EXAR: A Unified Experience-Grounded Agentic Reasoning Architecture[*]

Ralph Bergmann[1,2], Florian Brand[1,2], Mirko Lenz[1,2], and Lukas Malburg[1,2]

[1] Artificial Intelligence and Intelligent Information Systems, Trier University,
54296 Trier, Germany, https://www.wi2.uni-trier.de
{bergmann,brand,lenz,malburgl}@uni-trier.de
[2] German Research Center for Artificial Intelligence (DFKI)
Behringstraße 21, 54296 Trier, Germany, https://ebls.dfki.de
{ralph.bergmann,florian.brand,mirko.lenz,lukas.malburg}@dfki.de

Abstract. Current AI reasoning often relies on static pipelines (like the 4R cycle from Case-Based Reasoning (CBR) or standard Retrieval-Augmented Generation (RAG)) that limit adaptability. We argue it is time for a shift towards dynamic, experience-grounded agentic reasoning. This paper proposes *EXAR*, a new unified, experience-grounded architecture, conceptualizing reasoning not as a fixed sequence, but as a collaborative process orchestrated among specialized agents. *EXAR* integrates data and knowledge sources into a persistent Long-Term Memory utilized by diverse reasoning agents, which coordinate themselves via a Short-Term Memory. Governed by an Orchestrator and Meta Learner, *EXAR* enables flexible, context-aware reasoning strategies that adapt and improve over time, offering a blueprint for next-generation AI.

Keywords: Case-Based Reasoning · Retrieval Augmented Generation · Large Language Models · Agents · Reasoning

1 Introduction

Case-Based Reasoning (CBR) has long provided a well-founded methodology for solving problems through the reuse of experience in the form of cases. Since the seminal work by Aamodt and Plaza [1], the CBR community has developed robust methods for implementing the 4R cycle–retrieve, reuse, revise, and retain–and has demonstrated their effectiveness in a wide range of applications. However, CBR has typically been applied in well-defined, bounded problem spaces, where case structures, similarity measures, and adaptation knowledge can be explicitly engineered and maintained, causing considerable knowledge engineering efforts [4].

In contrast, recent advances in large language models (LLMs) [39], retrieval-augmented generation (RAG) [11], and agentic hybrid reasoning systems (e.g.,

[*] The Version of Record is available online: doi.org/10.1007/978-3-031-96559-3_1

[13,30,38]) have shifted the focus of AI research towards open-ended, dynamic, and incompletely modeled environments. These systems rely on external information retrieval, tool usage, and generative reasoning to solve problems that cannot be exhaustively predefined. However, they still suffer from hallucination, are constrained by limited context windows, typically lack persistent and structured memory, do not support experience-based analogical reasoning, and offer only rudimentary mechanisms for reflection and continuous learning from their own problem-solving behavior.

Recent research has begun to explore the integration of CBR with LLM-based approaches, for example, by using LLMs to support retrieval and adaptation in CBR, or by embedding structured case memory in LLM-centric architectures. A recent research manifesto by Bach et al. [3] articulates this emerging convergence and proposes a systematic integration of CBR into neuro-symbolic and agentic AI paradigms.

In this paper, we propose *EXAR*, a unified architecture for Experience-Grounded Agentic Reasoning that generalizes both traditional CBR and modern RAG systems. We argue that it is time to break the CBR cycle, originally conceived as "cycle of sequential steps" [1] and to overcome the rigidity of static RAG pipelines, in favor of plan-based, goal-directed orchestration of reasoning processes. In this agentic perspective, reasoning steps such as retrieval, adaptation, prompt construction, symbolic inference, LLM-based generation, evaluation, and reflection are dynamically selected and composed based on the problem, context, memory state, and task goals. We envision that this orchestration process is itself experience-grounded: it can learn from problem-solving traces over time and adapt its internal control flow accordingly. The resulting architecture enables the integration of symbolic and sub-symbolic reasoning services under the control of a persistent, structured memory of heterogeneous knowledge units. This positions CBR not only as a technique, but as the conceptual backbone for building explainable, adaptive, and modular reasoning agents and presents a renewed view of CBR as a central enabler of explainable, adaptive, and modular AI agents.

The remainder of this paper is structured as follows: Section 2 presents the state-of-the-art in CBR, LLMs, and their integration, as well as recent multi-agent architectures in agentic hybrid reasoning. Section 3 introduces our proposed experience-grounded agentic reasoning architecture, *EXAR*. Section 4 discusses interpreting existing CBR-LLM approaches within the *EXAR* architecture. Section 5 outlines challenges related to employing *EXAR*, and Section 6 describes initial implementation steps. The paper is summarized, and an outlook is provided in Section 7.

2 State-of-the-Art

Recent research has begun to explore the interplay between Case-Based Reasoning (CBR) and large language models (LLMs) [3], as well as their integration into more complex, agentic reasoning architectures.

2.1 Using LLMs for CBR Tasks

A growing number of approaches have explored the use of large language models to support individual components of the CBR cycle. Most prominently, LLMs have been applied to case adaptation [20], which is traditionally one of the most knowledge-intensive steps. Prompt-based adaptation strategies allow LLMs to transform retrieved solutions to fit new problems, either through zero-shot instruction prompting or in-context few-shot prompting using similar past cases. In addition, LLMs have been explored as similarity assessors [21,34], for example, by generating structured feature comparisons between cases and queries or by paraphrasing case descriptions to normalize semantic variations. More recently, LLMs have also been employed to generate new cases from existing ones from unstructured text corpora, such as manuals or web data, reducing the burden of manual case authoring [7, 29], or to aid in the creation of the vocabulary of the cases [7]. A first study has shown that LLMs can be effectively embedded into CBR pipelines to improve flexibility and expressiveness, although challenges remain with regard to consistency, control, and explainability [32].

2.2 Using CBR to Overcome LLM Weaknesses

Although LLMs demonstrate impressive fluency and generalization, they suffer from several well-documented limitations, including hallucinations, lack of long-term memory, poor control over reasoning behavior, and static knowledge based on the training data cutoff. In this context, CBR has been proposed as a means to ground LLM outputs in a structured, verifiable experience. Case-based memory structures can serve as an external memory layer for LLMs [33], enabling more context-aware and traceable reasoning. When integrated into RAG pipelines, case bases provide targeted, high-quality prompt context, outperforming unstructured retrieval from generic corpora [36]. Moreover, the reuse of past cases as exemplars for few-shot prompting allows for context-sensitive generalization, with better alignment to task-specific constraints [9]. CBR also contributes to explainability and traceability in LLM-based systems. Since each generated output can be linked back to specific retrieved cases, the reasoning trace becomes more interpretable and auditable [12]. Additionally, CBR structures enable reflection mechanisms, such as evaluating past outcomes, detecting failure patterns, and adapting future behavior accordingly, thus addressing key deficits in LLM autonomy.

2.3 Agentic Hybrid Reasoning

Beyond isolated interactions between CBR and LLMs, recent work has shifted toward agentic architectures that embed reasoning processes within autonomous, tool-using agents [13, 30, 38]. In such systems, reasoning is modeled not as a fixed pipeline, but as a dynamic sequence of actions selected to achieve a goal, possibly involving multiple modalities and memory structures. Multi-Agent architectures such as CAMEL [23], and MetaGPT [15] exemplify this trend. These systems operationalize reasoning through combinations of LLM-based generation, retrieval, tool use, subgoal formation, and reflection. However, most of them lack persistent, structured long-term memory and cannot effectively reuse prior problem-solving experience in a structured way. To address this, emerging proposals suggest augmenting agentic reasoning with case-based memory systems that serve both as episodic storage and strategic reasoning guidance [33]. This hybridization opens the door to plan-based orchestration of reasoning steps, such as retrieval, reuse, adaptation, generation, and validation, selected dynamically based on the current goal, task context, and memory state. In this view, CBR evolves from a standalone reasoning paradigm into a core memory and control structure within agentic, neuro-symbolic reasoning systems, supporting explainability, modularity, and continuous learning.

3 The *EXAR* Architecture

We introduce *EXAR*, a unified architecture for Experience-Grounded Agentic Reasoning, designed to integrate the core concepts of CBR, RAG, and other reasoning paradigms within a dynamic agentic control architecture. *EXAR* aims to overcome the limitations of static reasoning pipelines, such as in CBR and, thus, to overcome the "cycle of sequential steps" [1]. In the following, we present an architectural overview of *EXAR*. Afterwards, we discuss the individual layers of the architecture and their specific tasks and relationships to other layers in the architecture.

3.1 Architecture Overview

Figure 1 provides an overview of the proposed *EXAR* architecture, which is structured into four conceptual layers. The *Data & Knowledge Layer* encompasses all domain-relevant information sources within a given application context such as databases, documents, logs, or distributed information silos, but also implicit expert knowledge of humans. This layer represents what is sometimes referred to as the organizational memory in enterprise settings. In a preprocessing step, this information is extracted, analyzed, and partially formalized into (semi-)structured entities, which we refer to as *Knowledge Units (KUs)*. These KUs are persistently stored in the *Long-Term Memory (LTM)* and serve as the primary knowledge base for reasoning. The core reasoning processes take place in the *Neuro-Symbolic Reasoning Layer*, which consists of a set of interacting reasoning agents. These include the classical CBR and RAG processing steps as well as potentially other reasoning methods, including the usage of tools by the agents. The agents exchange reasoning artifacts and intermediate results via a shared *Short-Term Memory (STM)* that reflects the dynamic reasoning state, functionally similar to nodes in a search or inference graph. The overall reasoning flow is governed by the *Orchestrator* (sometimes also called Controller in the literature [30]), which selects, configures, and activates agents based on the current task, context, and internal reasoning state. This enables flexible,

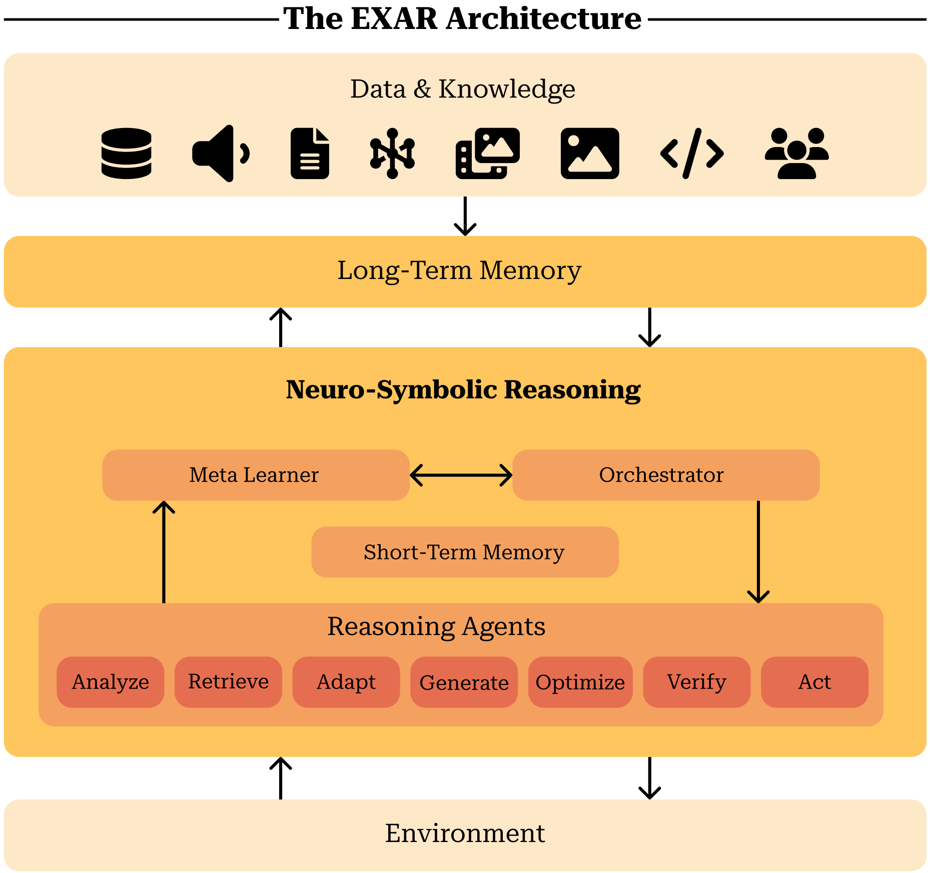

Fig. 1. Architecture Overview of *EXAR*.

plan-based reasoning strategies that go beyond static pipelines, breaking the sequential structure of the traditional CBR cycle, and overcome the rigidity of static RAG pipelines in favor of dynamic, goal-directed orchestration of reasoning processes. A dedicated *Meta Learner* observes the ongoing reasoning process in an introspective manner. It captures reasoning experience, detects patterns, and supports the continuous improvement of orchestration strategies by feeding learned meta-knowledge back into the Orchestrator. Finally, the *Environment Layer* represents the external environment in which the problem solver operates. It provides interfaces for interacting with users (e.g., via GUI, chat, or XR interfaces), external systems (e.g., cyber-physical systems, web applications, or even humans), and input/output modalities (e.g., problem definitions, solution delivery, explanations). Problem instances are supplied by the environment and solutions are communicated back upon completion. In the following, we provide a detailed description of each component of the *EXAR* architecture.

While *EXAR* adopts an agent-oriented architecture, it differs from classical multi-agent systems (MASs). In *EXAR*, reasoning agents do not represent autonomous entities with individual goals, but instead form a coordinated, goal-aligned cognitive agency whose internal collaboration is centrally orchestrated to solve a single problem instance.

3.2 Data and Knowledge Sources

The envisioned *EXAR* architecture aims to leverage a wide range of heterogeneous knowledge sources, depending on the specific domain and problem-solving context. First, structured data such as relational databases, formal process models or plans, system configurations, architectural layouts, code, or sensor readings could serve as important inputs for reasoning. Second, unstructured data sources, including free-text documents, technical or service reports, documents with embedded tables and figures, or even audio, images and video recordings, represent valuable but often underexploited reservoirs of domain knowledge. Third, *EXAR* is intended to incorporate formally represented knowledge, such as ontologies, knowledge graphs, rule bases, or planning domain models, to support symbolic reasoning. Finally, a crucial knowledge source is the tacit expertise of human domain experts, which requires explicit elicitation and formal structuring.

To make this heterogeneous knowledge usable for structured reasoning, *EXAR* proposes a transformation process that extracts and formalizes content into a persistent, queryable representation in the LTM. This transformation involves data connectors, parsers tailored to specific formats and structures, and information extraction techniques, including those based on fine-tuned LLMs. Additionally, interactive knowledge engineering approaches, potentially LLM-supported, are foreseen to capture and formalize implicit expert knowledge. The outcome of this process is a set of KUs, designed to encapsulate diverse knowledge representations in a form that reasoning agents can effectively access and process within the *EXAR* architecture.

3.3 Long-Term Memory

The LTM in *EXAR* serves as a persistent and structured memory layer for storing all relevant forms of knowledge and experience in the form of KUs. It extends the classical notion of knowledge containers from CBR [28] by supporting a broader range of representation formats, abstraction levels, and storage technologies. KUs typically result from pre-processing external sources, but may also emerge dynamically during neuro-symbolic or meta-reasoning processes.

Each KU comprises (a) a *content component*, which may include text fragments (chunks), prompt templates, time series, RDF/OWL graphs, logical rules, planning models, similarity functions, structured cases, or reasoning traces; (b) an *index component* supporting access and retrieval, which may consist of keywords, embeddings, or semantic annotations; (c) a *confidence score* estimating its trustworthiness; and (d) *provenance information*, such as the original source, transformation history, or generation context.

The LTM supports *index construction* and persistent storage using index-specific technologies such as vector databases, triple stores, graph databases, relational databases, or case bases. It enables various *retrieval strategies*, including keyword search, lexical search, SQL, SPARQL, and similarity-based retrieval using embeddings or explicitly modeled similarity measures. *Validation* is supported through consistency checking, external fact sources, or LLM-based judging, which may contribute to generating or updating confidence scores. Additionally, *reflection* on stored content, e.g., for summarization, generalization, abstraction, or improvement of retrieval performance through similarity learning or reorganization of KUs, is envisioned. In this way, the LTM constitutes a continuously evolving knowledge base that supports all reasoning activities within the *EXAR* architecture.

3.4 Reasoning Agents

According to the introduced neuro-symbolic reasoning layer, *EXAR* conceptualizes reasoning not as a monolithic process, but as the interaction of specialized agents that operate over a shared STM. This agent-based structure enables modular, compositional, and dynamically orchestrated reasoning strategies that go beyond classical pipelines such as the CBR 4R cycle or standard RAG flows. Agents are activated by the Orchestrator and exchange reasoning artifacts via the STM, allowing for both parallel and sequential execution of reasoning steps under a unified control model. The Orchestrator is also responsible for the management of the STM, e.g., for the deletion of entries after a (sub)task is completed successfully. Each agent is intended to perform a specific reasoning function such as retrieving knowledge, transforming representations, constructing plans, generating content, or evaluating outcomes while contributing to a larger, goal-oriented reasoning plan. This modular decomposition allows for fine-grained control, reuse of reasoning capabilities across tasks, and introspection through the Meta Learner. To achieve their goals, the agents may use tools to interact with other systems, such as running inference of neural networks, invoking APIs, running symbolic planners, or using the Python interpreter.

A first group of agents, the *Analyze agents*, supports the *analysis and transformation of high-level queries or problem descriptions.* These agents decompose tasks, interpret user intent, and translate input into internal representations for downstream reasoning. They may generate subgoals, planning templates, or structured queries. *Retrieve* agents perform retrieval in the sense of CBR and RAG, using symbolic conditions, semantic similarity, embeddings, or hybrid techniques to access relevant knowledge units from the LTM. They may exploit learned similarity models, ontological constraints, LLM supported query expansion, or decomposition.

Adapt agents are responsible for transforming retrieved knowledge units into usable solutions. They realize the reuse phase in CBR and may rely on handcrafted rules, learned operators, or LLM-based adaptation patterns. These agents can operate with explicit knowledge or learn their behavior from prior traces and reflective feedback. They implement structured problem-solving methods such as automated planning, rule-based inference, or constraint propagation. They operate on formal representations like logical rules, RDF graphs, or planning domains and may complement or refine outputs from other agents. Closely related are *generate agents*, which invoke LLMs or other generative models to produce problem solutions, content fragments, summaries, or completions. Such agents may be general purpose or tailored to domain-specific tasks.

Another group of agents (the *Optimize Agents*) focuses on *prompt construction and optimization.* These agents generate or refine prompts for generative models based on task context, user input, and current STM content, possibly optimizing for efficiency, coherence, or robustness.

To ensure output quality, *EXAR* includes *verification agents*, sometimes called judge agents. They verify results using consistency checks, symbolic systems, confidence estimation, or LLM-based grading, and feed their assessments into the STM, enabling downstream validation, feedback loops, and reflection—similar to the *revise* phase of the 4R cycle. Furthermore, these traces can be used to fine-tune agents using Reinforcement Learning with Verifiable Rewards (RLVR) [19]. Finally, *act agents* manage interaction and communication with the environment. They include user-facing interfaces such as chat or GUI agents.

This agent structure enables *EXAR* to integrate diverse reasoning paradigms such as retrieval based, rule based, model driven, and generative methods into a unified architecture, guided by dynamic orchestration and continuously improved through meta-level learning.

3.5 Orchestrator and Meta Learner

The orchestration component in *EXAR* governs the reasoning process by selecting, configuring, and sequencing reasoning agents according to the task, context, and internal system state. Rather than relying on a fixed reasoning cycle, *EXAR* envisions a spectrum of orchestration strategies, ranging from static pipelines to dynamically generated reasoning plans.

At the more static end of this spectrum, the Orchestrator may follow predefined reasoning procedures, such as the classical CBR 4R cycle or standard forms of RAG, where the sequence of retrieval followed by generation is fixed. Advanced variants of RAG such as iterative, recursive, or adaptive RAG extend this principle with limited feedback or multi-step reasoning but still rely on relatively rigid control flows. Such configurations are suitable for well-understood domains and recurring problem types with minimal variance in reasoning demands.

Toward the dynamic end of the spectrum, *EXAR* anticipates more flexible orchestration based on plan generation or search. This includes reasoning workflows that are dynamically constructed depending on the current problem state, retrieved knowledge, and prior experience. Approaches such as Process-Oriented CBR (POCBR) can be employed to represent and adapt reasoning strategies explicitly, treating reasoning steps as composable process fragments. In more complex cases, the Orchestrator may apply planning techniques or utilize learned reasoning graphs to construct agent execution sequences on demand.

The *Meta Learner* supports the Orchestrator by observing and analyzing executed reasoning processes over time. Its function is to reflect on the effectiveness, efficiency, and quality of reasoning strategies and to improve future orchestration decisions accordingly. This reflection process may rely on techniques developed in POCBR to adapt process structures or may employ Reinforcement Learning (RL) to optimize agent selection and sequencing based on task outcomes and performance signals. The results of meta-level reasoning are represented as new knowledge and stored persistently in the LTM in the form of knowledge units, enabling continuous learning and improvement of reasoning control. This corresponds to the *retain* phase in CBR, enabling reflective control learning.

Together, the Orchestrator and Meta Learner enable *EXAR* to shift from static reasoning pipelines toward context-sensitive, adaptive, and experience-grounded orchestration of diverse reasoning capabilities.

3.6 Environment

The environment layer in *EXAR* provides the interface between the reasoning system and its external world. It serves as the entry and exit point for problem-solving activities by enabling interaction with users, application systems, or physical environments. The environment component is essential for embedding *EXAR* into real-world scenarios, as it supports the acquisition of problem information, the delivery of solutions, and the communication of intermediate reasoning steps.

This interface may take different forms depending on the deployment context. It can be realized as a standalone user interface for entering queries, interacting via chat, or managing reasoning sessions. It may also be integrated into existing application software, acting as a reasoning backend that receives tasks from the host system and returns results or suggestions. In more immersive or sensor-based environments, the interface could include XR-based components for visual reasoning support or a direct connection to a cyber-physical system that provides sensor data and receives control actions.

The primary function of the environment interface is to obtain all relevant information required to define a problem instance. This includes user goals, constraints, preferences, and contextual events from technical systems. The second core function is to deliver solutions, recommendations, or action plans back to the user or connected systems in a structured and actionable way.

To foster transparency and user trust, the interface should also support visualization and simulation of reasoning processes and allow explanation of proposed solutions. This enables users and systems to assess the plausibility, applicability, and consequences of alternative outcomes, thus supporting informed decision-making in complex environments.

4 Existing Work in Light of *EXAR*

Table 1 provides an overview of current work in the intersection of CBR and LLMs, interpreted in the context of our proposed *EXAR* architecture. This overview is divided into more general methodologies consisting of CBR/POCBR, RAG, and agentic reasoning (see Sect. 2) and concrete hybrid approaches combining several of these general methods. For this purpose, we conducted a literature review of papers published by the CBR research community, with a focus on the proceedings of the *International Conference on Case-Based Reasoning (ICCBR)*. In this process, we selected six papers that propose a hybrid approach that combines CBR and LLMs. These papers are interpreted in the context of the *EXAR* architecture in the accompanying table, illustrating the generality of the proposed architecture.

Table 1. Applying *EXAR* Components to Existing Approaches.

Approach	**Data & Knowledge**	**Long-Term Memory**	**Neuro-Symbolic Reasoning**	**Environment**
CBR/POCBR [1,25]	Mostly structural data	Cases	Retrieve Adapt Verify	Domain-generic
RAG [11]	Mostly unstructured text and/or images	Text chunks with Embeddings	Retrieve Generate Optimize Verify	Domain-generic
Agentic Reasoning [15,23]	Structured or unstructured data	N/A	Analyze Retrieve Adapt Generate Optimize Verify Act	Domain-generic
CBR-RAG [36]	Unstructured legal documents	Textual cases consisting of Q–A pairs	Retrieve Generate	Domain experts asking legal questions
CBR-Ren [10]	Unstructured financial documents with text, numbers, and tables	Textual cases with explanations	Retrieve Generate	Domain experts doing financial analysis
RAG for Explaining Business Process Models [26]	Structured data in form of business processes	Procedural and textual cases with process model explanations	Retrieve Generate	Domain experts needing BPM explanations
On Implementing CBR with LLMs [34]	Structured medical-triage dataset	Attribute-value case for each patient with triage decision	Retrieve Adapt Generate	Domain experts making triage decisions
Adapting Graphs with WordNet and LLMs [20]	Structured argument graphs, lexical data	Graph cases, vector features, adaptation rules	Adapt Verify	Domain experts searching for arguments
LLsiM [21]	Structured data	Preference relations, similarity configs	Retrieve Generate	Domain-generic

5 Research Challenges

Based on the analysis of existing work and our own experience in building CBR systems integrating neuro-symbolic aspects, we propose a set of open research challenges. Although by no means exhaustive, this first version touches on the most important aspects of the proposed architecture. For a more general overview of challenges related to CBR and LLMs, we refer the interested reader to Bach et al. [3].

Data and Knowledge

C1 (Multimodality)**.** How to represent and process multimodal knowledge sources (text, images, video, etc.) in a unified way?

C2 (Vectorization)**.** When to use vector representation and when to use symbolic representation? How to combine both?

C3 (Expert Knowledge)**.** How to elicit and formalize expert knowledge in a way that is usable for reasoning?

Short-Term and Long-Term Memory

C4 (Reflection)**.** How to reflect on the stored knowledge to keep it relevant and maintainable?

C5 (Conflicts)**.** How to resolve potential conflicts from different knowledge sources? Should they be merged or kept separate?

C6 (Merging)**.** How to update the LTM with new external information or learned knowledge from the STM?

Reasoning Agents

C7 (Design)**.** How to design agents that are specialized enough to be effective, but general enough to be reusable across different tasks?

C8 (Prompting)**.** How to migrate away from manual prompt engineering to programmatic prompt generation?

C9 (Cost)**.** How to balance the cost of LLM generations and agent executions with the need for high-quality results?

Orchestrator and Meta Learner

C10 (Safety)**.** How to safely execute agents in a sandboxed environment while allowing interaction with required resources?

C11 (Determinism)**.** How to deal with non-deterministic results obtained from the reasoning agents?

C12 (Coupling)**.** How strong should the agents be coupled? Should they be allowed to communicate directly or only through the STM?

C13 (Planning)**.** How to select reasoning agents and generate, store, and re-use execution plans for them in a cost-effective manner?

Environment

C14 (Interaction)**.** How to design appropriate interaction types that allow for deep integration of the environment?

C15 (Personalization)**.** How to incorporate potential needs for personalization for different users?

C16 (Explainability)**.** How to provide introspection into the reasoning process to increase user trust?

6 First Steps Towards Implementation

While *EXAR* is presented as a conceptual architecture, its realization can build upon a variety of existing frameworks and libraries that address key components of the system. This section outlines suitable technologies that provide starting points for implementing each layer of the *EXAR* architecture. As the field is expanding rapidly, the selection of libraries and tools should be understood as a starting point, not a complete overview.

The diverse data sources from the *Data and Knowledge Layer* can be parsed and structured by popular libraries such as LangChain [8] or LlamaIndex [24], which support document ingestion, text chunking, metadata extraction, and embedding-based indexing, which are essential for transforming heterogeneous data into structured knowledge units (KUs). Transformer-based neural networks are increasingly used for various pre-processing steps such as the text extraction from documents like PDFs. HuggingFace offers a variety of models accessible through the transformers library [37].

The *Long-Term Memory* in *EXAR* requires persistent and queryable storage of KUs across multiple knowledge representations. Triple stores such as Apache Jena [2] support RDF and OWL-based KUs, while relational databases such as PostgreSQL offer scalable indexing[1]. Case-based reasoning components of the long-term memory can be implemented using the CBRkit [22], which provides reusable similarity functions, case models, and retrieval mechanisms, or ProCAKE [5], which additionally supports process-oriented CBR and structural reasoning on complex case representations.

In the *Reasoning Layer*, a wide variety of reasoning agents can be realized using existing tools and libraries. Adaptation agents can rely on symbolic tools, such as AI planning systems like FastDownward [14]. Libraries like Outlines [35] can help to generate well-formatted inputs for the symbolic tools. For generative agents, the transformers library [37] or dedicated inference libraries like vLLM [18] or SGLang [40] offer a wide support of models. Prompt engineering and optimization agents can be supported by frameworks like DSPy [17], which enable prompt programming and evaluation in structured workflows. Verification agents can use tools such as VAL [16], which can verify the validity of PDDL files.

The *Orchestrator* in *EXAR* may initially rely on fixed execution templates using standard orchestration libraries such as LangChain [8], LlamaIndex [24] or CBRkit [22]. More advanced agent-based orchestration can draw from emerging multi-agent frameworks like CAMEL [23] or PydanticAI [27] which allow dynamic, goal-driven interaction between agents. Process-oriented reasoning strategies can be modeled using ProCAKEs [5] process adaptation mechanisms. For the Meta Learner, machine learning libraries such as TorchRL [6] or verl [31] can be used to learn orchestration policies based on reasoning traces, performance

[1] PostgreSQL also support vectors through extensions such as pgvector(https://github.com/pgvector/pgvector)

signals, or feedback loops. Experience and strategy knowledge resulting from meta-reasoning can be represented and stored in the LTM as KUs.

The interaction with the *Environment Layer* can be realized using a variety of interface frameworks, for example by providing humans with a chat interface or a GUI or by interacting with the world through APIs.

These existing tools offer a solid foundation for implementing the components of *EXAR*. While no single framework covers all architectural elements, their combination enables flexible and modular prototyping of experience-grounded agentic reasoning systems.

7 Summary and Outlook

In this paper, we propose *EXAR*, a unified architecture for hybrid agentic reasoning. The aim of *EXAR* is to provide the basis for a more flexible orchestration of reasoning pipelines. This is not only relevant in the context of RAG approaches, but also in CBR, where the rather static "cycle of sequential steps" [1] should be broken up for a more flexible use of the CBR methodology. *EXAR* consists of four individual layers (i.e., *Data & Knowledge*, *Long-Term Memory*, *Neuro-Symoblic Reasoning* and *Environment*) that describe how knowledge can be transferred into a long-term memory, which is in turn used by agents to perform reasoning tasks in an environment. The main module of *EXAR* is the *Neuro-Symbolic Reasoning* layer, which enables the flexible orchestrator of individual reasoning agents to solve a faced problem by using a shared short-term memory. The *Meta Learner* component enables the self-incremental learning of the agents for future problem-solving situations.

In the future, we want to provide a reference implementation for the *EXAR* architecture, which can be used by the CBR-LLM research community. As a starting point, we extended our CBRkit library [22] with a *synthesis* module consisting of built-in LLM providers and prompting functions for straightforward integration of retrieval and adaptation results with the capabilities offered by generative AI models. Our vision is to provide an integrated framework for building experience-grounded applications that is both easy-to-use and customizable enough to support a wide range of use cases. In addition, we plan an extended literature review to demonstrate that existing approaches by the CBR research community can be interpreted in the *EXAR* architecture. Finally, we aim to evolve our own CBR approaches in alignment with the *EXAR* architecture, thereby creating a foundation for integrated implementations and joint research initiatives.

Acknowledgments. The authors would like to thank all members of the research group at Trier University and the German Research Center for Artificial Intelligence for their valuable discussions and contributions to the development of the ideas presented in this paper.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Commun. **7**(1), 39–59 (1994)
2. Apache Software Foundation: Apache Jena. https://jena.apache.org/ (2021)
3. Bach, K., et al.: Case-Based Reasoning Meets Large Language Models: A Research Manifesto For Open Challenges and Research Directions. HAL Science **hal-05006761v1** (2025), https://hal.science/hal-05006761, working paper or preprint
4. Bergmann, R., Althoff, K.: Methodology for Building CBR Applications. In: Case-Based Reasoning Technology, From Foundations to Applications. Lecture Notes in Computer Science, vol. 1400, pp. 299–326. Springer (1998)
5. Bergmann, R., et al.: ProCAKE: A Process-Oriented Case-Based Reasoning Framework. In: 27th ICCBR Workshops. CEUR Workshop Proceedings, vol. 2567, pp. 156–161. CEUR-WS.org (2019)
6. Bou, A., et al.: TorchRL: A data-driven decision-making library for PyTorch. In: ICLR 2024. OpenReview.net (2024)
7. Brand, F., Malburg, L., Bergmann, R.: Large language models as knowledge engineers. In: 32nd ICCBR Workshops. CEUR Workshop Proceedings, vol. 3708, pp. 3–18. CEUR-WS.org (2024)
8. Chase, H.: LangChain (2022), https://github.com/langchain-ai/langchain
9. Dannenhauer, D., et al.: A Case-Based Reasoning Approach to Dynamic Few-Shot Prompting for Code Generation. In: ICML Workshop on LLMs and Cognition (2024)
10. Feng, B., et al.: CBR-Ren: A Case-Based Reasoning Driven Retriever-Generator Model for Hybrid Long-Form Numerical Reasoning. In: 32nd ICCBR. LNCS, vol. 14775, pp. 111–126. Springer (2024)
11. Gao, Y., et al.: Retrieval-Augmented Generation for Large Language Models: A Survey. CoRR **abs/2312.10997** (2023)
12. Gates, L., Leake, D., Wilkerson, K.: Cases Are King: A User Study of Case Presentation to Explain CBR Decisions. In: 31st ICCBR. LNCS, vol. 14141, pp. 153–168. Springer (2023)
13. Hatalis, K., Christou, D., Kondapalli, V.: Review of Case-Based Reasoning for LLM Agents: Theoretical Foundations, Architectural Components, and Cognitive Integration. CoRR **abs/2504.06943** (2025)
14. Helmert, M.: The Fast Downward Planning System. J. Artif. Intell. Res. **26**, 191–246 (2006)
15. Hong, S., et al.: MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In: ICLR 2024. OpenReview.net (2024)
16. Howey, R., Long, D., Fox, M.: VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning Using PDDL. In: ICTAI 2004. pp. 294–301. IEEE Computer Society (2004)
17. Khattab, O., et al.: DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. CoRR **abs/2310.03714** (2023)
18. Kwon, W., et al.: Efficient Memory Management for Large Language Model Serving with PagedAttention. In: SOSP 2023. pp. 611–626. ACM (2023)
19. Lambert, N., et al.: TULU 3: Pushing Frontiers in Open Language Model Post-Training. CoRR **abs/2411.15124** (2024)
20. Lenz, M., Bergmann, R.: Case-Based Adaptation of Argument Graphs with WordNet and Large Language Models. In: 31st ICCBR. LNCS, vol. 14141, pp. 263–278. Springer (2023)

21. Lenz, M., Hoffmann, M., Bergmann, R.: LLsiM: Large Language Models for Similarity Assessment in Case-Based Reasoning . In: 33rd ICCBR. LNCS, Springer (2025), Accepted for Publication.
22. Lenz, M., Malburg, L., Bergmann, R.: CBRkit: An Intuitive Case-Based Reasoning Toolkit for Python. In: 32nd ICCBR. LNCS, vol. 14775, pp. 289–304. Springer (2024)
23. Li, G., et al.: CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society . In: NeurIPS 2023 (2023)
24. Liu, J.: LlamaIndex (2022), https://github.com/run-llama/llama_index
25. Minor, M., Bergmann, R., Görg, S.: Case-based adaptation of workflows. Inf. Syst. **40**, 142–152 (2014)
26. Minor, M., Kaucher, E.: Retrieval augmented generation with llms for explaining business process models. In: 32nd ICCBR. Lecture Notes in Computer Science, vol. 14775, pp. 175–190. Springer (2024)
27. Pydantic: PydanticAI (2024), https://github.com/pydantic/pydantic-ai
28. Richter, M.M.: Knowledge Containers. In: Readings in Case-Based Reasoning. Morgan Kaufmann Publishers (2003)
29. Sen, A., et al.: Counterfactual-Based Synthetic Case Generation. In: 32nd ICCBR. LNCS, vol. 14775, pp. 388–403. Springer (2024)
30. Shen, Y., et al.: HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face . In: NeurIPS 2023 (2023)
31. Sheng, G., et al.: HybridFlow: A Flexible and Efficient RLHF Framework. In: EuroSys 2025. pp. 1279–1297. ACM (2025)
32. Sourati, Z., Ilievski, F., Sandlin, H., et al.: Case-Based Reasoning with Language Models for Classification of Logical Fallacies. In: IJCAI 2023. pp. 5188–5196. ijcai.org (2023)
33. Watson, I.: A Case-Based Persistent Memory for a Large Language Model. In: 32nd ICCBR Workshops. CEUR Workshop Proceedings, vol. 3708, pp. 19–25. CEUR-WS.org (2024)
34. Wilkerson, K., Leake, D.: On Implementing Case-Based Reasoning with Large Language Models. In: 32nd ICCBR. LNCS, vol. 14775, pp. 404–417. Springer (2024)
35. Willard, B.T., Louf, R.: Efficient guided generation for large language models. CoRR **abs/2307.09702** (2023)
36. Wiratunga, N., et al.: CBR-RAG: Case-Based Reasoning for Retrieval Augmented Generation in LLMs for Legal Question Answering . In: 32nd ICCBR. LNCS, vol. 14775, pp. 445–460. Springer (2024)
37. Wolf, T., et al.: Transformers: State-of-the-Art Natural Language Processing. In: EMNLP 2020 - Demos. pp. 38–45. ACL (2020)
38. Yang, J., et al.: SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering. In: NeurIPS 2024 (2024)
39. Zhao, W.X., et al.: A Survey of Large Language Models. CoRR **abs/2303.18223** (2023)
40. Zheng, L., et al.: SGLang: Efficient Execution of Structured Language Model Programs. In: NeurIPS 2024 (2024)