Korean Nuclear Society Autumn Meeting
Changwon, Korea October 30-31, 2025

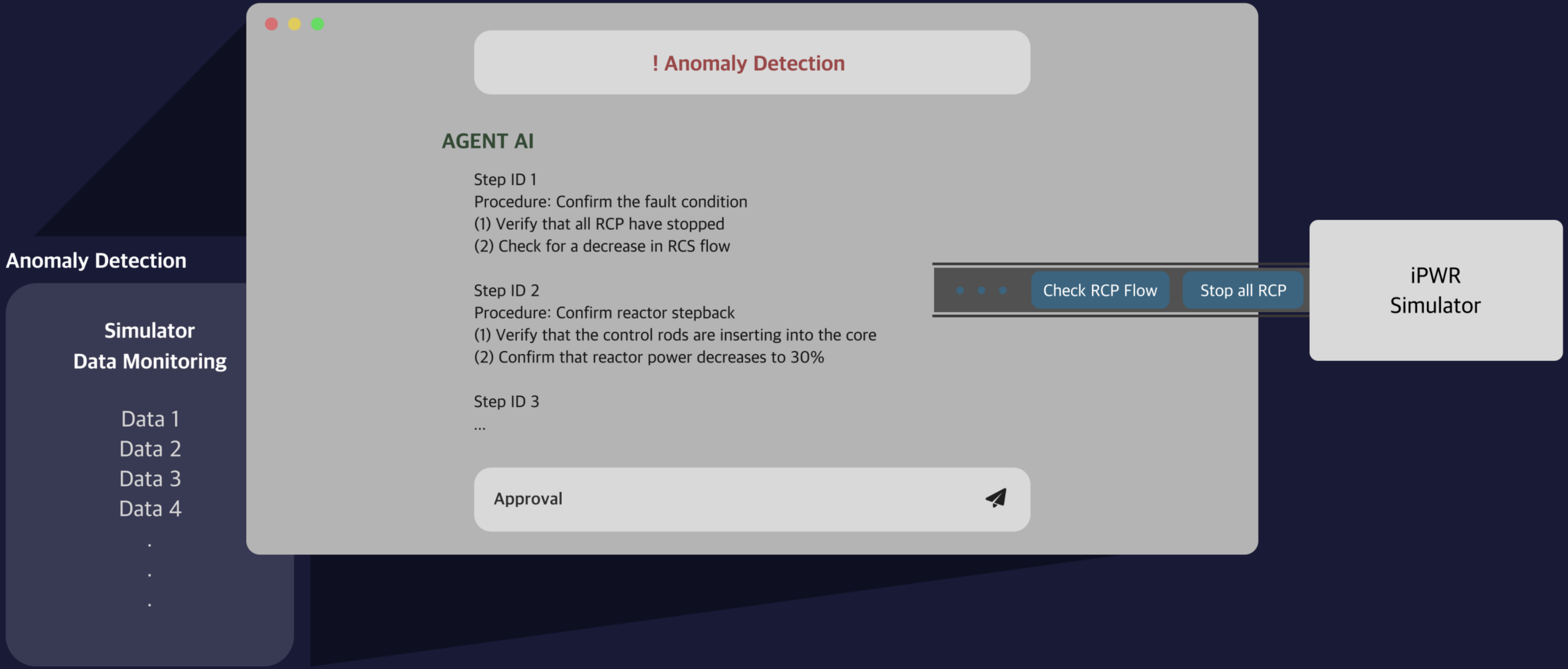
Enabling Robust Parallel Tool Calling in Agentic AI via OS-Level Scheduling for the iPWR Simulator

Keywords : Model Context Protocol, parallel tool calling, agentic AI, OS-level scheduling, iPWR simulation

Seongsu Chae [a, b], Yonggyun Yu [a], Seung Geun Kim [a], Yujong Kim [a*]

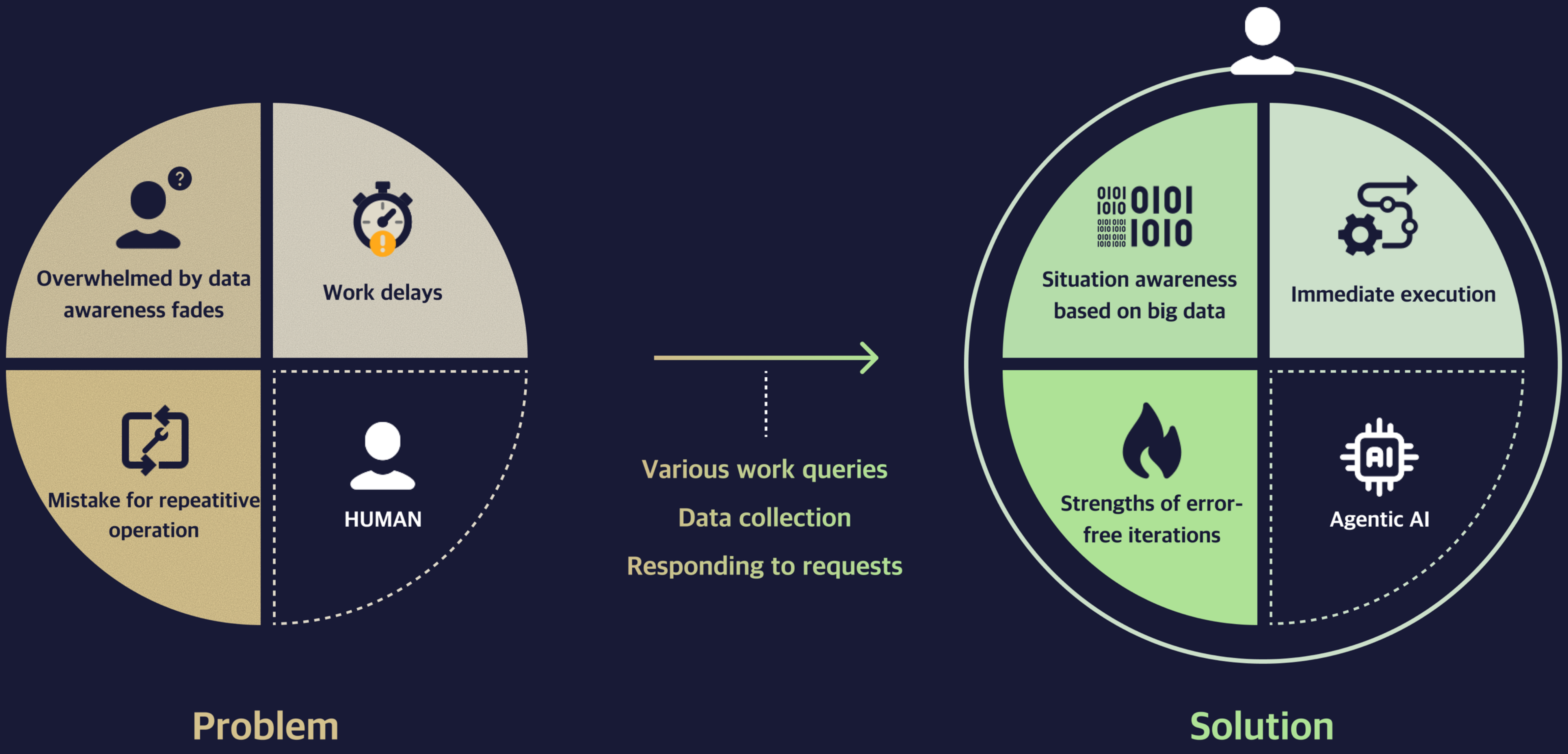
[a] Applied Artificial Intelligence Section, KAERI, Daejoen 34057, Republic of Korea
[b] Department of Computer Engineering, Hanbat National University, Daejeon 34158, Republic of Korea

CONTENTS

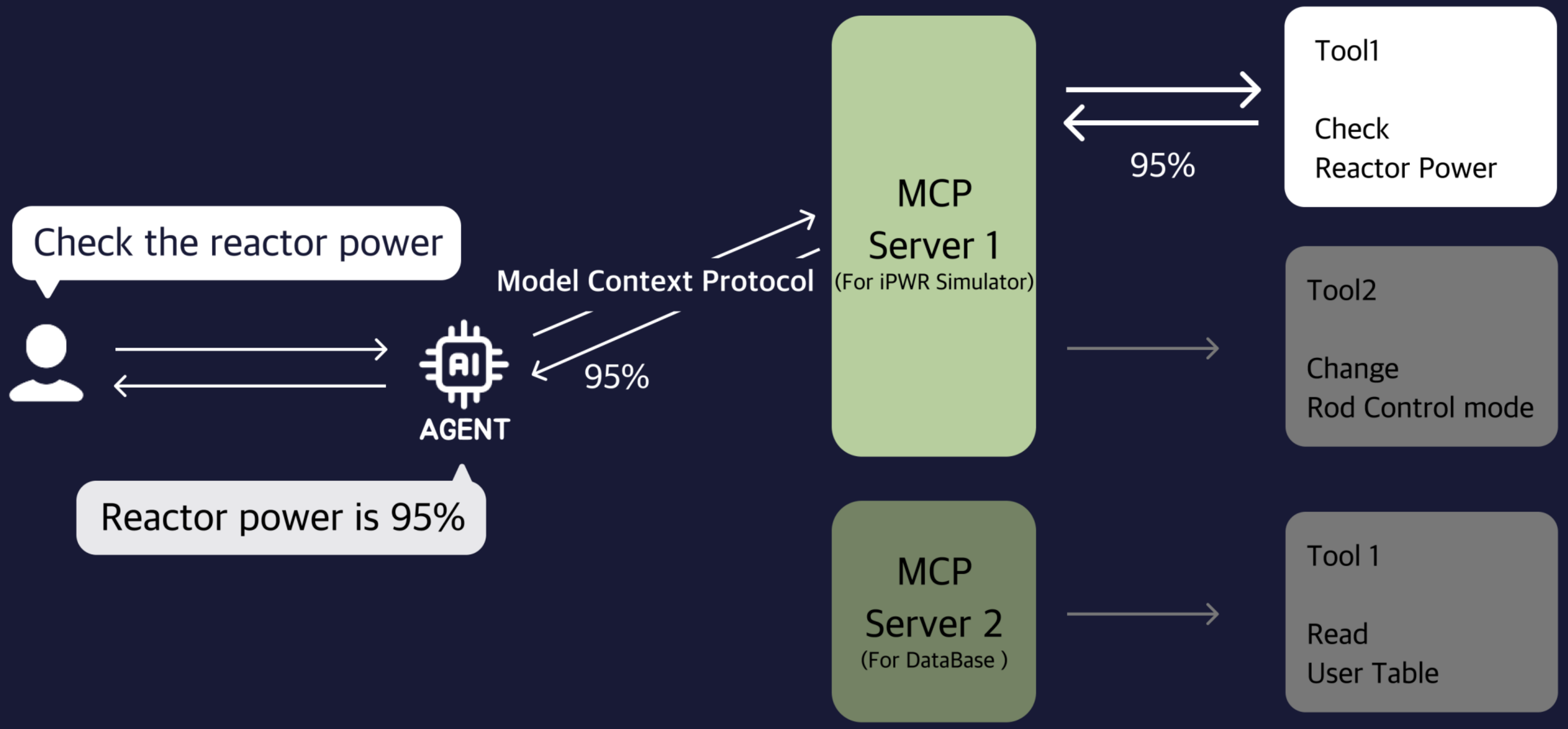


Data monitoring → Background anomaly detection → Refer to action documentation → AI summarization and synthesis → Invoke action functions → iPWR simulator control

Background & Objectives

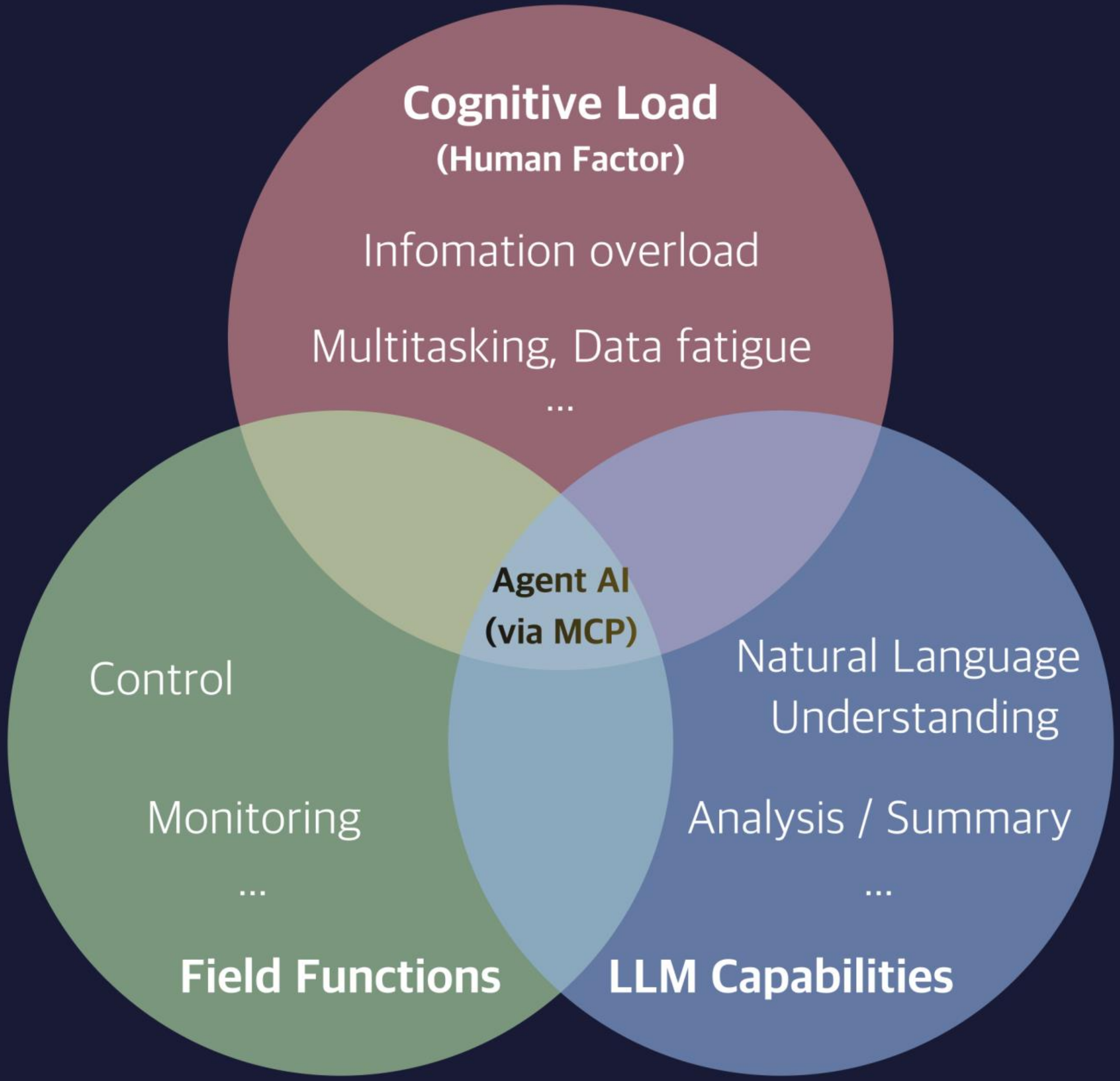


Overview of the Model Context Protocol (MCP)

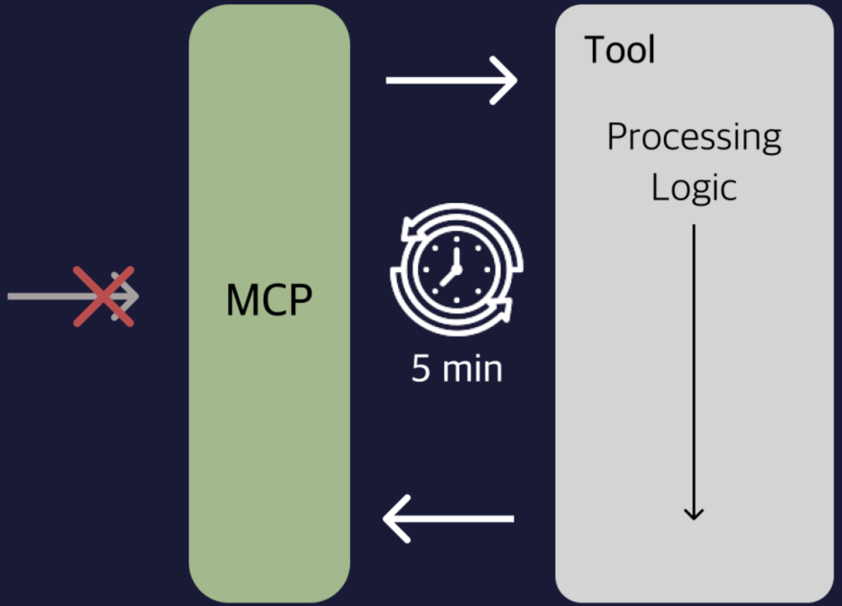


“MCP creates a standardized bidirectional connection for AI applications, enabling LLMs to easily connect with various data sources and tools.”
-Google Cloud-

Why Agent AI Needs MCP in Nuclear Power Plant Operations

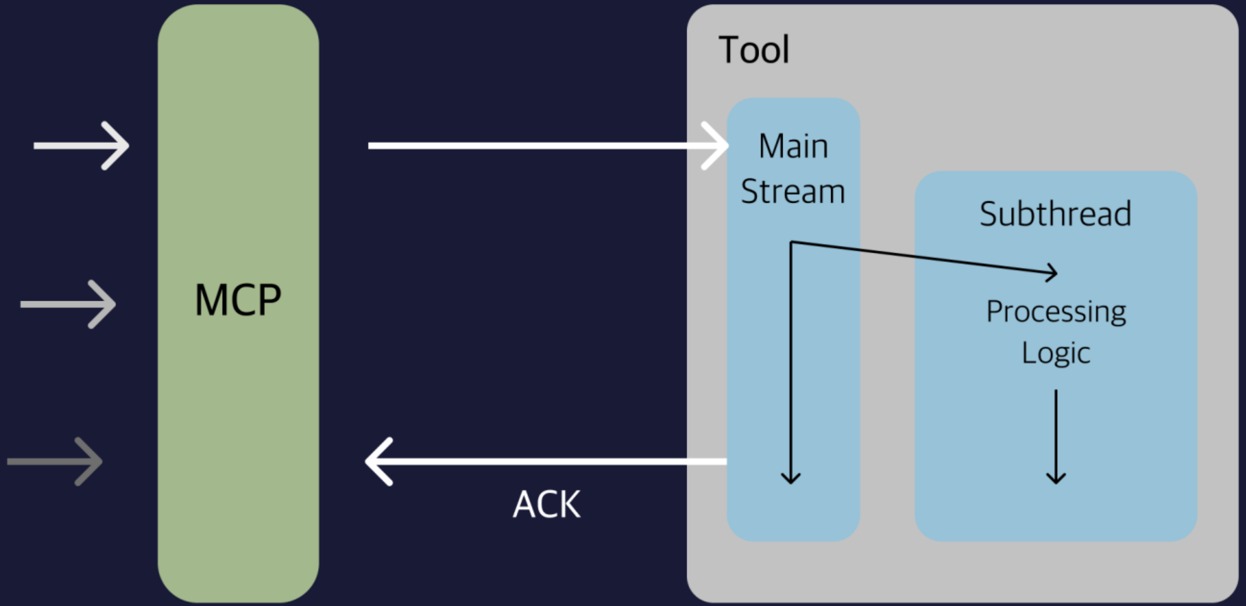


Limitations of Synchronous Operation & the Parallelization Hypothesis



Limitations of Synchronous Operation

PAIN POINT
"Waiting to handle other requests during
a long-running operation."



Solution Idea

HYPOTHESIS
"If the request handling and processing logic are isolated,
parallel execution can be achieved."

Evaluation Criteria

Qualitative Evaluation

Property	Evaluation Criterion	Reason for selection
Responsiveness	Upon a long-running request, is an immediate accept(ACK) returned and does subsequent interaction continue without dropped responses or perceived pause?	Interaction pauses delay checking and judgment during operations.
Non-blocking	While a long-running task is executing, are other requests remain continuously accepted and get processed without perceptible delay?	Operations require concurrent tasks; blocking impairs procedural execution and reduces safety margin.
Continuity	Do long-running tasks persist without interruption and complete reliably through termination?	Ensures trustworthiness of prolonged operations and a consistent process flow.

Experiment

- Subthread
- Subprocess
- Session Management Process Termination

Scenario

01
Request an indefinite task (Monitoring) from the AI

Start a long-running background operation

02
Send multiple short and long tasks during execution

Verify if parallel requests are processed properly

03
Check for missing or delayed responses

Ensure no response loss or delay occurs

04
Confirm completion status of all tasks

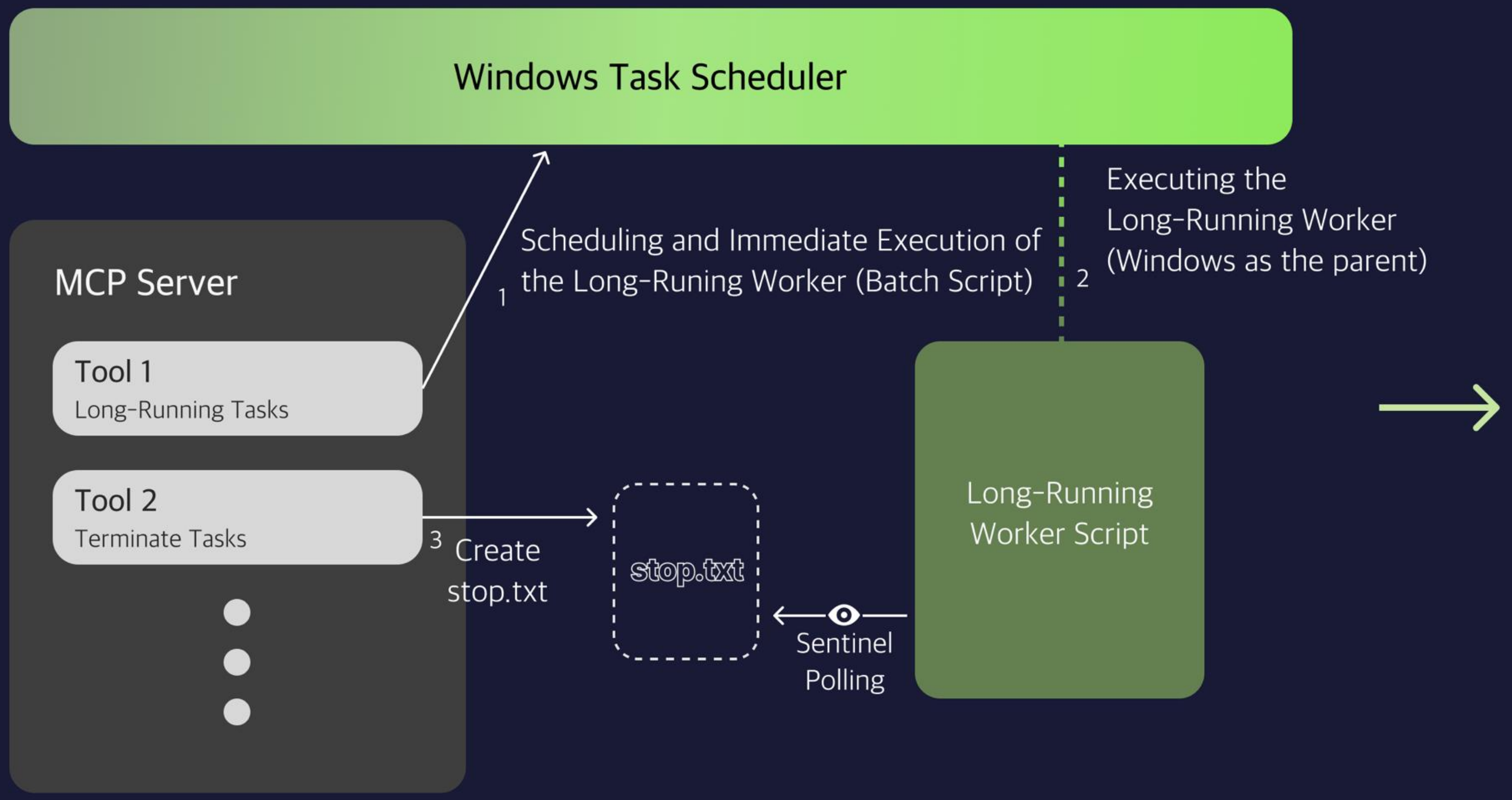
Verify all requests finish correctly without interruption

Experiment Result

Property	Responsiveness	Non-blocking	Continuity	Result Description
Subthread	✓	✓	✗	Session cleanup under a host idle timeout causing MCP server termination and co-disappearance of main flow and child threads → continuity degradation of long-running tasks.
Subprocess	✓	✓	✗	Separate process, but parented to the MCP server (under the LLM host session); child reaped at session cleanup → continuity loss.
Session Management Process Termination	✓	✗	✓	By forcibly terminating the host-spawned session-management helper processes, continuity was maintained; however, other requests became unavailable, degrading the non-blocking property.

"In conclusion, because MCP tools are tied to the host session and are terminated when the session times out, task continuity is lost. Hence, long-running operations need a way to detach from the host and re-parent themselves to an independent process."

Proposed Method: OS-Level Scheduling for Parallel Tool Calls



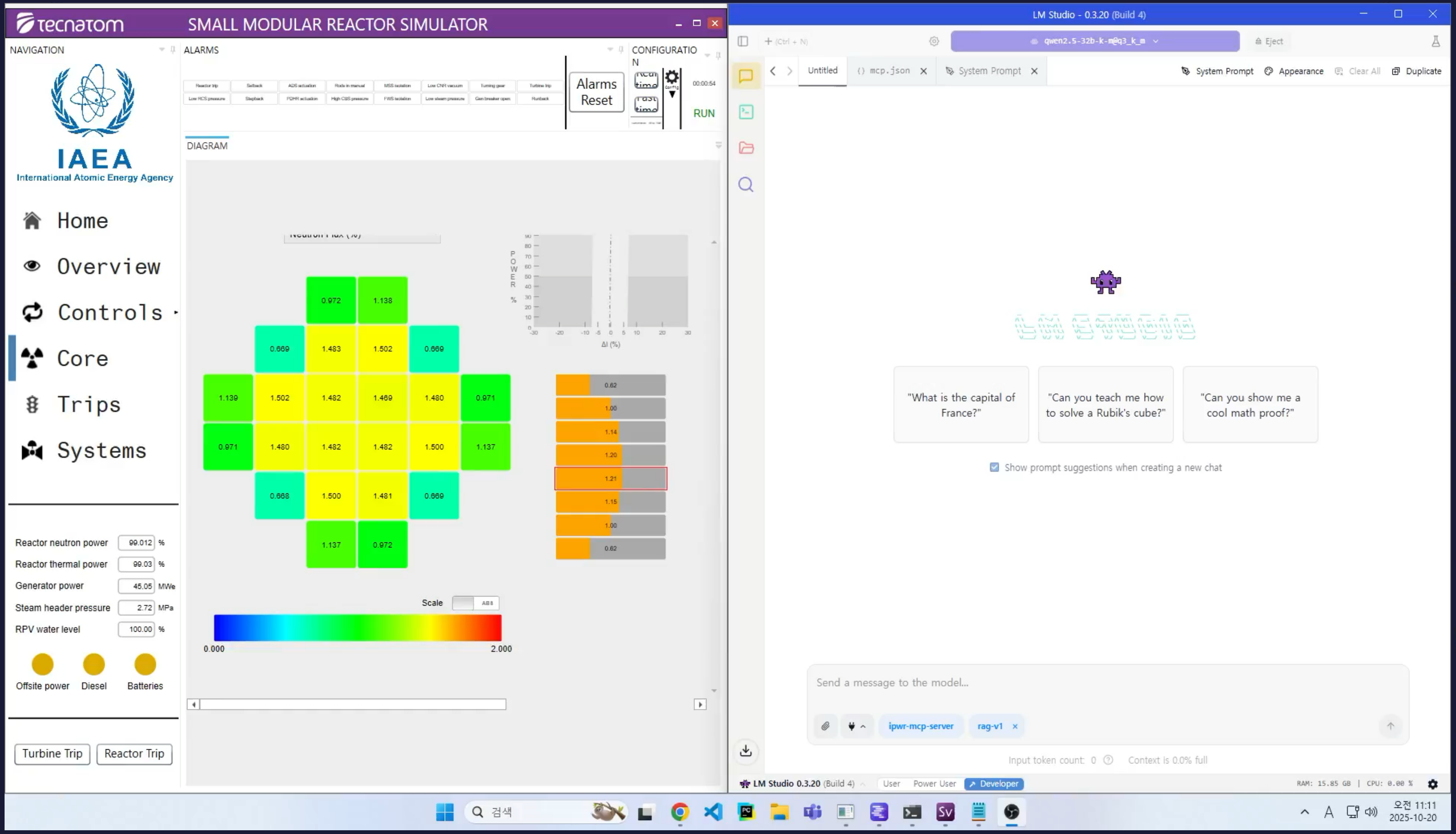
Python Code Tool 1

```
subprocess.run(
    [
        ‘schtasks’, ‘/Create’, ‘/TN’, task_name
        ‘/TR’, run_command, ‘/SC’, ‘ONCE’,
        ‘/ST’, run_time, ‘/F’
    ], check=TRUE, capture_output=True,
)

subprocess.run(
    [
        ‘schtasks’, ‘/Run’, ‘/TN’, task_name,
    ], check=TRUE, capture_output=True,
)

subprocess.run(
    [
        ‘schtasks’, ‘/Delete’, ‘/TN’,
         task_name, ‘/F’,
    ], check=TRUE, capture_output=True,
)
```

iPWR Application & Validation



Experimental environment (LOCAL SETTING): LLM Host(LM Studio), LLM(Qwen2.5-32b-Q3), MCP Server(FastMCP)

Evaluation Metrics

Responsiveness

The MCP Tool submits the task to the OS scheduler and immediately returns an acceptance response (ACK).

→ Responsiveness satisfied: Interaction with the user is not interrupted.

Non-blocking

Users can send different requests in succession while long-running operations continue in the background.

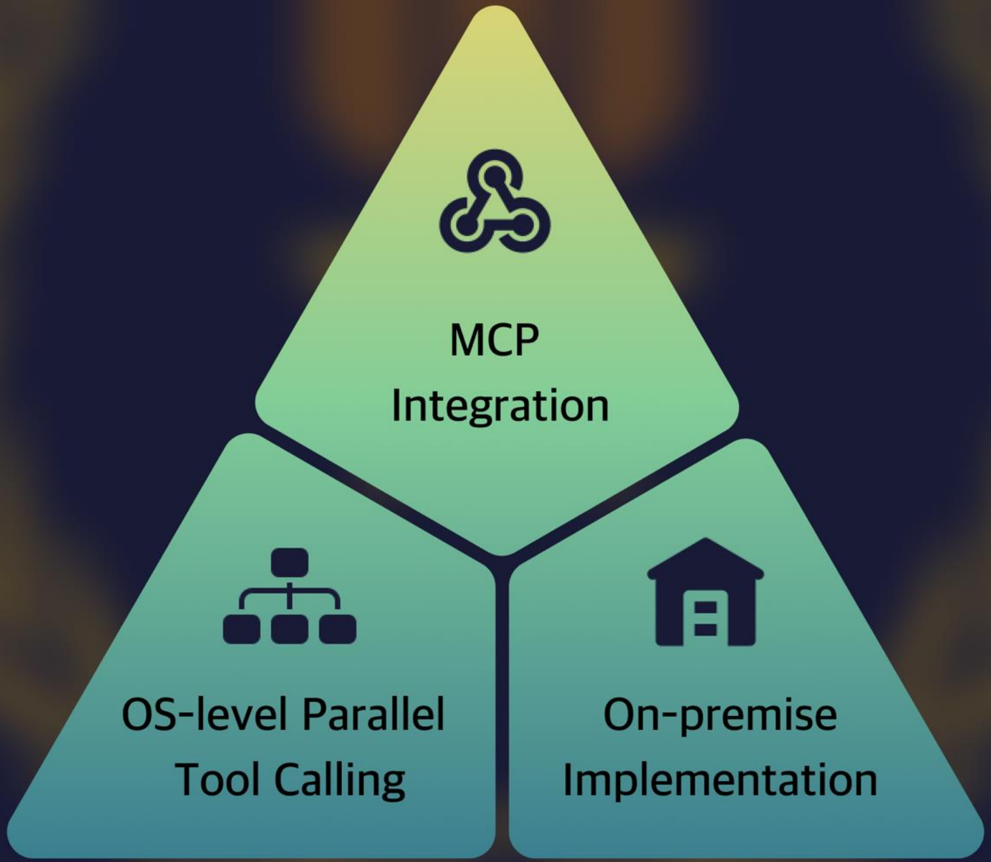
→ Non-blocking satisfied: task is handled in parallel.

Continuity

Continuously backgrounded. When you send a stop request, it detects the stop.txt and closes with a normal shutdown.

→ Continuity satisfied: independent lifetime and safe termination, independent of the host session.

Contributions



"This study established a robust Agent AI framework for the iPWR simulator by integrating the MCP protocol, enabling OS-level parallel tool calling, and deploying the system in an on-premise environment for secure operation."

Conclusions & Future Work

Conclusions

AI Assistance
AI Transformation (AX)

OS-Level Scheduling
Parallel Tool Calling

Enhanced portability
Platform independence

Achieving

Future Work

iPWR Training & Education Kit

Safe practice for parallel orchestration and recovery procedures.

Agentic AI for Nuclear Operations

Human-in-the-loop decision support using domain-adapted LLMs.

ACKNOWLEDGEMENTS

This work was supported in part by Korea Atomic Energy Research Institute R&D Program under Grant KAERI-524540-25.

Thank you for your attention

"Our goal is not full automation, but safer and more reliable human-in-the-loop operation"

REFERENCES

[1] Y. P. Lee and J. Cha, Large Language Model Agent for Nuclear Reactor Operation Assistance, Nuclear Engineering and Technology, Vol. 57, Article 103842, 2025. doi:10.1016/j.net.2025.103842
[2] International Atomic Energy Agency (IAEA), Integral Pressurized Water Reactor Simulator Manual, Training Course Series No. 65, Vienna: IAEA, 2017.
[3] International Atomic Energy Agency (IAEA), Integral Pressurized Water Reactor Simulator Manual: Exercise Handbook, Training Course Series No. 65, Vienna: IAEA, 2017.
[4] FastMCP, Welcome to FastMCP 2.0!, Getting Started, 2025. (accessed Jul. 25, 2025).
[5] Microsoft, Task Scheduler Reference, Microsoft Learn, 2019. (accessed Aug. 7, 2025).