DYNO: Dynamic Neurosymbolic Orchestrator for Multi-Agent Systems

Ritvik Garimella, Chathurangi Shyalika, Renjith Prasad, Amit Sheth

Artificial Intelligence Institute, University of South Carolina, USA
ritvikg@sc.edu, jayakodc@email.sc.edu, kaippilr@mailbox.sc.edu, amit@sc.edu

Abstract

Large Language Model (LLM)-based multi-agent systems (LaMAS) represent an emerging paradigm for tackling complex, multi-step reasoning and decision-making problems. As these systems scale, orchestration, which is the ability to coordinate, manage, and evaluate the interactions among diverse agents, becomes central to their success. While recent orchestrators such as AgentFlow have demonstrated promise in managing communication and task delegation, they remain limited in their ability to understand task semantics, coordinate heterogeneous agent types (e.g., reactive vs. cognitive), and adaptively align outputs with human-defined goals. In this position paper, we introduce the ***DYNO*** *(Dynamic Neurosymbolic Orchestrator)*, a system developed as part of our broader research framework on neurosymbolic AI for robust, interpretable, and trustworthy composite intelligence. DYNO integrates interdependent components to plan, execute, evaluate, and refine workflows iteratively. Each component cooperates through shared registries of agents, data, knowledge, and evaluation metrics, allowing the system to continuously optimize task performance. We argue that such dynamic orchestration, combining symbolic decomposition with neural adaptability, is essential for achieving scalable, interpretable, and self-correcting multi-agent intelligence. The paper positions dynamic orchestration as a foundational step toward reliable, trustworthy, and human-aligned multi-agent systems.

Introduction and Related Work

The automation of complex tasks has long been a central focus of technological advancement aimed at improving efficiency and reducing human effort. Process Automation (PA) (Cichocki et al. 1997) operationalizes this goal by automating repetitive procedures to enhance speed, accuracy, and consistency. Robotic Process Automation (RPA) extends this paradigm by translating manual tasks into structured workflows that coordinate multiple agents, functions, and tools to accomplish defined objectives (Ivančić, Suša Vugec, and Bosilj Vukšić 2019; Agostinelli, Marrella, and Mecella 2020). Despite its widespread adoption, RPA remains labor-intensive and brittle, requiring extensive manual configuration and demonstrating limited adaptability to dynamic environments (Eulerich et al. 2024).

Large Language Models (LLMs) have enabled automation that extends beyond static rule execution (Ahn et al. 2022; Cheng, Li, and Bing 2023; Qian et al. 2023). This progress has led to a shift from RPA towards *Agentic Process Automation* (APA) (Wornow et al. 2024; Li et al. 2024; Ye et al. 2023), where the orchestration of workflows is guided by LLMs. However, current APA systems rely heavily on LLMs that lack semantic alignment, leading to low reliability in mission-critical applications such as healthcare, autonomous driving, and manufacturing. This lack of semantic grounding limits their ability to orchestrate complex, interdependent workflows that define how agents plan, communicate, and refine shared goals at scale.

Recent advancements in enterprise-oriented Multi-Agent Systems (MAS) (Wu et al. 2024; Hu et al. 2025; Hong et al. 2023; OpenAI 2025) attempt to address this limitation by distributing specialized roles across multiple collaborating LLM-based agents. However, the absence of iterative, learning-driven improvement cycles continues to constrain their performance and adaptability. Orchestrators such as AgentFlow (Li et al. 2025) partially mitigate this issue through trainable planning mechanisms, yet they still fall short in understanding task semantics, coordinating heterogeneous agent types (e.g., reactive, cognitive, neural, symbolic), and aligning outputs with human-defined goals.

Another key limitation lies in the fact that progress in automation has largely focused on enhancing the intelligence of individual agents, while the coordination layer governing their interaction remains underexplored (Cemri et al. 2025; Gao et al. 2025). As multi-agent systems grow in complexity and heterogeneity, effective orchestration emerges as the key determinant of system reliability and alignment. We view orchestration not as an auxiliary mechanism but as a central architectural principle that unifies learning, reasoning, and evaluation across agents.

To address these gaps, we propose ***DYNO*** (Dynamic Neurosymbolic Orchestrator), developed within the broader ***C3AN*** framework: *Custom, Compact, Composite AI with Neurosymbolic Integration* (Sheth et al. 2025). C3AN is an emerging AI paradigm emphasizing robustness, intelligence, and trust in composite AI systems. The *Custom* component refers to domain-specific data and workflows; *Compact* emphasizes efficiency and deployability across infrastructures, including edge devices; and *Composite* denotes the modular orchestration of neural, symbolic, and decision modules into unified systems. The *Neurosymbolic* principle integrates neural learning with symbolic reasoning within each module, enabling transparent, knowledge-aligned, and explainable decision-making (Sheth, Roy, and Gaur 2023).

Position: In this paper, we argue that orchestration, rather than individual agent intelligence, should serve as the central design principle for next-generation LaMAS. This approach combines symbolic constraints with neural adaptation to achieve continual learning, interpretability, and alignment in autonomous agent collectives.

The structure of this paper is organized as follows. We first formalize the concept of neurosymbolic orchestration, followed by DYNO's three-stage architecture for task planning, workflow orchestration, and iterative improvement. We conclude with an illustrative implementation that demonstrates the application of DYNO in a smart manufacturing context.

Neurosymbolic Orchestration

We formalize the concept of ***Neurosymbolic Orchestration***, which unifies neural adaptability with symbolic structure to enable reasoning-driven coordination among agents. Unlike conventional orchestrators, which focus on managing communication and control flow, neurosymbolic orchestration augments these capabilities with *semantic reasoning*. It not only routes information but also interprets, verifies, and refines the meaning of interactions among agents. Neural modules propose and explore potential workflows whereas symbolic modules constrain, validate, and adjust them through iterative feedback. This continual interplay grounds complex workflows in semantics, ensuring that orchestration remains interpretable, adaptive, and self-improving.

DYNO instantiates this paradigm through a deliberately three-stage framework comprising *Task Planning*, *Workflow Orchestration*, and *Iterative Improvement* (as shown in Figure 1). These stages mirror the essential phases of reasoning, *generation*, *execution*, and *reflection* and together form a closed cognitive loop. The **Task Planner** integrates neural creativity with symbolic validation to decompose high-level goals into semantically coherent sub-tasks by using planning ontologies. The **Workflow Orchestration** stage operationalizes these sub-tasks by composing agents, data, and evaluations into executable workflows. The resulting workflow is then verified for compatibility within the main pipeline by orchestration verifier and integrated into the final setup. The **Iterative Improvement** stage then closes the loop, feeding evaluation signals and cached outputs back into the system to refine its knowledge, agents, and ontologies. This closed loop enables continual learning throughout the planning and workflow generation process.

The reasoning cycle in DYNO is grounded in two foundational constructs. They are ***Operations*** and ***Registries***. Together, they define the symbolic substrate that enables the system to compose, interpret, and continually refine complex workflows. *Operations* provide the formal grammar of orchestration, while *Registries* supply the shared semantic memory that maintains coherence across planning, execution, and learning.

Operations

DYNO's orchestration semantics are grounded in a minimal set of three symbolic operators that define relationships among tasks, agents, and data streams. These operators constitute the primitive algebra of composition, allowing the system to generate complex yet interpretable workflows while constraining them to remain semantically valid. Limiting the operator set ensures transparency and prevents incoherent task structures (Figure 2). The operators are:

- **Sequential (seq)**: defines a directed dependency between components a and b, where the output of a becomes the input of b, enforcing causal consistency in linear flow.
- **Branch (brn)**: propagates the output of a component a to multiple downstream components $\{b_1, b_2, ..., b_n\}$, enabling controlled parallelism and exploration.
- **Aggregation (agg)**: merges the outputs of multiple components $\{a_1, a_2, ..., a_n\}$ into a unified component b, supporting synthesis and ensemble reasoning.

This triad forms DYNO's symbolic grammar for reasoning-driven orchestration. Complex many-to-many interactions can be represented as compositions of these primitives, providing explainability while allowing neural modules to adapt their instantiations dynamically based on task semantics (Figure 1).

Registries

Complementing the compositional logic of *Operations*, DYNO maintains four interconnected registries that act as persistent semantic memory. They provide context, traceability, and adaptability across all stages of orchestration, enabling the system to reason over its own history and continually improve. Each registry serves a distinct but interlinked role:

- **Data Registry**: maintains structured metadata for all data assets, capturing modality, provenance, relevance, and embeddings to enable transparent, versioned retrieval.
- **Knowledge Registry**: stores ontologies, semantic graphs, and reasoning rules, linking symbolic entities with neural embeddings to ensure consistent neurosymbolic alignment.
- **Agent Registry**: describes agent capabilities, modalities, and performance profiles, supporting adaptive agent selection and fine-tuning within orchestration loops.
- **Evaluation Registry**: unifies evaluation metrics and feedback histories, providing continual signals for self-assessment and learning.

These registries form DYNO's evolving knowledge substrate. They ensure that each orchestration cycle benefits from accumulated context and feedback, transforming DYNO from a static coordinator into a continuously learning orchestration ecosystem.

Stage-1: Task Planner

The ***Task Planner*** is the first stage of DYNO's neurosymbolic orchestration pipeline, transforming high-level objectives into structured, executable sub-tasks. It comprises of

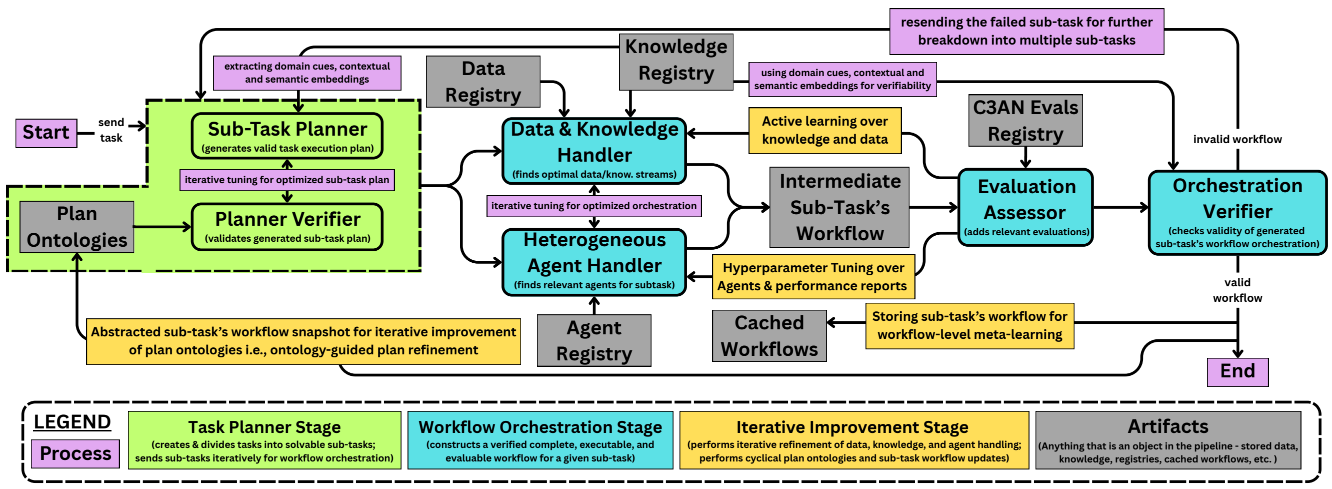

resending the failed sub-task for further breakdown into multiple sub-tasks
Knowledge Registry
Data Registry
extracting domain cues, contextual and semantic embeddings
using domain cues, contextual and semantic embeddings for verifiability
Start
send task
Sub-Task Planner
(generates valid task execution plan)
iterative tuning for optimized sub-task plan
Plan Ontologies
Planner Verifier
(validates generated sub-task plan)
Data & Knowledge Handler
(finds optimal data/know. streams)
iterative tuning for optimized orchestration
Heterogeneous Agent Handler
(finds relevant agents for subtask)
Active learning over knowledge and data
C3AN Evals Registry
Intermediate Sub-Task's Workflow
Evaluation Assessor
(adds relevant evaluations)
Orchestration Verifier
(checks validity of generated sub-task's workflow orchestration)
invalid workflow
valid workflow
Hyperparameter Tuning over Agents & performance reports
Agent Registry
Cached Workflows
Storing sub-task's workflow for workflow-level meta-learning
Abstracted sub-task's workflow snapshot for iterative improvement of plan ontologies i.e., ontology-guided plan refinement
End
LEGEND
Process
Task Planner Stage
(creates & divides tasks into solvable sub-tasks; sends sub-tasks iteratively for workflow orchestration)
Workflow Orchestration Stage
(constructs a verified complete, executable, and evaluable workflow for a given sub-task)
Iterative Improvement Stage
(performs iterative refinement of data, knowledge, and agent handling; performs cyclical plan ontologies and sub-task workflow updates)
Artifacts
(Anything that is an object in the pipeline - stored data, knowledge, registries, cached workflows, etc.)

Figure 1: *Architecture of DYNO*. Task Planner divides the given task into solvable sub-tasks, Workflow Orchestration creates executable workflows for solving the sub-tasks, and Iterative Improvements provide loops for continual training. All Registries, Ontologies and Workflows are stored in shared memory space called Artifacts.

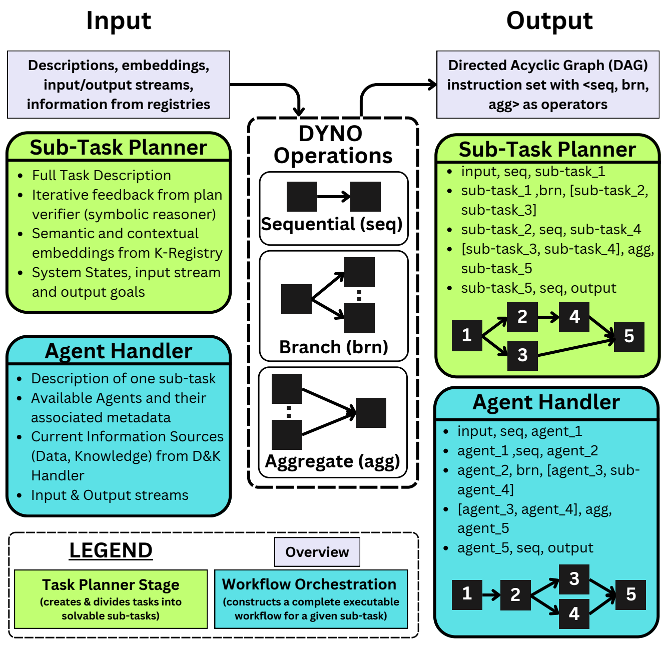

Input
Output
Descriptions, embeddings, input/output streams, information from registries
Directed Acyclic Graph (DAG) instruction set with <seq, brn, agg> as operators
DYNO Operations
Sequential (seq)
Branch (brn)
Aggregate (agg)
Sub-Task Planner
Full Task Description
Iterative feedback from plan verifier (symbolic reasoner)
Semantic and contextual embeddings from K-Registry
System States, input stream and output goals
Agent Handler
Description of one sub-task
Available Agents and their associated metadata
Current Information Sources (Data, Knowledge) from D&K Handler
Input & Output streams
Sub-Task Planner
input, seq, sub-task_1
sub-task_1 ,brn, [sub-task_2, sub-task_3]
sub-task_2, seq, sub-task_4
[sub-task_3, sub-task_4], agg, sub-task_5
sub-task_5, seq, output
1
2
3
4
5
Agent Handler
input, seq, agent_1
agent_1 ,seq, agent_2
agent_2, brn, [agent_3, sub-agent_4]
[agent_3, agent_4], agg, agent_5
agent_5, seq, output
1
2
3
4
5
LEGEND
Overview
Task Planner Stage
(creates & divides tasks into solvable sub-tasks)
Workflow Orchestration
(constructs a complete executable workflow for a given sub-task)

Figure 2: *Descriptive diagram of DYNO's operation set.* Sub-Task Planner leverages operations for creating a DAG of solvable sub-tasks from a given task, Agent Handler leverages operations for constructing DAGs of workflows for a given sub-task.

two interdependent modules: ***Sub-Task Planner***, which proposes task decompositions and ***Planner Verifier***, which refines and validates the decompositions using evolving ontological knowledge. This iterative process continues until a plan achieves both generative adequacy and symbolic validity or reaches user-set the iteration limit (K).

1. Sub-Task Planner

The Sub-Task Planner functions as the neural engine, powered by a LLM-based reasoning core. It decomposes the input task into a directed acyclic graph (DAG) of solvable sub-tasks, where nodes represent sub-tasks and edges denote dependencies constrained by symbolic operators (*seq*, *brn*, *agg*) as illustrated in Figure 2. By leveraging contextual, semantic embeddings and domain cues from the *Knowledge Registry* and iterative feedback from *Planner Verifier*, the planner aligns its generative reasoning with available knowledge assets, ontological context, and current system state.

2. Planner Verifier

The Planner Verifier serves as the symbolic reasoning component, validating the semantic and structural integrity of the sub-task plans. Implementation consists of either a rule-based engine or a hybridized architecture depending on the scope, strictness and constraints of the implemented domain to produce a deterministic reasoner for verification and iterative improvement of the generated plan by the sub-task planner. It references dynamic *plan ontologies* that encode permissible hierarchies, constraints, and dependencies. These ontologies evolve continuously through feeback from prior planning episodes and verified workflows of *Workflow Orchestration* stage. This adaptive ontology accumulation enables each plan to remain semantically grounded, operationally feasible, and aligned with available agents and data assets.

Once a verified plan is generated, the **Task Planner dispatches sub-tasks sequentially to *Workflow Orchestration* stage** for controlled workflow generation and verification. By integrating adaptive planning and symbolic grounding, this neurosymbolic mechanism underpins DYNO's goal of scalable, interpretable, and self-correcting multi-agent orchestration.

Stage-2: Workflow Orchestration

The ***Workflow Orchestration Stage*** receives a verified sub-tasks and associated plan from the Task Planner and is responsible for constructing a complete, executable, and evaluable workflow for that sub-task and then consolidating the constructed workflow into given plan. It consists of four interdependent components: ***Data and Knowledge Handler***, ***Heterogeneous Agent Handler***, ***Evaluation Assessor***, and ***Orchestration Verifier***. Together, they design, validate, and iteratively refine the control, data and knowledge flows necessary for successful orchestration.

1. Data and Knowledge Handler

The Data and Knowledge Handler is responsible for discovering, retrieving, and contextualizing the most relevant data and knowledge assets for the sub-task. This handler queries the *Data Registry* and *Knowledge Registry* to design the data and knowledge flow. It selects inputs (datasets, streams, indices), contextual knowledge (ontologies, schemas, graphs), and auxiliary artifacts (prompts, exemplars, retrieval views) that minimize information bottlenecks and maximize semantic coverage for the sub-task. Retrieved assets are annotated with provenance, version identifiers, and usage constraints to support traceability and reproducibility during downstream execution.

This handler communicates directly with agents through the **MCP (Model Context Protocol)** (Hou et al. 2025), which standardizes how data and knowledge assets are requested, transferred, and contextualized during orchestration. The MCP protocol ensures bidirectional synchronization between agents and information sources: agents can dynamically query or update contextual knowledge, while the handler maintains provenance, consistency, and usage constraints for each asset. This design guarantees that every agent operates over well-grounded, up-to-date information throughout the workflow.

2. Heterogeneous Agent Handler

The Heterogeneous Agent Handler manages agent selection, composition, and coordination for the current sub-task. It draws from the *Agent Registry*, which includes multiple classes of agents: reactive (tool-like), generative (LLM-based), symbolic (rule-based), etc. Agent selection follows an *importance-weighted strategy*, preferring *lightweight, deterministic, and statistically reliable algorithmic agents* whenever feasible, and escalating to heavier cognitive or neural agents only when task complexity or reasoning depth demands it.

This handler is responsible for designing the *control flow* of the sub-task by constructing an executable control graph governed by the constrained operator set (*seq*, *brn*, *agg*). To enable seamless inter-agent communication, DYNO employs **A2A (Agent-to-Agent) Protocol** (Google A2A 2025), which standardizes message passing, state synchronization, and dependency signaling between heterogeneous agents. A2A ensures that agents of diverse modalities and agent types can interoperate coherently without manual bridging, enabling compositional reasoning and decentralized collaboration.

Rationale for Dual-Handler Design The decision to maintain separate handlers for data/knowledge flow from agent/control flow is foundational aspect of DYNO's orchestration design. The Data and Knowledge handler focuses on information quality and sematic coverage whereas the Heterogeneous Agent handler focuses over agent selection and coordination. This separation offers key benefits:

1. *Parallel optimization paths:* Parallel handlers ensure optimization paths reach desired equilibrium without one taking precedence over another.
2. *Smaller search space:* Separated exploration reduces joint optimization permutations.
3. *Modularity and Re-usability:* Cached workflows can be reused for new domains by swapping data assets and vice-versa without full redesign.

Through standardized bidirectional interactions between the handlers enabled via dual-protocol system (*A2A* for intra-agent connectivity and *MCP* for agent–knowledge interfacing) over operation sets, DYNO's workflow orchestration stage ensures a fully bound orchestration graph with complete provenance metadata.

3. Evaluation Assessor

The resulting ***Intermediate Sub-Task Workflow*** generated from prior modules is augmented with suitable evaluation metrics that assesses performance at both component and system levels. DYNO employs a multidimensional evaluation approach grounded in the 14 principles of **C3AN framework** (Sheth et al. 2025): *reliability, consistency, alignment, analogy, abstraction, causality, instructability, reasoning, planning, grounding, explainability, attribution, interpretability, and safety*. Each dimension is operationalized through quantitative metrics such as accuracy, precision, and recall, and qualitative indicators such as coherence, contextual relevance, and causal soundness. Together, these measures provide a balanced assessment of DYNO's robustness, transparency, and alignment with responsible AI standards.

Beyond static evaluation, the Evaluation Assessor also serves as a ***checkpoint*** within the Iterative Improvement stage, where results are routed through two feedback loops: one enabling **active learning** over data and knowledge assets, and the other supporting **hyperparameter tuning** for adaptive agents. Further details are provided in the Iterative Improvement section.

4. Orchestration Verifier

This module checks the structural and logical validity of each generated sub-task workflow before integration into global workflow. The validation steps include:

1. *Data flow validity:* Ensuring type, schema, and modality compatibility across the workflow.
2. *Control flow validity:* Confirming correct operator usage (*seq*, *brn*, *agg*), absence of deadlocks, adherence to dual-protocols (A2A and MCP), and completeness of dependencies.
3. *Pipeline coherence:* Verifying seamless integration of the sub-task workflow into the global orchestration DAG while preserving alignment with the original root task plan.

Workflows that pass all checks are finalized for execution and merged into global workflow. If validation fails, the Verifier produces a structured diagnostic report identifying failure points, implicated agents or assets, and recommended corrective actions to the Task Planner. This feedback triggers an iterative refinement process analogous, to *iterative deepening*, where the sub-task is decomposed, re-orchestrated, and re-evaluated until a valid configuration is achieved.

Beyond Validation, this module serves as another ***checkpoint*** within the Iterative Improvement stage, where results are routed through two feedback loops: one enabling **meta-learning** of workflow orchestration, and the other supporting **replanning** under Task Planner tuning for adaptive agents. Further details are provided in the Iterative Improvement section.

Stage-3: Iterative Improvement

The iterative improvement stage enhance the DYNO pipeline with loops aimed at providing continual self-optimization. This optimization is enabled through feedback loops at two checkpoints - *Evaluations Assessment* and *Orchestration Verification*.

The feedback loops at evaluations checkpoint aim at optimizing the pipeline via results from evaluation suites, namely - *(i) Agent Hyperparameter Fine-Tuning*, *(ii) Active learning over Knowledge and Data*. The feedback loops at orchestration verification checkpoint aim at optimizing the pipeline at cognitive choke points of the system, namely task planning and workflow generation leading to formalization of *(iii) Workflow-Level Meta-Learning*, and *(iv) Ontology-Guided Plan Refinement (Re-Planning)* feedback loops.

Agent Hyperparameter Tuning

The results obtained from Evaluations Assessor module are used to optimize the performance of agents in two ways:

1. *Hyperparameter tuning:* Hyperparameters of individual agents tuned by bayesian optimization for optimal performance.
2. *Performative feedback for human judgement:* Analytical performance reports containing scores, calibrations and historical trace records would be sent to users to determine the necessity of either fine-tuning current agent or finding better replacements.

At the evaluation checkpoint, agent hyperparameter tuning is prioritized since Bayesian model training and agent optimization are comparatively less expensive.

Active Learning over Knowledge and Data

Results obtained from Evaluation Assessor module are used to optimize the selection of datasets and knowledge assets either for training or inference purposes. This adaptive process operates along two dimensions:

1. *Data refinement and augmentation:* Resampling to reduce bias and enhance representational diversity (Yang, Huang, and Crowley 2024).
2. *Knowledge and ontology refinement:* Improved symbolic representations selection for domain task and suggestive enhancements for representations via new entities, relations, or constraints derived from evaluation insights (Kim et al. 2025).

This loop consists of five steps: model training, scoring, sampling, human verification, and subsequent retraining.

Workflow-Level Meta-Learning

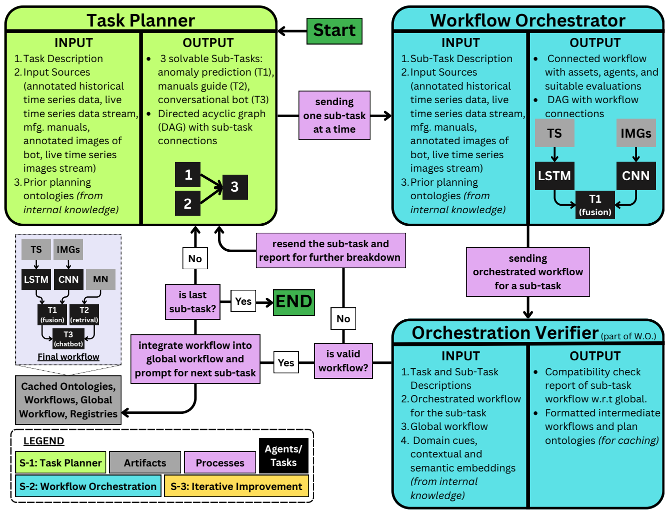
This loop caches generated sub-task workflows and their associated metadata, which serve as training substrates for meta-learning (Hospedales et al. 2021) within the Agent Handler. This process learns implicit correlations between data assets, agent configurations, and evaluation outcomes to generate priors that bootstrap new sub-tasks and enhance the system's understanding of workflow semantics.

Ontology-Guided Plan Refinement

This loop augments the plan ontologies with task oriented abstractions by capturing decomposition patterns, sub-task dependencies and associated success probabilities. The resulting refined ontologies serve as informed priors that guide **replanning** (Cushing and Kambhampati 2005) of the Sub-Task planner.

Illustrative implementation of DYNO

In this section, we present an illustrative example to demonstrate the implementation of DYNO (Illustrated in Figure 3).



Task Planner
INPUT
1. Task Description
2. Input Sources (annotated historical time series data, live time series data stream, mfg. manuals, annotated images of bot, live time series images stream)
3. Prior planning ontologies (from internal knowledge)
OUTPUT
3 solvable Sub-Tasks: anomaly prediction (T1), manuals guide (T2), conversational bot (T3)
Directed acyclic graph (DAG) with sub-task connections
1
2
3
Start
sending one sub-task at a time
Workflow Orchestrator
INPUT
1. Sub-Task Description
2. Input Sources (annotated historical time series data, live time series data stream, mfg. manuals, annotated images of bot, live time series images stream)
3. Prior planning ontologies (from internal knowledge)
OUTPUT
Connected workflow with assets, agents, and suitable evaluations
DAG with workflow connections
TS
IMGs
LSTM
CNN
T1 (fusion)
sending orchestrated workflow for a sub-task
resend the sub-task and report for further breakdown
No
is last sub-task?
Yes
END
No
is valid workflow?
Yes
integrate workflow into global workflow and prompt for next sub-task
TS
IMGs
LSTM
CNN
MN
T1 (fusion)
T2 (retrival)
T3 (chatbot)
Final workflow
Cached Ontologies, Workflows, Global Workflow, Registries
Orchestration Verifier (part of W.O.)
INPUT
1. Task and Sub-Task Descriptions
2. Orchestrated workflow for the sub-task
3. Global workflow
4. Domain cues, contextual and semantic embeddings (from internal knowledge)
OUTPUT
Compatibility check report of sub-task workflow w.r.t global.
Formatted intermediate workflows and plan ontologies (for caching)
LEGEND
S-1: Task Planner
Artifacts
Processes
Agents/ Tasks
S-2: Workflow Orchestration
S-3: Iterative Improvement

Figure 3: Illustrative diagram of Smart Manufacturing pipeline creation using DYNO.

Preface. Manufacturing industries in Industry 4.0 require an integrated, intelligent workflow to address interconnected challenges such as downtime, forecasting inaccuracies, and multimodal data interpretation that existing AI solutions treat in isolation.

Input Stream from User

Task Description: *"Develop a conversational chatbot that alerts an engineer when an anomaly occurs in the manufacturing pipeline, suggests relevant mitigation strategies from* *existing manuals, and can answer any engineer's questions related to the pipeline or the anomaly. The chatbot should also predict future anomalies using historical data."*

Input Sources (Data and Knowledge assets): Annotated historical time-series data, Live time-series data streams, manufacturing manuals, annotated images (from both anomalous and regular operations)

Stage-1 (Task Planner) Execution

The task description and input sources are processed and sub-modules of Task Planner (*Sub-Task Planner and Planner Verifier*) iteratively interact to generate a set of sub-tasks and their connections in form of a directed acyclic graph(illustrated in Figure 3):

- **Anomaly Prediction (ST-1):** Build an agent that uses multimodal sensor data to detect and classify anomalies in real time.
- **Manuals Guide (ST-2):** Build an agent that retrieves relevant mitigation steps from manufacturing manuals.
- **Conversational Chatbot (ST-3):** Integrate ST-1 and ST-2 within an interactive conversational interface.

The planner sends the sub-tasks sequentially to the Workflow Orchestrator.

Stage-2 (Workflow Orchestration) Execution

Workflow Orchestration receives sub-task and their objectives from task planner. For illustration, the process for ST-1 (Anomaly Prediction) proceeds as follows:

- **A. Data and Knowledge Handler:** Retrieves relevant data and knowledge assets for the sub-task and sends it to the *Heterogeneous Agent Handler*.
- **B. Heterogeneous Agent Handler:** Reviews available assets and sub-task requirements and proposes possible configurations (e.g., multimodal LLMs, CNNs, LSTMs, ViTs, Temporal Fusion Transformers, etc.).
- **Iterative refinement (A and B):** The iterative loop between A and B ensure optimal choice having input/output compatibility, computational efficiency, and semantic alignment. The system eliminates unsuitable options (e.g., LLM-based methods due to computational cost).
- **Workflow Generation:** The two-module framework finalizes CNN for image data and LSTM for time-series data as the optimal combination and sends the intermediate workflow to evaluation assessor.
- **C. Evaluation Assessor:** Adds relevant evaluation suites to the workflow (e.g., consistency, explainability, roc/mac, precision, accuracy).
- **D. Orchestration Verifier:** Validates the sub-task's workflow and integrates into global workflow.

This process repeats for all sub-tasks until the complete **Smart Manufacturing Pipeline** is constructed and verified. Once all the sub-tasks' associated workflows are constructed and integrated, the pipeline for smart manufacturing task would be marked as complete.

Conclusion

LaMAS marks a major step towards scalable, reasoning-driven systems, but their potential for enterprise adaptation requires navigating complex task coordination across diverse agents, multimodal assets, and evolving task semantics. Our proposed neurosymbolic orchestrator, **DYNO**, addresses this through a three-stage process: *task plan generation, workflow orchestration*, and *iterative improvement*. This division mirrors phases of reasoning, *generation, execution* and *reflection* and forms a closed cognitive loop. This position paper looks beyond the world of LLMs, agent paradigms and infrastructures to propose a holistic framework for generating complete pipelines for mission critical enterprise tasks, potentially serving as a foundational design principle for next-generation LaMAS. By unifying agents, data, knowledge and evaluation within a dynamic orchestration framework, DYNO moves the field closer to realizing interpretable, self-correcting, and human-aligned multi-agent intelligence.

References

Agostinelli, S.; Marrella, A.; and Mecella, M. 2020. Towards intelligent robotic process automation for BPMers. *arXiv preprint arXiv:2001.00804*.

Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Cemri, M.; Pan, M. Z.; Yang, S.; Agrawal, L. A.; Chopra, B.; Tiwari, R.; Keutzer, K.; Parameswaran, A.; Klein, D.; Ramchandran, K.; et al. 2025. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*.

Cheng, L.; Li, X.; and Bing, L. 2023. Is gpt-4 a good data analyst? *arXiv preprint arXiv:2305.15038*.

Cichocki, A.; Ansari, H. A.; Rusinkiewicz, M.; and Woelk, D. 1997. *Workflow and process automation: concepts and technology*, volume 432. Springer Science & Business Media.

Cushing, W.; and Kambhampati, S. 2005. Replanning: A new perspective. *Proceedings of the International Conference on Automated Planning and Scheduling Monterey, USA*, 13–16.

Eulerich, M.; Waddoups, N.; Wagener, M.; and Wood, D. A. 2024. The Dark Side of Robotic Process Automation (RPA): Understanding Risks and Challenges with RPA. *Accounting Horizons*, 38(2): 143–152.

Gao, M.; Li, Y.; Liu, B.; Yu, Y.; Wang, P.; Lin, C.-Y.; and Lai, F. 2025. Single-agent or Multi-agent Systems? Why Not Both? *arXiv preprint arXiv:2505.18286*.

Google A2A. 2025. A2A: A New Era of Agent Interoperability. https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/. Accessed: 2025-10-15.

Hong, S.; Zhuge, M.; Chen, J.; Zheng, X.; Cheng, Y.; Wang, J.; Zhang, C.; Wang, Z.; Yau, S. K. S.; Lin, Z.; et al. 2023. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Hospedales, T.; Antoniou, A.; Micaelli, P.; and Storkey, A. 2021. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9): 5149–5169.

Hou, X.; Zhao, Y.; Wang, S.; and Wang, H. 2025. Model context protocol (mcp): Landscape, security threats, and future research directions. *arXiv preprint arXiv:2503.23278*.

Hu, M.; Zhou, Y.; Fan, W.; Nie, Y.; Xia, B.; Sun, T.; Ye, Z.; Jin, Z.; Li, Y.; Chen, Q.; et al. 2025. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation. *arXiv preprint arXiv:2505.23885*.

Ivančić, L.; Suša Vugec, D.; and Bosilj Vukšić, V. 2019. Robotic process automation: systematic literature review. In *International Conference on Business Process Management*, 280–295. Springer.

Kim, D.; Yang, H.; Hwang, S.; Yeom, K.; Shim, M.; and Lee, K.-H. 2025. Active Learning Framework for Improving Knowledge Graph Accuracy. *IEEE Access*, 13: 47500–47513.

Li, Z.; Xu, S.; Mei, K.; Hua, W.; Rama, B.; Raheja, O.; Wang, H.; Zhu, H.; and Zhang, Y. 2024. Autoflow: Automated workflow generation for large language model agents. *arXiv preprint arXiv:2407.12821*.

Li, Z.; Zhang, H.; Han, S.; Liu, S.; Xie, J.; Zhang, Y.; Choi, Y.; Zou, J.; and Lu, P. 2025. In-the-Flow Agentic System Optimization for Effective Planning and Tool Use. arXiv:2510.05592.

OpenAI. 2025. API Agents — OpenAI. https://openai.com/agent-platform/. Accessed: 2025-11-02.

Qian, C.; Liu, W.; Liu, H.; Chen, N.; Dang, Y.; Li, J.; Yang, C.; Chen, W.; Su, Y.; Cong, X.; et al. 2023. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*.

Sheth, A.; Roy, K.; and Gaur, M. 2023. Neurosymbolic Artificial Intelligence (Why, What, and How). *IEEE Intelligent Systems*, 38(3): 56–62.

Sheth, A. P.; Roy, K.; Venkataramanan, R.; Nadimuthu, V.; and Shyalika, C. 2025. Composite AI With Custom, Compact, Neurosymbolic Models: The Emergent Enterprise Artificial Intelligence Paradigm. *IEEE Internet Computing*, 29(2): 37–49.

Wornow, M.; Narayan, A.; Opsahl-Ong, K.; McIntyre, Q.; Shah, N. H.; and Re, C. 2024. Automating the enterprise with foundation models. *arXiv preprint arXiv:2405.03710*.

Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; et al. 2024. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*.

Yang, C.; Huang, L.; and Crowley, E. J. 2024. Plug and Play Active Learning for Object Detection. arXiv:2211.11612.

Ye, Y.; Cong, X.; Tian, S.; Cao, J.; Wang, H.; Qin, Y.; Lu, Y.; Yu, H.; Wang, H.; Lin, Y.; et al. 2023. Proagent: From robotic process automation to agentic process automation. *arXiv preprint arXiv:2311.10751*.