ICRA2026 Late Breaking Results Poster presented at
2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Hybrid Agentic AI-FSM Framework for Instruction-Based Industrial Manipulation Tasks

Sungmoon Joo, Ikjune Kim

***Abstract*—This paper proposes a hybrid Agentic AI–FSM framework for robust natural-language-driven automation in safety-critical industrial robotics applications. Although natural-language procedures are commonplace in manufacturing, translating them into reliable robot programs remains labor-intensive. While Large Language Models (LLMs) offer strong parsing and planning capabilities, their inherent non-determinism and susceptibility to hallucinations preclude their direct use for robot control. To bridge this gap, our architecture employs an LLM-based planning agent to translate instructions offline into a structured task plan. Execution is then delegated to a deterministic Finite State Machine (FSM)-style execution engine to ensure reliability. Safety is further guaranteed by a multi-stage validation–simulation pipeline that verifies schema compliance and operational constraints through dry runs prior to deployment. For runtime anomalies, a RAG-enhanced Exception Handling Agent proposes recovery options, which are strictly mediated through a human-in-the-loop (HIL) interface for operator approval. Finally, a rule-based Safety Agent enforces physical constraints and provides an independent protection layer.**

I. Introduction

Industrial robots perform repetitive tasks in high-reliability, high-risk, and safety-critical environments, such as manufacturing, logistics, and nuclear power. The stability and reliability of these robots directly impact industrial productivity and safety. In most industrial settings, robotic tasks are provided as natural language documents, such as Work Procedures or Standard Operating Procedures (SOPs). Converting these into executable robot programs requires significant manual effort.

With recent advancements in Large Language Models (LLMs), there has been an increasing number of attempts to transform natural language instructions into structured robotic task representations. These developments have evolved rapidly, starting from early attempts to link natural language with robotic affordances [1] to Vision-Language-Action (VLA) models that directly generate control commands through integrated learning of vision, language, and action data. Prominent examples include the RT-2 [2] and Open X-Embodiment (RT-X) [3] projects. More recently, open-source models like OpenVLA [4] and others emphasizing "Physical Intelligence" [5] have emerged, expanding the possibilities for instruction-driven general-purpose robot control.

However, even these state-of-the-art VLA models still face fundamental limitations such as hallucinations, nondeterminism, lack of real-time performance, and unverifiability. Models like OpenVLA operate on an "End-to-End" black-box approach where the process from input (e.g. image/text) to output (control command) is handled entirely within a massive neural network. While useful in general environments, this approach struggles to meet industrial safety standards that require strict adherence to defined procedures and immediate root-cause analysis in case of malfunction. Conversely, traditional industrial robot control relies on deterministic structures like Finite State Machines (FSM) or Behavior Trees (BT), which ensure predictability and safety but lack the flexibility to automatically convert natural language documents

To resolve this dilemma, this study proposes a "Hybrid Agentic AI–FSM Framework" that combines the flexible interpretation capabilities of LLM/VLA with the reliability of traditional FSM. Unlike VLA, the proposed architecture adopts a modular approach. It decouples the LLM from the robot's real-time control loop, limiting its role to the offline tasks of "document parsing and procedural reasoning." Actual control is handled by a verified FSM-style execution engine. Generated procedures are executed safely only after passing through schema validation, constraint-based verification, and simulation-based dry runs.

In particular, when a runtime exception occurs, a RAG-based "Exception Handling Agent" does not perform recovery autonomously; instead, it "proposes" recovery options to the operator. Only tasks approved by the operator (Human-in-the-Loop, HIL) are resumed by the execution engine, while an independent "Safety Agent" enforces physical constraints. This approach eliminates the opacity of end-to-end language-based action models and ensures the explainability and structural safety essential for industrial environments.

II. Hybrid Agentic AI Framework

The framework proposed in this study combines Agentic AI structures [7–9] with deterministic execution control [10–11] to simultaneously satisfy LLM reasoning capabilities and robotic safety and real-time requirements.

A. Planning Agent & Task Hierarchy

Referencing existing LLM-based robot control research (SayCan, PaLM-E, RT-2), this study decouples the LLM from the robot control loop, utilizing it for "offline document parsing and procedural structuring." The LLM's role is strictly limited to converting natural language procedures into task plan that the execution engine can interpret, using constrained decoding and schema-based formatting.

To enable structured task planning from natural language instructions, we introduce a hierarchical task representation

Authors are with Korea Atomic Energy Research Institute, Daejeon, 34057 Republic of Korea (corresponding author, phone: 042-866-6326; e-mail: smjoo@kaeri.re.kr).

that decomposes robot actions into three levels: composite tasks, unit tasks, and primitive actions. A composite task represents a high-level objective that can be achieved through a sequence of unit tasks, each of which corresponds to a semantically meaningful and executable subtask with well-defined preconditions and termination criteria. At the lowest level, primitive actions denote atomic control operations that directly interface with the robot's motion and manipulation capabilities. Based on this hierarchy, we design a schema-driven planning agent in which a language model parses natural language instructions and generates a task plan that conforms to the predefined hierarchical structure. The use of an explicit schema ensures that the generated plan is both syntactically valid and semantically grounded, enabling reliable execution by downstream modules such as finite state machines and control policies.

B. Validation Layer

The Validation Layer verifies the validity of the task graph generated by the Planning Agent. This includes schema checks, rule-based constraint checks (e.g., action sequences, parameter ranges), and parameter sanity checks. Task graphs that pass validity verification can then undergo physical validation through simulation-based dry runs. This stage, which can be applied selectively, pre-verifies the LLM-generated procedures (e.g., paths, grasp poses) for anomalies such as collisions, joint-limit violations, or dynamics-limit breaches in the actual environment.

C. Execution Engine

The Execution Engine is responsible for the deterministic orchestration of the parsed task plan. It does not directly perform robot actions; instead, it interprets the validated task graph step by step, manages the execution state, and invokes the appropriate actor for each unit task.

Conceptually, the Execution Engine can be implemented using a Finite State Machine (FSM) or an equivalent deterministic task-sequencing mechanism. By design, the Execution Engine operates at the task-logic level. It determines what should be executed next and when execution should transition, but it does not define the low-level embodiment-specific commands required to physically perform the task.

D. Monitoring Agent

During and immediately after the execution of a unit task, the Monitoring Agent monitors robot sensor feedback (e.g., gripper force/position sensors, end-effector cameras) and robot state in real-time. As proposed in previous execution monitoring studies [18], this agent compares and analyzes physical sensor data against the logical task state.

Unlike the Safety Agent, which monitors physical limits like collisions or over-speeding, the Monitoring Agent judges success or anomalies based on task logic. For example, even if the engine performs a "Pick" action, the Monitoring Agent defines and detects "exceptional situations"—scenarios that are not safety accidents but clear task failures—such as a "grasp failed" signal from the gripper sensor, an object remaining in the gripper after a "Place" action, or task duration significantly exceeding expectations. Once an exception is detected, the agent transitions the engine to an error state and calls the Exception Handling Agent.

E. Exception Handling Agent

The Exception Handling Agent references troubleshooting manuals via RAG [7-9] to generate recovery options (e.g., retry, modify parameters, skip task, abort). These options are not executed autonomously but are proposed to the operator through an HMI (Dashboard). If the operator approves a proposed option or chooses a separate recovery plan, the step passes through the Validation-Simulation Pipeline again and returns to the execution engine. If the operator selects "Abort," the engine safely terminates the task in a predefined failsafe mode.

F. Safety Actor Layer

The Safety Actor is a final safety device independent of the execution engine and the LLM. It is distinct from LLM-based guardrails; while the Validation Layer is a software filter for Planning Agent output, the Safety Agent is a physical filter ensuring safety based on robot behavior, such as workspace boundaries, maximum speed/torque limits, and collision avoidance. It monitors the robot in real-time to ensure it does not deviate from physical constraints. Even if an incorrect "command" not caught during logical verification is performed, the Safety Actor blocks physical execution, preventing system damage and accidents [19-20].

E. Actor Layer

To explicitly separate task logic from physical execution, the proposed framework introduces an Actor Layer, also referred to as the Skill Executor Layer. This layer performs the actual unit tasks requested by the Execution Engine. Each actor corresponds to a primitive or semi-primitive skill, such as move, pick, place, inspect, or wait.

An actor receives a validated action specification from the Execution Engine and translates it into executable robot-level commands. In other words, the Actor Layer determines how a step is physically executed. This separation offers several advantages. First, the Execution Engine remains simple, since embodiment-specific execution details are delegated to the actors. Second, actor modules can be reused across multiple task plans. Third, simulation and real-robot deployment can share the same task logic while using different actor implementations.

III. SUMMARY & CONCLUSION

In this paper, we proposed a reliable "Hybrid Agentic AI–FSM framework" for performing robotic tasks based on natural language instructions in industrial settings. The proposed architecture provides a practical solution for safely utilizing LLMs even in safety-critical industrial environments by combining the superior natural language interpretation of LLMs with the deterministic safety of traditional FSM control with human-in-the-loop validation.

ACKNOWLEDGMENT

This work was supported by the Ministry of Trade, Industry and Energy, Republic of Korea, under Grant No. RS-2024-00416440