Cognitive Exoskeleton for Reasoning, Execution, and Strategy (CERES)

Architecture Overview

Federico Formentini, Programme Manager in Governance (Justice, Digitalisation, Digital Infrastructure), Independent Researcher, 0009-0003-9401-797

Carlo Burelli, Associate Professor, Università del Piemonte Orientale, https://orcid.org/0000-0001-7012-2669

Abstract

As large language models become increasingly agentic, the need for structured human–machine collaboration is more urgent than ever. While "human-in-the-loop" is widely endorsed, it often lacks architectural substance. This paper introduces CERES (Cognitive Exoskeleton for Reasoning, Execution, and Strategy), a domain-agnostic system designed to integrate human judgment with AI-powered reasoning, planning, and execution. Uniquely, CERES departs from conventional flat tool orchestration frameworks by implementing a vertically layered hierarchy—from Governance and Operational strategy down to Methodology, Technical Interface, and Content/Execution. Each layer runs its own control loop while remaining tightly coupled to the others through doctrine, telemetry, and feedback. Drawing on Cognitive Systems Engineering, control theory, modular agent design, and knowledge management, CERES offers a principled architecture for augmenting human cognition at multiple levels of abstraction. It treats reflection, adaptation, and traceability as first-class features, enabling not just automation but true thinking-for-thinking. The result is a scalable, auditable, and strategically aligned cognitive scaffold for high-stakes decision environments.

Keywords: cognitive exoskeleton, human-AI teaming, layered control, CSE, agentic AI

1 Introduction

Large language models can generate fluent text but often struggle with long term, multi-step planning and domain awareness [1]. Recent advances in reasoning-augmented prompting and tool-integrated agents (e.g., ReAct [2], Toolformer [3], AutoGen [4]) have improved performance on structured tasks, but these systems still fall short when applied to real-world organisational contexts. However, as long as AGI remains a distant prospect [4], automation alone cannot handle the legal and moral responsibilities, nor the nuanced, context-dependent decisions, that organisational life demands. Hence many advocated for Human-in-the-loop (HITL) systems, to keep people actively involved in AI-mediated workflows, embedding oversight and accountability to mitigate hallucinations, bias and context loss [5], [6].

However, the problem is that HITL too often remains an empty slogan, leaving individual users to improvise without systematic guidance for ensuring both effective AI functionality and meaningful human control. This paper aims to fill that gap.

We aim to provide clear, actionable guidance for a system-agnostic "cognitive exoskeleton" [7], a framework designed to integrate human judgment with AI-driven data gathering, reasoning, and decision-making. We call this architecture CERES: Cognitive Exoskeleton for Reasoning, Execution, and Strategy. Like the Roman goddess of agriculture from whom it takes its name, CERES is intended to cultivate human decision-making and yield more coherent, informed, and productive outcomes.

Collectively, the Cognitive Exoskeleton for Reasoning, Execution and Strategy (CERES) offers a scalable architecture that couples human judgment with agentic AI to close the gap between high-level intent and reliable, auditable action. Its five nested layers—Governance, Operational, Methodology, Technical Interface, and Content/Execution—form a continuous sense-model-decide-act loop in which each stratum both constrains and enriches the one below. Governance establishes ethical and strategic boundaries; the Operational layer translates those boundaries into mission objectives and performance metrics; Methodology selects and adapts reasoning templates; the Technical Interface brokers data, models, and agents; and the Content/Execution layer carries out concrete tasks while emitting telemetry for real-time feedback. By treating reflection and adaptation as first-class functions, CERES maintains transparency, traceability, and participatory control even as underlying models evolve. The result is a "human-in-the-loop" cognitive scaffold that accelerates expert workflows, reduces error propagation, and continuously learns from both successes and failures—ultimately enabling organizations to navigate complex, dynamic problem spaces with greater speed and confidence.

In developing CERES, this paper draws from a broad set of disciplinary traditions. Its foundations lie in Cognitive Systems Engineering, which provides key principles for structuring effective human–machine collaboration, such as joint cognitive systems, adaptive control, and adjustable autonomy. At the same time, CERES departs from prevailing current human/AI interaction architectures by introducing a vertically layered hierarchy for the human/AI system as such. The practice of cascading intent through hierarchical layers originates in military theory, where Clausewitz distinguished tactical actions from strategic goals. Management science and systems theory later generalised this logic: means–ends hierarchies help bounded actors decompose complexity, while near-decomposable subsystems enable scalable control. Contemporary AI planning continues this lineage through hierarchical task networks and option-based reinforcement learning, offering a cross-disciplinary rationale for CERES's vertically layered architecture. Beyond these anchors, the design is shaped by insights from human factors, cybernetics, knowledge engineering, and emerging debates in AI governance and ethics. Together, these fields inform CERES's core commitments to transparency, traceability, and organisational learning, positioning it not as a narrow tool orchestration framework, but as a full cognitive architecture for high-stakes, multi-scale decision support.

The paper proceeds as follows. Section 2 situates CERES within relevant theoretical frameworks, including human–AI teaming, layered control architectures, knowledge management, and modular agentic systems. Section 3 details the five-layer architecture of CERES, explaining the role and function of each component in the continuous decision loop. Section 4 articulates the core design principles that guide the system's construction. Section 5 provides an illustrative use case to demonstrate the architecture in action, and Section 6 concludes with a summary of contributions and directions for future development.

2 Theoretical Framework

2.1 Human-AI Teaming

Human-AI teaming frames people and artificial agents as co-actors that pool complementary strengths—situational awareness, contextual judgment, pattern search and relentless execution—inside a single "joint cognitive system." [8] Early cognitive-systems work argued that effectiveness depends on how well the ensemble, not either component alone, closes the loop between perception, decision and action [9]..

In practice this demands a continual negotiation of authority, initiative and trust, rather than static hand-offs: the agent should step forward when speed or scale dominate, and yield when nuance or moral accountability matter.

Large surveys of human-in-the-loop machine-learning echo the point [5]. They show that performance and acceptance rise when systems (i) surface their internal state, (ii) accept iterative human guidance, and (iii) calibrate autonomy levels to the operator's expertise and task criticality.

At the interaction level, transparency and explainability foster calibrated trust; at the governance level, adjustable autonomy—the ability to ratchet control up or down on demand—prevents over-reliance and sharpens accountability chains. Hollnagel's First Law of Collaborative Systems crystallises the design imperative: "it's not collaboration, if either you do it all or I do it all".

Design implication for CERES.

Grounding the interaction model in Cognitive Systems Engineering —thus treating humans and automation as a single joint cognitive unit— implies that the human–AI team itself must manage a clearly articulated hierarchy of goals. CERES therefore equips the system with its own strategic, operational, and tactical objective stack, giving both human stewards and agentic components a shared frame of reference for maintaining aligned mental models and adjustable autonomy across situations. This vertically structured goal set is curated by the Governance layer and propagated downward so that every subsequent sense-model-decide-act cycle remains anchored to the same evolving intent. The system thus anchors a dedicated Governance layer that enables human operators to set —and dynamically revise— doctrines, strategic objectives, delegation policies, operational methodologies/knowledge frameworks and acceptable risk thresholds. Rather than hard-coding fixed authority levels, CERES supports flexible negotiation and escalation through natural-language planning dialogues, ensuring that strategic intent, rationale, and objection handling remains mutually intelligible and persistent between work sessions. Each decision path is logged for post-hoc review, fostering traceability and calibrated trust across successive sense–model–decide–act cycles. In this way, CERES operationalises the principles of human–AI teaming by treating shared identity, objectives, and mental models not as auxiliary features, but as cross-cutting architectural requirements.

2.2 Layered Control Loops and Feedback Architectures

Across a range of disciplines—from military strategy to software engineering and systems design—a recurrent insight emerges: intelligent action in dynamic environments requires continuous closed-loop control. The basic cycle is deceptively simple: sense the environment, update an internal representation, select and execute an action, and assess its results to inform the next cycle. But how this loop is structured—and especially how multiple such loops are coordinated—determines whether a system remains responsive, robust, and aligned with higher-order goals.

One of the earliest influential models of such control is Boyd's OODA loop (Observe–Orient–Decide–Act), developed to explain tactical advantage in aerial combat [10] . Boyd argued that the speed and adaptability with which an agent moves through this loop—especially its ability to revise its orientation—determine whether it can "get inside" the opponent's decision cycle and create cascading disruptions. OODA remains foundational not only in military strategy but also in agile decision-making paradigms, where rapid and recursive sense-making is essential to navigating complex, adversarial, or fast-changing environments.

A similar logic informs MAPE-K, the Monitor–Analyze–Plan–Execute–Knowledge architecture introduced in IBM's autonomic computing vision [11] . Here, the emphasis is not on adversarial speed, but on self-regulation in the face of environmental volatility. Each loop is governed by a shared knowledge base that ensures consistency and traceability across cycles. Comparable models can be found in the PDCA) loop and in System-of-Systems Engineering, where multiple semi-autonomous units must coordinate through nested feedback structures to maintain coherence at the enterprise level[12].

What unites these approaches is the recognition that effective control must be both layered and reflexive. Fast, local loops allow subsystems to respond swiftly to proximal stimuli; slower, global loops provide strategic oversight and ensure long-term mission alignment. Yet this layering introduces perennial tensions. Speed may come at the expense of assurance; local optimisation may drift from collective priorities; and machine autonomy may clash with human accountability. These trade-offs are not merely technical—they are deeply normative, especially when system decisions affect public goods, institutional trust, or individual rights.

Yet closed-loop control alone is insufficient; those loops must be embedded in a vertical hierarchy of objectives that ties fast tactical reactions to slower operational campaigns and overarching strategic intent. Classical military theory first formalised this distinction: Clausewitz drew the line between "the use of armed forces in the engagement" (tactics) and "the use of engagements for the object of the war" [13]. Mid-twentieth-century organisation scholars generalised the idea—March and Simon showed that bounded-rational actors tame complexity by decomposing grand aims into means–ends chains, thereby reducing individual cognitive load and synchronising attention to the relevant level of context [14] . Systems theory offers the underlying logic: Herbert Simon's principle of near-decomposability explains how hierarchical layers insulate high-frequency local dynamics from low-frequency global dynamics, letting each controller focus on a manageable slice of variety [15]. Cybernetics extends the argument—Stafford Beer's Viable System Model shows that nested controllers, each with its own feedback loop, collectively meet Ashby's law while preserving enterprise coherence [16]. Contemporary AI planning adopts the same principle through Hierarchical Task Network methods that recursively expand goals into executable subtasks, allowing agentic components to act at the appropriate level of abstraction [17]. Coupling these vertical means–ends chains with horizontal feedback cycles lets fast tactical loops adapt fluidly while remaining subordinated to slower operational and strategic cycles, balancing agility with assurance and keeping human and AI elements jointly attuned to the task and situation at hand.

Design implication for CERES.

The CERES architecture addresses these tensions by embedding a hierarchy of interlocking control loops across its five layers. Each layer—Governance, Operational, Methodology, Technical Interface, and Content/Execution—runs its own internal sense–model–decide–act cycle, tuned to the time-scale and abstraction level appropriate to its function. Higher layers impose constraints and performance criteria on the layers beneath; lower layers emit telemetry and exception signals upward, triggering recalibration when needed. In doing so, CERES directly operationalises key insights from OODA and MAPE-K: it maintains continuous feedback, enforces knowledge stewardship, and ensures that machine-level adaptations remain transparent, revisable, and tethered to human intent. It thereby offers not just an architecture for action, but a framework for sustained human oversight in evolving operational contexts.

Closing the feedback loop is impossible without a living organisational memory: raw telemetry must be distilled into structured knowledge that can be consulted, challenged, and revised across successive cycles. Classic studies of organisational learning describe this as a continual conversion of tacit insights into explicit artefacts and back again [18], [19] and warn that true adaptation demands double-loop review of the governing assumptions themselves [20] . In agentic systems the same principle holds, but the memory layer must now be machine-navigable, versioned, and auditable in real time.

CERES therefore anchors its nested control loops to a versioned semantic knowledge graph that records every doctrine, plan, metric, and observation together with full provenance metadata. The graph functions as the "spine" that threads through all five layers: lower-level loops write new facts and confidence scores, while higher-level loops query lineage to diagnose drift or trigger policy revision. Provenance fields—source links, timestamps, and revision history—follow current best practice in knowledge-graph quality control [21], ensuring that rapid machine adaptations remain transparent and traceable to human intent. In short, CERES couples reflexive control with a reflexive memory, turning auditability and organisational learning into architectural, not incidental, features.

2.3 Modular Agentic Architectures & Cognitive Exoskeletons

Artificial-intelligence practice has swung, across decades, between monolithic problem-solvers [22] and modular, loosely coupled agent societies [23]. The contemporary resurgence of the latter owes as much to cloud micro-services as to advances in large language models: breaking capabilities into small, specialised agents reduces brittleness and lets new tools be swapped in without retraining entire stacks [23]. Classic blackboard and BDI (Belief–Desire–Intention) architectures framed this idea conceptually [24], [25] , while more recent multi-agent toolkits—SpaDE, JADE, ROS2—demonstrated its engineering viability for distributed sensing, planning, and actuation.

The arrival of LLM-centred agents has pushed modularity further. Studies of "planner-executor" patterns show that chaining a reasoning-heavy model to domain-specific tool calls outperforms single-model approaches on complex tasks [3] . Frameworks such as LangChain [26], AutoGen [27], and AgentOS [28] now treat tools as first-class citizens, attaching each to a capability schema that the language model can query at run-time. This orchestration layer effectively turns the model into a meta-planner that decomposes goals, selects tools, checks pre- and post-conditions, and delegates execution—while maintaining a symbolic memory of what has been tried and why. Yet the literature also warns that uncontrolled tool-calling can generate cascading errors or security risks; guard-rails and fine-grained monitoring are therefore considered indispensable [29]

A recent prototype that crystallises these issues is Ling's "Brain Cache," a generative-AI cognitive exoskeleton positioned as a "second brain" for knowledge workers [30] . Brain Cache augments cognition through three intertwined mechanisms: (i) it externalises volatile memory into personalised knowledge vaults; (ii) it organises fragmented insights into evolving semantic networks; and (iii) it retrieves this structured knowledge via context-aware, dialogue-based prompts. Ling argues that most current exoskeleton tools automate atomic tasks yet neglect thinking-for-thinking because they lack mechanisms for dynamic knowledge structuring and proactive recall. Brain Cache therefore points to the value of holistic augmentation while also revealing its limits: the architecture is largely horizontal and tool-driven, leaving open questions about strategic oversight, escalation paths, and doctrinal consistency.

CERES builds directly on these insights but resolves the open questions by introducing a vertical, military-inspired hierarchy. Its Governance and Operational layers provide the strategic spine that other proposed architectures like Brain Cache lacks, ensuring that modular planning, prediction, or evaluation plug-ins remain traceable to mission-level doctrine rather than drifting into siloed optimisations.

CERES adopts a plug-in, capability-described framework at its Technical Interface and Content/Execution layers. Each external tool or micro-service is wrapped in a declarative contract that specifies inputs, outputs, side-effects, and privilege scope. The Methodology layer's reasoning templates consult this registry to compose task plans, while the Governance layer can approve, throttle, or revoke capabilities on policy grounds. By fusing human and LLM-driven planning with disciplined tool orchestration, CERES achieves the flexibility heralded by modern multi-agent research without sacrificing auditability or operational safety.

3 System Architecture

CERES consists of five layers, hierarchically ordered, that together implement a continuous sense→model→decide→act→feedback loop. Each layer encapsulates specific responsibilities and exposes clear interfaces to adjacent layers, enabling modularity and domain independence. The architecture is grounded in several theoretical foundations: joint cognitive systems, which treat humans and machines as a single adaptive cognitive unit; Rasmussen's Abstraction Hierarchy and Skill–Rule–Knowledge (SRK) framework, which ensure that doctrinal knowledge and system states and objectives are represented at appropriate levels of abstraction to reduce the cognitive workload of the system ; traditional control-cycle models such as Boyd's OODA loop and IBM's MAPE-K loop that emphasise feedback-driven adaptation; Klein's Recognition-Primed Decision (RPD) model, where experts rely on pattern recognition and mental simulation rather than exhaustive options analysis; double-loop learning, which updates underlying goals and assumptions rather than merely adjusting tactics; Nonaka's SECI model of knowledge conversion (Socialization, Externalization, Combination, Internalization) for capturing and evolving tacit knowledge. These theories collectively inform CERES's layering, interfaces and learning mechanisms. Figure 1 shows the high-level layer diagram, while Figure 2 illustrates the sense→model→decide→act→feedback loop that threads through these layers.

[Insert Table 1 here]

In line with Hollnagel's First Law of Collaborative Systems—"it's not collaboration, if either you do it all or I do it all" [31] —CERES positions the human and LLM as co-actors and in fact understands them as a single system. Decision authority and perception are shared via natural-language dialogue rather than delegated wholesale, so that plans are co-constructed and continuously aligned with the operational purposes of the human/AI system .

3.1 Governance Layer

The Governance Layer is the constitutional nucleus, providing verticality to the system. It defines the non-negotiable "rules of the game" that every downstream decision must obey. Governance sets direction by codifying doctrines (first-principle heuristics), strategic objectives (end-states that define what success looks like for the system) and operational areas (i.e. the permanent domains where structured action takes place). It provides consistency and coherence to the system's actions by encoding invariant principles, preventing local optimisations from drifting away from global intent. It also enables traceability: every plan, action or artefact must reference the doctrine and objectives that justify it. In practical terms, t his layer stores versioned doctrine lists strategic and operational higher level objectives and injects them into prompts to bias the LLM toward context and purpose aligned answers. Importantly, human stewards author and update these artefacts and review any proposed doctrinal changes.

3.2 Operational Layer

The Operational Layer translates strategic intent into where sustained effort is organised and measured. It carves the domain into Operational Areas (OAs) that are strategically complete and ideally mutually exclusive. Each OA hosts operational objectives, the first measurable targets that roll up to strategic objectives. The Operational Layer acts as a routing matrix: every structured action (operation, project, routine, improvisation) must declare which OA—and therefore which strategic objective—it advances. OAs maintain knowledge frameworks (e.g area specific mental models to recall and tools ; operational objectives are defined via reverse planning from strategic objectives and updated as new milestones emerge. Generally, i n multi-domain complex environments, the choice and naming of OAs is arbitrary and pragmatic (it must reflect the complexity that the system intends to tackle); what matters for the functional success of this layer is for each action to have a clear lineage back to top-level intent.

3.3 Methodology Layer

The Methodology Layer provides the system's structured long-term operational memory—the adaptive "how" that converts high-level objectives into executable plans while progressively refining its own mental models. At the top of this tiered store sit *knowledge frameworks*, curated bodies of theory and domain insight such as ecological interface design, antifragility, or social-capital theory, which supply first principles for judgement. Beneath them reside *methodologies*—prescriptive process models like the Military Decision-Making Process, PRINCE2, OKR cascades, or agile sprints—that translate objectives into sequenced actions. These methods call upon specialised *analytic tools* (decision matrices, Bayesian updating routines, Monte-Carlo simulations, premortem checklists) to evaluate alternatives and manage uncertainty, and they emit *lower-tier artefacts*—SOPs, SITREP templates, COA matrices, risk logs—that preserve best practice under governance authority. When invoked, the Methodology Layer retrieves situational context, applies current governance constraints, selects the most appropriate method, and orchestrates the requisite tools before passing structured guidance to the execution layer. Critically, it also absorbs lessons from feedback loops—capturing, for example, a newly validated risk-assessment heuristic or an emerging theory of stakeholder alignment—and updates its frameworks and methods over time, ensuring that each new operation benefits from an expanding, continuously refined knowledge base.

3.4 Technical Layer

The function of this layer is to enable human/AI collaboration, ideally with as little friction as possible. In general, this layer should consist of several core components: a language model stack serving as the primary reasoning engine, a project-scoped vector memory store for persistent context and retrieval-augmented generation, and a toolchain gateway that standardizes access to external functions and automation tools. It should also incorporate a file repository that handles structured storage, chunking, and indexing of user-uploaded documents, ensuring efficient lookup and integration into the reasoning workflow. The layer should also include a domain–channel routing map to direct interactions through the correct operational pathways, and support a custom instruction module for runtime parameterization.

In terms of functionalities this layer must provide long-term conversational memory, allowing the system to recall, reference, and synthesize relevant details from past sessions and uploaded files. Seamless cross-session context retention will ensure that prior discussions, artefacts, and operational parameters remain accessible and actionable throughout different work sessions. Through retrieval-augmented generation, the layer dynamically surfaces pertinent knowledge for every interaction, increasing both accuracy and relevance of responses. Toolchain integration empowers the assistant to invoke external resources—such as search, code execution, and automation—while robust file repository support enables direct interaction with user documents for tasks like question answering or data extraction. Routing and scheduling components facilitate multi-channel operations and enable the automation of periodic or event-driven actions.

Crucially, this layer incorporates the injection of governance-derived parameters and hierarchical objectives into the system's prompt composition or instruction stack. By programmatically referencing the current governance layer each interaction is contextually grounded and policy-aware. This mechanism guarantees that the assistant's responses are consistently aligned with user identity, system context, and domain-specific protocols, thereby improving the pertinence and relevance of all outputs. Such solution should also ease the cognitive workload for the LLMs having to resurface all these elements from the vector database of previous conversations. The architecture thus ensures not only technical robustness and flexibility, but also that every response is consistently tailored to the governing rules and intent underpinning the system's ultimate aims.

Although still in development, similar implementation pattern are rapidly becoming the industry standard for advanced AI assistant systems. Major platforms now routinely combine vector databases such as Pinecone, Weaviate, and Chroma for persistent memory; integrate with file management libraries like LangChain's document loaders or OpenAI's Retrieval Plugin; and use toolchain gateways built around standardized function calling or plugin registries, as seen in OpenAI's Assistants API and Microsoft Copilot's orchestration layers. The modular use of context routers and instruction stacks mirrors patterns adopted in frameworks like LlamaIndex and LangChain, which abstract vector storage, chunking, tool invocation, and multi-modal extensions for general-purpose or domain-specific AI copilots. These components—often open-source or available as cloud services—enable organizations to assemble, extend, and scale their own context-aware AI systems by following compositional best practices that are now widely accepted in both enterprise and research deployments.

3.5 Content / Execution Layer

The Content / Execution Layer is where the architecture's enabling components—such as the vector memory, file repository, toolchain gateway, and routing map—are activated in daily operations. Human–AI interaction takes place directly within this layer, as users select relevant operational channels and pose prompts or task requests. The system orchestrates the retrieval of context from persistent memory and file repositories, injects governance and methodology logic, and invokes appropriate external tools through the standardized gateway. As a result, it delivers outputs ranging from narrative guidance and structured reasoning to concrete artefacts such as situational reports, course-of-action matrices, KPI dashboards, or draft SOPs. Users can review, accept, edit, or reject these outputs, maintaining active control over all generated content.

Execution in the real world is coupled to system output, with users implementing recommended actions and subsequently reporting outcome data, key metrics, or emergent lessons. This feedback is systematically written back into the project-scoped memory, updating the assistant's knowledge base and reinforcing the closed-loop, data-driven cycle. When performance data indicates that an operational threshold has been crossed—for example, a KPI target is met—the system can recommend the revision or closure of specific objectives, always leaving final confirmation to the user.

A diverse suite of tools is accessible on demand, including on-the-fly charting, diagram generation, text-to-speech synthesis, and canvas-based drafting. Each tool invocation inherits context, policy, and doctrine parameters from the interface layer and is logged for traceability. Quality controls in the technical layer are continuously enforced, with automated consistency checks, reasoning accuracy evaluations, latency monitoring, and storage quota management, all ensuring that day-to-day operations remain reliable, efficient, and well-aligned. The same quality controls could be implemented at the level of operational performance via usual loops (e.g. OODA, MAPE-K etc., see next section) on mean/ends vertical. This continuous alignment loop —where execution feeds outcome metrics and lessons back into the system—guarantees that tactical actions consistently serve long-term strategic intent, closing the gap between strategy and lived operational reality or, at the very least, greatly improve the relevance and quality of the system.

3.6 Data & Decision Flow

At the heart of the architecture, a dynamic sense→model→decide→act→feedback sequence orchestrates information flow and adaptation across all layers, drawing on established control-cycle frameworks such as OODA and MAPE-K. These cycles operate not only horizontally—within each operational area and task—but also vertically, ensuring ongoing alignment between immediate actions and overarching strategic objectives (see above Sections 3.4 and 3.5).

[Insert Table 2 here]

Sense: Within the Operational Layer, the system continuously ingests and assembles data, constructing a live context from current states, KPIs, and external or environmental signals.

Simultaneously, the Governance Layer injects up-to-date doctrines and hierarchical objectives, providing the guiding parameters for subsequent processing.

Model: The Methodology Layer interprets this contextual data using selected knowledge frameworks and formal decision models. Here, objectives are decomposed into actionable tasks, predicted states are generated, and candidate actions are proposed, drawing on specialized modules such as state prediction or conflict monitoring. These processes are inspired by modular agentic planning paradigms and enable the system to flexibly adapt its internal representations as circumstances evolve.

Decide: In this phase, Governance evaluates available options against current doctrines, policy constraints, and risk thresholds. The Methodology Layer applies decision heuristics—such as recognition-primed models—to select an optimal course of action. Human stewards remain actively involved, reviewing, validating, and where necessary, adjusting or overriding recommendations before any plan is finalized.

Act: The Technical Interface Layer coordinates the invocation of required tools and integration with external systems, while the Content / Execution Layer generates narrative outputs or artefacts and issues operational directives to downstream actors, whether automated agents or human team members.

Feedback: Throughout execution, outcomes and performance metrics are systematically captured. Anomalies, surprises, and lessons learned are not only written back into the persistent memory (see Section 3.4), but may also result in permanent updates to the Methodology Layer's models, decision rules, or process templates. At the same time, the Operational Layer updates its state, and the Governance Layer may revise doctrines or objectives in response to sustained trends or disruptive events. In this way, the architecture supports both continuous single-loop learning—adapting actions and models based on immediate feedback—and vertical, double-loop adaptation, ensuring the entire system remains agile and coherently aligned with both tactical requirements and strategic intent over time.

4 Design Principles

CERES incorporates design principles derived from CSE, control theory, and modular agentic research:

- **Authenticity & Observation:** Following CSE, the system is grounded in observing work in context. It captures real user workflows, environmental signals and organizational dynamics to ensure that decisions reflect authentic conditions.
- **Abstraction & Modularity:** It abstracts cross-domain patterns and encapsulates functions into layers and modules, enabling reuse and substitutability [32], [33].
- **Innovation & Participatory Design:** The architecture invites experimentation and stakeholder involvement. Prototypes serve as tools for discovering promising designs, and human stewards can extend or refine methodologies and tools.
- **Interoperability & Scalability:** Standardised interfaces and data schemas facilitate integration with external systems. Modular decomposition allows the system to scale across data volume and complexity.
- **Operational awareness & context fidelity:** Recognising that machines cannot fully perceive their environment—"computers can't tell if their model of the world is the world they are in" [34]— CERES cultivates operational awareness through human–AI dialogue. Natural-language interactions and structured memory enable the LLM to be situated, aligning model predictions with evolving operational conditions and user feedback.
- **Adaptability & Learning:** Multi-level feedback loops support both single-loop and double-loop learning. Models, SOPs and doctrines evolve based on data, after-action reviews and user feedback.
- **Thinking-for-thinking:** Many current generative AI tools automate discrete tasks yet fail to scaffold users' thinking processes. Inspired by Ling's Brain Cache framework [30], CERES integrates mechanisms for externalising memory, structuring knowledge and proactively resurfacing context to support continuous cognitive development.
- **Transparency & Traceability:** Decision rationale is recorded, and every artefact references the doctrines and objectives that informed it, aiding auditability and trust.
- **Robustness & Security:** Error detection, resilience mechanisms and privacy-preserving techniques protect against uncertainty and protect sensitive data

5 Example Usage (Illustrative)

Consider a project management application. A programme manager defines strategic objectives (e.g. deliver a software release by a certain date) and doctrines (e.g., pragmatism, efficiency) in the Governance Layer. The Operational Layer maintains the current project state—tasks, resources and risks—mapped to different Operational Areas (e.g. project management, coding, legal&finance etc.). From t he Methodology Layer appropriate frameworks for each task (e.g., agile sprints,) and tools (critical path analysis, Monte Carlo simulation) can be selected . The human/AI operator can decompose objectives into sprints, tasks and milestones, predicts effort and risks, and generate, evaluate and execute candidate plans. The presence of a Governance Layer ensures that there is no deviation in task execution from the main objectives of the overall activity. During execution, the system monitors task completions and KPI metrics; deviations trigger recalibration of the operational objectives. Situation reports summarise progress; lessons learned feed back into the methodology library and may prompt adjustments to higher level Governance Layer. Similar logical flows apply to all workflows including personal life management, logistics, business management etc..

Importantly, because CERES encodes doctrines and strategic objectives at the Governance layer, thus using them to create and control the lower level operational objectives that guide execution, substituting one normative framework for another – e.g., egoism for reciprocity, or efficiency for resilience – would propagate entirely different operational objectives and outcomes without requiring a change to the operational areas definition or the Methodology Layer. This makes CERES uniquely adaptable among cognitive architectures, allowing it to be fine-tuned to the specific goals, values, and risk profiles .

6 Conclusion & Next Steps

CERES transforms the cognitive exoskeleton metaphor into a concrete, layered architecture. By embedding doctrinal governance, structured knowledge, modular planning, frictionless interfaces and continuous feedback, it bridges insights from CSE, decision support, control theory, knowledge management and modern agentic frameworks. The architecture remains domain-agnostic yet actionable; it can instantiate tailored operational cognition augmentation systems for private individuals or enterprise workers. In the authors' view, the introduction of this and similar human augmentation systems will become a necessary requirement in light of the ongoing development of fully autonomous agentic systems.

This research path will be expanded by (1) developing a detailed data model and ontology for the Operational Layer; (2) implementing a minimal set of LLM modules and decision heuristics in the Methodology Layer; (3) populating the knowledge framework and methodology library with CSE-derived patterns and modern best practices; (4) building a prototype technical interface with memory, tool gateway and domain-channel mapping; and (5) evaluating CERES in real-world scenarios, iterating based on feedback.

References

[1] T. Webb, S. S. Mondal, e I. Momennejad, «Improving Planning with Large Language Models: A Modular Agentic Architecture», 3 ottobre 2024, *arXiv*: arXiv:2310.00194. doi: 10.48550/arXiv.2310.00194.

[2] S. Yao *et al.*, «ReAct: Synergizing Reasoning and Acting in Language Models», 10 marzo 2023, *arXiv*: arXiv:2210.03629. doi: 10.48550/arXiv.2210.03629.

[3] T. Schick *et al.*, «Toolformer: Language models can teach themselves to use tools», *Adv. Neural Inf. Process. Syst.*, vol. 36, pp. 68539–68551, 2023.

[4] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, e Y. Zhuang, «HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face», 3 dicembre 2023, *arXiv*: arXiv:2303.17580. doi: 10.48550/arXiv.2303.17580.

[5] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, e L. He, «A survey of human-in-the-loop for machine learning», *Future Gener. Comput. Syst.*, vol. 135, pp. 364–381, 2022.

[6] E. Mosqueira-Rey, E. Hernández-Pereira, D. Alonso-Ríos, J. Bobes-Bascarán, e Á. Fernández-Leal, «Human-in-the-loop machine learning: a state of the art», *Artif. Intell. Rev.*, vol. 56, fasc. 4, pp. 3005–3054, apr. 2023, doi: 10.1007/s10462-022-10246-w.

[7] M. Lauritsen, «Toward a phenomenology of machine-assisted legal work», *RAIL*, vol. 1, p. 67, 2018.

[8] D. D. Woods e E. Hollnagel, *Joint Cognitive Systems: Patterns in Cognitive Systems Engineering*. Boca Raton: CRC Press, 2006. doi: 10.1201/9781420005684.

[9] E. Hollnagel e D. D. Woods, «Cognitive systems engineering: new wine in new bottles», *Int. J. Hum.-Comput. Stud.*, vol. 51, fasc. 2, ago. 1999, doi: 10.1006/ijhc.1982.0313.

[10] B. Brehmer, «The dynamic OODA loop: Amalgamating Boyd's OODA loop and the cybernetic approach to command and control», in *Proceedings of the 10th international command and control research technology symposium*, 2005, pp. 365–368. Consultato: 18 luglio 2025. [Online]. Disponibile su: https://www.researchgate.net/profile/Berndt-

Brehmer/publication/237290828_The_Dynamic_OODA_Loop_Amalgamating_Boyd%27s_OODA_Loop_and_the_Cybernetic_Approach_to_Command_and_Control_ASSESSMENT_TOOLS_AND_METRICS/links/0deec52bc4c85a9868000000/The-Dynamic-OODA-Loop-Amalgamating-Boyds-OODA-Loop-and-the-Cybernetic-Approach-to-Command-and-Control-ASSESSMENT-TOOLS-AND-METRICS.pdf

[11] P. Arcaini, E. Riccobene, e P. Scandurra, «Modeling and analyzing MAPE-K feedback loops for self-adaptation», in *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, IEEE, 2015, pp. 13–23. Consultato: 18 luglio 2025. [Online]. Disponibile su: https://ieeexplore.ieee.org/abstract/document/7194653/

[12] A. Realyvásquez Vargas, J. L. García Alcaraz, S. Satapathy, e J. R. Díaz-Reza, *The PDCA Cycle for Industrial Improvement: Applied Case Studies*. in Synthesis Lectures on Engineering, Science, and Technology. Cham: Springer Nature Switzerland, 2023. doi: 10.1007/978-3-031-26805-2.

[13] C. von Clausewitz, *On War*. Oxford: Oxford University Press, 2008.

[14] H. L. Tosi, «James March and Herbert Simon, Organizations», *Theor. Organ.*, 2008.

[15] S. Herbert, «The Architecture of Complexity», *Proc. Am. Philos. Soc.*, vol. 106, fasc. 6, pp. 467–482, 1962.

[16] S. Beer, *Brain of the Firm*. John Wiley & Sons, 1995. Consultato: 19 luglio 2025. [Online]. Disponibile su: https://books.google.com/books?hl=it&lr=&id=HEzYEAAAQBAJ&oi=fnd&pg=PR9&dq=Brain+of+the+Firm+(1972&ots=e3cquHJFIQ&sig=DAvvDWZCw4p985rGjmrLFQd-X9w

[17] I. Georgievski e M. Aiello, «An Overview of Hierarchical Task Network Planning», 28 marzo 2014, *arXiv*: arXiv:1403.7426. doi: 10.48550/arXiv.1403.7426.

[18] I. Nonaka e H. Takeuchi, *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, 1995. doi: 10.1093/oso/9780195092691.001.0001.

[19] S. L. Hoe, «Tacit knowledge, Nonaka and Takeuchi SECI model and informal knowledge processes», *Int. J. Organ. Theory Behav.*, vol. 9, fasc. 4, pp. 490–502, 2006.

[20] C. Argyris e D. A. Schön, *Organizational Learning: A Theory of Action Perspective*. Addison-Wesley Publishing Company, 1978.

[21] X. Wang *et al.*, «Knowledge graph quality control: A survey», *Fundam. Res.*, vol. 1, fasc. 5, pp. 607–626, set. 2021, doi: 10.1016/j.fmre.2021.09.003.

[22] A. Newell e H. A. Simon, *Human problem solving*. Prentice-Hall, 1972.

[23] M. Wooldridge, *An introduction to multiagent systems*. John wiley & sons, 2009. Consultato: 18 luglio 2025. [Online]. Disponibile su: https://books.google.com/books?hl=it&lr=&id=X3ZQ7yeDn2IC&oi=fnd&pg=PR13&dq=Wooldridge,%E2%80%AFM.+An+Introduction+to+MultiAgent+Systems.+2nd%

E2%80%AFed.+Chichester:+John+Wiley+%26+Sons,%E2%80%AF2009.&ots=WImg vs8v45&sig=pqoI0DMLsI7N--LCWQh80gwQF0M

[24] M. Bratman, «Intention, plans, and practical reason». Consultato: 18 luglio 2025. [Online]. Disponibile su: https://philpapers.org/rec/BRAIPA

[25] A. S. Rao e M. P. Georgeff, «BDI agents: From theory to practice.», in *Icmas*, 1995, pp. 312–319. Consultato: 18 luglio 2025. [Online]. Disponibile su: https://cdn.aaai.org/ICMAS/1995/ICMAS95-042.pdf

[26] H. Chase, «Langchain: Building applications with llms through composability (2022)», *URL Httpsgithub Comlangchain-Ailangchain*, 2023.

[27] Q. Wu *et al.*, «Autogen: Enabling next-gen LLM applications via multi-agent conversations», in *First Conference on Language Modeling*, 2024. Consultato: 19 luglio 2025. [Online]. Disponibile su: https://openreview.net/forum?id=BAakY1hNKS

[28] L. Chen, «AgentOS: the agent-based distributed operating system for mobile networks», *XRDS Crossroads ACM Mag. Stud.*, vol. 5, fasc. 2, pp. 12–14, nov. 1998, doi: 10.1145/333151.333156.

[29] Z. Xiang *et al.*, «GuardAgent: Safeguard LLM Agents by a Guard Agent via Knowledge-Enabled Reasoning», 29 maggio 2025, *arXiv*: arXiv:2406.09187. doi: 10.48550/arXiv.2406.09187.

[30] L. Ling, «Brain Cache: Generative AI as a Cognitive Exoskeleton for Externalizing, Structuring, and Activating Knowledge», *Github*, 2025, Consultato: 18 luglio 2025. [Online]. Disponibile su: https://generativeaiandhci.github.io/papers/2025/genaichi2025_51.pdf

[31] D. Woods, *Laws that Govern Joint Cognitive Systems at Work (JCS)*. 2006. doi: 10.5281/zenodo.4421807.

[32] D. B. Acharya, K. Kuppan, e B. Divya, «Agentic AI: Autonomous Intelligence for Complex Goals—A Comprehensive Survey», *IEEE Access*, vol. 13, pp. 18912–18936, 2025, doi: 10.1109/ACCESS.2025.3532853.

[33] S. Hosseini e H. Seilani, «The role of agentic AI in shaping a smart future: A systematic review», *Array*, vol. 26, p. 100399, lug. 2025, doi: 10.1016/j.array.2025.100399.

[34] D. Woods, *Laws that Govern Joint Cognitive Systems at Work (JCS)*. 2006. doi: 10.5281/zenodo.4421807.