INL/RPT-22-70076

Light Water Reactor Sustainability Program

Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER) Demonstration: Part 2, Model Runs of Operational Scenarios

November 2022

U.S. Department of Energy

Office of Nuclear Energy

INL/RPT-22-70076

Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER) Demonstration: Part 2, Model Runs of Operational Scenarios

Roger Lew
University of Idaho

Thomas A. Ulrich and Ronald L. Boring
Idaho National Laboratory

November 2022

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy

This page intentionally left blank.

ABSTRACT

In this report, we provide enhancements to the Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER) software for dynamic human reliability analysis. HUNTER serves to create a virtual operator that can be used in Monte Carlo modeling to determine human error probabilities and other measures of human performance. We document the coupling of HUNTER to an external plant model. The Rancor Microworld Simulator is a simplified model of a pressurized water reactor. The simplified model allows easy data collection through human-in-the-loop studies and easier customization of features to reflect advances in plant technologies. The coupling of HUNTER with Rancor allows faster model development and execution that was previously possible. It also aligns HUNTER to ongoing empirical data collection efforts on operator performance, affording insights into actual human performance during modeling and validation of HUNTER model runs. In addition to documenting the coupling of HUNTER and Rancor, this report introduces model runs for two new industry-derived scenarios: loss of feedwater and startup. The results of HUNTER model runs for these scenarios demonstrate the capabilities of the coupling. HUNTER-Rancor accurately models operator performance. Future work will aim at further calibrating HUNTER-Rancor with more nuanced treatment of performance shaping factors and with specific industry-oriented applications like the application of aiding procedure developers in anticipating error traps for novel operating procedures.

ACKNOWLEDGMENTS

The authors wish to thank the Risk-Informed System Analysis (RISA) Pathway of the U.S. Department of Energy's Light Water Reactor Sustainability (LWRS) Program for its support of the research activities presented in this report. In particular, we thank Svetlana Lawrence, pathway lead for RISA, for championing the demonstrations found in this report. We also acknowledge helpful discussions with Jooyoung Park of Idaho National Laboratory and Olugbenga Gideon of University of Idaho, which helped shape the direction of the research presented here.

CONTENTS

FIGURES

TABLES

ACRONYMS

AOP	abnormal operating procedure
API	application programming interface
CDF	cumulative distribution function
COSS	Computerized Operator Support System
COTS	commercial off the shelf
CSV	comma-separated value
DOE	Department of Energy
EOP	emergency operating procedure
GOMS	Goals, Operators, Methods, and Selection rules
GPWR	Generic Pressurized Water Reactor
GUI	graphical user interface
h	human action
HEP	human error probability
HRA	human reliability analysis
HSSL	Human System Simulation Laboratory
HUNTER	Human Unimodel for Nuclear Technology to Enhance Reliability
ID	identifier
INL	Idaho National Laboratory
JSON	JavaScript Object Notation
LWRS	Light Water Reactor Sustainability
PRA	probabilistic risk assessment
PSF	performance shaping factor
PWR	pressurized water reactor
RAVEN	Risk Analysis Virtual Code Environment
RISA	Risk-Informed Systems Analysis
RISMC	Risk-Informed Safety Margin Characterization
RNO	response not obtained
s	system state
SGTR	steam generator tube rupture
SME	subject matter expert
SPAR-H	Standard Plant Analysis Risk-Human
SRO	senior reactor operator
t	current time
t + 1	future time
U.S.	United States
UUID	universally unique identifier
WPF	Windows Presentation Foundation

This page intentionally left blank.

HUMAN UNIMODEL FOR NUCLEAR TECHNOLOGY TO ENHANCE RELIABILITY (HUNTER) DEMONSTRATION: PART 2, MODEL RUNS OF OPERATIONAL SCENARIOS

Figure 1. The HUNTER project logo.

1. INTRODUCTION

1.1 HUNTER Background

Dynamic human reliability analysis (HRA) provides a simulation of human performance that can be used to inform quantitative risk assessment. One dynamic HRA approach, the Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER; see Figure 1) framework, was initially conceived in 2016 (Boring et al., 2016) as the offshoot of dynamic risk modeling work to support severe accident scenarios like flooding. There were two driving factors behind the decision to bring together different risk modeling ideas into a single framework:

1. The need to integrate human elements of risk with other simulation-based tools as part of a suite of probabilistic risk assessment (PRA) tools being developed under the Risk-Informed Safety Margin Characterization (RISMC; now Risk-Informed Systems Analysis or RISA) Pathway under the U.S. Department of Energy's (DOE's) Light Water Reactor Sustainability (LWRS) Program.
2. The desire to create a simplified and dynamic version of the Standard Plant Analysis Risk-Human Reliability Analysis (SPAR-H; Gertman et al., 2005) method as a proof of concept for simplified dynamic HRA.

HUNTER emerged as a loose framework that included separate software codes and solutions to address these goals:

1. A dynamic version of SPAR-H that autocalculated performance shaping factors (PSFs; Boring et al., 2017a) to calculate human error probabilities (HEPs).
2. A subtask taxonomy called Goals-Operators-Methods-Selection rules (GOMS)-HRA that included elemental human activities called *task level primitives* (Boring and Rasmussen, 2016), which were linked to procedure steps as *procedure level primitives* (Boring et al., 2017). In turn, error mechanisms for each primitive were identified and classified as *task level errors* (Boring et al., 2018). A byproduct of this work was identifying not only error rates for each task level

primitive but also time distributions to allow GOMS-HRA to estimate task durations (Ulrich et al., 2017a).

3. A dynamic treatment of dependency, which considers the relationship between sequences of human errors (Boring, 2015; Park, Boring, & Kim, 2019). This research introduced concepts of PSF distributions and PSF lag and linger, meaning PSFs have delay and decay functions that must be considered as part of HRA.
4. Integration of HUNTER codes into dynamic PRA tools such as Risk Analysis Virtual Code ENvironment (RAVEN; Rabiti et al. 2017). Initial implementations of HUNTER used RAVEN to couple HUNTER to thermohydraulic simulation codes (Boring et al., 2016).

Demonstrations of HUNTER were successful and established the overall value of the approach. A RISA review of tools and frameworks (Choi, 2020) suggested that the HUNTER framework would benefit from further steps to increase its technology readiness level, including developing a single application framework for HUNTER to ensure its utility beyond a research tool. A concerted effort was undertaken to develop a version of HUNTER that could operate as a standalone software analysis tool (Boring et al., 2022). Conceptually, HUNTER is seen as a virtual operator or digital human twin that couples with a virtual nuclear power plant model (i.e., a simulator) or similar digital twin. A revised version of HUNTER (called HUNTER 2 to distinguish it from earlier efforts) was released in March 2022. It included several key features:

1. A unified software architecture consisting of three main modules as shown in Figure 2. The *individual* module consists of dynamic PSFs that influence the performance of the virtual operator (Park et al., 2022a). The *task* module consists of the tasks being performed and is captured in HUNTER as a procedure engine that walks through the steps of plant operating procedures (Ulrich et al., 2022). The *environment* module is a simulation of a plant, typically an external model that is coupled to HUNTER (Heo et al., 2022) via an application programming interface (API). The task module drives the virtual operator through a step-by-step interaction with the plant model based on the logic of each procedure step applied to the plant state, whereby the individual module moderates the performance accuracy and timing of the virtual operator at each step.

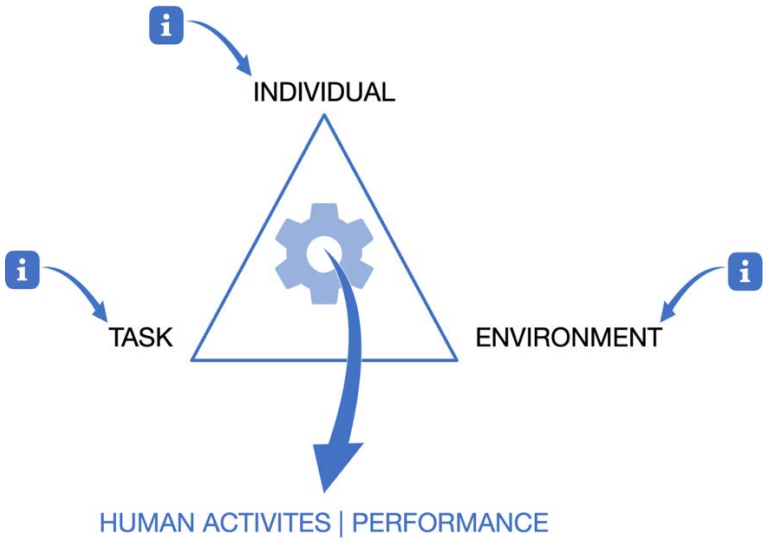


Figure 2. The conceptual model of HUNTER 2.

2. These modules operate under a common graphical user interface (GUI) with a single-click launch of the HUNTER software (Ahn et al., 2022). A sample screen from the HUNTER GUI is shown in Figure 3, featuring a procedure flow diagram for building and monitoring the task module (left) and fields for coupling HUNTER to the external environment module and for adjusting task module performance for each procedure step (right).

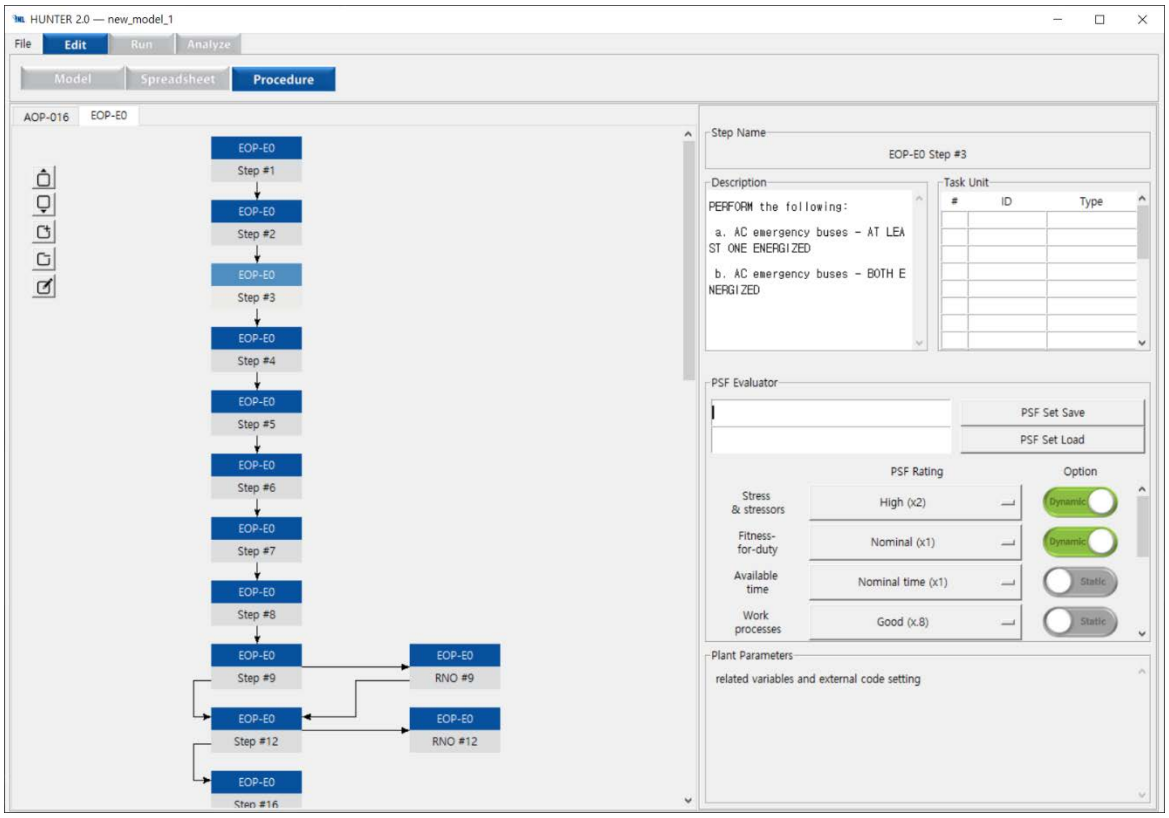


Figure 3. Example GUI for HUNTER.

3. The demonstration included a realistic version of a steam generator tube rupture (SGTR) scenario, with the HUNTER outputs compared to empirical data derived from reactor operators performing the same scenario in a qualified nuclear power plant simulator (Bye et al., 2011).

1.2 Toward HUNTER 2.1

The demonstration of HUNTER 2 represented a significant maturation of the HUNTER framework toward standalone software that could address industry needs for dynamic HRA and PRA (Lawrence et al., 2021). However, the release of HUNTER 2 cannot be considered the capstone of the research and development efforts for HUNTER. Notably, two shortcomings still need to be addressed:

1. The environment module demonstrated in HUNTER 2 features a coupling to the RELAP5-3D thermohydraulic software. There are limitations of this approach that will be discussed in Section 2 of this report. There is a desire to couple HUNTER with additional plant models.
2. The SGTR scenario is well understood and documented in the HRA community (Bye et al., 2011) but represents a limited treatment of human error. To demonstrate HUNTER's utility

for broader industry application, it is necessary to build more sample use cases to refine and prove HUNTER capabilities.

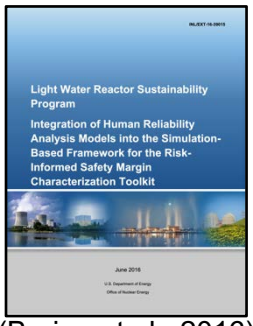
This report is Part 2 of a series of reports to address these two shortcomings. In Part 1 of the report (Park et al., 2022b), human performance findings from an operator-in-the-loop study involving ten common reactor scenarios are presented. The empirical data of actual operator performance inform the modeling of additional scenarios in HUNTER and allow benchmarking of the HUNTER generated simulation results against actual operator performance. This report, Part 2, captures the coupling of another simulation software code to HUNTER in the environment module to afford greater HRA modeling ease and flexibility.

The evolution of HUNTER can be seen in Figure 4, which shows the key milestone reports to date. This report is not meant to be read in isolation. To fully understand HUNTER, the reports should be read in sequence. Further, to understand this report, the following sections are best read in the order presented.

1. *Section 1* (this chapter) provides a succinct review of HUNTER.
2. *Section 2* discusses concepts related to coupling HUNTER to external simulations that represent nuclear power plants.
3. *Section 3* introduces the Rancor Microworld Simulator, which is the plant simulator to which HUNTER has recently been coupled. This section also reviews some of the empirical studies using human operators in the loop that help build and validate HRA models in HUNTER.
4. Section 4 discusses coupling HUNTER and the Rancor Microworld Simulator, including implementational details of necessary added features in HUNTER. Because the coupling of a new plant model involves development of new features beyond HUNTER 2 (Boring et al., 2022), the resulting software version of HUNTER presented in this report is called HUNTER 2.1.
5. Section 5 reviews the implementation and performance of two new plant scenarios in HUNTER, namely loss of feedwater and startup.
6. Section 6 concludes this report with a discussion of shortcomings and planned future development work in HUNTER.

HUNTER 1

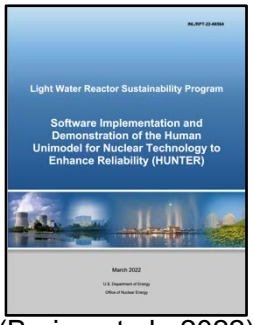
Initial framework and demonstration of HUNTER concepts



(Boring et al., 2016)

HUNTER 2

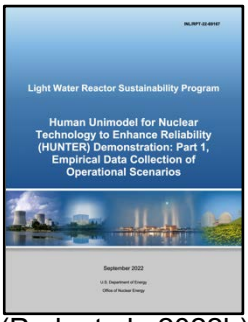
Initial standalone software demonstration of HUNTER



(Boring et al., 2022)

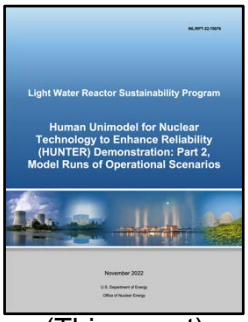
HUNTER 2.1

Data collection of human operators



(Park et al., 2022b)

New scenarios and simulator coupling



(This report)

Figure 4. Evolution of HUNTER.

This page intentionally left blank.

2. SIMULATOR COUPLING

As noted in the previous section, HUNTER versions 1 and 2 were coupled to RELAP5-3D. Both approaches used RAVEN, which served as an API between HUNTER and the thermohydraulic modeling code (Choi, 2021). This worked as intended, but it featured a challenging asynchronous mode of coupling HUNTER to the plant model. For the present purposes, we define *asynchronous coupling* as an interaction between the human and plant model, whereby each operates independently, and values are passed between the models only at the beginning or end of sequences. For example, in an SGTR scenario, the plant model might use inputs on how long a human response takes to initiate safety injection. The plant model would use that input (or a distribution of inputs) to shape the plant model runs. This is considered loose coupling, whereby the plant model runs in batch mode without runtime sharing of inputs and outputs with the human model.

In contrast, *synchronous coupling* involves a continuous feedback loop, whereby the human and plant models exchange information at regular intervals during runtime. The human model does not just serve as an input for the plant model; rather, the human and plant models advance in a stepwise function—the human model impacts the plant model at any point in time, and the plant model provides inputs that determine the human model's next course of action. This type of coupling is considered tight coupling. The two types of coupling are depicted in Figure 5.

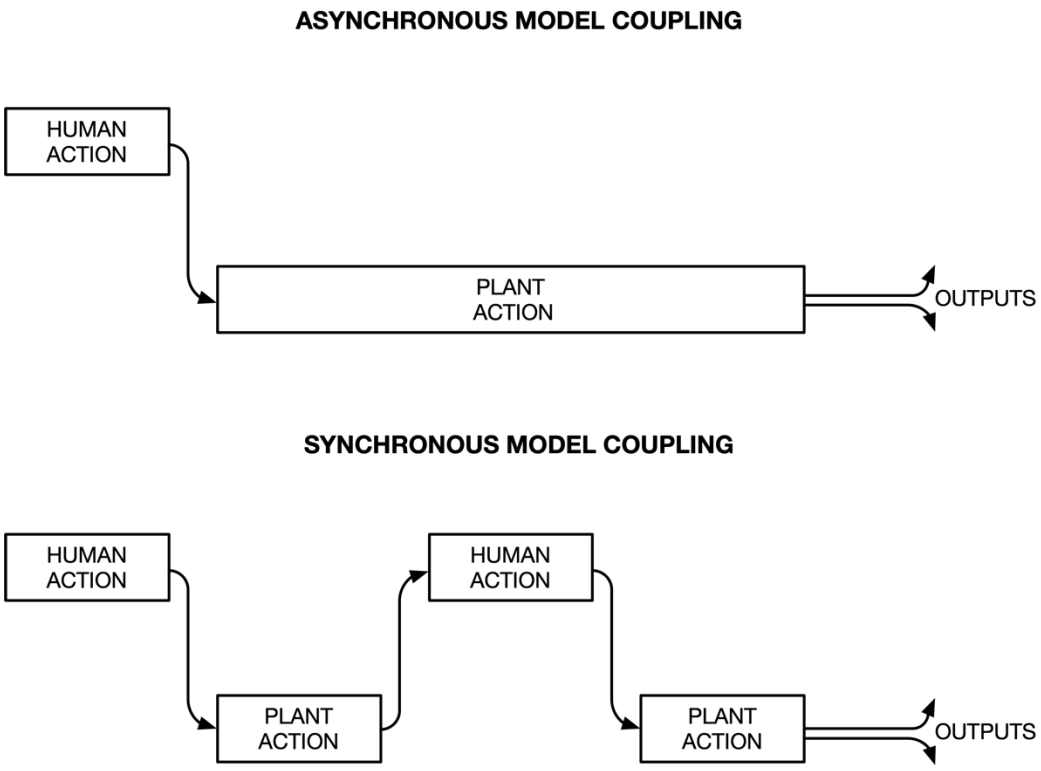


Figure 5. Asynchronous and synchronous human-plant model coupling.

Asynchronous model coupling is most often found in the use of thermohydraulic codes like RELAP5-3D, which are designed to run without interruption to determine the evolution of plant parameters from a particular set of starting conditions. Within RELAP5-3D, it is possible to schedule changes in the configuration, but conditions are determined a priori and not changed once a particular simulation run is

started. Synchronous model coupling is most commonly found in interactive simulators,[1] which feature a system model linked to real-time inputs from a human user. For example, a training simulator at a nuclear power plant operates in such a manner that an input from the reactor operator at any point in time will change the evolution of the simulation run. The simulator provides an evolving response to dynamic contexts that reflect operator actions. The ability to change the simulation run mid-course is the hallmark of synchronous coupling.

HUNTER, as virtual operator, most accurately reflects human-system interactions when it is coupled synchronously with a plant model. While it is possible in the context of procedure-based operations to anticipate most permutations of human-system interactions in an a prior fashion, the process nonetheless becomes complicated when linked to other modules like the individual module. If a model run is measured in time steps, a particular human action might, for example, be anticipated to run between 30 seconds and 3 minutes after a particular plant state is achieved. The range of human response times could be put in the input deck of RELAP5-3D, but the actual responsiveness of the operator might be determined by their stress at that moment, higher stress slowing the time it takes to respond. The stress level is linked to the precursor event that triggered the need for the particular action. Thus, response time is a function of the immediate past event and may not be readily predicted in advance. Multivariate interactions between factors in the individual module, which are linked to the antecedents of the environment model, affect the response of the task model.

The challenge to predict future ($t+1$) human actions (h) can be approximated as follows:

$$h_{t+1} = h(h_t ; s_t) \tag{1}$$

which means that future human actions are a function of current (t) actions given the current plant or system states (s). Likewise, future system states can be approximated as follows:

$$s_{t+1} = s(s_t ; h_t) \tag{2}$$

which means that future system states are a function of the current system state given current human actions. There is a circular chronology if trying to predict the back and forth between the system and human models in advance of either happening. The human actions are dependent on the system state just as the system state is dependent on the human actions. Trying to compute one without the other is nearly impossible in a discrete event simulation.

For this reason, we sought a tighter coupling with the external module or plant model. While a process to allow RELAP5-3D to operate synchronously with HUNTER was developed (Choi, 2021; Heo et al., 2022), the complexity of coupling HUNTER with RELAP5-3D undermines the aim to make HUNTER a standalone code. Additionally, because specific accident scenario models are needed for each RELAP5-3D instantiation, the scalability of HUNTER for running a wide range of scenarios was limited. For example, the SGTR scenario used in a previous HUNTER use case (Boring et al., 2022) required a specific RELAP5-3D model for SGTR. Modeling additional scenarios would require integrating HUNTER with additional RELAP5-3D models, including custom code to facilitate the synchronous coupling.

Instead, we have sought to couple HUNTER with actual plant simulators. Plant training simulators operate in synchronous mode and feature an integrated suite of systems, allowing for a broad range of simulated plant failure conditions. Thus, coupling HUNTER to a plant simulator allows ready scalability across a variety of industry-relevant scenarios while only needing to create and use a single API across all scenarios. In many cases, this API has already been developed, owing to the INL development team's

[1] A model that is executed is a *simulation*, while a *simulator* is a simulation designed to interact with human input. Simulation is typically asynchronous to other models or humans, whereas simulators are synchronous with regular exchanges to other models or humans.

involvement with control room modernization activities (e.g., Boring et al., 2019). The API used to interface advanced digital control room mockups (Lew, Boring, and Ulrich, 2014; Boring, Lew, and Ulrich, 2017) can often be reused for HUNTER, since the digital mockups share base code with HUNTER or at least have a similar software architecture that can be readily linked. Additionally, the tie-in to plant training simulators answers the overriding goal of ensuring HUNTER is a tool that can be used by industry. By coupling HUNTER to actual plant training simulators used in industry, it becomes easy for that plant to use HUNTER for its own modeling activities.

Plant training simulators, also known as full-scope simulators, typically include 100,000 or more internal plant parameters linked to approximately 10,000 input and outputs in the form of indicators and controls on the control room panels. Figure 6 illustrates a glasstop rendering of a control room in the form of the Human System Simulation Laboratory (HSSL; Boring, 2020) at Idaho National Laboratory (INL). A full-scope plant simulator is an ideal platform for coupling with HUNTER. However, it represents a complex environment that can prove difficult to benchmark to crew performance without direct access to operators from that specific plant. For this reason, we have focused our initial proof of concept for synchronous coupling on a simplified simulator, as presented in the next section. The simplified simulator also gives us access to operational data, as described in Part 1 of this report (Park et al., 2022b). The simplified simulator allows similar interactivity as the full-scope plant simulator, just with fewer parameters. This reduced set of parameters is sufficient to develop, test, and validate the synchronous coupling concept, thereby opening the door to future synchronous coupling with a full-scope simulator.

Figure 6. The Human System Simulation Laboratory at INL in 2022.

This page intentionally left blank.

3. RANCOR MICROWORLD SIMULATOR

3.1 Background

The Rancor Microworld Simulator, henceforth referred to simply as Rancor, was developed to address challenges facing human factors and HRA researchers investigating human performance in nuclear process control settings (Ulrich, 2017). Rancor is a simplified nuclear power plant simulator. Over the years we have conducted several studies with naïve and expert operators. Here we tightly couple Rancor with HUNTER. This section describes Rancor and discusses why it is well suited for this effort.

3.2 Rancor Features and Capabilities

Nuclear process control simulators have existed for decades, starting when nuclear utilities developed and deployed mimics of their nuclear power plant main control rooms to support operator licensing and training (Boring, 2011). The requirements for these simulators followed high-profile nuclear incidents, with a goal to ensure reactor operators could safely respond to upset conditions at their plants. Rancor was originally developed to serve as a simulator platform to evaluate situation awareness and attention metrics while participants performed normal and emergency plant operations. The original version of Rancor was developed and evaluated in a series of studies in 2017 and has since undergone several revisions that have enhanced its capability to capture human performance data and increase its fidelity as a process control simulator.

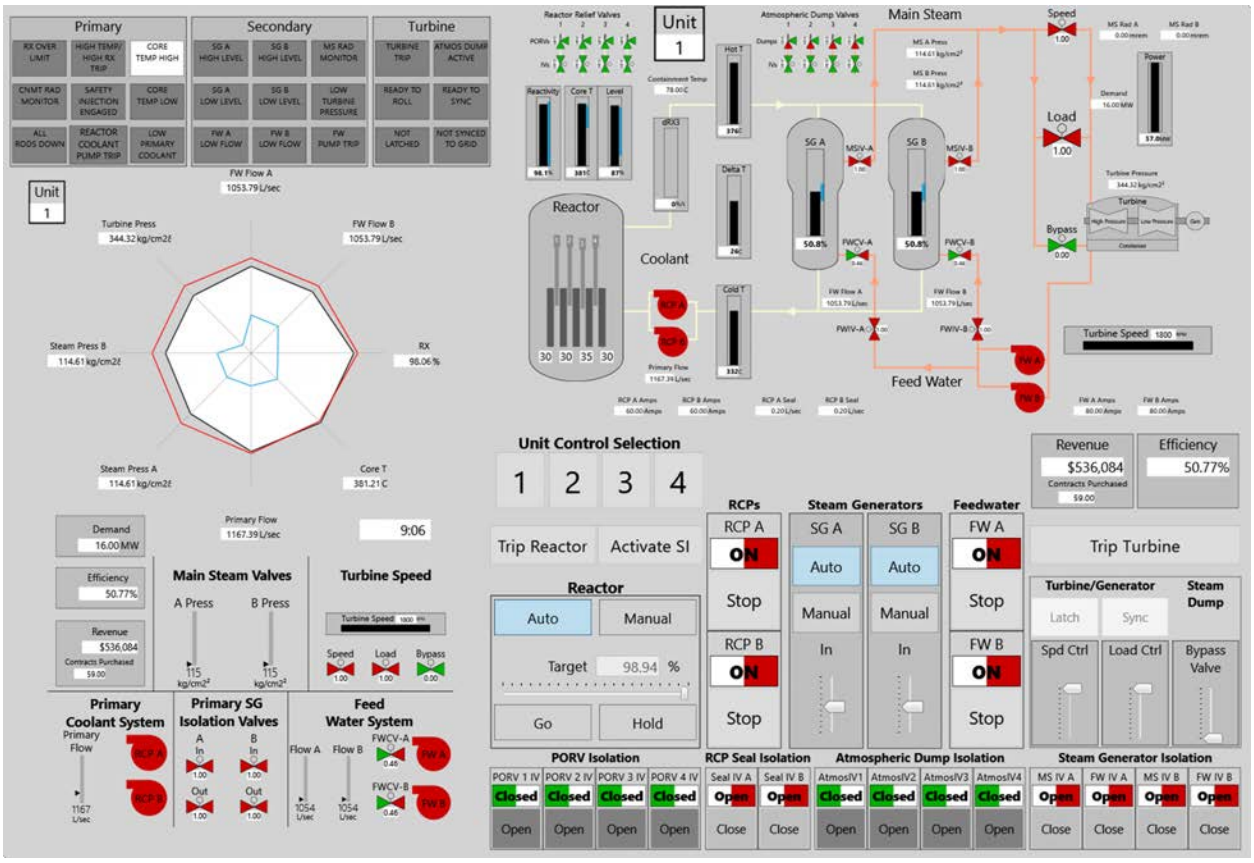


Figure 7. Rancor display as presented to participants in its default configuration.

Since its original development, Rancor has evolved to support specific research questions encountered through collaborative efforts between the INL development team and the fledgling user base. The original version of Rancor was developed to support data collection, and therefore it has a human-machine interface that allows participants to view the system state and make control inputs to adjust the process (see Figure 7). Rancor contains an automatic and manual rod controller for the reactor core, pumps and valves for the primary and secondary systems, setpoint controllers, digital and analog indicators, and alarms. The design and aesthetics are representative of actual indicators and controls found within existing analog nuclear power plant control rooms. Much like full-scope counterparts, Rancor has an executive window that allows the researcher to setup the simulator; load different plant initial conditions; and start, stop, and "snap" (i.e., save) a simulator state. The speed of the simulator can be modified. By default, the simulator speed is tailored to support short trials but also ensure participants have enough time to identify and respond to the plant dynamics.

3.3 Nuclear Process Control Simulators

Prior to its development and use, INL human performance experiments relied almost exclusively on training simulators residing at each nuclear power plant. These training simulators are high fidelity since they replicate the actual process control environment of the main control room. Following legislation issued in response to high-profile nuclear accidents such as Three Mile Island, each plant was required to build a simulator facility and perform ongoing operator training to combat potential human error traps.

However, training simulators have several disadvantages that significantly limit human performance process control research. First, these full-scope simulators can be difficult for researchers to use. They were purpose-built to support operator licensing via training and recertification, and they require highly skilled trainers and simulator staff to operate. In contrast, Rancor is a simplified simulator that represents the same systems and process fundamentals of a real plant, but is accessible to naïve users and researchers that do not have extensive nuclear engineering backgrounds. The training simulator is fully scheduled to support the 8-12 operating crews at a given site, and therefore the schedule affords little time for researchers to use it for purposes other than training. Rancor is flexible because it can be run on a commercial-off-the-shelf (COTS) laptop or desktop computer. Training simulators also contain proprietary and sensitive nuclear operations information that prevents researchers from sharing raw results for scenarios in terms of plant parameters and their relationship to human performance. Rancor does not contain proprietary information that would limit its usage, and a growing corpus of Rancor human-in-the-loop study data is available for HRA research. Additionally, the training simulators are qualified for the plant they support, meaning any system modifications potentially result in a deviation from the required close alignment between simulator and plant. Lastly, the training simulators are part of each plant's license, making utilities reticent to allow researchers to use their simulator as an experimental platform that may elicit negative operator performance. Rancor is a dedicated research tool and does not need to navigate regulatory and legal hurdles involved with using full-scope plant simulators.

Given the limited access and sensitivities to nuclear process control simulators, much of the research examining operator performance issues was completed as secondary activities for some other human factors evaluation performed in collaboration with a nuclear utility (Medema, Mohon, and Boring, 2021). For example, the research team that developed Rancor has participated in over a dozen full-scope simulator studies using training simulators. To bypass the issues with the physical training simulator at the plant, the HSSL was developed to provide an alternative facility that could host a digitally recreated glasstop panel representation of a given plants control room (Boring, 2020). Rancor is a complementary tool to the HSSL. Where the HSSL aims to provide a tool for high-fidelity full-scope scenario simulations, Rancor is a tool for naïve users where fidelity is less important.

Within roughly this same time frame, simulator vendors began to package generic versions of full-scope training simulators. These simulators were based on actual plants, but the models were obscured to

remove proprietary and some sensitive nuclear modeling aspects to support the research community. This was a revolutionary step towards opening the door to researchers. However, the complexity of the simulators still poses a significant barrier to human factors and human reliability researchers. Furthermore, the cost of these simulators and the requisite facilities to display them at full-scale for an experiment is beyond the means of many university laboratories, and therefore only a handful of universities have been able to procure the models and implement a full-scope simulator facility.

3.4 Simulator Features Framework

To better understand the requirements of a simulator for research a literature review and survey study was conducted. The features present in existing training simulators were developed with the singular purpose to support educational and instructional use for initial training and license maintenance for operators. As noted, these features unfortunately do not necessarily align with capabilities that enable effective research. INL staff evaluated existing simulators in terms of capabilities and how these capabilities support nuclear process control research. A literature review was performed to identify simulators actively being used by the research community to evaluate human performance topics in nuclear process control. The literature review identified 14 studies using full-scope simulators with descriptions of the experimental methods, hypotheses, and simulator use to identify capabilities. For each of the identified studies, capabilities provided by the simulator were inferred by examining the collection methods and metrics for assessing human performance and plant status. For measures taken outside of the simulator this provided insight as to the extensibility of the simulator to support external data collection. Descriptions of how data from a simulator was integrated with other data served as another method to identify compatibility between human performance evaluation methods and simulator features. The metrics and data collected from the simulator during each experimental scenario could then be used as a proxy for the capabilities the simulator supports.

From this literature review a framework of the various capabilities was constructed. The framework was then refined through a yet-to-be published survey study eliciting researcher needs and evaluations of their current simulators. The resulting framework identified eight key simulator features or capabilities pertinent to researchers as illustrated in Table 1 below. As can be seen, there are capabilities important to researchers that are not readily available in existing simulators used to conduct nuclear process control human performance research. The following sections briefly describe each of the simulator capability dimensions including sub-elements that collectively represent each of the eight dimensions.

Table 1. Simulator features organized by their level of support in existing full-scope simulators

Standard	Common	Generally Unsupported
Fidelity	Interfacing	Advanced Reactor Concepts
Scenario Configuration	Remote Access	Cybersecurity
Data monitoring and Logging		Human Performance Evaluation Integration

3.4.1 Standard Supported Features

Fidelity, scenario configuration, and data monitoring and logging are features that were available in all simulators identified in the literature review. These features represent the basic functionality for a

simulator. In fact, the fidelity dimension of the simulator feature framework is inherent to all simulators, as their fundamental purpose is to recreate the real-world process with a reasonable degree of accuracy. Though these three features are all present in all evaluated simulators, each feature is uniquely represented within any given simulator, and the differences in the representation can impact their suitability for supporting research.

3.4.1.1 Fidelity

Fidelity refers to the overall accuracy of the physics models and control systems as they reflect the actual plant. Scenario configurability refers to the ability to generate plant initial starting conditions and insert malfunctions on components. Fidelity can be further divided into the following four subdimensions:

1. Thermohydraulic and neutronic steady state
2. Thermohydraulic and neutronic transients
3. Control System transients (timing precision)
4. Control System transients (process value precision).

3.4.1.2 Simulator Configuration

Full-scope simulators, such as GSE Solutions' Generic Pressurized Water Reactor (GPWR), typically include initial plant states known as initial conditions to support a suite of basic scenarios. Trainers can then insert malfunctions to cause system casualties, such as a failed high temperature sensor, that may cause a spurious controller to adversely impact the system. Most malfunctions include parameters to control the severity and dynamics of the malfunction. New initial conditions can be created by maneuvering the plant to a desired state and "snapping" the state of the plant to support a broader range of scenarios. Collectively these features allow for a broad range of scenarios to be simulated. However, sometimes it is not possible to create a scenario using a particular component of the system, since that component is either not represented sufficiently or does not have a malfunction capability associated with it. As a result, some simulators that model components as models of subcomponents can provide richer scenario configurations. The subdimensions include:

1. Provided suite of preconfigured initial conditions (common)
2. Provided suite of preconfigured malfunctions (common)
3. Custom initial condition creation (common, i.e., snapshots or snaps)
4. Paramertization of malfunctions (common)
5. Custom malfunction creation (generally unsupported, requires amending plant model code).

3.4.1.3 Data Monitoring and Logging

The ability to monitor plant process parameters is important to understand the context for any behavioral data collected from operators. Monitoring allows researchers to understand the interactions between the plant states, as viewed through the human-system interface, and operator performance. Data logging and monitoring vary in terms of how easy it is to setup what elements are tracked during a scenario, what types of elements can be tracked, and when data is viewable to a researcher as captured in the following subdimensions:

1. Real-time parameter logging (i.e., live monitoring of parameters)
2. Parameter logging (i.e., time-stamped log runoff parameters)
3. Real-time operator action logging (i.e., live monitoring of operator user inputs)
4. Operator action logging (i.e., time-stamped log of operator inputs).

3.4.2 Commonly Supported Features

This category of features is not present in all simulators, but some do support aspects of these features. Commonly supported features include interfacing and remote access.

3.4.2.1 Interfacing

Research with simulators typically entails some type of modification to the existing simulator system in terms of changing the underlying model to test new system designs. Within human factors and human reliability research domains the human-machine interfaces are often modified to evaluate new methods for presenting information. Simulator integration with additional simulations or physical equipment may also be required to evaluate new designs. As a result, it is important for researchers that the simulator can interact with external software or hardware.. Many simulators provide one or more APIs. The functionality of these APIs are typically fairly simplistic, with the ability to connect, disconnect, retrieve parameters, set parameters, and run simulator commands. It is important that the APIs are documented so that researchers can use them without extensive trial and error. Providing researchers with a means of identifying database parameters and the schemas of parameters is crucial.

3.4.2.2 Remote Access

Full-scope simulators typically assume the operator's physical presence and do not support remote access to the simulator. They do offer portable versions intended to run on laptops that allow operators to take a copy of the simulator across the job site, but there is not active connection to the models running on the server at the full-scope simulator. Furthermore, there are ways to allow simulator support staff to view the simulator remotely for technical support purposes. Live remote access is currently unsupported in most simulator configurations outside of custom implementations. For example, the INL implemented a live data exchange with an intelligent prognostic support system run from Argonne National Laboratory as part of a proof-of-concept Computerized Operator Support System (COSS) demonstration (Lew, Ulrich, and Boring, 2017). The live data exchange allows remote models to interact with the simulator to support advanced concept research such as how a COSS could support operators diagnosing faults. Additionally, other types of models and potentially operators acting on those models could also be included towards integrated testing, which has applications such as flexible plant operations and generation with an emphasis on thermal energy dispatch or steam extraction for hydrogen production as a secondary plant revenue stream. Lastly, from a research perspective, the ability to integrate remote participants has promise to expand the potential for data collection opportunities. In an emerging control room remote concept of operations, the senior reactor operator (SRO) is largely in charge of the operational oversight and with limited displays could possibly perform their duty remotely by directing an operator in the physical control room. This would eliminate the need for SRO participants to travel to the facility. Little research has been conducted using these configurations—only one to the authors' knowledge during the COVID shutdown period—but remote access has potential to reduce the cost of simulator studies while exploring more distributed system interactions.

3.4.3 Generally Unsupported Features

This last category of features represents identified shortcomings in existing simulators based on research trends toward supporting the burgeoning advanced reactor development activities. Advanced reactors include modular, passive safety, and autonomous designs that have no existing simulators at present. The models themselves for these advanced reactors are needed in some form, not necessarily full fidelity, to support a dynamic simulator that can serve research needs for future concept of operations development including fleet level remote monitoring and operation topics. Interestingly and anecdotally, some advanced reactor vendors are still pursuing their simulator development from purely a regulatory basis with simulator features restricted to basic training requirements that largely fall under the base, commonly supported category.

3.4.3.1 Advanced Reactor Concepts

At the time of writing this report, the models for these systems are not readily available outside of the vendors or specific researchers collaborating with those vendors. Future simulators will certainly include the ability to model advanced reactor concepts that are listed below:

1. Passive safety concepts
2. Versatility of thermal energy concepts
3. Digital instrumentation and control concepts.

3.4.3.2 Cybersecurity Support

Existing simulator models do not typically model network communication between components, controllers, sensors, and distributed control systems. Components are modeled to receive inputs and outputs, but the logic and communication pathways between those components are not represented at the network traffic level, and it is therefore challenging to incorporate cybersecurity scenarios within most existing simulator frameworks. The end effects, i.e., a casualty resulting in the loss of a component or the invalid signal sent from one component to another, can be modeled, but to evaluate the more nuanced aspects of the control systems for future advanced reactors, the network layer of the control system will be increasingly important to model. Common requirements for cybersecurity research include:

1. Spoofed process value (i.e., adjusted indication and controls)
2. Degraded process value (i.e., masked indication and controls)
3. Simulated network traffic (i.e., modified communication of indication and controls)
4. Incorporation of hardware in the loop (controllers).

3.4.3.3 Integrated Human Performance Measurement

Despite the initial training impetus for existing simulators, integrated human performance measurement is typically restricted to logging actions of operators taken on the controls. In some cases, even operator actions are not specifically coded in the database, and researchers must comb through the database to determine controller states and identify if the change in process parameter was due to operator input or an automatic control actuation. Simulator training is largely instructionally based with trainers shadowing operators in the control room to evaluate their performance with paper-based criteria-driven evaluation forms. Video recording is included in most if not all training facilities, since it allows instructors to unobtrusively view and review operator actions when evaluating an operator during a scenario. The video recording is not typically linked to the simulator logs themselves. Due to the criteria-driven approach, instructors do not typically evaluate performance from a continuous plant process parameter scale perspective. For example, operators are evaluated in terms of their ability to achieve the objective within a procedure, but not necessarily on how well they restricted flow in a particular line during that procedure if they did not exceed a predefined value—the evaluation is a binary decision of satisfactory or deficient. Researchers optimizing control strategies are more interested in correlating operator performance metrics with specific and continuous plant process parameters to differentiate subtle performance differences. Operators rarely fail at tasks, and the subtle plant process parameter differences contain much of the human performance variability that can be used to optimize and inform future concepts of operations. Currently, the bulk of the human performance and plant process parameter integration is done with expensive commercial aggregation software or in-house custom built aggregation techniques and software. These subdimensions capture specific human performance measurements that are useful for researchers:

1. Plant state event coding (i.e., parameter combinations for event coding)
2. Operator action coding (i.e., operator action sequence patterns)

3. Operator action coding with process parameters (i.e., operator action logging with time-synchronized parameter values)
4. Mouse or cursor position logging
5. Mouse click or cursor selection logging.

The simulator feature framework serves as a tool to evaluate the nuanced capabilities of a simulator that can be informative for a researcher aiming to procure a simulator or determine what types of hypotheses could be addressed based on a given simulator. However, the greater value of this framework is that it highlights the needs of future simulators to bolster the necessary research needed to advance nuclear concept of operations to support new plant builds and large-scale modifications to allow existing plants to continue to operate. The next section translates key aspects of the simulator feature framework into research simulator requirements.

3.5 Research Simulator Requirements

The training simulator research challenges described in the previous sections undergirded the rationale for the development of Rancor. Rancor was conceived out of necessity, and that necessity predated the formalization of our simulator capabilities framework. However, a post-hoc analysis suggests that we were innately aware of many of these challenges and developed much of Rancor's design to address these shortcomings. The capabilities framework was merely presented in this report in this order since it provides context for the design choices made for Rancor. In fact, many of the use cases that Rancor revealed served as an input to what nuclear process control simulators could provide. It was created as a nuclear process control simulator with three critical requirements outlined in the following subsections.

3.5.1 Researcher Accessibility

The simulator platform must be accessible to non-nuclear engineer researchers. Full-scope simulators require immense knowledge of nuclear engineering and operations to build scenarios that can address human performance in a meaningful way. As such, non-nuclear researchers, such as human factors or cyber, must obtain plant subject matter experts (SMEs) to construct scenarios, which may be secondary to the human performance hypotheses. Often the challenging aspects of constructing these scenarios subjugates the original intent of the experiment or makes it challenging to create a suitable experimental design that can address the primary human performance-based hypothesis. Rancor simulator model and operations must be sufficiently simple to allow researchers outside of nuclear engineering to construct scenarios that address the human performance aspects of nuclear process control. The fidelity of the physical process represented by the simulator is deprioritized in this approach, though it emulates basic physical aspects sufficiently to mimic actual plant dynamics. Furthermore, the nature of tasks performed by the operators remains functionally unchanged. This creates the opportunity to advance collective understanding of human performance issues in complex environments that can then be directly applied and further evaluated in a high-fidelity setting as appropriate. Expanding the research opportunity greatly increases the potential to advance the science of human error towards the betterment of human factors processes that are used to ensure safe nuclear concepts of operations.

This general use case allows researchers to construct scenarios to place cognitive demands or challenge participants based on predetermined hypotheses. Researchers administer post trial questionnaires to evaluate psychological constructs such as workload, attentional demands, and situation awareness, identically to how full-scope simulator studies are performed. Debriefs after each trial simply ask the participants to report issues they encountered are surprisingly quite powerful as an analysis tool and also employed extensively in full-scope simulator studies. However, quite often researchers may need to modify the simulator to support research aims. For example, online measures of situation awareness, such as the Situation Awareness Global Assessment Technique (Endsley, 1988), require the simulation to pause, obscure the displays, and then ask targeted situation awareness questions based on predefined

coded events, such as the reactor reaching 100% power. Rancor has a freeze-probe module that supports this type of situation awareness assessment, but this is just one approach. The simplicity of the Rancor architecture and codebase makes it much easier to modify the simulator itself to add data collection capabilities. Furthermore, modifying the human-machine interface or adding new or different functionality fall within the spectrum of use cases. Modifying Rancor is straightforward compared to a full-scope simulator and, indeed, the development team has made numerous alterations to Rancor to support collaborators as described in a subsequent section on studies using Rancor. The simplified simulator architecture allows the entire simulation to be run on a single computer, which eliminates the networking required for full-scope simulators and allows researchers to install the simulator much more easily. The layout of Rancor can be modified to adjust how the displays are presented to participants, such as across multiple displays or in the default condensed single display format.

3.5.2 Naïve Participants

The simulator model and interface must be sufficiently simple such that it affords data collection with naïve participants. Naïve participants are non-licensed operators with limited process control experience (Boring, Ulrich, and Lew, 2018). University undergraduate students are the intended target population since they are numerous, inexpensive, and readily recruitable through university experimental systems already in place across U.S. universities. The ability to use naïve participants carries the greatest potential to bolster nuclear human factors and human reliability research, since it extends the research capability outside of full-scope simulators using licensed operating crews, thereby opening up a much larger pool of research participants. The simplicity of the simulator model and interface is the most beneficial aspect of the platform, since it extends data collection capabilities to those with little training to act as operators, monitoring and controlling representative systems of actual nuclear power plants.

3.5.3 Rapid Scenario Administration

Full-scope simulators represent plant dynamics and timing as accurately as possible to provide accurate operator training. There are short scenarios, but typical full-scope scenarios range from 30 minutes to several hours. Much of this time is simply the operators waiting for the plant response until they are then required to take another action. Trainers and researchers using full-scope simulators bypass this dead time by pausing the scenario and leapfrogging forward in time to the next point in the scenario with relevant human actions. Rancor operates at an accelerated speed, such that the same scenarios performed on a full-scope simulator, can be completed within 25 minutes and more commonly within 10 minutes. The speed of simulator is tailored to provide participants with enough time to respond, whereby the relative timing of different components was held equivalent to the full-scope counterpart and validated by operators to ensure the simulator emulated, albeit an accelerated pace, the nuclear control process they were familiar with. Take a steam generator tube rupture scenario as a comparative example since it is perhaps the most trained on and studied scenario. In a full-scope simulator scenario, the crew can identify and resolve the faulted steam generator in approximately 15-40 minutes, while a single participant can identify and resolve the faulted steam generator in 5-15 minutes in Rancor. As a result, participants can perform many more scenarios and in turn researchers can collect much larger volumes of data within the same time frame.

3.5.4 Generalizability

Simplifying the models representing the plant systems inherently requires diverging from representing the full fidelity of the process control environment. However, strategic simplifications can reduce the complexity without unduly altering the fundamental operating principles or task procedures. The simulator models and interface must functionally reflect the process control tasks as operators perform them in a full-scope simulator. The basic plant systems must all be represented, but in reduced form. For example, a full-scope simulator may have four channels representing redundant sensors for a given component parameter. Rancor may only use one channel to represent the parameter. If there is no functional reason, from an operational perspective, to include these redundant details, they can be

removed from the model. Rancor also achieves complexity by eliminating many of the unnecessary support systems that provide plants with defense in depth or boost plant efficiency. Lastly, Rancor reduces much of the complexity by relying on simplified physics models that would not be sufficiently accurate for it to serve as an engineering reference model, but the dynamics of the physics are close enough to mimic the operational experience in terms of the cognitive demands required to work through a component casualty or bring a system from a cold shutdown state to online.

3.6 Previous Rancor Studies and Use Cases

3.6.1 Initial Testing

The focus for the first series of experiments using Rancor was on the validation and viability of the platform to serve as a research tool (Ulrich et al., 2017b). First, as psychology students represent one of the largest available participant pools, it was deemed necessary for the platform to be relatively easy to learn without a nuclear or engineering background. Second, it was evaluated in terms of its ability to generate variability in nuclear process control performance and cognitive measures. Third, the platform was evaluated in terms of performance and cognitive measure variability, as they differed between naïve students and experts with process control experience. Lastly, through training and experience using Rancor, the learning time course for students to demonstrate performance resembling experts was evaluated. For any results to be generalizable to nuclear process control domain, the simulator should generate variable effects between and within experts and naïve students. It is challenging but nonetheless necessary to ensure the simulator is sufficiently complex to capture variability among experts but also sufficiently simple that it can simultaneously generate variability among naïve participants. Of course, it is also important to establish the extent to which naïve participant results generalize to represent the results of more experienced operators.

During this initial set of experiments, participants received an hour of training including background on the systems, indicators, and controls for Rancor. Participants completed an experimenter-guided practice session prior to four experimental trials. Each trial required the participant to manipulate the system from a shutdown state, i.e., reactor core with control rods fully inserted, to online power producing mode of operation, i.e., full reactivity with steam generators producing steam for the turbine with the generator synced to the grid. This represents a compressed set of normal operations.

Faults are often used to evaluate operator diagnostic and action responses to a degraded component. Perturbations comprised of a spurious reactor or turbine trip were included in each trial, and these forced participants to diagnose and then execute a series of control actions specific to the trip type to restore normal operations. These scenarios are perturbations and not true faults, since no components, sensors, or controls were compromised and all functionality was available. An actual fault entails a component casualty that forces operators to address the issue with varying levels of reduced system functionality. Future versions of Rancor addressed this shortcoming and will be described in subsequent sections.

Student performance and learning effects were compared to a sample of steam plant and nuclear power plant operators. There was evidence for an initial effect of expertise on primary task performance, but the relationship was more complicated due to interactions for the cognitive factors of situation awareness, workload, and attention. This initial set of experiments established basic viability for Rancor to support nuclear process control research.

3.6.2 Thermal Dispatch System Design and Evaluation

The next research activity Rancor support focused on using student participants to evaluate usability issues for a prototype thermal dispatch system human-machine interface (see Figure 8) intended for integration with existing U.S. commercial light water reactors. This research demonstrates the use of Rancor in tandem with a full-scope simulator. The thermal dispatch system extracts steam from the main steam line before the turbine and diverts it to a nearby chemical process, such as hydrogen production

plant. The research aimed to develop an prototype display to control the thermal dispatch system and demonstrate this with a full-scope simulator running GSE Solutions' GPWR (Ulrich et al., 2021). The prototype display was developed with input from licensed operators during a quasi-dynamic evaluation. After this evaluation, Rancor was modified with a model of the thermal dispatch system and the prototype interface, which in itself is relatively simple for a nuclear process control system. Students performed the same test scenarios presented to the operators in the quasi-dynamic evaluation, and their feedback was collected. Though students lack the nuclear expertise to comment on operational impacts of the thermal dispatch functionality, their perceptual capabilities are similar to licensed operators, and therefore human factors issues were expected to be reported. Despite differences in expertise, the students and operators reported similar types of usability issues. Therefore, the students augmented the small sample of operators and afforded a more comprehensive evaluation with greater confidence in successfully detecting all pertinent human factors issues. The university collaborators then refined the display design (see Figure 9) based on the outcomes of the student evaluation. This refined prototype was then tested again with licensed operators across fifteen different trials including normal and fault scenarios using the full scope simulator.

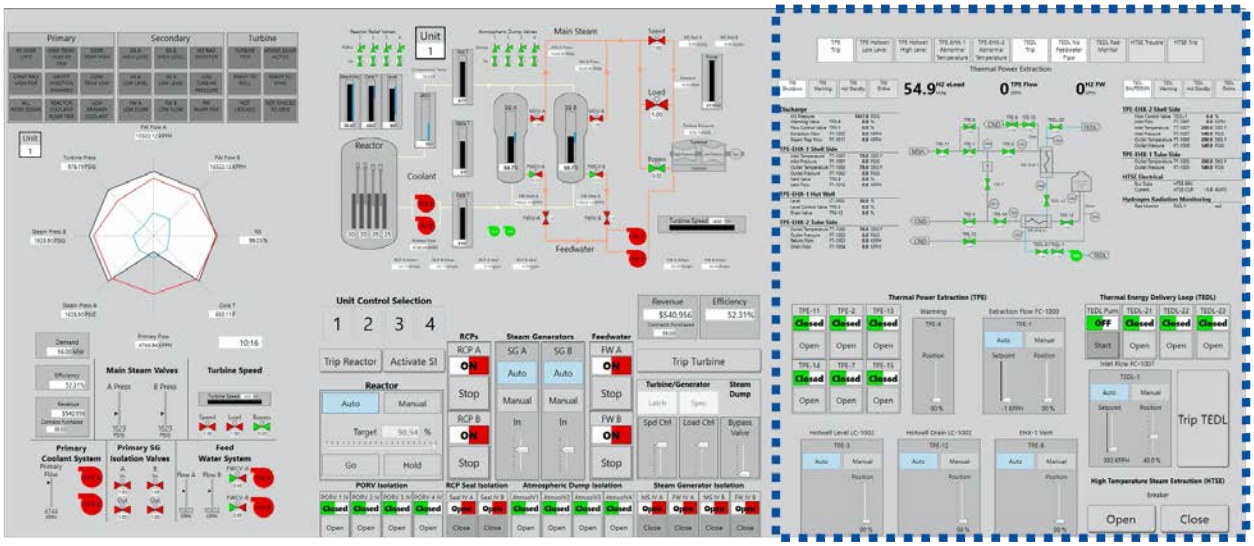


Figure 8. Rancor augmented with the thermal dispatch prototype evaluated by students to resolve human factors issues prior to its operational evaluation with licensed operators in a full-scope simulator.

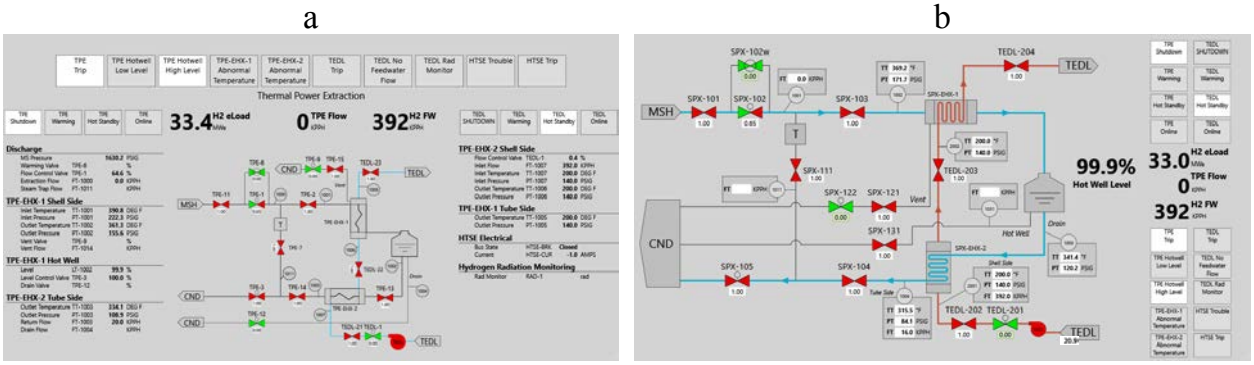


Figure 9. The initial thermal dispatch prototype display depicted in (a) contrasted with the revised display (b) based on student feedback and performance data.

The simplified plant model highlights both the advantage and disadvantage for Rancor to support control room operations research. Using Rancor appropriately is critical to ensure that the results translate back to the nuclear process control domain. Within the context of this thermal dispatch system development, it was necessary to first have a viable but still immature design and concept of operations before modifying the Rancor models and testing the general usability of the display and operations within the simplified environment with naïve participants. Experts must provide input and verify the design before it is translated to the reduced order model and tested with students; otherwise, there is no basis for the validity of the testing, and it yields little benefit. Subsequently, the usability issues identified by the students must then be validated, which is already ensured in control room development activities through NUREG-0711, *Human Factors Engineering Program Review Model* (O'Hara et al., 2012).

Rancor is best suited to serve as a formative phase simulator platform performed as an intermediate evaluation. The primary benefit for employing Rancor to address usability issues is the cost savings avoided by not using the full-scope simulator to vet the design for general usability issues. The full-scope simulator can then be used for high-fidelity scenario testing in which the plant responses and operator feedback are of more concern. The scenarios can avoid becoming bogged down in usability issues and the focus can remain on evaluating the human-machine interface's ability to support the primary task for controlling the thermal dispatch system as was the case for the second operator-in-the-loop study of the thermal dispatch system development (Ulrich et al., 2021).

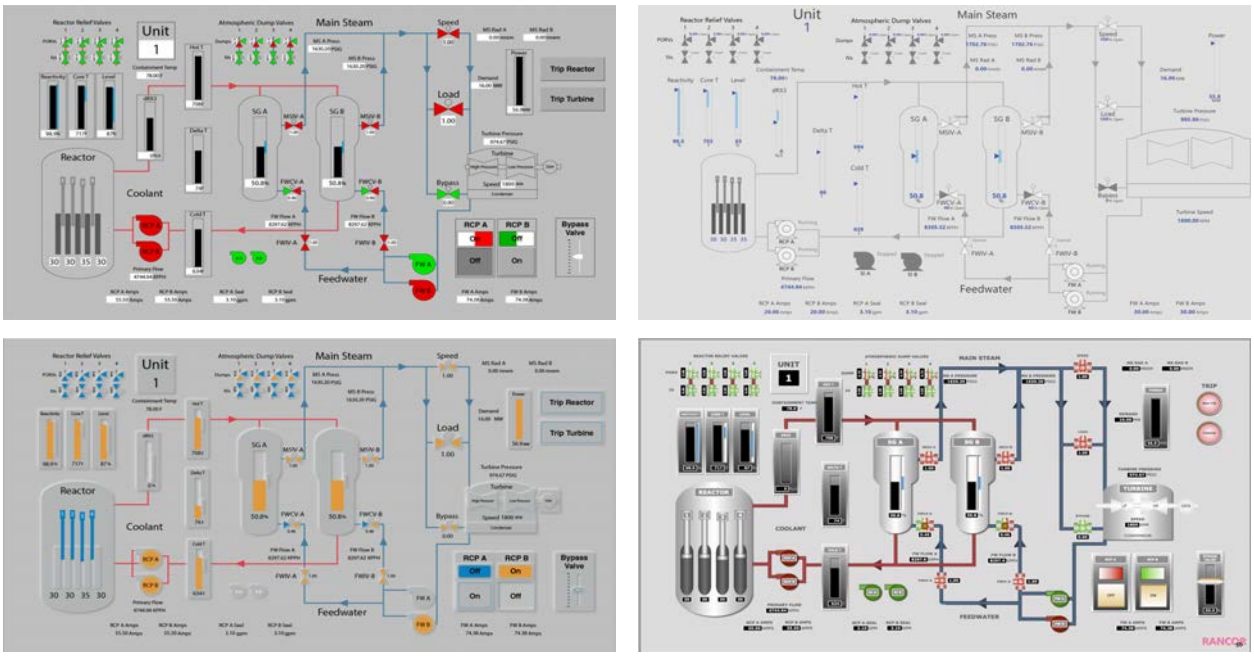


Figure 10. Different Rancor styles used to evaluate user preference and performance data for an advanced reactor vendor control room concept development study.

3.6.3 Advanced Reactor Control Room Development

Numerous advanced reactor vendors are faced with developing a control room concept that affords efficient operations through high levels of automation. Unlike existing commercial plants, fully developed thermohydraulic models are typically not available to support designing and testing the control room concept. Deferring the control room design and evaluation until the reactor design is finalized is unwise,

as this would push the operational considerations to the tail end of the overall system design process and leave little room to address any deficiencies. This is especially prescient for advanced reactor vendors, since reduced staffing is a requirement to remain cost competitive during operations. Rancor provides an agnostic simplified model that can be coupled to control room design concepts to support evaluation. Through a collaboration with an advanced reactor vendor, Rancor was used to perform user preference and performance evaluations for some basic design concepts (see Figure 10), representing initial efforts to develop a control room concept. As mentioned in the previous section, Rancor is not well suited to test the robustness of the displays to support specific operations in comparison to a full-scope simulator. Testing the specific implementation of control schemes for specific systems is not useful in the absence of fully matured system design. However, Rancor can and has been used to develop the reactor vendor's style guide comprised of the graphical elements, nomenclature, navigation, and layout of the displays for the control room design concept. To date one study has been performed and at the time of this writing. This first study illustrates how Rancor can be used to support advanced reactor control room design.

In the study, the Rancor display graphics were stylistically altered to represent competing industrial process control design styles. Four different styles were implemented including the basic Rancor design, high performance human-machine interface, three-dimensional graphics, and neumorphic design (respectively represented clockwise in Figure 10). Static images were generated for each design with several different Rancor states. An online survey was developed to assess user preference and capture limited performance data with questions prompting participants to identify states or values within each display style. The collaborating vender provided engineering and operational personnel to complete the survey. Preferences, accuracy, and performance were captured and analyzed. The results identified a preference for the basic Rancor style. The three-dimensional style exhibited the greatest accuracy and shortest response times to identify states or values. These results were contrary to expectations that predicted the neumorphic style to be the most preferred and the three-dimensional to demonstrate the poorest performance. Rancor follows a simple industrial design aesthetic and therefore its preference is not surprising. Familiarity may also have driven performance, as participants working in this domain encounter three-dimensional graphics routinely despite their typically poor human factors. Familiarity may have augmented participants' ability to interpret the display states and improve their accuracy and response times. Regardless of the study outcome, Rancor was an effective platform in identifying preference and performance across different styles. Furthermore, since Rancor is system design agnostic, the style evaluation alleviated vendor reported issues in which participants became distracted by the system engineering design as opposed to the style. The follow-up study moves beyond the static image survey and tasks participants with controlling Rancor across small scenarios.

3.6.4 South Korean Rancor Validation Study

The final major Rancor-based research activity aimed to gather human performance data to support HRA. HRA typically uses full-scope simulator data with operating crews (Massaiu, et al., 2010); however, this proves to be an expensive and time-consuming process. Furthermore, it is difficult to acquire sufficient data and effect sizes to properly quantify human error classification categories and rates. The suitability of Rancor as a platform to evaluate human performance was the central question for this study as it could potentially expand data collection capabilities if the results are representative for a traditional full-scope simulator study. To validate the use of Rancor for HRA data collection, it was necessary to compare metrics of human error in Rancor using an existing framework to determine if the metric assessments were representative to human error measured in a full-scope simulator.

To perform the validation against existing full-scope simulator data, a collaboration between INL, Chosun University, and the University of Idaho was performed to test Rancor's ability to evaluate human errors (Park et al., 2022b). Twenty nuclear engineering students and twenty licensed reactor operators from South Korea completed ten scenarios requiring the detection, diagnosis, and response to plant faults. Rancor was updated to support actual component faults beyond the initial perturbation scenarios used in

the original validation experiment. Furthermore, procedures were drafted to support normal operations and emergency responses, such as a steam generator tube rupture response procedure.

The results of the study demonstrated promise for Rancor to generalize, at least with respect to HRA. There were some notable differences. The ratio of errors of commission to errors of omission was larger in Rancor than what was observed in previous full-scope data collection studies. The participants committed more errors than they omitted correct actions respectively. Furthermore, the rate of errors overall was significantly higher in Rancor. Rancor appeared to overestimate the human error probability in comparison to a full scope simulator based on this result. Additional research is needed to further understand these differences, but the ability to capture these errors is a fruitful outcome. It is unreasonable to assume the error rates would match, but the amount of variability should be predictable. Future work is planned to further explore the use of Rancor to gather HRA data and examine how to extrapolate the results systematically to augment existing human error data sets. The data from this study, which were summarized in Part 1 of this report (Park et al., 2022b), provide empirical data to help understand, model, and validate a HUNTER implementation of the same scenarios.

3.6.5 Rancor as an HRA Plant Simulator

As the four use studies reflect, Rancor was developed to support existing research on control room concepts by offering a simplified nuclear process control environment to evaluate human performance constructs using naïve student participants. It has been used to evaluate human factors issues and HRA across several different research efforts. Because Rancor addresses the full range of simulator features while remaining a relatively simple model, Rancor is an ideal choice for a process simulator to tightly couple with HUNTER (see Section 2 of this report). The tight coupling allows the virtual operator in HUNTER to observe and interact with the Rancor nuclear power plant. Note that the version of Rancor coupled with HUNTER in this report contains only the simulation models and not the human-machine interface. This non-graphical version of Rancor runs faster than real time at a highly accelerated rate to support Monte Carlo simulations (see Figure 11).

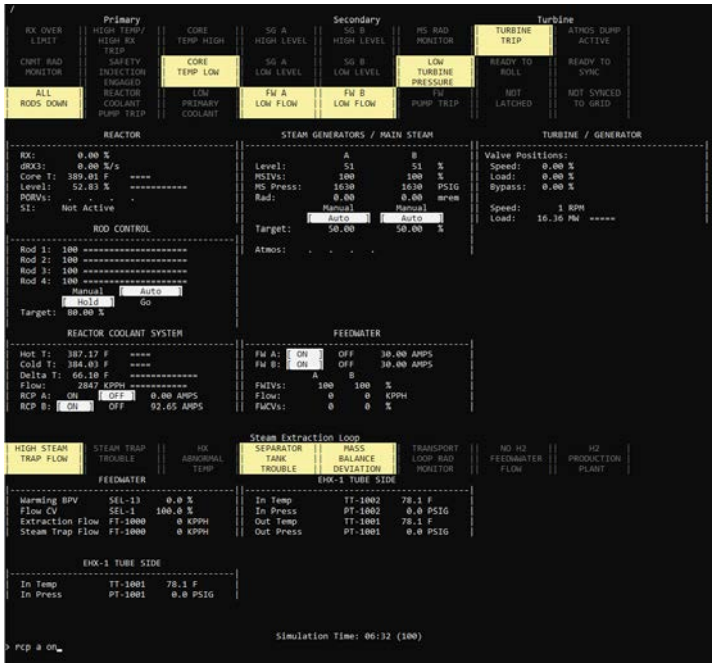


Figure 11. Early non-graphical console interface used for debugging the simulator model when running Rancor.

This page intentionally left blank.

4. IMPLEMENTATION OF HUNTER-RANCOR

4.1 HUNTER Software Architecture

Before we discuss HUNTER coupling with Rancor, we first describe some details of HUNTER's software architecture that are necessary for interfacing with Rancor. As previously described, HUNTER is a virtual operator framework for dynamic human reliability modeling. The version 1 framework was translated into a simulation application written in the Python programming language as HUNTER 2. The simulation code supports the ability to execute analyst-defined scenarios organized as tasks comprised of procedures. The procedures within the tasks are predefined as inputs to the application and contain all the necessary data elements to execute proceduralized tasks based on the simulated nuclear plant state. The state is used to evaluate the logic within each procedure step, and the virtual operator completion of each step is evaluated with a dynamic HRA module that uses the simulation context to calculate completion durations, HEPs, and success or failure.

4.2 Proceduralized Operator Task Models

A central aspect of this dynamic HRA software implementation is the proceduralization of activities to drive the virtual operator within the simulation. This model forms the roadmap that the scheduler references as it progresses through the simulation (Ulrich et al., 2022). When performing a task, operators execute a series of procedures or sections of procedures to arrive at the intended goal, i.e., plant state. Given its central importance to the simulation, the structure of procedures influences the software architecture. NPP operators use several different types of procedures, but from an HRA perspective, the most pertinent procedures are emergency operating procedures (EOPs) and abnormal operating procedures (AOPs).

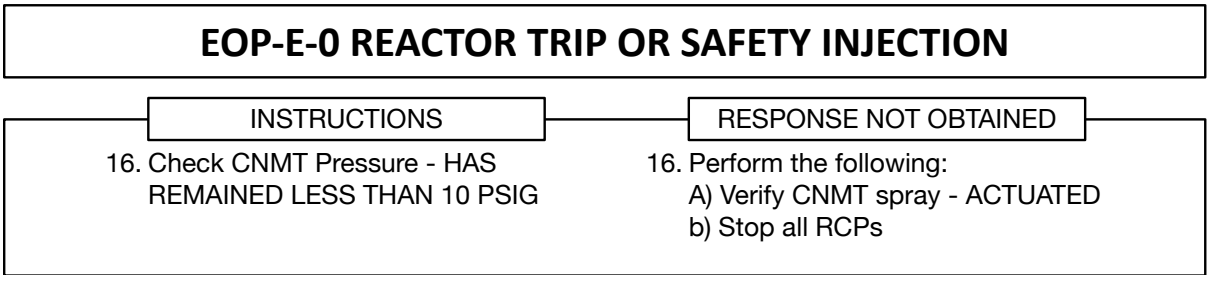


Figure 12. Example emergency operating procedure in the two-column format.

For present purposes, we will focus on pressurized water reactors (PWRs). PWR AOPs and EOPs follow the two-column format (see Figure 12) in which the left Instruction column is the primary route towards a goal plant state and the right Response Not Obtained (RNO) column is used when criteria are not satisfied while completing a step in the left column. To complete a procedure, the operator executes each procedure step sequentially until all are completed, or the operator encounters a step with criteria that are unsatisfied and alternative activities are a required. Immediately after encountering failed criteria, the operator branches into the corresponding RNO column step, denoted with the same step number or substep identifier. The RNO step may entail performing other actions or require a transition to another step or another procedure entirely. The Rancor procedures are represented as a single column, but use the same logic in that a response obtained leads the operator to the next step and a response not obtained can branch to a different step. Thus, the logic of two-column and Rancor procedures are functionally equivalent. The procedure representation in the HUNTER software architecture can be seen in Figure 13. These elements include the following basic structures:

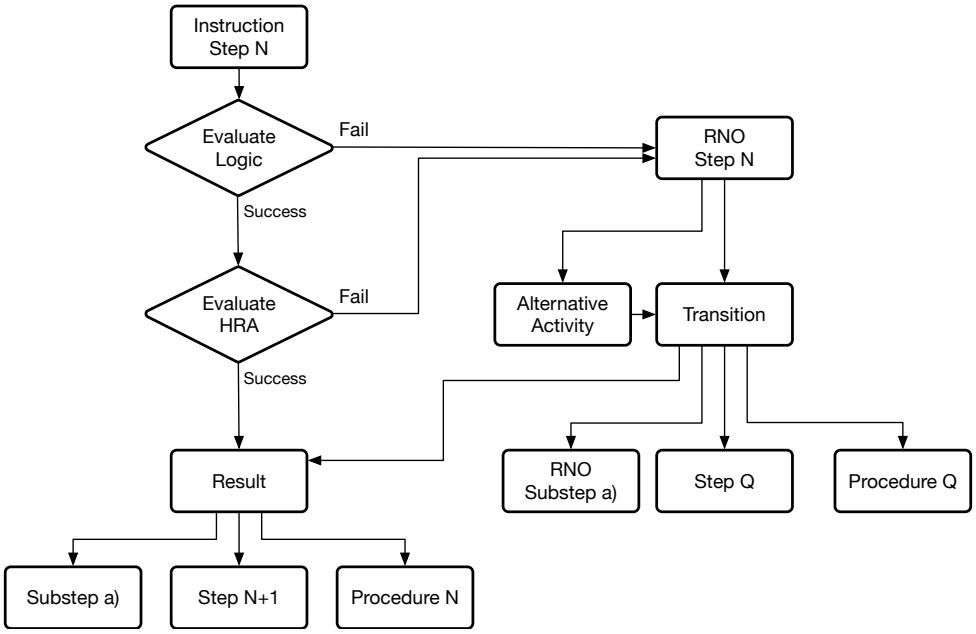


Figure 13. Basic procedure step execution in HUNTER.

1. ***Procedure***—within the HUNTER architecture, the Procedure class contains a collection of ordered Steps, metadata regarding the procedure (authors, date, plant).
2. ***Element***—within the HUNTER architecture, Element is a base class for Step, Condition, and Action classes.
3. ***Step***—refers to a small set of activities that represent the base organization unit for a procedure (a procedure is an ordered collection of Steps. As noted, procedures are commonly organized in two columns of steps as exemplified by those used at PWRs. The primary tasking steps occur in the left column, denoted as instructions. If the criteria for the instructions are not met, then the operator moves to the right RNO column, which offers an alternative set of activities that may align with corresponding instruction column's goal. Not all instruction column steps have a corresponding RNO step, depending on the nature of the procedure. By design, all procedures attempt to be closed such that any step's success or failure branches to another step within the same procedure or transitions to a different procedure.
4. ***Preconditions***—refer to an ordered collection of Condition elements that must be affirmed to proceed to the next element of the step. Preconditions can include aggregated individual conditions following different types of logic including any or all being affirmed. Preconditions can trigger a prescribed action, or they constitute the entirety of the step itself to represent diagnostic steps within a procedure.
5. ***Actions***—refer to operator action taken to manipulate the state of a component or system. Actions may have preconditions that must be met prior to their execution, but a step could contain only an action. Postconditions are implicitly present in each step containing an action as the controller state change. They can also be explicitly included as defined plant parameters expected from the control state change.
6. ***Postconditions***—refer to an ordered collection of Condition elements that must be affirmed following an action taken within a step. The postconditions represent the plant response to an operator action. As such, postconditions must be affirmed before proceeding to the next serial

step. When postconditions are not affirmed, a response is not obtained and the step is unsuccessful.

7. ***Substeps***—refer to an ordered collection of Steps that are performed within a procedure step. Often a procedure step requires multiple substeps by operators to complete the desired tasking. Substeps function as a grouping mechanism to link a series of steps that accompany one another to complete a subtask. Steps and Substeps are enumerated. Substeps can be enumerated alphabetically (e.g. 1a, 1b, 1c) or numerically (e.g. 1.1, 1.2, 1.3).
8. ***Condition***—Preconditions and Postconditions contain statements that instructs the operator to determine a particular plant state. The Condition elements contain logic evaluates to a Boolean (i.e., True or False) based on one or more plant parameters.
9. ***Transition***—specifiesy the next element to evaluate based on whether or not the current element can successfully be completed.

To support the dynamic human reliability aspects of the simulation, contextual information is linked to the Step, Condition, Action, and Transition elements. This is done because each step activity is assigned a GOMS-HRA primitive to provide a time duration associated with completion of the task and HEP for the task (Boring and Rasmussen, 2016; Ulrich et al., 2017a). The simulation is able to use the plant state to evaluate the context to calculate nominal time durations and HEPs. Additional details on the modules evaluating the context and plant state are provided in a subsequent section. Each procedural Step, Precondition, Action, Ppostcondition, and Substep includes the following information:

1. ***Primitives***—refer to the GOMS-HRA task level primitives (Boring and Rasmussen, 2016), which the HUNTER code uses to determine time durations and nominal HEPs for tasks. More than one task level primitive may be associated with a step.
2. ***PSFs***—refer to performance shaping factors that are dynamically or statically calculated during the simulation and applied to the *primitive*. The nominal HEP and time duration sampled during the simulation are multiplied by the combined PSFs to adjust the simulated time duration and the HEP value at each time point in the simulation.

4.3 Implementational Modules

This section describes the modules as they are used to run each of the Monte Carlo simulations. Note that these modules describe pieces of software code used to execute HUNTER. The *functional* modules described earlier include the individual, task, and environment modules. The *implementational* modules include software code that supports the overall execution of the functional modules. In many cases these are transparent to the user of HUNTER and are therefore backend code required to make HUNTER run.

4.3.1 Plant Model

Before describing the implementational modules, the plant model itself (a.k.a., the environment module) is worth describing. The plant model is not in itself a module, but it is a crucial aspect of the overall simulation and enables many of the dynamic capabilities provided by HUNTER. A plant model is run in tandem to track the plant state as the virtual operator manipulates components based on the procedure as well as the natural progression of the simulation without intervention, which is typically initiated with a fault and progresses towards a system failure state without operator intervention. Therefore, the virtual operator's success or failure is largely governed by the amount of time remaining to mitigate a given fault based on the amount of time required to complete the mitigating activities. The scheduler travels along the procedure path while polling the plant state through an API module to determine whether key parameters have exceeded threshold values indicating a system failure and the end of that simulation run. In its current version, the API module supports communication with RELAP5-3D and Rancor simulations.

4.3.2 Scheduler Module

The Scheduler module includes several classes that collectively perform Monte Carlo based simulations of the task defined through the comma-separated value (CSV) input files. The scheduler acts as the executive for what is being done by which module and when. Practically, it serves as the placekeeper for the other modules. The scheduler stores analyst-defined configurations for the overall simulation in a configuration class that is accessible throughout the simulation to serve as a central data repository for the application. The Scheduler module has access to all the other modules and contains several subclasses itself, most notably the log class that outputs data to CSV log files.

The log class serves as the historian and performs input/output functions to record each simulation run in CSV output files. The log class also contains some debugging capabilities to assist analysts in testing the CSV input files and logging errors in procedure path execution, such as an unclosed procedure path with no possibility to advance. As the scheduler is executing simulation runs, it is monitoring the runs to cease any failed runs and move to the next run attempt.

4.3.3 Task Module

Within each simulation run, the Scheduler calls the Task module to execute the procedure steps. The Task module contains the classes that store and manipulate the activity executed during each simulation run. The Task module contains the procedures with their steps. The module can execute unlinked procedures in sequence, but in practice the task typically begins on a specific procedure at a specific step, and then the steps themselves contain all the information to guide the simulation since each step contains the appropriate transition to other procedures in the task. Specifically, each step contains a transition object that contains an identifier for the next step based on the results of its own successful or failed execution. These transitions can be:

1. The next instruction column step within the procedure sequence, or
2. The adjacent RNO column step or substep, or a step in another procedure.

Each step is executed, and the result of the execution contains the appropriate next step item. The steps within the procedures of the task are completed until no more instruction steps remain or an end simulation flag is encountered. While executing each step, the HRA module calculates the elapsed time and an HEP for each step.

4.3.4 HRA Module

The Task module relies on the HRA module to evaluate the plant context and performs two key functions. First, it calculates an elapsed time for each step or substep activity and increments the simulated time for the main simulation *and* the plant simulation. This is important to progress the plant state in step with the time durations required to perform each step activity. Second, the HRA module calculates an instantaneous HEP, which is used to alter the course of the procedure path or incur a time debt. The HEP aspects of the module are still quite limited, but the simulator can fail a step based on a calculated HEP exceeding an analyst defined threshold. Based on the analyst defined configuration, the step is repeated, and a time debt is incurred, which over several failed steps can lead key parameters within the plant simulation to exceed their thresholds such that the simulation run ends in a system failed state. The other approach that is supported currently is to treat the HEP threshold exceedance as a procedure logic failure and follow the failed procedure step transition to the next appropriate step. In this configuration, the failure leads the virtual operator down the wrong path. The critical element that binds the Task and HRA modules together is the Step module, which contains the parameters used by the HRA module.

4.3.5 Step Module

The Step module is responsible for executing each step, which entails evaluating the plant state against the logic defined within the step and evaluating the HRA context. The Step module contains a step class, which serves as the data structure to store parameter objects defining each step. One key parameter object is the logic which contains the plant model component parameter and the logic test used to evaluate the state of the component. The step class also contains several HRA parameter objects. GOMS-HRA task-level primitives are defined in the primitive subclass parameter object. Each step can contain multiple primitives to represent more complicated activities, but in practice there should never be more than two based on initial simulator validations. Ideally a single primitive should be assigned to each step or substep, but due to the variability within procedure steps, more complicated steps that do not contain any substeps may require two or more primitives to capture the intended tasks. Primitives are defined with the GOMS-HRA coding scheme, such as C_C to denote the "check in the control room" task-level primitive. Each task-level primitive type has a predefined execution time distribution and HEP associated with it. The various contexts surrounding the virtual operator are defined in the PSF class. Relevant PSFs from the SPAR-H method can be assigned with default levels corresponding to specific multiplier values that are applied on top of the base HEP linked to the GOMS-HRA primitive.

Shown earlier, Figure 13 depicts the different data elements and their relationships in terms of how the Step module evaluates each procedure step. The evaluation of the step serves both as a datapoint recorded in the log of the simulation run and dictates the next procedure step selection for the simulation. Figure 14 below depicts how the Step module evaluates each step by using the API to the plant model and HRA module to gather plant state data to evaluate the step logic and evaluate the HRA context.

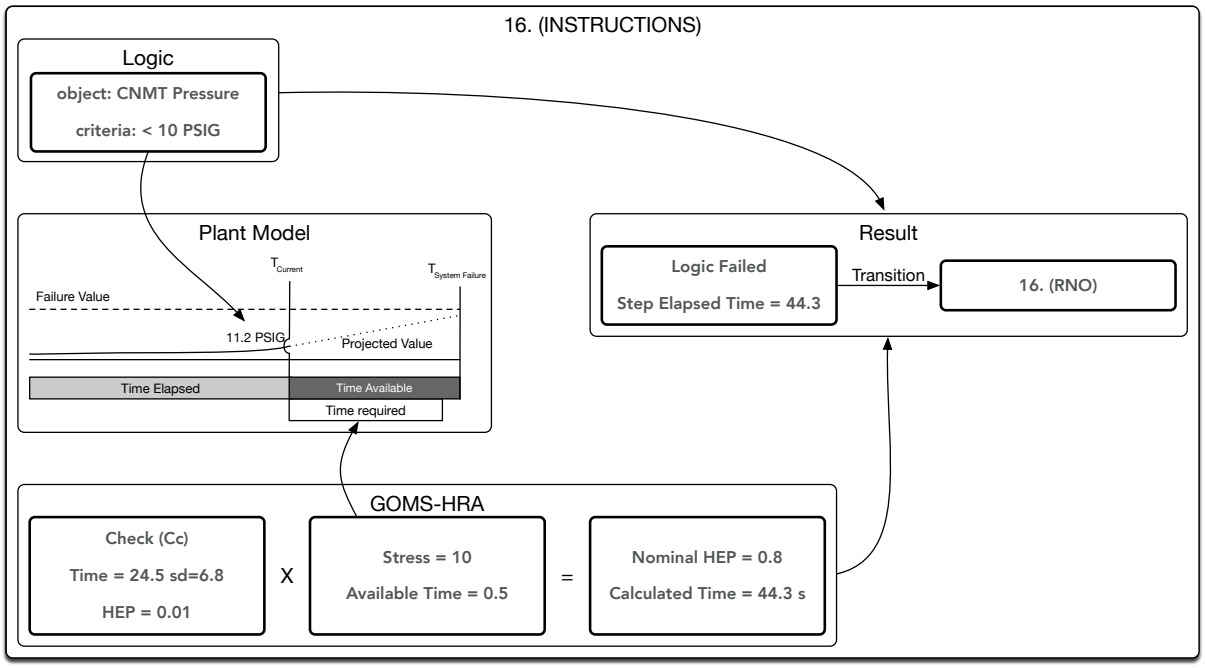


Figure 14. Step execution example using the instruction step, left column, from the sample procedure in Figure 12.

All of these elements do need to be defined within each step. In fact, HUNTER can be run without the plant model to perform time uncertainty quantification estimates that purely look at the variability in the execution of particular tasks. In this manner, the simulation model must define the plant state outcome for each step, since there is no way to evaluate the logic in the absence of a plant model running in tandem. The intent of the dummy mode is to support examining the human reliability variables along a prescribed path to examine known scenarios or validate a model against an empirically observed scenario data set.

4.4 Procedure Schemas

Our previous efforts described a procedure schema for two-column procedures commonly found in nuclear power plant AOPs and EOPs for use in the HUNTER framework. Here we have extended the schema such that it is possible to represent the logic of procedures with single-column instructions and maintain existing two-column format logic support. The new schema also still allows for defining substeps within steps, but has added some additional elements to support new step subelements. In its base form, procedures are a collection of steps that are generally followed in a consecutive order. The simplified schema is as follows in Table 2 with the additional precondition, Actions, and Postcondition elements.

Table 2. Simplified procedure schema.

Procedure
Steps = List[Step]
Step
Preconditions = List[Condition]
Actions = List[Action]
Postconditions = List[Condition]
Substeps = List[Step]

The schema is intended to provide flexibility for defining procedures. However, best practices for writing procedures should likely only include a subset of what the schema allows for. For instance, procedure guidance could dictate that a step should either contain substeps or preconditions, actions, and postconditions; but not both. Procedure guidance could also dictate that substeps should not contain substeps. Conditions, Actions, and Steps can also define transitions that specify procedure flow when the response is obtained or when the response is not obtained. Though there are basic elements and structures common across the industry, there is still some variability in how procedures are implemented. Therefore, the use of these different elements may vary depending on the plant being modeled.

4.4.1 Advanced Procedure Authoring Features

The procedure authoring feature in HUNTER has incorporated advanced features to assist in the authoring of plant procedures to eliminate repetitive and tedious task declarations.

4.4.1.1 Proxy Definitions for Steps, Conditions, and Actions

Procedures often contain repeated steps, conditional checks, or actions. Proxy definitions allow a procedure author to define *macros* for Steps, Conditions, and Actions that can be referenced by their

identifier (ID) and reused in procedures. This increases the consistency of procedures and increases the efficiency of procedure writing.

4.4.1.2 Conditions and Aggregate Conditions

The procedure schema supports two types of conditions. The plain Condition type evaluates whether a single statement regarding plant variable is true. The author specifies a component ID and a logical evaluation statement such as "x >= 3." The value of the component ID is replaced with x before evaluating the statement. To enforce code security, the evaluation statement can only contain x, <, >, <=, >=, ==, numbers, and Booleans as tokens.

Procedures often have verbiage with multi-item r logic such as: "If any of the following conditions are met, then perform the following…." The Aggregate Condition type allows for multiple conditional logic statements to be evaluated. The analyst can specify whether any of the conditions satisfy the Aggregate Condition (i.e., "OR" logic) or whether all the conditions must be met to satisfy the Aggregate Condition (i.e., "AND" logic).

4.4.1.3 Procedural Control Statements

The transitions can be defined as procedure and step IDs (e.g., AOP-0001, Step-1) or as Procedure Control Statements similar to established procedural programming languages. HUNTER has implemented the following control statements (see Figure 15):

1. Pass
    - Do not evaluate transition. Transition based on parent element
2. Next
    - Go to the next sibling element
3. Continue
    - Stop evaluating this element and move to the next sibling element
4. Break
    - Stop evaluating this element and skip remaining sibling elements
5. Exit
    - Exit the procedure
6. Recurse
    - Repeat current substep or step.

The control statements allow procedure authoring without needing to specify procedure step numbers explicitly. The alternative to procedures is to specify a procedure ID and element ID. The use of control statements serves to make the procedures more generic and easier to modify. For example, steps can easily be inserted or reordered without having to worry about renumbering all the subsequent transitions in a procedure.

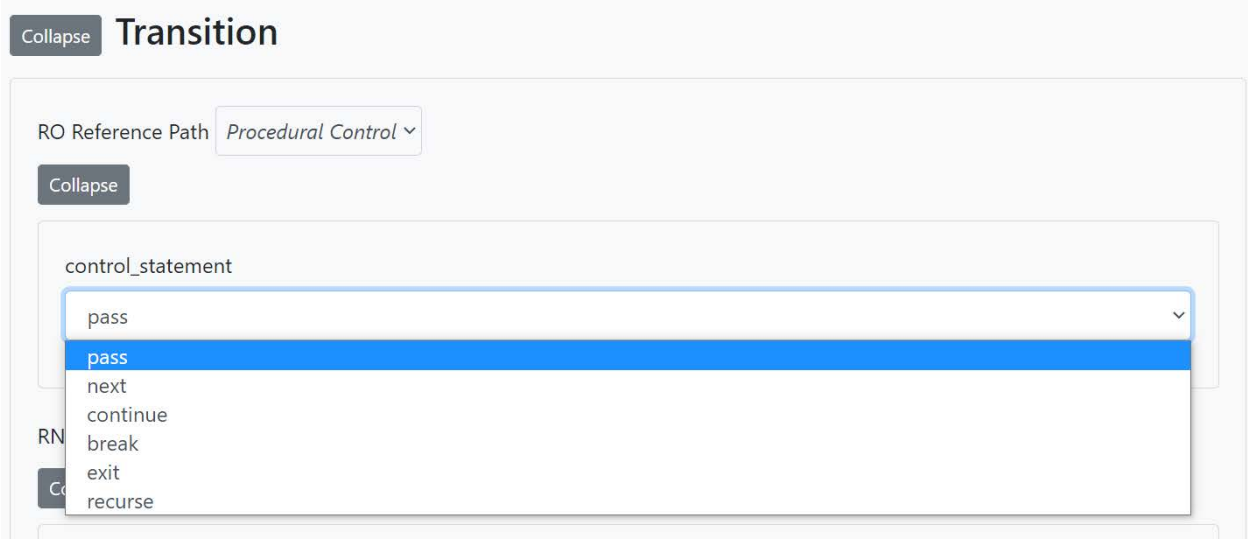


Figure 15. Screenshot of the dialog for selecting a procedural control statement transition.

4.5 Configuration Schemas

HUNTER simulates human activities and human performance within those activities. All simulation models must contain model definitions for the three functional modules: individual, task, and environment (see Figure 2). HUNTER uses JSON files for configuring model parameters (see Table 3). The environment includes the plant simulator and faults defined in a scenario configuration. The individual's reliability is defined by parameters in the HRA-Engine configuration. Lastly, the task is specified by a task configuration file, which includes a list of procedures to execute.

Table 3. HUNTER configuration schemas

Scheduler
HRA-Engine
Task
Procedures
Scenario
Simulator Units
Initial Conditions
Faults

The JSON files are used to deserialize HUNTER model instances. The configuration could be specified as a single object with a single schema. However, to support code use and organization we have divided the configuration across five nested schemas: Scheduler, HRA-Engine, Task, Procedures, and Scenario. The configuration schemas are arranged hierarchically as shown in Table 3 and described next.

4.5.1 Scheduler

The scheduler configuration contains the high-level simulation parameters (see Figure 16). This includes the number of iterations that should be performed, the maximum number of elements that should be evaluated for each iteration, the starting time on shift for the virtual operators (as a static time or a dynamic function), the ID for the HRA-Engine configuration, and ID for the task to be performed, and the working directory to save model outputs. The scenario depicted specifies that 490 Monte Carlo iterations of the task should be performed. The virtual operator should use HRA parameters specified by the dynamic_fatigue_random_t0 model, and the task that should be performed is the loss_of_feedwater task. The starting time on shift specifies that each iteration task should start after the virtual operator has been on shift for 0 to 12 hours (i.e., 0 to 43,2000 seconds). This latter configuration is useful for consideration of fatigue, which is one of the dynamic PSFs modeled in HUNTER (Park et al., 2022a). Note that other configuration parameters can be included for other PSF modeling. Fatigue parameters are included for illustrative purposes only and should not be considered the main drivers on performance.

```
def_id:                 "scheduler"
hra_engine_id:          "dynamic_fatigue_random_t0"
iterations:             490
max_evaluation_limit:   500
task_id:                "loss_of_feedwater"
time_zero:
  def_id:               "random_generator"
  distribution:         "uniform"
  high:                 43200
  low:                  0
wd:                     "test_rancor_lofw_scram"
```

Figure 16. Scheduler configuration for a loss of feed water scenario.

4.5.2 HRA-Engine

The HRA-Engine configuration defines the parameters for the operator's human reliability model, as shown in Figure 17. The static_tmult parameter specifies a baseline fatigue value that is used to calculate the time to complete the GOMS-HRA primitives, the assumption being that performance time is influenced by fatigue. The time_on_shift_fatigue parameter specifies whether the dynamic fatigue model should be used. The static_tmult is set to 0 and time on shift fatigue modeling is on so the HRA modeling will use the fatigue function previously discussed. The static_tmult is a bias parameter for this function, which could be used to calibrate the timing or emulate operators with varying experience.

```
def_id:                 "hra_engine"
description:            "Operator with Dynamic Fatigue and random start of shift"
static_tmult:           "0.0"
time_on_shift_fatigue:  true
```

Figure 17. Example HRA-Engine configuration.

4.5.3 Task

The task configuration specifies a list of procedures needed for the task, as shown in Figure 18. By default, the operator starts the first procedure in the list. The configuration also specifies a Simulator Scenario ID, which links the simulator configuration to the task. The operators should enter the eop-0002_loss_of_feedwater scenario. The aop-0001_rapid_shutdown is performed if feedwater cannot be restored. The task should run the loss_of_feedwater scenario in the simulator or plant model.

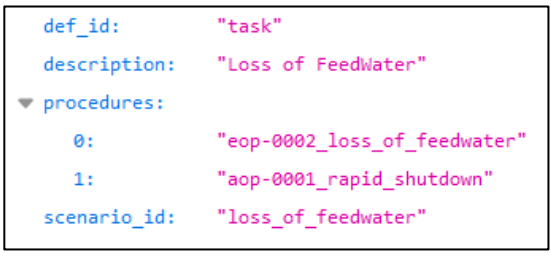


Figure 18. Task configuration for the loss of feedwater scenario.

Figure 19. Example simulator scenario configuration file for a loss of feedwater scenario using the Rancor Python model.

4.5.4 Simulator Scenario

The simulator scenario configuration specifies the plant and the faults (see Figure 19) as interfaced to Rancor in the environment module. The plant can contain 1 or more units, and each unit has an initial condition specified as well as optional faults. The faults can be triggered by a time step, a simulation time, or an event trigger. Each fault contains a list of malfunctions. The malfunctions specify a value to set in the plant or method to execute.

4.6 HUNTERweb Tool

A low-code (no programming required) web-based configuration analyst interface has been developed using Python/Flask. The low-code approach allows analysts to build HUNTER models without needing extensive programming skills. Flask is a lightweight web development framework that connects with the Python programming language. The HUNTERweb tool can be conceptualized as a database creation and authoring tool. Generally, JSON schemas are defined by a developer and added to HUNTERweb. HUNTERweb is a powerful authoring and model-building tool that complements the original GUI tools developed in HUNTER 2 (Ahn et al., 2022).

HUNTERweb uses a JSON Schema Code Editor library (https://github.com/json-editor/json-editor) to provide a web-form based tool for authoring configuration files (see Figure 20).

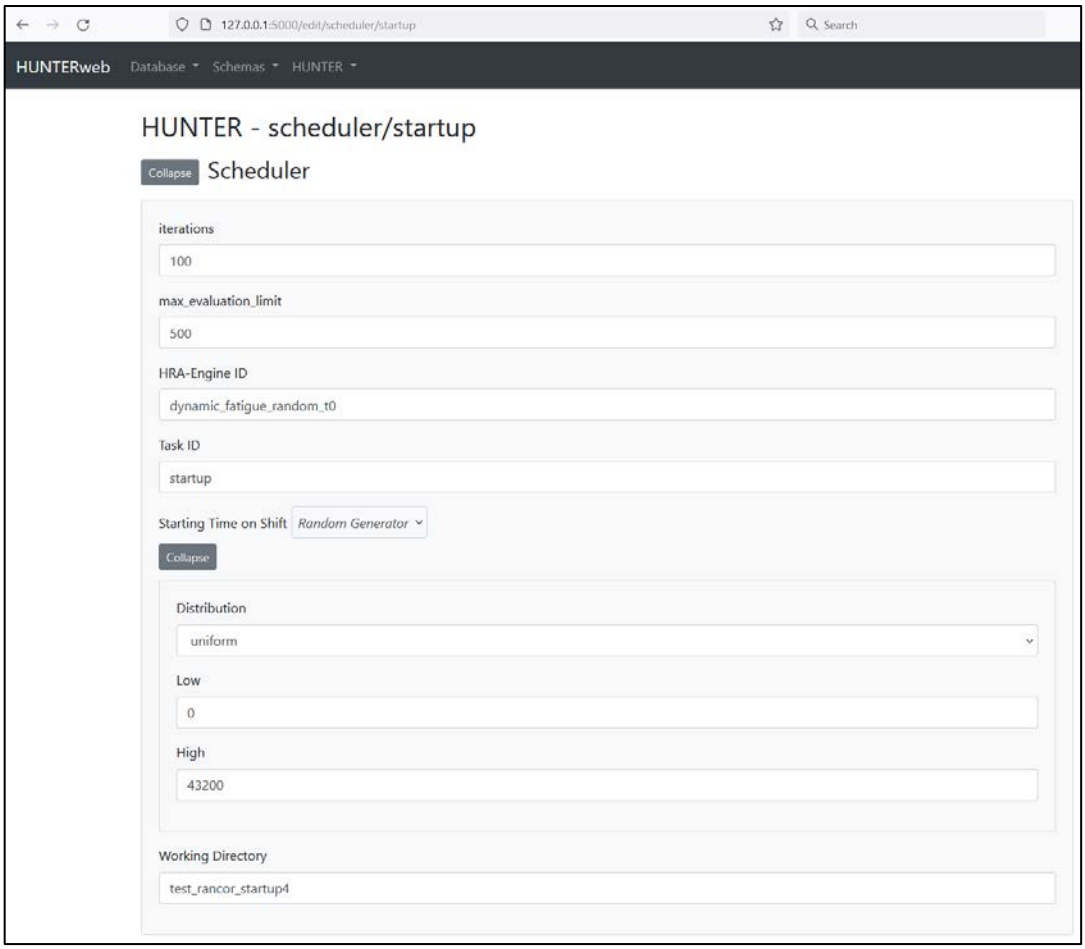


Figure 20. Screenshot of HUNTERweb editor with example startup scenario.

Flask/JavaScript functionality fills in additional features for providing a catalog listing of configuration files, viewing files, and saving and deleting files. The database functionality is implemented in a manner that is agnostic to the underlying schemas (see Figure 21 and Figure 22). The databases are directory based, allowing HUNTERweb to be web-deployed. In such a scenario, users could have logins to maintain separate databases from other users or ensure data protection. The databases may also be locally stored on a local computer drive or a corporate network shared drive.

Figure 21. HUNTERweb database management is schema agnostic.

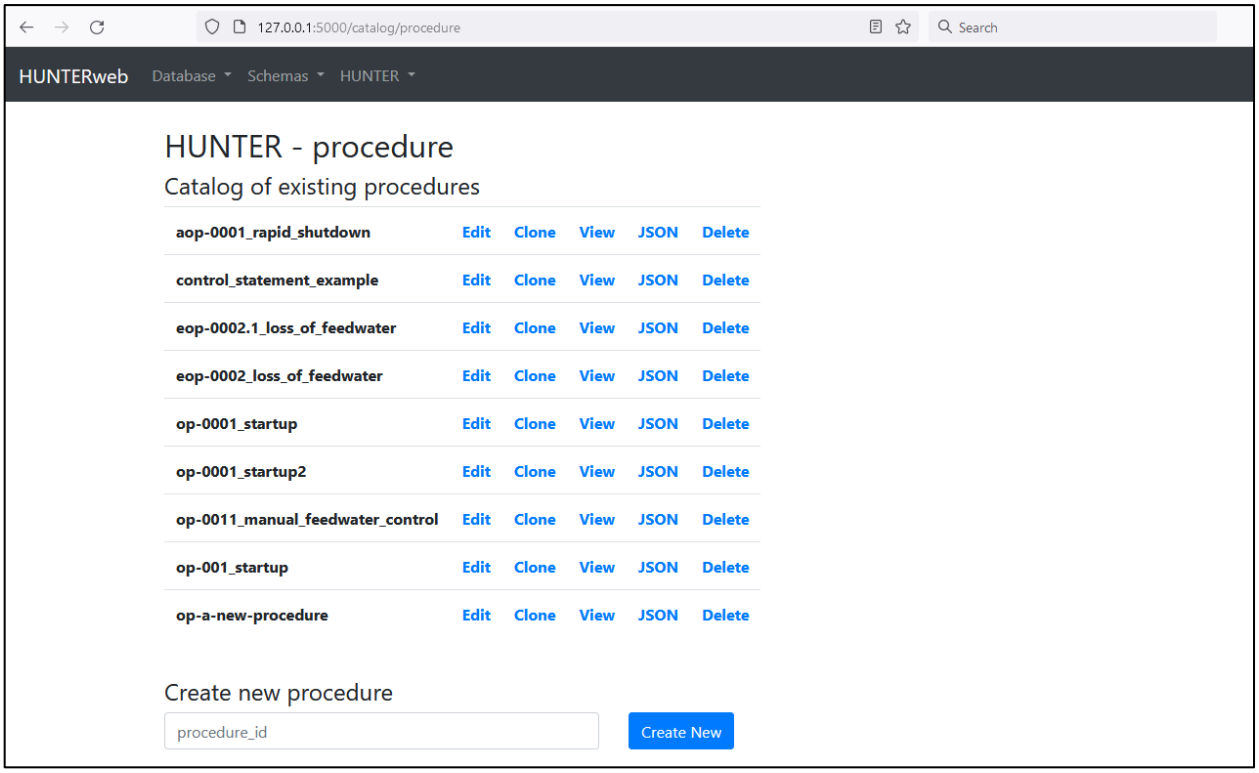


Figure 22. Screenshot of the HUNTERweb tool displaying a catalog of procedures.

4.7 Rancor Python Model

The University of Idaho and INL jointly developed Rancor. The primary implementation of Rancor is in Microsoft's Windows Presentation Foundation (WPF), which runs as a standalone application in the Microsoft Windows operating system. The Rancor model was developed as a human factors research tool for nuclear power operations with naïve participants. The plant model from this version was ported to Python. This implementation significantly reduces the complexity of integrating HUNTER with a simulator-in-the-loop by allowing the simulator to be a class instance within the HUNTER framework. Alternatively, the simulator would need to be run independently, and communication would need to occur through an API or a remote procedure call framework.

The fully Python approach eliminates these complexities and allows debugging through both HUNTER and Rancor source code during runtime to troubleshoot code. After verifying that the HUNTER code meets quality expectations it will be easier to integrate HUNTER-in-the-loop with non-Python models such as the GPWR simulator.

Another key advantage integrating a reduced order model like Rancor is the ability to run faster than real-time. Python Rancor can, for example, run 140x real time on a ten-year-old Windows based machine. This allows hundreds or even thousands of task runs to be modeled in a couple of hours. In contrast, full-scope simulators are typically locked at real-time or limited to 2x or 4x real-time. Because the authors developed Rancor and HUNTER, we have full knowledge of the code-base and were able to simplify the integration to keep Rancor in lock-step with the virtual operator in HUNTER.

Python Rancor also has a newly developed interactive user interface implemented with Textual 0.2.0 (https://www.textualize.io/blog/posts/textual-0-point-2-point-0) to allow analysts and operators to run and

interact with the model (see Figure 23). This is a purpose-built monitoring interface for HUNTER that builds on the non-graphical version of Rancor shown earlier in Figure 11.

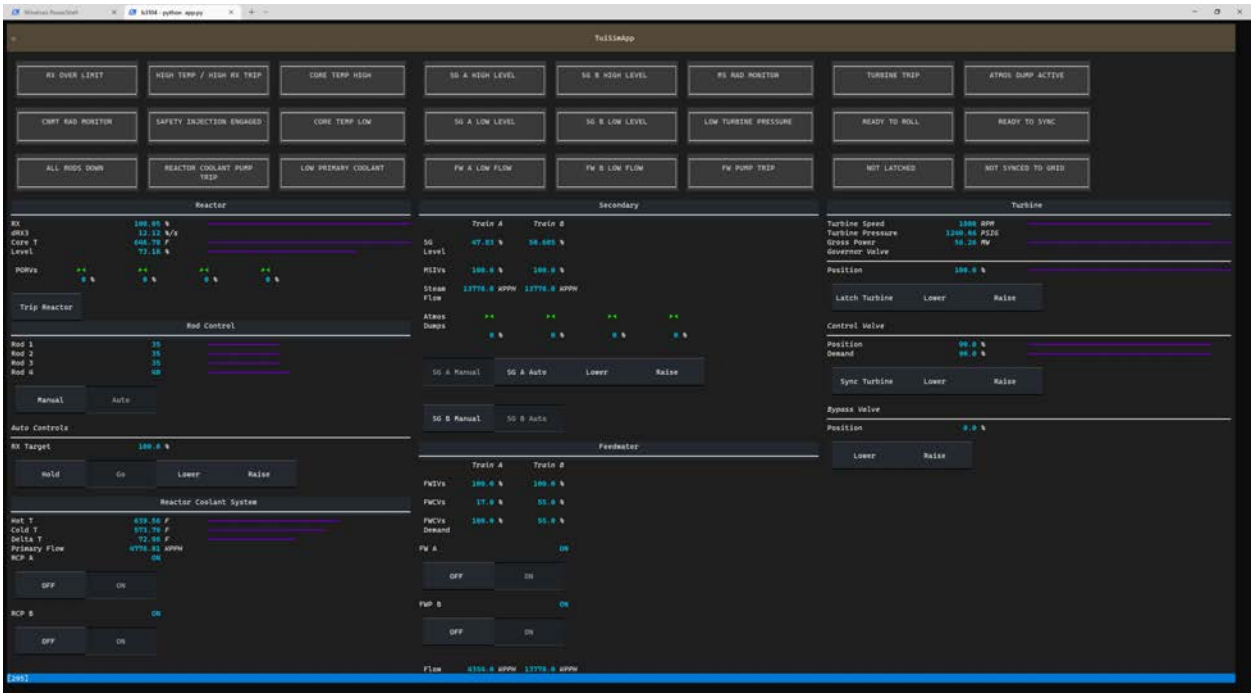


Figure 23. Textual user interface for Rancor Python model running in display terminal mode.

5. HUNTER-RANCOR SIMULATOR RUNS

5.1 Introduction

Each scheduler specifies an output directory for HUNTER outputs. The HUNTER Monte Carlo simulation outputs four hierarchically organized CSV files for primitive level, element level, procedure level, and task level data. As seen in Figure 24, the top panel contains task level statistics. The second panel contains procedure level statistics. The third panel contains element level statistics. And lastly the bottom panel contains task level primitive statistics.

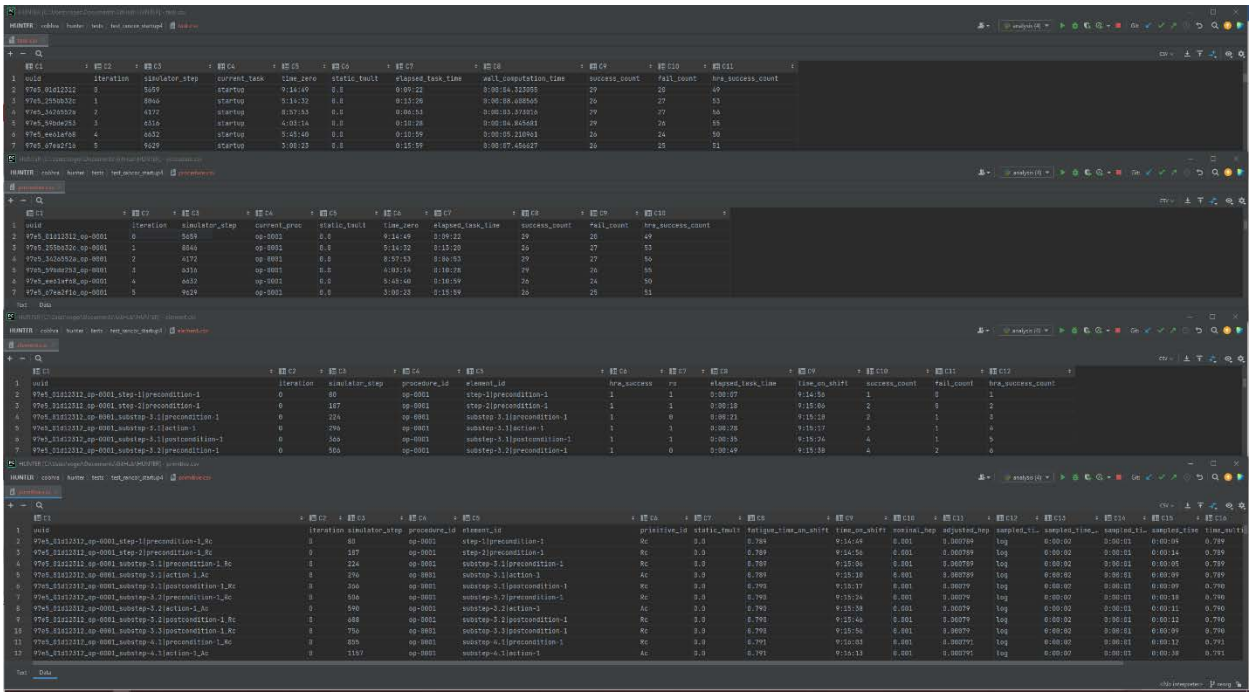


Figure 24. HUNTER outputs from a startup scenario Monte Carlo simulation.

Each time the scheduler is run, a 4-hexidecimal digit universally unique identifier (UUID) is generated. Then, for each iteration, a unique 8-hexidecimal digit identifier is appended to the scheduler parent. Then for procedure the procedure ID is appended. And for each step/element an enumerated ID is appended. Lastly, for each primitive the primitive ID is appended. So, for example the UUID "97e5_01d12312_op-0001_step-1|precondition-1_Rc" came from the *97e5* scheduler, from task *01d12312*, on the *op-0001* procedure when they were performing *step-1|precondition-1* and evaluating the R_c primitive.

For each event the HUNTER logs record the simulator step. This corresponds to Rancor outputs that can be cross-referenced during analyses to determine the state of the plant. For each iteration Rancor produces a timeseries CSV file and an event JSON file. The names of these files follow the format: <scheduler_id>_<task_id>.

Two of the ten scenarios from Park et al. (2022a) were selected as development and demonstration scenarios. The two selected scenarios are:

1. Loss of feedwater, and
2. Startup from cold-shutdown to 100% power.

The loss of feedwater scenario is an emergency scenario with low complexity requiring operators to rapidly shutdown the plant by following a series of actions in a prescribed order. The startup procedure is a normal operating procedure with higher complexity due to the need to coordinate the operation of plant subsystems.

The requisite procedures were re-written using the HUNTERweb tool, and GOMS-HRA primitives were assigned. The GOMS-HRA primitives used for these scenarios had the parameters listed as follows in Table 4.

Table 4. GOMS-HRA talk level primitives for scenarios in HUNTER.

Primitive	**Distribution**	**Location**	**Scale**	**Nominal HEP**
R_C - Retrieval Control Room	lognormal	2.11	0.6	0.001
A_C - Action Control Room	lognormal	2.23	1.18	0.001
D_P - Decision based on Procedures	exponential	0.02	N/A	0.001

5.2 Scenario 1: Loss of Feedwater

5.2.1 Description

The loss of feedwater scenario is a simple fault condition where both of the feedwater pumps spuriously trip, causing abnormally low feedwater flow. The loss of feedwater occurs with the plant online and at 100% power. The operators are tasked with following an EOP to try and restore feedwater. The plant fault does not allow for feedwater to be restored, and the operators must enter and complete a rapid shutdown procedure.

5.2.2 Implementation

The virtual operator follows *EOP-0002 Loss of Feedwater* and *AOP-0001 Rapid Shutdown* to verify that feedwater flow has been lost and attempts to restore feedwater flow by manual turning both pumps back on. When the feedwater flow cannot be restored, the virtual operator rapidly shuts down the plant by placing the turbine on bypass and manually tripping the turbine and reactor. HUNTER was configured to run 500 iterations of the startup scenario with starting time-on-shift between 0 and 12 hours. The time-on-shift affects the dynamic fatigue calculation. The loss of feedwater procedure was authored using the HUNTERweb procedure authoring interface, and Appendix A contains a rendered version of the procedure.

5.2.3 Results

The virtual operator was able to complete the *Loss of Feedwater Procedure* and *Rapid Shutdown Procedure* in all 500 simulated evolutions, as shown in Figure 25.

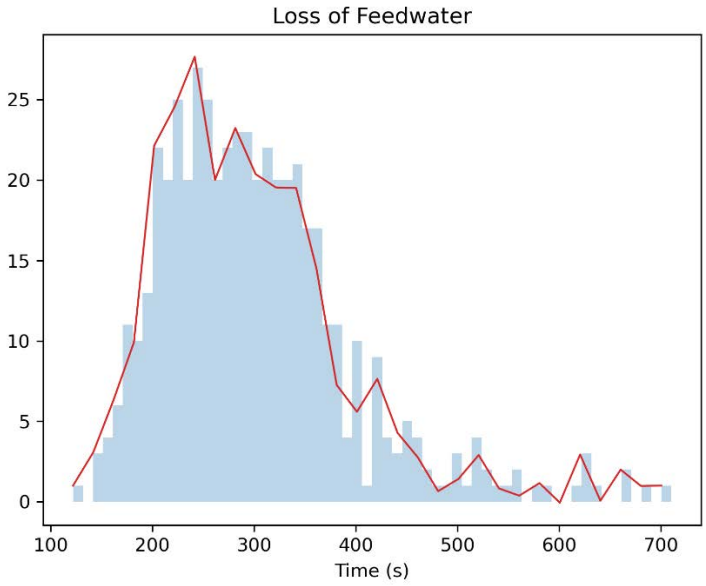


Figure 25. Distribution of times for HUNTER to complete loss of feedwater.

5.2.4 Discussion

The virtual operator was able to complete the loss of feedwater scenario, and the timing and successful task completion is consistent with human operators (see Table 5) as reported the Part 1 report (Park et al., 2022a). The virtual operator had an average completion time of 302 seconds. The Chosun University dataset had 10 students and 4 operators complete the Loss of Feedwater Scenario. The students took 195 seconds on average to complete the scenario, and the operators took an average of 154 seconds to complete the scenario. Here we can observe the virtual operator is slower than the human operators. In Section 6 of this report we will discuss this difference in more detail. While this scenario is simple, it demonstrates that the HUNTER-Rancor integration is functional.

Table 5. Comparison of HUNTER and Chosun timing results for the loss of feedwater scenario.

Study	**Count**	**Average**	**Standard Deviation**
Chosun Human Students	10	3:15	0:30
Chosun Human Operators	4	2:34	0:55
HUNTER Virtual Operators	500	5:02	1:34

5.3 Scenario 2: Startup

5.3.1 Description

The startup procedure is a normal operating procedure to transition Rancor from a cold shutdown state to producing electricity with the reactor at 100% power. The scenario is fairly complex as it involves

coordinating plant subsystems. The startup procedure for Rancor follows the same basic steps as a real PWR. The reactor is started by first establishing primary coolant flow and raising the control rods. The reactor is brought up to a low power level of around 10% in order to ramp the turbine. When the reactor is stable and at operating temperature the turbine can be latched and the governor valve can be raised to bring the turbine to its synchronization speed of 1800 RPM. Once the turbine is at synchronization speed the generator can be synced to the grid. At this point the plant is producing about 10% of its power capacity. The final phase of the plant evolution is to simultaneously raise reactor power and grid load while controlling reactor temperature and power to make sure it doesn't fall out of band and trigger a reactor trip. Real PWRs have more sophisticated control systems that maintain primary side temperatures and pressures, but coordination is still required between the primary and secondary sides during load changes. The scenario is considered successful if the plant is online and stable with the reactor at 100% power.

5.3.2 Implementation

Rancor had a startup procedure that was adapted for use for HUNTER. The procedures for HUNTER need to be much more explicit than traditional procedures so that the virtual operator can complete the steps without having to have internal knowledge of the plant and operational controls. The startup procedure was also adapted to the procedure schema described in Section 4.2. The startup procedure was authored using the HUNTERweb procedure authoring interface, and Appendix B contains a rendered version of the procedure. The procedures for HUNTER were serialized as JSON and deserialized as Python class instances.

A notable shortcoming of the current HUNTER implementation is that the virtual operator cannot follow continuous actions. Continuous actions are a procedure mechanism that allows operators to asynchronously follow multiple procedure steps throughout a plant evolution. During the startup procedure for Rancor, human operators are tasked with making sure the reactor temperature does not get too high or too low while also completing the steps to bring the plant online.

HUNTER was configured to run 500 iterations of the startup scenario with starting time-on-shift between 0 and 12 hours. The time-on-shift affects the dynamic fatigue calculation.

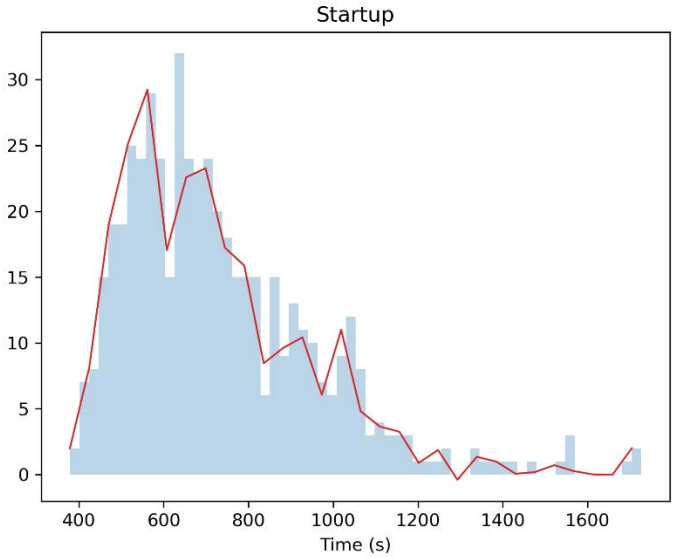


Figure 26. Distribution of times for HUNTER to complete startup.

5.3.3 Results

The virtual operator was able to complete the startup procedure on 404 of 500 iterations. The average completion time was 733 seconds (12 minutes 13 seconds). The distribution of startup times is shown in Figure 26.

Across the 500 iterations, the HUNTER virtual operator performed 8078 action attempts and had an error of commission rate of 0.001238 (i.e., failed 10 actions). The virtual operator checked 23,824 indicators with an error rate of 0.000839 (i.e., failed 20 checks).

5.3.4 Discussion

The virtual operator was able to successfully latch the turbine in 494 of 500 iterations and sync the turbine/generator in 498 of 500 iterations. However, the virtual operators were only able to bring the plant to 100% power in 404 of 500 iterations. During the ramping phase, actual operators manually monitor plant parameters and increase reactor power and load. The startup procedure for the virtual operator model implemented a strategy to accomplish ramping the plant but did so less successfully than real human operators due to slower response time for retrieving and taking actions. During this phase of the evolution the human reactor operators did not need to reference the procedures and took faster actions. The HUNTER model could be improved by adding additional GOMS-HRA primitives for modeling manual control that does not rely on formally referencing written procedures for each control decision and action.

Table 6. Comparison of HUNTER and Chosun timing results for the startup scenario.

Study	**Count**	**Average**	**Standard Deviation**
Human Students	19	9:40	3:08
Human Operators	9	12:06	6:47
HUNTER Virtual Operators	500	12:13	3:48

Nineteen Chosun University student participants performed the Rancor startup procedure and had comparatively more consistent and faster performance with an average time of 9 minutes and 40 seconds compared to the 12 minutes observed with the virtual operators (see Table 6). Nine operators completed the startup procedure and completed the scenario in 12 minutes and 6 seconds. The timing of the HUNTER model is consistent with the operators from the Chosun University study.

The Chosun University study found a task error rate of 0.009 and student operators and an error rate of 0.006 for licensed operators. The HEPs of the virtual operator were lower by an order of magnitude from the observed dataset. Future work is needed to improve the HEP modeling of HUNTER so HEPs are more representative of human operators.

This page intentionally left blank.

6. CONCLUSIONS

In this report, we demonstrated HUNTER can be synchronously coupled to Rancor to perform dynamic HRA Monte Carlo simulations. We modeled two scenarios and demonstrated that the virtual operator can complete task evolutions. This represents a significant and successful demonstration for coupling a virtual operator with a virtual plant model to support dynamic HRA. This approach shows promise, but it also demonstrates the need for further refinement in the modeling to better calibrate virtual operator to actual operator performance.

6.1 Lessons Learned from HUNTER-Rancor Coupling

The HUNTER-Rancor coupling demonstrated that HUNTER could be synchronously coupled to a process-based simulator. Rancor was an ideal model for this exercise because it was available as a Python model, meaning it could be used without an API and the two software codes could be combined as a single environment. The successful coupling brings HUNTER closer to its goal of being a standalone, single-click-to-launch software, independent of external codes for the environment module. The authors wrote the code for both HUNTER and Rancor, and therefore have extensive experience with both code bases. During development, we could freely alter the models as needed to assist with the HUNTER-Rancor integration. The ability to debug from HUNTER to Rancor source code greatly eased development of the complicated node-based procedure following logic. Because Rancor mimics the functionality of full-scope simulators, we are confident this coupling could be done with more complicated full-scope simulators. For the purposes of proof-of-concept demonstrations, HUNTER-Rancor provides an ideal, non-proprietary code base for developing and testing scenarios.

6.2 Lessons Learned from Demonstration Scenarios

The observed data show that students are faster than operators with the normal evolutions (i.e., the startup scenario), but with the abnormal evolution (i.e., the loss of feedwater scenario) the operators are faster than the students. This suggests that the operators have the ability to work slowly and cautiously or more expediently and quickly depending on plant circumstances. Our HUNTER timing closely matched the timing of the slow and cautious operators with the startup procedure, but was much slower completing the loss of feedwater scenario with rapid shutdown. This suggests that the HUNTER model cannot accurately model the expediency felt by the operators yet. For each element we need the ability to assign PSFs that could also alter the sampled time from the primitives to account for this discrepancy.

Human operators perform procedure guided control actions, but also engage in faster freestyle manual control actions. Our virtual operator was very capable of syncing the turbine to the grid, but struggled to ramp the reactor and generator to full power. This portion of the evolution requires visually monitoring of critical parameters to avoid tripping the reactor and/or turbine, and the virtual operator was not able to respond fast enough with the existing GOMS-HRA primitives implemented in a procedural fashion. HUNTER needs GOMS-HRA primitives for both types of human control mode. GOMS-HRA primitives are too slow to model faster freestyle manual control actions. Even when GOMS-HRA was developed, it was acknowledged that refinements to the primitives would likely be necessary (Boring and Rasmussen, 2016). The startup scenario provides illustrative evidence to suggest one such revision.

As noted, HUNTER does not currently support continuous action procedure steps. In human operators, continuous actions are not performed as part of periodic surveys of the plant or in a continuous monitoring fashion, depending on the type of action. Time-slicing activities will likely require a virtual buffer of actions that will be affected by the overall workload of the operator. Continuous actions may take a second priority to immediately required actions, for example, and HUNTER will need a way to reflect the availability of the operator to shift and prioritize between actions. Another form of continuous action is alarm monitoring. Given the anecdotally reported complexities of operators filtering multiple alarms to determine which alarms to prioritize, more empirical work will be required before we are able to develop a HUNTER multitasking model that supports alarm response.

6.3 Next Steps

We have found that the virtual operator is currently less error prone than observed humans for information retrieval and carrying out actions. Work is needed to implement a broader range of dynamic PSFs. Currently HEPs are influenced by fatigue from being on shift and from the GOMS-HRA time distributions defined for each task level primitive. With the current implementation, the HEP increases linearly as a function of fatigue(time_on_shift), and the primitive's time distribution (see Figure 27).

```
def evaluate_hra(self, primitive_id, time_on_shift):
    static_tmult = self.static_tmult
    time_on_shift_fatigue = self.time_on_shift_fatigue

    time_multiplier = static_tmult

    _fatigue = None
    if time_on_shift_fatigue:
        _fatigue = fatigue_index(time_on_shift)
        time_multiplier += _fatigue

    primitive = self.primitives[primitive_id]
    nominal_hep = primitive.nominal_hep
    adjusted_hep = nominal_hep * time_multiplier
    sampled_time = primitive.rng_sample_time()
    adjusted_time = sampled_time * time_multiplier
    hra_success = random.random() >= adjusted_hep
```

Figure 27. HUNTER code stub from HRA-Engine to model a GOMS-HRA primitive.

With this model longer times always yield higher HEPs. So, if the virtual operator is *rushing* it will always have probabilistically lower HEPs. A potential solution is to use the cumulative distribution function (CDF) of the primitives' time distribution to set HEP multipliers. For instance, extreme quantiles could have higher multipliers, and middle quantiles could have nominal multipliers.

The HUNTERweb tool has proven to be helpful during the development process to author and edit schemas. Future work should deploy this tool and make it available to non-developer users for testing and running of HUNTER. The HUNTERweb tool currently stands apart from the harmonized GUI of HUNTER 2, and it is planned to integrate HUNTERweb into the overall interface during the next development cycle.

While HUNTER is intended as an HRA tool, some of its main utility may be outside conventional risk analysis. HUNTER could, for example, be invaluable for optimizing procedures and validating their functionality. A challenge in the nuclear industry with control room modernization and with advanced reactor concepts of operation is the lack of operational experience. Yet, procedures must be written for novel human-machine interactions. With even simple simulator models it is possible to validate that the procedure accomplishes what it is intended to and to identify human error traps. The Monte Carlo approach adds resilience, as it is able to tease out edge conditions that would require fairly extensive operator testing and that procedure writers might not anticipate of during authoring. A planned application of HUNTER in the next fiscal year is procedure performance prediction.

7. REFERENCES

Ahn, J, Park, J., Boring, R., Ulrich, T.A., & Heo, Y. (2022). The HUNTER dynamic human reliability analysis tool: Graphical user interface. Proceedings of the 16th Probabilistic Safety Analysis and Management Conference, Paper JE124.

Boring, R.L. (2011). The use of simulators in human factors studies within the nuclear industry. In A.B. Skjerve and A. Bye (eds.), Simulator-Based Human Factors Studies Across 25 Years (pp. 3-17). London: Springer-Verlag.

Boring, R.L. (2015). A dynamic approach to modeling dependence between human failure events. Proceedings of the 2015 European Safety and Reliability (ESREL) Conference, pp. 2845-2851.

Boring, R.L. (2020). The first decade of the Human Systems Simulation Laboratory: A brief history of human factors research in support of nuclear power plants. Advances in Artificial Intelligence, Software and Systems Engineering, 1213, 528-535.

Boring, R., Lew, R., & Ulrich, T. (2017). Advanced nuclear interface modeling environment (ANIME): A tool for developing human-computer interfaces for experimental process control systems. Lecture Notes in Computer Science, 10293, 3-15.

Boring, R., Mandelli, D., Rasmussen, M., Herberger, S., Ulrich, T., Groth, K., & Smith, C. (2016). Integration of Human Reliability Analysis Models into the Simulation-Based Framework for the Risk-Informed Safety Margin Characterization Toolkit, INL/EXT-16-39015. Idaho Falls: Idaho National Laboratory.

Boring, R.L., & Rasmussen, M. (2016). GOMS-HRA: A method for treating subtasks in dynamic Human Reliability Analysis. Risk, Reliability and Safety: Innovating Theory and Practice, Proceedings of the European Safety and Reliability Conference, pp. 956-963.

Boring, R., Rasmussen, M., Smith, C., Mandelli, D., & Ewing, S. (2017a). Dynamicizing the SPAR-H method: A simplified approach to computation-based human reliability analysis. Proceedings of the 2017 Probabilistic Safety Assessment (PSA) Conference, pp. 1024-1031.

Boring, R. L., Rasmussen, M., Ulrich, T., Ewing, S., & Mandelli, D. (2017b). Task and procedure level primitives for modeling human error. Advances in Intelligent Systems and Computing, 589, 30-40.

Boring, R., Ulrich, T., Ahn, J., Heo, Y., & Park, J. (2022). Software Implementation and Demonstration of the Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER), INL/RPT-22-66564. Idaho Falls: Idaho National Laboratory.

Boring, R., Ulrich, T., & Lew, R. (2018). Parts and wholes: Scenarios and simulators for human performance studies. Advances in Intelligent Systems and Computing, 778, 116-127.

Boring, R.L., Ulrich, T.A., Lew, R., Kovesdi, C., & Al Rashdan, A. (2019). A comparison of operator preference and performance for analog versus digital turbine control systems in control room modernization. Nuclear Technology, 205, 507-523.

Boring, R.L., Ulrich, T.A., & Rasmussen, M. (2018). Task level errors for human error prediction in GOMS-HRA. Proceedings of the European Safety and Reliability (ESREL) Conference, 433-439.

Bye, A., Lois, E., Dang, V.N., Parry, G., Forester, J., Massaiu, S., Boring, R., Braarud, P.Ø., Broberg, H., Julius, J., Männistö, I., & Nelson, P. (2011). International HRA Empirical Study—Phase 2 Report: Results from Comparing HRA Method Predictions to Simulator Data from SGTR Scenarios, NUREG/IA-0216, Vol. 2. Washington, DC: U.S. Nuclear Regulatory Commission.

Choi, Y.-J. (2020). Assessment of Verification and Validation Status—EMRALD and HUNTER, INL/EXT- 20-59904. Idaho Falls: Idaho National Laboratory.

Choi, Y.-J. (2021). Coupling of the Dynamic Human Reliability Assessment Capability with RELAP5-3D Thermal-Hydraulics Code. INL, INL/EXT-21-01497. Idaho Falls: Idaho National Laboratory.

Endsley, M. R. (1988). Situation awareness global assessment technique (SAGAT). In Proceedings of the IEEE 1988 National Aerospace and Electronics Conference, pp. 789-795.

Gertman, D., Blackman, H., Marble, J., Byers, J., & Smith, C. (2005). The SPAR-H Human Reliability Analysis Method, NUREG/CR-6883. Washington, DC: U.S. Nuclear Regulatory Commission.

Heo, Y., Ulrich, T.A., Boring, R.L., Park, J., & Ahn, J. (2022). The HUNTER dynamic human reliability analysis tool: Coupling an external plant code. Proceedings of the 16th Probabilistic Safety Analysis and Management Conference, Paper YY168.

Lawrence, S., Smith, C.L., Mandelli, D., & Boring, R.L. (2021). Improved economics and reliability while maintaining high levels of safety—Achievements and on-going R&D with RISA pathway. Proceedings of the 2021 Probabilistic Safety Assessment (PSA) Conference, pp. 360-369.

Lew, R., Boring, R.L., & Ulrich, T.A. (2014). A prototyping environment for research on human-machine interfaces in process control: Use of Microsoft WPF for microworld and distributed control system development. 7th International Symposium on Resilient Control Systems (ISRCS), pp. 1-6.

Lew, R., Ulrich, T.A., and Boring, R.L. (2017). Nuclear reactor crew evaluation of a computerized operator support system HMI for chemical and volume control system. Lecture Notes in Artificial Intelligence, 10285, 501-513.

Massaiu, S., Bye, A., Braarud, P. Ø., Broberg, H., Hildebrandt, M., Dang, V. N., ... & Forester, J. A. (2010). International HRA Empirical Study, Overall Methodology and HAMMLAB Results. In Simulator-based Human Factors Studies Across 25 Years (pp. 253-269). Springer, London.

Medema, H., Mohon, J., & Boring, R. (2021). Extracting human reliability findings from human factors studies in the Human Systems Simulation Laboratory. Proceedings of the 2021 Probabilistic Safety Assessment (PSA) Conference, pp. 98-107.

O'Hara, J., Higgins, J., & Fleger, S. (2012). Human Factors Engineering Program Review Model, NUREG-0711, Rev. 3. Washington, DC: U.S. Nuclear Regulatory Commission.

Park, J., Boring, R.L., & Kim, J. (2019). An identification of PSF lag and linger effects for dynamic human reliability analysis: Application of experimental data. IEEE Human-System Interface Conference, pp. 12-16.

Park, J., Boring, R., Ulrich, T., Ahn, J., & Heo, Y. (2022a). The HUNTER dynamic human reliability analysis tool: Development of a module for performance shaping factors. Proceedings of the 16th Probabilistic Safety Analysis and Management Conference, Paper JO69.

Park, J., Boring, R.L., Ulrich, T.A., Yahn, T., & Kim, J. (2022b). Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER) Demonstration: Part 1, Empirical Data Collection of Operational Scenarios, INL/RPT-22-69167. Idaho Falls: Idaho National Laboratory.

Rabiti, C., Alfonsi, A., Cogliati, J., Mandelli, D., Kinoshita, R., Sen, S., ,... & Chen, J. (2017). RAVEN User Manual. Idaho Falls: Idaho National Laboratory.

Ulrich, T. A. (2017). The Development and Evaluation of Attention and Situation Awareness Measures in Nuclear Process Control Using the Rancor Microworld Environment. Dissertation, University of Idaho.

Ulrich, T., Boring, R.L., Ahn, J., Heo, Y., Park, J., & Lew, R. (2022). The HUNTER dynamic human reliability analysis tool: Procedurally driven operator simulation. Proceedings of the 16th Probabilistic Safety Analysis and Management Conference, Paper TH196.

Ulrich, T., Boring, R., L., Ewing, S., & Rasmussen, M. (2017a). Operator timing of task level primitives for use in computation-based human reliability analysis. Advances in Intelligent Systems and Computing, 589, 41-49.

Ulrich, T. A., Lew, R., Mortenson, T., Medema, H., Boring, R. L., Werner, S. & Minard, N. (2021). A dual full-scope and reduced-scope microworld simulator approach to evaluate the human factors of a coupled hydrogen production concept of operations. Lecture Notes in Networks and Systems, 264, 179-186.

Ulrich, T. A., Lew, R., Werner, S., & Boring, R. L. (2017b). Rancor: A gamified microworld nuclear power plant simulation for engineering psychology research and process control applications. In Proceedings of the 61st Human Factors and Ergonomics Society Annual Meeting, pp. 398-402.

This page intentionally left blank.

APPENDIX A – RANCOR LOSS OF FEEDWATER PROCEDURES FOR HUNTER

The procedure was authored using the HUNTERweb procedure authoring interface and is rendered here to demonstrate the procedure flow and logic. Note that the procedure represents a sanitized and significantly simplified version of a loss of feedwater procedure and is not tied to an actual plant. The procedure is based on the Rancor Microworld Simulator simplified operating procedure.

eop-0002_loss_of_feedwater

Authors
    Roger Lew
Plant
    Rancor (JabbaPy)
Updated
    2022-10-05
Procedure Type
    OP

Purpose

This procedure provides actions to diagnose and mitigate a loss of feedwater.

Step 1. Verify Entry Conditions

Verify the following preconditions:

– Check the Following

ANY of the following conditions are met:

– FW Pump A Off (Rc)

FeedWaterPumpA, x == False

– FW Pump A Off (Rc)

FeedWaterPumpB, x == False

– FW A Flow Low (Rc)

FwALowFlow, x == True

– FW Pump B Off (Rc)

FwBLowFlow, x == True

If response was NOT obtained exit this procedure.

Step 2. Verify Feedwater Pumps are Running

Perform the following Actions:

SubStep 2.1. Verify FeedWater Pump A is Running

Verify the following preconditions:

– Check if FeedWater Pump A is already running (Rc , Dp)

FeedWaterPumpA, x == True

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Turn on FeedWater Pump A (Rc)

FeedWaterPumpA <= False

Verify the following postconditions:

– Verify Feedwater Pump A is Running (Rc)

FeedWaterPumpA, x == True

SubStep 2.2. Verify FeedWater Pump B is Running

Verify the following preconditions:

– Check if FeedWater Pump B is already running (Rc , Dp)

FeedWaterPumpB, x == True

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Turn on FeedWater Pump B (Rc)

FeedWaterPumpB <= False

Verify the following postconditions:

– Verify Feedwater Pump B is Running (Rc)

FeedWaterPumpA, x == True

SubStep 2.3. Check if Feedwater Flow is normal

Verify the following preconditions:

– Check the Following

ANY of the following conditions are met:

– FW Pump A Off (Rc)

FeedWaterPumpA, x == False

– FW Pump A Off (Rc)

FeedWaterPumpB, x == False

– FW A Flow Low (Rc)

FwALowFlow, x == True

– FW Pump B Off (Rc)

FwBLowFlow, x == True

If response was NOT obtained exit this procedure.

Step 3. Verify Feedwater IVs are Open

Perform the following Actions:

SubStep 3.1. Verify FeedWater IV A is Open

Verify the following preconditions:

– Check if FeedWater IV A is already Open (Rc , Dp)

FWIVA, x == 1

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Turn on FeedWater IV A (Rc)

FWIVA <= 1

Verify the following postconditions:

– Verify Feedwater IV A is Open (Rc)

FWIVA, x == 1

SubStep 3.2. Verify FeedWater IV B is Open

Verify the following preconditions:

– Check if FeedWater IV B is already Open (Rc , Dp)

FWIVB, x == 1

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Turn on FeedWater IV B (Rc)

FWIVB <= 1

Verify the following postconditions:

– Verify Feedwater IV B is Open (Rc)

FWIVA, x == 1

SubStep 3.3. Check if FeedWater Flow is Normall

Verify the following preconditions:

– Check the Following

ANY of the following conditions are met:

– FW Pump A Off (Rc)

FeedWaterPumpA, x == False

– FW Pump A Off (Rc)

FeedWaterPumpB, x == False

– FW A Flow Low (Rc)

FwALowFlow, x == True

– FW Pump B Off (Rc)

FwBLowFlow, x == True

If response was NOT obtained exit this procedure.

Step 4. Rapid Shutdown

Perform the following Actions:

SubStep 4.1. Verify FeedWater IV A is Open

Verify the following preconditions:

– Check if FeedWater IV A is already Open (Rc , Dp)

FWIVA, x == 1

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Turn on FeedWater IV A (Rc)

FWIVA <= 1

Verify the following postconditions:

– Verify Feedwater IV A is Open (Rc)

FWIVA, x == 1

SubStep 4.2. Verify FeedWater IV B is Open

Verify the following preconditions:

– Check if FeedWater IV B is already Open (Rc , Dp)

FWIVB, x == 1

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Turn on FeedWater IV B (Rc)

FWIVB <= 1

Verify the following postconditions:

– Verify Feedwater IV B is Open (Rc)

FWIVA, x == 1

If response was obtained Go to *aop-0001_rapid_shutdown — step-1*

If response was NOT obtained Go to *aop-0001_rapid_shutdown — step-1*

aop-0001_rapid_shutdown

Authors
: Roger Lew

Plant
: JabbaPy

Updated
: 2022-10-31

Procedure Type
: OP

Purpose

This procedure shuts down the plant in an expedient manner

Step 1. Verify that one of the steam generator (SG) levels is normal

Verify the following preconditions:

–

ANY of the following conditions are met:

– SGLevelA is normal (Rc)

SGLevelA, 0.4 <= x <= 0.6

– SGLevelB is normal (Rc)

SGLevelB, 0.4 <= x <= 0.6

Step 2.

Perform the following Actions:

– Adjust the Bypass Valve to 10% (Ac)

BypassValveDemand <= 0.1

Step 3. Manually Trip Turbine

Verify the following preconditions:

– Turbine not tripped (Rc)

TurbineTrip, x != True

If response was NOT obtained go to parent's next (sub)step.

Perform the following Actions:

– Trip Turbine (Ac)

ScramTurbine()

Step 4. Manually Trip Reactor

Verify the following preconditions:

– Reactor not tripped (Rc)

AllRodsDown, x != True

If response was NOT obtained go to parent's next (sub)step.

Perform the following Actions:

– Trip Reactor (Ac)

ScramReactor()

Step 5. Manually Activate Safety Injection

Perform the following Actions:

– (Ac)

ManualSafetyInjectionActive <= True

Step 6.

Perform the following Actions:

– Close PORV 1 IV (Ac)

PorvDump1Stuck <= 0

– Close PORV 2 IV (Ac)

PorvDump2Stuck <= 0

– Close PORV 3 IV (Ac)

PorvDump3Stuck <= 0

– Close PORV 4 IV (Ac)

PorvDump4Stuck <= 0

Step 7.

Perform the following Actions:

SubStep 7.1. Open Bypass Valve

Perform the following Actions:

– (Ac)

BypassValveDemand <= 1

SubStep 7.2. Wait for Core Temp to Stabilize

Verify the following postconditions:

– (Rc)

RXvesselTemperature, x > 250

If response was obtained go back to the start of this (sub)step.

SubStep 7.3. Close Bypass Valve

Perform the following Actions:

–

BypassValveDemand <= 0

If response was obtained exit this procedure.

If response was NOT obtained exit this procedure.

This page intentionally left blank.

APPENDIX B – RANCOR STARTUP PROCEDURE FOR HUNTER

The procedure was authored using the HUNTERweb procedure authoring interface and is rendered here to demonstrate the procedure flow and logic. Note that the procedure represents a sanitized and significantly simplified version of a plant startup procedure and is not tied to an actual plant. The procedure is based on the Rancor Microworld Simulator simplified operating procedure.

op-0001_startup

Authors
Plant
    Rancor Jabba
Updated
    2022-10-03
Procedure Type
    OP

Purpose

This procedure describes how to start up and operate the Rancor Nuclear Power Plant in Auto mode. It assumes the Reactor is in shut down state, and the turbine is offline.

Step 1. Verify the Reactor is Shutdown

Verify the following preconditions:

– Verify the ALL RODS DOWN annuciator is lit (Rc)

AllRodsDown, x == True

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Trip the Reactor (Ac)

ScramReactor()

Verify the following postconditions:

– Verify the ALL RODS DOWN annuciator is lit (Rc)

AllRodsDown, x == True

Step 2. Verify the Turbine is Shutdown

Verify the following preconditions:

– Verify the TURBINE TRIP Annunciator Light is ON (Rc)

TurbineTrip, x == True

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Trip the Turbine (Ac)

ScramTurbine()

Verify the following postconditions:

– Verify the TURBINE TRIP Annunciator Light is ON (Rc)

TurbineTrip, x == True

Step 3. Start Reactor Coolant Pumps

Perform the following Actions:

SubStep 3.1. Start Reactor Coolant Pump A

Verify the following preconditions:

– Check if Reactor Coolant Pump A is already running (Rc)

RecircPumpA, x == True

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Start Reactor Coolant Pump A (Ac)

RecircPumpA <= True

Verify the following postconditions:

– Verify Reactor Coolant Pump A is running (Rc)

RecircPumpA, x == True

SubStep 3.2. Start Reactor Coolant Pump B

Verify the following preconditions:

– Check if Reactor Coolant Pump B is already running (Rc)

RecircPumpB, x == True

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Start Reactor Coolant Pump A (Ac)

RecircPumpB <= True

Verify the following postconditions:

– Verify Reactor Coolant Pump A is running (Rc)

RecircPumpA, x == True

SubStep 3.3. Verify primary coolant flow level

Verify the following postconditions:

– The LOW PRIMARY COOLANT Annunciator Light is off (Rc)

LowPrimaryCoolantFlow, x == False

Step 4. Bring the Reactor Online

Perform the following Actions:

SubStep 4.1. Set Rod Control to Auto Mode

Verify the following preconditions:

– Check if Rod Control is in AUTO (Rc)

RodCtrlAutoMode, x == True

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Set Rod Control To Auto (Rc)

SetRodCtrlAuto()

SubStep 4.2. Set Rod Control Target

Perform the following Actions:

– Set RX Target To 8% (Ac)

RodCtrlTargetRX <= 8

SubStep 4.3. Place Rod Control in Go

Perform the following Actions:

– Press the Rod Control Go Button (Ac)

SetRodCtrlGo()

Step 5. Start Feedwater Pumps

Perform the following Actions:

SubStep 5.1. Start Feedwater Pump A

Verify the following preconditions:

– Check if Feedwater Pump A is already running (Rc)

FeedWaterPumpA, x == True

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Start Feedwater Pump A (Ac)

FeedWaterPumpA <= True

Verify the following postconditions:

– Verify Feedwater Pump A is running (Rc)

FeedWaterPumpA, x == True

SubStep 5.2. Start Feedwater Pump B

Verify the following preconditions:

– Check if Feedwater Pump B is already running (Rc)

FeedWaterPumpB, x == True

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Start Feedwater Pump A (Ac)

FeedWaterPumpB <= True

Verify the following postconditions:

– Verify Feedwater Pump A is running (Rc)

FeedWaterPumpA, x == True

Step 6. Open Feedwater Isolation Valves

Perform the following Actions:

SubStep 6.1. Open FW IV A

Verify the following preconditions:

– Check if FWIV A is already Open (Rc)

FWIVA, x == 1

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Open FWIV A (Ac)

FWIVA <= True

Verify the following postconditions:

– Verify FWIVA is Open (Rc)

FWIVA , x == 1

SubStep 6.2. Open FW IV B

Verify the following preconditions:

– Check if FWIVB is already Open (Rc)

FWIVB, x == 1

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Open FWIVB (Ac)

FWIVB <= True

Verify the following postconditions:

– Verify FWIVB is Open (Rc)

FWIVB, x == 1

Step 7. Open Main Steam Isolation Valves

Perform the following Actions:

SubStep 7.1. Open MS IV A

Verify the following preconditions:

– Check if MSIV A is already Open (Rc)

MSIVA, x == 1

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Open MSIV A (Ac)

MSIVA <= True

Verify the following postconditions:

– Verify MSIVA is Open (Rc)

MSIVA , x == 1

SubStep 7.2. Open MS IV B

Verify the following preconditions:

– Check if MSIVB is already Open (Rc)

MSIVB, x == 1

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Open MSIVB (Ac)

MSIVB <= True

Verify the following postconditions:

– Verify MSIVB is Open (Rc)

MSIVB, x == 1

Step 8. Place SG Level Controllers in Auto

Perform the following Actions:

SubStep 8.1. Place SG Level Controller A in Auto

Verify the following preconditions:

– Check if SG Level Controller A is already in Auto (Rc)

SGACtrlAuto, x == True

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Place SG Level Controller A in Auto (Ac)

SetSGACtrlAuto()

SubStep 8.2. Place SG Level Controller A in Auto

Verify the following preconditions:

– Check if SG Level Controller B is already in Auto (Rc)

SGBCtrlAuto, x == True

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Place SG Level Controller B in Auto (Ac)

SetSGBCtrlAuto()

Step 9. Latch Turbine

Verify the following preconditions:

– Check if Reactor Scramed (Rc)

AllRodsDown, x == True

If response was obtained exit this procedure.

– Verify The Turbine is Ready to Latch

ALL of the following conditions are met:

– Verify Core Temperature is greater than 400 DEG F (Rc)

CoreLowTemp, x == False

– Verify the Turbine is not Latched (Rc)

Latched, x == False

– Reactor is not Scramed (Rc)

AllRodsDown, x == False

If response was NOT obtained Go to *op-0001_startup — step-9*

Perform the following Actions:

– Latch the Turbine (Ac)

LatchTurbine()

Verify the following postconditions:

– Verify the NOT LATCHED Annuciator is Off (Rc)

NotLatched, x == False

Step 10. Ramp-up Turbine to 1800 RPM

Perform the following Actions:

– Set Governor Valve Position Demand to 100% (Ac)

GovernorValveDemand <= 1

Verify the following postconditions:

– Verify Turbine Speed is Increasing (Rc)

dTurbineSpeed, x > 0

Step 11. Sync Generator to Grid

Verify the following preconditions:

– Ready to Sync Conditions are met

ALL of the following conditions are met:

– Turbine is at 1800 RPM (Rc)

TurbineSpeed, 1799 < x < 1804

– Reactor is Online (Rc)

AllRodsDown, x == False

– Primary Flow is Normal (Rc)

LowPrimaryCoolantFlow, x == False

If response was NOT obtained go back to the start of this (sub)step.

Perform the following Actions:

– Sync Turbine

SyncTurbine()

Step 12. Increase Load

Perform the following Actions:

SubStep 12.1. Make sure Core Temperature is below 650 F

Verify the following preconditions:

– Is Core Temperature less than 650 F? (Rc)

RXvesselTemperature, x < 650

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Increase Load by 5% (Ac)

IncreaseLoad()

Verify the following postconditions:

– Wait for Core Temperature to Decrease

ANY of the following conditions are met:

– Check Core Temp < 650F (1) (Rc)

RXvesselTemperature, x < 650

If response was obtained Go to *op-0001_startup — substep-12.2*

– Check Core Temp < 650F (2) (Rc)

RXvesselTemperature, x < 650

If response was obtained go to parent's next (sub)step.

– Check Core Temp < 650F (3) (Rc)

RXvesselTemperature, x < 650

If response was obtained go to parent's next (sub)step.

– Check Core Temp < 650F (4) (Rc)

RXvesselTemperature, x < 650

If response was obtained go to parent's next (sub)step.

If response was NOT obtained go back to the start of this (sub)step.

SubStep 12.2. Make sure Core Temperature is greater than 600 F

Verify the following preconditions:

– Is Core Temperature less than 600 F? (Rc)

RXvesselTemperature, x < 600

If response was obtained go to parent's next (sub)step.

Perform the following Actions:

– Increase RX Target by 5% (Ac)

IncreaseTargetRX()

Verify the following postconditions:

– Wait for Core Temperature to Decrease

ANY of the following conditions are met:

– Check Core Temp > 600F (1) (Rc)

RXvesselTemperature, x > 600

If response was obtained Go to *op-0001_startup — step-13*

– Check Core Temp > 600F (2) (Rc)

RXvesselTemperature, x > 600

If response was obtained Go to *op-0001_startup — step-13*

– Check Core Temp > 600F (3) (Rc)

RXvesselTemperature, x > 600

If response was obtained Go to *op-0001_startup — step-13*

– Check Core Temp > 600F (4) (Rc)

RXvesselTemperature, x > 600

If response was obtained Go to *op-0001_startup — step-13*

If response was NOT obtained Go to *op-0001_startup — substep-12.2*

Step 13. Increase Reactivity and Load

Perform the following Actions:

SubStep 13.1. Increase Load

Perform the following Actions:

– Set Target Load to 90%

ControlValveDemand <= .90

SubStep 13.2. Increase Target RX

Perform the following Actions:

– Increase RX Target to 100%

RodCtrlTargetRX <= 100

Step 14. Monitor Until Stable

Verify the following preconditions:

– Check if Reactor Scramed (Rc)

AllRodsDown, x == True

If response was obtained exit this procedure.

– Check if not Online (Rc)

ControlValve, x == 0

If response was obtained exit this procedure.

Verify the following postconditions:

– RX at 100% and Load at 90%

ALL of the following conditions are met:

– RX is at ~100% (Rc)

RX, x > 99.5

– Load at 90% (Rc)

ControlValve, 0.89 < x < 0.91

If response was obtained exit this procedure.

If response was NOT obtained Go to *op-0001_startup — step-14*