieval and multimodal extraction can supply such evidence without transferring decision authority to an LLM. We therefore place RAG at the belief-formation boundary and retain BDI plans for reasoning, fusion, and coordination. This separation does not make the complete pipeline deterministic. Rather, it isolates probabilistic extraction and generation from the rules that select authorization outcomes and coordination steps. The paper evaluates what this separation guarantees, what remains probabilistic, and how the design behaves in a multimodal access-control implementation.

The design problem has three parts. First, retrieval can ground an LLM response without making generation or workflow control deterministic. Second, BDI plans can make a decision traceable, but only after relevant information has been represented as beliefs. Third, the boundary between these functions must prevent a probabilistic model from becoming the component that selects a policy outcome.

Retrieval reduces some causes of hallucination but does not remove them [3,4]. An LLM can still add an unsupported detail, associate a statement with the wrong source, or return different formulations for the same query. These outcomes arise because generation remains probabilistic even when the retrieved context is fixed. Existing RAG research therefore evaluates retrieval quality and generation faithfulness, among other properties [5], but retrieval alone cannot provide a rule-level account of why a security or compliance outcome was selected.

The architecture proposed here addresses variability at the reasoning and orchestration layer. Authorization judgments are derived by AgentSpeak plans, and the next coordination step is selected through plan execution and the agents' message protocol. The final natural-language response remains an LLM output and may still contain an error. The same is true of facts produced during extraction. These boundaries are part of the claim, not exceptions to it, and are examined again in Section 6.10.

AgentSpeak plans operate on symbolic beliefs. In the repository considered in this study, however, authorization evidence may be a sentence in a poster, a logo, a badge, or another visual element. Encoding every such observation by hand would require repeated knowledge-engineering work whenever documents or policies changed. A practical BDI-based solution therefore needs a controlled way to obtain candidate beliefs from unstructured and multimodal sources while preserving the plan library as the place where decisions are made.

The two research lines leave a specific implementation gap. RAG studies generally evaluate retrieval and generation, while BDI studies commonly assume that the beliefs required by a plan are already available in symbolic form. Neither assumption is sufficient for our setting: an authorization decision must use evidence retrieved from current multimodal documents, but the rule that converts that evidence into a decision must remain inspectable. This requirement, rather than the broader claim that no previous integration is principled, is the design point for the proposed architecture.

We selected BDI because the authorization rules in this study can be represented as conditions over explicit beliefs and goals. This gives each outcome a program-level explanation: an audit can identify the beliefs present at the time of the decision, the applicable AgentSpeak plan, and the action produced by that plan. We do not rely on the stronger analogy that BDI reproduces human cognition; its value here is operational and inspectable rule execution.

The JASON platform [6] implements the AgentSpeak programming language with formal semantics for BDI reasoning. JASON provides belief revision, goal handling, plan selection, and failure recovery. The authorization policy is consequently stored in plan rules rather than prompts. An auditor can inspect these rules without inferring them from model weights or generated explanations.

JADE (Java Agent Development Framework) [7] provides the distributed runtime infrastructure on which JASON agents are deployed. JADE implements the FIPA-ACL communication standard, allowing messages to distinguish requests to act from reports of beliefs or outcomes. This distinction is useful in the fixed protocol described in Section 4.3. In the implemented system, JASON is responsible for deliberation within an agent, while JADE carries the messages through which specialized agents coordinate.

1.1. Proposed Approach and Contributions

We implemented a boundary between probabilistic services and rule-based agents. MCP services retrieve or structure information; domain agents insert the returned facts into their belief bases and evaluate them through AgentSpeak plans. The orchestrator controls the message sequence but does not select an authorization outcome. When several domain judgments must be combined, it forwards them to a designated fusion reasoning agent and waits for that agent's plan-derived result before continuing. This paper is a substantial extension of our earlier conference paper [8], which first introduced the BDI-governed multimodal RAG pipeline for the access-control use case revisited in Section 5. That work demonstrated the feasibility of a single, access-control-specific instantiation; the present paper generalizes it into a domain-independent architecture— formalizing the four canonical agent roles, the three-layer interaction model, and the fusion-as-reasoning principle, and adding the structured architectural comparison, the eight-campaign sensitivity study, and the decision-quality evaluation of Section 6.

The contributions of this paper are the following:

1. **Hybrid BDI + RAG Architecture:** A general architectural framework that specifies where retrieved information becomes a belief and where a belief becomes a decision. LLMs and other models are invoked through MCP services; authorization and fusion outcomes are produced by AgentSpeak plans.
2. **Multi-Level Interaction Model:** A layered interaction design spanning three levels: agent-to-agent coordination via FIPA-ACL speech acts; agent-to-service communication via MCP JSON-RPC; and service-to-knowledge-base access via semantic vector retrieval. The layers can be inspected and replaced separately, subject to their interface contracts.
3. **Multimodal Belief Extraction:** A mechanism for populating BDI belief bases from textual and visual content through parallel extraction pipelines. The agents reason over symbolic terms even though the evidence originates in different modalities.
4. **Use Case Demonstration:** A concrete instantiation of the architecture for multimodal access control over a multi-institutional document repository, in which authorization evidence appears in text, in images, or in both.

1.2. Paper Organization

The remainder of this paper is structured as follows. Section 2 provides background on the component technologies. Section 3 reviews related work and positions the hybrid architecture against existing approaches. Section 4 presents the general BDI + RAG hybrid architecture at both conceptual and interaction levels. Section 5 demonstrates the architecture through an access control use case. Section 6 specifies the baselines and metrics, reports the eight-campaign orchestration sensitivity study and the decision-quality results,

compares fusion policies, and discusses the architectural properties that distinguish the approach from existing paradigms. Section 7 concludes the paper.

2. Background

The hybrid BDI + RAG architecture combines four component technologies, each supplying a property the others lack: Retrieval-Augmented Generation grounds agent beliefs in dynamically queried knowledge, the Belief–Desire–Intention model provides a deterministic, inspectable reasoning substrate, Multi-Agent Systems with the FIPA-ACL standard provide the coordination infrastructure, and the Model Context Protocol provides the boundary that keeps probabilistic computation outside the reasoning loop. This section summarizes the property of each that the hybrid design relies on; no single component offers all of them, which is what motivates the integration.

2.1. Retrieval-Augmented Generation and Language Models

Retrieval-Augmented Generation (RAG) [1] addresses the central limitation of parametric language models—knowledge frozen at training time—by retrieving relevant external passages at inference time and conditioning generation on them rather than on memorized associations alone. A pipeline indexes a corpus offline by chunking documents and encoding each passage into a dense vector with an embedding model; at query time it embeds the query, retrieves the top- k passages by cosine similarity, and supplies them as context to a language model that generates the response. Embedding models are trained so that semantic similarity corresponds to geometric proximity, which lets retrieval find topically relevant passages without exact keyword matches; domain-specialized embedding models capture finer distinctions than generic ones.

The language and embedding models on which RAG rests are probabilistic. A large language model [9] generates text by autoregressively sampling tokens from a learned conditional distribution, so the same query and context can yield different responses across invocations, and the boundary between retrieved fact and generated content is not formally enforced: the model may blend, hallucinate, or misattribute claims [3]. Ref. [5] surveyed advanced RAG variants—reranking, query rewriting, iterative retrieval—that improve faithfulness but leave the probabilistic nature of generation intact. The hybrid architecture therefore scopes RAG narrowly: it acts as a belief-update mechanism feeding an execution-deterministic BDI reasoning layer, confining this non-determinism—together with the multimodal extraction performed by vision-language models—to an extraction boundary rather than letting it propagate into reasoning and orchestration.

2.2. Belief–Desire–Intention (BDI) Agents

The BDI model [2] formalizes practical reasoning through three mental attitudes drawn from Bratman’s account of intentional action [10]: *beliefs* (the agent’s revisable knowledge, encoded as logical propositions), *desires* (the outcomes it would pursue), and *intentions* (the desires it has committed to). An agent’s *plan library* holds conditional rules, each with a triggering event, an applicability context, and a body of actions or subgoals; at each cycle the agent matches triggered plans, filters them by context, selects one to execute—means-end reasoning—and falls back to alternatives on failure.

The property that motivates BDI’s role here is *execution determinism*: given the same belief base and plan library, an agent always selects the same plan and produces the same decision, and that decision is fully traceable to the belief that triggered it and the plan that handled it. The beliefs may inherit uncertainty from probabilistic extraction, but the reasoning over them stays predictable and auditable. JASON [6] interprets the AgentSpeak(L) language, expressing plans as human-readable rules, while JADE [7] supplies the

distributed runtime and FIPA-ACL messaging. The limitation the hybrid design overcomes is the *knowledge integration problem*: classical BDI assumes beliefs arrive as structured symbolic predicates, whereas modern knowledge resides in unstructured and multimodal content—a gap that RAG, mediated through MCP, closes.

2.3. Multi-Agent Systems and FIPA-ACL

Multi-Agent Systems (MASs) distribute reasoning across autonomous agents that pursue individual goals and coordinate toward outcomes none could achieve alone [11]. The paradigm suits the hybrid architecture because its concerns decompose cleanly: a retrieval agent handles query-to-vector search, one or more reasoning agents apply domain logic over private belief bases, and an orchestrator sequences the exchange—so adding a reasoning dimension means writing a new agent rather than modifying existing ones. Coordination uses the FIPA-ACL standard [7,12], which applies speech-act theory by tagging each message with a *performative* that declares its intent—request, achieve, inform, tell—alongside content, ontology, and conversation identifiers. Because every message is a communicative act rather than an opaque packet, an observer can reconstruct from the message log what each agent intended, without reading its source code, making coordination auditable. JADE’s Directory Facilitator lets new reasoning agents advertise their capabilities at startup and be discovered by the orchestrator at runtime, supporting extensibility without changes to coordination logic.

2.4. Model Context Protocol (MCP)

MCP provides the typed JSON-RPC interface used in our implementation to call external tools. A JASON agent reaches an MCP server through a Java internal action and invokes a named service, such as embedding, vector retrieval, or LLM-based extraction. The service response is returned as data that the agent can validate and insert into its belief base.

We use MCP for boundary definition rather than for reasoning. AgentSpeak plans determine when a service is called and how the returned value is used. The service cannot add a new plan or directly choose the next FIPA-ACL message. This division also allows a model or vector store to be replaced without rewriting the authorization plans, provided that the replacement preserves the agreed input and output schema.

The boundary should not be interpreted as a trust guarantee. A typed response can still contain an incorrect or manipulated fact. MCP separates components and makes calls observable; validation of belief content remains a separate requirement addressed in Section 6.10.

3. Related Work

Retrieval-Augmented Generation [1] connects large language models with external knowledge sources, reducing reliance on parametric knowledge and enabling contextually grounded responses at scale. Ref. [5] documented the rapid adoption of this paradigm across enterprise, legal, and biomedical domains. However, RAG inherits the probabilistic nature of the LLMs it augments: the answering stage samples from learned probability distributions rather than computing from formal inference rules, and the same query may produce different responses across runs [3,4]. Ref. [3] provided a foundational taxonomy of hallucination in natural language generation, distinguishing content that contradicts the source from content that cannot be verified against it; Ref. [4] extended this analysis to the LLM era, surveying the principles, causes, and taxonomy of hallucination in large language models and noting that retrieval augmentation mitigates but does not eliminate factual errors, as generation remains a probabilistic process over learned distributions.

Further studies show that this probabilistic nature creates privacy risks, as models may expose sensitive training data through membership inference attacks [13] or unintentionally reproduce copyrighted or private content verbatim [14]. Ref. [15] identified seven recurring failure points in the engineering of production RAG systems, illustrating that reliability and correctness remain difficult to guarantee in deployed pipelines. In our own earlier work [16], we developed an RAG-based authorization system that enforced access control without database dependencies. Nevertheless, because authorization decisions originated from the LLM's own reasoning stage, the system lacked a deterministic reasoning layer: no individual decision could be traced back to a specific, inspectable rule, and the framework offered no formal guarantee that identical inputs would always yield identical access outcomes.

LLM-based agentic frameworks have been proposed to extend what such systems can accomplish. LangGraph [17] models multi-agent workflows as stateful, cyclic graphs in which LLM-driven nodes make routing and decision choices at runtime; AutoGen [18] enables conversation-based coordination across multiple LLM agents. These frameworks substantially expand the operational scope of LLM-driven pipelines, yet their reasoning substrate remains probabilistic. When an LLM serves as the orchestrator, agent decisions emerge from generation steps rather than from formal plan execution against an explicit belief base. Identical inputs may yield different agent behaviors across runs, and no individual decision can be traced to a specific, inspectable rule. For applications requiring auditable outcomes, this is a fundamental architectural limitation rather than an engineering trade-off.

The substantial deep-learning literature addresses the same demand for dependable behavior from the opposite direction, by improving the learned component itself; it is complementary to the present work rather than an alternative to it. Representation-learning methods that couple spatial and channel attention with multi-scale graph interaction and deep transfer learning report strong cross-domain results in unsupervised rotating machinery fault diagnosis [19], and machine-vision-driven physics-informed neural networks embed governing equations into visually acquired data so that predictions about the monitored physical system remain physically admissible [20]. Dependability has also been pursued at the control layer rather than the perception layer, for instance, through fixed-time adaptive smooth control with relaxed performance constraints for a vehicle subject to aerodynamic interferences and actuator faults [21], where the guarantee is obtained analytically from the closed-loop dynamics rather than statistically from held-out data. These approaches improve what a learned or controlled component computes; the architecture developed here takes the learned component as given and constrains where its output is permitted to influence a decision. The two directions compose: a more accurate extractor raises the belief quality entering the plan layer, which Section 6.7 identifies as the binding constraint on end-to-end correctness, without changing which component holds decision authority.

The Belief–Desire–Intention architecture [2] offers a complementary design philosophy. BDI agents maintain explicit belief bases, pursue declarative goals, and select plans through deterministic deliberation cycles: given the same belief base and plan library, a BDI agent always produces the same decision, and every decision is traceable to specific beliefs and plans. JASON [6] implements this model in the AgentSpeak language with formal semantics; JADE [7] provides the FIPA-compliant distributed runtime on which JASON agents operate. These platforms have demonstrated effectiveness in structured domains such as robotics, simulation, and process automation. However, classical BDI systems assume that beliefs arrive in symbolic, structured form—a prerequisite that fails when knowledge resides in unstructured documents, natural language policies, or dynamically evolving

knowledge bases. This limits BDI adoption to pre-structured environments, far from the unstructured document corpora that dominate modern knowledge-intensive applications.

Recent work has begun to bridge this gap by connecting BDI reasoning with language-based environments. Ref. [22] deployed BDI agents in natural language environments, using language-model-based natural language inference to match a natural-language belief base against plan contexts during plan selection. Ref. [23] presented ChatBDI, which integrates LLMs as a communication layer for JASON agents, enabling natural language interaction while preserving BDI reasoning semantics. Both contributions establish the viability of treating LLMs as peripheral tools for BDI agents rather than as the reasoning substrate itself. Our own prior work [8] took this a step further, demonstrating that BDI agents can govern a multimodal RAG pipeline for a concrete access control use case on JASON/JADE infrastructure, with parallel text and visual extraction pipelines feeding structured beliefs into agent deliberation. That work established the practical feasibility of BDI + RAG integration for a specific instantiation, which the present paper generalizes into the domain-independent architecture developed in Section 4.

Taken together, these studies demonstrate either flexible access to unstructured information or explicit BDI deliberation, and several combine elements of both. The narrower question addressed here is how to make retrieval and extraction update beliefs while keeping every domain judgment and the fusion decision in inspectable plans, and then to compare the resulting orchestration trace with LLM-agent baselines. Sections 4–6 specify and evaluate that arrangement.

4. Hybrid BDI + RAG Architecture

The architecture is organized around one implementation constraint: a probabilistic service may supply evidence, but it may not select an authorization outcome or the next coordination step. A query is first handled by a retrieval agent. Domain reasoning agents then request the structured facts needed by their plans, evaluate those facts, and return judgments. The orchestrator collects the judgments and sends them to a designated fusion reasoning agent. Answer generation begins only after the fusion result has been returned.

This organization does not prevent a model from affecting the evidence available to a plan. It prevents the model response from being executed as a plan or treated as the authorization rule itself. The distinction is important when interpreting the guarantees in Section 6.

4.1. Three Layers of Interaction

The architecture is organized into three interaction layers that correspond to three distinct concerns, as illustrated in Figure 1.

The *BDI layer* is where reasoning happens. It contains the orchestrator agent, one or more specialized reasoning agents, a retrieval agent, and a generation agent—all implemented as JASON BDI agents running on JADE infrastructure. Agents communicate exclusively through FIPA-ACL speech acts, which carry not just data but communicative intent: an *achieve* performative signals a goal to be pursued, a *tell* performative adds a belief to the recipient's belief base, and a *request* performative asks for a specific action. This semantically-rich communication is what makes the coordination auditable: every message is an inspectable record of what one agent asked or told another.

The *service layer* sits below the BDI layer and provides LLM inference, embedding computation, and domain-specific knowledge querying as stateless services accessed through MCP JSON-RPC calls. Agents invoke these services through Java internal actions that convert AgentSpeak terms into JSON-RPC calls and return structured results. Critically, this layer is unidirectional from the reasoning perspective: services compute and return

facts; they do not initiate actions, hold state, or make decisions. An LLM in this layer is a function, not an agent.

The *knowledge layer* holds the queryable knowledge sources that services access. The Content KB is populated offline as a one-time construction step: document embeddings are computed and stored to support semantic retrieval at inference time. The knowledge source behind the MCP Knowledge Service is deployment-defined—it may be a pre-indexed vector store, a relational database, an external API, or a domain ontology, depending on what structured facts the reasoning agents require. In all cases, the architecture favors separating knowledge construction from online querying: whatever preparation a knowledge source requires is performed once, so that agents query stable, pre-computed entries at inference time rather than re-running extraction or indexing pipelines per query.

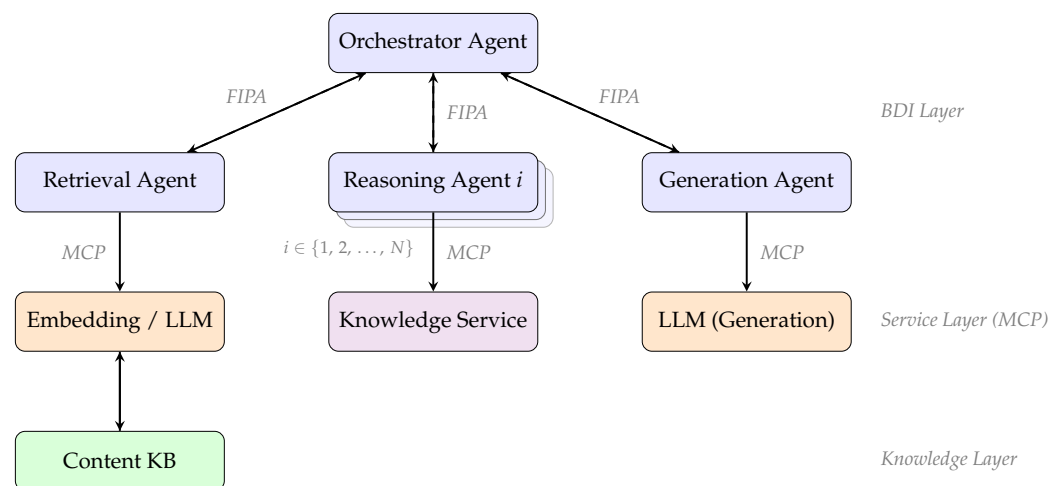


Figure 1. Conceptual architecture of the hybrid BDI + RAG system. BDI agents coordinate through FIPA-ACL speech acts; LLMs are accessed as stateless services through MCP. The Content KB is populated offline and queried at inference time. The stacked Reasoning Agent boxes indicate that N domain-specific agents ($i \in \{1, 2, \dots, N\}$) can be registered and operate in parallel, each contributing an independent domain-scoped judgment to the Orchestrator Agent. The MCP Knowledge Service is deployment-defined and may be realized as a vector store, relational database, external API, or ontology.

4.2. Four Canonical Agent Roles

The architecture defines four agent roles whose responsibilities are fixed regardless of the domain. What varies across deployments is the number of reasoning agents instantiated and the content of their plans.

The *orchestrator agent* is the system's conductor. It receives user queries, decomposes them into subgoals, and coordinates the execution of other agents by posting achievement goals via FIPA. As results arrive from specialized agents in the form of tell messages, the orchestrator accumulates beliefs and triggers subsequent stages. Importantly, the orchestrator contains no domain-specific logic: it knows how to coordinate agents, not how to apply domain-specific reasoning logic or evaluate retrieval quality. This separation ensures that adding a new reasoning agent requires no changes to the orchestrator. This principle extends to fusion: the orchestrator does not evaluate or combine the judgments of reasoning agents itself. It forwards them to a designated fusion reasoning agent and waits for the final decision to be delivered back before proceeding to the generation step.

The *retrieval agent* converts user queries into embedding vectors via MCP, queries the content knowledge base for semantically similar document chunks using cosine similarity, and returns ranked results to the orchestrator. It is deliberately modality-agnostic: it

operates over whatever content was indexed during knowledge base construction and does not know what those documents contain.

The *reasoning agent* role is where domain-specific intelligence lives. A deployment may instantiate any number of reasoning agents, each responsible for a different knowledge domain, reasoning dimension, or content modality. Each agent maintains a belief base relevant to its domain, queries the corresponding knowledge source via MCP when called upon, applies its AgentSpeak plans to derive a domain-specific judgment, and reports that judgment to the orchestrator. The decision logic is entirely in the plans—the Knowledge Service call via MCP returns structured facts, and the plan determines what those facts mean for the current decision context. New reasoning agents can be introduced simply by writing new plans and registering the agent’s capabilities with the orchestrator through standard FIPA agent description protocols; existing agents are unaffected. One reasoning agent in each deployment serves additionally as the fusion agent, receiving the collected judgments of all other reasoning agents from the orchestrator and delivering a single consolidated decision back—ensuring that every decision in the system, including the final one, originates from plan-based reasoning rather than from orchestration logic.

The *generation agent* operates at the end of the pipeline. It receives a set of content chunks that the fusion reasoning agent has determined to include, with that decision delivered to the orchestrator via `tell` and forwarded to the generation agent as part of its activation goal. It invokes an LLM via MCP to compose a natural language response grounded in those chunks and returns the response to the user. If the approved chunk set is empty, the generation agent returns a structured response indicating that no content is available, without invoking the LLM at all. This is a deliberate design choice: by preventing LLM invocation when the reasoning layer produces an empty result set, the architecture eliminates the risk that the generation stage might inadvertently surface information from its training data rather than from the knowledge base.

4.3. The Interaction Protocol

The coordination between these agents follows a fixed protocol expressed entirely in AgentSpeak plans, as mapped out in the abstract sequence diagram in Figure 2. The protocol does not change across deployments; only the plans inside the reasoning agents change.

When a user query arrives, the orchestrator posts it as an achievement goal. The retrieval agent responds by fetching top-*k* semantically similar chunks from the content knowledge base and reporting them to the orchestrator via `tell`. The orchestrator then activates all registered reasoning agents in parallel, sending each a request that includes the retrieved chunks and the user’s context. Each reasoning agent independently queries its knowledge source via MCP, applies its AgentSpeak plans, and reports a per-document judgment back to the orchestrator via `tell`.

When all reasoning agents have reported their domain-scoped judgments, the orchestrator forwards them to a designated fusion reasoning agent via an `achieve` goal. The fusion reasoning agent applies a configurable rule expressed in its own AgentSpeak plans—logical AND requiring all agents to agree, logical OR requiring at least one agent to affirm a positive decision, or any custom fusion logic—and delivers the final decision to the orchestrator via `tell`. Only after receiving this decision does the orchestrator activate the generation agent. At no point does the orchestrator evaluate or interpret the judgments itself: every decision, including fusion, originates from a reasoning agent and is delivered to the orchestrator before any subsequent step is triggered.

The plan and message structure makes the coordination path interpretable: an audit can inspect which beliefs triggered which plans and which FIPA messages were exchanged.

A production deployment should nevertheless retain timestamped service outputs, message identifiers, and plan identifiers so that the concrete execution can be reconstructed reliably.

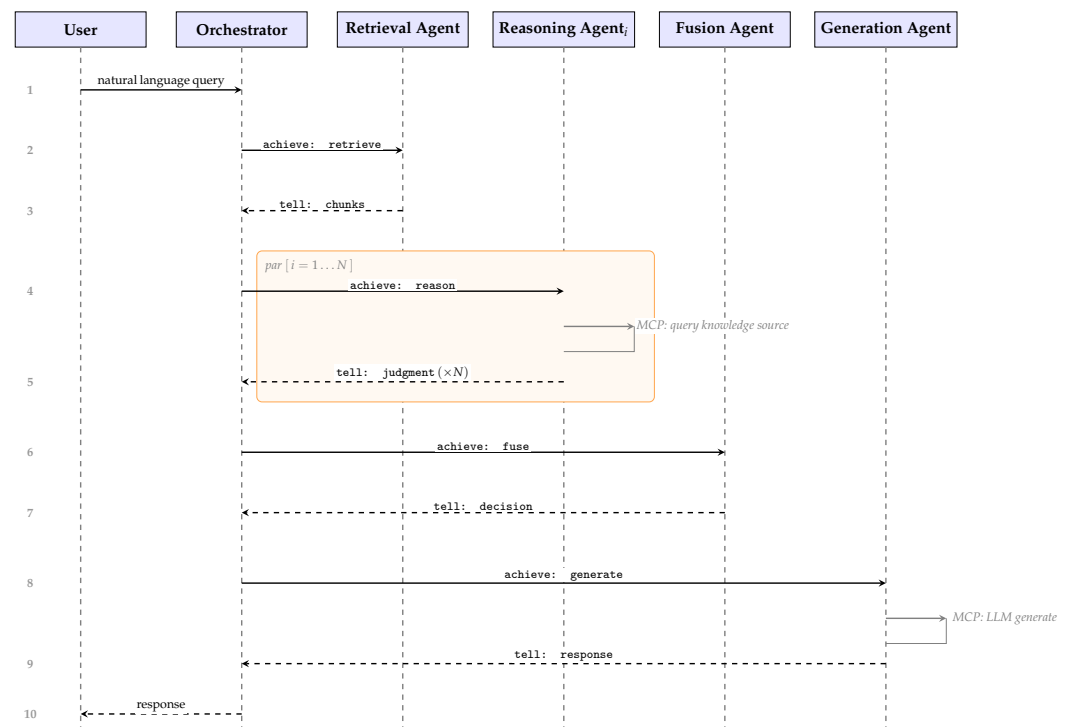


Figure 2. Sequence diagram of the hybrid BDI + RAG interaction protocol. Solid arrows represent *achieve* goals (delegation from orchestrator to specialized agents); dashed arrows represent *tell* messages (belief updates and judgment reports). The orange parallel region (Steps 4–5) indicates that all N reasoning agents are activated simultaneously and report independently. The orchestrator forwards all judgments to the Fusion Agent (Step 6) and waits for the consolidated decision (Step 7) before activating the Generation Agent, without interpreting any judgment directly.

5. Use Case: Multimodal Access Control for a Multi-Institutional Document Repository

Section 4 presented the hybrid BDI + RAG architecture as a general framework: four canonical agent roles, three interaction layers, and a fixed coordination protocol whose internals vary only in the plans that reasoning agents carry. This section demonstrates what that framework looks like when it is instantiated for a concrete, non-trivial problem domain. The domain is multimodal access control over a repository of research posters spanning multiple institutions—a setting where authorization-relevant information appears in text, in visual elements, or in both simultaneously, and where no single modality is sufficient to make a reliable decision.

The instantiation is drawn from our prior work [8], in which we built a BDI agent-based access control system on JASON/JADE infrastructure and demonstrated it across a range of multimodal authorization scenarios. Here, we revisit that system through the lens of the general architecture: we show precisely how each canonical role from Section 4 maps to a concrete agent in the use case, how the four-agent skeleton expands to six agents by instantiating three reasoning agents for this domain, and how the fixed interaction protocol from Section 4.3 plays out across six stages of the pipeline. The goal is not merely to describe the system, but to demonstrate that the general architecture is not abstract—every component, every protocol step, and every design decision in the use case traces back to a principle established in Section 4. All experiments reported in Section 6 were designed and

executed for the present paper under the configuration stated in Section 6.1; no numerical result is carried over from the conference paper.

5.1. Dataset

The evaluation uses a collection of 167 research posters from a Flanders Make conference held in 2025. Flanders Make is a strategic research center for the manufacturing industry in Belgium. The dataset was made available directly by Flanders Make and covers research contributions from all six institutions. As shown in Figure 3, the distribution across institutions is uneven—KU Leuven contributes 86 posters, Flanders Make 30, University of Ghent 22, Vrije Universiteit Brussel 13, University of Antwerp 11, and University of Hasselt 5—a distribution that mirrors real enterprise settings where different organizations produce different volumes of documentation.

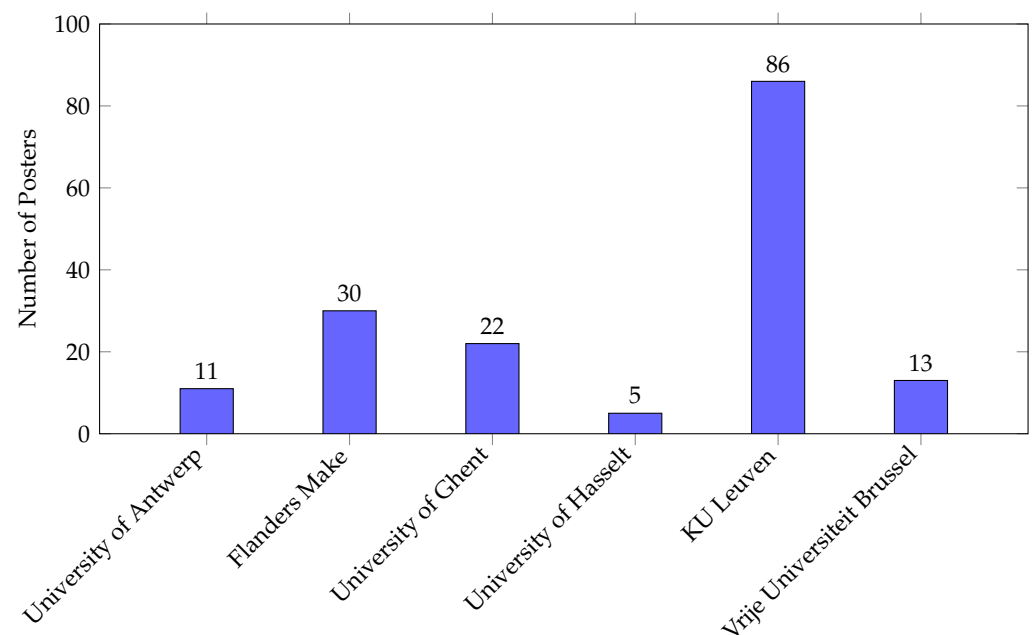


Figure 3. Distribution of research posters across the six Belgian institutions in the Flanders Make dataset. Adapted from our prior work [8].

Because some of the raw posters carry no native access control metadata, a systematic annotation pipeline was applied to introduce the security semantics required for evaluation. Each poster was enriched along two orthogonal dimensions: a *security level classification* (one of *public*, *private*, *confidential*, or *copyright*) and an *organizational affiliation identifier* indicating document ownership. Both dimensions were embedded using a deliberate hybrid strategy—metadata could appear in textual content, visual elements, or both—reflecting the heterogeneous nature of real enterprise documents where authorization-relevant information is rarely confined to a single modality.

For the security level dimension, three indicator strategies were applied across the poster collection. A first subset of posters carries security labels exclusively within document text, with no visual badges present. A second subset conveys security level solely through visual badge overlays, with no explicit textual classification. A third category combines both channels, enabling cross-modal consistency checks and testing the system’s ability to reconcile information arriving through different representational formats. Organizational affiliation follows the same pattern independently: some posters carry the owning institution in text only, some carry it only as an institutional logo overlay with no written reference, and some carry it in both channels. Because the two dimensions were assigned

independently, one group of posters carries all of its authorization metadata in text and another carries all of it in images, so neither modality alone is sufficient to authorize the whole corpus.

The annotation pipeline was implemented in Python 3.10.11 using PyMuPDF 1.25.1 for PDF manipulation and text injection, and OpenCV 4.10.0.84 for logo resizing and badge compositing. The pipeline is fully deterministic: given identical poster and annotation parameters, output is identical across runs, ensuring that any observed variation in system behavior reflects the access control framework rather than inconsistencies in the dataset. Each poster's assignment—which dimension is carried by which channel—is recorded in the ground-truth table that the evaluation harness scores against. That table is published with the code in anonymized form: each poster is identified by a sequential label and the original filename and repository path are removed. The annotator itself is not released; this subsection is its specification, and the synthetic benchmark generator applies the same annotation semantics to a corpus that can be redistributed, so the protocol can be exercised without the proprietary posters. The two resulting knowledge bases are not released. Their entries reproduce content extracted from the posters and are covered by the same confidentiality terms as the corpus itself.

The released ground-truth table is the authoritative record of that assignment. It carries one row per poster—identified by its anonymized label rather than by filename—with the institution and its normalized key, the security level, the annotation group, four binary indicators recording which of the two dimensions is present in which channel (textual security label, visual security badge, textual organization statement, institutional logo), and the per-document outcome of each knowledge-base construction pass. The annotation-distribution and joint-distribution tables reported later in this section are computed from that file and reproduce it exactly.

Table 1 states the channel assignment itself, which is what the annotation protocol actually fixes. The assignment is complete within a group: every poster in a group carries exactly the signals its row marks and none of the others, so the design is a partition rather than a distribution.

Table 1. Annotation design by group: which of the four authorization signals each poster carries. A check mark means the signal is present for every poster in the group and a dash means that it is present for none; the assignment admits no exceptions. The two dimensions were assigned independently, which is why group B2 exists—its security level is visual only while its organization is stated in both channels.

Group	Posters	Security Level		Organization	
		Text Label	Visual Badge	Text Statement	Logo Overlay
A	67	✓	—	✓	—
B1	30	—	✓	—	✓
B2	12	—	✓	✓	✓
C	58	✓	✓	✓	✓
Total	167	125	100	137	100

Three consequences of this design govern everything the fusion policy can do, and they are properties of the corpus rather than of any model.

Neither dimension is ever absent from both channels. Every poster's security level is recoverable from at least one channel, and so is its organization. A document is therefore never undecidable for want of annotation; where a decision fails, the cause is extraction, not the corpus.

The two dimensions have different coverage. The security level appears in text for 125 posters, as a badge for 100, and in both channels for only 58. The organization appears in text for 137, as a logo for 100, and in both for 70. The asymmetry follows from the independent assignment of the two dimensions and means that “the modalities agree” is a claim that can be tested on 58 posters for the security level and on 70 for the organization, not on the whole corpus.

A conjunctive rule is bounded by the corpus, not by accuracy. A channel can decide a document when it carries either of the two signals for it, so the textual channel can decide 137 posters and the visual channel 100, and both can decide only the 70 in groups B2 and C. The remaining 97 posters—the 67 in group A and the 30 in group B1, or 58.1% of the corpus—are described by exactly one modality, and the opposite knowledge base is empty for every one of them. A fusion rule requiring both modalities to grant would deny all 97 regardless of the requesting user’s entitlement. This is the concrete reason the Decision Engine Agent applies a disjunctive rule, and Section 6.4 measures what the conjunctive alternative costs on the subset where it is defined.

The ground-truth table also records, per poster, whether each knowledge-base construction pass recovered that poster’s annotation and, when it did not, which of four failure modes occurred: the security level was not extracted, the security level was extracted incorrectly, the organization was not extracted, or the organization was extracted incorrectly. At the document level the textual pass reproduced 115 of the 125 posters it covers and the visual pass 78 of 100. Group C, where both channels are populated, is the only group in which the two knowledge bases can disagree, and they do so for 20 of its 58 posters: 16 where the textual entry is right and the visual entry wrong, 3 where the reverse holds, and 1 where both are wrong. Those 20 posters are the corpus’s real conflicting-modality cases, and Section 5.6 draws its conflict scenario from them. Section 6.3 reports the same extraction pass in a different unit, per field rather than per poster: a poster counts as recovered here only when every field it carries is correct, while there each field is scored on its own, so a poster with two wrong fields is one failure here and two there.

The two authorization knowledge bases are the observable output of that pipeline, and their entry format is what the reasoning agents ultimately consume. Each entry is keyed by poster filename and states the two dimensions in a fixed form: a line declaring the file’s security level as one of *public*, *private*, *confidential* or *copyright*, and a line declaring the organization for which permission is defined, or an explicit *not defined* marker when that channel carries no such signal for that poster. The Text Authorization Knowledge Base is produced from the textual channel and the Image Authorization Knowledge Base from the visual channel, independently and by different models, so that a poster annotated in one channel only yields an entry in one knowledge base and an explicit absence in the other. That explicit absence, rather than a silent gap, is what allows the Decision Engine Agent to distinguish “this modality denies access” from “this modality has nothing to say”, a distinction on which the fusion comparison of Section 6.4 depends.

Two properties of this construction bound what the evaluation can show, and we state them here rather than in the results. First, the security semantics are introduced rather than native: the labels are unambiguous, present exactly where the protocol places them, mutually consistent within a channel, and never obsolete or adversarially manipulated. Real enterprise repositories satisfy none of those conditions reliably, and Section 6.10 treats the gap as a limitation on external validity rather than as a detail of the pipeline. Second, because the badges and logos are composited from a fixed visual vocabulary at consistent positions, the visual extraction accuracies reported in Section 6.3 are an upper bound on what the same extractor would achieve on naturally occurring visual markings. Both points are the reason the released benchmark includes a synthetic variant that reproduces the full

pipeline without the proprietary posters, so that the annotation protocol can be varied by others independently of the corpus. Table 2 reports the resulting annotation distribution. Combining the security level and organization channels yields four groups. Group A carries both the security label and the organization in text only, with no badge and no logo. Group B1 is its mirror image: the security level appears only as a visual badge and the organization only as an institutional logo, so these posters carry no authorization metadata in text at all. Group B2 places the security level in a badge while the organization appears in both text and logo. Group C places the security level in both channels and the organization in both channels. Table 3 reports the joint distribution of institution and security level.

Table 2. Distribution of the 167 annotated posters across security levels and annotation groups. Group A: security level and organization both in text only. Group B1: security level in a visual badge only and organization in an institutional logo only, so no authorization metadata is present in text. Group B2: security level in a badge only, organization in both text and logo. Group C: both dimensions present in both channels.

Security Level	A (Text Only)	B1 (Image Only)	B2 (Badge + Text Org)	C (Both)	Total
Public	9	6	0	7	22
Private	23	11	3	20	57
Confidential	13	6	3	11	33
Copyright	22	7	6	20	55
Total	67	30	12	58	167

Table 3. Joint distribution of institution and security level across the 167 annotated posters.

Institution	Public	Private	Confidential	Copyright	Total
KU Leuven	9	27	25	25	86
Flanders Make	6	12	0	12	30
University of Ghent	4	9	0	9	22
Vrije Universiteit Brussel	1	5	2	5	13
University of Antwerp	2	3	3	3	11
University of Hasselt	0	1	3	1	5
Total	22	57	33	55	167

5.2. Knowledge Base Construction

Three specialized knowledge bases support the use case, each serving a distinct role in the pipeline. Their construction is an offline, one-time process; at inference time, agents query pre-computed entries rather than re-running extraction, eliminating per-query overhead and ensuring that retrieval latency is independent of extraction complexity. This is a direct application of the offline construction principle stated in Section 4.2.

Poster Content Knowledge Base. This knowledge base supports the Chunk Retrieval Agent’s semantic search function. Document text is segmented into chunks and encoded as 1024-dimensional dense vectors using the `mxbai-embed-large` embedding model. At inference time, the Chunk Retrieval Agent embeds the user’s query using the same model and retrieves the top-*k* most semantically similar chunks by cosine similarity. This knowledge base carries no authorization metadata; its sole function is relevance-based retrieval.

Text Authorization Knowledge Base. This knowledge base is populated by running each poster through the `qwen2.5:72b` language model offline, which parses document text to identify institutional affiliations and security classification labels. Each entry records the document identifier, the detected organization name, and the inferred security level

derived from textual content. At inference time, the Text Access Control Agent queries this knowledge base via MCP to evaluate whether each retrieved chunk satisfies the textual access conditions for the requesting user.

Image Authorization Knowledge Base. This knowledge base mirrors the structure of the text authorization knowledge base but is populated exclusively from visual content. Offline extraction was performed using the qwen2.5-v1:72b vision-language model, which analyzed poster images to recognize institutional logos and interpret visual security badge overlays. The resulting entries associate each document with an organization and security level derived purely from visual evidence. At inference time, the Image Access Control Agent queries this knowledge base independently and in parallel with the Text Access Control Agent, providing a modality-separate authorization signal.

The three knowledge bases embody the offline construction principle established in Section 4.2: computationally expensive extraction—whether semantic embedding or multimodal LLM inference—is performed once at construction time, and the resulting structured entries are stored for deterministic lookup at query time. This ensures that the online reasoning path contains no probabilistic generation steps beyond the final answer synthesis.

5.3. Mapping the General Architecture to This Domain

The general architecture defines four canonical roles. When instantiated for multimodal access control, the orchestrator and retrieval roles map directly—one agent each—while the reasoning agent slot expands to three agents, each responsible for a distinct dimension of the access control problem. The generation role maps to a single agent. The result is six agents in total, whose relationships to the canonical roles are illustrated in Figure 4.

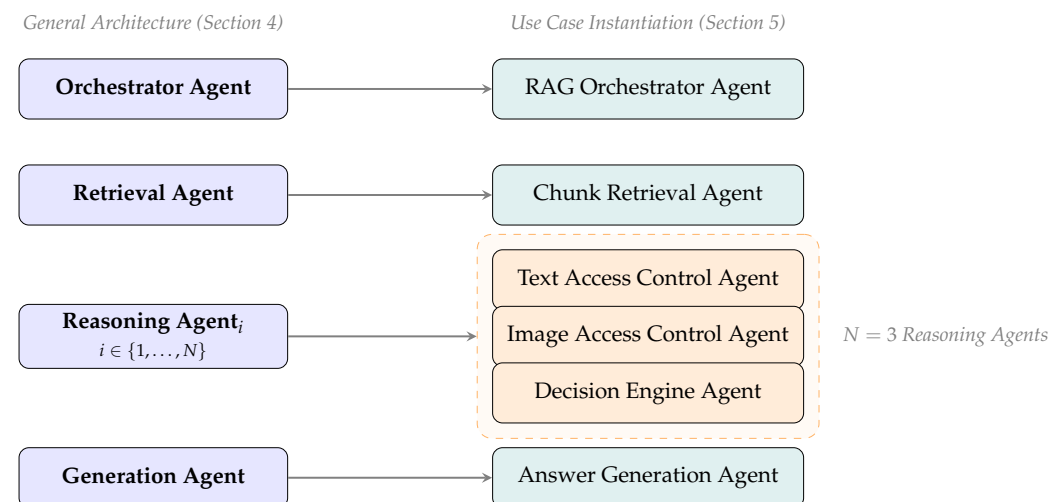


Figure 4. Mapping from the four canonical agent roles of the general architecture (Section 4) to the six concrete agents of the multimodal access control use case. The Reasoning Agent slot is instantiated here with $N = 3$: the Text Access Control Agent, the Image Access Control Agent, and the Decision Engine Agent, each responsible for a distinct dimension of the reasoning problem.

The mapping reveals an important design decision: the fusion step, which in the general architecture is assigned to a designated fusion reasoning agent, is here instantiated as the Decision Engine Agent—a third reasoning agent whose sole responsibility is to receive the collected judgments of the other two agents from the orchestrator and apply a domain-specific fusion rule. This delegation is not arbitrary. Fusing two independent, modality-specific signals is itself a non-trivial reasoning task: it must evaluate whether

text-derived and image-derived metadata agree, conflict, or are partially absent, and it must apply a well-defined fusion policy in a fully auditable way. By expressing this logic in an AgentSpeak plan within a dedicated reasoning agent rather than embedding it in the orchestrator, the architecture ensures that the fusion policy is inspectable, independently testable, and replaceable without touching coordination logic. The orchestrator remains the sole coordination hub throughout: it collects judgments from the two access control agents, forwards them to the Decision Engine Agent via an `achieve: fuse` goal, and receives the consolidated outcome via `tell: decision`.

The full interaction architecture of the use case—including all six agents, their FIPA communication links, their MCP service connections, and their associated knowledge bases—is shown in Figure 5.

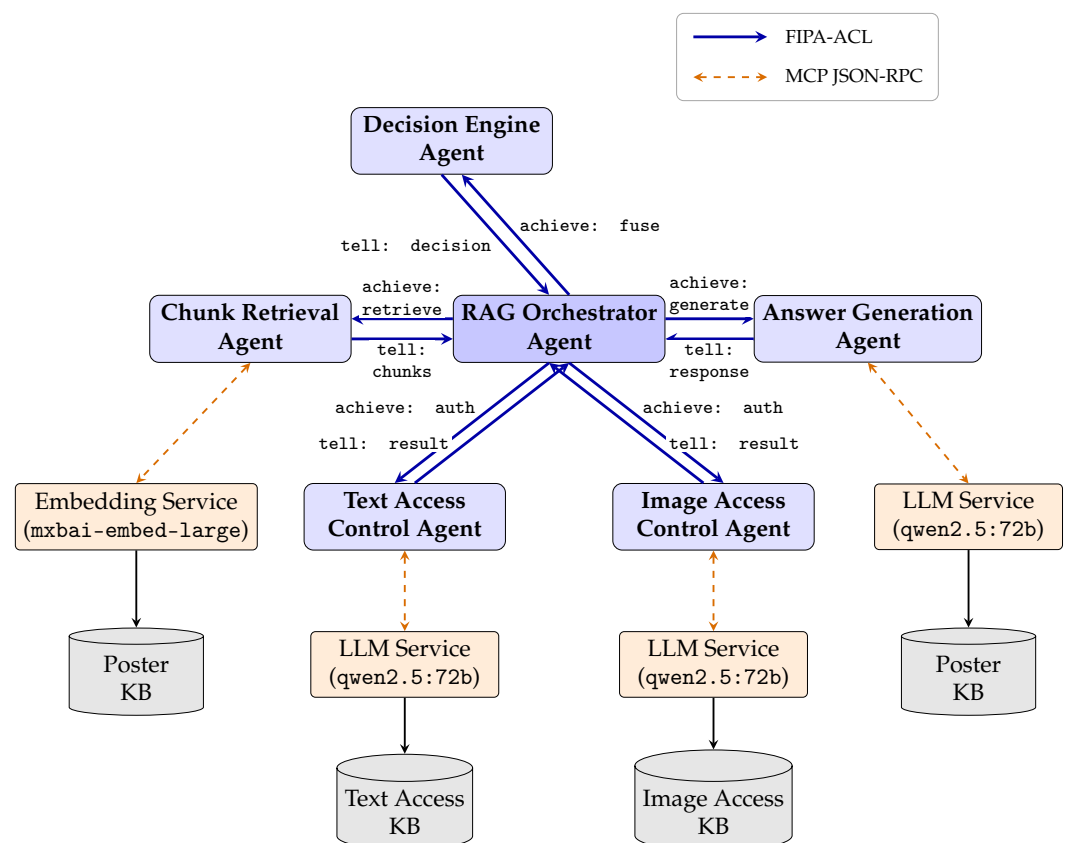


Figure 5. Full interaction architecture of the multimodal access-control use case, showing all six agents, their FIPA-ACL communication links, their MCP service connections, and their associated knowledge bases. The service boxes name the models of the deployed instantiation. In the evaluation of Section 6 every arm is served by qwen3.5:9b, and the plan-based arm shown here is re-run under that configuration in campaigns C1 and C2. Adapted from our prior work [8].

5.4. Reasoning Agent Instantiation

The three reasoning agents carry the domain-specific intelligence of this use case. Each is implemented as a JASON BDI agent running on JADE infrastructure, with decision logic expressed entirely in AgentSpeak plans. Their collective responsibility is to determine, for each retrieved document chunk, whether the requesting user is permitted to receive that content—using two independent authorization signals and a principled fusion rule.

Text Access Control Agent. This agent handles the textual dimension of access evaluation. When activated by the orchestrator via an `achieve: auth` goal, it receives the retrieved chunk identifiers and the user’s contextual information. It queries the Text Authorization Knowledge Base via MCP, retrieving the organization and security level associated with

each document. Its AgentSpeak plans then apply the authorization rules: a document classified as *public* is unconditionally accessible; a document classified as *private*, *confidential*, or *copyright* requires the user's organizational domain to match at least one approved organization in the knowledge base entry. The qwen2.5:72b LLM is invoked via MCP solely to parse natural language fields from the knowledge base into structured symbolic terms—the authorization judgment itself is produced by the plan, not the LLM. The agent reports its per-document judgment to the orchestrator via `tell: result`.

Image Access Control Agent. This agent operates in parallel with the Text Access Control Agent and is structurally identical in role, but draws its authorization signal exclusively from the Image Authorization Knowledge Base. Although that knowledge base was populated from poster visuals offline by the qwen2.5-vl:72b vision-language model, its entries are stored as structured textual records; at query time the agent therefore reads them exactly as the Text Access Control Agent reads its own knowledge base, invoking qwen2.5:72b via MCP only to parse the natural-language fields into symbolic terms, with no vision model on the online query path. Its plans apply the same authorization logic over organization and security level metadata—here derived from institutional logos and visual badge overlays rather than from document text. The parallel execution of these two agents reflects a key property of the general framework: reasoning agents are independent and do not share state. Each produces a judgment based solely on its own knowledge base, and neither agent is aware of the other's conclusions. The agent reports its per-document judgment to the orchestrator via `tell: result`.

Decision Engine Agent. This agent is activated by the orchestrator once both access control agents have reported their per-document judgments. The orchestrator collects these two sets of judgments and forwards them to the Decision Engine Agent packaged within an `achieve: fuse goal`—so the Decision Engine Agent never communicates directly with the access control agents, and the orchestrator remains the sole coordination hub. The Decision Engine Agent's plans then apply a logical disjunction rule over the forwarded judgments: a document is considered authorized if either the text-derived or image-derived signal grants access. This fusion policy is particularly suited to the multimodal annotation strategy of the dataset, where authorization metadata may be present in only one modality for a given poster; requiring both signals to agree would produce false denials in cases where one modality carries no metadata. The resulting fused decision for each chunk is reported back to the orchestrator via `tell: decision`, at which point the orchestrator holds a single, consolidated authorization outcome for every retrieved chunk and proceeds to activate the Answer Generation Agent.

5.5. Interaction Protocol in Action

The six-stage interaction protocol of this use case is a direct instantiation of the general protocol described in Section 4.3, with each abstract step populated by the concrete agents and knowledge bases introduced above, and visually conceptualized in the runtime sequence diagram in Figure 6.

A user submits a natural language query. The RAG Orchestrator Agent receives it and posts it as an achievement goal, decomposing the overall task into a sequenced set of subgoals.

In **Stage 1**, it activates the Chunk Retrieval Agent via `achieve: retrieve`. The Chunk Retrieval Agent embeds the query using `mxbai-embed-large` through an MCP call, queries the Poster Content Knowledge Base by cosine similarity, and returns the top five most semantically relevant chunks to the orchestrator via `tell: chunks`. These chunks may originate from a single poster or from multiple documents.

In **Stage 2**, the orchestrator performs a knowledge base filtering step: it loads entries from both the Text and Image Authorization Knowledge Bases and retains only those entries corresponding to the documents from which the retrieved chunks were drawn. This reduces the authorization scope from the full 167-poster repository to at most five documents per query, bounding the computational cost of the authorization stages that follow.

In **Stage 3** and **Stage 4**, the orchestrator activates the Text Access Control Agent and the Image Access Control Agent in parallel via `achieve: auth` goals, forwarding to each the relevant knowledge base entries and the user's contextual information. Each agent queries its respective authorization metadata via MCP and applies its plans independently, reporting per-document judgments to the orchestrator via `tell: result` upon completion.

In **Stage 5**, the orchestrator collects both sets of judgments and forwards them to the Decision Engine Agent via `achieve: fuse`. The Decision Engine Agent applies its disjunctive fusion plans and reports a single consolidated authorization outcome per chunk back to the orchestrator via `tell: decision`. The orchestrator now holds a complete, chunk-level access decision set.

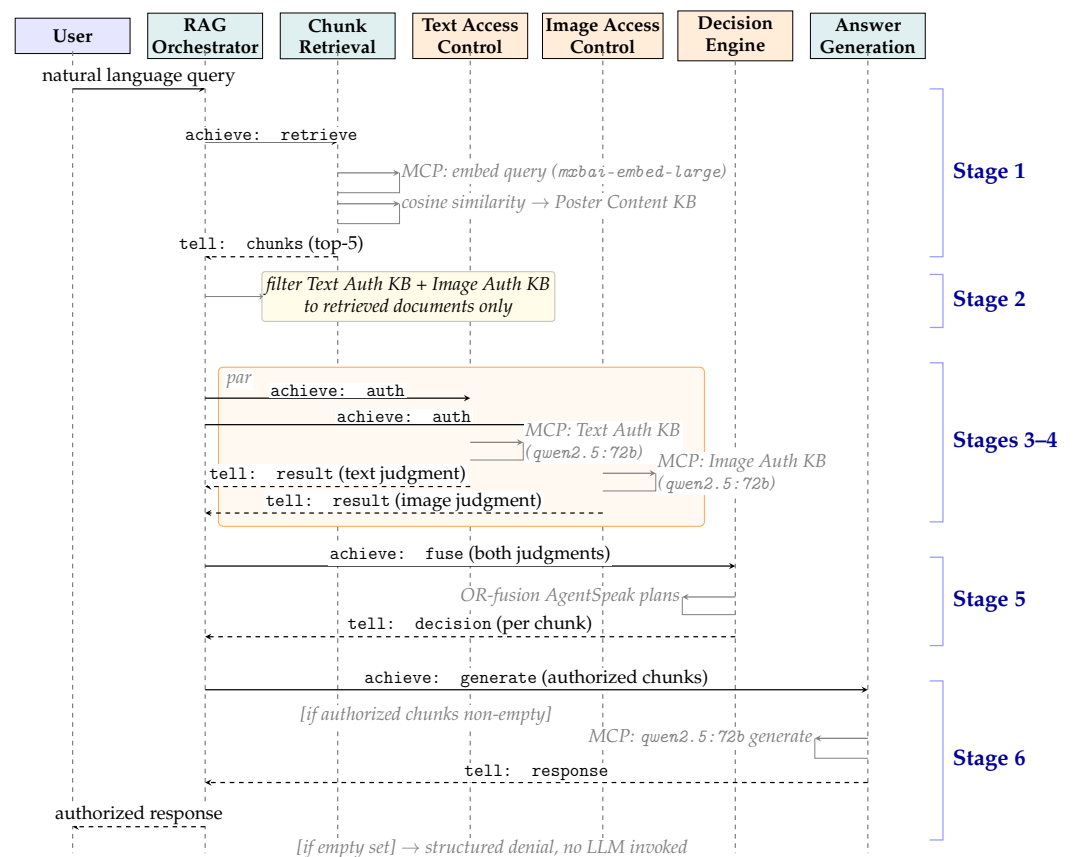


Figure 6. Sequence diagram of the six-stage interaction protocol for the multimodal access control use case. Solid arrows denote `achieve` goals, dashed arrows denote `tell` replies, gray self-loops with italic notes mark MCP service calls, and the blue brackets on the right delimit the six stages. Actor boxes follow the colour coding of Figure 4: reasoning agents in orange, the other agents in teal. Stage 2 is an internal orchestrator filtering step requiring no agent activation. Stages 3 and 4 execute in parallel within the orange region, with the Text and Image Access Control Agents operating independently over their respective knowledge bases via MCP. The Decision Engine Agent receives both judgment sets from the orchestrator and applies disjunctive fusion plans before returning a chunk-level decision. The Answer Generation Agent invokes the LLM only when the authorized chunk set is non-empty. The model named on the service calls is the one used by the deployed instantiation; the campaigns of Section 6 use `qwen3.5:9b` for every arm.

In **Stage 6**, the orchestrator activates the Answer Generation Agent via `achieve:generate`, forwarding only the chunks for which the Decision Engine Agent returned a positive decision. If this set is non-empty, the Answer Generation Agent invokes the `qwen2.5:72b` LLM via MCP and composes a natural language response grounded exclusively in the authorized content. If the set is empty, the agent returns a structured denial message without invoking the LLM, eliminating the risk of the generation stage surfacing information from model training data rather than from the knowledge base.

Every step of this protocol is recorded in the belief traces of the participating agents. A complete audit of any query—including which plans fired, which beliefs triggered them, what each agent concluded, and what FIPA messages were exchanged—can be reconstructed without any additional logging infrastructure.

5.6. Illustrative Scenarios

Four scenarios illustrate how the protocol handles conditions that are architecturally representative: cross-modal authorization, complete denial, partial authorization across multiple documents, and conflict resolution between modalities.

Scenario 1—Cross-Modal Authorization. A University of Antwerp researcher queries “*knowledge graph engineering workflows*”. The Chunk Retrieval Agent returns chunks from Document A. The Text Access Control Agent finds that the textual knowledge base records University of Antwerp as an approved organization for this document; the Image Access Control Agent finds that the visual knowledge base records a *private* security badge but no conflicting organization restriction. Both agents report positive judgments to the orchestrator, which forwards them to the Decision Engine Agent. The disjunction rule produces an authorized outcome, and the Answer Generation Agent composes a response from Document A’s chunks. This scenario illustrates the common case where the two modalities provide complementary rather than redundant information.

Scenario 2—Complete Multimodal Denial. A KU Leuven professor queries “*brushless motor exciter designs*”. All five retrieved chunks originate from Document D. The Text Access Control Agent finds that the textual knowledge base records a *private* security level and University of Ghent as the sole approved organization. The Image Access Control Agent finds the same information from the visual badge and institutional watermark. Both agents report negative judgments to the orchestrator. The Decision Engine Agent’s disjunction over two negative signals produces a denial outcome. The orchestrator forwards an empty chunk set to the Answer Generation Agent, which returns a structured denial message without invoking the LLM. No content from Document D reaches the user, and the denial decision is fully traceable to the specific plan rules and belief values that produced it.

Scenario 3—Partial Authorization Across Multiple Documents. A Flanders Make researcher queries “*collaborative research findings*”. The Chunk Retrieval Agent returns five chunks: three from Document E (security level *private*, approved organization Flanders Make), one from Document F (security level *public*), and one from Document G (security level *private*, approved organization Vrije Universiteit Brussel). The Text and Image Access Control Agents authorize the chunks from Documents E and F—Document E because the user’s domain matches, Document F because it is publicly accessible—and deny the chunk from Document G due to an organizational mismatch. The Decision Engine Agent produces four positive outcomes and one negative outcome. The Answer Generation Agent receives only the four authorized chunks and composes a response that naturally excludes any content specific to Document G. This scenario demonstrates that chunk-level granularity allows the system to serve partial results rather than issuing an all-or-nothing decision at the document level.

Scenario 4—Conflicting Modalities, Fusion Resolution. This scenario is taken from the corpus rather than constructed, so that the behavior it illustrates can be checked against the released ground truth, where the poster appears as poster9 (University of Antwerp, *private*). It belongs to group C: its security level and its organization are each stated both in text and visually. A University of Antwerp researcher queries it. The Text Access Control Agent reads a textual entry that records the document as *private* with University of Antwerp as the approved organization, and reports a positive judgment. The Image Access Control Agent reads a visual entry whose institutional logo was misidentified during knowledge-base construction, attributing the poster to a different institution, and therefore reports a negative judgment. The Decision Engine Agent receives one positive and one negative signal and applies the disjunction rule, producing an authorized outcome—which is the correct outcome, reached despite an incorrect belief in one modality.

This is the favorable direction of the trade-off, and it is the common one in this corpus: of the 20 group C posters whose two knowledge bases disagree, 16 have a correct textual entry and an incorrect visual one. The unfavorable direction also occurs, in three posters, and it is the case that motivates the policy comparison of Section 6.4: when the textual channel is the one that is wrong and denies, disjunction grants on the strength of a visual signal alone, and if a badge or logo could be manipulated rather than merely misread, disjunction would require an attacker to compromise only the weaker channel. The architectural point is that this is a policy parameter and not a model property: the rule is a single AgentSpeak plan, the conjunction alternative is a one-rule edit, and Section 6.4 states the threat model under which each is defensible and measures what the exchange costs in false authorizations and false denials.

6. Evaluation and Discussion

What is evaluated is the architecture of Section 4; the access-control instantiation of Section 5 is the harness in which it is exercised. The agents, the knowledge bases and the query set are held fixed, and the component that decides the next step is exchanged for an AutoGen assistant, a LangGraph router, or the plan library. All three arms are served by one locally served model, qwen3.5:9b, so that no arm runs on a stronger backend than another and the control layer is the only independent variable; the larger models that built the knowledge bases do their work offline, before any campaign starts.

The evaluation has two parts. The first is empirical and concerns the behavior that can be measured on running systems: whether a workflow reaches a valid schedule, whether it reaches the same schedule again, how sensitive that outcome is to the configuration under which it was obtained, and—separately—whether the authorization decision that the workflow produces is correct. The second part is architectural and concerns properties that are established by construction rather than by measurement: auditability, resistance to direct prompt injection at the reasoning layer, and extensibility.

Reproducibility and correctness are reported separately throughout, because a control layer can reproduce the same orchestration trace on every repetition while acting on incorrect beliefs. Section 6.1 specifies the baselines, the metrics and the campaign design; Sections 6.2–6.5 report orchestration behavior, decision quality, fusion policy and cost; and Sections 6.6–6.9 treat the architectural properties.

6.1. Evaluation Methodology

Classical BDI supplies an explicit plan-based reference point but normally assumes that beliefs are already structured. LangGraph [17] and AutoGen [18] supply relevant LLM-agent baselines because they can implement the same functional sequence of retrieval, specialist judgments, and fusion while allowing model outputs to influence workflow

control. The comparison asks where decision authority resides, whether the applied rule can be inspected, and whether repeated execution follows the same coordination trace.

6.1.1. Baseline Specification

A reproducibility comparison is only interpretable if the baselines are specified to the point where a reader can rebuild them. The two LLM-agent arms were therefore reconstructed so that the control layer is the single independent variable in the experiment.

Shared backend. The six agents of the hybrid system are exposed to both baselines as six tools over one identical backend. Both arms import the same service module—retrieval, the authorization prompt, fusion, and generation—carried over unchanged from the MCP server used by the hybrid system, and the same tool module, which defines the tool signatures, the tool descriptions, the shared system message, and the per-run session state. A difference between the two arms therefore cannot originate in worker output; it can only originate in which component decides the next step. In the hybrid arm that component is AgentSpeak plan selection on JASON; in the AutoGen arm it is an AssistantAgent choosing the next tool call; in the LangGraph arm it is a router node whose tool calls drive a conditional edge.

What the control layer is told. Each tool description states what its agent does and which other agents it requires—the direct analogue of an AgentSpeak plan’s context condition—and nothing more. The terminal agent additionally states that it is the last step. For example, the description of the fusion tool reads: *“Fuses the textual and visual authorization decisions into the final per-document decision. Requires text_access_control_agent and image_access_control_agent”*. The shared system message names the number of available agents, instructs the model to respect each agent’s stated prerequisites, and states that the workflow is complete once the final answer has been generated. The reasoning block of the router model is suppressed through the system message rather than through a transport-level flag, because the two frameworks expose different request-body escape hatches and the two arms must receive an identical prompt.

What the control layer is not told. The functional sequence that a correct run must produce,

```
orchestrator → chunk_retrieval → text_access_control  
→ image_access_control → decision_engine → answer_generation,
```

is never shown to a model and never enforced at run time. There is no ordering instruction in the system message, no fixed-order fallback, and no repair of a bad decision. A wrong order, a skipped tool, a duplicated tool or an extra call is the phenomenon under measurement, not an error to be handled. A call that arrives before its prerequisites still executes: it satisfies them itself and records what it had to fill in, so that a single campaign yields both the orchestration numbers and the decision-quality numbers instead of discarding every deviating run. Runs are abandoned only after a cap of 18 control rounds.

The query set. All campaigns are driven by the same set of 100 access-control queries, held fixed across every arm and every configuration so that a difference between two rows is never a difference in what was asked. Each entry pairs a natural-language request with a requesting user identified by organizational affiliation, and names the poster the request targets together with the authorization outcome the corpus annotation entails. The 100 queries address 97 distinct posters and are balanced on the decision: 50 of them should be allowed and 50 denied. Their targets are spread over the annotation groups of Section 5.1 in the proportions the corpus itself carries, 30 in group A, 25 in group B1, 10 in group B2 and 35 in group C, and over the four security levels, 14 public, 37 private, 25 confidential and 24 copyright. Requesting users are drawn from all six institutions of the corpus. Nine queries target posters the annotation marks as conflicting-modality cases, which at three

repeats gives the 27-run subset scored in Section 6.3.4. The per-query ground-truth table behind these counts is released in the GitHub repository whose address is given in the Data Availability Statement, with every poster identified by its anonymized label. The query strings are withheld, because each one paraphrases the title of the poster it targets and those titles are corpus content; the synthetic benchmark ships an equivalent query set that is runnable end to end.

Models, versions and determinism pins. The reference arms were driven by qwen3.5:9b served locally through Ollama, used both as the control-layer model and as the service model, so that a control layer can never run on a different backend from the services it drives. Embeddings were produced with mx-bai-embed-large. Decoding used temperature 0 and a fixed sampling seed (42 unless a campaign varies it), with the top-5 retrieval setting used throughout. Every model is served locally from a pinned local weight file, with no hosted or online inference endpoint in the loop. The campaigns ran under Python 3.13.5; the package versions at campaign time were autogen-agentchat 0.7.5, langgraph 1.2.11, langchain-openai 1.6.0, openai 3.3.1, ollama 0.6.2 and torch 2.13.0.

Two model configurations appear in this paper and they belong to two different stages, so we state the division explicitly. *Knowledge-base construction* is an offline, one-time step performed once over the corpus, described in Section 5.2 and evaluated in Section 6.3.2; it uses qwen2.5:72b for the textual channel and qwen2.5-v1:72b for the visual channel, as illustrated in Figures 5 and 6. The *online query path* exercised by the eight campaigns uses qwen3.5:9b throughout: for authorization parsing, for answer generation, and, in the baseline arms, for control. Fusion is not on that list. It is a rule rather than a model call—an AgentSpeak plan in the Decision Engine Agent of the plan-based arm, and the identical rule in the shared service module the baseline arms import—so no arm consults a model at the fusion step. The hybrid arm is re-run under that same configuration in campaigns C1 and C2: the deployed instantiation of Section 5, whose figures name qwen2.5:72b on its online service calls, is the demonstration system, while every arm measured in this section is served by qwen3.5:9b, so that the control-layer model and the service model are identical in every arm.

Availability. The service module, the tool module with the two complete description sets, and both control layers are published in the public GitHub repository whose address is given in the Data Availability Statement, with every document identifier replaced by its anonymized label.

6.1.2. Measured Quantities

Orchestration. The primary measure is the *valid-schedule rate*: the share of runs that called each agent exactly once and satisfied every stated prerequisite. The dependency graph is a partial order—the text and image access-control agents are independent of each other—so two linearizations are correct and both count as valid. The stricter *canonical-order rate* compares against the single sequence the hybrid system schedules; the gap between the two columns is runs that took the other valid linearization, which are not errors. Rates are computed with Wilson 95% confidence intervals over the 300 runs of the arm they describe. The tables report the point estimates, and the interval quoted in the text is the one on the primary measure at the reference configuration, [97.1%, 99.7%] in both LLM arms. Failures are decomposed into missing tools, extra calls, duplicated calls, order inversions, prerequisite violations, unknown tool names and framework errors, and the mean edit distance to the canonical sequence is reported alongside.

Two further columns separate reproducibility from correctness at the query level, and they measure different things. A query whose repeats are *all* invalid is being handled reproducibly wrong: the control layer is stable, and stably mistaken. A query that is valid

in some repeats and invalid in others—the *mixed* column—is the only case in which the *verdict* changes between repeats. The *identical across repeats* column is the stricter question and the one that speaks to repeatability of the trace: it counts queries whose three runs produced exactly the same sequence. A query can be valid three times over and still fall outside that column, because the arm took one valid linearization on one repeat and the other on the next; the control decision varied, and only the verdict did not. Runs in which a single assistant turn emitted several tool calls at once are reported separately, because such a turn fixes the order as an array rather than as successive control decisions and the ordering claim is correspondingly weaker for them.

Decision quality. Retrieval is scored with Recall@1, Recall@3, Recall@5 and MRR. Authorization is scored at the document level and, more strictly, at the level of every retrieved chunk, with accuracy, precision, recall, F_1 , the full confusion matrix, and—reported separately, because they are not equally consequential—the false-allow rate (FAR) and the false-deny rate (FRR). The positive class is *allow*, so FAR counts documents released to a user who was not entitled to them. Two variants are reported: *end-to-end*, in which a document that was never decided counts as denied, and *authorization only*, restricted to runs in which the target document was retrieved and decided. Every authorization metric is additionally broken down by modality group. Because a channel whose JSON output cannot be parsed denies every document and is indistinguishable from a correct deny, the number of unparsed authorization channels is reported as a validity check on the accuracy numbers; it is zero in every campaign reported here.

Extraction. Knowledge-base extraction is scored offline against the corpus annotation, per channel and per field, as correct, missed, wrong or hallucinated. A document is scored on a field only when its annotation actually carries that signal, so a poster with no visual badge is not counted against the image channel’s security level accuracy.

Cost. Mean, median and 95th-percentile end-to-end latency, and control-layer token consumption per run, are recorded for every arm.

6.1.3. Campaign Design

A single reproducibility percentage is a property of one configuration, not of a system. To separate the two, eight campaigns were run in which exactly one setting changes at a time from a common reference arm, as listed in Table 4. Each axis is a degree of freedom that the LLM-controlled arms possess and the plan-controlled arm does not: the presentation order of the tool declarations, the presence of a prerequisite sentence, the presence of the terminal-step sentence, the sampling seed, the length of the pipeline, and the fusion policy. None of these can move the hybrid arm’s schedule, which is fixed by its plan library.

Table 4. The eight single-knob campaigns. Every campaign differs from the reference arm C1 in exactly one setting (bold). The grid is run on the two LLM-controlled arms, AutoGen and LangGraph, at full scale: 100 queries $\times$ 3 repeats $\times$ 2 arms = 600 runs per campaign and 4800 baseline runs in total. The plan-based arm is not part of this grid. It is measured in C1 and C2 only, 300 runs each, for the reason given in Section 6.2.2. The *Scale* column therefore reports the two baseline arms.

ID	Campaign	Setting Changed from C1	Question It Answers	Scale
C1	core_flat	—(reference: 6 agents, flat, shuffled, seed 42, OR)	How good is the baseline in a favorable setting?	full (600)
C2	ext_flat	pipeline: 6 $\rightarrow$ 9 agents	Does a valid schedule survive a longer pipeline?	full (600)
C3	seed1337	seed: 42 $\rightarrow$ 1337	Does the sampling seed move the outcome?	full (600)
C4	seed7	seed: 42 $\rightarrow$ 7	Third point on the seed axis	full (600)
C5	abl_canonical	tool order: shuffled $\rightarrow$ canonical	Is the observed order a decision or a list artefact?	full (600)
C6	abl_noprereq	prereq sentence: stated $\rightarrow$ removed	What remains when dependencies are unstated?	full (600)
C7	abl_noterminal	terminal hint: stated $\rightarrow$ removed	What does one hint sentence carry?	full (600)
C8	fusion_and	fusion: OR $\rightarrow$ AND	How does policy move the FAR/FRR trade-off?	full (600)

Two run counts follow from this design and they should not be merged. The *baseline grid* is complete at full scale in every campaign: 100 queries $\times$ 3 repeats $\times$ 2 LLM-controlled arms = 600 runs per campaign, 300 runs per arm, and 4800 runs over the eight campaigns. The two arms are two orchestration frameworks and not two models: a single control-layer model, qwen3.5:9b, drives every arm in every campaign, and the plan-based arm has no control-layer model at all, as plan selection replaces it. That figure counts AutoGen and LangGraph executions only, and it is the size of the grid rather than a count of executions of the proposed system. The hybrid BDI + RAG arm is run in the two campaigns whose varied setting has a counterpart in a plan library, C1 with six agents and C2 with nine, for 300 runs each and 600 runs in total. The six remaining axes are a sampling seed, the presentation order of the tool declarations, a prerequisite sentence, a terminal sentence and a fusion policy applied downstream of the schedule; the first five have no counterpart in a plan library and the last one does not touch the schedule, which is why the arm has no row for C3–C8 in Section 6.2.2 and why it is not re-run there. The study therefore reports 5400 measured runs in all, 4800 of them in the baseline arms and 600 in the plan-based arm.

6.2. Orchestration Reproducibility and Its Sensitivity

The claim evaluated here is narrower than end-to-end determinism. If agents begin with the same beliefs, use the same plan libraries, and receive the same structured service outputs, JASON selects the same applicable plans and the orchestrator follows the same message protocol. Embedding, extraction, and natural-language generation remain probabilistic and are outside this claim. We therefore measure the reproducibility of orchestration traces rather than textual identity.

In an LLM-based agentic framework, by contrast, the orchestration layer is itself generative. When an LLM serves as the orchestrator in a LangGraph or AutoGen system, each step—deciding which tool to call, interpreting the tool’s output, selecting the next action—is sampled from a distribution, and the reasoning that connects inputs to outputs is never available in symbolic form. It exists implicitly in the sequence of generated tokens that guided the pipeline, not in a human-readable rule that can be examined, tested, or certified.

Classical BDI systems are deterministic at this level by design: given the same belief base and the same plan library, the same plans are selected. The hybrid architecture inherits that property—reasoning and orchestration live entirely in AgentSpeak plans on JASON, and the LLMs are invoked only as MCP services that receive structured inputs and return structured outputs—and extends it to a setting in which the beliefs are drawn from dynamically queried knowledge bases.

What the experiments below add is the empirical counterpart of that argument, measured against two LLM-controlled baselines across eight configurations.

6.2.1. The Reference Configuration

Table 5 reports campaign C1: 100 access-control queries drawn from the multimodal poster repository of Section 5, each issued three times against each arm.

Two things follow from this table, and they are not the same thing. The first is that under a favorable configuration—six agents, a flat topology, and every dependency stated in the tool descriptions—both LLM-controlled arms reach a valid schedule in essentially every run, and no query was valid in some repeats and invalid in others: the *verdict* does not change under repetition. The second is that a stable verdict is not an identical trace. The three repeats of a query produce exactly the same sequence for 52% of queries in the AutoGen arm and 86% in the LangGraph arm; for the remainder the arm takes one valid linearization of the partial order on one repeat and the other on the next, swapping the

two independent access-control agents. The 99.0% column therefore says that the runs are correct, not that they are identical, and the two readings should not be merged.

Table 5. Campaign C1 (*core_flat*), the reference configuration: 6 agents, flat topology, shuffled tool declaration order, prerequisite and terminal hints present, seed 42, OR fusion. The valid-schedule rate carries a Wilson 95% confidence interval of [97.1%, 99.7%] in both LLM arms. *Identical across repeats* counts queries whose three runs produced exactly the same sequence. Every row measures the orchestration trace and none of them measures what the run decided: the authorization outcome is reported in Section 6.3 and is not 100.0% for any arm, including the plan-based one.

Measure	AutoGen	LangGraph	Hybrid BDI + RAG
Runs (per arm)	300	300	300
Valid schedule (primary)	99.0%	99.0%	100.0%
Canonical order	76.7%	94.0%	100.0%
Queries identical across all three repeats	52%	86%	100%
Queries valid in some repeats but not others	0	0	0

Where each property comes from is what separates the arms. In the hybrid arm the schedule is settled before the run starts: the same plans are applicable to the same events, so the same sequence is produced. Its 100.0% valid, 100.0% canonical and 100% identical across repeats are measured over the same 300 runs as the other two columns, and they are three readings of one property rather than three independent ones—a single sequence occurs in the whole campaign. In the baseline arms the schedule is re-derived by the model on every request. It comes out correct under this configuration, but the canonical-order and identical-across-repeats columns (76.7%/94.0% and 52%/86%) show that the order is being chosen at run time rather than fixed. Those choices are not errors, as the dependency graph admits both orders. A single measurement at this operating point therefore cannot separate a schedule that holds by construction from one that holds under the conditions in which it was measured; only varying those conditions can, which is what Section 6.2.2 does.

6.2.2. Sensitivity of the Control Layer to Its Configuration

Table 6 places side by side the seven campaigns that vary an input of the control layer. C8 varies the fusion policy applied downstream of the schedule, so its result is a decision-quality result and is reported in Section 6.4.

Four findings follow from the table.

The sampling seed does not move the outcome (C3, C4). Changing the seed from 42 to 1337 and to 7, with every other setting held fixed, leaves the valid-schedule rate at 100.0%/100.0%. The canonical-order rate does move—from 76.7% to 70.7% and 75.0% for AutoGen—but the movement is confined to the choice between two equally valid linearizations. The verdict stability observed at the reference configuration is therefore not an artifact of the particular seed it was measured with.

Most of the observed ordering variation is positional bias, not deliberation (C5). When the six tools are presented in canonical order instead of a shuffled order, the canonical-order rate rises to 99.7% and the share of queries whose three repeats are all identical rises to 99%. In other words, the text/image swaps observed in C1 are largely a function of the order in which the tool declarations appear in the request, not of a judgment about the task. The canonical-order gap at C1 therefore measures the influence of the tool list's presentation on the schedule: the order the control layer produces is a derivative of how the request is laid out rather than of its deliberation about the workflow.

One hint sentence matters, the other does not (C6, C7). Removing the prerequisite clause from the tool descriptions leaves the valid-schedule rate nearly intact (99.0%/97.3%), but it is the one edit anywhere in the grid that makes a control decision itself vary: it produces

queries that are invalid in every repeat (1 for AutoGen, 2 for LangGraph) and, in the LangGraph arm, the only *mixed* query anywhere in the grid—1 query out of 100, valid in some repeats and invalid in others. A single clause of tool documentation, with no counterpart in the hybrid arm, is therefore load-bearing for both the correctness of the schedule and the stability of its verdict.

Removing the sentence “This is the last step.” from the terminal agent, by contrast, changes nothing measurable at full scale: the valid-schedule rate stays at 100.0%/100.0%, the canonical-order rate is unchanged at 76.7% for AutoGen, and the share of queries whose three repeats are identical moves from 52% to 54%—inside the noise of the 100-query set. Which sentence of the tool documentation is load-bearing cannot therefore be read off its content: the clause that states a dependency carries the schedule, the clause that states termination does not.

Table 6. Valid-schedule and canonical-order rates across the seven campaigns that vary an input of the control layer. Every row is one arm under one configuration; C1 is the reference and each other row differs from it in exactly one setting; C8 varies the fusion policy instead and is reported in Section 6.4. The plan-based arm is measured at C1 and C2, the two campaigns that have a counterpart in a plan library; C3–C7 vary a sampling seed or a sentence of tool documentation, neither of which exists in that arm, so it has no row there. In C2, 9 of its runs (3 queries, all repeats) authorized no document at all; the plan then stops at the decision agent and does not invoke answer generation, so that no model sees denied content, and such a run counts as valid when every other agent was called exactly once in dependency order. *Never valid* counts queries that were invalid in every repeat; *mixed* counts queries valid in some repeats and not others—the only direct evidence of a varying control decision. Every arm row is 300 runs (100 queries $\times$ 3 repeats), with qwen3.5:9b as the control layer in every arm. Canonical-order rates are set in bold: a valid schedule admits either of two linearizations, whereas the canonical order is the single sequence the dependency graph makes natural, so it is the column that tracks how often a run reproduces one and the same trace.

ID	Configuration	Arm	Runs	Valid Schedule	Canonical Order	Never Valid	Mixed
C1	reference	AutoGen	300	99.0%	76.7%	0	0
		LangGraph	300	99.0%	94.0%	0	0
		Hybrid BDI + RAG	300	100.0%	100.0%	0	0
C2	pipeline: 6 $\rightarrow$ 9 agents	AutoGen	300	32.0%	22.3%	68	0
		LangGraph	300	28.0%	20.7%	72	0
		Hybrid BDI + RAG	300	100.0%	100.0%	0	0
C3	seed: 42 $\rightarrow$ 1337	AutoGen	300	100.0%	70.7%	0	0
		LangGraph	300	100.0%	94.7%	0	0
C4	seed: 42 $\rightarrow$ 7	AutoGen	300	100.0%	75.0%	0	0
		LangGraph	300	100.0%	93.3%	0	0
C5	tool order: shuffled $\rightarrow$ canonical	AutoGen	300	100.0%	99.7%	0	0
		LangGraph	300	100.0%	99.7%	0	0
C6	prereq sentence removed	AutoGen	300	99.0%	79.3%	1	0
		LangGraph	300	97.3%	92.3%	2	1
C7	terminal hint removed	AutoGen	300	100.0%	76.7%	0	0
		LangGraph	300	100.0%	94.3%	0	0

Pipeline length breaks the schedule outright (C2). This is the decisive row. Extending the pipeline from six agents to nine—adding query rewriting, knowledge-base parsing and a backend health check, each with stated prerequisites and none requiring new reasoning—drops the valid-schedule rate from 99.0% to 32.0% (AutoGen) and 28.0% (LangGraph). Of the 100 queries, 68 and 72 respectively, were invalid in *every* repeat, and 201 runs in the AutoGen arm and 213 in the LangGraph arm contain at least one prerequisite violation. Nothing in this campaign requires the control layer to reason about anything new: the

three added agents declare their dependencies in exactly the form the six original ones do. What changes is only the number of steps that have to be sequenced correctly in one run.

The plan-based arm was run through the same two campaigns, and its rows in Table 6 are measurements rather than assertions: 300 runs at six agents and 300 at nine, same corpus, same query set, same qwen3.5:9b services, seed 42. Both give 100.0% valid schedules and 100.0% canonical order, with every query producing an identical sequence in all three repeats and no query mixed. Only one sequence occurs in the six-agent campaign and one in the nine-agent campaign.

The reason is mechanical rather than statistical. Extending the arm to nine agents adds three plans and three message registrations; each plan carries a triggering event and a context condition that encode the same dependency the tool description states, and JASON's deliberation cycle selects, at every event, the applicable plan whose context condition holds against the current belief base. That selection is local: it is settled by the event and the beliefs it is evaluated against, not by how many agents the pipeline contains or how many of them have already executed, so the deliberation step does not grow with the pipeline the way a control prompt grows with its transcript. The nine-agent schedule is established once, when the plans are written and checked, which is why 100.0% is the same figure at six agents as at nine while the LLM-controlled arms fall to 32.0% and 28.0%: the cost of the extension is paid at design time instead of being re-derived on every request.

One detail of the nine-agent campaign is worth stating, because it is a property of the design rather than of the schedule. In 9 runs—3 queries, all three repeats each—no retrieved document was authorized. The decision agent then reports the denial and does not activate the generation agent, so no language model is invoked on content the policy has just refused. Those runs call every other agent exactly once in dependency order and stop where the plan says to stop; they are counted as valid schedules on that basis, and the rule is applied to every arm.

C1 and C2 are also the only two campaigns this arm can be run in, which is why Table 6 gives it no row elsewhere. The seed campaigns C3 and C4 change the sampling seed: that changes the tokens a service call produces, not which plan is applicable, because applicability is decided by a triggering event and a context condition evaluated against the belief base, and the dispatch order is fixed in the coordinator's plan. A different seed can change the content of a decision—a field parsed differently—but not the schedule. C5, C6 and C7 permute the tool declarations, delete the prerequisite sentence and delete the terminal hint; all three are properties of a prompt, and a plan library has no tool descriptions to permute or clauses to remove, as dependencies live in context conditions. These axes are therefore absent from the plan-based arm rather than left unmeasured in it—which is the same fact the comparison is about, seen from the other side.

6.2.3. What the Comparison Establishes

The eight campaigns measure a *guarantee*: whether a correct schedule is a property a design holds across configurations, or an outcome obtained at the configuration where it was measured. Two results establish it, and they work together.

The first concerns what repetition settles and what it leaves open. The *mixed* column—queries valid in some repeats and invalid in others—is zero throughout Table 6 except for a single query in C6: 1 cell out of the 700 query-campaign cells of the grid, so at temperature 0 with a pinned seed a query's verdict does not move between repeats. The trace is the weaker case: at the reference configuration only 52% of queries in the AutoGen arm and 86% in the LangGraph arm produce an identical sequence in all three repeats, the rest alternating between the two valid linearizations. The grid rests on the first property and not the second: because the verdict is stable under repetition, every movement in

the valid-schedule column can be attributed to the setting that was changed rather than to sampling.

The second is the size of that movement. The rate stands at 99.0% at the reference configuration and at 32.0% and 28.0% once three more agents are added, and the ordering behavior shifts when the tool declarations are permuted or a prerequisite sentence is deleted. In the baseline arms a correct schedule is therefore an outcome of the configuration it was measured in. In the hybrid arm the same quantity is a property of the plan library: the plans that fire are the same whatever the query, the tool declaration order, the sampling seed, the fusion policy or the model behind the services, and extending the pipeline is a design-time edit to that library. That invariance is what the architecture contributes, and isolating it—rather than a percentage-point gap under one favorable setting—is what this comparison is built to do.

6.3. Decision Quality

This subsection reports what the pipeline decided, which the schedule alone does not settle. All numbers are measured on the shared backend and characterize the retrieval–extraction–authorization chain that every arm drives: the same knowledge bases, the same authorization service and the same disjunction rule. That rule is evaluated in code and never by a language model—by an AgentSpeak plan in the Decision Engine Agent of the hybrid arm, and by the identical rule in the shared service module for the two baselines. The two control layers produce identical figures wherever they produce identical schedules, and the plan-based arm, measured on its own 300 runs of the same campaign over the same 100 queries, returns the same document-level figures, down to the confusion matrix. That is the expected result rather than a lucky one: a schedule fixed by construction determines which agents run and in what order, not what those agents extract or conclude. A single column therefore suffices for most rows.

6.3.1. Retrieval

Under the reference configuration, Recall@1 is 93.0%: the target document is placed first in that share of runs. Recall@3 and Recall@5 are both 95.0%, and the MRR is 0.937. The figures are identical in every arm and in every campaign that leaves the query text untouched, as they must be: retrieval is a service call, not a control decision, and the embedding model is the same throughout. The nine-agent pipeline is the one exception, because a query-rewriting agent runs before retrieval there: Recall@1 rises to 95.0% and the MRR to 0.950. Retrieval is therefore not the limiting factor in the authorization results that follow: in essentially every run the evidence needed to decide correctly was present in the retrieved set.

6.3.2. Knowledge-Base Extraction Accuracy

Table 7 reports the accuracy of the offline extraction step that populates the two authorization knowledge bases, scored per channel and per field against the corpus annotation over all 167 posters. A document is scored on a field only when its annotation carries that signal.

Two observations bear directly on the architecture's claims. First, the visual security badge is the weakest channel at 83.0%, and it is weakest precisely on the multimodal group, where a badge coexists with a textual label—an indication that the visual extractor is more often distracted than blind. Second, and more importantly for the security argument, every *wrong* entry in this table is a fact that the BDI layer will subsequently process perfectly reproducibly. Extraction error is the belief-contamination surface discussed in Section 6.7: determinism downstream of a wrong belief preserves the error rather than correcting it.

Table 7. Multimodal extraction accuracy of the two authorization knowledge bases, scored offline against the corpus annotation. *Scored* is the number of posters whose annotation carries the field. *Missed* means the field was not extracted; *wrong* means a different value was extracted; no hallucinated values—fields extracted for posters that carry no such signal—were observed in any channel.

Channel	Field	Scored	Accuracy	Dominant Error
Text	security label	125	94.4%	missed (5), wrong (2)
Text	permitted organization	137	87.6%	wrong (15), missed (2)
Image	visual security badge	100	83.0%	missed (12), wrong (5)
Image	institutional logo	100	95.0%	missed (3), wrong (2)

6.3.3. Authorization Accuracy

Table 8 reports authorization at the document level and at the level of every retrieved chunk, with the false-allow and false-deny rates separated. The positive class is *allow*; FAR therefore counts documents released to a user who was not entitled to them, which in an access-control setting is the consequential error.

Table 8. Authorization accuracy under the reference configuration C1 (OR fusion), end-to-end (a document never decided counts as denied). FAR is the false-allow rate, FRR the false-deny rate. Chunk-level figures score every retrieved chunk against the corpus annotation, not only the target document. The AutoGen and LangGraph arms produce identical figures, because they drive the identical service module.

Level	Unit	Accuracy	FAR	FRR
Document	300 runs	62.0%	72.0%	4.0%
Chunk	1461 chunks	59.3%	74.7%	0.0%

At the document level the confusion matrix behind Table 8 is 144 true allows, 108 false allows, 42 true denials and 6 false denials over 300 runs, which is a precision of 57.1%, a recall of 96.0% and an F_1 of 71.6%; at chunk level the same three figures are 52.9%, 100.0% and 69.2%. Restricting the document-level count to the 285 runs in which the target document was retrieved and decided raises accuracy to 62.1% and the false-allow rate to 76.6%, with 0.0% false denials. The figure that governs deployment in an access-control setting is the false-allow rate of 72.0% at the document level. Under a disjunctive fusion policy a document is authorized when *either* modality grants access, so a single incorrectly extracted permissive label—or a correctly extracted label in a channel whose counterpart would have denied—releases the document. Recall is correspondingly near-perfect (4.0% FRR): almost nothing that should have been released is withheld. This is not an incidental weakness of the implementation; it is the designed behavior of the policy, and Section 6.4 measures the alternative.

What this table adds to the orchestration results is the separation between the two. The reference configuration is the campaign in which the control layer performs best: 99.0% and 99.0% of runs reach a valid schedule and no query is decided differently across its repeats. The same campaign returns a 72.0% false-allow rate. Every run behind that rate executed a valid schedule and called every agent exactly once in a correct order, and no query behind it changed its verdict between repeats. That is the clearest available illustration of why an orchestration metric cannot stand alone: a system evaluated only on schedule reproducibility would report this configuration as a success.

6.3.4. Breakdown by Modality Group

Table 9 separates the four annotation groups of Section 5.1 plus the conflicting-modality subset, in which the textual and visual channels disagree about a retrieved target document.

The error is not concentrated in the visual channel. The false-allow rate is high in every group, and it is highest in the two groups whose evidence lives in a single modality: 93.8% for the text-only group A and 90.9% for the image-only group B1. It is lowest in group C, the only group in which both channels carry the signal (40.0%).

Table 9. Authorization accuracy by annotation group under campaign C1. Group A carries all authorization metadata in text; group B1 carries all of it in images; group B2 has the security level in a badge and the organization in both channels; group C carries both dimensions in both channels. Row C* is the conflicting-modality subset of group C: the posters for which the two channels disagree about the target document (27 runs), the only group in which such a disagreement can occur. All figures are at full scale (600 runs) and identical for the two control arms.

Group	Where the Evidence Is	Acc.	FAR
A	text only	46.7%	93.8%
B1	image only (badge + logo)	60.0%	90.9%
B2	badge only, organization in both	50.0%	62.5%
C	both channels	80.0%	40.0%
C*	conflicting channels	66.7%	50.0%

The pattern therefore tracks modality *coverage* rather than extractor accuracy. For a poster whose annotation carries no signal in one channel, that channel’s knowledge-base entry holds no restrictive evidence and cannot deny; under a disjunctive rule one such channel is sufficient to release the document, no matter what the other channel found. This is the same structural fact that Section 6.4 quantifies from the corpus side—97 of the 167 posters, 58.1%, are described by exactly one modality—seen here from the decision side. Extraction error, measured in Table 7, adds to this but does not explain it: the visual badge is the weakest extractor at 83.0%, yet group B2, whose level is badge-only, has a *lower* false-allow rate than group A, whose level is a textual label extracted at 94.4%.

The conflicting-modality subset is the security-critical row. The two channels returned opposite verdicts on the target document in 117 of the 300 runs of each arm, whereas the 27-run subset scored here is the narrower one whose target posters the corpus itself annotates as conflicting cases. On that subset the decision is 66.7% accurate with a 50.0% false-allow rate: half of the conflicting cases that should have been denied were released. The reason is structural rather than statistical. A disjunctive rule has no defined behavior on disagreement at all: it grants whenever either channel grants, so a conflict is resolved in favor of access by construction. Section 6.4 measures what the conjunctive rule does to the same subset, and the answer is not simply better.

6.4. Fusion Policy and Threat Model

The Decision Engine Agent authorizes a document when either modality grants access. The rule is a disjunction over the two modality judgments and it is applied by an AgentSpeak plan: the model is not consulted at this step and produces no part of the outcome. Where a language model is used at all in the authorization path, it is one service call earlier, to parse a knowledge-base entry’s natural-language fields into the structured terms the plan reasons over; the same decision, taken over the same two judgments, is implemented by the identical rule in the service module the baseline arms import, which is why the two designs return the same decision-quality figures. What follows is therefore about a policy choice rather than about model behavior, and it needs to be stated as one, together with the threat model under which it is defensible and the alternative it was chosen over.

Threat model. The policy assumes an operator-controlled corpus in which annotation may be *incomplete* but is not *adversarial*: a poster may carry its security level in only one

channel, and an extractor may miss a signal, but no party is assumed to be inserting misleading badges or labels in order to obtain access. Under that assumption the dominant failure mode is a missing signal, and the cost of a conjunctive rule is structural rather than statistical. As Section 5.1 shows, the text knowledge base holds an entry for 137 of the 167 posters and the image knowledge base for 100, but both hold an entry for only 70; the remaining 97 posters—58.1% of the corpus—are described by exactly one modality, and the opposite knowledge base is empty for every one of them. These counts are properties of the annotation design recorded in the released ground-truth table (Table 1), not estimates. A rule requiring both modalities to grant would deny all 97 regardless of entitlement.

Under an adversarial threat model this assumption fails. If a document's badge or label can be manipulated, a disjunctive rule requires the attacker to compromise only the weaker of two channels, and Table 7 identifies which channel that is. The policy is therefore appropriate to the deployment assumed here and inappropriate to a deployment in which the corpus is untrusted.

Measured alternative. Campaign C8 re-runs the reference configuration with conjunctive fusion and no other change. Table 10 reports both policies on the same queries.

Table 10. Disjunctive (OR) against conjunctive (AND) fusion, campaign C1 against campaign C8, both at full scale on the same 100 queries and 600 runs per policy, so that the policy change is the only difference between the two columns.

Measure	OR (C1)	AND (C8)
Document accuracy	62.0%	91.0%
False-allow rate (FAR)	72.0%	4.0%
False-deny rate (FRR)	4.0%	14.0%
Chunk-level accuracy	59.3%	93.5%
Chunk-level FAR	74.7%	4.8%
Chunk-level FRR	0.0%	8.6%
<i>conflicting-modality subset only (27 runs)</i>		
Document accuracy	66.7%	55.6%
False-allow rate (FAR)	50.0%	0.0%
False-deny rate (FRR)	20.0%	80.0%

The trade-off is large, and it is a policy parameter rather than a model property. Conjunctive fusion drives the document-level false-allow rate from 72.0% to 4.0%—within measurement noise of zero—and raises document accuracy from 62.0% to 91.0%. The price is false denial: FRR rises from 4.0% to 14.0%. Because the policy is evaluated inside an AgentSpeak plan rather than inferred by a model, switching between the two is an edit to a single rule. Each run records the two modality judgments separately as well as the fused outcome, so `deny_on_conflict`, `text_priority`, `image_priority` and `defer` could be scored on the same runs without executing the campaign again. We have not done so: the OR/AND pair is what this study measures, those four rules are unevaluated here, and the comparison is left to Section 6.11.

Neither policy handles disagreement. The last block of Table 10 isolates the conflicting-modality subset, and it is the result that should govern the deployment choice. Under OR, that subset is decided at 66.7% accuracy with a 50.0% false-allow rate: a conflict is resolved in favor of access by construction. Under AND it is decided at 55.6% accuracy with a 80.0% false-deny rate: the conflict is resolved against access, equally by construction, and four in five entitled requests are refused. Neither figure is a model behavior—each is what the fusion rule does when its two inputs disagree. A corpus with conflicting annotation therefore needs a policy that names disagreement explicitly (`deny_on_conflict`, or `defer`

to human review), not a choice between two rules that both dispose of it silently. Because the rule is a single AgentSpeak clause, adding that third policy is an edit rather than a redesign, and it is inspectable and switchable without retraining or re-prompting anything. Whether it decides the conflicting subset any better is an open question in this paper: no such rule was run in these campaigns.

For a deployment whose corpus is untrusted, or whose false-authorization cost dominates its false-denial cost, the conjunctive rule—restricted to the 70 posters described by both modalities, with the single-modality remainder and the conflicting subset routed to human review—is the defensible default. That configuration follows from the measurements above rather than being one of them: it was not run as a campaign, and every number in this section was measured under the disjunctive policy or its conjunctive counterpart.

6.5. Latency and Computational Cost

Under the reference configuration on `qwen3.5:9b`, mean end-to-end latency per run was 85.3 s for the AutoGen arm and 89.0 s for LangGraph, with control-layer consumption of 10,479 and 10,253 tokens per run, respectively. Almost all of the wall-clock time is spent inside the shared service backend—six model-backed agent invocations per run against a locally served 9-billion-parameter model—so the two arms differ mainly in control-layer token cost, where the difference is under 3%.

The comparison with the hybrid arm is structural rather than numerical, and we make it in tokens rather than in seconds, because token cost is a property of the design while wall-clock time is a property of the machine that happened to run the campaign. The hybrid system spends no tokens on control: plan selection is a JASON deliberation cycle, not a generation step, and its campaigns record zero control-layer tokens in every run against the roughly ten thousand per run the LLM-controlled arms consume. Its per-run service cost is the same six invocations. The margin widens with pipeline length, as the LLM-controlled arms carry the full transcript of prior tool calls into every subsequent control decision, whereas plan selection does not grow with the number of agents already executed. A systematic scaling study over corpus size and agent count is left to future work (Section 6.11).

6.6. Auditability

For a plan-derived judgment, three artifacts are relevant: the beliefs present when the event was handled, the AgentSpeak plan selected for that event, and the FIPA-ACL messages that carried inputs and results. Together, these artifacts explain which programmed rule produced the authorization outcome. An LLM-generated explanation does not provide the same evidence because it may be a post hoc account rather than a record of the mechanism that controlled the workflow [24,25].

A deployed system logs belief updates, service provenance, plan identifiers, message identifiers and timestamps; what the architecture supplies is the interpretation of that trace at the level of the rule that fired.

6.7. Security: Direct Injection Resistance and Belief Contamination

Instructions contained in a user query are not executed as AgentSpeak code. They reach the reasoning layer only through typed data returned by the service boundary, while the available actions and transitions remain defined by the plan library. Under this threat model, a user cannot directly replace the authorization rule through a prompt [26,27]. The argument is structural: it follows from where the service boundary is drawn, and no injection campaign was run against it in this study.

The complementary surface sits one layer earlier, at belief formation: retrieved text or visual content can induce the extraction model to produce an incorrect structured fact, which the BDI layer then processes reproducibly [27]. Table 7 measures the non-adversarial baseline of that surface—between 83.0% and 94.4% per-field accuracy, with 15 wrong organization assignments in the text channel and 5 wrong badges in the image channel across the corpus—and Table 9 traces the resulting false authorizations to the groups whose evidence is visual only. Every one of those errors was produced by a run with a valid schedule whose verdict did not change between repeats.

Determinism at the reasoning layer therefore changes the shape of the security problem rather than removing it: it converts an unpredictable failure into a predictable one, which is a precondition for auditing and for regression testing, but it does not validate the belief. The controls this implies—schema validation at the service boundary, provenance recording per belief, agreement between independent extractors on the same document, confidence-gated deferral, and human review for high-impact facts—are architecturally available at the MCP boundary but were not evaluated in this study. Adversarial evaluation of this surface is the first item of Section 6.11.

6.8. Extensibility

Adding a reasoning dimension requires a new agent, its belief schema and AgentSpeak plans, and registration of the messages it exchanges with the orchestrator. The plans of the existing domain agents need not be edited. The fusion agent may, however, require a new rule if the additional judgment changes the fusion policy.

Campaign C2 gives this claim an empirical edge. Adding three agents to the pipeline is, for the hybrid arm, three new plans and three new message registrations, after which the schedule is again fixed by construction. For the LLM-controlled arms the same change is not an edit at all—nothing in the implementation is modified except the tool roster—and the valid-schedule rate falls from 99.0% to 32.0%. Extensibility in the plan-based design is work that is done once, at design time, and then holds; in the model-controlled design it is a re-derivation performed on every request, whose success rate degrades as the number of agents grows.

Classical BDI systems offer a similar modularity at the agent level, but their extensibility is constrained by the knowledge integration problem: adding a new reasoning agent requires that its belief base be manually curated or pre-structured, a process that does not scale to dynamic, evolving, or multimodal knowledge sources. In the hybrid architecture, a new reasoning agent's knowledge source can be any queryable service exposed through MCP—a vector store, a relational database, an external API, or a domain ontology—without any modification to the agent coordination layer. The key differentiator over classical BDI is therefore not runtime belief extraction, but, rather, *dynamic querying of pre-constructed knowledge sources*: the orchestration and querying mechanism is generic and extensible, while the knowledge construction step may be performed offline.

Extensibility here means that a reasoning component is added locally, behind defined interfaces: the three agents of Section 5 were introduced as independent units, each with its own plans and knowledge-base connection, and a fourth would leave the two modality-specific agents untouched. A new domain still supplies its own rules, schemas, service mappings and tests.

6.9. Comparative Summary

Table 11 summarizes the evaluation across the properties examined in Sections 6.2–6.8, comparing the hybrid BDI + RAG architecture against classical BDI and LLM-based agentic frameworks. The ratings reflect architectural properties rather than empirical measure-

ments: a property is marked as supported if the architecture provides a formal or structural guarantee, partially supported if it provides a mitigation without a guarantee, and not supported if the architecture offers no structural basis for the property. Two rows are stated in the terms the measurements above make available: *schedule validity under configuration change* is what Table 6 tests, and *belief integrity* is the property that no arm in this study possesses.

Table 11. Architectural comparison of the hybrid BDI + RAG system against classical BDI and LLM-based agentic frameworks across decision-critical properties. Cell colours duplicate the cell text for quick scanning: green = Yes, orange = Partial, red = No.

Property	Classical BDI	LLM-Agent (LangGraph/AutoGen)	Hybrid BDI + RAG (Ours)
Deterministic Reasoning & Orchestration	Yes	No	Yes
Schedule validity invariant to configuration	Yes	No	Yes
Auditability	Yes	Partial	Yes
Direct Prompt Injection Resistance	Yes	No	Yes
Belief integrity under indirect injection	Partial	No	No
Extensibility	Partial	Partial	Yes
Dynamic Knowledge Querying	No	Yes	Yes
Multimodal Support	No	Partial	Yes

The table reveals the precise positioning of the hybrid architecture. Classical BDI and the hybrid architecture share the determinism, invariance, auditability and direct-injection rows because both ground their reasoning in deterministic, inspectable plan execution with no LLM in the reasoning or orchestration loop. Where they diverge is on dynamic knowledge querying and multimodal support: classical BDI cannot query pre-constructed knowledge sources at runtime in a generic, extensible way, and cannot draw beliefs from visual content. The hybrid architecture inherits BDI's formal guarantees while extending them to arbitrary, pre-constructed knowledge sources accessed through MCP-connected services.

LLM-based agentic frameworks present the inverse profile. They support dynamic knowledge querying natively and can process multimodal content through vision-capable models, but at the cost of the formal properties. Auditability is marked as partial rather than absent because these systems can produce chain-of-thought traces that approximate an explanation; however, these traces are generated text rather than formal execution records and may not accurately reflect the actual computational process [24,25]. Multimodal support is marked as partial because LLMs with vision capabilities process multiple modalities through the same probabilistic generation layer, with no structural separation between modality-specific reasoning paths and no formal mechanism for resolving conflicts between signals from different modalities. The hybrid architecture provides both through independent reasoning agents per modality with explicit conflict resolution logic in AgentSpeak plans, as demonstrated by the Decision Engine Agent in Section 5.

The belief-integrity row is negative in all three columns: a deterministic reasoner placed downstream of a probabilistic extractor does not validate what the extractor produced, and classical BDI is marked partial because its beliefs are curated rather than extracted, which moves the question to the curation step.

The rows also scope the claim. Generation quality and conversational fluency remain properties of the LLM components both paradigms share, as the generation agent uses the same infrastructure as its LLM-agent alternatives. What the hybrid architecture changes is where those components sit: their outputs feed a deterministic reasoning layer rather than constituting the reasoning itself.

6.10. Limitations

End-to-End Variability. The architecture does not provide end-to-end determinism. Embeddings, multimodal extraction, and final response generation remain probabilistic. The reasoning trace is reproducible only when the agents receive the same relevant beliefs and structured service outputs.

Belief Integrity and Indirect Prompt Injection. The MCP boundary does not establish the truth of the beliefs it carries. A source document can contain instructions or visual patterns that cause the extraction model to return an incorrect structured fact [27]. The BDI layer will process that fact reproducibly, but reproducibility does not make it correct. Controls such as schema validation, provenance checks, agreement between independent extractors, and human review for high-impact facts were not evaluated in this study, and no adversarial condition—manipulated badges, conflicting injected labels, poisoned retrieval chunks—was tested. The extraction accuracies of Table 7 are a benign-corpus baseline and should be read as an upper bound.

Dataset Scale and Security Semantics. The evaluation uses one corpus of 167 posters, unevenly distributed across six institutions, whose access-control metadata were introduced by a deterministic annotation pipeline. Real enterprise labels can be ambiguous, absent, inconsistent, obsolete, or adversarially manipulated; none of those conditions is represented here. The distribution of authorization evidence across modalities is a property of the annotation design, and a different distribution would change the fusion results of Section 6.4 directly.

Campaign Scale and Model Coverage. All eight campaigns are complete at 600 runs per campaign across the two LLM-controlled arms, the plan-based arm adds 600 runs in C1 and C2, and the seed axis is covered by three points, but the whole grid was obtained with a single control-layer model, qwen3.5:9b. Every sensitivity result in Section 6.2.2 is therefore a statement about that model under this harness rather than about LLM control layers in general: another model family, or a larger model of the same family, may be more or less sensitive on any of these axes, and these measurements do not establish which of the reported effects are properties of the design and which are properties of this particular control model. Replicating the grid on further control-layer models is accordingly part of the work set out in Section 6.11. The axes covered are pipeline length, sampling seed, tool declaration order, the prerequisite and terminal hint sentences, and the fusion policy; coordination topology is not varied, so nothing is claimed here about how either control layer behaves when a coordination protocol is stated as a requirement. Confidence intervals are reported per rate, but no claim is made about long-run behavior across model versions, hardware, or service implementations.

Baseline Configuration Dependence. The comparison depends on the implemented workflows. The baselines are given tool descriptions stating their prerequisites and no ordering instruction, which is the closest available analogue of an AgentSpeak context condition. Alternative prompting, explicit ordering instructions, tool-call constraints, or a deterministic controller wrapped around the model would change their results—indeed, campaign C5 shows how far a purely presentational change moves them. What such a controller would demonstrate, however, is that scheduling correctness had been moved out of the model, which is the architectural point at issue.

Answer Faithfulness. Whether the generated answer is faithful to the authorized chunks is not scored. The generated answers are retained and are available from the corpus owner on reasonable request, so that an independent judge or human rating pass can be performed.

Audit Records. Plan inspection makes the applied rule interpretable, but end-to-end auditability also requires retention of service inputs and outputs, belief updates, messages,

plan identifiers, and timestamps. The extraction step is not auditable in the same rule-level sense as AgentSpeak execution.

Plan Authoring and Domain Transfer. The reasoning logic must be written as AgentSpeak plans by someone who understands both the domain and the plan language. Extensibility was demonstrated through separate text, image, and fusion agents, not through multiple independent domain migrations.

6.11. Future Work

Adversarial belief contamination. The most consequential open item follows directly from Section 6.7. A dedicated campaign should introduce malicious text spans, adversarial or forged visual badges, deliberately conflicting text–image labels, removed metadata, and poisoned retrieval chunks, and measure how far each propagates into an authorization decision. The same campaign should evaluate the defenses the MCP boundary makes available—schema validation, per-belief provenance, cross-extractor agreement, confidence-gated deferral to human review—as an ablation, so that each is costed in false denials as well as credited in prevented false allowances.

Scale, latency and cost. Section 6.5 reports cost at one corpus size and two pipeline lengths. A scaling study across corpus size, retrieved-set size, agent count and concurrent request load would establish whether the control-layer token saving observed here grows as predicted with pipeline length, and where the plan-based design’s own limits lie.

Replication on further control-layer models. The most direct extension of the study is to repeat the eight-campaign grid on additional control-layer models—a larger model of the same family, and models from other families—under the identical harness, service module, tool descriptions, query set and seeds, so that the model is the only thing that changes. That replication is what separates a finding about LLM control layers from a finding about one model. It would establish, in particular, whether a stronger control layer buys away the presentational sensitivities reported here—the order in which tool declarations are listed, the prerequisite clause—while leaving the collapse under a longer pipeline in place, and where the boundary between those two classes of effect lies as a function of model capability. Varying the coordination topology, by requiring the coordinator to be re-entered after each worker as the FIPA-ACL protocol of the hybrid design does, belongs to the same extended grid and was not measured here.

Wider evaluation surface. Extending the seed axis and repeating the study on a larger and more naturally annotated multimodal access-control corpus are both required before the sensitivity result of Section 6.2.2 can be stated as a general property rather than as a measurement on this corpus and this control model.

Answer faithfulness and fusion policies. An LLM-judge or human faithfulness pass over the retained answers, and a full comparison of the `deny_on_conflict`, modality-priority and confidence-based fusion rules against the OR/AND pair reported here would close the two remaining gaps in the decision-quality picture.

7. Conclusions

We developed and evaluated a hybrid BDI + RAG architecture for cases in which decisions must be traceable even though relevant evidence is stored in unstructured text and images. Its central implementation choice is to separate evidence extraction from decision making. MCP services retrieve or structure information; JASON agents apply AgentSpeak plans; FIPA-ACL messages define the coordination sequence; and a designated reasoning agent performs judgment fusion.

The access-control implementation used 167 posters from six Belgian institutions and three reasoning agents for text, image, and fusion judgments. It was compared against

AutoGen and LangGraph reconstructions that drive the identical service backend, so that the component choosing the next step was the only independent variable, across eight campaigns that each change one setting at a time. Those eight campaigns are the baseline grid, 4800 AutoGen and LangGraph runs in total; the plan-based arm was run in the two of them that have a counterpart in a plan library, 300 runs each.

Under a favorable configuration, both baselines schedule the workflow correctly, and at temperature 0 with a pinned seed, no query changes its verdict between repeats, although the exact sequence is reproduced in only 52% and 86% of queries, the rest alternating between two equally valid orders. What the campaigns add is that this correctness is a property of the configuration rather than of the system: the valid-schedule rate stands at 99.0% for both arms on the six-agent pipeline and falls to 32.0% (AutoGen) and 28.0% (LangGraph) when it grows to nine, although nothing in that campaign asks the control layer to reason about anything new, and the ordering behavior moves when the tool declarations are merely permuted or a single prerequisite clause is removed. Because the whole grid was obtained with one control-layer model, these are findings about that model under this harness; separating what belongs to the model from what belongs to the design requires the replication on further control-layer models set out in Section 6.11. The hybrid BDI + RAG arm needs no such qualification. Run through the same two campaigns, 300 runs each, it produced a single sequence throughout: 100.0% valid and 100.0% canonical at six agents and again at nine, with every query identical across its three repeats. The axes that move the LLM-controlled arms—a sampling seed, the order of the tool declarations, a sentence of tool documentation—have no counterpart in a plan library, because in that design the schedule is written once, at design time, rather than re-derived on each request.

The decision-quality results are the necessary counterpart. Under the reference configuration—the campaign in which the control layer performs best—every arm produced a valid schedule in essentially every run, with no query changing its verdict between repeats, while returning a 72.0% false-allow rate at the document level. A schedule can therefore be valid, canonical and stable under repetition, and the authorization decision still wrong. The false allowances are not confined to one channel: they track how many modalities hold evidence for a poster at all, and are highest in the two groups described by a single modality. Conjunctive fusion reduces the false-allow rate to 4.0% at the cost of 14.0% false denials, and on the subset where the two channels disagree, neither rule is defensible—which makes the fusion policy a parameter to be set against a stated threat model, and the conflicting subset a case that needs a rule of its own, rather than a property of either architecture.

The implementation also clarifies the trade-off introduced by the design. Plans make authorization rules and message transitions inspectable, but decision quality still depends on the beliefs supplied by probabilistic services. In particular, indirect prompt injection and extraction errors remain possible before a belief enters the reasoning layer, and determinism downstream of a contaminated belief preserves the error rather than correcting it.

Future work follows three lines. The first is adversarial robustness at the belief-formation boundary: constructing evaluation sets with malicious text, adversarial badges, conflicting text-image labels, missing metadata, and intentionally poisoned retrieval chunks, and measuring how far controls such as schema validation, provenance checks, agreement between independent extractors, and human review of high-impact facts contain the resulting belief errors. The second is runtime characterization: measuring latency, computational cost, and scalability as the document corpus, the number of reasoning agents, and the query load grow. The third is repeating the comparison on further control-layer models, over more queries, and across application domains, so that the sensitivity results

reported here can be attributed to the design rather than to one model. Sustainable IoT automation is a suitable target for that third line because it combines probabilistic sensor interpretation with policy- and safety-relevant actuation; whether the proposed separation remains practical under live streams and real-time constraints is an empirical question for that study.

Author Contributions: Conceptualization, B.T.T. and M.C.; methodology, H.Y.; software, H.Y.; validation, H.Y.; formal analysis, H.Y.; investigation, H.Y.; resources, B.T.T. and M.C.; data curation, H.Y.; writing—original draft preparation, H.Y.; writing—review and editing, B.T.T. and M.C.; visualization, H.Y.; supervision, B.T.T. and M.C.; project administration, B.T.T. and M.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset analyzed in this study consists of 167 research posters from six Belgian institutions. The posters are the property of Flanders Make and are not publicly available because of confidentiality and privacy restrictions. Access may be granted by Flanders Make upon reasonable request, subject to its applicable data-sharing agreements. The same conditions apply to the two authorization knowledge bases, whose entries reproduce content extracted from the posters, as well as to the generated answer texts, which are required for an independent faithfulness assessment. They also apply to the 100 query strings, because each query paraphrases the title of its target poster, and to the raw per-run traces, which contain the query text, retrieved poster filenames, and generated answers. The released artifacts are derived from the proprietary corpus and do not reproduce its content. In all released artifacts, documents are identified only by anonymized sequential labels; original filenames, repository paths, and source locations are removed. Two annotation tables are released. The first is the anonymized corpus ground-truth table, which records, for each poster, its institution, security level, annotation group, four binary channel indicators, and the per-document outcome of each knowledge-base construction pass. The second is the per-query ground-truth table for the 100 evaluation queries, which records the query identifier, annotation group, whether the query corresponds to a conflicting-modality case, and the ground-truth decision. The scored results of all eight baseline campaigns and the two campaigns in which the plan-based arm was executed are also released as per-run tables, together with the overall and group-wise metric files, the observed control sequences, and the tool-call analysis. These artifacts allow every composition figure in Section 6.1.3 and every score in Section 6.3 to be recomputed, and the alternative fusion rules to be re-scored, from the released tables without access to the raw traces. The complete implementation is publicly available at <https://github.com/halilyesilceng/hybrid-bdi-rag> (release v1.0; accessed on 6 September 2026). The repository contains the JASON/JADE agents and their AgentSpeak plans; the MCP service backend; the shared service and tool modules imported by both baseline arms; the AutoGen and LangGraph control layers; the campaign runner; the pre-flight checks; and the scorer that produces all metrics reported in Section 6, including the group-wise and fusion-policy analyses. It also contains a prompt and tool-schema reference specifying the authorization prompt, answer-generation system message, MCP tool schemas, both complete tool-description sets, the shared system message exactly as sent to the models, and the AgentSpeak plans of the hybrid arm. The two annotation tables and the campaign result sets described above are included as well. Finally, the repository provides a synthetic benchmark generator that constructs an anonymized corpus with the same structure and annotation semantics as the annotated corpus, together with its own runnable query set. This allows the complete pipeline and evaluation harness to be executed end to end without access to the proprietary posters. The annotation pipeline that introduced the security semantics is not released. Its specification—including the two annotated dimensions, the channel carrying each dimension for each annotation group, and the determinism guarantee—is provided in Section 5.1, together with the distributions reported in Tables 1–3.

Acknowledgments: We would like to express our sincere gratitude to Siskon Industrial Automation and Software company for their generous support throughout this research. We are especially thankful for their genuine interest and wholehearted responsiveness in meeting our needs.

Conflicts of Interest: Author Halil Yesil is employed by the company Siskon Industrial Automation and Software. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. The Siskon Industrial Automation and Software had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

ACL	Agent Communication Language
AI	Artificial Intelligence
API	Application Programming Interface
BDI	Belief–Desire–Intention
DF	Directory Facilitator
FIPA	Foundation for Intelligent Physical Agents
FIPA-ACL	Foundation for Intelligent Physical Agents Agent Communication Language
FAR	False-Allow Rate (false-accept rate)
FRR	False-Deny Rate (false-reject rate)
HTTP	HyperText Transfer Protocol
IoT	Internet of Things
JADE	Java Agent Development Framework
JSON	JavaScript Object Notation
JSON-RPC	JavaScript Object Notation Remote Procedure Call
KB	Knowledge Base
LLM	Large Language Model
MAS	Multi-Agent System
MCP	Model Context Protocol
MRR	Mean Reciprocal Rank
PDF	Portable Document Format
RAG	Retrieval-Augmented Generation
RPC	Remote Procedure Call

References

1. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; tau Yih, W.; Rocktäschel, T.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 9459–9474.
2. Rao, A.S.; Georgeff, M.P. Modeling Rational Agents within a BDI-Architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, Cambridge, MA, USA, 22–25 April 1991; pp. 473–484.
3. Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y.J.; Madotto, A.; Fung, P. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.* **2023**, *55*, 1–38. [\[CrossRef\]](#)
4. Huang, L.; Yu, W.; Ma, W.; Zhong, W.; Feng, Z.; Wang, H.; Chen, Q.; Peng, W.; Feng, X.; Qin, B.; et al. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Trans. Inf. Syst.* **2025**, *43*, 1–55. [\[CrossRef\]](#)
5. Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; Guo, Q.; Wang, M.; et al. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv* **2023**, arXiv:2312.10997.
6. Bordini, R.H.; Hübner, J.F.; Wooldridge, M. *Programming Multi-Agent Systems in AgentSpeak Using Jason*; Wiley Series in Agent Technology; John Wiley & Sons: Chichester, UK, 2007. [\[CrossRef\]](#)
7. Bellifemine, F.L.; Caire, G.; Greenwood, D. *Developing Multi-Agent Systems with JADE*; Wiley Series in Agent Technology; John Wiley & Sons: Hoboken, NJ, USA, 2007. [\[CrossRef\]](#)

8. Yesil, H.; Tezel, B.T.; Challenger, M. BDI Agent-Based Access Control Reasoning for Multimodal Retrieval-Augmented Generation. In Proceedings of the 39th International Florida Artificial Intelligence Research Society Conference, FLAIRS-39 2026, Marco Island, FL, USA, 17–20 May 2026; Volume 39. [\[CrossRef\]](#)
9. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 1877–1901.
10. Bratman, M.E. *Intention, Plans, and Practical Reason*; Harvard University Press: Cambridge, MA, USA, 1987.
11. Wooldridge, M. *An Introduction to Multiagent Systems*, 2nd ed.; John Wiley & Sons: Chichester, UK, 2009.
12. FIPA. *FIPA ACL Message Structure Specification*; Technical Report SC00061G; Foundation for Intelligent Physical Agents: Geneva, Switzerland, 2002.
13. Mireshghallah, F.; Goyal, K.; Uniyal, A.; Berg-Kirkpatrick, T.; Shokri, R. Quantifying Privacy Risks of Masked Language Models Using Membership Inference Attacks. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022), Abu Dhabi, United Arab Emirates, 7–11 December 2022; pp. 8332–8347. [\[CrossRef\]](#)
14. Carlini, N.; Tramèr, F.; Wallace, E.; Jagielski, M.; Herbert-Voss, A.; Lee, K.; Roberts, A.; Brown, T.; Song, D.; Erlingsson, Ú.; et al. Extracting Training Data from Large Language Models. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security 21)*; USENIX Association: Berkeley, CA, USA, 2021; pp. 2633–2650.
15. Barnett, S.; Kurniawan, S.; Thudumu, S.; Brannelly, Z.; Abdelrazek, M. Seven Failure Points When Engineering a Retrieval Augmented Generation System. In Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering–Software Engineering for AI (CAIN 2024), Lisbon, Portugal, 14–15 April 2024; pp. 194–199. [\[CrossRef\]](#)
16. Yesil, H.; Gultekin, S.; Bozyigit, F.; Tezel, B.T.; Challenger, M. A Database-Independent LLM Framework for Real-Time Authorization in Retrieval-Augmented Generation. In Proceedings of the AAAI 2026 Workshop on Trustworthy Agentic AI, Singapore, 27 January 2026.
17. LangChain Inc. LangGraph: A Framework for Building Stateful, Multi-Actor Applications with Large Language Models. 2024. Available online: <https://github.com/langchain-ai/langgraph> (accessed on 6 September 2026).
18. Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; et al. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. *arXiv* **2023**, arXiv:2308.08155.
19. Wang, X.; Jiang, H.; Dong, Y.; Mu, M. Spatial-channel collaborative multi-scale graph interaction deep transfer learning for unsupervised rotating machinery fault diagnosis. *Eng. Appl. Artif. Intell.* **2026**, *176*, 114691. [\[CrossRef\]](#)
20. Yan, J.; Zhou, N.; Cheng, Y.; Zhang, F.; Wang, H.; Wang, M.; Jin, B.; Li, M.; Lu, Q.; Zhang, W. Application of machine-vision-driven physics-informed neural networks in pantograph–catenary system state detection. *Mech. Syst. Signal Process.* **2026**, *257*, 114577. [\[CrossRef\]](#)
21. Zhao, H.; Wang, B.; Fu, Y.; Zhou, M.; Li, N.; Gao, Z. Fixed-time adaptive smooth control with relaxed performance constraints for a canard rotor/wing UAV subject to aerodynamic interferences and actuator faults. *Aerosp. Sci. Technol.* **2026**, *178*, 112959. [\[CrossRef\]](#)
22. Ichida, A.Y.; Meneguzzi, F.; Cardoso, R.C. BDI Agents in Natural Language Environments. In Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), IFAAMAS, Auckland, New Zealand, 6–10 May 2024; pp. 880–888. [\[CrossRef\]](#)
23. Gatti, A.; Mascardi, V.; Ferrando, A. ChatBDI: Think BDI, Talk LLM. In Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), IFAAMAS, Detroit, MI, USA, 19–23 May 2025.
24. Turpin, M.; Michael, J.; Perez, E.; Bowman, S.R. Language Models Don’t Always Say What They Think: Grounding Errors in Chain-of-Thought Reasoning. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*; Curran Associates, Inc.: Red Hook, NY, USA, 2023; Volume 36, pp. 74070–74090.
25. Lanham, T.; Chen, A.; Radhakrishnan, A.; Zhao, J.; Denison, C.; Perez, E.; Hubinger, E.; Samunke, L.; King, N.; Amodei, D.; et al. Measuring Faithfulness in Chain-of-Thought Reasoning. *arXiv* **2023**, arXiv:2307.13702.
26. Perez, F.; Ribeiro, I. Ignore Previous Prompt: Attack Techniques for Language Models. *arXiv* **2022**, arXiv:2211.09527.
27. Greshake, K.; Abdelnabi, S.; Mishra, S.; Endres, C.; Holz, T.; Fritz, M. Not What You’ve Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec 2023)*; ACM: New York, NY, USA, 2023; pp. 79–90. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.