A Traceable AI-Agent Operation Framework Using Graph-Based Procedures in Nuclear Abnormal Scenario Simulation

Seongeun Park[a], Christopher McComb[a], Jinghua Xiao[b], Joonsun Hwang[a], and Pingbo Tang[a*]

[a] Carnegie Mellon University, Pittsburgh, United States, seongeup@andrew.cmu.edu; ccm@andrew.cmu.edu; joonsunh@andrew.cmu.edu; ptang@andrew.cmu.edu

[b] Circular Water Solution, Fort Washington, United States, jxiao@circularwatersolution.com

[*]Corresponding author

Abstract: In anomaly handling of nuclear operations, understanding safety-related outcomes requires more than observing whether a scenario is ultimately resolved. It also requires tracing how plant states are interpreted, how procedure logic is followed, which actions are selected, and how the system responds after each action. As AI-based operator support and autonomous decision-making technologies are increasingly explored in nuclear operation contexts, evaluating an AI agent only through final outcomes or isolated recommendations may obscure the procedural context and execution path behind its actions. To address this limitation, a graph-based procedure representation is used to encode procedure steps, branching logic, and execution context so that AI-selected actions can be traced within the procedural structure. This paper proposes a traceable AI-agent operation framework using graph-based procedures in a nuclear abnormal scenario simulation. The framework enables an AI agent to select candidate actions based on the current simulator state and procedure context, pass those actions through a constrained action space and validation layer, and record the resulting execution process as an action-level decision path. A prototype implementation demonstrates the feasibility of separating actions generated by an LLM-based decision module into accepted action sequences and rejected action attempts, with each action recorded together with its procedure step, validation result, rejection reason when applicable, and pre-action simulator state, as well as post-action simulator state when the action is accepted. The major outcome of the prototype is an action-level decision-path log that distinguishes accepted actions from rejected attempts and links each AI-selected action to procedure context, validation results, and simulator state transitions. A preliminary verification run illustrates that the proposed log can summarize accepted-action and rejected-attempt patterns while preserving the procedural context needed for future human-in-the-loop comparisons of human operator and AI-agent decision paths.

1. INTRODUCTION

Normal, abnormal, and emergency operations in nuclear power plants rely heavily on operating procedures. In abnormal operating conditions, operators must connect changing plant states with procedural conditions, select appropriate actions, and verify system responses after each action. Therefore, safety cannot be fully explained only by whether a scenario is ultimately completed successfully. To understand safety-related outcomes in abnormal situations, it is necessary to analyze how state interpretation, procedure step selection, control action execution, and post-action state changes are connected in sequence and context.

Conventional paper-based procedures leave much of this connection process to operator interpretation and manual tracking. Operators must read static procedural documents while continuously comparing dynamically changing plant states with procedural conditions and manually managing conditional branches and execution history. This can impose additional cognitive burden during the parallel management of multiple procedures, conditional branch decisions, and place-keeping [1]. Accordingly, research on computer-based procedures has progressed beyond simply displaying procedures on a

screen to linking procedure steps with relevant plant states, tracking procedure execution status, and, in some cases, incorporating embedded indications, soft controls, and automation support [2,3].

Although this body of CBP research has made important progress in reducing the limitations of paper-based procedures, its focus has largely remained on procedure execution support, state display, place-keeping, condition evaluation, and control-function integration [2–4]. In parallel, research on AI-based operator support and nearly autonomous management in nuclear power plants has shown that plant knowledge, state information, and predictive models can support diagnosis, consequence assessment, and control recommendation [5–7]. Building on these developments, this study implements conditions under which an AI agent selects or executes the next action based on the current simulator state, procedural context, and constrained action space. In this setting, a graph-based procedure is needed not only to represent procedure content, but also to preserve the execution context of AI-selected actions. By explicitly representing state, condition, action, and branch relationships, the graph-based procedure links each AI-selected action to its procedure step, branch context, validation result, and associated simulator state transition.

This study proposes an AI-agent execution framework and decision-path logging schema using graph-based procedures in a nuclear abnormal operation simulator. The AI agent selects the next candidate action within a constrained action space based on the current simulator state and procedure context, and the validation layer examines whether the action conforms to the allowed action schema, basic simulator-interface constraints, and available procedure context. Only accepted actions are applied to the simulator, and each action is recorded together with its procedure context, validation result, rejection reason when applicable, and pre-action simulator state, as well as post-action simulator state when the action is accepted. Through this structure, AI-agent execution can be reconstructed not as a simple output, but as an action-level decision path.

The objective of this study is not to evaluate the operational performance of the AI agent, but to structure the AI execution process as a traceable and analyzable decision path. Rather than treating the AI output as a final success/failure result or an isolated recommendation, the proposed framework records each AI-selected action together with the corresponding simulator state, procedure step, branch context, validation result, and state transition. Therefore, the research question of this paper is as follows: How can the action selection, execution, and state transition of an AI agent in nuclear abnormal operation simulation be constrained and recorded so that they can be reconstructed as an analyzable decision path? To answer this question, this paper presents an AI-agent execution framework and decision-path logging schema using graph-based procedures.

2. RELATED WORK

2.1. Computer-Based Procedures in Nuclear Operations

Computer-based procedures (CBPs) in nuclear operations have been developed to address limitations of paper-based procedures. Paper-based procedures require operators to read procedural instructions, connect the current plant state with procedural conditions, and manually track completed and remaining steps. In contrast, CBPs can support procedure step display, procedure navigation, place-keeping, presentation of relevant plant state information, and, in some cases, integration with control functions. Human factors guidance for computerized operating procedure systems in nuclear power plants emphasizes not only procedure readability, but also how operators interact with procedure steps, plant information, and control interfaces [2]. A qualitative study of field operators also identified dynamic plant-condition integration, management of multiple procedures, and manual place-keeping as major burdens in paper-based procedure use, and suggested that CBPs should provide logical sequence guidance, automatic place-keeping, and dynamic context-sensitive information [1].

Recent work has also explored the integration of procedure systems with simulators for experimental analysis. Simulator-integrated CBP platforms such as Rancor and RIPS show that procedures can be

treated not merely as displayed instructions, but as part of an integrated environment in which procedure execution status, operator interaction, and simulator state changes can be recorded and analyzed together [4,8]. This line of work suggests that procedure use should be understood not as a static document-reading process, but as an observable performance process that unfolds over time. Such platforms provide a basis for experimentally comparing how different forms and levels of procedure support affect operator performance and cognitive burden [4].

Recent studies have further extended procedure research toward connections between procedures and digital human-system interfaces (HSIs), or between procedures and structured knowledge representations. AutoGraph models interface elements as a knowledge graph and maps procedural sentences to interface-level operation paths in digital control rooms to support multi-action step identification and automated execution [9]. The authors' prior work, BuildLink, transformed emergency procedures into procedural knowledge graphs with explicit state-condition-action relationships and linked them with an LER-based incident knowledge graph for evidence retrieval in a GraphRAG-based advisor [10]. However, these studies mainly focus on translating procedural instructions into interface-level actions or providing procedural and incident evidence for operator support. In contrast, the present study focuses on recording the process by which an AI agent refers to graph-based procedures to select and execute actions as an action-level decision path that connects simulator state, procedure context, validation result, and state transition.

2.2. AI-Based Operator Support

Research on AI-based operator support in the nuclear domain has generally developed toward bounded support, such as diagnosis, recommendation, action validation, and decision support, rather than toward fully replacing human operators with autonomous control. Computerized Operator Support Systems (COSSs) have been proposed as advisory systems that support plant fault detection, diagnosis, mitigation, monitoring, and recovery while leaving final judgment and control authority to the operator [11]. Operation validation systems, such as the concealed intelligent assistant (CIA), have also sought to reduce human error by evaluating whether an operator's intended action is consistent with procedural requirements and safety impacts and intervening when inappropriate actions are anticipated [12]. These studies show that AI support in nuclear operations should be constrained by procedure logic and safety conditions and should remain reviewable.

In parallel, research has explored approaches that combine autonomous operation management with decision support. Nearly Autonomous Management and Control (NAMAC) integrates digital twins, diagnosis, prognosis, strategy assessment, and discrepancy checking to provide control recommendations to operators in advanced reactor scenarios [6,7]. This work demonstrates that AI can support complex operational judgment using plant models and risk information. However, the primary focus of such studies is on recommendation quality, control strategy selection, and system-level operation management. The logging schema for recording the procedural context and state transition associated with each individual AI-selected action has received less attention as a design object in itself. This study therefore focuses not on the performance of an AI control policy, but on the recordability of the AI-agent execution process. Specifically, it treats AI actions not as isolated outputs, but as interpretable analytical units connected with a constrained action space, procedure context, and pre- and post-action state transitions.

2.3. Structured Logs and Traceability

The view that procedure-based performance should be analyzed as a connected sequence of events, rather than only as a final outcome, has long been addressed in human reliability analysis (HRA). In traditional HRA, THERP structured human error probability through task decomposition and event trees, while ATHEANA analyzed how unsafe actions can occur within specific operating contexts by focusing on unsafe actions and error-forcing contexts [13,14].

Dynamic HRA research has since extended this perspective by modeling operator behavior as cognitive and physical action sequences that unfold over time. Tools such as HUNTER show that procedure paths, task durations, and error opportunities can be structured and used in simulator-based HRA analysis [15]. Studies using simplified simulators and student operators for HRA data collection have also demonstrated the potential of Rancor Microworld experiments to collect human performance data and error opportunities, thereby supplementing HRA quantification and full-scope data collection [16]. These studies indicate the need to record and analyze not only final outcomes, but also action sequences and related contexts during operational performance.

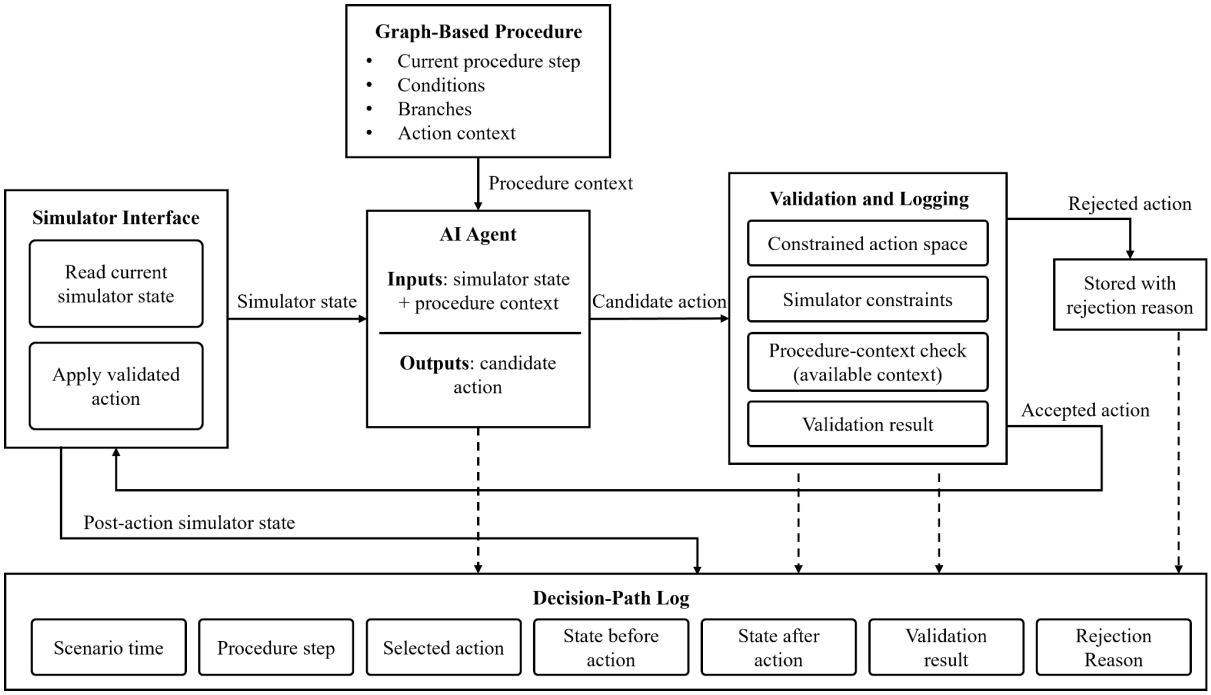
Structured execution records also play an important role in simulator-integrated CBP research. RIPS integrates operator procedure-use logs with Rancor simulator logs, providing a basis for collecting procedure execution processes and simulator state changes together [4]. This approach enables procedure use to be treated not as static document reference, but as an observable performance process in which simulator state, procedure steps, and operator interactions are temporally connected. Existing work on structured logs and traceability has primarily focused on human operator procedure use, simulator-based HRA data collection, or performance records for post hoc analysis. In contrast, less attention has been given to how run-time traces should be recorded when an AI agent selects and executes actions within a simulator. For AI-agent operation, this suggests a need to treat each AI-selected action as an analytical unit: a proposed task action that occurs under a specific simulator state and procedure context, is either accepted or rejected by validation constraints, and produces or fails to produce a state transition. The next section addresses this gap by presenting a framework that constrains AI-agent actions within a graph-based procedure context and records each action in a decision-path log.

3. Framework and Prototype Implementation

3.1. Overall Framework

This study proposes a framework for recording the execution process of an AI agent using graph-based procedures as an action-level decision path. The framework focuses not on the final operational performance of the AI agent, but on tracing the simulator state and procedure context in which each action is selected, how the action is validated and executed, and how the system state changes after execution. Figure 1 shows the overall structure of the proposed framework. The framework consists of a simulator interface, a graph-based procedure, an AI agent, and a validation and logging layer. The simulator interface reads the current simulator state and applies validated actions to the simulator. The graph-based procedure provides the current procedure step, relevant conditions, branches, and action context. The AI agent selects a candidate action based on the simulator state and procedure context. The validation and logging layer verifies whether the candidate action conforms to the constrained action space and basic simulator execution constraints, while also recording the available procedure context. Accepted actions are applied to the simulator, while rejected actions are stored with rejection reasons. Each action record includes the scenario time, procedure step, selected action, validation result, state before action, and, for accepted actions, state after action.

Figure 1: Conceptual structure of the proposed AI-agent execution and decision-path logging framework



3.2. Graph-Based Procedure and Constrained Action Space

In the proposed framework, the graph-based procedure constrains the AI agent's action selection within the procedural context. This study builds on the procedure graph structure used in the prior BuildLink implementation [10] and uses it as a procedural constraint for AI-agent execution. The procedure graph consists of procedure step nodes, procedure logic/feature nodes, and facility entity nodes, representing the sequential flow, conditional checks, and manipulation targets of procedures as explicit node-edge relationships. Procedure step nodes represent the basic units of procedure execution, procedure logic/feature nodes represent conditions, threshold evaluations, branch logic, or step-completion criteria, and facility entity nodes represent simulator elements such as components, controls, indicators, and alarms.

The AI agent refers to both the current simulator state and the procedure graph context when selecting a candidate action. For example, if the current step requires checking a feedwater flow condition, the agent may read the relevant indicator or select a recovery action allowed by the corresponding branch when the condition is not satisfied. Conversely, actions that are not clearly connected to the current procedure context, or control changes that violate value constraints, may be treated as rejected actions during validation.

To support this process, this study defines a constrained action space based on the procedure graph and simulator interface. The constrained action space is a predefined list of action types, targets, and value ranges that the AI agent can select from. Examples in the prototype include starting or stopping components, adjusting control valves, selecting procedure branches, advancing procedure steps, and completing the scenario. These actions are defined at a higher level than raw mouse clicks or simulator variable writes, but at a smaller semantic unit than an entire procedure step. As a result, AI-selected actions can be recorded as human-interpretable manipulation units while also being converted into executable units through the simulator interface.

3.3. Execution Loop and Validation

AI-agent execution follows an iterative sense, select, validate, act, and log structure. First, the framework reads the current simulator state through the simulator interface and checks the current step, relevant conditions, possible branches, and available procedure context from the procedure graph. In

the prototype implementation, the AI agent is implemented as an LLM-based decision module that receives the current simulator state, procedure graph context, and constrained action candidates as input and generates a structured candidate action. The prototype uses a Gemini 2.5 Flash backend endpoint as the LLM-based decision module. The LLM module receives a structured prompt containing the current simulator state, compact procedure graph context, and constrained action candidates, and returns a JSON object specifying the action type, target, optional value, and rationale. The purpose of the LLM module in this prototype is not to optimize or benchmark model performance, but to generate structured candidate actions that can be passed through the proposed validation and logging pipeline. The candidate action includes an action type, target, value, and rationale, and is not directly applied to the simulator before being reviewed by the validation and logging layer.

The validation layer in the current prototype focuses on two functions. First, it checks whether the proposed action conforms to the allowed action schema. This includes checking action type and value constraints, while the action target is normalized based on predefined action definitions and stored in the log record. Second, it checks whether the proposed action can be executed through the simulator interface under the current simulator state. In this prototype, the procedure graph constrains action generation by defining the available action candidates associated with the current procedure context, and the same context is stored in the log record for post hoc interpretation. However, full semantic procedure-context validation, in which each generated action is explicitly verified against the current procedure step, branch condition, and procedural intent, remains a future extension.

Only accepted actions are applied through the simulator interface, and the framework then reads the post-action simulator state and stores the state transition before and after the action in the log record. Rejected actions are not applied to the simulator, but are preserved in the action-level trace together with rejection reasons. This makes it possible to distinguish accepted action sequences from rejected attempts and to retrospectively examine nonexistent target manipulation, schema violations, simulator constraint violations, or potential mismatches with the procedural context.

3.4. Decision-Path Logging Schema and Prototype Output

The main output of this study is an action-level decision-path log. Each log record corresponds to a proposed action and includes the scenario time, current procedure step, selected action, action target, action value, reasoning, state before action, state after action, accepted status, and rejection reason. The reasoning field stores the AI agent's explanation for selecting an action, but this study does not treat it as a truthful representation of the model's internal reasoning. The primary basis for post hoc analysis is not the reasoning text itself, but the evidence chain connecting procedure context, validation result, state before action, and state after action. Therefore, the decision-path log is interpreted primarily through action-state transitions and procedure context, rather than through the AI-generated explanation alone. Table 1 summarizes the main fields of the decision-path log generated by the prototype implementation. The schema is designed to record not only the action selected by the AI agent, but also the procedure context in which the action occurs, the validation result, and the pre- and post-action simulator states. Each log record therefore allows the agent output to be interpreted not as a simple recommendation, but as an action-level record that connects simulator state with procedural context.

Table 1: Decision-path logging schema

Category	Logged fields	Purpose
Scenario context	scenario_time, procedure_step	Locates the action in time and procedure context
Selected action	selected_action, action_target, action_value	Records the agent's proposed action
Procedure context	current_step, graph_allowed_actions	Links the action to the procedure graph context
Validation result	accepted, validation_result, rejection_reason	Separates accepted actions from rejected attempts

State evidence	state_before_action, state_after_action	Captures the pre-action state and the post-action state when the action is accepted
Agent explanation	reasoning	Stores the agent-provided rationale as auxiliary information

The prototype implementation is presented as a preliminary demonstration showing that the proposed framework can operate end-to-end. In the prototype, the simulator state is read during an abnormal operation scenario, the graph-based procedure is referenced, the AI agent selects a candidate action, and only accepted actions are applied to the simulator through the validation layer. Each run starts from an initial state and generates both an action-level log and a run-level summary. In the verification run, the prototype loaded a procedure graph with 99 nodes and 85 edges and exposed a constrained action space containing 25 action definitions. The verification script produced two accepted actions, four intentionally rejected action attempts, and a full action trace containing all records. This verification is intended to demonstrate the operation of the execution and logging pipeline, rather than to evaluate AI-agent operation performance. Table 2 shows representative action-level log records generated during an illustrative prototype verification run. These records are not intended as evidence of AI-agent operational performance or procedure-following capability. Rather, they are used to verify that the proposed logging schema can capture accepted actions, rejected attempts, rejection reasons, and post-action state transitions within a single execution trace. The cases include accepted procedure advancement, graph-context rejection, accepted control adjustment, value-constraint rejection, unavailable branch-transition rejection, and invalid branch-value rejection. Accepted actions are applied to the simulator and include post-action state transitions, such as a procedure-step change or a feedwater control valve change. Rejected actions are not applied to the simulator, but are preserved in the trace with rejection reasons.

Table 2. Representative action-level log records from prototype verification

Procedure	Proposed action	Validation result	Rejection reason	State transition
Step 1.1	advance_procedure	Accepted	-	current_step changed to Step 1.2
Step 1.2	trip_reactor	Rejected	Not linked to current procedure context	Not applied
Step 2.2	set_feedwater_control_valve(value=1)	Accepted	-	feedwater_control_valve changed to 1
Step 2.2	set_feedwater_control_valve(value=1.5)	Rejected	Value must be between 0 and 1	Not applied
Step 2.3	select_procedure_branch(value=FALSE)	Rejected	No valid graph transition found	Not applied
Step 2.3	select_procedure_branch(value=MAYBE)	Rejected	Branch value must be TRUE or FALSE	Not applied

Table 3 summarizes the main process patterns observed from the action-level records in Table 2. These patterns show that the proposed log can distinguish executed actions from rejected attempts and preserve the information needed for later interpretation.

Table 3: Summary of process patterns captured in the prototype verification run

Pattern type	Observed pattern	Log-based evidence

Accepted action sequence	Valid procedure advancement and feedwater control adjustment were applied to the simulator.	Accepted actions include post-action state transitions.
Procedure-context rejection	An action outside the current procedure context was blocked.	The reactor trip attempt was rejected because it was not linked to the current procedure context.
Validation-based rejection	Control adjustment outside the allowed value range and invalid or unavailable branch selections were blocked.	Rejected attempts include value-range violation, unavailable graph transition, and invalid branch value.

These records show how an AI run can be interpreted as an action-level trace rather than only as a final outcome. The accepted action sequence shows the order of actions actually applied to the simulator, while rejected action attempts show actions that were attempted by the agent but blocked by the validation layer. Because each record includes both the procedure context and simulator state, it is possible to retrospectively examine the state and procedural context in which each selected action occurred.

4. DISCUSSION

The most direct implication of the proposed framework is that AI-agent behavior can be recorded and interpreted not only as a final scenario outcome, but as a traceable sequence of constrained task actions connected to procedure context and simulator state. The proposed decision-path log makes it possible to reconstruct, at the action level, which procedure step the AI agent was in, which action it selected, whether the action was accepted or rejected, and how the simulator state changed after execution. The graph-based procedure serves as a reference for both action selection and log interpretation, and each action record includes both procedure context and state transition information. As a result, AI output can be interpreted not as an isolated recommendation, but as an execution record that connects procedural context with system response. This structure supports basic analyses such as step alignment, action frequency, rejected-attempt patterns, and state-transition patterns. This allows future comparisons to examine not only whether human operators and AI agents reach the same final scenario outcome, but also where their procedure paths diverge, which actions are repeatedly rejected, and how state transitions differ after similar procedural contexts.

This study has several limitations. First, the paper presents a framework and prototype implementation and does not quantitatively demonstrate the quality of AI-agent actions, procedure compliance rate, or improvement in safety margins. Second, the quality and completeness of the procedure graph determine the upper bound of action constraints and log interpretation. Third, simulator-specific latency, synchronization, and asynchronous event handling require further validation. Fourth, the reasoning field is only auxiliary explanatory information and does not guarantee the truthfulness of the model's internal reasoning. Therefore, future explainability evaluation should focus not on the reasoning text itself, but on the evidence chain connecting procedure context, validation result, and action-state transition. Future work can extend the framework to human-in-the-loop experiments comparing the decision paths of human operators, AI agents, and AI-advisory conditions. In addition, multiple AI policies or model variants can be executed under the same procedure graph and simulator scenario to compare decision-path consistency and unsafe-attempt patterns.

5. CONCLUSION

This study presented a preliminary framework for recording the execution process of an AI agent using graph-based procedures in nuclear abnormal operation simulation as a traceable action-level decision path. The proposed framework is designed so that the AI agent refers to the current simulator state and procedure graph, selects a candidate action within a constrained action space, and applies the action to the simulator only after it is validated by the validation layer. This study also presented a decision-path logging schema that records the procedure context, selected action, validation result, reasoning, and pre- and post-action simulator states for each action. Through this structure, AI-agent execution can be

reconstructed not as a simple final success or failure outcome, but as an execution record that combines procedural context with changes in system state. This study does not aim to quantitatively evaluate AI-agent operation performance or directly compare it with human operators. Instead, it focuses on establishing an execution and logging structure that connects simulator state, procedure context, selected action, validation result, and state transition, thereby supporting future human-in-the-loop comparisons of human-operator and AI-agent decision paths.

Acknowledgements

This material is based upon work supported by the Pennsylvania Infrastructure Technology Alliance (PITA), a program of the Pennsylvania Department of Community and Economic Development (DCED), in collaboration with Carnegie Mellon University and Westinghouse; the Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) through the Hydroinformatics Innovation Fellowship under NSF Award No. EAR-1849458; and the Chishiki-AI SCIPE Graduate Fellowship funded by NSF under Award No. 2321040.

References

[1] K. Le Blanc, J. H. Oxstrand, and T. Waicosky. *"Requirements for Computer-Based Procedures for Nuclear Power Plant Field Operators: Results from a Qualitative Study"*, Idaho National Laboratory, (2012).
[2] IEEE. *"IEEE Guide for Human Factors Applications of Computerized Operating Procedure Systems (COPS) at Nuclear Power Generating Stations and Other Nuclear Facilities"*, IEEE, (2022).
[3] J. M. O'Hara, J. C. Higgins, W. F. Stibler, and J. Kramer. *"Computer-Based Procedure Systems: Technical Basis and Human Factors Review Guidance"*, U.S. Nuclear Regulatory Commission, (2000).
[4] R. L. Boring, R. Lew, and T. A. Ulrich. *"Rancor Integrated Procedure System (RIPS): A Computer-Based Procedure Platform for Advanced Reactor Research"*, Idaho National Laboratory, (2025).
[5] B. Hanna, T. C. Son, and N. Dinh. *"AI-Guided Reasoning-Based Operator Support System for the Nuclear Power Plant Management"*, Annals of Nuclear Energy, 154, 108079, (2021).
[6] L. Lin, P. Athe, P. Rouxelin, M. Avramova, A. Gupta, R. Youngblood, J. Lane, and N. Dinh. *"Development and Assessment of a Nearly Autonomous Management and Control System for Advanced Reactors"*, Annals of Nuclear Energy, 150, 107861, (2021).
[7] L. Lin, P. Athe, P. Rouxelin, M. Avramova, A. Gupta, R. Youngblood, J. Lane, and N. Dinh. *"Digital-Twin-Based Improvements to Diagnosis, Prognosis, Strategy Assessment, and Discrepancy Checking in a Nearly Autonomous Management and Control System"*, Annals of Nuclear Energy, 166, 108715, (2022).
[8] T. A. Ulrich, R. Lew, S. Werner, and R. L. Boring. *"Rancor: A Gamified Microworld Nuclear Power Plant Simulation for Engineering Psychology Research and Process Control Applications"*, Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 61, pp. 398-402, (2017).
[9] X. Xiao, J. Tong, J. Sun, Z. Sui, J. Liang, H. Zhao, J. Zhao, and H. Wang. *"AutoGraph: A Knowledge-Graph Framework for Modeling Interface Interaction and Automating Procedure Execution in Digital Nuclear Control Rooms",* arXiv, (2025).
[10] S. Park, J. Hwang, K. Jeon, R. L. Boring, and P. Tang. *"BuildLink: Graph-Structured Procedural Representation with Incident Integration"*, Proceedings of the International Conference on Computing in Civil Engineering 2026, ASCE, (2026).
[11] R. L. Boring, K. D. Thomas, T. A. Ulrich, and R. T. Lew. *"Computerized Operator Support Systems to Aid Decision Making in Nuclear Power Plants"*, Procedia Manufacturing, 3, pp. 5261-5268, (2015).
[12] J. Ahn, J. Bae, B. J. Min, and S. J. Lee. *"Operation Validation System to Prevent Human Errors in Nuclear Power Plants",* Nuclear Engineering and Design, 397, 111949, (2022).
[13] J. Forester, A. Kolaczkowski, S. Cooper, D. Bley, and E. Lois. *"ATHEANA User's Guide"*, U.S. Nuclear Regulatory Commission, Washington, DC, (2007).
[14] A. D. Swain and H. E. Guttmann. *"Handbook of Human-Reliability Analysis with Emphasis on Nuclear Power Plant Applications"*, Sandia National Laboratories, (1983).

[15] R. Boring, T. Ulrich, J. Park, Y. Heo, and J. Ahn. *"The HUNTER Dynamic Human Reliability Analysis Tool: Overview of the Enhanced Framework for Modeling Human Digital Twins"*, Probabilistic Safety Assessment and Management PSAM 16, (2022).
[16] J. Park, R. L. Boring, T. A. Ulrich, R. Lew, S. Lee, B. Park, and J. Kim. *"A Framework to Collect Human Reliability Analysis Data for Nuclear Power Plants Using a Simplified Simulator and Student Operators"*, Reliability Engineering & System Safety, 221, 108326, (2022).