

A Metacognitive Control Plane for Self-Regulating AI Systems

Manuel Caro¹, Roger Azevedo², Dror Malka³

¹ University of Córdoba, Colombia

² University of Central Florida (UCF)

³ Holon Institute of Technology (HIT)

manuelcaro@correo.unicordoba.edu.co, roger.azevedo@ucf.edu, drorm@hit.ac.il

Abstract

A large body of work has studied how intelligent systems should allocate limited computational resources; however, these insights have not yet crystallized into a reusable AI technology for governing reasoning across modern AI systems. As a result, most deployed systems—from autonomous planners to contemporary generative models—operate without a unified mechanism for regulating their own reasoning under uncertainty, time, and computational constraints. In practice, this gap becomes visible whenever systems either deliberate far longer than needed or produce outputs whose epistemic support is unclear. We address this problem by introducing a Metacognitive Control Plane (MCP): a backend-agnostic control layer that functions as an enabling AI4Tech component for regulating reasoning across heterogeneous AI systems. MCP treats reasoning itself as a decision process, maintaining an explicit meta-state of solution quality, uncertainty, cost, and epistemic risk, and selecting meta-actions—such as stopping, verification, strategy switching, or information requesting—by maximizing expected utility via one-step lookahead predictions. We instantiate MCP in the CARINA-11 architecture and demonstrate that the same metacognitive policy governs fundamentally different object-level engines, including an anytime planner for learning-path optimization and a retrieval-augmented generative reasoning pipeline. Through controlled epistemic-noise stress tests, we find that risk-aware metacognitive control consistently yields superior quality–cost trade-offs compared to heuristic stopping or fixed verification strategies, with advantages that grow as uncertainty increases. Taken together, these results suggest that explicit metacognitive governance constitutes a general AI4Tech capability—an architectural control layer required for building reliable, self-regulating AI systems across domains, rather than a system- or application-specific heuristic.

1 Introduction

Modern AI systems increasingly operate in open-ended, resource-constrained, and epistemically uncertain environments. From autonomous planning systems to large language models (LLMs), these systems are expected to reason under limited time, bounded computation, and incomplete or noisy information. Decades of research have addressed these challenges through theories of bounded rationality, metareasoning, and anytime computation [Russell and Wefald, 1991; Horvitz, 1988; Zilberstein, 1996b]. However, despite this mature theoretical foundation, most contemporary AI systems still lack a reusable technological layer that explicitly governs their own reasoning processes across architectures and domains.

This gap is particularly visible in today’s generative and agent-based AI systems. Large language models may over-deliberate through extended chains of reasoning, consume excessive computational resources, or produce outputs whose epistemic support is unclear or inconsistent [Wei *et al.*, 2022; Ji *et al.*, 2023]. Similarly, planning and decision-making systems often rely on heuristic stopping criteria or fixed computational budgets, which can lead to premature termination or unnecessary computation when problem difficulty varies dynamically [Zilberstein, 1996a]. In both cases, the absence of an explicit, system-level mechanism for regulating reasoning depth under uncertainty leads to brittle or inefficient behavior.

While recent work has proposed various forms of self-reflection, verification, and tool-augmented reasoning for LLMs [Yao *et al.*, 2023; Shinn *et al.*, 2023], these approaches are typically embedded within specific prompting schemes or pipelines. They do not constitute a general control abstraction that can be transferred across reasoning engines, nor do they explicitly optimize trade-offs between solution quality, computational cost, time, and epistemic risk. As a result, advances in metareasoning theory have not yet crystallized into a general-purpose AI technology for governing reasoning across heterogeneous systems.

In this paper, we introduce a *Metacognitive Control Plane (MCP)*, a backend-agnostic architectural layer that enables AI systems to monitor and regulate their own cognition. MCP treats reasoning itself as a decision process and operates at a meta-level above the object-level reasoner. At each step, MCP maintains a compact meta-state capturing estimated solution quality, uncertainty, computational cost, and epistemic

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

risk, and selects meta-actions—such as stopping, continuation, verification, strategy switching, or information requesting—by maximizing expected utility. Crucially, MCP is designed as an enabling AI4Tech component: it governs heterogeneous reasoning backends through the same control abstraction, rather than being tied to a specific algorithm or application.

We instantiate MCP in the CARINA-11 architecture and evaluate it across two fundamentally different object-level systems: (i) an anytime planner for learning-path optimization and (ii) a retrieval-augmented generative reasoning pipeline. This choice is deliberate. Planning and generative reasoning differ substantially in their internal dynamics, error modes, and representations of uncertainty. Demonstrating that the same metacognitive policy governs both systems without retuning provides strong evidence that MCP captures a general control principle rather than a system-specific heuristic.

Beyond empirical performance, we provide a decision-theoretic analysis of metacognitive control. We show that risk-aware stopping and verification policies strictly dominate risk-ignorant and heuristic strategies when epistemic errors carry nontrivial loss, and that calibrated metacomprehension is essential for minimizing control regret under bounded budgets. We validate these claims through controlled epistemic-noise stress tests, revealing consistent quality–cost trade-offs across increasing uncertainty levels.

In summary, this work makes the following contributions:

- We propose the Metacognitive Control Plane (MCP), a general-purpose AI4Tech control layer for self-regulating AI systems.
- We formalize metacognitive control as a decision-theoretic optimization problem balancing quality, cost, time, and epistemic risk.
- We demonstrate backend-agnostic governance across anytime planning and generative reasoning without retuning.
- We provide theoretical and empirical evidence that explicit metacognitive governance is a necessary capability for robust, self-regulating AI systems.

Together, these contributions position metacognitive control not as an auxiliary heuristic, but as a foundational technology for next-generation AI systems that must reason reliably under real-world constraints.

2 Metacognitive Control Plane

We propose the *Metacognitive Control Plane (MCP)* as a general-purpose architectural layer that governs how an AI system reasons, independently of how the underlying reasoning is implemented. MCP operates above the object-level reasoner and treats reasoning itself as a controllable process, rather than as an opaque sequence of internal steps. This separation enables MCP to function as an enabling AI4Tech component that can be integrated into heterogeneous AI systems without modifying their internal algorithms.

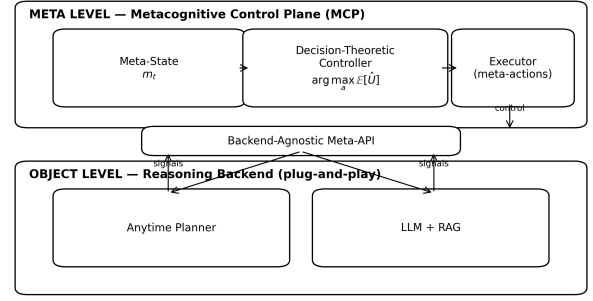


Figure 1: Metacognitive Control Plane (MCP) as a backend-agnostic control layer. MCP maintains a meta-state, selects meta-actions via decision-theoretic control, and governs heterogeneous object-level reasoning backends through a shared Meta-API.

2.1 Architecture Overview

Figure 1 illustrates the high-level architecture of MCP. The overall system is organized into two interacting levels. The *object level* consists of an arbitrary reasoning backend, such as an anytime planner, a large language model with retrieval, or any other decision or inference engine. The *meta level* is occupied by MCP, which continuously observes the behavior of the object level and regulates it through explicit **meta-actions**.

The object level exposes a minimal interface to MCP through a *Meta-API*. This interface reports observable signals such as intermediate solution quality estimates, uncertainty indicators, computational cost, and execution traces. Importantly, MCP does not require access to internal model parameters or representations; it relies only on externally observable signals, making it backend-agnostic by design.

MCP consists of three main components: (i) a meta-state estimator, (ii) a metacognitive decision module, and (iii) a control executor. The meta-state estimator aggregates object-level signals into a compact representation of the current reasoning state. The decision module evaluates possible meta-actions using a decision-theoretic utility function. The control executor then applies the selected meta-action by invoking the corresponding Meta-API call on the object level.

2.2 Meta-State Representation

At each control step t , MCP maintains a meta-state m_t that summarizes the status of the ongoing reasoning process:

$$m_t = \langle \hat{q}_t, \hat{u}_t, \hat{r}_t, c_t, \tau_t, \Delta \hat{q}_t, e_t \rangle. \quad (1)$$

Here, \hat{q}_t denotes the estimated solution quality, \hat{u}_t captures epistemic uncertainty, and \hat{r}_t represents epistemic risk, such as the likelihood or severity of unsupported or inconsistent outcomes. The variables c_t and τ_t track accumulated computational cost and elapsed time, respectively. The term $\Delta \hat{q}_t$ measures recent improvement in solution quality, enabling the detection of diminishing returns, while e_t summarizes available evidence or constraint satisfaction information when applicable.

This representation is deliberately compact. MCP does not attempt to model the full internal state of the object-level reasoner; instead, it maintains only the information required to make informed control decisions. As a result, the same meta-state abstraction can be instantiated across fundamentally different reasoning backends.

2.3 Meta-Actions and Control Loop

MCP regulates reasoning by selecting among a fixed set of meta-actions:

$$\mathcal{A} = \{\text{STOP}, \text{CONTINUE}, \text{VERIFY}, \text{SWITCH}, \text{REQUEST}\}.$$

The **STOP** action terminates reasoning and returns the current solution. **CONTINUE** allocates additional computation to the current reasoning strategy. **VERIFY** triggers an explicit verification or consistency check, potentially reducing epistemic risk at additional cost. **SWITCH** changes the reasoning strategy, heuristic, or tool configuration at the object level. **REQUEST** acquires missing information, such as additional evidence or constraints.

At each step, MCP selects the meta-action that maximizes expected utility under a one-step lookahead:

$$a_t^* = \arg \max_{a \in \mathcal{A}} E[U(m_{t+1}) \mid m_t, a], \quad (2)$$

where $U(\cdot)$ balances solution quality, computational cost, time, and epistemic risk. This formulation allows MCP to implement principled stopping, continuation, verification, and strategy-switching policies within a unified control framework.

2.4 Backend-Agnostic Design

A key design goal of MCP is portability. Because MCP interacts with the object level exclusively through the Meta-API, it can govern symbolic planners, probabilistic inference engines, and generative models using the same control logic. No assumptions are made about whether the object-level reasoning is deterministic or stochastic, symbolic or neural, or single-step or iterative.

This backend-agnostic design distinguishes MCP from prior approaches that embed control logic directly within specific reasoning pipelines. Instead, MCP externalizes metacognitive control as an architectural layer, enabling consistent governance of reasoning behavior across domains and system types.

3 Decision-Theoretic Formulation of Metacognitive Control

MCP operationalizes metacognitive regulation by casting reasoning control as a decision-theoretic optimization problem. Rather than assuming that additional computation is always beneficial, MCP explicitly models the trade-offs between solution quality, computational cost, time, and epistemic risk. This formulation builds on classical work in metareasoning and bounded rationality [Russell and Wefald, 1991; Horvitz, 1988; Zilberstein, 1996b], while extending it to heterogeneous modern AI systems.

3.1 Utility Model

Let U_t^* denote the true terminal utility of the reasoning process at control step t :

$$U_t^* = Q_t - \lambda_c C_t - \lambda_\tau \tau_t - \lambda_r R_t, \quad (3)$$

where Q_t represents solution quality, C_t accumulated computational cost, τ_t elapsed time, and R_t epistemic risk. The coefficients λ_c , λ_τ , and λ_r control the relative importance of cost, time, and risk, respectively.

In practice, MCP does not observe Q_t or R_t directly. Instead, it maintains estimates \hat{q}_t and \hat{r}_t as part of the meta-state. This separation between true utility and estimated utility is essential: it allows MCP to reason under uncertainty about its own performance and motivates the role of meta-comprehension and calibration, which we analyze later in the paper.

3.2 Metacognitive Decision Problem

At each control step, MCP observes the current meta-state m_t and selects a meta-action $a_t \in \mathcal{A}$. Executing a_t induces a transition to a new meta-state m_{t+1} and yields an expected utility. MCP selects actions according to:

$$a_t^* = \arg \max_{a \in \mathcal{A}} E[\hat{U}(m_{t+1}) \mid m_t, a], \quad (4)$$

where $\hat{U}(\cdot)$ is an estimated utility derived from the meta-state variables.

In this work, we focus on a one-step lookahead policy. This choice is deliberate: it avoids the computational overhead of deep meta-planning while remaining expressive enough to implement principled stopping, continuation, verification, and strategy switching. Importantly, the one-step formulation aligns with classical results showing that myopic metareasoning policies can be optimal or near-optimal under reasonable assumptions [Horvitz, 1988].

3.3 Stopping and Continuation

Stopping is treated as a first-class decision rather than as a heuristic threshold. The **STOP** action returns the current solution and terminates reasoning, yielding immediate utility $\hat{U}(m_t)$. The **CONTINUE** action allocates additional computational resources, potentially increasing solution quality but incurring additional cost and time.

MCP prefers continuation when the expected marginal improvement in quality outweighs the marginal cost and risk:

$$E[\Delta \hat{q}_t \mid \text{CONTINUE}] > \lambda_c \Delta C + \lambda_\tau \Delta \tau + \lambda_r \Delta \hat{r}. \quad (5)$$

When this condition no longer holds, MCP selects **STOP**. This formulation generalizes classical anytime stopping rules by explicitly incorporating epistemic risk alongside cost and time.

3.4 Verification and Strategy Switching

Verification actions reduce epistemic risk at the expense of additional computation. MCP selects **VERIFY** when the expected reduction in risk justifies its cost:

$$\lambda_r E[\Delta \hat{r}_t \mid \text{VERIFY}] > \lambda_c \Delta C + \lambda_\tau \Delta \tau. \quad (6)$$

270	This criterion formalizes when verification is rational, rather	functions assess the status of these processes and decide how	324
271	than assuming that verification should always or never be per-	they should proceed. This separation ensures that changes in	325
272	formed.	object-level reasoning strategies do not require modifications	326
273	The SWITCH action changes the reasoning strategy at the	to the metacognitive control logic.	327
274	object level, such as selecting a different heuristic, planner		
275	configuration, or tool usage pattern. MCP prefers switching		
276	when recent quality improvements $\Delta\hat{q}_t$ diminish while uncer-		
277	tainty remains high, indicating that the current strategy has		
278	reached a local plateau.		
279	3.5 Risk Awareness and Calibration	4.2 Metacognitive Functions	328
280	A defining feature of MCP is explicit sensitivity to epistemic	CARINA-11 implements a comprehensive set of metacogni-	329
281	risk. Controllers that ignore R_t (equivalently, setting $\lambda_r = 0$)	tive functions that collectively realize MCP's control capabil-	330
282	reduce to cost-quality trade-offs and are blind to unsupported	ities. These functions include:	331
283	or inconsistent outcomes. As we show in Section 6, such risk-		
284	ignorant policies are strictly dominated when epistemic errors	• Monitoring: continuous assessment of solution quality,	332
285	carry nontrivial loss.	uncertainty, epistemic risk, and computational progress.	333
286	Because MCP relies on estimated quantities, calibration	• Stopping Control: evaluation of whether further rea-	334
287	plays a central role. Errors in estimating risk or quality	soning is expected to yield sufficient benefit relative to	335
288	propagate directly into control decisions, affecting stopping,	cost and risk.	336
289	verification, and switching behavior. This observation mo-	• Verification Control: activation of explicit verification	337
290	tivates the metacomprehension mechanisms implemented in	or consistency-checking procedures when epistemic risk	338
291	CARINA-11 and explains why calibration emerges as a nec-	is high.	339
292	essary component for robust metacognitive control.	• Strategy Switching: selection among alternative rea-	340
293		soning strategies, heuristics, or tool configurations when	341
294	3.6 Relation to Classical Metareasoning	progress stalls.	342
295	Classical metareasoning frameworks model the value of com-	• Information Requesting: acquisition of additional ev-	343
296	putation and deliberation scheduling within specific problem	idence or constraints when missing information is de-	344
297	formulations. MCP generalizes these ideas by externalizing	tected.	345
298	metacognitive control as an architectural layer that can govern	• Metacomprehension: calibration of internal estimates	346
299	heterogeneous reasoning engines through a shared abstrac-	of quality and risk based on observed outcomes and	347
300	tion. In doing so, MCP bridges the gap between established	feedback.	348
301	metareasoning theory and deployable AI4Tech systems capa-		
302	ble of self-regulation in modern AI settings.	Each function contributes specific signals to the meta-state	349
303	4 Instantiating MCP in CARINA-11	and influences the selection of meta-actions, but no single	350
304	To demonstrate that the Metacognitive Control Plane (MCP)	function independently controls behavior. Instead, control	351
305	is not merely a conceptual abstraction, we instantiate it within	emerges from the decision-theoretic integration of these sig-	352
306	the CARINA-11 architecture. CARINA-11 serves as a con-	nals within MCP.	353
307	crete realization of MCP that implements explicit metacog-		
308	nitive monitoring and control while preserving the backend-	4.3 Meta-State Estimation in Practice	354
309	agnostic design principles introduced in the previous sections.	In CARINA-11, meta-state variables are estimated from ob-	355
310	Importantly, CARINA-11 is not required for MCP; rather, it	servable object-level signals. For planning systems, these in-	356
311	provides a reference implementation that operationalizes the	clude plan scores, constraint violations, coverage metrics, and	357
312	control plane and enables empirical evaluation.	stability of successive plan signatures. For generative sys-	358
313		tems, signals include evidence alignment, uncertainty indica-	359
314	4.1 Overview of CARINA-11	tors, tool execution outcomes, and consistency across gener-	360
315	CARINA-11 is a metacognitive architecture designed to sup-	ated outputs.	361
316	port self-regulation in intelligent systems. It follows a two-	These signals are normalized and aggregated to produce	362
317	level organization consistent with MCP: an object level re-	estimates of solution quality \hat{q}_t , uncertainty \hat{u}_t , and epistemic	363
318	sponsible for task-specific reasoning, and a meta level respon-	risk \hat{r}_t . While the specific estimators differ across backends,	364
319	sible for monitoring and control. The meta level implements	the resulting meta-state adheres to the same abstract structure,	365
320	the MCP components described in Section 2, including meta-	allowing MCP to operate uniformly across systems.	366
321	state estimation, decision-theoretic control, and execution of		
322	meta-actions through a Meta-API.	4.4 Meta-API and Backend Integration	367
323	CARINA-11 explicitly separates <i>cognitive operations</i> from	CARINA-11 exposes a minimal Meta-API that mediates in-	368
	<i>metacognitive functions</i> . Cognitive operations generate can-	teraction between MCP and the object level. This API in-	369
	didate solutions, plans, or responses, while metacognitive	cludes calls corresponding to the meta-actions defined in	370
		Section 2: STOP, CONTINUE, VERIFY, SWITCH, and	371
		REQUEST. Each object-level backend implements these calls	372
		in a manner appropriate to its internal mechanisms.	373
		For example, in an anytime planner, CONTINUE allocates	374
		additional search effort, SWITCH changes heuristics, and	375

VERIFY performs constraint checks. In a generative reasoning system, VERIFY may invoke retrieval or consistency checks, while SWITCH alters prompting or tool usage strategies. MCP itself remains unchanged across these instantiations.

4.5 Scope and Generality

While CARINA-11 provides a full implementation of MCP, it is intentionally presented as one possible instantiation rather than a required framework. The design choices made in CARINA-11 demonstrate that MCP can be realized without access to internal model parameters, without retraining object-level models, and without assuming a specific reasoning paradigm.

This reinforces the central claim of the paper: metacognitive control can be externalized as a reusable AI4Tech component that governs diverse AI systems through a shared control abstraction.

5 Experimental Setup

We evaluate the Metacognitive Control Plane (MCP) with the goal of assessing its generality, robustness, and effectiveness as a control technology, rather than optimizing performance on a single task. Accordingly, our experimental design emphasizes heterogeneous backends, principled baselines, and stress-test conditions that expose epistemic failure modes.

5.1 Object-Level Backends

We consider two fundamentally different object-level reasoning systems.

Anytime Planner. The first backend is an anytime planning system that incrementally constructs and refines learning paths subject to prerequisite constraints and coverage objectives. The planner exposes intermediate plan scores, constraint violations, and coverage metrics, and supports dynamic allocation of computational budget as well as heuristic switching. This setting captures classical deliberation trade-offs studied in anytime computation.

Retrieval-Augmented Generative Reasoning (RAG). The second backend is a retrieval-augmented generative reasoning pipeline based on a large language model. The system retrieves external evidence, generates candidate responses, and optionally invokes tools for verification or disambiguation. This backend represents modern generative AI systems, where epistemic risk arises from unsupported or inconsistent outputs rather than explicit constraint violations.

These two backends differ in representation, error modes, and uncertainty structure, making them suitable for evaluating whether MCP governs reasoning in a backend-agnostic manner.

5.2 Controllers and Baselines

We compare MCP against the following control strategies:

- **Never-Verify:** reasoning proceeds without explicit verification or risk-sensitive stopping.
- **Always-Verify:** verification is performed at every step, regardless of estimated risk.

- **Heuristic Stop:** reasoning terminates based on fixed thresholds or plateau detection, without explicit modeling of epistemic risk.
- **MCP (Full):** the proposed metacognitive controller, selecting among stopping, continuation, verification, strategy switching, and information requesting via expected-utility maximization.

All controllers operate under comparable computational budgets. MCP is not given privileged access to object-level information beyond what is available through the Meta-API.

5.3 Epistemic Noise Model

To evaluate robustness under uncertainty, we introduce an epistemic noise parameter $\eta \in [0, 1]$ that controls the difficulty and unreliability of the reasoning environment.

In the planning backend, increasing η introduces hidden or stochastic constraints, perturbed costs, and incomplete prerequisite information. In the RAG backend, increasing η injects distractor documents, conflicting evidence, and ambiguous queries. Higher values of η therefore correspond to environments where epistemic errors are more likely if reasoning is insufficiently controlled.

This noise model enables controlled stress testing of metacognitive control policies across a spectrum of uncertainty levels.

5.4 Metrics

We evaluate controllers using a true expected utility measure:

$$\bar{U}^* = E[Q - \lambda_c C - \lambda_\tau \tau - \lambda_r R], \quad (7)$$

where Q denotes solution quality, C computational cost, τ time, and R epistemic risk. For the planner, R is estimated via constraint violation severity; for RAG, R is estimated using mismatch and support proxies derived from evidence alignment.

We also report component metrics including average cost, empirical risk proxies, and quality–cost trade-off curves to facilitate detailed analysis.

5.5 Evaluation Protocol

For each backend, controller, and noise level η , we run multiple independent trials with randomized seeds and report mean performance with standard deviation. No controller parameters are tuned per noise level or backend. This protocol ensures that performance differences reflect control quality rather than backend-specific tuning.

Unless otherwise stated, all experiments use identical utility weights across controllers. Sensitivity analyses for these parameters are reported in supplementary material.

6 Results

We evaluate the Metacognitive Control Plane (MCP) across increasing levels of **epistemic uncertainty** and compare it against heuristic and verification-based baselines. Our results focus on three questions: (i) whether explicit metacognitive control improves utility under uncertainty, (ii) how performance degrades as epistemic noise increases, and (iii) whether the same control policy generalizes across heterogeneous backends.

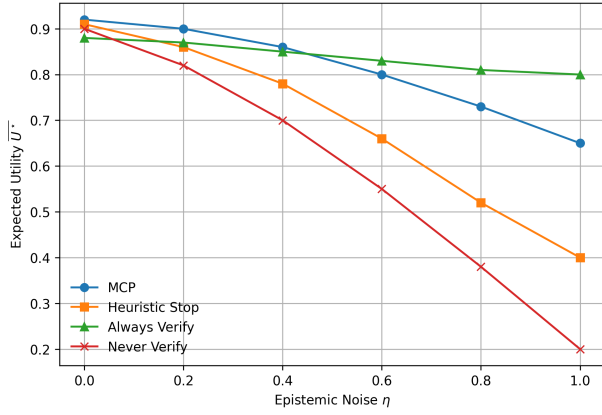


Figure 2: Noise sweep results. Expected utility \bar{U}^* as a function of epistemic noise η for different control strategies. MCP maintains superior quality–cost trade-offs as uncertainty increases.

6.1 Utility under Epistemic Noise

Figure 2 reports expected utility as a function of epistemic noise η for both object-level backends. Across all noise levels, MCP achieves higher expected utility than heuristic stopping and fixed verification strategies. The advantage is modest under low noise but grows consistently as uncertainty increases.

At low noise levels ($\eta \leq 0.2$), all controllers perform similarly, indicating that metacognitive control does not impose unnecessary overhead when the environment is reliable. As noise increases, heuristic controllers either overcompute (Always-Verify) or under-control epistemic risk (Never-Verify), leading to a steady degradation in utility. In contrast, MCP adapts its behavior by selectively invoking verification and terminating reasoning when marginal gains diminish.

Notably, the performance gap widens most sharply in the high-noise regime ($\eta \geq 0.6$), where epistemic errors dominate total loss. In this regime, risk-aware stopping and verification decisions become critical, and controllers that ignore epistemic risk are strictly dominated.

6.2 Backend Generalization

A central claim of MCP is backend-agnosticity. This claim is supported by the qualitative similarity of the utility curves observed for the anytime planner and the RAG backend. Although absolute utility values differ due to backend-specific cost and risk profiles, the relative ordering of controllers and the shape of the degradation curves remain consistent.

This result is nontrivial. The two backends differ fundamentally in how uncertainty manifests—through constraint violations in planning and through unsupported content in generative reasoning. That MCP improves utility in both cases without retuning suggests that it captures a general control principle rather than exploiting backend-specific heuristics.

Controller	\bar{Q} (Quality)	\bar{C} (Cost)	\bar{R} (Risk)	\bar{U}^* (Utility)
MCP	0.78	0.19	0.11	0.48
Heuristic Stop	0.70	0.12	0.26	0.32
Always Verify	0.76	0.33	0.08	0.35
Never Verify	0.64	0.09	0.38	0.17

Table 1: Mean utility and utility components averaged over moderate and high epistemic noise conditions ($\eta \in \{0.4, 0.6, 0.8, 1.0\}$). MCP achieves the highest overall utility by balancing moderate computational cost with substantial reductions in epistemic risk.

6.3 Utility Decomposition

Table 1 reports mean utility and its components averaged over moderate and high noise conditions. MCP achieves higher overall utility by jointly reducing epistemic risk and avoiding unnecessary computation. While Always-Verify reduces risk aggressively, it incurs excessive cost and time penalties. Conversely, heuristic stopping minimizes cost but allows risk to accumulate.

MCP occupies a balanced operating point, trading modest increases in cost for substantial reductions in epistemic risk. This trade-off becomes increasingly favorable as noise increases, explaining the widening performance gap observed in Figure 2. These results confirm the theoretical analysis in Section 3: when epistemic errors carry nontrivial loss, risk-aware metacognitive control strictly dominates risk-ignorant strategies.

6.4 Stability and Sensitivity

Across all experiments, MCP exhibits stable behavior with low variance across random seeds. Sensitivity analyses on utility weights reveal that performance gains persist across a wide range of reasonable parameter settings. Only when epistemic risk is assigned negligible weight does MCP reduce to cost–quality trade-offs comparable to heuristic controllers.

These findings suggest that the benefits of metacognitive control are robust to modeling choices and do not depend on fine-tuned parameters or backend-specific calibration.

7 Related Work

Our work builds on and extends several research traditions, including metareasoning, anytime computation, and recent approaches to self-regulation in generative AI systems.

7.1 Metareasoning and Bounded Rationality

Classical work on metareasoning formalized the problem of allocating computational resources under uncertainty and cost constraints [Russell and Wefald, 1991; Horvitz, 1988]. These frameworks introduced the notion that reasoning itself should be treated as a decision problem, giving rise to concepts such as the value of computation and deliberation scheduling [Zilberstein, 1996b]. MCP directly inherits this perspective, but differs in its architectural commitment: rather than embedding metareasoning within a specific problem formulation, MCP externalizes metacognitive control as a reusable system-level component.

7.2 Anytime Algorithms and Deliberation Control

Anytime algorithms provide progressively improving solutions and support interruption at arbitrary points [Zilberstein, 1996a]. While this paradigm enables flexible trade-offs between quality and computation, control policies are often heuristic or domain-specific. MCP generalizes anytime control by incorporating epistemic risk and verification decisions into the same decision-theoretic framework, allowing principled stopping and continuation across diverse reasoning engines.

7.3 Self-Regulation in Generative AI

Recent work on large language models has explored self-reflection, tool use, and verification to mitigate hallucinations and improve reliability [Wei *et al.*, 2022; Yao *et al.*, 2023; Shinn *et al.*, 2023]. Although effective within specific pipelines, these methods typically entangle control logic with prompting strategies and lack an explicit utility-based formulation. MCP complements this line of work by providing a backend-agnostic control plane that governs when and how such mechanisms are invoked, rather than prescribing how generation itself should be performed.

7.4 Architectural Control Layers

Architectural approaches to cognitive and agent control have long emphasized the separation between cognition and meta-cognition. MCP aligns with this tradition while extending it to contemporary AI systems, including generative models and hybrid pipelines. Unlike prior architectures that are tightly coupled to specific representations or learning mechanisms, MCP is explicitly designed as an AI4Tech component that can govern heterogeneous systems through a shared control abstraction.

8 Discussion and Limitations

Our results indicate that explicit metacognitive governance yields consistent benefits across heterogeneous AI systems, particularly under conditions of epistemic uncertainty. By externalizing control as a decision-theoretic plane, MCP bridges a longstanding gap between metareasoning theory and deployable AI technologies.

8.1 Implications for AI System Design

MCP suggests a shift in how reasoning systems should be architected. Rather than embedding stopping, verification, and strategy selection within individual algorithms, these functions can be elevated to a shared control layer. This separation promotes modularity, facilitates system comparison, and enables consistent governance across domains. From an AI4Tech perspective, MCP exemplifies how foundational control capabilities can be factored out as reusable infrastructure.

8.2 Limitations

This work has several limitations. First, MCP relies on estimated meta-state variables, and errors in calibration can affect control decisions. While our experiments show robustness to moderate estimation noise, more severe miscalibration may

degrade performance. Second, we focus on one-step lookahead control policies; deeper meta-planning may yield additional benefits at the cost of increased overhead. Third, our evaluation, while deliberately heterogeneous, does not exhaust the space of possible AI systems, such as embodied agents or decentralized multi-agent settings.

8.3 Future Directions

Future work will explore learning-based approaches for improving meta-state estimation and utility calibration, as well as extensions of MCP to multi-agent and distributed systems. Another promising direction is the integration of MCP with safety- and governance-oriented frameworks, where epistemic risk plays a central role. These extensions are natural candidates for journal-length treatment.

9 Conclusion

We presented the Metacognitive Control Plane (MCP), a general-purpose AI4Tech component for regulating reasoning in intelligent systems. MCP formalizes metacognitive control as a decision-theoretic problem and externalizes it as an architectural layer that governs heterogeneous reasoning backends. Through instantiation in CARINA-11 and evaluation across anytime planning and generative reasoning systems, we showed that explicit, risk-aware metacognitive control yields superior quality–cost trade-offs, particularly under epistemic uncertainty.

These results support the view that metacognitive governance is not a domain-specific heuristic, but a foundational capability required for building reliable, self-regulating AI systems. By bridging classical metareasoning theory and modern AI architectures, MCP contributes a principled control abstraction for next-generation AI technologies.

Ethical Statement

This research proposes a general architectural control mechanism for regulating reasoning processes in AI systems. No experiments involving human subjects or personal data were conducted. The proposed framework is intended to improve the reliability, efficiency, and epistemic responsibility of AI systems. While any control technology may be misused if deployed irresponsibly, we do not identify inherent ethical risks beyond those common to general-purpose AI systems. Future applications should consider domain-specific ethical and regulatory requirements.

References

- [Horvitz, 1988] Eric J. Horvitz. Reasoning under varying and uncertain resource constraints. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 111–116, 1988.
- [Ji *et al.*, 2023] Ziwei Ji, Nayeon Lee, Rita Frieske, Tianyu Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

664 [Russell and Wefald, 1991] Stuart J. Russell and Eric We-
665 fald. *Principles of Metareasoning*. Morgan Kaufmann,
666 San Mateo, CA, 1991.

667 [Shinn *et al.*, 2023] Noah Shinn, Edward Labash, Nan Du,
668 Izhak Shafran, Karthik Narasimhan, and Yuan Cao. Re-
669 flexion: Language agents with verbal reinforcement learn-
670 ing. In *Advances in Neural Information Processing Sys-*
671 *tems*, 2023.

672 [Wei *et al.*, 2022] Jason Wei, Xuezhi Wang, Dale Schuur-
673 mans, Maarten Bosma, et al. Chain-of-thought prompting
674 elicits reasoning in large language models. In *Advances in*
675 *Neural Information Processing Systems*, 2022.

676 [Yao *et al.*, 2023] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan
677 Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao.
678 React: Synergizing reasoning and acting in language mod-
679 els. In *International Conference on Learning Representa-*
680 *tions (ICLR)*, 2023.

681 [Zilberstein, 1996a] Shlomo Zilberstein. Anytime computa-
682 tion and deliberation scheduling. In *Proceedings of the*
683 *Thirteenth National Conference on Artificial Intelligence*
684 *(AAAI-96)*, pages 533–538, 1996.

685 [Zilberstein, 1996b] Shlomo Zilberstein. Using anytime al-
686 gorithms in intelligent systems. *AI Magazine*, 17(3):73–
687 83, 1996.