

A Conceptual Architectural Design for Symbolic-Modular Cognitive Systems Assisted by Large Language Models

Edervaldo Jose de Souza Melo
Independent Researcher
Campo Grande, MS, Brazil
ORCID: 0009-0003-6835-135X
email: edersouzamelo@gmail.com

Abstract—This paper presents a conceptual architectural design for symbolic-modular cognitive systems assisted by large language models (LLMs). The contribution focuses on architectural clarity prior to implementation, articulating system structure before algorithmic or empirical commitments. The proposed Symbolic-Modular Cognitive System (SCM) is described in terms of its core inputs, internal states, and operational cycles that structure reflexive human–AI interaction. Symbolic modularity is formalized as a configuration of functional roles and semantic topologies, distinguishing these constructs from autonomous agents, memory representations, or algorithmic cognitive subsystems. The architecture is intentionally human-in-the-loop and positions LLMs as linguistic engines operating under symbolic and structural constraints. The contribution of this work lies in providing a technically precise conceptual blueprint that can inform future implementations while preserving clear boundaries between symbolic control, linguistic generation, and human cognition.

Index Terms—symbolic-modular systems, conceptual architecture, human-in-the-loop interaction, large language models, system design

I. INTRODUCTION

Recent advances in large language models (LLMs) have enabled the rapid emergence of hybrid human–AI systems that integrate natural language generation with complex human-centered workflows [1], [5]. These systems are increasingly used for reasoning support, decision assistance, planning, and reflective interaction. Despite their practical success, most contemporary applications rely on ad hoc prompt engineering or loosely coupled agent frameworks, often lacking an explicit architectural design that distinguishes linguistic generation from higher-level cognitive organization.

This absence of architectural clarity creates ambiguity at both technical and conceptual levels. From an engineering perspective, it becomes difficult to reason about system boundaries, control structures, and failure modes when symbolic organization, human intent, and language generation are conflated. From a design standpoint, the lack of pre-implementation architecture encourages premature comparisons with autonomous agents or artificial cognitive systems, even when such claims are neither intended nor justified.

In response to this gap, this paper adopts a design-first perspective and argues for the value of explicit conceptual architecture prior to implementation [3], [4]. Rather than presenting a functional system or empirical evaluation, the focus is on formalizing how a symbolic-modular cognitive system assisted by LLMs could be structured at an abstract level. This approach aligns with established engineering practices in which conceptual and architectural design precede detailed implementation, enabling clearer reasoning about system roles, interfaces, and limitations.

The proposed Symbolic-Modular Cognitive System (SCM) is framed as a human-in-the-loop architecture in which LLMs operate as constrained linguistic engines rather than autonomous cognitive entities. Symbolic modularity is treated as a configuration of functional roles and semantic topologies that organize interaction, reflection, and feedback at an architectural level. By explicitly separating symbolic control, linguistic generation, and human cognition, the architecture provides a disciplined reference model for structuring reflexive human–AI interaction and for reasoning about system boundaries prior to implementation.

The contribution of this work is therefore architectural rather than algorithmic. By articulating inputs, internal states, operational cycles, and design patterns at a conceptual level, the paper provides a technically precise blueprint that can inform future implementations while maintaining clear and defensible boundaries. This positioning is intended to support disciplined system design, enable more meaningful technical critique, and allow incremental development without narrative pressure toward premature claims of cognitive autonomy.

II. CONCEPTUAL POSITIONING

The conceptual positioning of the proposed architecture is guided by a clear separation between linguistic capability, symbolic organization, and human cognition. Large language models (LLMs) are treated as powerful language generation and transformation engines, but not as cognitive systems in themselves. The architecture deliberately avoids attributing agency, intentionality, or internal cognitive states to LLMs,

positioning them instead as components operating under externally defined symbolic and structural constraints.

This work differentiates symbolic-modular cognitive systems from both agent-based frameworks and formal cognitive architectures. In contrast to multi-agent systems—where autonomous agents pursue goals, exchange messages, and execute plans—the proposed architecture models personas as functional roles within a symbolic configuration. These roles do not possess autonomy, memory persistence, or decision authority independent of the human user. Their function is to structure interaction, reflection, and perspective-taking within a controlled human-in-the-loop process.

Similarly, the notion of symbolic modularity is not equated with computational memory systems such as vector databases or embedding-based retrieval. While such mechanisms may be employed in future implementations, the architecture conceptualizes symbolic modules as semantic topologies that organize meaning, context, and functional differentiation. These topologies describe how information is framed and interpreted rather than how it is stored or retrieved at a computational level. This distinction is critical to prevent conflation between architectural intent and implementation detail.

From an engineering standpoint, the proposed positioning emphasizes architecture as an organizing discipline rather than a claim of functional completeness [4]. The SCM is neither a formal cognitive model nor an artificial intelligence system intended to replicate human reasoning. Instead, it is a structured interaction framework that coordinates human intent, symbolic configuration, and linguistic output. The human user remains the sole locus of agency, judgment, and responsibility throughout the system’s operation.

By adopting this conceptual stance, the architecture avoids premature alignment with contested domains such as artificial general intelligence, cognitive science modeling, or psychological theory. This positioning enables more precise technical discussion, clearer boundary-setting, and a reduction of interpretive ambiguity. It also supports incremental development by allowing implementation choices to evolve independently from the conceptual framework, provided that the core architectural distinctions are preserved.

III. SYSTEM OVERVIEW

The Symbolic-Modular Cognitive System (SCM) is conceived as an abstract, human-in-the-loop architectural framework that structures reflexive interaction between a human user and large language models (LLMs). At a high level, the system is organized around the coordination of three elements: human intent, symbolic configuration, and linguistic generation. The architecture does not prescribe a specific implementation stack, algorithm, or execution environment, focusing instead on the functional relationships between these elements as a conceptual reference architecture defined prior to implementation.

Figure 1 illustrates the high-level conceptual organization of the SCM, highlighting the separation between symbolic

control, architectural configuration, and LLM-based linguistic generation.

In the SCM, the human user occupies a central and irreducible role. All system activity is initiated, configured, and interpreted by the user, who remains responsible for goal setting, evaluative judgment, and contextual grounding. The system does not operate autonomously and does not persist independent objectives, internal drives, or self-directed behaviors. This human-centered design is a defining characteristic of the architecture and a deliberate constraint.

At an abstract level, the system can be described as comprising three primary layers. The first layer consists of inputs, which include natural language expressions, contextual cues, and reflective states provided explicitly by the user. These inputs establish both the immediate task context and the symbolic framing within which interaction occurs. The second layer corresponds to internal states, representing the active configuration of symbolic modules and semantic topologies that shape interpretation and response. The third layer encompasses operational cycles, through which input, configuration, and linguistic output are iteratively coordinated.

LLMs are situated within this architecture as linguistic processing components. Their function is to generate, transform, or articulate language based on prompts and constraints derived from the current symbolic configuration. They do not maintain system state, enforce control logic, or determine architectural transitions. All structural decisions—such as which symbolic modules are active or how interaction is framed—are external to the LLM and governed by the system’s conceptual design and user control.

Figure-level abstraction of the SCM can therefore be understood as a closed-loop interaction process: user-provided inputs inform symbolic configuration; symbolic configuration constrains linguistic generation; generated language is interpreted by the user; and user interpretation feeds back into subsequent configuration. This loop enables reflective iteration without collapsing symbolic organization into language generation or attributing cognitive authority to the underlying model.

By presenting this high-level overview, the SCM establishes a clear architectural identity independent of implementation detail. This overview serves as a foundation for the subsequent specification of components, states, cycles, and design patterns, while preserving the central architectural principles of human agency, symbolic modularity, and constrained linguistic assistance.

IV. ARCHITECTURAL COMPONENTS

The architecture of the Symbolic-Modular Cognitive System (SCM) is defined through a small set of core components that together structure human–AI interaction at a conceptual level. These components are organized into inputs, internal states, and operational cycles, providing a clear and minimal architectural decomposition without prescribing specific computational implementations.

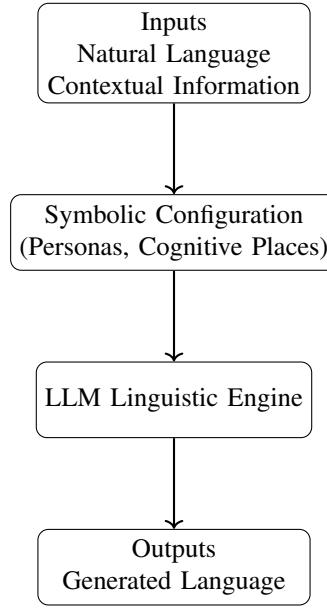


Figure 1. High-level conceptual architecture of the Symbolic-Modular Cognitive System (SCM).

A. Inputs

Inputs to the SCM originate exclusively from the human user and define both the immediate interaction context and the symbolic framing of the system. These inputs include natural language expressions, such as queries, prompts, or reflective statements, as well as contextual cues that specify intent, scope, or constraints. In addition, the architecture recognizes the user’s reflective state as an implicit input, representing how the user frames the interaction in terms of purpose, perspective, or analytical stance.

Importantly, inputs are not limited to task-oriented instructions. They may also encode meta-level guidance, such as requests for alternative viewpoints, structured reflection, or reconfiguration of symbolic roles. The architecture does not infer or estimate user state autonomously; all relevant information is provided explicitly through interaction.

B. Internal States

Internal states represent the active symbolic configuration of the system at a given moment. These states are defined by the selection and arrangement of symbolic modules, such as functional roles or semantic topologies, that shape how inputs are interpreted and how linguistic output is generated. Internal states are conceptual constructs rather than data structures and do not imply persistent memory, autonomous adaptation, or hidden system variables.

State transitions occur only through explicit user action or deliberate reconfiguration within the interaction loop. The system does not modify its own structure or activate symbolic modules independently. This constraint ensures that symbolic organization remains transparent, inspectable, and under human control.

C. Operational Cycles

The SCM operates through iterative cycles that coordinate input, state configuration, and linguistic output. Each cycle begins with user-provided input, which informs or updates the current symbolic configuration. This configuration then constrains the generation of linguistic output by the LLM. The generated output is presented to the user, who interprets it and determines subsequent actions, including whether to maintain, adjust, or replace the current symbolic state.

These cycles support reflexive interaction by enabling feedback without internal autonomy. Reflection arises from the user’s interpretation of output within the symbolic frame they have configured, not from internal system reasoning. As such, the operational cycle reinforces the system’s role as an assistive structure rather than an independent cognitive agent.

By articulating these components and their relationships, the architecture provides a clear framework for reasoning about system behavior, boundaries, and potential extensions. This decomposition also facilitates future implementation by allowing concrete design choices to be mapped onto well-defined conceptual roles without altering the core architectural intent.

V. DESIGN PATTERNS

The conceptual architecture of the Symbolic-Modular Cognitive System (SCM) is informed by a set of design patterns that guide how symbolic structure, linguistic generation, and human control are combined. These patterns define a coherent architectural style that preserves clarity of roles, maintains externalized control, and ensures a disciplined separation between symbolic organization and linguistic generation.

A. Functional Roles Instead of Autonomous Agents

A central design pattern of the SCM is the treatment of personas as functional roles rather than autonomous agents. Unlike agent-based architectures, where agents maintain internal state, pursue goals, and interact through message passing, symbolic modules in the SCM do not exhibit autonomy or persistence. They are activated, combined, or deactivated solely through user configuration.

Each role represents a specific perspective, constraint, or interpretive stance applied to linguistic generation. Roles do not initiate actions, store memory independently, or resolve conflicts among themselves. This pattern avoids attributing agency or intentionality to system components and reinforces the human user as the sole decision-making entity.

B. Semantic Topologies Instead of Computational Memory

Another key pattern is the distinction between semantic topologies and computational memory mechanisms. In the SCM, symbolic organization is conceptualized as a structured space of meaning that shapes interpretation and expression, rather than as a repository for data storage or retrieval. This contrasts with architectures that rely on vector databases or embedding-based memory as primary organizing principles.

While such memory mechanisms may be compatible with future implementations, they are not intrinsic to the conceptual architecture. The semantic topology pattern emphasizes how information is framed and related at a meaning level, independent of how it may be indexed or stored computationally.

C. Externalized Control and Transparency

Control logic in the SCM is fully externalized and transparent. All configuration decisions—such as which symbolic roles are active or how interaction is framed—are visible to and controlled by the user. There are no hidden adaptive mechanisms, self-modifying structures, or opaque decision layers within the architecture.

This pattern supports inspectability and accountability, enabling users to understand how outputs are produced and to adjust system behavior deliberately. It also reduces the risk of unintended emergent behavior associated with autonomous adaptation.

D. Constrained Linguistic Generation

LLMs are integrated according to a constrained generation pattern. Rather than acting as central reasoning entities, they function as linguistic engines whose output is shaped by symbolic configuration and user intent. Constraints are applied externally, through prompt structure and interaction design, rather than through internal modification of the model.

This pattern ensures that linguistic generation remains subordinate to symbolic organization and human judgment. It also clarifies the boundary between language production and higher-level cognitive framing.

Collectively, these design patterns define a coherent architectural approach that prioritizes role clarity, human agency, and conceptual separation of concerns. By articulating these

patterns explicitly, the SCM avoids conflation with agent-based systems, autonomous cognitive architectures, or memory-centric designs, while remaining compatible with a wide range of future implementation strategies.

VI. LIMITATIONS AND NON-GOALS

The Symbolic-Modular Cognitive System (SCM) is intentionally constrained in scope. This section explicitly states what the proposed architecture does not aim to achieve, in order to prevent misinterpretation, overextension, or inappropriate comparison with unrelated systems.

First, the SCM does not claim to implement artificial general intelligence, artificial cognition, or any form of autonomous reasoning. The architecture does not model cognition computationally, nor does it attempt to replicate human mental processes. All cognitive activity, interpretation, and judgment remain exclusively with the human user.

Second, the SCM is not a formal cognitive architecture in the sense used in cognitive science or artificial intelligence research. It does not provide mechanisms for perception, learning, memory consolidation, planning, or decision-making as computational processes. The use of terms such as “cognitive” and “symbolic” reflects an interaction-level organizational perspective rather than a claim about internal cognitive modeling.

Third, the system does not perform psychological inference, diagnosis, or assessment. It does not infer mental states, emotions, intentions, or traits from user input, nor does it adapt its structure based on inferred psychological variables. As such, the architecture has no clinical, therapeutic, or diagnostic applicability and should not be interpreted as a mental health or behavioral assessment system.

Fourth, the SCM does not operate autonomously. It does not initiate actions, maintain independent goals, or modify its own symbolic configuration without explicit user intervention. There is no self-directed learning, optimization, or adaptation within the architecture. Any change in system configuration is the result of deliberate human choice.

Finally, the architecture does not define performance metrics, benchmarks, or evaluation criteria. Since the contribution is architectural rather than algorithmic, no claims are made regarding efficiency, effectiveness, accuracy, or comparative advantage over existing systems. Questions of empirical validation and quantitative evaluation are explicitly deferred to future implementation-specific work.

By articulating these limitations and non-goals, the SCM establishes clear conceptual boundaries that support responsible interpretation and disciplined system design. These constraints are not shortcomings but defining characteristics of the architecture, ensuring that it is understood as a conceptual blueprint rather than a deployed or validated system.

VII. DISCUSSION

The conceptual architectural design presented in this work positions the Symbolic-Modular Cognitive System (SCM) as a blueprint rather than a finished system. This positioning has several implications for how the architecture should be

interpreted, critiqued, and potentially extended. By remaining at the architectural level, the SCM enables technical discussion focused on structure, boundaries, and roles, without conflating these concerns with implementation-specific performance or empirical claims.

One implication of this approach is that multiple implementation strategies may be compatible with the same conceptual architecture. Different programming languages, orchestration frameworks, storage mechanisms, or interface designs could be employed without altering the core architectural distinctions between symbolic configuration, linguistic generation, and human control. This flexibility is particularly relevant in the current landscape of rapidly evolving LLM-based systems, where implementation practices often outpace architectural clarity.

Another implication concerns comparison with existing systems. Because the SCM is not an agent-based architecture, a memory-centric retrieval system, or a formal cognitive model, direct comparisons along performance or autonomy dimensions are not meaningful at this stage. Instead, the architecture should be evaluated in terms of clarity, coherence, and its ability to support disciplined system design. In this sense, the primary contribution is not functional superiority but conceptual differentiation.

The explicit separation of symbolic roles from computational agents also has practical consequences for system safety and interpretability. By externalizing control and maintaining the human user as the sole locus of agency, the architecture reduces the risk of unintended autonomous behavior and simplifies accountability. This design choice aligns with broader engineering concerns related to transparency, controllability, and responsible human–AI interaction.

Finally, the SCM highlights the value of architectural articulation in emerging domains where implementation practices often outpace conceptual clarity. As large language models continue to be integrated into increasingly complex workflows, the absence of clear architectural framing can lead to misaligned expectations and fragile designs. By foregrounding architecture before code, the SCM contributes a reference model that can guide both technical development and critical evaluation.

Overall, this discussion underscores that the significance of the proposed architecture lies not in immediate deployment but in its role as a stable conceptual foundation. Future work can build upon this foundation by exploring implementation choices, tooling, and evaluation methods, provided that the architectural boundaries and assumptions articulated here are respected.

VIII. CONCLUSION

This paper has presented a conceptual architectural design for symbolic-modular cognitive systems assisted by large language models. The contribution is deliberately architectural rather than algorithmic, focusing on the explicit articulation of system structure, roles, and boundaries prior to implementation. By defining inputs, internal states, operational cycles,

and design patterns at a conceptual level, the work establishes a coherent framework for structuring reflexive human–AI interaction without overextending technical claims.

A central design principle of the proposed architecture is the preservation of human agency. The system is explicitly human-in-the-loop, with symbolic configuration, control, and interpretive authority remaining external to the language model. LLMs are positioned as constrained linguistic engines rather than autonomous cognitive entities, and symbolic modularity is treated as an organizational mechanism rather than a computational memory or reasoning system.

The explicit articulation of limitations and non-goals further clarifies the intended scope of the architecture. The SCM does not claim artificial general intelligence, formal cognitive modeling, psychological inference, or autonomous operation. These constraints are not incidental but foundational, ensuring that the architecture can be discussed, critiqued, and extended without ambiguity or misplaced expectation.

By offering a technically precise conceptual blueprint, this work contributes to the broader engineering discourse on how complex human–AI systems may be designed responsibly and transparently. The architecture provides a stable reference point for future implementations and evaluations, while allowing development to proceed incrementally and without narrative pressure. In this sense, the SCM serves as a design foundation upon which concrete systems may be built, assessed, and refined as the field continues to evolve.

REFERENCES

- [1] M. Holzinger, “Human-in-the-loop machine learning,” *Computer*, vol. 49, no. 9, pp. 48–54, 2016.
- [2] T. B. Brown et al., “Language models are few-shot learners,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.
- [3] A. Hevner, S. March, J. Park, and S. Ram, “Design science in information systems research,” *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.
- [4] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Boston, MA, USA: Addison-Wesley, 2013.
- [5] B. Shneiderman, “Human-centered artificial intelligence: Reliable, safe and trustworthy,” *International Journal of Human–Computer Interaction*, vol. 36, no. 6, pp. 495–504, 2020.