사단법인 한국원자력학회
KOREAN NUCLEAR SOCIETY

March 6 - 8, 2026 | ICC, JEJU

KAERI Korea Atomic Energy Research Institute

Empirical Evaluation of Single-Agent and Multi-Agent LLM Systems for Nuclear Reactor Control

Joowon Cha

Korea Atomic Energy Research Institute
University of Science and Technology

Presentation Outline

1 Introduction

2 System Implementation

3 Experiments & Results

4 Discussion

5 Conclusion

Introduction

Background · AI Technologies · Prior Work · Research Objective

Background

Growing Adoption of AI in Nuclear

- IAEA encourages using AI to reduce human mistakes (IAEA NES, 2022)
- IAEA NR-T-1.26 (2025): first official guide for using AI in nuclear — *"early-stage adoption"*
- LLMs are growing into AI agents — they can now think, plan, and carry out complex procedures

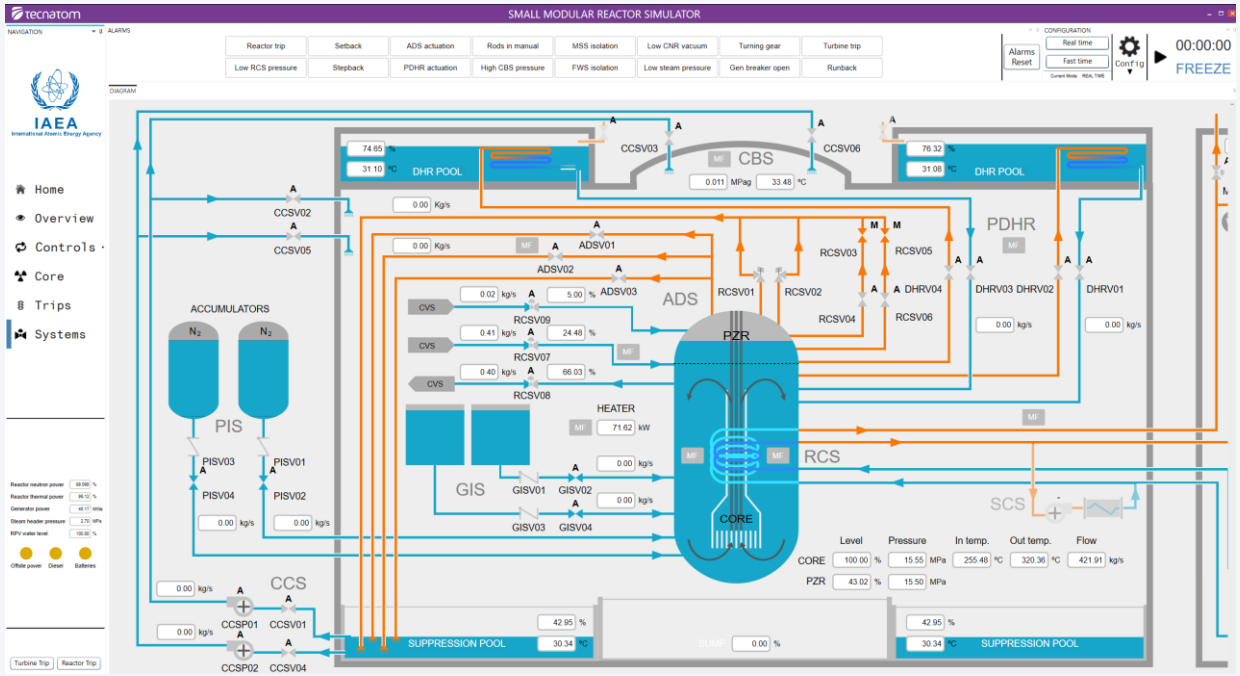
Research Gap

- Existing AI research in nuclear focuses on prediction and anomaly detection
- Using AI agents to actually run procedures step by step is still in early stages
- **No study has compared Single-Agent and Multi-Agent** systems for running nuclear procedures

Therefore, in this work —

We build and test both Single-Agent and Multi-Agent systems to control an iPWR simulator using Agentic AI

What We Control: iPWR Simulator



IAEA iPWR Simulator

Developed by Tecnatom (2017), distributed by IAEA Basic principle simulator for education and training

Key Specifications

Thermal output **150 MW** / Electrical output **45 MWe** Primary system + BOP behavior simulation Passive safety systems (PDHR, Pressurizer, GIS) Real-time control of control rods, valves, boron concentration, etc. **DLL-based memory address access**

AI Technologies Used in This Work

LLM (Large Language Model)

- AI model that reads and writes natural language
- It can read procedure text and think step by step — just like an operator following a manual
- On its own, it has no memory, cannot use tools, and cannot act by itself

Agentic AI

- Adds tools, memory, and a control loop to an LLM — so it can carry out tasks on its own
- **Reasoning** — interprets procedure steps and decides actions
- **Memory** — tracks simulator state and completed steps
- **Tool Use** — sends control commands directly to the simulator

Function Calling

- A fixed format the agent uses to send commands to the simulator
- Translates the LLM's natural language decisions into executable simulator actions
  e.g., ***withdraw_control_rod(bank="B", steps=5)***

RAG (Retrieval-Augmented Generation)

- Stops the AI from making up steps — it looks up and follows the real procedure documents
- Procedure PDFs are saved in advance; the right steps are pulled out when the agent needs them
- **Like an operator checking the manual before each step**

In our system, these four technologies work together: the LLM looks up procedures, decides what to do, and sends commands to the simulator.

Prior Work & Motivation

Our Group's Prior Work — Lee et al. (Nucl. Eng. Tech., 2025, KAERI)

We demonstrated that a single-agent LLM with RAG and function calling can control an iPWR simulator — handling abnormal situations (96.67%) and creating procedures (100%)

→ Based on our previous work, we now ask: can a Multi-Agent system do better than our single-agent system?

Theoretical Concerns with Single-Agent Architecture

Can One Agent Handle Everything?

- One agent must handle everything — from reading input to taking action
- This may lead to overload — could critical steps be missed?

No Built-in Way to Check Its Own Work

- If a mistake is made, will there be any way to catch it?

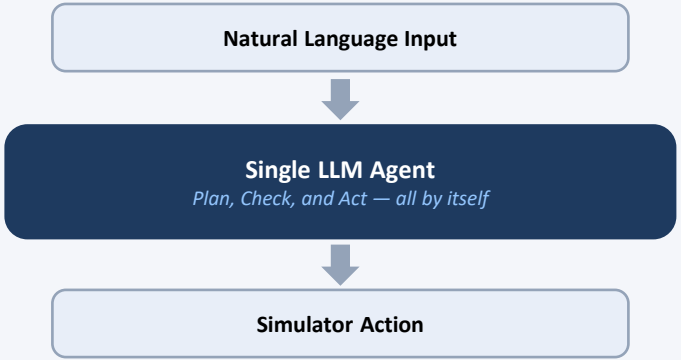
Risk of Undetected Wrong Outputs

- The AI may produce outputs that sound right but are actually wrong
- Without a dedicated checking step, such errors may go unnoticed

→ Do these concerns actually affect performance? We run experiments to find out.

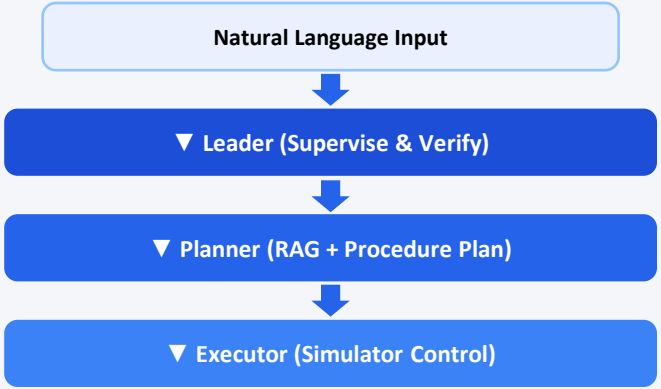
Two Approaches and Their Trade-offs

Single-Agent



✓ Simple, low cost, fewer API calls
✗ Cognitive overload · No cross-verification · Hallucination risk

Multi-Agent



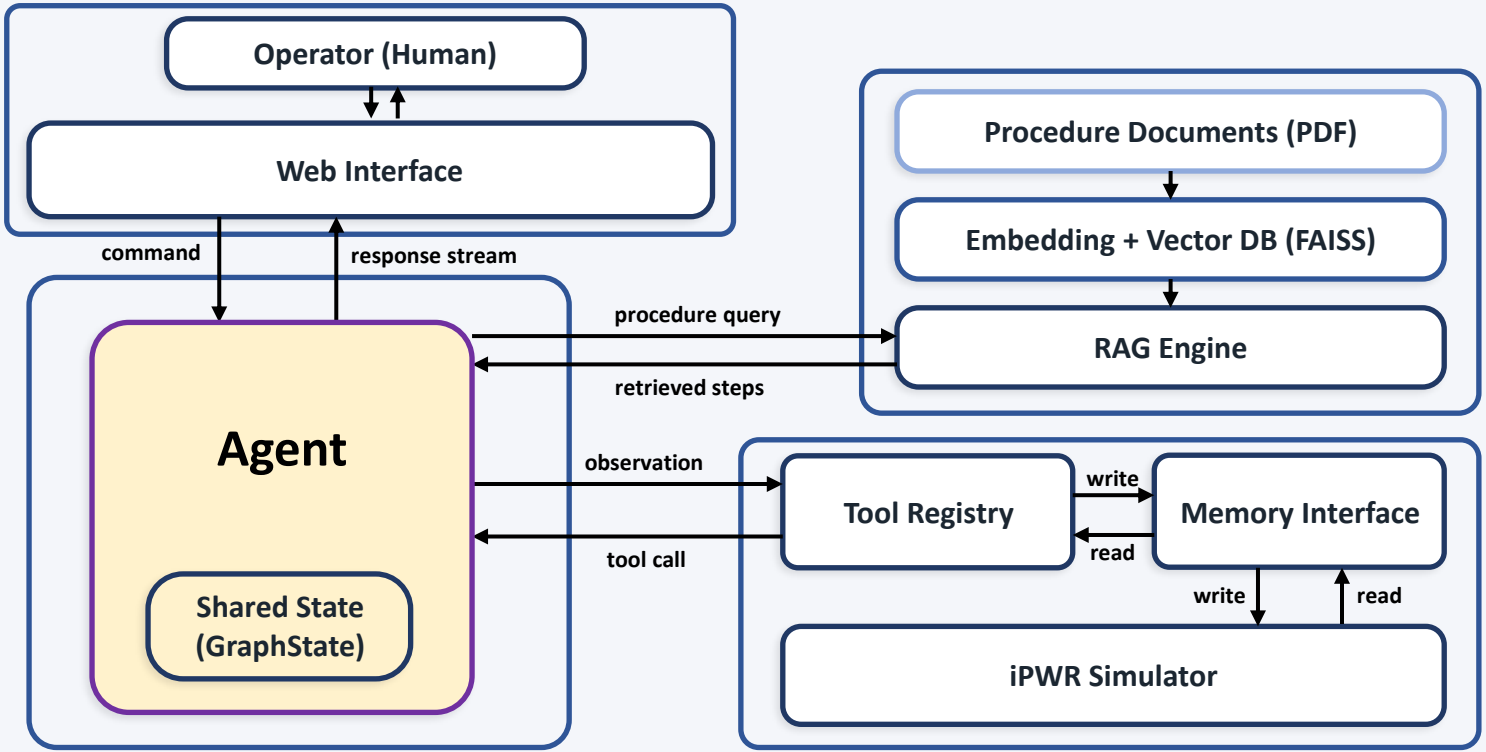
✓ Role separation · Cross-verification · Error recovery
✗ Higher token cost · Communication overhead

Nuclear operation requires multi-step verification & procedure compliance → ideal testbed for this comparison

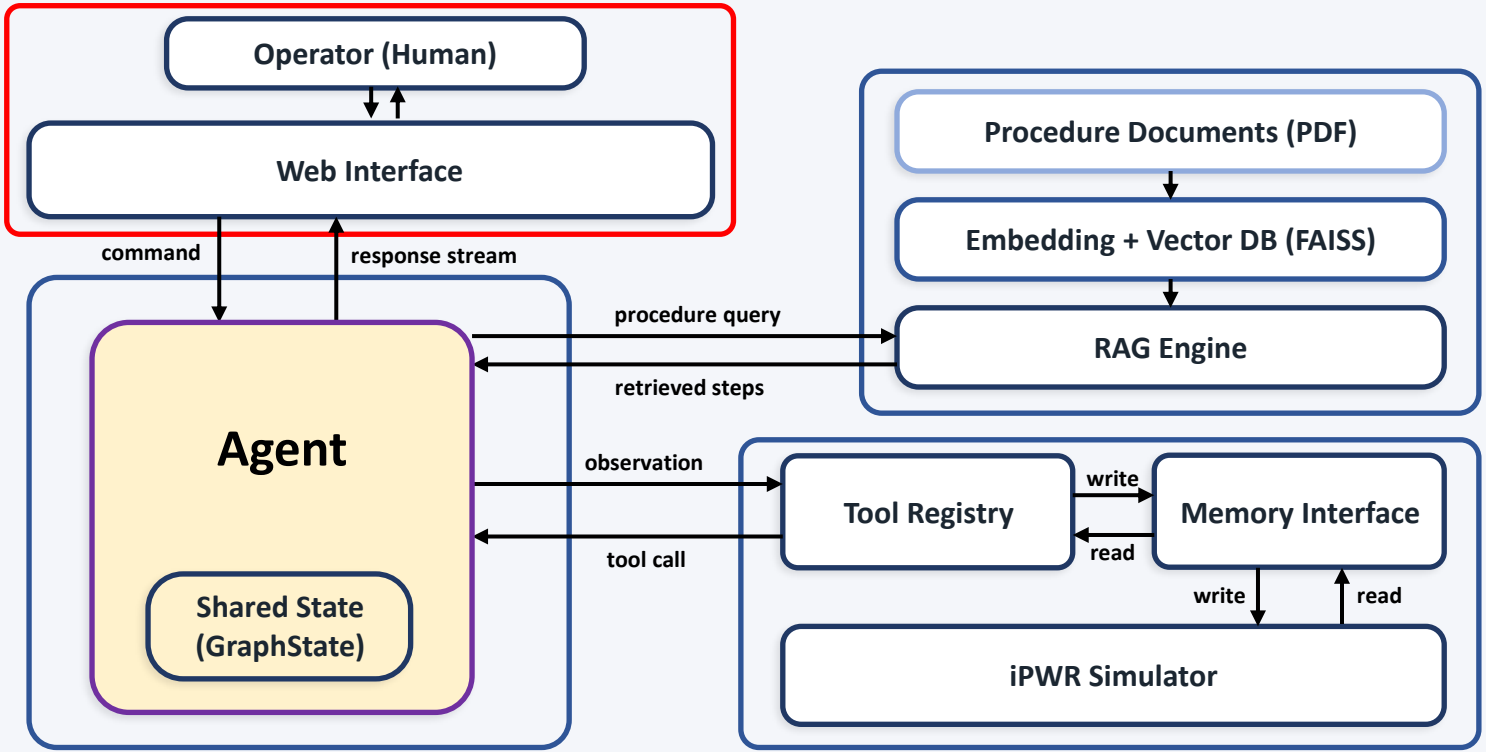
System Implementation

Overall architecture, Single-Agent and Multi-Agent design

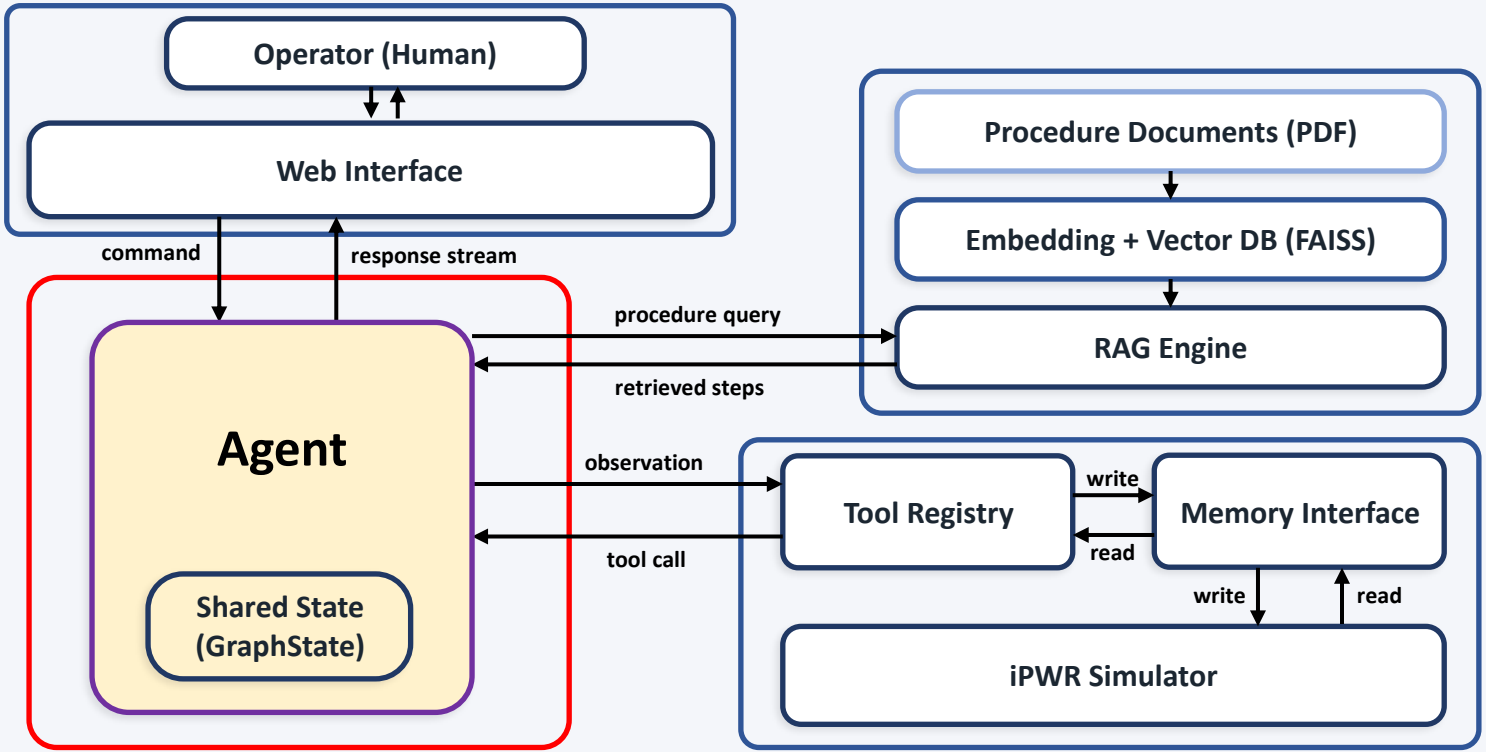
Overall System Architecture



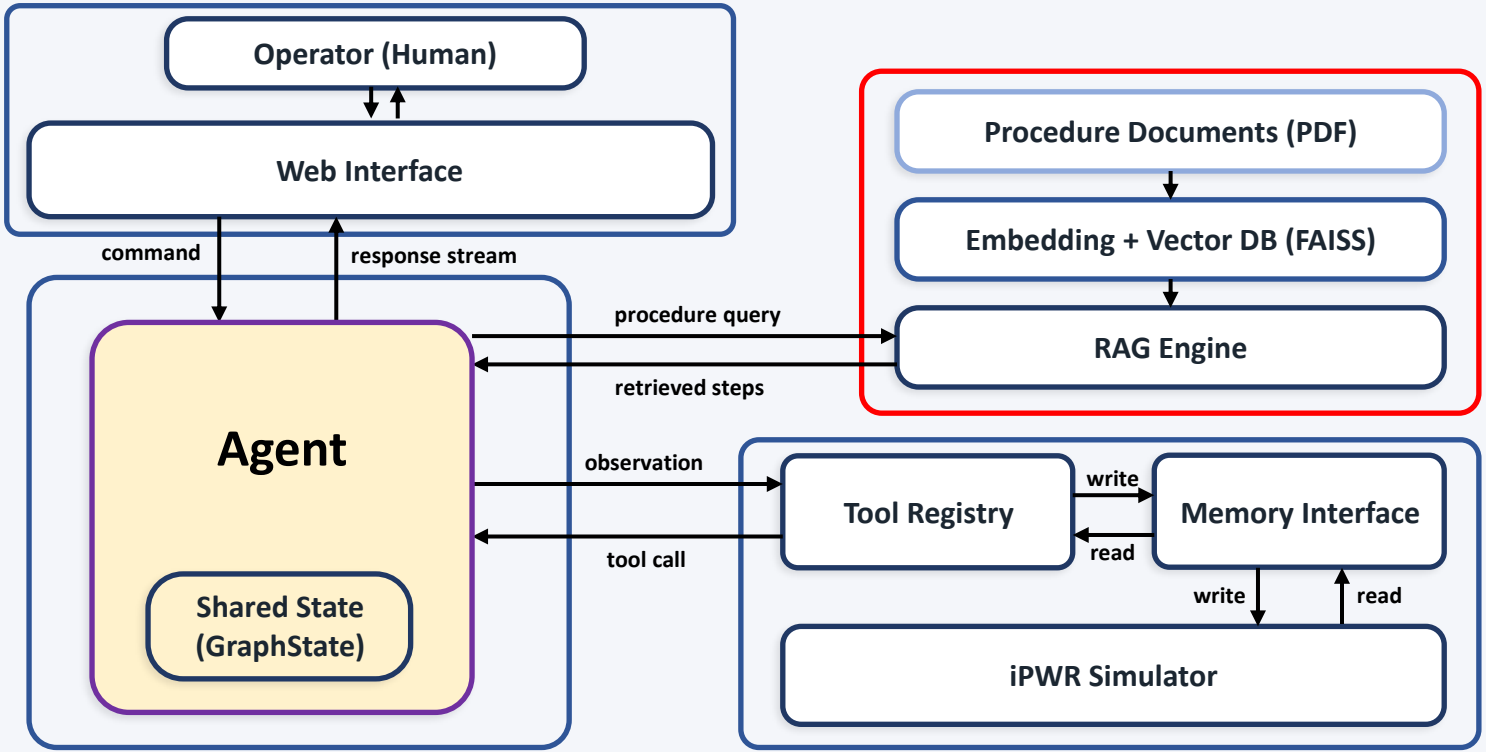
Overall System Architecture



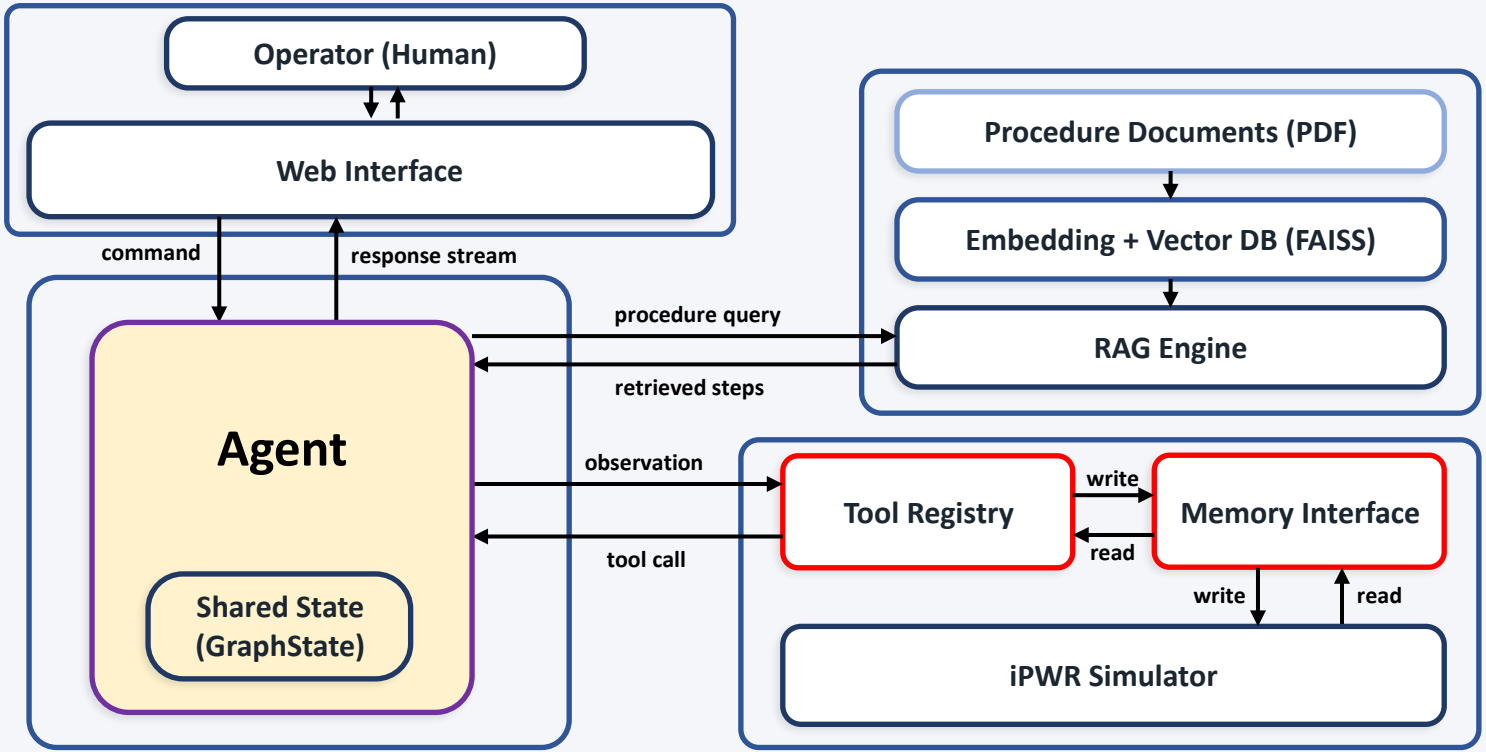
Overall System Architecture



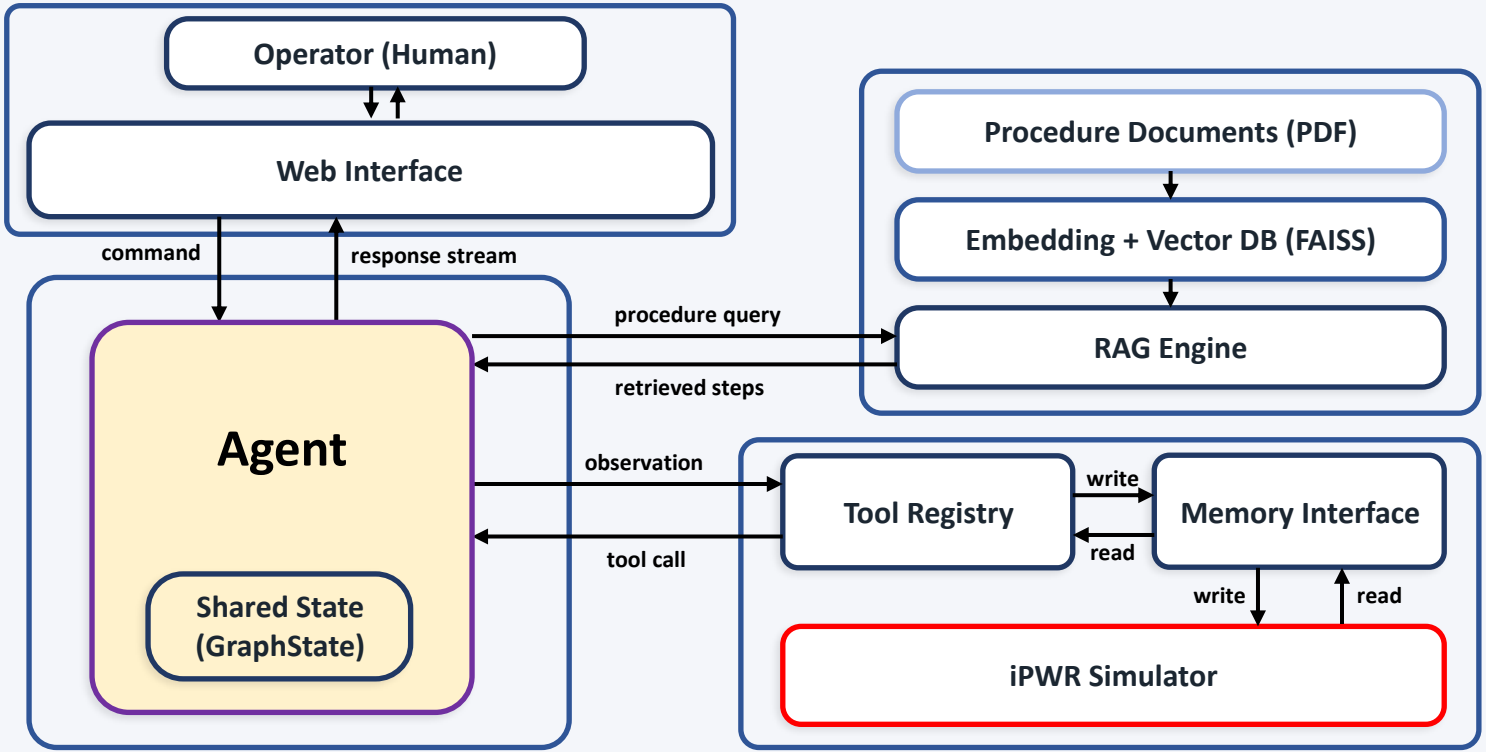
Overall System Architecture



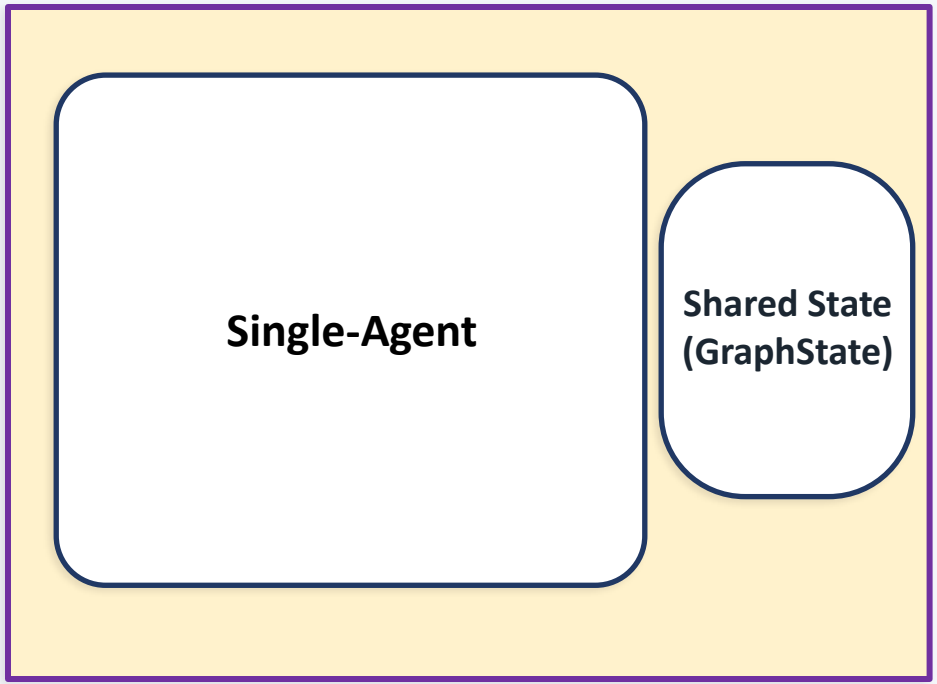
Overall System Architecture



Overall System Architecture



Single-Agent: Unified Architecture

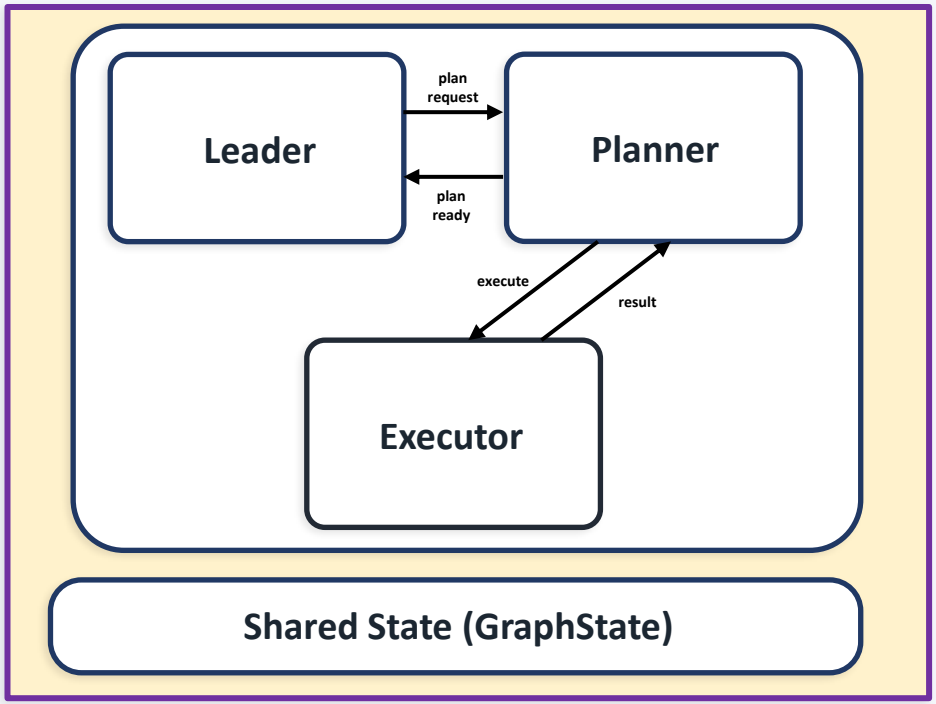


Operation Mechanism

A single agent adopts the role of an operator persona and internally alternates between Leader, Planner, and Executor roles.

The implementation is simple and maintains a unified context.
However, there is **no structural separation for independent verification** between planning and execution stages.

Multi-Agent: Role-Separated Architecture



Leader

Responsible for overall supervision and safety verification
Rigorously reviews plans, including physical control actions

Planner

Retrieves relevant procedures using RAG
Translates natural language into structured execution plans

Executor

Applies the approved plan to the simulator
Logs results and reports back to the Leader

Experiments and Results

Repeated experiments based on the iPWR simulator

Experimental Setup

Experimental Environment

Model	GPT-4o (2024-08-06)
Temperature	0.3
Framework	LangChain 1.2.8 + LangGraph 1.0.7
Python	3.12.12
Trials	100 runs per configuration

Scenario

IAEA iPWR Reactor Trip and Restart

Manual trip → Restart to full power
Steps 1–26 (66 sub-steps)
Multi-step procedure with explicit verification criteria
Suitable for architecture comparison

What We Measure — 6 Metrics

Task Completion Rate	success rate	**Token Usage**	I/O tokens	**API Cost**	LLM+RAG
Execution Time	total elapsed time	**RAG Embed Request Count**	RAG retrieval	**RAG Embed Tokens**	query embedding

Experimental Scenario: Reactor Trip and Restart

Procedure Flow (26 Steps, 66 Sub-steps)

Step 1–5: Initial condition check → Manual reactor trip → Trip verification

Step 6–11: Check main steam flow, pressurizer pressure, thermal power, boron, and Xe reactivity

Step 12–16: Trip reset → Control rod withdrawal (Bank A, B, C)

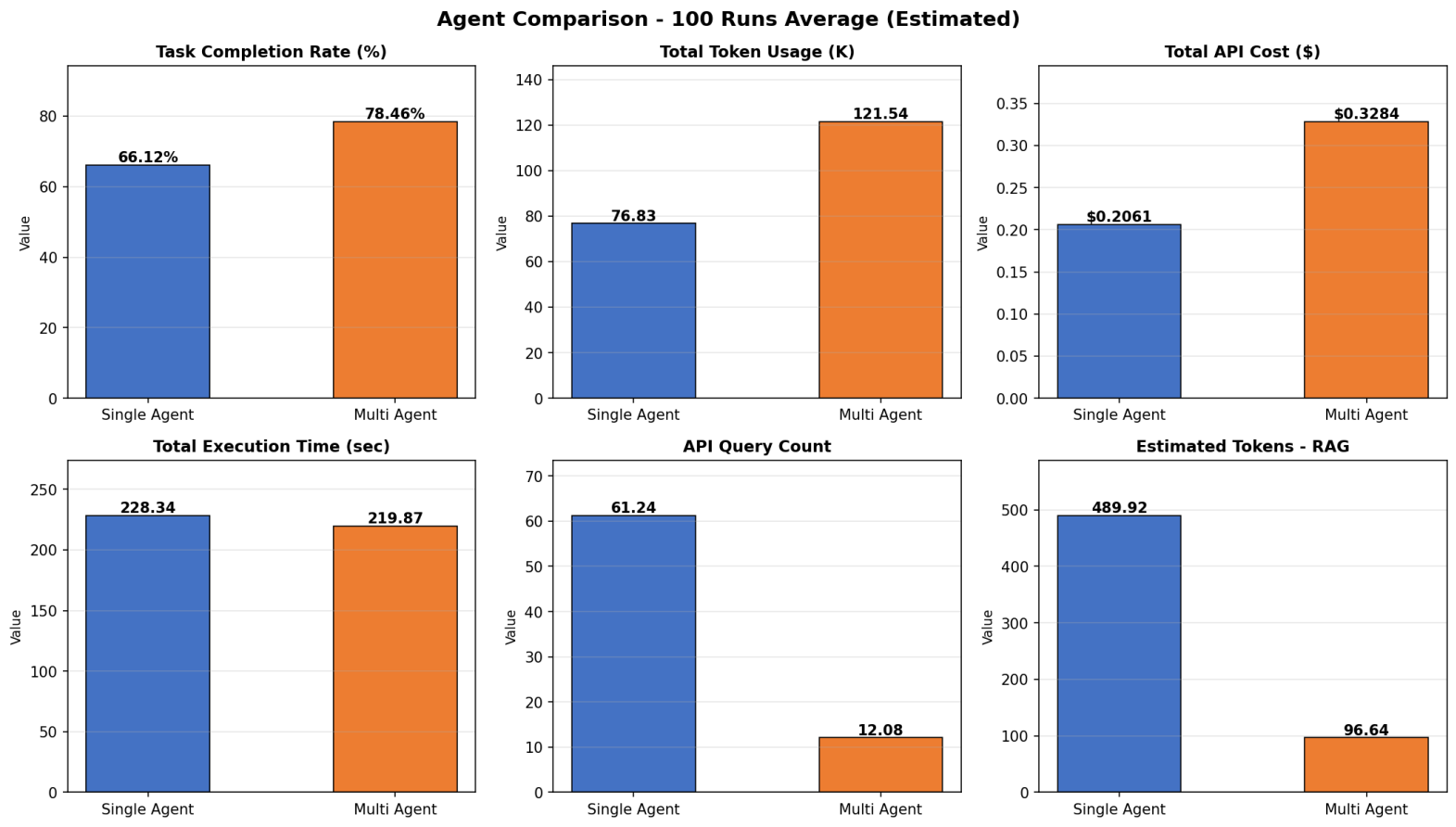
Step 17–19: Boron dilution → Reach criticality → Power increase (8% → 15%)

Step 20–26: Turbine startup → Load increase → Reach full power

Why This Scenario?

- Multi-step sequential procedure (66 sub-steps)
- Clear verification criteria at each step
- Hybrid structure combining physical control and decision-making
- Appropriate complexity to reveal differences between Single-Agent and Multi-Agent

Results: Average Comparison over 100 Trials



Discussion

How should we interpret the results?

Analysis and Limitations

Why does Multi-Agent achieve higher completion rate?

The Leader–Planner–Executor structure stops important steps from being missed

Cost goes up because agents send messages to each other — this was expected

Run time is similar because agents work at the same time

Unexpected Finding:

RAG lookups dropped by 80% because agents share information

— Single: retrieves procedures at every step

— Multi: **retrieves once and shares**

Limitations of This Study

100 runs per system — more runs are needed to be statistically reliable

We only tested one agent setup — other setups should also be compared

Only Reactor Trip & Restart used — validation on diverse scenarios required

Conclusion

Conclusion and Implications

Multi-Agent achieves 78.46% completion rate (vs. 66.12%) and reduces RAG queries by 80%

Reliability

Step-by-step checking and clear roles stop steps from being skipped — this fits nuclear safety needs

Trade-off

Higher cost but better reliability — our experiment clearly shows this trade-off

Future Work

Testing on more scenarios, improving the agent setup, and finding ways to lower cost

Thank You

This research was supported by a grant from Korea Atomic Energy Research Institute (KAERI) (No. KAERI-526140-26).