INL/RPT-24-80110

Light Water Reactor Sustainability Program

Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER 3.0) User Guide

September 2024

U.S. Department of Energy

Office of Nuclear Energy

INL/RPT-24-80110

Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER 3.0) User Guide

**Ronald Boring,[1] Roger Lew,[2] Thomas Ulrich,[1] Jooyoung Park,[1]
Jisuk Kim,[1] Olugbenga Gideon[2]**

**[1]Idaho National Laboratory
[2]University of Idaho**

September 2024

**Idaho National Laboratory
Idaho Falls, Idaho 83415**

http://www.lwrs.gov

**Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Light Water Reactor Sustainability Program**

This page intentionally left blank.

SUMMARY

Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER) is a software system to simulate human performance in support of human reliability analysis in nuclear power plants. HUNTER was developed at Idaho National Laboratory (INL) in support of the U.S. Department of Energy's Light Water Reactor Sustainability (LWRS) Program. This report summarizes recent work to integrate HUNTER with a plant simulator, namely the Rancor Microworld Simulator. Rancor is an offshoot of earlier LWRS work at INL to support plant modernization. The graphical software tools used to mimic digital human-machine interface upgrades at INL's Human Systems Simulation Laboratory were linked to an INL-developed simplified simulator to allow more ready human performance data collection than was possible with full-scope training simulators for nuclear power plants. The purpose of coupling HUNTER to Rancor is to build on the ease of use and customizability of the Rancor software platform, thereby allowing widespread release of the HUNTER software. Previous HUNTER software implementations have been well suited for research applications, but the complex software dependencies made transfer of the HUNTER software to end users difficult. By bundling the HUNTER software libraries with Rancor, HUNTER gains a software platform that may be distributed as a standalone executable application to users under license. Additionally, because Rancor has been used extensively in human performance studies, the scenarios and operational data can inform and validate HUNTER analyses.

The purpose of this report is to provide a user guide to aid human reliability analysts and researchers in simulating human performance in Rancor-HUNTER. The report includes brief histories of HUNTER and Rancor and a thorough discussion of the integrated Rancor-HUNTER software. An example of a Rancor startup scenario is provided to demonstrate virtual human-in-the-loop data generation with Rancor-HUNTER.

It is planned that future versions of HUNTER will couple to full-scope training simulators at plants. The current version of Rancor-HUNTER in this user guide provides a mature proof of concept and platform for identification of key software features necessary for coupling HUNTER to such simulators. The goal is to make HUNTER's advanced human simulation toolset more easily accessible to support commercial plant uses in probabilistic risk assessment.

This page intentionally left blank.

CONTENTS

FIGURES

This page intentionally left blank.

TABLES

This page intentionally left blank.

ACRONYMS

ANIME	Advanced Nuclear Interface Modeling Envrionment
AOP	Abnormal Operating Procedure
CBP	computer-based procedure
CNS	Compact Nuclear Simulator
CONOPS	concept of operations
COPS	computerized operating procedure system
CSV	comma-separated value
Ctrl	controller
DOE	Department of Energy
DPS	Digital Procedure System
dRX3	reactivity change rate in reactor core
EMRALD	Event Modeling Risk Assessment using Linked Diagrams
EOC	error of commission
EOO	error of omission
EOP	Emergency Operating Procedure
FATR	Failsafe Timed Automated Response
FPOG	Flexible Plant Operations and Generation
GOMS	Goals, Operators, Methods, and Selection
GUI	graphical user interface
HCR	Human Cognitive Reliability
HEP	human error probability
HFE	human failure event
HITL	human in the loop
HMI	human-machine interface
HRA	human reliability analysis
HSI	human-system interface
HSSL	Human Systems Simulation Laboratory
HTSE	high temperature steam electrolysis
HUNTER	Human Unimodel for Nuclear Technology to Enhance Reliability
ID	identifier
INL	Idaho National Laboratory
JSON	JavaScript Object Notation
K-HRA	Korean Standard Human Reliability Analysis

LOFW	loss of feedwater
LWRS	Light Water Reactor Sustainability
NPP	nuclear power plant
NRC	Nuclear Regulatory Commission
NUREG	Nuclear Regulatory document
P3	Procedure Performance Predictor
P&ID	piping and instrumentation diagram
PNG	Portable Network Graphic
PORV	pilot operated relief valve
PRA	probabilistic risk assessment
PSF	performance shaping factor
PWR	pressurized water reactor
R&D	research and development
Rancor	Rankine Core Microworld Simulator
RAVEN	Risk Analysis is a Virtual ENvironment
RISA	Risk-Informed Systems Analysis
RNO	Response Not Obtained
RO	Response Obtained
SAPHIRE	Systems Analysis Programs of Hands-on Integrated Reliability Evaluations
SG	steam generator
SGTR	steam generator tube rupture
SHEEP	Simplified Human Error Experimental Program
SHERPA	Systematic Human Error Reduction and Prediction Approach
SI	Système International d'unités (International Standard Unit)
SI	safety injection
SMR	small modular reactor
SPAR-H	Standardized Plant Analysis Risk-Human
SVG	Scalable Vector Graphics
THERP	Technique for Human Error Rate Prediction
TPD	Thermal Power Dispatch
U.S.	United States
VHITL	virtual human in the loop
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language
YAML	Yet Another Markup Language

Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER 3.0) User Guide

1 INTRODUCTION

Human reliability analysis (HRA) has historically been a worksheet-based approach (or software version of a worksheet) to predict the different types of human error and quantify those errors. The Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER) represents a different approach to HRA through a software tool to simulate operator performance in nuclear power plants. Now in its third version, HUNTER creates a virtual operator that interfaces with a nuclear power plant simulator. HUNTER is essentially an automation system for the simulator, except it can capture imperfect performance. Whereas classic automation strives for ideal performance, a tool designed to mimic human performance will necessarily capture degradations in human performance. HUNTER simulates the stochastic nature of human performance to reflect realistic operational outcomes. By introducing deleterious effects on performance, it is possible to simulate those contexts that are error prone, allowing HUNTER to screen out contexts that are risk important. Additionally, by repeating tasks across a range of performance, HUNTER can provide a probability of human error for complex scenarios. The key value of HUNTER is not for well understood human scenarios for which existing HRA approaches adequately capture the human errors. Rather, HUNTER is a tool that helps:

- Model complex scenarios that are not easily captured in HRA worksheet approaches
- Model what-if scenarios for which the consequences of human performance are not well understood
- Model novel use contexts such as the introduction of new technologies in the control room and corresponding new procedures
- Model aspect of human performance beyond human error such as time duration.

This document serves as the user guide for the current Version 3 implementation of HUNTER. It also represents the first planned widescale release of HUNTER as a standalone software package that may be used by human reliability analysts such as found at utilities and by human factors researchers who explore the varieties of human performance found in different operating contexts at nuclear power plants.

As described in Chapter 2, HUNTER has evolved across multiple versions to its current executable software form. Currently, it is coupled to a simplified nuclear power plant simulator, called the Rancor Microworld Simulator, which is described in Chapter 3. The details of Rancor-HUNTER are described in Chapter 4, while a sample analysis is demonstrated in Chapter 5. This user guide concludes with a discussion of other applications and future implementation plans for HUNTER.

As a user guide, this document strives to balance background information necessary to understand and use Rancor with a screen-by-screen tutorial of how to use HUNTER. It is anticipated that future versions of the HUNTER software will add to this user guide, making it a living document.

This page intentionally left blank.

2 HUNTER

2.1 Introduction[2]

HUNTER (Boring et al., 2016 and 2022) is a computational HRA tool to dynamically model human cognition and actions as well as incorporate the outputs into probabilistic risk assessment (PRA) models. As many researchers have emphasized the importance of simulation and human performance modeling in HRA, the HUNTER framework has been developed by the support of the Risk-Informed Systems Analysis (RISA) Pathway within the United States (U.S.) Department of Energy (DOE) Light Water Reactor Sustainability (LWRS) program. HUNTER continues to address challenges within existing static HRA to more realistically and accurately evaluate human-induced risks in nuclear power plants (NPPs). In this section, we introduce: (1) why we need to research dynamic HRA and develop a dynamic HRA tool (i.e., the HUNTER tool), (2) how the HUNTER tool has been historically developed, and (3) future directions of dynamic HRA.

2.2 Need for HUNTER

Most HRA methods currently used by regulatory institutes or utilities are called static HRA, which is carried out by simple worksheets or software equivalents. Static HRA reviews a particular snapshot of possible outcomes, but it does not model a dynamic event progression. The changing event progression—the defining characteristic of dynamic HRAs and PRAs—allows the modeling of the range of activities and outcomes as well as the consideration of a variety of what-if scenarios that would prove onerous to perform manually with static methods. Dynamic HRA can also be used to model scenarios for which there is minimal operational experience to explore what outcomes may emerge as a result of different human responses. This capability is especially useful for emerging areas of interest in risk modeling, such as severe accidents, HRA for human interactions with advanced technologies like digital and automated human-system interfaces (HSIs), balance-of-plant activities beyond the main control room that is the main focus of conventional HRA methods, and specialized areas like mobile equipment use and physical security modeling. As work on developing sample analyses in HUNTER continues, it is important to demonstrate the additional risk insights afforded by dynamic modeling that would not be possible with conventional static methods. An easy-to-use software tool that can help bring new risk insights is essential for industry as it supports new risk requirements.

Development of dynamic HRA tool can be used beyond simply producing a quantitative output in the form of a human error probability (HEP). Dynamic HRA can provide qualitative insights into the types of activities plant personnel will perform in novel contexts. For example, it might reveal that certain courses of action elicit a large workload in plant personnel, suggesting the need for alternate, less mentally demanding pathways to ensure positive outcomes. Dynamic HRA can also provide other quantitative measures like time-on-task estimates that aren't readily available in existing static methods. Additional illustrations of the unique uses of HUNTER's dynamic modeling will be explored in future research and development (R&D) activities.

2.3 History of HUNTER

The study of dynamic HRA at Idaho National Laboratory (INL) was began in earnest in 2014 (Boring et al., 2014). Over the course of a few years, it grew to be called HUNTER, and its framework and methodology have been matured, as INL developed various functions and methods needed for dynamic HRA and then assembled them in HUNTER.

Figure 1 shows the representative LWRS reports to date, which include how we have scaffolded out the HUNTER framework for dynamic HRA and integrated diverse tools into a common software platform. In addition, Appendix A lists relevant publications related to HUNTER.

[2] Note that some portions of this chapter are excerpted from previously published reports as referenced at the end of each section.

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

Error data collection of human operators

(Park et al., 2022)

New scenarios and simulator coupling

(Lew et al., 2022)

EMRALD-HUNTER

Coupling HUNTER with EMRALD PRA tool

(Lew et al., 2023)

HUNTER 2.2

HUNTER Procedure Performance Predictor (P3)
and time distribution data collection of human operators

(Park et al., 2024)

Figure 1. Evolution of HUNTER to date

2.3.1 HUNTER 1.0: Initial Conceptual Version

Under the HUNTER 1.0 research activities, the initial concept of HUNTER (see Figure 2) was proposed as a framework that combines a variety of methods and tools required for dynamic HRA. These efforts consist of:

- Dynamic Standardized Plant Analysis Risk-HRA (SPAR-H) (Boring et al, 2017; Gertman et al., 2005),
- Dynamic dependency (Boring, 2015)
- Goals, Operators, Methods, and Selection rules (GOMS) – HRA (GOMS-HRA) (Boring & Rasmussen, 2016)
- Risk Analysis in a Virtual ENvironment (RAVEN)-HUNTER (Boring, et al., 2016).

The details on each effort are briefly described in the following sections.

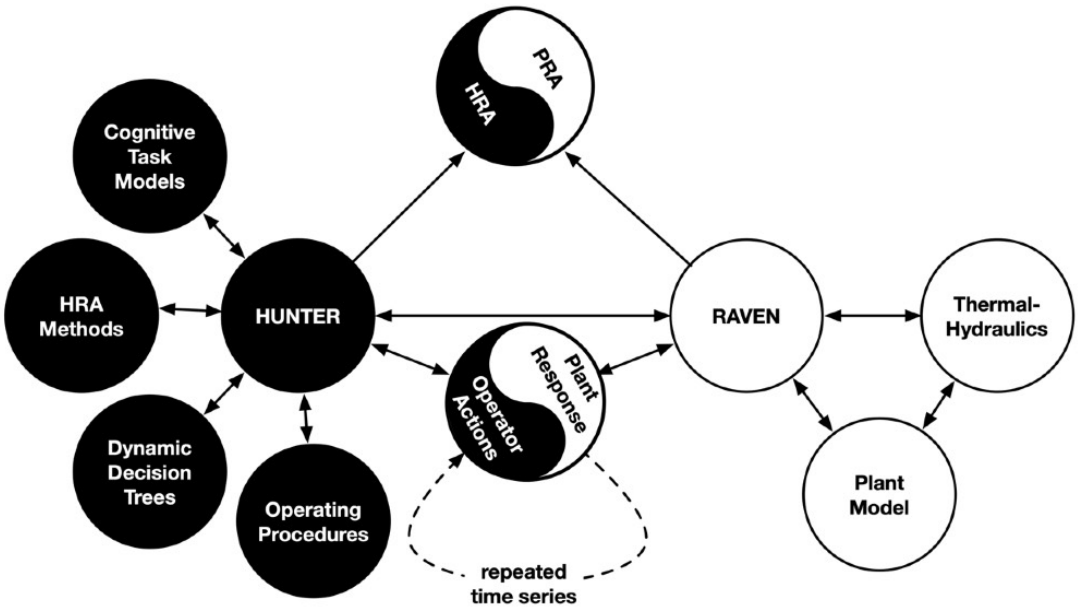


Figure 2. The original HUNTER framework from Boring et al. (2016)

2.3.1.1 Dynamic SPAR-H

In HUNTER 1.0 development, how to adapt the existing static SPAR-H method for dynamic HRA was researched. The SPAR-H method (Gertman et al., 2005) is an easy-to-use HRA method developed by INL and endorsed by the U.S. Nuclear Regulatory Commission (U.S. NRC). It has been widely used by both industry and regulators in its intended area of use as well as in other industries (Rasmussen et al., 2015). In traditional HRA approaches including the existing SPAR-H method, human actions are manually analyzed by human reliability analysts using worksheets. Specifically, for the HEP calculation, the analysts need to allocate a nominal HEP (i.e., a default error rate that serves as the starting value for HRA quantification) for a human failure event (HFE) or a smaller task-unit, rate performance shaping factors (PSFs) (i.e., any factors that influence human performance such as stress or experience) representing contextual impacts, and then modify the nominal HEP by multiplying the multiplier values for PSFs. In contrast, dynamic HRA is required to automatically allocate a nominal HEP, evaluate PSFs,

and then quantify HEPs based on the given information without HRA analysts. In HUNTER-adjacent research at this time, GOMS-HRA was used for determining nominal HEPs, while autocalculating the PSF level and quantifying the HEP were studied based on the SPAR-H method. The details on the dynamic SPAR-H approach are well described in Boring et al. (2017).

2.3.1.2 Dynamic Dependency

Dependency analysis in HRA is a method of adjusting the failure probability of a given action by considering the impact of the action preceding it (Swain and Guttmann, 1983; Podofillini et al., 2010). It plays an important role in reasonably accounting for human actions in the context of PRAs and prevents PRA results from being estimated too optimistically based on the HRA results. The dependency analysis has been known to significantly affect the overall result of PRA by being a main driver on the error rate. If the result of dependency analysis is inadequate, it could be unconvincing for explaining the human failures in the context of PRA. In other words, the risk metrics such as core damage frequency can be significantly under-estimated in cut sets or sequences containing multiple human failure events if dependency is not properly considered.

Dynamic HRA facilitates modeling operator actions over time as well as straightforwardly analyzing dependencies between the actions. Existing static HRA methods generally do not consider human performance changes over time or the event progression, nor do they provide a truly dynamic account of human actions (Park et al., 2019). Accordingly, human reliability analysts have performed dependency analysis by only relying on static PRA and HRA information. Over-reliance on static snapshots of human performance in these analyses could underestimate risk. In dynamic HRAs, however, we consider human actions dynamically and model types of activities and events even where the human role is not clearly understood or predicted, i.e., unexampled events such as severe accidents. Furthermore, a dynamic simulation straightforwardly represents a sequence of operator actions, which optimizes identifying dependency candidates within contextual impacts. Boring (2015) conceptually suggested PSF lag and linger effects as an option to treat dependence between operator actions. PSF lag indicates that the effect of the PSF on performance does not immediately psychologically or physically appear, while PSF linger means that the influence of PSFs on previous operator actions is not finished after the actions, resulting in residual effects on the next operator actions. Park and colleagues (2019) and Park and Boring (2020) validated the effects on the basis of experimental data and applied the concept to the dynamic dependency analysis.

2.3.1.3 GOMS-HRA

The GOMS–HRA method (Boring et al., 2016; Boring & Rasmussen, 2016) was developed to provide cognition-based time and HEP information for the dynamic HRA calculation in the HUNTER framework. It is theoretically based on the GOMS method, which has been used to model proceduralized activities and evaluate user interactions with HSIs in human factors research. As a predictive method, GOMS-HRA is suited to simulate human actions under a specific circumstance in a scenario. The basic approach of GOMS-HRA consists of three steps: 1) breaking human actions into a series of task primitives, 2) allocating time and error values to each task primitive, then 3) predicting human actions or task durations.

In GOMS-HRA, human actions are broken into a couple of task primitives. The GOMS-HRA method uses six types of task level primitives based on Systematic Human Error Reduction and Prediction Approach (SHERPA) (Stanton et al., 2013).. The following are the GOMS task level primitives:

- *Actions* (A)—Performing required physical actions on the control boards (A_C) or in the field (A_F)
- *Checking* (C)—Looking for required information on the control boards (C_C) or in the field (C_F)
- *Retrieval* (R)—Obtaining required information on the control boards (R_C) or in the field (R_F)

- *Instruction Communication* (I)—Producing verbal or written instructions (I_P) or receiving verbal or written instructions (I_R)
- *Selection* (S)—Selecting or setting a value on the control boards (S_C) or in the field (S_F)
- *Decision* (D)—Making a decision based on procedures (D_P) or without available procedures (D_W)

The time and error values are allocated for task level primitives of human actions analyzed in the first step. The time information includes statistic distribution, mean, standard deviation, 5th percentile, and 95th percentile, which have been derived out from the time data collected through experiment using actual operators and Human Systems Simulation Laboratory at INL (Ulrich et al., 2017). For the HEP information, these are assumed based on data suggested in the Technique for Human Error Rate Prediction (THERP) (Swain & Guttmann, 1983) method. The task level primitives are mapped to procedure steps called procedure level primitives (Boring et al., 2017). Additionally, the task level primitives are mapped to error types called task level errors (Boring et al., 2018) to help predict the most common human errors.

2.3.1.4 RAVEN-HUNTER

The HUNTER framework was designed to interact with other dynamic risk analysis tools like the Risk Analysis Virtual Environment (RAVEN) framework (Rabiti et al., 2021). The first version of HUNTER included efforts to connect with the RAVEN framework (Boring et al., 2016). As the HRA counterpart to RAVEN, HUNTER was tested to quantify HEPs for operator actions in a station blackout scenario based on time-dependent plant response data and operator actions provided by RAVEN and provide the probabilities to RAVEN. The details on RAVEN-HUNTER are described in Boring et al. (2016).

2.3.2 HUNTER 2.0: Integrated Software

The HUNTER 2.0 development effort (Boring et al., 2022) concentrated on how to develop a standalone software based on the initial framework and demonstration of HUNTER concepts researched in HUNTER 1.0. To conduct dynamic HRA, HUNTER conceptually includes three modules and four classes as shown in Figure 3.

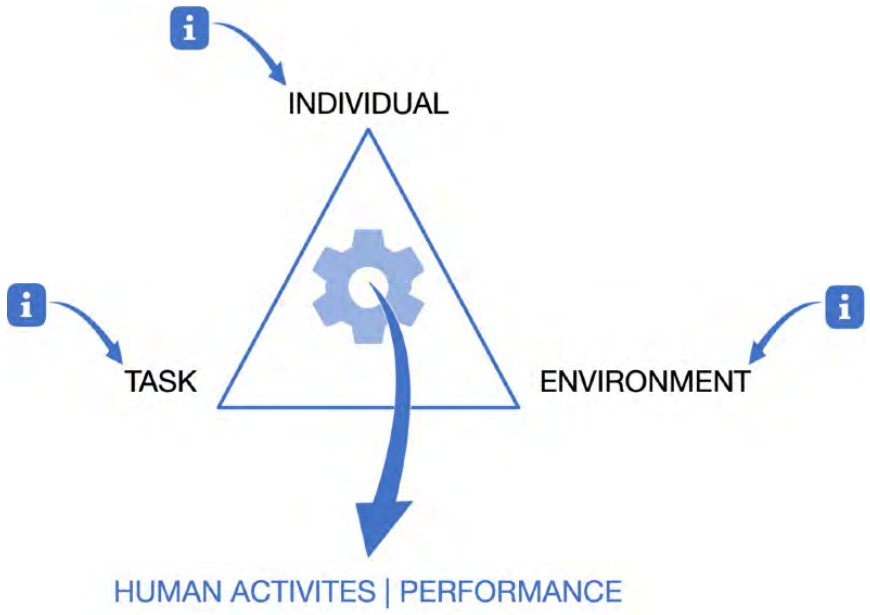


Figure 3. Conceptual modules (in black) and classes (in blue) of HUNTER 2 (Boring, et al., 2022).

The three modules, depicted as corner nodes in black text in Figure 3, are briefly noted at a conceptual level here. We use the example of a virtual control room operator model for illustration here, but HUNTER is not limited to only this representation of plant personnel.

- *Individual Module*—this is the representation of the human performing the activity, sometimes referred to as a "virtual operator." It can be also called as *PSF Module*. It incorporates relevant characteristics of the individual that impact that individual's performance. Such factors could be considered internal PSFs, which are the psychological considerations—like internal stress, experience, knowledge, and fitness for duty—that the individual brings to the task. These factors may contribute directly to error rates (e.g., stress causes poor decision-making) or indirectly (e.g., performance is slowed when fatigued). The individual module may, when so configured, include a cognitive model that accounts for crucial aspects of performance like decision-making.
- *Task Module*—this is the representation of what activity the human is performing. The human follows a course of action, whether guided by an operating procedure, a mental schema, or decision-making according to emergent stimuli and strategic goals. In the simplest form of the task module, the task is represented by a script that mirrors procedures. The task advances step by step, responding to a set of if-then queries to plant states. For example, if a high-priority alarm sounds, the script will direct a specific response by the virtual operator. In a simple model, the operator's ability to perform that task may be influenced by factors contained in the individual and environment modules, but the operator does not deviate from the script. Of course, actual reactor operators are not merely automata, and they will weigh in on the suitability of scripts and even improvise when appropriate. A richer model of the task would include provisions for skill of the craft and acting outside of rote script. An even more detailed model would incorporate tradeoffs and decision-making, including decision heuristics indicative of operator expertise.
- *Environment Module*—this is the representation of the world in which the human is acting. In this sense, the "world" consists of the systems and tools the human uses. It is the virtual world counterpart to the virtual operator represented in the individual module. For most NPP modeling, this world model corresponds to a plant simulation. The environment may often only encompass the immediate environment and not necessarily consider the broader environment such as the natural setting of the plant if that is not central to the task at hand.

There are four classes of the HUNTER framework illustrated in blue in Figure 3. They are:

- *Input Class*—the context is set by the scenario at hand. This is shown in Figure 3 as an input (i.e., ) into each of the modules, representing the influences that feed into the scenario. A preprocessor sets the context—the initial configuration for the individual, the task at hand, and the state of the plant—in which HUNTER operates.
- *Scheduler Class*—the glue that holds the other modules together. In the figure, this is signified by the lines of the triangle. It coordinates the interactions between different modules and also paces the progression of the event. Modules may complete their calculations at different rates, and the scheduler synchronizes the inputs, outputs, and interdependencies to a common time scale.
- *Processor Class*—the processing that occurs at each step of the task, which is depicted by a gear in the center of the Figure 3. A step occurs when all modules have completed their modeling refresh cycle and exchanged information. For example, the environment has advanced a time step, updating plant parameters, which have been perceived by the virtual operator (individual), who has responded by activating a virtual switch (task). This task may be driven by a procedure, which must meet certain requirements to advance. The processor

class determines the point of advancement to the next task. The processor may include logical assertions, such as actions predicated on conditions met, branching points, and operator decisions.

- *Output Class*—the results of each incremental step in the model. Outputs are changes in the state of the model, which are logged as activities, parameter states, and human performance logs. The output class records the actual outputs, such as the calculated HEPs that allow HUNTER to be used as an HRA method.

These classes may be considered the support functions behind driving the model execution. The classes are collectively referred to tongue-in-cheek as the "Gatherer" classes. The three HUNTER modules combine with the Gatherer classes to form the HUNTER-Gatherer underpinnings of the software.

Figure 4 shows the original HUNTER framework from Figure 2 superimposed with the more generic modules and classes from HUNTER 2. HUNTER 2 is both an extension and generalization of the original HUNTER framework. As can be seen, there is a direct mapping of some elements. The modules, as would be expected, are comprised of multiple sub-elements, while the classes link these modules functionally. The details on the software design are described in Boring et al. (2022). Figure 5 shows the HUNTER Version 2 software interface. The ways to interact with three HUNTER modules are represented in the interface.

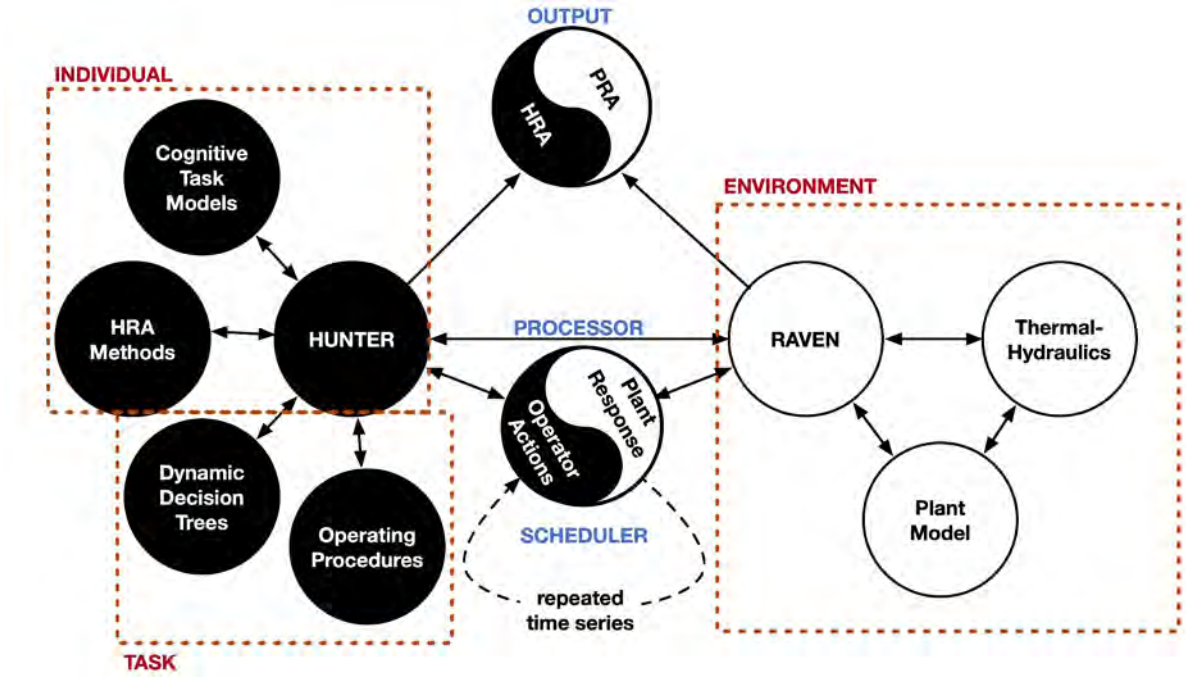


Figure 4. Crosswalk of HUNTER 1 to HUNTER 2 (Boring, et al., 2022).

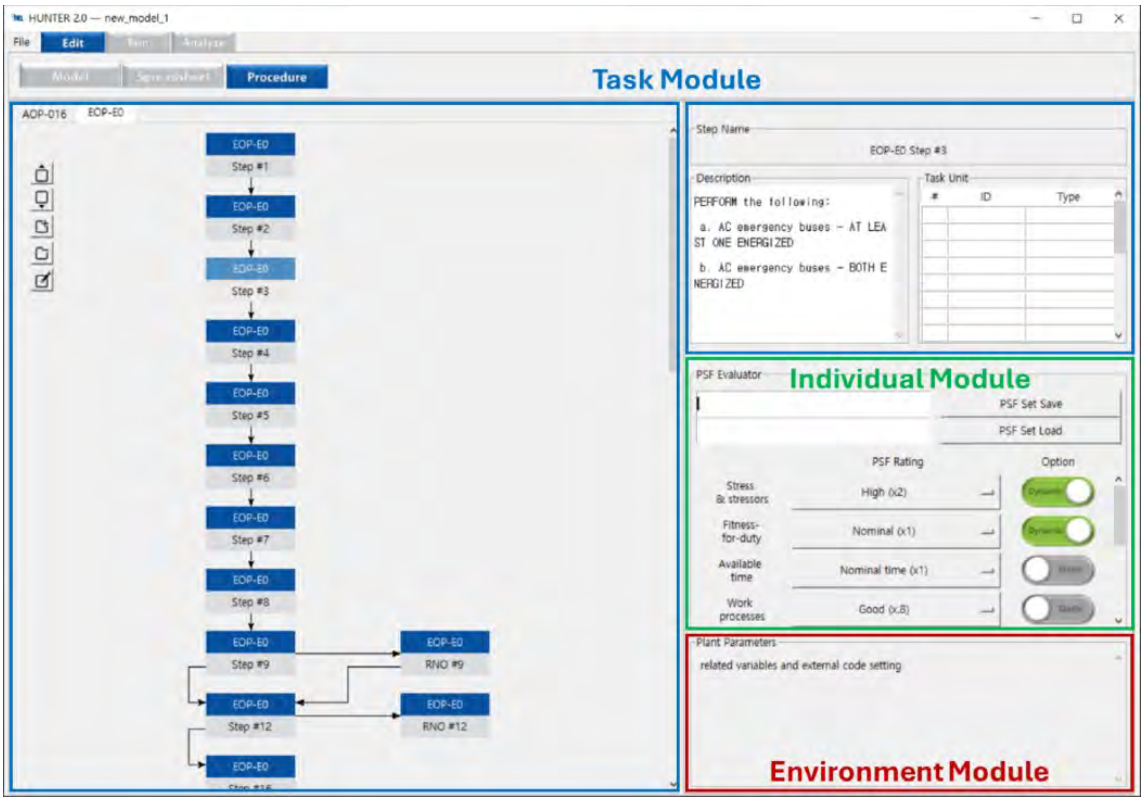


Figure 5. The HUNTER 2.0 interface

2.3.3 HUNTER 2.1: Test Runs and Validation

In the HUNTER 2.1 research (Lew et al., 2022), the HUNTER 2.0 software was coupled with a Python-based version of the Rancor Microworld Simulator (henceforth referred to simply as Rancor) to test and validate the HUNTER software. HUNTER 2.1 corresponds to the first informal version of Rancor-HUNTER, which shows new features beyond HUNTER 2.0 by coupling the Rancor simulator model with the HUNTER software. Rancor is a simplified nuclear power plant simulator developed by Ulrich et al. (2017). Basically, HUNTER is a virtual operator framework and models human actions based on procedures, which ask parameters or states from NPPs or simulators. Let's assume that there is a task that operators stop a pump. Although the action can be modeled via HUNTER, initiating the action depends on the status of the pump, which is an output from NPPs or simulators. In this research, Rancor plays a role in providing the information based on the simulation. The simulated information is used to evaluate the logic within each procedure step. Then, the virtual operator completion of each step is evaluated by HUNTER modules with calculating completion durations, HEPs, and success or failure.

In this effort, two of the ten scenarios from Park et al. (2022) were selected to test and validate the HUNTER software. The two selected scenarios are: (1) loss of feedwater and 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 HUNTER Monte Carlo simulation outputs four hierarchically organized comma-separated value (CSV) files for the task level primitive (i.e., the smallest task unit), element level (i.e., the task unit for step, condition, and action), procedure level (i.e., a collection of ordered steps), and other task level plant

parameter data. Figure 6 shows an example of HUNTER outputs from a startup scenario from the Monte Carlo simulation. 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.

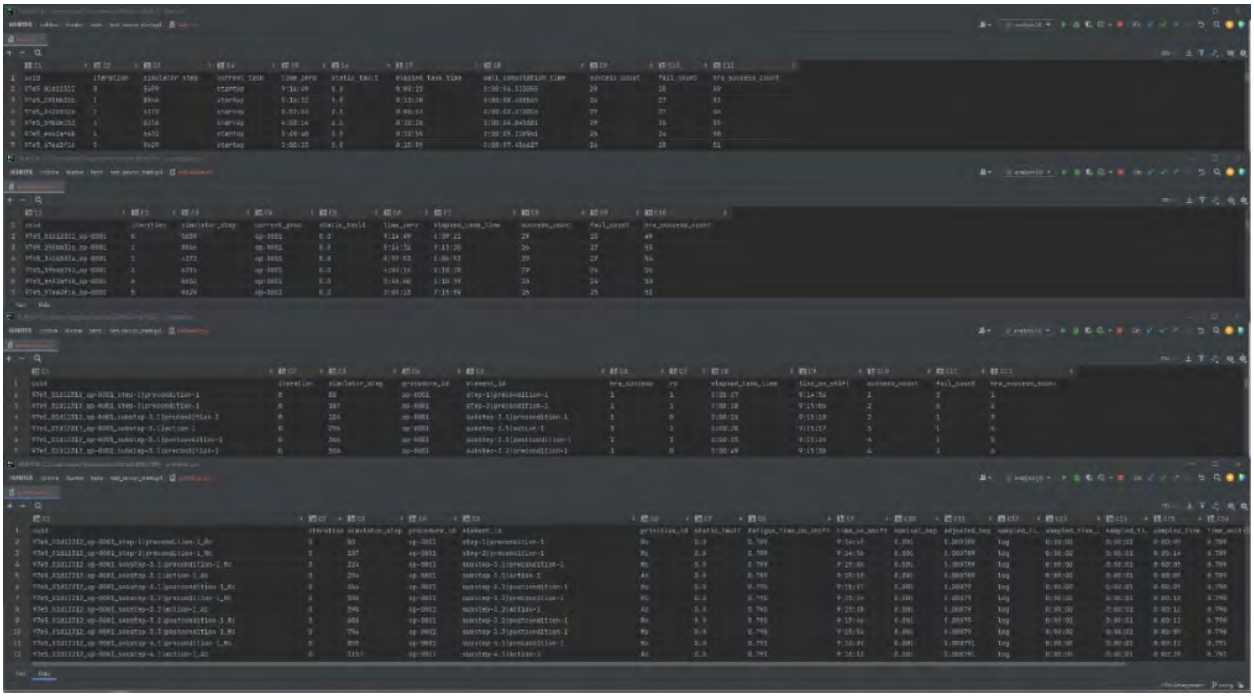


Figure 6. HUNTER outputs from a startup scenario based on the Monte Carlo simulation

As a result of test runs and validation for the two scenarios, Figure 7 shows the distribution of times for a virtual operator (i.e., the HUNTER model) to complete loss of feedwater. The virtual operator was able to complete procedures used in the scenario in all 500 simulated evolutions, as shown in the figure. The virtual operator had an average completion time of 302 seconds. Figure 8 shows time distribution of times for a virtual operator to complete startup. The virtual operator was able to complete all relevant procedures on 404 of 500 iterations. The average completion time was 733 seconds. Across the 500 iterations, the HUNTER virtual operator performed 8,078 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). This study also attempted to compare these results with data from a Rancor experimental study (Park et al., 2022) to discuss how many differences there are between human performance measures for the HUNTER virtual operator and human operators. The details on the comparison are described in Lew et al. (2022).

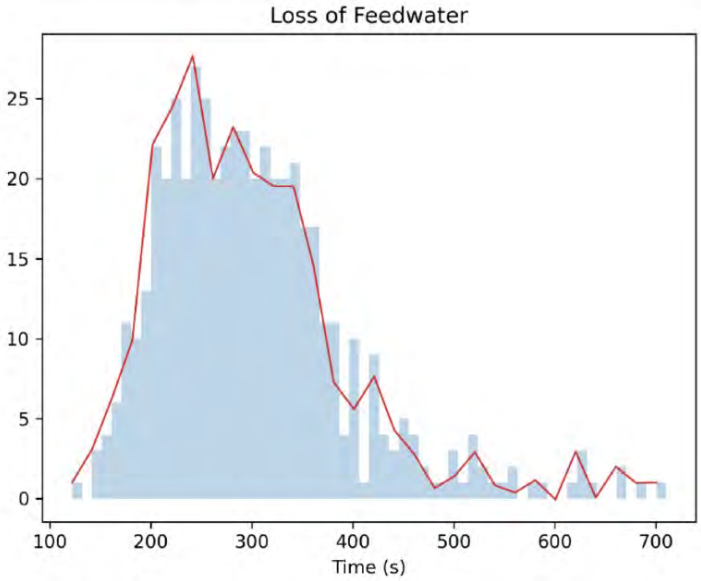


Figure 7. Distribution of times for HUNTER to complete loss of feedwater

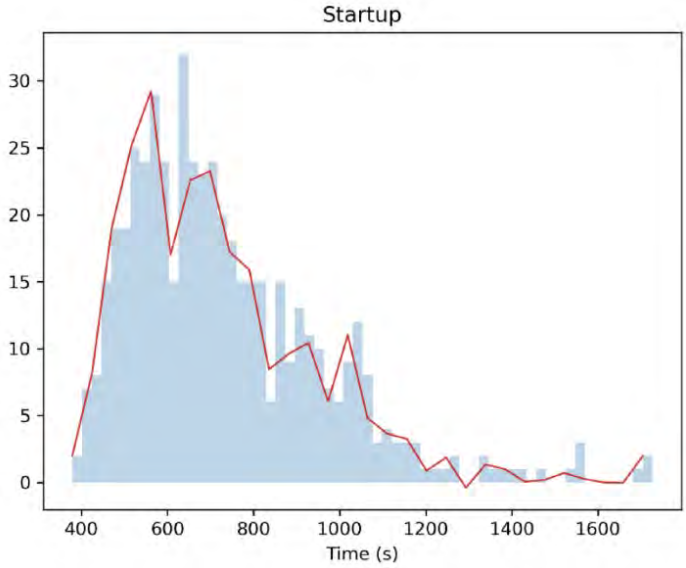


Figure 8. Distribution of times for HUNTER to complete startup

2.3.4 EMRALD-HUNTER

Event Modeling Risk Assessment using Linked Diagrams (EMRALD) (Prescott et al., 2018) was developed by INL as a dynamic PRA software tool to help model causes and mitigations for hardware failures. Both EMRALD and HUNTER have undertaken significant activities to make them more useful for industry applications. For example, ERMALD has recently been coupled to the widely used static PRA event and fault tree modeling software called Systems Analysis Programs of Hands-on Integrated Reliability Evaluations (SAPHIRE), which allows EMRALD to more readily interface with existing plant PRA models (Prescott et al., 2022). As introduced in the previous section (i.e., HUNTER 2.0), HUNTER has evolved from a collection of disparate dynamic HRA models into a standalone, integrated software

tool that also includes an embedded plant simulation to facilitate accurate human-technology interactions (Boring et al, 2022; Lew et al., 2022).

According to this trend, here we integrated these dynamic PRA and HRA tools to create EMRALD-HUNTER for enabling both human and plant risk modeling in a single tool (Lew et al., 2023). Such a tool would benefit analysts by providing a one-stop tool to cover hardware and operational risk. The decision to embed HUNTER was based on the existing wider user base for PRA who would benefit from the addition of greater HRA functionality into PRA modeling tools like EMRALD. HUNTER will continue to exist as a standalone tool for more detailed HRA and human performance efforts, but a streamlined version of HUNTER for general HRA applications has been embedded in the EMRALD code. This streamlined version of HUNTER aims to provide a limited subset of functionality that is within a reasonable expectation of knowledge and expertise for existing probabilistic risk analysts. The intent is the analyst is not required to contend with the bulk of nuances of dynamic HRA models and instead can select from a suite of prebuilt procedures to represent human failure events in their model. This suite of models was created and can be refined by human reliability analysts as needed using the standalone HUNTER software.

Figure 9 shows an example of EMRALD model with HUNTER model elements overlaid for a loss of feedwater scenario. The details on this EMRALD-HUNTER model are described in Lew et al., (2023).

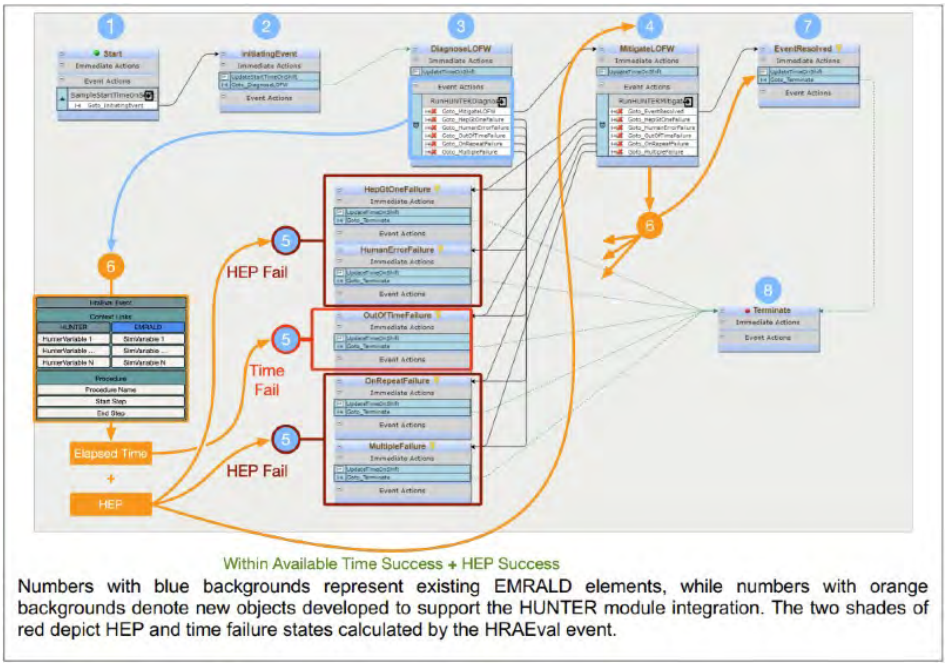


Figure 9. Loss of feedwater EMRALD model with HUNTER module elements overlaid

2.3.5 HUNTER 2.2: New Features

In HUNTER 2.2, new features that can specifically reflect characteristics of dynamic HRA were proposed. These are:

- Spatial HRA
- HUNTER Procedure Performance Predictor (HUNTER-P3)
- HUNTER timing data.

These features are briefly described in the following sections.

2.3.5.1 Spatial HRA

Spatial HRA (Boring, 2023) is the consideration of locations and distances in the determination of human error likelihoods. For many tasks, humans are not stationary, and the only way to model the true risk of their activities is to consider location and changes in location relative to emerging hazards. While it is possible to consider spatial aspects of risk in static HRA, mobility and corresponding changes in risk are best reflected in dynamic HRA.

To understand spatial HRA within HUNTER, consider the three main modules implemented in HUNTER: individual, environment, and task. Spatial HRA allows a way to account for the relative placement of the individual relative to the environment in which they are working. The task accounts for the interplay between the individual and environment and any changes in location. For example, if the main control room calls a field operator (i.e., the individual) to check a pressure level on a local indicator in the turbine building (i.e., the environment), the task involves walking from the current location where the phone is located to the indicator, checking the level, and returning to the phone to report back the value. These three tasks can actually be broken into considerably more detail in a task analysis, including the sub-process of walking between two locations. Dynamic HRA requires a time course to account for the time spent waiting for the remote reading, something not fully covered in simple task level primitives like checking in the field or waiting. Any activity involving a change in location must account for the distance traveled to give a reasonable time estimate. Similarly, any activity involving a change in location introduces new opportunities for human error in the chain of activities. Moreover, those opportunities for human error may change throughout the journey, as the hazards and error traps vary at different locations.

Spatial HRA is influenced by movement factors, including:

- Distance between point A and point B
- Directional changes that may complicate or slow movement
- Navigational ease or difficulty that affect mental workload
- Topographical changes requiring climbing or descending
- Terrain considerations like characteristics of the surface being traversed or obstructions
- Load while moving such as when carrying objects or wearing personal protective gear
- Mobility restrictions such as personal protective gear or weather factors like snow or flooding
- Method of travel such as walking vs. driving a vehicle
- Personnel factors like injury or fatigue.

More details on Spatial HRA are described in Boring et al. (2023a).

2.3.5.2 HUNTER-Procedure Performance Predictor (P3)

Much has been written about control room upgrades and the transition from analog to digital systems (Boring et al., 2019), but relatively little research has been conducted specifically on procedure use with these new systems. An exception is the case of computer-based procedures, where procedures represent one of the technological systems being introduced into the modernized control room (Lew et al., 2018). Despite the minimal research specifically on procedure use amid changing concept of operations (CONOPS), the procedures used to operate any system of the plant are an important part of the overall HSI at the plant.

In recent industry forums to discuss uses of HUNTER, a strong use case has emerged outside traditional applications of HRA for risk assessment. Given the focus in HUNTER on running procedures with a plant simulator, there is a unique and much-needed application of HUNTER to evaluate new

procedures. Existing operating procedures at plants benefit from extensive operating experience, industry benchmarking, sharing lessons learned such as through the Pressurized Water Reactor Owners Group, and continuous improvement through procedure revisions. However, two new situations challenge this process:

- Plant upgrades that introduce new digital systems in the main control room that require new or extensively modified procedures.
- New plants that feature entirely neoteric main control rooms that likewise require new procedures.

These Version Null procedures present potential safety and efficiency concerns for operator performance.

To address this challenge, HUNTER is incorporating a new function called HUNTER-Procedure Performance Preditor (P3). This function uses HUNTER's built-in Monte Carlo tools with human performance variability to identify where in procedures there might be error traps. In this manner, HUNTER-P3 can be used to flag problems with the procedures themselves or issues with the execution of the procedures by reactor operators. HUNTER-P3 can serve as a screening tool for novel procedures to help iterate and refine them prior to deployment. Identified error traps serve to prioritize scenarios where empirical evaluation is warranted.

More details on HUNTER-P3 are well described in Boring (2023).

2.3.5.3 HUNTER Timing Data

Time information for human actions is very important in HRA. HRA methods such as THERP (Swain and Guttmann, 1983), Human Cognitive Reliability (HCR) (Parry et al., 1992), and Korean Standard HRA (K-HRA) (Jung et al., 2005) have used the time information and time response curves to estimate diagnosis HEPs. In the human factors engineering program outlined in NUREG-0711 (O'Hara et al., 2012) HRA methods have been used for investigating the feasibility of HFEs by comparing time required and time available. Time required refers to the duration of time needed by operators to perform a task, while time available is the time in which the operators must complete the task. If the time required for human actions exceeds the time available for an HFE, this is considered a guaranteed failure (HEP = 1.0), and the plant state is assumed to be irreversible.

To date, time windows for calculating the time available have been calculated using thermo-hydraulic analysis, which produces accurate values based on simulations. On the other hand, determining the time required relies on structured interviews with instructors, operators, and other knowledgeable experts. Structured interviews are useful to easily collect approximate estimation on time required for human actions. However, it may be difficult to objectively explain how time required is measured, estimate time distributions or uncertainties depending on each individual or every trial, or judge if human actions can be completed within the timeline when there are unexpected variables interrupting the actions.

An important feature of HUNTER is the focus on timing data and the overall time duration of task performance. Due to the extremely dynamic nature of human performance, this focus is critical to ensuring a robust understanding of human error in complex systems. PSFs impact human performance at a variable level as time during a task progresses; so, including this timing structure can help understand more precisely when human error is more likely. HUNTER uses GOMS-HRA to hold and manage these task timings and durations. GOMS-HRA allows for each task to be broken down into subtask primitives, which can then be summed at various levels to provide timing data for steps of a procedure or entire task performance. While this allows for capturing instances when a task's failure is linked to running out of time, rather than overtly making an error, it also provides a critical contextual data point that can be used to dig into human performance data and better capture when error rates rise and fall and when various PSFs trigger human errors.

Our research team has developed an HRA data collection framework called the Simplified Human Error Experimental Program (SHEEP) to complement full-scope simulator studies as well as collect input

data for dynamic HRA like HUNTER (Park et al, 2022). The SHEEP framework aims to infer full-scope data based on experimental data collected from simplified simulators, specifically Rancor and the Compact Nuclear Simulator (CNS) (Kwon et al., 1998). Within the SHEEP framework, our research team has experimentally collected human reliability data from 36 student operators and 36 professional operators when they used CNS and Rancor.

From within the umbrella of the SHEEP framework, our research team investigated time distributions for GOMS-HRA task level primitives using the SHEEP database, which includes experimental data when twenty student operators and twenty professional operators using Rancor. From the experimental data, time required for GOMS-HRA task level primitives to satisfy thirteen statistical distributions was investigated. Then, the time distributions from student operators and professional operators are compared and discussed.

Table 1 shows an example of goodness-of-fit test result for thirteen statistical distributions on elapsed time of the five GOMS-HRA task primitives in the emergency operating procedure (EOP)-E0 (using Westinghouse numbering nomenclature) depending on participant type (i.e., novice student vs. licensed operator). For statistical distributions for goodness of fit, p-values over 0.05 (i.e., $p>0.05$) mean that there is not enough evidence to reject the null hypothesis that the data follow the hypothesized distribution. This is not the same as proving the sample data follow the specified distribution, but instead provides statistical evidence that the distribution fits the data. The grey-highlighted boxes in the table indicate that elapsed time for a task level primitive has a $p>0.05$ for each distribution. For example, A_C for student operators supports a normal distribution after a Johnson transformation, as shown in the table. For this result, Figure 10 shows the more detailed analysis. The details on the HUNTER timing data research are described in Park et al. (2024).

Table 1. Time distribution analysis on the five GOMS-HRA task level primitives in EOP-E0 depending on participant type (student vs. operator)

Distribution	**P-value of Goodness of Fit Test**									
	Student					**Operator**				
	A_C	**C_C**	**R_C**	**S_C**	**D_P**	**A_C**	**C_C**	**R_C**	**S_C**	**D_P**
Normal	<0.005	<0.005	<0.005	0.014	<0.005	<0.005	<0.005	<0.005	0.237	<0.005
Normal (after Box-Cox transformation)	0.374	<0.005	0.010	0.653	0.070	0.340	<0.005	<0.005	0.237	0.041
Lognormal	0.374	<0.005	0.010	0.404	0.070	0.340	<0.005	<0.005	0.031	0.041
Exponential	0.023	<0.003	0.018	0.486	0.051	<0.003	<0.003	<0.003	0.021	<0.003
2-parameter exponential	0.083	<0.010	<0.010	>0.250	0.011	<0.010	<0.010	<0.010	0.012	0.034
Weibull	<0.010	<0.010	<0.010	0.189	<0.010	0.015	<0.010	<0.010	0.236	0.022
3-parameter Weibull	0.013	<0.005	<0.005	0.404	<0.005	0.084	<0.005	<0.005	0.254	0.006
Smallest extreme value	<0.010	<0.010	<0.010	<0.010	<0.010	<0.010	<0.010	<0.010	0.092	<0.010
Largest extreme value	<0.010	<0.010	<0.010	0.037	<0.010	0.088	<0.010	<0.010	0.227	0.016
Gamma	<0.005	<0.005	0.007	0.208	0.006	0.167	<0.005	<0.005	0.182	0.047
Logistic	<0.005	<0.005	<0.005	0.016	<0.005	<0.005	<0.005	<0.005	0.235	<0.005
Loglogistic	>0.250	<0.005	<0.005	>0.250	0.032	0.233	<0.005	<0.005	0.104	0.017
Normal (after Johnson transformation)	0.563	N/A	N/A	0.763	N/A	0.364	N/A	N/A	N/A	N/A

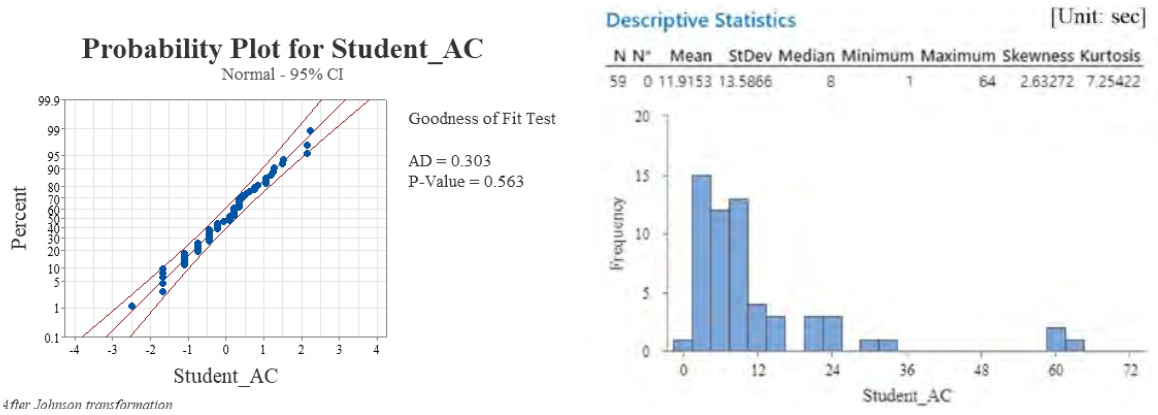


Figure 10. Normal distribution (after Johnson transformation) of the task level primitive A_C for student tasks in the EOP-E0 procedure

2.4 New Directions in HUNTER

The previous discussion on the environment module demonstrates *coupling* between two types of models. Coupling is defined as how two simulations exchange information and what information they exchange. In the case of coupling between HUNTER and RELAP5-3D, one of the interfaces in HUNTER 1.0 and 2.0, the human and thermohydraulic models may be said to operate asynchronously. *Asynchronous coupling* occurs when each model operates independently such that information is exchanged only at the beginning or end of model runs.

For example, the rare but risk-significant event of a steam generator tube rupture (SGTR) in a pressurized water reactor requires mitigative actions by operators using at least four procedures:

- An annunciator response procedure that triggers a set of immediate memorized actions by the operator when an alarm goes off
- When the entry conditions warrant, this will elicit an abnormal operating procedure (AOP) that will prioritize a series of rapid checks to determine the severity of the plant upset condition
- If the plant automatically trips (meaning it drops the fuel rods into a graphite sheath to neutralize reactivity) or if the conditions escalate to the point of requiring a manual trip by the operators, a post-trip EOP is referenced, which will prescribe protective measures such as ensuring adequate cooling of the reactor and further diagnosing the source and corrective actions of the problem
- Within the post-trip EOP, there will be a branching point to a more specific EOP to mitigate the SGTR once the specific ruptured steam generator is identified.

In HUNTER, these procedures would be coded as a continuous action monitoring procedure that detects alarms, followed by entry into AOP-16, then EOP-E0 for post-trip actions, and finally EOP-E3 for SGTR mitigative actions (using Westinghouse procedure labeling conventions here for pressurized water reactors).

Using the SGTR scenario as an example, this means that the RELAP5-3D SGTR-specific model runs on its own, initiated by the HUNTER scheduler with inputs to RELAP5-3D related to operator performance such as how long it would take to initiate safety injection. The RELAP5-3D model executes using this information and produces outputs related to plant parameters after the scenario completes. The inputs on human performance would be a distributions, e.g., a range of how long it takes operators to initiate safety injection.

Synchronous coupling incorporates the interactions of the human and the system and runs in response to these changing conditions. Both task and environment modules continuously exchange information in the form of a feedback loop—the operator responds to plant conditions and acts to make changes to the plant, which in turn changes the subsequent plant conditions to which the operator responds. The back and forth between the operator and the plant represents an infinite scope of interactions requiring both modules to respond step-by-step to each other.

Asynchronous and synchronous model coupling are shown in Figure 3 (Boring et al., 2023b). To truly capture interactive dynamics, HUNTER functions best through synchronous coupling. Synchronous coupling allows not only a continuous feedback loop between the task and environment modules, it also allows the individual module to shape the performance of the virtual operator to ensure realistic downstream effects. For example, an operator response to a particular plant condition might vary between 1 and 5 minutes. A distribution to cover the response time could be constructed a priori and fed asynchronously into a thermohydraulic plant model. What such an a priori model might fail to consider is that in the longer time windows, eroded plant margins cause many alarms, which may elevate the stress of the operator. Stress can have the effect of slowing decision making, thereby further slowing the operator response and exacerbating the plant upset. The interplay between the individual, task, and environment cannot be determined a priori for an evolving event. Each of the modules is dynamic, but none are independent. Thus, realistic dynamic HRA requires synchronous coupling in most cases. 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.

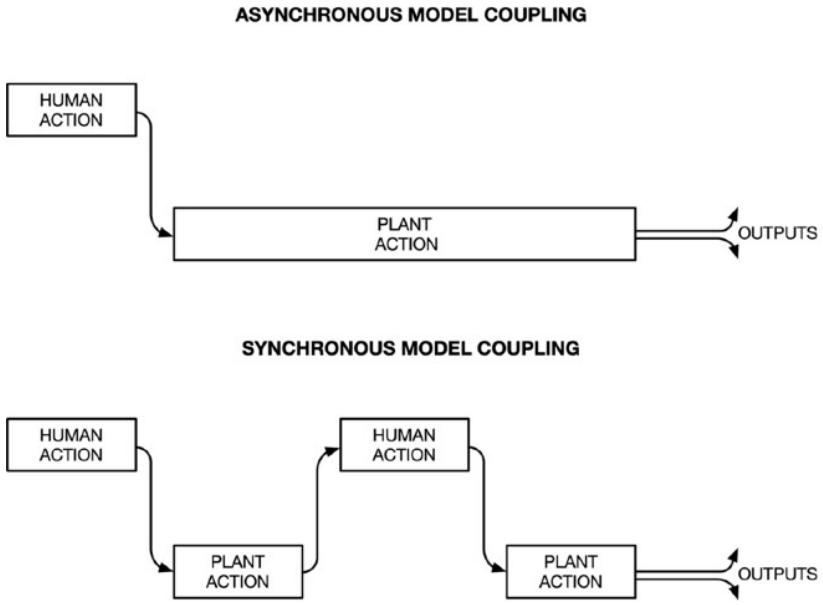


Figure 3. Human-plant model interaction for asynchronous and synchronous coupling

The problem of forecasting future (t +1) human actions (h) can be roughly depicted as follows:

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

This implies that future human actions are dependent on the present (t) actions in relation to the present status of the system or plant (s). Similarly, future states of the system can be approximately expressed as:

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

This implies that future states of the system depend on the current state of the system as influenced by the current human actions. The issue here is the cyclical timeline involved when attempting to anticipate the interactions between human and system models before either actually occurs. Human actions and the state of the system are interdependent—the state of the system affects human actions just as human actions affect the state of the system. Attempting to calculate one without considering the other is impossible in a discrete event simulation.

The most ubiquitous plant model capable of synchronous coupling is a plant simulator. A model in operation essentially forms a simulation, whereas a simulator represents an interactive simulation built to accommodate human inputs. Generally, simulations function independently from other models or human involvement, exhibiting asynchronous characteristics. Simulators, on the other hand, operate synchronously, enabling regular communication with other models or humans. Every nuclear power plant in the world is required to have a full-scope (i.e., high fidelity) plant simulator that is capable of representing realistic plant responses to normal and abnormal operations. Plant simulators are used in training reactor operators for their initial license to operate the plant and for recurring refresher training, including just-in-time training suitable for known challenging scenarios at the plant such as startup after a refueling outage. Because simulators are designed to interface in real-time with actual operators, they accurately reflect this feedback loop and serve as an ideal environment module for coupling with HUNTER.

A typical plant simulator may feature 100,000 plant parameters on the backend, with up to 10,000 indicators and controls displayed in the main control room HSI (see Figure 4). The simplified simulator found in the Rancor Microworld Simulator provides an excellent first-order model for validating coupling with HUNTER. Rancor has been benchmarked between student and licensed reactor operators and against higher fidelity simulators (Park et al., 2022). Unlike full-scope plant training simulators, it is able to run at considerably faster than real time, making it well suited for the multiple thousand scenario runs common in Monte Carlo simulations. Such large numbers of runs are required for HRA, where many HEPs are in the range of 1E-3 (errors at 1/1000 times a task is performed) or smaller, requiring considerable resampling to evidence errors.

Figure 4. The Human System Simulation Laboratory at INL (Boring, 2020)

This page intentionally left blank.

3 RANCOR

3.1 Need for Rancor

Control room simulators are virtual representations of NPPs' physical, dynamic, and operational aspects in part or whole. Full-scope simulators represent a high level of plant fidelity, including the entire power plant from the reactor core to the electrical switchyard and auxiliary systems. The history of simulators in NPP process control began with low-fidelity prototypes of control panels in the early 1970s and morphed rapidly into high-fidelity full-scope simulators with functional control panels interfacing with underlying thermal-hydraulic code (Skjerve and Bye, 2011). From a mere tool for visual illustration, by the end of the decade, control room simulators had evolved into highly representative and fully interactive systems used in training reactor operators and assessing their knowledge of diverse concepts of operations, scenarios, and procedures under different plant operating conditions.

Beyond their traditional use for operator training, control room simulators occupy a crucial place in NPP human factors research. They act as testbeds for conducting human performance and human factors studies and to validate new plant designs, models, and concepts of operations. While training simulators at NPP facilities could potentially serve as environments for some human factors research, several barriers prevent them from being used for research purposes. The most notable barriers relate to the availability and fitness-for-purpose of plants' control room simulators for research. The continuous operator training requirement mandated by Regulatory Guide 1.1149 (U.S. Nuclear Regulatory Commission, 2001) renders control room facilities almost unavailable for non-training-related use. Furthermore, plant simulators are not built for research purposes, and their design fundamentally limits their use in human factors research. For example, regulatory requirements mandate plant simulators to functionally represent the actual main control room, making it almost impossible to achieve experimental manipulations required for research without license violations due to alterations of the simulator's rigid metal control boards. The overburdened nature of training simulators and their lack of flexibility to accommodate experimental manipulation necessitated the need for dedicated control room simulators to support research and development activities.

Glasstop simulators offer a promising solution with virtual analog bays representing the physical control boards digitally displayed (Boring et al., 2013). Armed with virtual digital emulation afforded by modern computing, human factors researchers can use full-scope glasstop simulators as testbeds for validating new plant models, digital control systems, and concepts of operations that might prove difficult (or impracticable) to test in training simulators. Human System Simulation Laboratory (HSSL; Boring et al., 2013) at INLis one of the leading examples of a full-scope glasstop simulator built to conduct control room modernization research. The control room simulator at the HSSL has virtual reconfiguration capabilities, meaning the glasstop simulator can display the virtual control room panels from different plants by switching the underlying simulator software. Full-scope simulators like the HSSL can digitally represent the main control room. As such, modifying the digitally represented control boards with prototypes of new interfaces becomes feasible. Together, the capabilities of full-scope simulators address the challenge of availability and fitness-for-purpose posed by training simulators. However, they also introduce a new challenge regarding operator recruitment for control room studies.

Control room operators work as a crew with NPP facilities maintaining a couple of operating crews. Plants run 24 hours a day and 7 days a week operations. Outside the operational demands of their schedules, operating crews spend considerable time in training when not in charge of operating the plant. This means a limited supply of operators and, by extension, a lower likelihood of recruiting a sufficient number of operators to run a control room study spanning about a week (Boring et al., 2013). The challenge of limited operators for conducting control room research necessitates using an alternative approach that minimizes pulling a large group of operators together over a considerable amount of time.

Discount usability enables assessing user performance using simplified measurement techniques as alternatives to complicated data-rich ones under resource constraints. Discount usability involves three main concepts: simplified user testing, narrowed-down prototypes, and heuristic evaluation (Nielsen, 1989). Applying the idea of discount usability, a reduced fidelity version of a full-scope simulator can be used to represent only essential elements required to enable the administration of simple research techniques that yield valuable usability insights in a short, iterative manner. Also, discount usability allows novices, like college students, to be used as participants to test some basic psychological issues operators encounter. Indeed, even well-trained novices' performance will differ from that of operators, as novices may not understand the underlying operational significance of most of their actions. However, novices have been shown to be equally suitable candidates for usability evaluations after sufficient training using a reduced-scope simulator (Ulrich et al., 2017). Another significant benefit of adopting the discount usability approach to simplified testing is the ability to extend research that would otherwise be performed using a full-scope simulator in large research facilities to smaller settings such as academia with a large enough sample of participants required to yield statistically significant results. Implementing the principles of discount usability in a microworld environment, led to the development of Rancor. Rancor is a gamified reduced-scope simulator platform for conducting basic nuclear process control research with operator surrogates to produce meaningful results that are generalizable to expert operators.

3.2 History of Rancor

Rancor is a simulator of a simplified pressurized water reactor (PWR) process that includes a backend simulation and a frontend HSI. The collaborative exploration of simpler and cost-effective alternatives for studying theoretical and practical concepts related to process control by University of Idaho and INL researchers led to the idea of using the reduced scope and complexity of microworlds to address human error and performance-related issues in process control environments (Ulrich, Werner, & Boring, 2015). The initial idea was to create a reduced-order-magnitude (ROM) simulator platform by implementing the concepts of discount usability in a microworld to enable simple usability research on the HSI and basic operational aspects of nuclear operations.

ROM implies only essential processes of the reference plant, and the corresponding monitoring and control components are represented in the simulator. Heat production by the reactor core and its onward transmission to the secondary system through the grid is a core process of a nuclear reactor. Using a gamified water-based Rankine cycle simulation, Rancor simulates a simplified PWR heat production process. The name, Rancor was derived from a combination of the first three letters of the RANkine cycle and the system where the process it simulates takes place in the reference plant, namely the reactor CORe.

The water-based Rankine cycle implemented in Rancor uses a mathematical model showing how thermal energy generated from the phase change of liquids is converted into mechanical energy for driving a turbine to generate electricity (Ulrich, 2017). The thermohydraulic simulation in the Rancor simulator simplifies the general water-based Rankine cycle in line with gamification principles. However, to simplify the system according to gamification principles, some level of divergence between the Rancor microworld's underlying thermohydraulic simulation and the general water-based Rankine cycle was accommodated.

Rancor's HSI uses a piping and instrumentation diagram (P&ID) to illustrate the plant systems and components graphically. Generally, the HSI contains two segments representing the primary and secondary loops depicted in a conventional NPP control room. The primary loop is on the left side and labeled in red, while the secondary loop is on the right side and labeled in yellow (see Section 3.3 for illustrations). The primary side contains the reactor vessel, recirculating coolant pumps, piping, and temperature instrumentations. Components on the secondary side include steam generators, turbines, condensers, feedwater pumps, and piping. The layout of the HSI has a horizontal alarm annunciator tile at the top and controllers at the bottom, all corresponding to the components on the primary and secondary loops. The controllers of the primary loop include rod controls for manipulating reactivity, translating to

Table 2. Versions of the Rancor Microworld Simulator

Version	Description	HSI	Code	Characteristics	Users/Domain
1.1	Used to examine attention.	WPF (XAML)	C#	Combined overview and control display, situation awareness freeze probes (cognitive constructs)	Student
1.2	Multi-unit, expanded for SMRs additional displays	WPF (XAML)	C#	4-units, radar display	Demonstration in HSSL
1.3	Validation study comparing student performance with expert operators	WPF (XAML)		SI units, green-yellow color scheme, Korean/English procedures (9 scenarios; faults)	Operators, nuclear engineering students
1.4	Implemented the "Jabba" command line interface. Coupled with HUNTER.		C#	Dynamic HRA plant simulator v4.1 C# and v4.2 python variants	HUNTER
1.5	Thermal power dispatch (TPD) steady state		Python	Python-bases	Thermal Dispatch
1.6	TPD concept of operation verification using students	WPF (XAML)	C#	Integrated with thermal dispatch HSI	Human factors graduate students
1.7	Agnostic HSI style generator to conduct style assessment for a U.S. Nuclear Power Startup Vendor	WPF (Adobe Illustrator Graphics)	C#	4 style variants, computer-based procedures, Mini-scenarios	U.S. nuclear power plant personnel
1.8	Failsafe Automated Timed Response (FATR): Light-weight Computerized Operator Support System (COSS), used to examine human error dependency	WPF (XAML)	C#		Dependency study
2.0.py	Thermal power dispatch (TPD) control room simulator	Reactjs (node API)	Python	Physical equipment coupling/control	Thermal dispatch
2.0.fl	Reimplemneted in Dart/Flutter framework to test different levels of computer-based procedures (CBP) in control room	Flutter	Flutter	CBP introduced	Students
2.5/2.6	Digital Procedure System (DPS) including CBP in WPF Rancor version. Replication of study conducted with Rancor 2.0.fl using an older population	WPF	C#, JSON Procedures		Aging study
3.0	Rancor-HUNTER: Version that has HUNTER built in. Can run as real-time simulator or as a Monte Carlo simulation environment for Rancor	WPF, Flask Python	C#	Procedure editor, HUNTER integration	HUNTER

the heat energy produced by the reactor. The secondary side controller includes valves to control water and steam flow through the loop.

3.2.1 Versions of Rancor

Since its first development, Rancor has undergone incremental iterations. Table 2 briefly summarizes the evolution of the different versions of Rancor, with corresponding attributes and use cases that necessitated them. These are elaborated below.

3.2.1.1 Rancor 1.1

The first version of Rancor was identified as Rancor 1.1, and subsequent versions followed the same *major.minor* naming convention: 1.2, 1.3, and so on. Several development tools, environments, and code bases have been used across the evolution of Rancor versions, resulting in different graphical user interface (GUI) displays based on corresponding use cases. Rancor 1.1 was a single-unit display reactor simulator with GUI developed in Microsoft Windows Presentation Foundation (WPF) using Extensible Application Markup Language (XAML) code editor. The dynamic control behavior of the GUI components was scripted in Microsoft's C# programming language. Using college students, Rancor 1.1 was developed to measure cognitive constructs of situation awareness and attention in a nuclear process control setting (Ulrich, 2017). This use case necessitated the integration of two unique features, freeze probes, and attention markers, for the sole purpose of measuring situation awareness and attention.

3.2.1.2 Rancor 1.2

The subsequent versions of Rancor had their unique defining attributes. The single unit of the first version was expanded to include multiunits for small modular reactor (SMR) display in Rancor 1.2. For economic efficiency, SMRs are forecasted to be deployed as a fleet, introducing the need to monitor multiple simultaneous operations from a single control room (O'Hara et al., 2010). Rancor 1.2 simulated a four-unit radar display and evaluated its implementation feasibility, which was used for demonstration in the HSSL (Boring, 2020).

3.2.1.3 Rancor 1.3

The need to compare student performance with expert operators using Rancor gave impetus for Rancor 1.3 to be used in conjunction with Chosun University in South Korea. Park et al. (2022) conducted a study comparing students vs. operators across multiple normal and abnormal scenarios. An additional study was conducted to investigate undergraduate nuclear engineering students' learning effects and performance trends using Rancor, focusing on HRA data collection (Yang et al., 2023). The results show significant improvements in situation awareness, task completion time, and accuracy across trials, with students' accuracy levels eventually matching those of trained operators.

3.2.1.4 Rancor 1.4

Rancor 1.4 used the "Jabba" command line and encapsulated the model of Rancor in a .NET library that could be used independently of WPF and decoupled from the HSI. The new model enabled the use of Rancor across non-WPF platforms like Unity3D. The Python version of this model was also tightly coupled to HUNTER (Lew et al., 2022).

3.2.1.5 Rancor 1.5/1.6

The Rancor platform has been used to prototype advanced control room concepts, such as the Thermal Power Dispatch (TPD) system. The TPD diverts excess thermal energy generated from nuclear power plants during periods of low electricity demand to produce hydrogen through high-temperature steam electrolysis (HTSE; Ulrich et al., 2021). The TPD system is modeled using Python code in Rancor 1.5, and the GUI for TPD was developed in WPF (XAML) using C# code in Rancor 1.6. Rancor 1.6 was

used to conduct usability tests to identify HSI issues with the participation of students (Gideon et al., 2024).

3.2.1.6 Rancor 1.7

As a platform that enables prototyping of advanced control room operations concepts, Rancor was used by a group of researchers from INL, in collaboration with a nuclear vendor, to conduct style assessments (Hall et al., in press; Boring et al., 2023c). The GUI was designed in Adobe Illustrator and exported as Scalable Vector Graphics (SVG), then migrated to XAML for incorporation with WPF. Style variants, including HSIs with skeuomorphic and neuromorphic styles, were prototyped and evaluated.

3.2.1.7 Rancor 1.8

Rancor 1.8 was born out of efforts to simulate a platform that enables empirical research into human error dependency. This version's unique feature was the implementation of a Failsafe Timed Automated Response (FATR) operator support system to conduct simple dependency studies in nuclear process control operations (duBois, Lew, and Boring, 2023). In version 1.8, Rancor's GUI was adapted to a simplified two-column procedure format for novice operator studies.

3.2.1.8 Rancor 2.0.py

The Rancor 2.0.py is a thermal power dispatch (TPD) control room simulator with physical equipment to support a high-fidelity control room demonstration and testing (Lew and Ulrich, 2023). The TPD simulator was modeled using Python code as in Rancor 1.5. However, Rancor 2.0.py had two major modifications, differentiating it from its preceding version 1.5. First, Rancor 2.0.py involved physical equipment coupling/control. Second, this version used Reactjs (node API) for the GUI development. These two modifications enabled the use of the simulator for high-fidelity control room studies.

3.2.1.9 Rancor 2.0.fl

The need to test different levels of computer-based procedures (CBT) in control room operations necessitated the development of Rancor 2.0.fl. With both GUI and code-based implementation in the Dart/Flutter framework, Rancor 2.0.fl developed and tested three levels of CBP for the first time in Rancor using student participants (Hall et al., 2023). This study was subsequently used to establish empirical evidence for dependency in HRA (Boring et al., 2023d)

3.2.1.10 Rancor 2.5/2.6

A replication of the study conducted with Rancor 2.0.fl was done using an older population. The new study was conducted using Rancor versions 2.5/2.6. Rancor 2.5 and 2.6 implemented 3 levels of Digital Procedure System (DPS) and 3 types of CBP, respectively. The general code-based was in C#, with the procedures written in JavaScript Object Notation (JSON). Rancor 2.5/2.6 had a GUI developed in WPF.

3.2.1.11 Rancor 3.0

Rancor 3.0 was the first extension of Rancor to simulate human error. This version integrated HUNTER with Rancor in C# to form the hybrid Rancor-HUNTER described in this report. Rancor 3.0 enables scenario runs as a real-time simulator or as a Monte Carlo simulation environment. These capabilities provide an opportunity for a more realistic estimation of the HEP for procedures to perform existing plant scenarios and for procedures involving novel concepts of operations like TPD.

3.3 Visual Guide to Rancor

3.3.1 Rancor Interface

Figure 45 depicts one of many possible layouts of the interface for Rancor. In this configuration with three physical monitors, Rancor includes an overview display with alarms on the top, the P&ID display in the middle, and the control and procedure displays located on the left and right sides of the bottom monitor.

In this configuration, Rancor relies on glasstop bays within the HSSL at INL. The operator manipulates Rancor by using the bottom control display and the digital procedure display. The same types of interactions contained within a full-scope simulator are also found within Rancor that allow the operator to raise and lower control rods, open and close various pumps, and even synchronize the plant to the electrical grid. Rancor was designed for flexibility to enable university research groups to take advantage of the simplified simulator to collect student data. As such, Rancor can be reconfigured as a desktop workstation using two 4K monitors with a mouse and keyboard as a preferred setup.

Figure 11. Rancor simulator displayed on a virtual bay supporting touch interaction with four distinct displays including overview (a), piping and instrumentation diagram (b), controls (c), and digital procedures (d)

3.3.1.1 The Overview Window

The top screen of the Rancor simulator interface is the overview window. The overview window collects much of the important information from the P&ID and puts it in one convenient location.

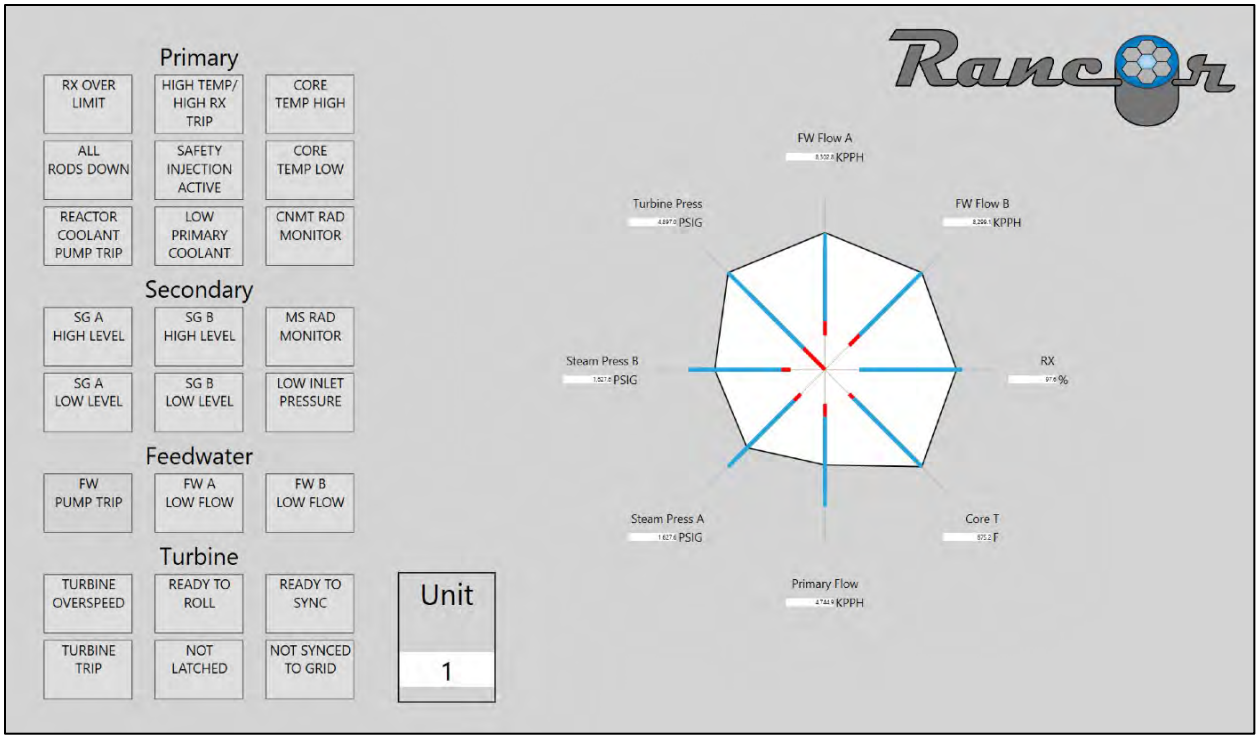


Figure 12. The overview window of Rancor simulator

On the right side of Figure 12, there are indicators showing important parameters of the system, such as reactor power, core temperature, feedwater flow, steam generator pressure, and primary coolant flow. These indicators have the radar plot. This is a good 'at a glance' way to see how the plant is behaving. The blue and red lines represent low normal and high normal operating range. So long as the radar plot is between these two lines, the plant is operating safely. If everything is running perfectly, the shape will be octagonal or circular, as depicted in Figure 12.

The left side of Figure 12 is the alarm panel. (Note that different versions of Rancor may place the alarms right or left, according to preference.) These alarms light up if serious situations arise in the plant such as if there's radiation detected where there shouldn't be any, or if the steam generator water levels are too low. In addition to critical events, the alarm panel also tells the operator when the plant is ready to start spinning the turbine, indicated by the 'ready to roll' alarm being illuminated, and when it's ready to synchronize to the power grid, indicated by the 'ready to sync' alarm being illuminated. It is a good idea to keep an eye on the alarm panel, throughout the simulation.

3.3.1.2 The Piping and Instrumentation Diagram (P&ID) Window

The middle screen of Rancor simulator is the P&ID window. The P&ID window depicts the relationship between piping, process equipment, instrumentation, and control components. This shows the state of the plant. All the information presented in the P&ID window is located approximately where it would physically be in relation to other systems at an actual plant.

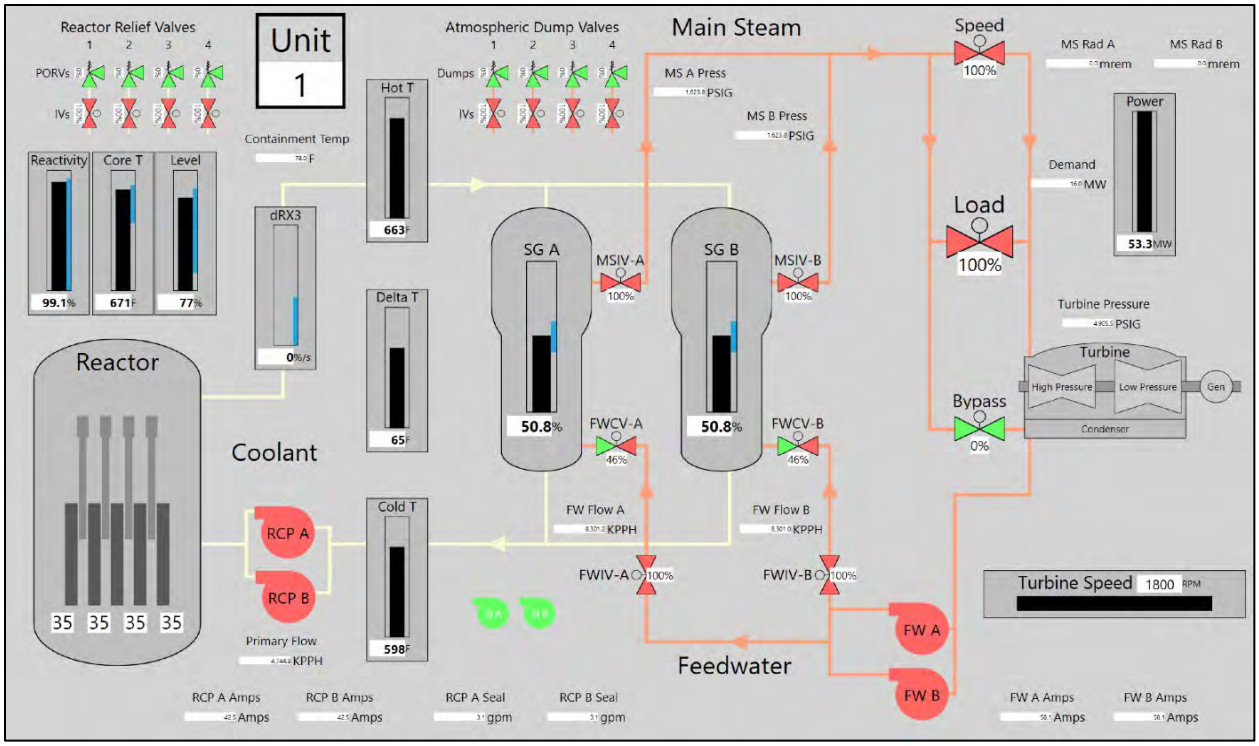


Figure 13. The P&ID window of Rancor simulator

3.3.1.3 The Control Window

The bottom screen includes the control window as shown in Figure 14. The operator controls the reactor by touching different components on this window.

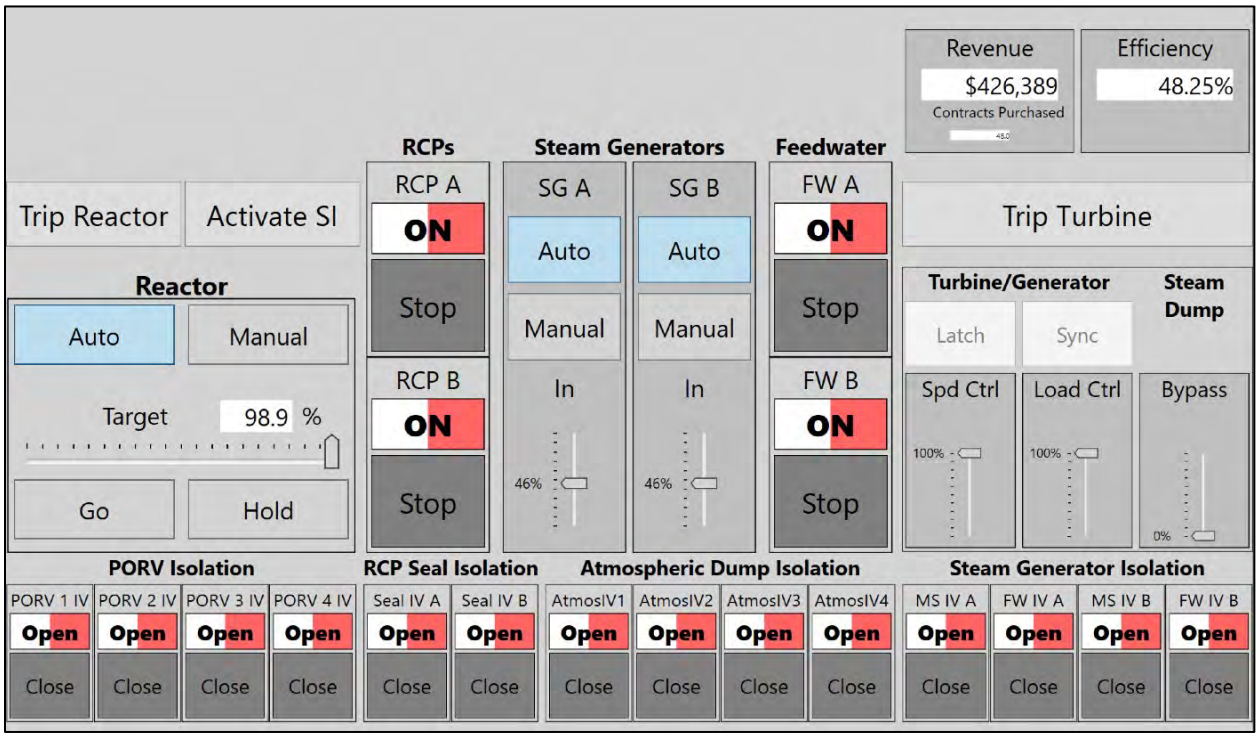


Figure 14. The control window of Rancor simulator

3.3.1.4 The Procedure Window

The bottom screen includes a procedure window as shown in Figure 15. The procedure window provides a list of selectable procedures arranged along the left edge. A single procedure can be displayed by selecting from the list of procedures on the left. The procedures provide a sequential series of steps in a single or two-column format depending on the type of operation. Details on the Digital Procedure System can be found in Section 3.4.6 of this report. Rancor supports Types 1-3 computer-based procedures, corresponding to (1) digital versions of paper-based procedures with placekeeping, (2) digital versions with embedded indicators, and (3) digital versions with embedded indicators and controls.

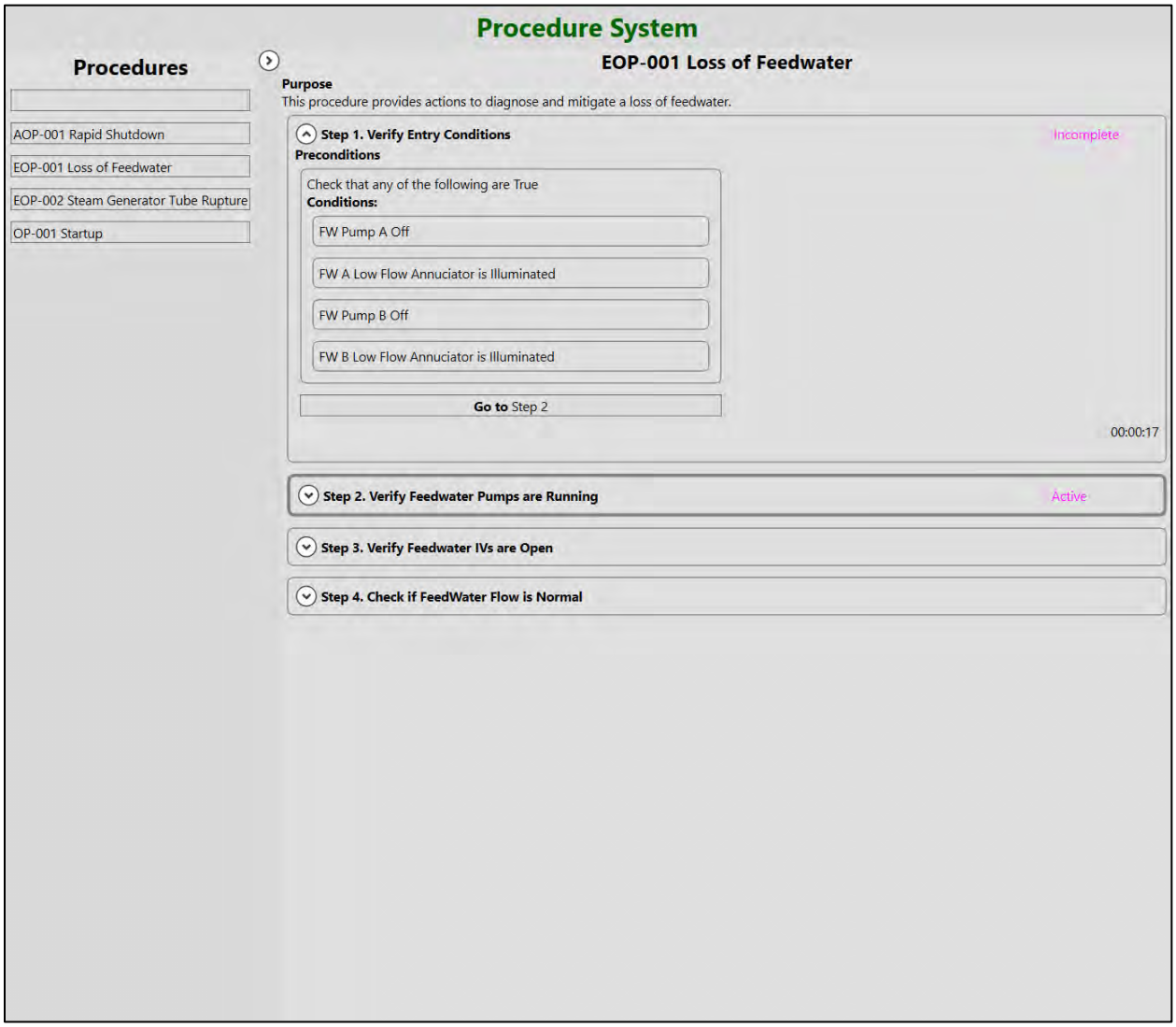
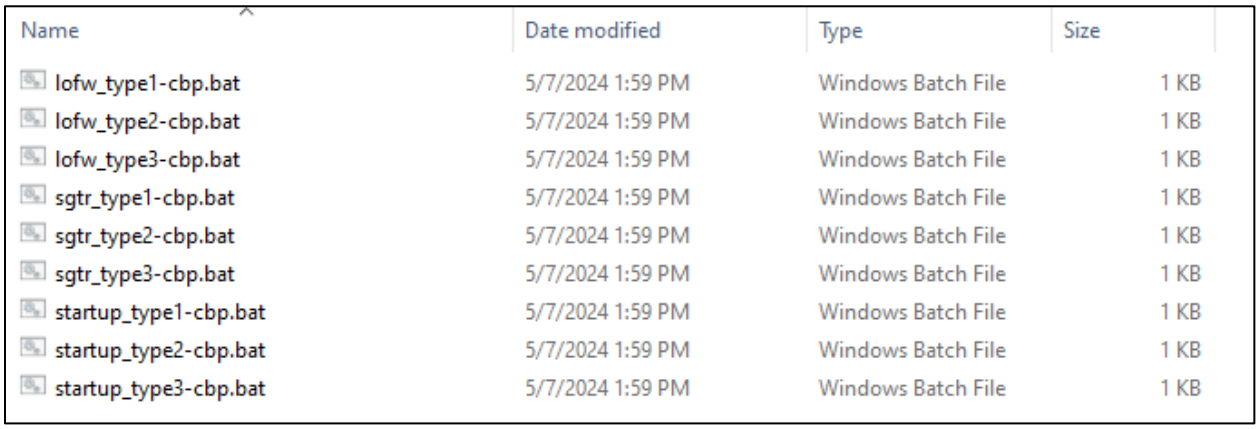


Figure 15. The procedure window of Rancor simulator

3.3.2 Open Simulator

This section explains how to launch Rancor. When installed, there will be several predefined batch (.BAT) files that can be selected. There are several scenarios in the following path in the Rancor simulator folder (see Figure 16). The accident scenarios include loss of feedwater (LOFW), SGTR, and Startup. Users can select from type 1 to type 3 depending on the computer-based procedure type.

- Path: [..\Application\Executables\deploy]



Name	Date modified	Type	Size
lofw_type1-cbp.bat	5/7/2024 1:59 PM	Windows Batch File	1 KB
lofw_type2-cbp.bat	5/7/2024 1:59 PM	Windows Batch File	1 KB
lofw_type3-cbp.bat	5/7/2024 1:59 PM	Windows Batch File	1 KB
sgtr_type1-cbp.bat	5/7/2024 1:59 PM	Windows Batch File	1 KB
sgtr_type2-cbp.bat	5/7/2024 1:59 PM	Windows Batch File	1 KB
sgtr_type3-cbp.bat	5/7/2024 1:59 PM	Windows Batch File	1 KB
startup_type1-cbp.bat	5/7/2024 1:59 PM	Windows Batch File	1 KB
startup_type2-cbp.bat	5/7/2024 1:59 PM	Windows Batch File	1 KB
startup_type3-cbp.bat	5/7/2024 1:59 PM	Windows Batch File	1 KB

Figure 16. The scenarios of Rancor simulator

When users select a scenario with a computer-based procedure type, they will be asked to enter a participant identifier (ID) as shown in Figure 17. This will be the file name of the data being saved, so it is important to avoid duplication.

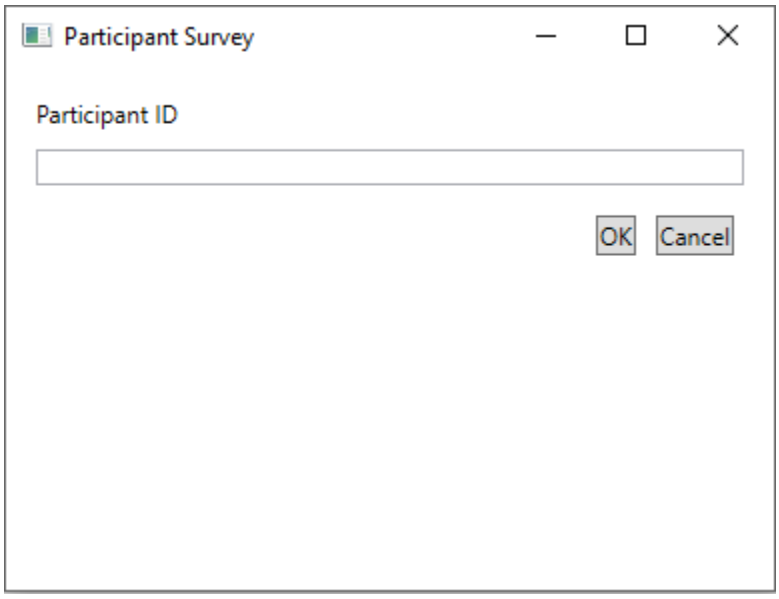


Figure 17. The data file name

When users enter the participant ID, four windows of Rancor simulator (i.e., overview, P&ID, control, and procedure windows) and the execution window will pop up. In the execution window (see Figure 18), the operator or experimenter can run or pause the simulator, and the simulation time is recorded. When the user clicks the 'Exit' button, the simulator is terminated.

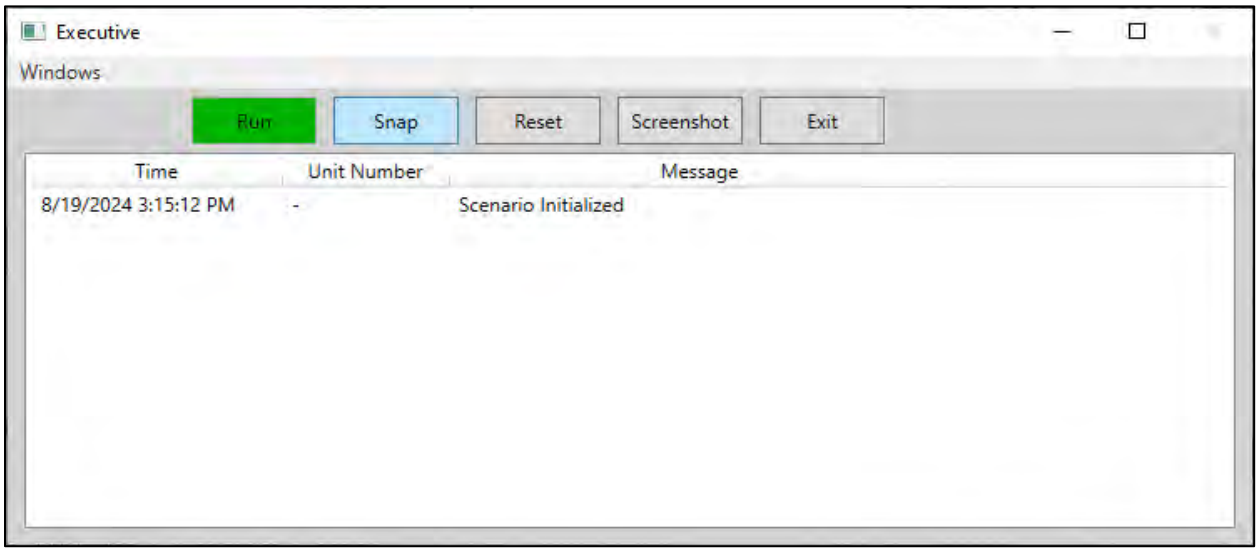


Figure 18. The execution window for Run the simulator

Upon activation of the 'Run' button by the user, the simulator will begin as shown in Figure 19. The screen is activated with a bright color. It will set under the initial condition and, after a certain amount of time, may encounter a malfunction as dictated be the given scenario.

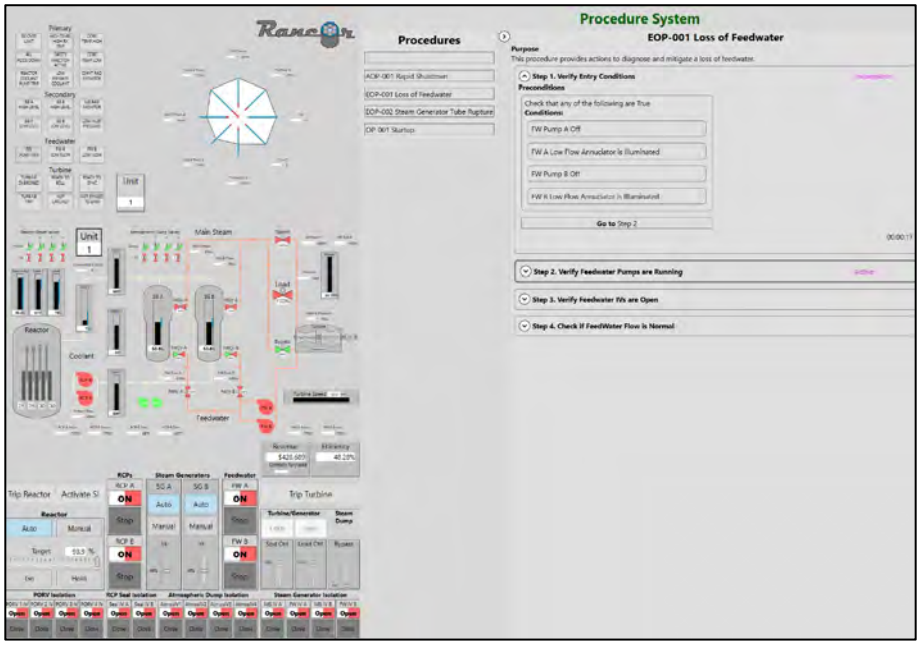


Figure 19. Rancor simulator windows for running the simulator

The users can use the following keys to adjust the Rancor simulator windows. The windows can be fixed or moved and zoomed in or out. As shown in Figure 20, when the yellow top bar of the window is activated, the user can adjust the screen size and location.

- ***Ctrl*** + ***'R'*** : Resize, move, or fix selected screen
- ***Ctrl*** + **'+'** : Zoom in (make bigger) the selected screen
- ***Ctrl*** + **'-'** : Zoom out (make smaller) the selected screen

The user enters this mode by pressing Ctrl + R. Each window is selected individually and adjusted. It is possible to drag the windows to preferred locations and resize them using the Ctrl keys for zooming or by dragging the corners to rescale. Rancor will automatically scale objects to the window dimensions. When the user is done configuring the windows, they again press Ctrl + R to exit layout mode and save the configuration for future runs.

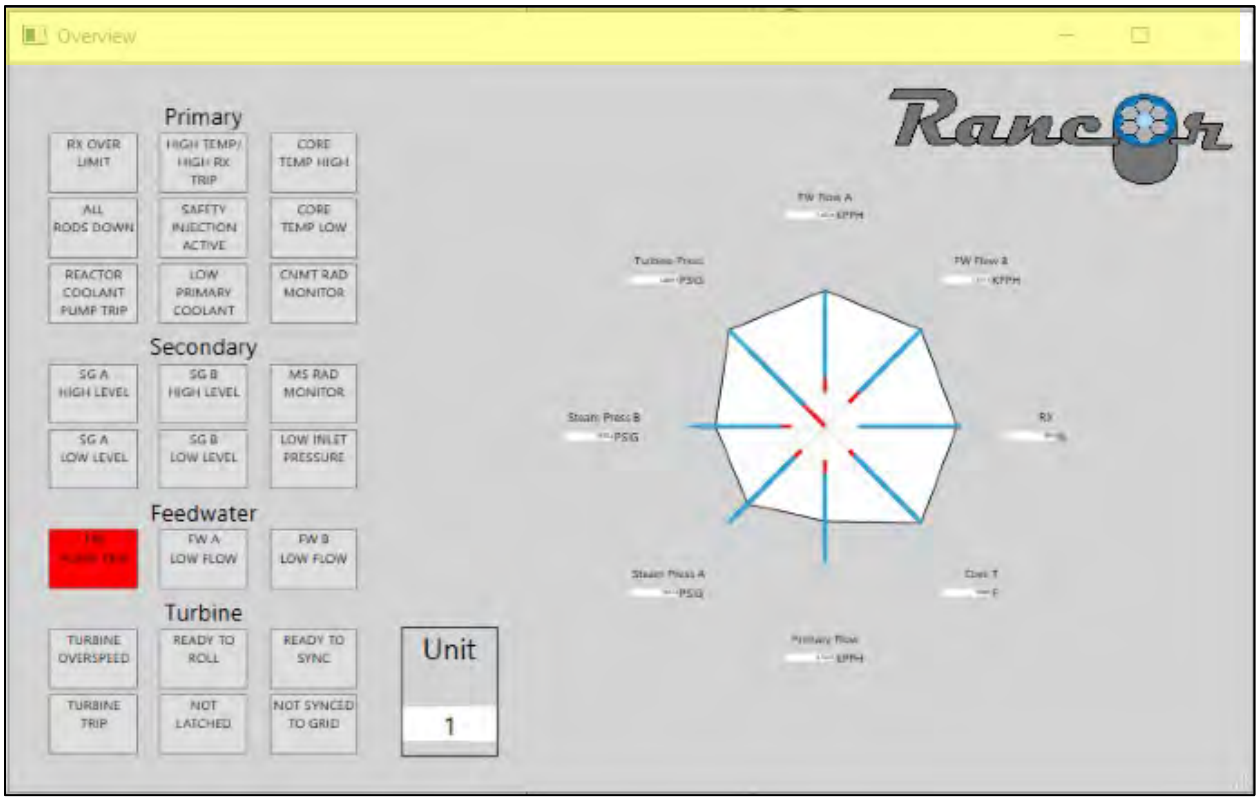


Figure 20. Activation of adjusting the windows

As shown in Figure 21 and Figure 22, when the simulator is paused, the screen window will dim and deactivate. The time is also recorded in the execution window.

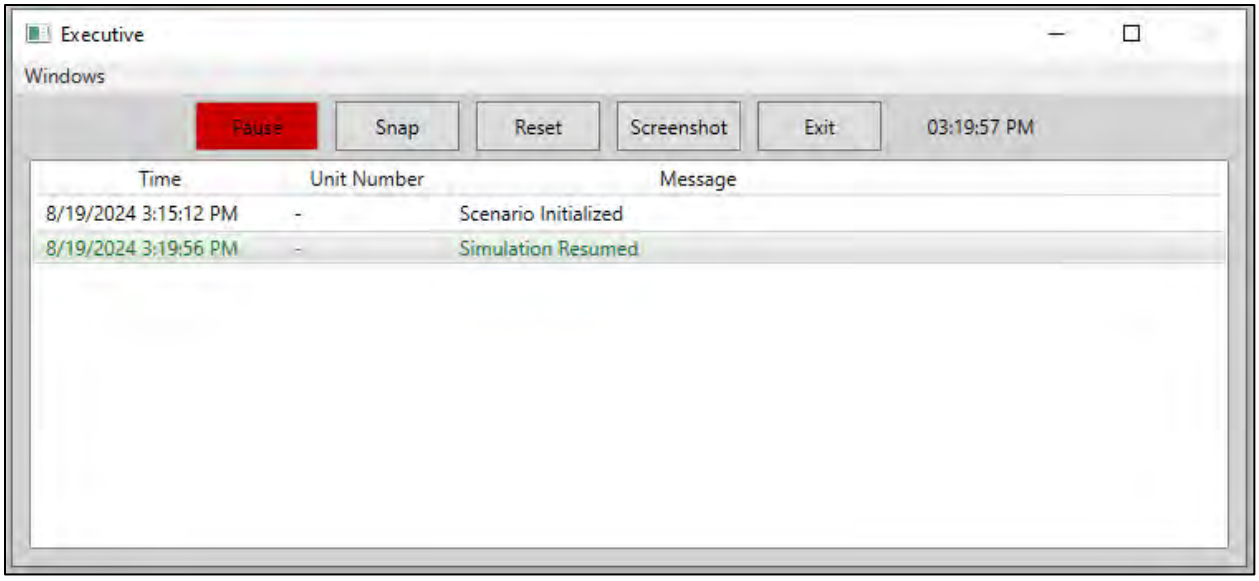


Figure 21. The execution window for Pause the simulator

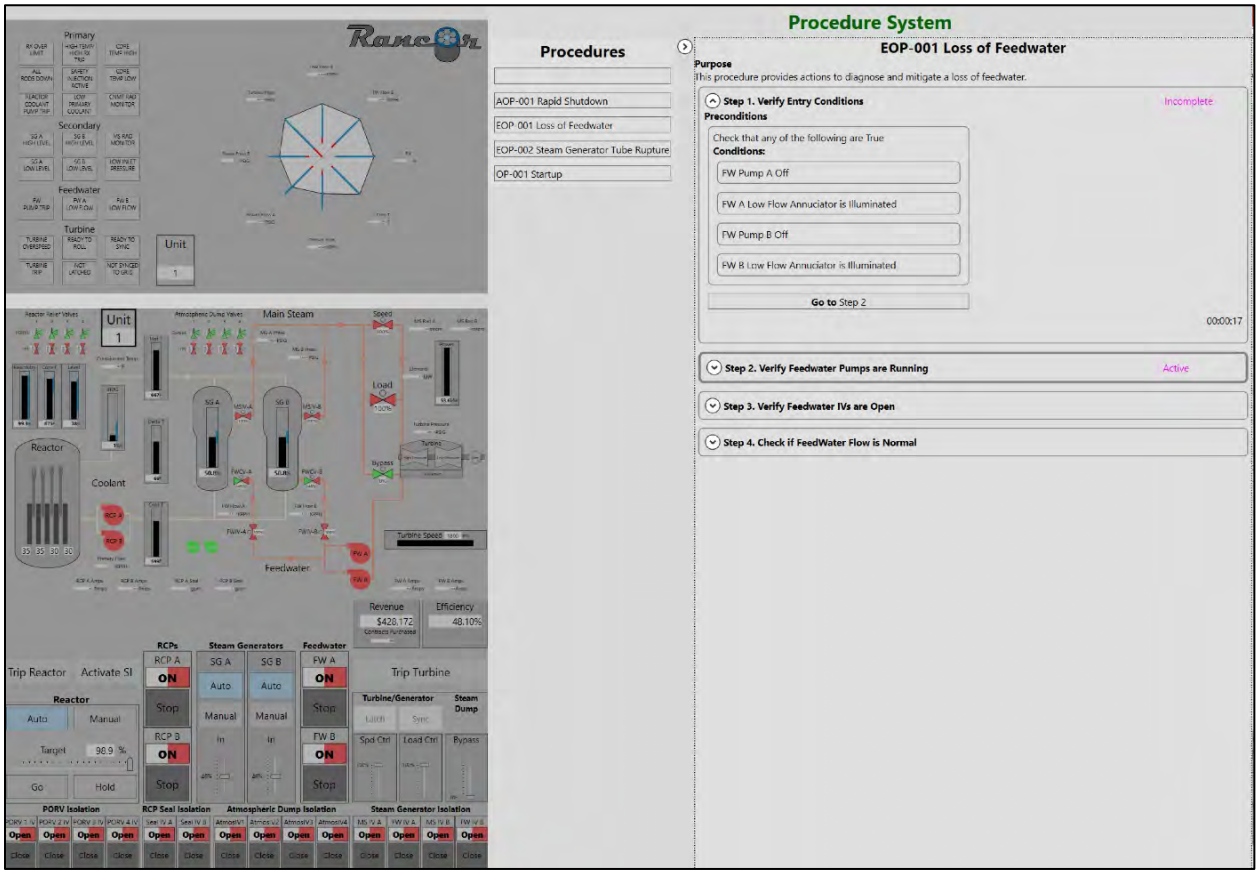


Figure 22. Rancor simulator windows for Pause the simulator

3.3.3 Run Simulator

Once the simulator has been initialized, which entails launching the application and loading a scenario, the simulator waits in a paused mode. Typically, the experimenter provides contextual information about the scenario and any specific instructions as called for in the protocol. The experimenter either selects run to begin the simulation or instructs the participant to select run. Once in Run, the simulator begins to cycle through its updating calculations and the controls become live to accept user input. Scenarios may be defined to trigger specific faults to invoke abnormal operations or may simply include preset initial conditions to train or test normal operations.

3.3.3.1 Alarms/Overview

The operator can check the alarms in the overview window (see Figure 23). If there are no alarms, the color of alarm panel remains grey. However, if a change in a major parameter such as core temperature, primary coolant flow, or reactivity exceeds the setpoint or causes a trip, the alarms are activated with the color changed to red. When the condition is dismissed, the alarm is de-activated and changes to a slightly darker grey. On the other hand, if a safety injection (SI) alarm is activated, it is displayed as a yellow alarm so that the operator can check it more carefully.

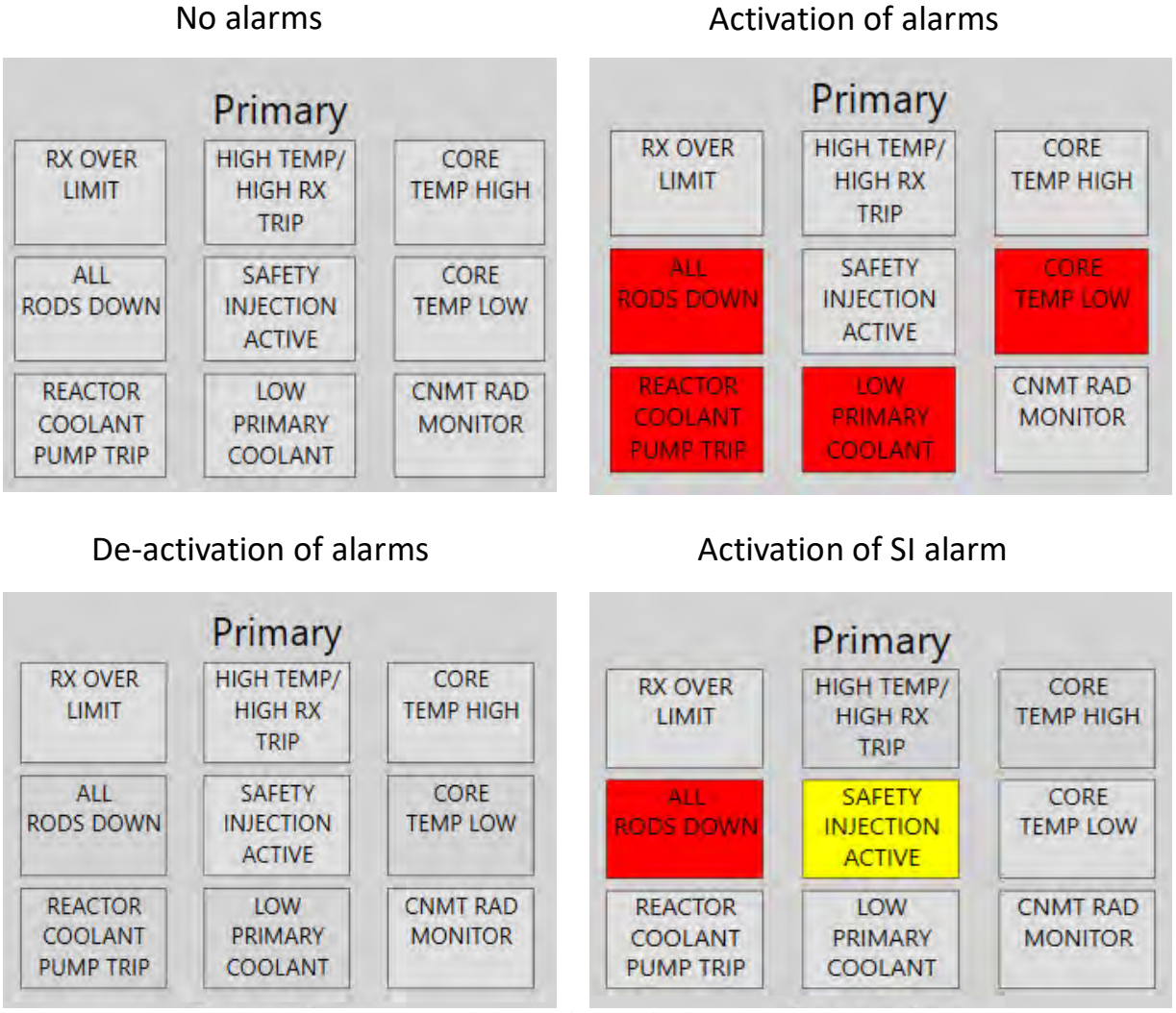


Figure 23. The color of alarms for the primary system

Similarly, the secondary system such as steam generator and feedwater system will also activate the alarms with the color changed to red if the variable exceeds the alarm setpoint. There are alarms such as steam generator high level, low level, low pressure, main steam line radiation monitor, feedwater low flow, and feedwater pump alarms as shown in Figure 24.

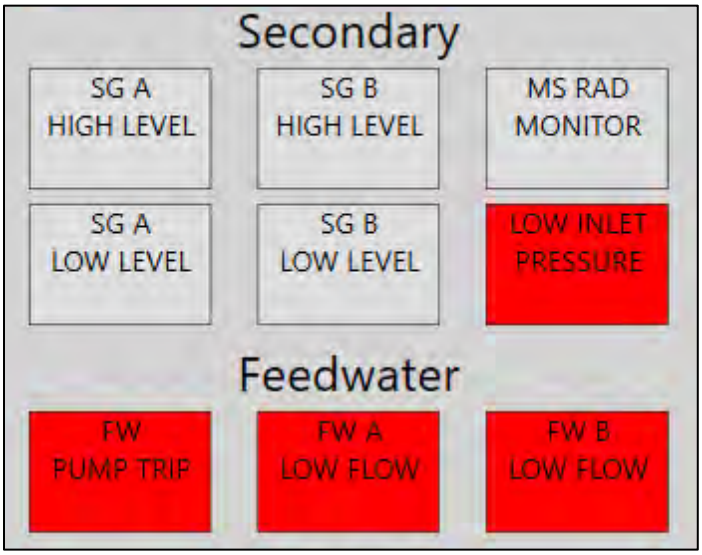


Figure 24. The color of alarms for the secondary system

In the turbine system alarm (see Figure 25), if a turbine overspeed or turbine trip occurs, a red alarm will be activated. However, in relation to Latch or Synch, a green alarm will be generated, which means that the condition is not met. When the Latch or Synch condition is met, the green alarm will disappear and change to a slightly dark grey alarm. Then, the operator can try to latch the turbine or synchronize it to the grid.

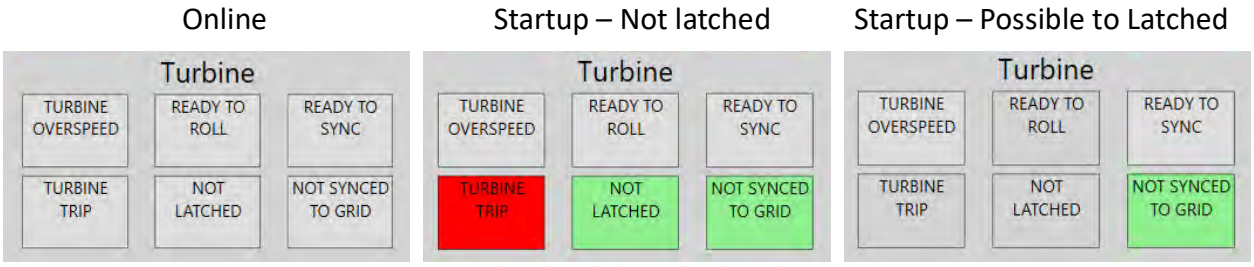


Figure 25. The color of alarms for the turbine system

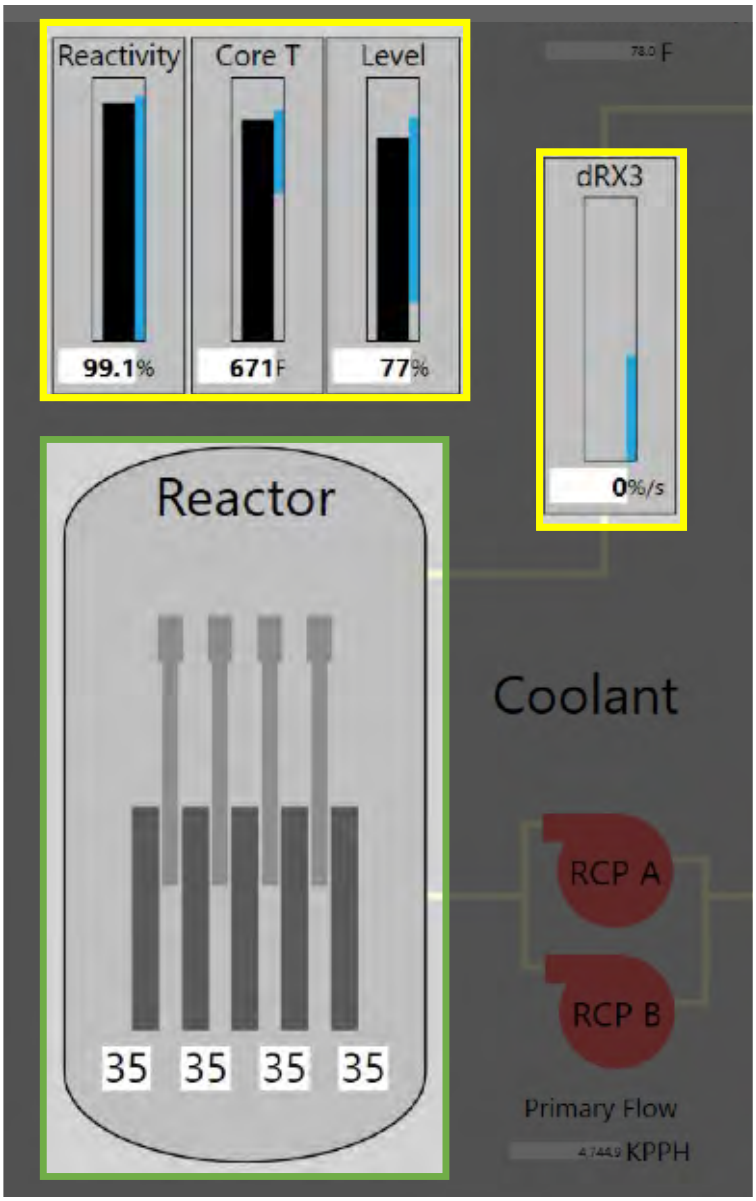


Figure 26. Reactor core in the P&ID window

3.3.3.2 P&ID

This panel includes important components and systems with primary parameters. First, it has the reactor core. This simulates where fission happens, which generates a large amount of heat. Importantly, Rancor has three variables to represent the state of the reactor core as shown in Figure 26. The reactivity controls the heat generation rate, whereby the higher the reactivity, the faster the core temperature increases. The core temperature refers to the temperature of the reactor core. The last indicator on the left side is the core level. Another indicator relevant to the reactor core is the dRX3, which measures the reactivity change rate in the reactor core.

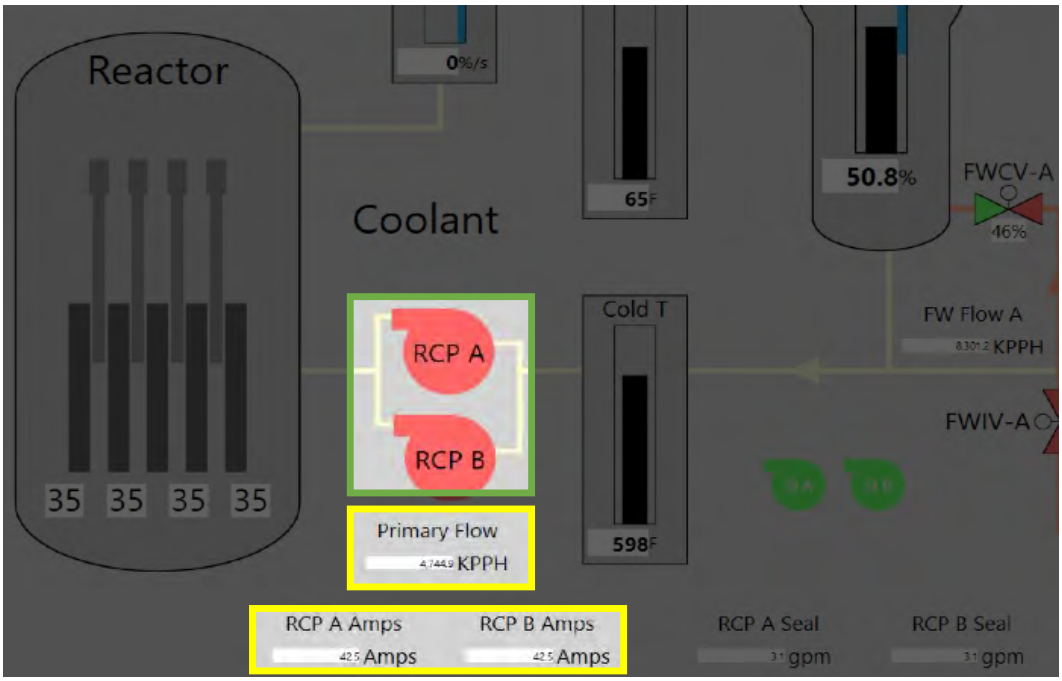


Figure 27. Reactor coolant pump in the P&ID window

Figure 27 shows the reactor coolant pump (RCP) system that pumps water to the reactor core via two pumps. There are indicators for pump status (i.e., on or off, as determined by color), primary flow rate of coolant, and the of the pump electrical amperage.

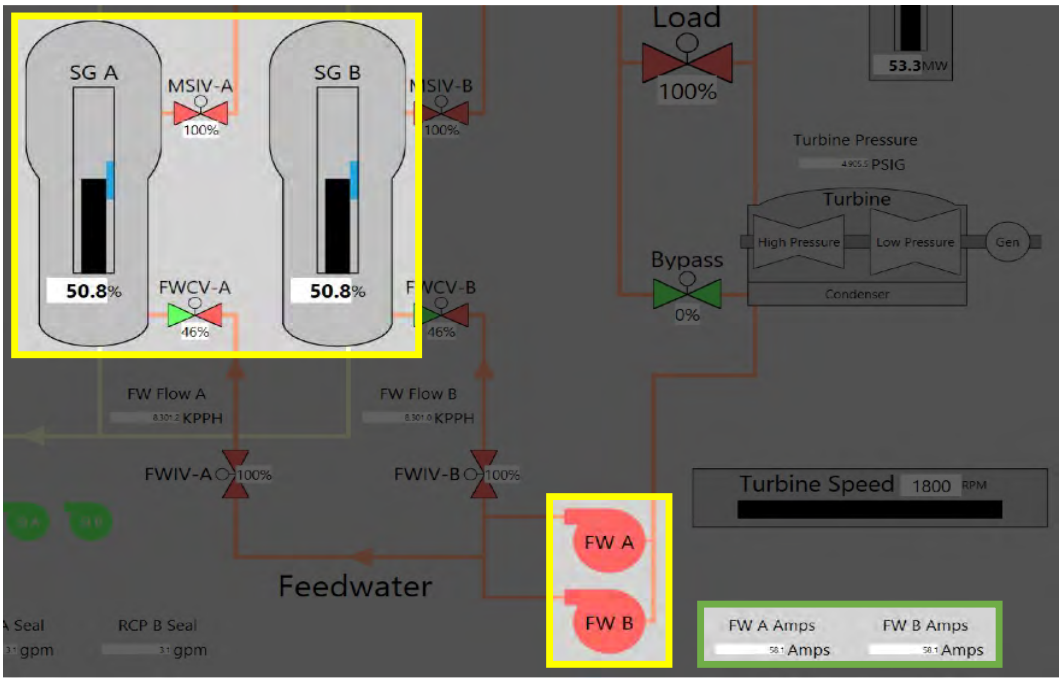


Figure 28. Steam generator in P&ID window

Figure 28 represents the steam generator where the heat exchange happens. Feedwater (FW) pumps help return the condensed water to the steam generator (SG).

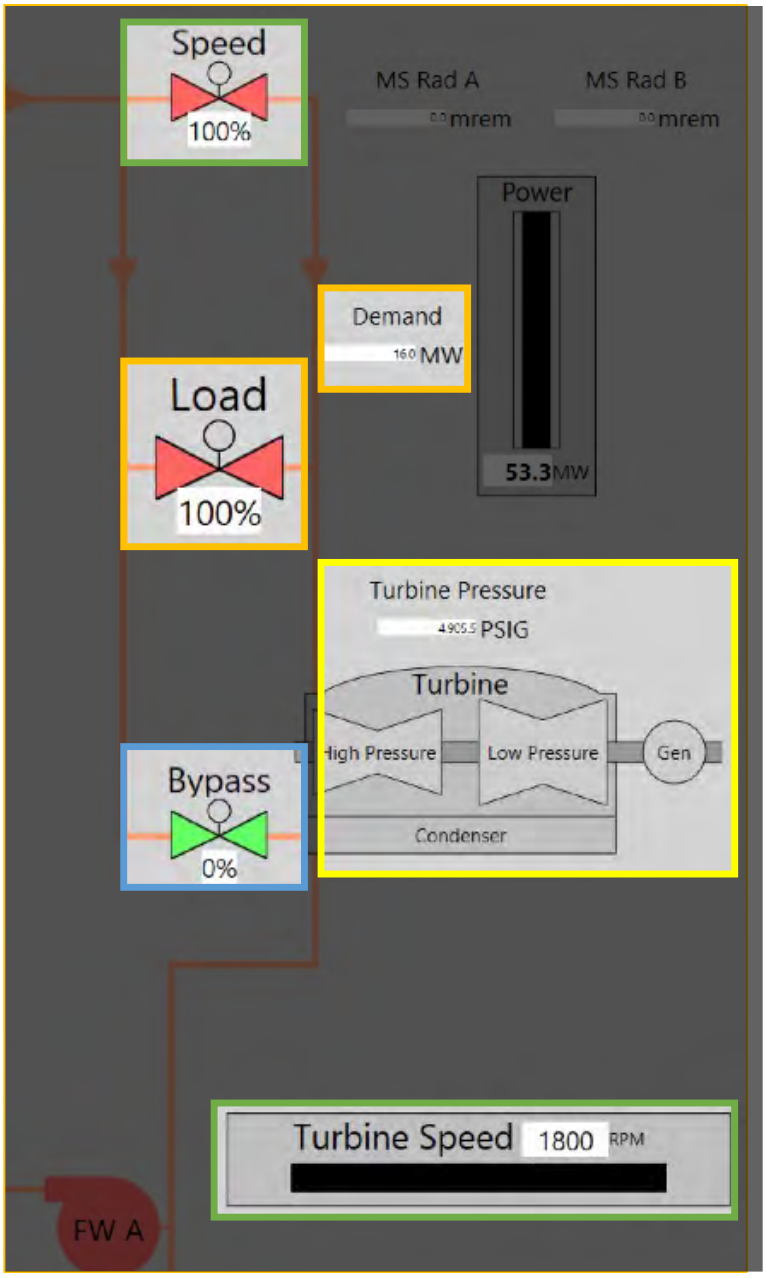


Figure 29. Turbine generator in the P&ID window

The steam spins the turbine to generate electricity. In Rancor, the operator has three valves to represent the state of the turbine (see Figure 29). The top valve is the speed valve, which controls of how fast operator wants the turbine to rotate. The turbine speed depicted in the bottom right corner of the window shows the current turbine rotation speed. The middle valve is the load valve, and it indicates the maximum power the turbine can produce. In the upper right side is the power demand indicator. The operator can adjust the turbine load valve to follow the demand. The bottom valve is the bypass valve, which releases steam to the condenser. If the valve is open, this can help drop the turbine pressure and core temperature.

3.3.3.3 Controls

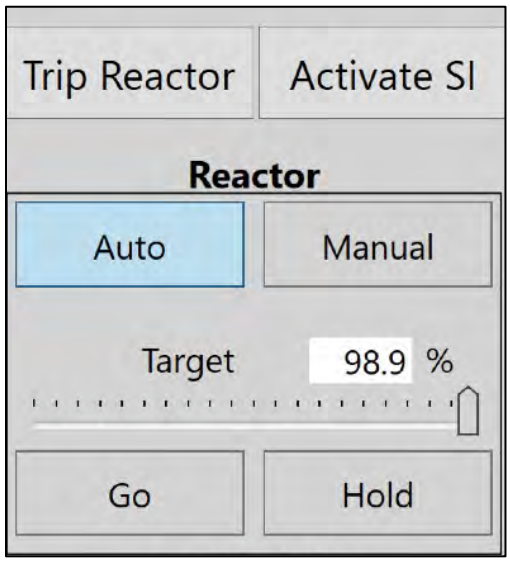


Figure 30. The reactivity controller

The controller in Figure 30 allows the operator to control the reactivity in the reactor core. The operator can transfer the auto mode to manual mode and adjust the target level of the reactivity. The target can be adjusted by moving the slider to the desired target. After adjusting the target, the controller will be executed when the 'Go' button is pressed. The operator can also trip the reactor or activate the SI system in this controller.

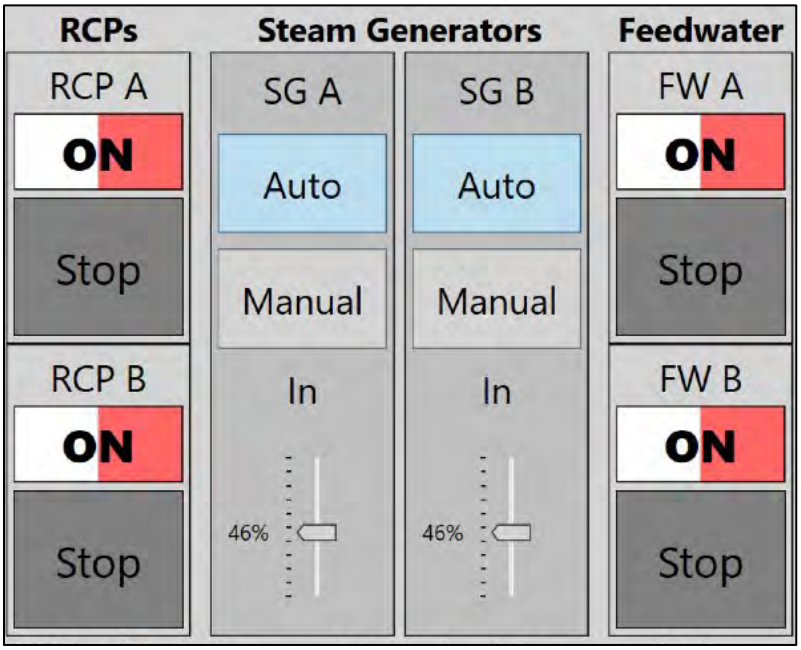


Figure 31. The controller of running pumps and steam generators in auto control mode

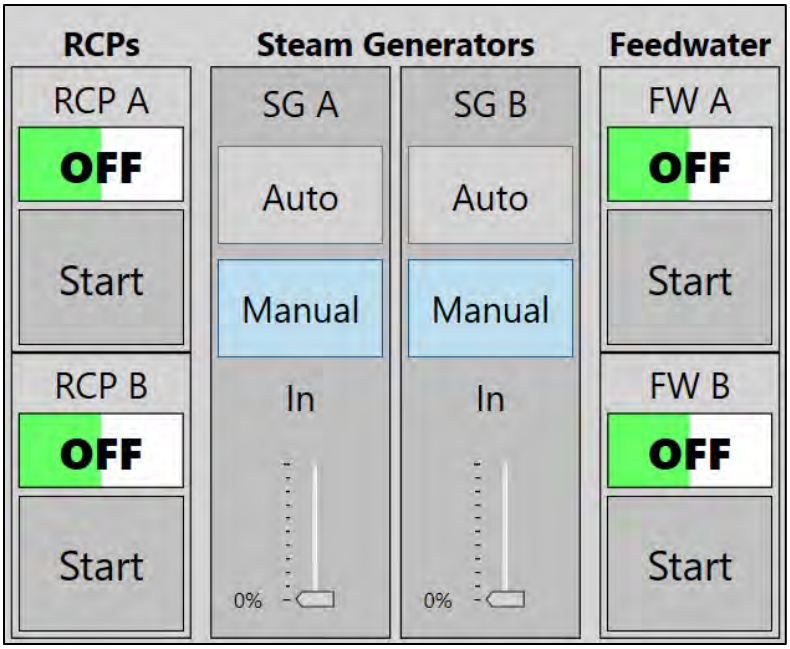


Figure 32. The controller of stopped pumps and steam generators in manual control mode

The controller in Figure 31 and Figure 32 allows the reactor coolant pumps and the feedwater pumps to run or stop. Additionally, it can control the level of the steam generator with auto or manual mode. The color of the pump controller in Figure 31 is written as ON in red, which means that the pump is currently running. The pump can be stopped by pressing the 'Stop' button below. When the pump is stopped, it is written as OFF in green, and the 'Start' control is activated below, as shown in Figure 32. The red-green color dichotomy may seem counterintuitive, but per nuclear conventions, green means ‘ready to go,’ while red means ‘activated or energized.’ The steam generator control is indicated in sky blue in the set control mode. In manual control mode, the slider can be moved to set the desired level of each steam generator.

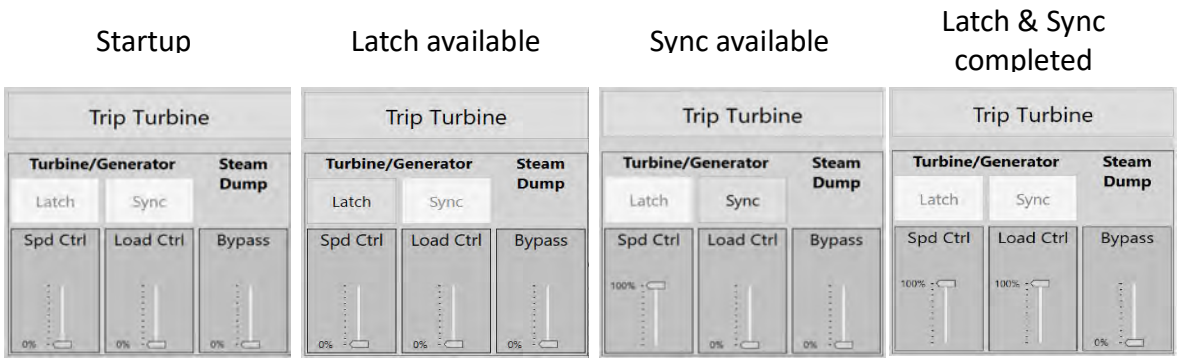


Figure 33. The turbine generator controller across four plant states

The controller shown in Figure 33 is the turbine control. The speed valve controller (Spd Ctrl) is for the turbine rotation speed, the load valve controller (Load Ctrl) sets the maximum power the turbine can generate, and the bypass valve controller (Bypass) helps reduce turbine pressure and core temperature. The first state depicted in Figure 33 is the startup, where no operation has been performed. The second state is where Latch is possible, and the Latch button is activated. The third state is the state where Latch is performed, the speed control valve is opened to 100%, and the Synch condition is satisfied. The last

state is where Latch and Synch are completed, and the load is also increased to 100%. If it is needed, the operator can trip the turbine using this controller.

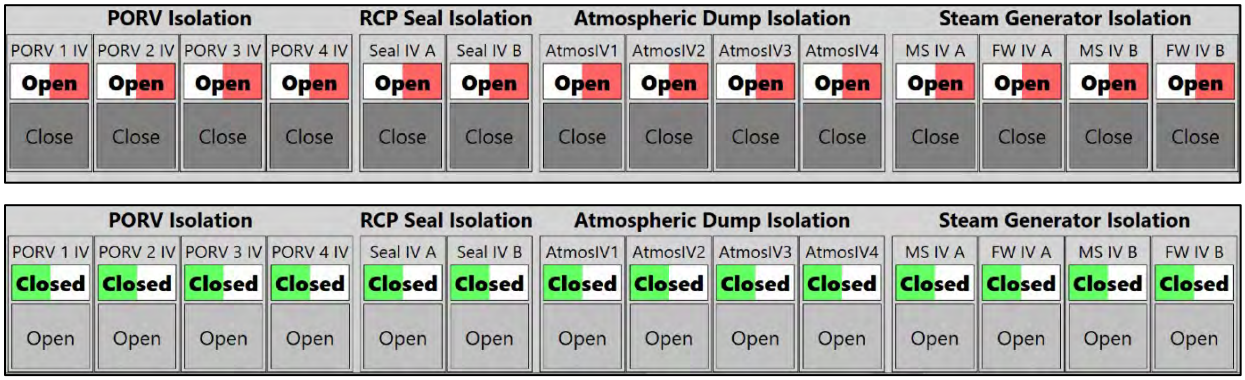


Figure 34. The valve controller

The controller in Figure 34 allows the operator to open or close the valves such as pilot operated relief valve (PORV), reactor coolant pump (RCP) seal isolation valves, atmospheric dump isolation valves, and steam generator isolation valves. The open valve is indicated in red and has the word 'Open' written on it. The operator can close the valve with the 'Close' button below. Valves that are indicated in green and have the word 'Closed' are closed, and the operator can open them with the 'Open' button below the indicators.

3.3.3.4 Computer-based Procedures

Computer-based procedures include operating procedures for various scenarios. Clicking on the appropriate procedure on the left in Figure 35vwill display the procedure in the right column as shown in Figure 36. The operator can click the arrow next to the step to get a drop-down window, or they can collapse the description for the step.

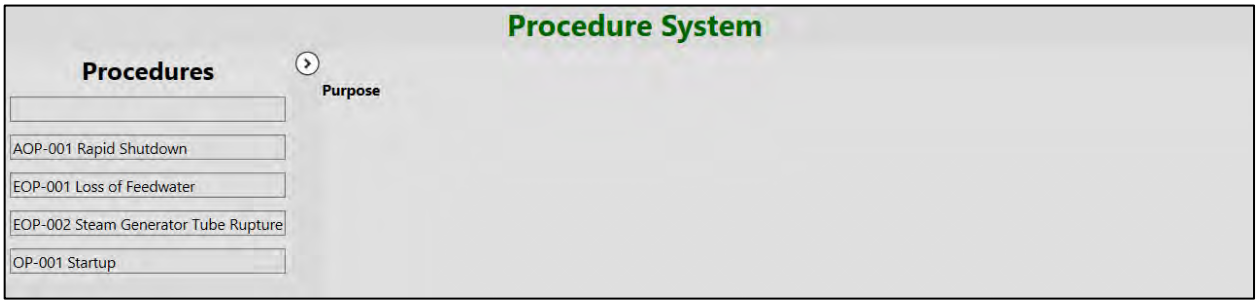


Figure 35. The computer-based procedures of Rancor simulator

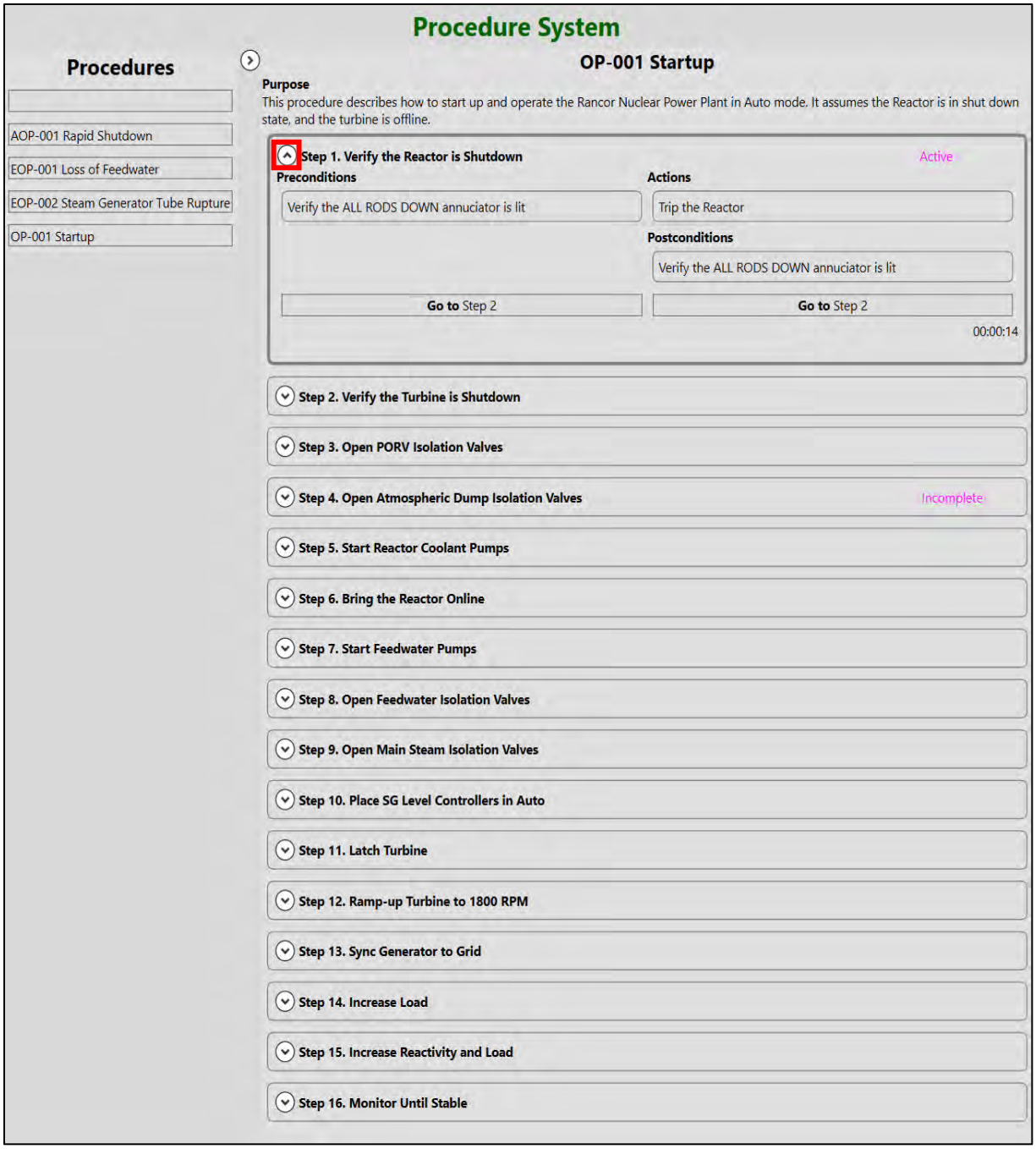


Figure 36. Steps for the startup procedure

The operator should follow the procedure step by step. When the relevant step is clicked, it is marked as 'Active' in the upper right corner. If the 'Go to Step' button is not clicked in each step, 'Incomplete' is displayed (e.g., Step 2 in Figure 37), and if the appropriate action is performed, 'Completed' is displayed (e.g., Step 3 in the Figure 37).

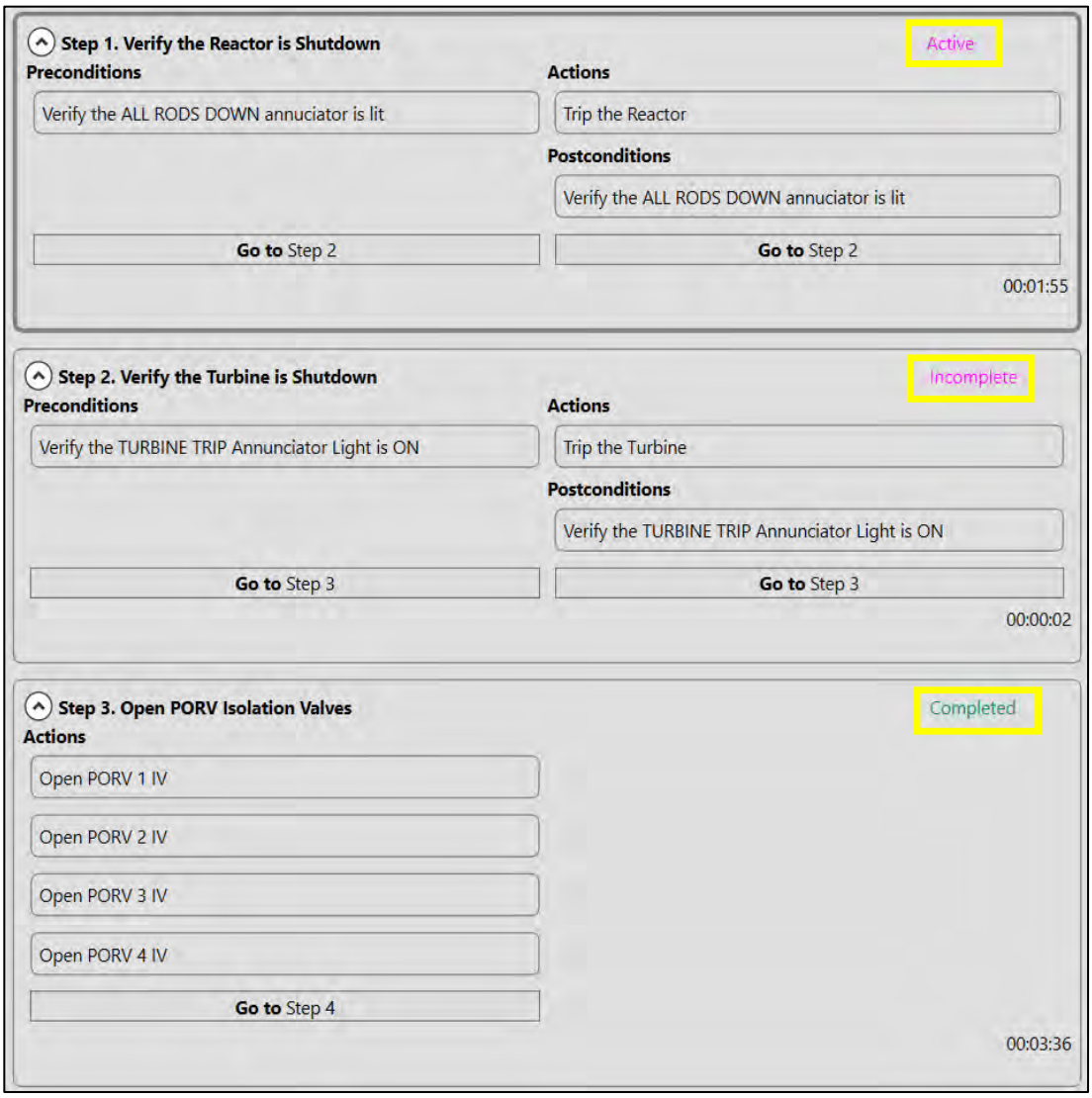


Figure 37. Completion of each procedure step

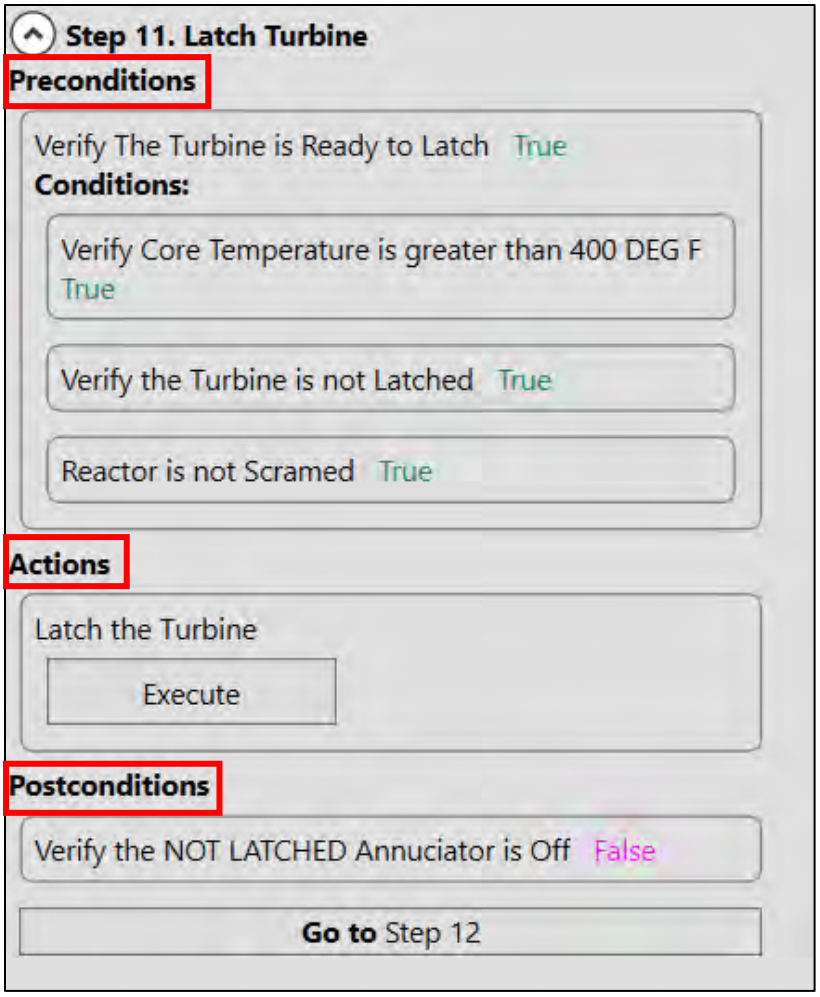


Figure 38. The configuration of each step in the procedure

Each step can have substeps including preconditions, actions, postconditions (see Figure 38). The operator needs to verify the precondition before the action. If the condition is met, the operator should perform the indicated actions. After the action, the operator should verify any postconditions.

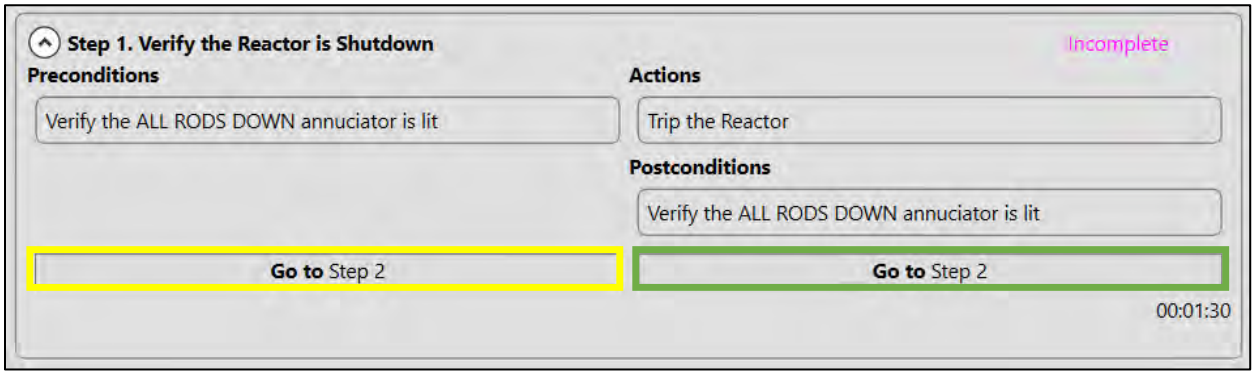


Figure 39. The way to follow a two-column procedure

It's important to note that for many procedures, the operator has two columns as depicted in Figure 39. In the first column, there are Preconditions, and in the second column there are Actions and Postconditions. The operator performs actions in the left column. If the lefthand action cannot be completed, then they perform the actions in the righthand column. In the step shown, the operator needs to verify ALL RODS DOWN ANNUNCIATOR IS LIT in order to go to Step 2. However, if the condition is not met, then the operator should move to the second column on the right to trip the reactor. Then, the operator can verify the ALL RODS DOWN ANNUCIATOR IS LIT and click the "Go to step 2" in the second column.

3.4 New Features of Rancor Version 3

3.4.1 Server-Client Architecture

Previous versions of Rancor could run multiple units but only running as a single application. This limited the ability of other applications or services to communicate with Rancor and did not allow Rancor instances running on separate machines to share common information.

In this version of Rancor, a Web API server and client has been integrated into Rancor as schematically shown in Figure 40. The Rancor executable application contains both the server and the client. When Rancor starts, it references a WebAPI.ini file to identify whether it should run as the server or run as the client. When it runs as the server it runs the simulation update loops to model the plant. When run as a client, Rancor obtains the state of the plant and units from the Web API. When operators perform actions, those actions are sent to the simulation model via the Web API. Both the Rancor server and client can be configured to display the HSI windows.

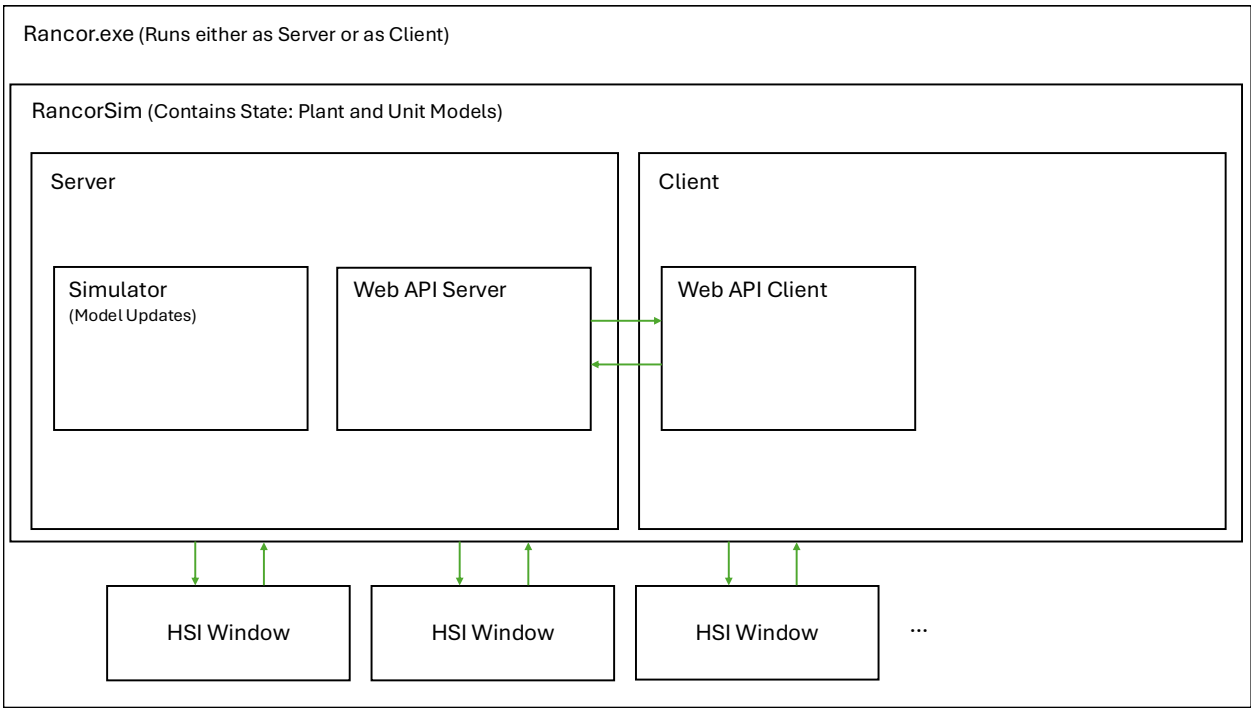


Figure 40. Rancor.exe contains both the server and the client

The server-client architecture provides additional flexibility for external applications to communicate with Rancor as shown in Figure 41. These external applications could be used to display alternate HSIs or for co-simulation purposes with external models of systems.

Figure 41. Rancor HSI running in client mode with the client is executing actions through the API

3.4.2 Rancor Web API

The Rancor Web API is designed to interact with the Rancor Simulator. It provides various endpoints for managing unit states, simulator states, logging, and retrieving data in different formats (e.g., JSON and MessagePack). Table 3 documents the endpoints that are available to a developer or researcher.

Table 3. Rancor Web API endpoints

Unit State Management

Get Unit State (JSON)

- Endpoint: GET /api/units/{unitNumber:int}
- Description: Retrieves the current state of the specified unit in JSON format.
- Parameters:

  unitNumber (int): The unit number to retrieve the state for.
- Response: A JSON object representing the state of the specified unit.

Set Unit Property

- Endpoint: POST /api/units/{unitNumber:int}/setproperty
- Description: Sets the value of a specified property for a specific unit.
- Parameters:

  unitNumber (int): The unit number to set the property for.

  propertyName (string, query): The name of the property to set.

  value (string, query): The value to assign to the property.
- Response: A success status if the property is set successfully.

Execute Unit Method

- Endpoint: POST /api/units/{unitNumber:int}/executemethod
- Description: Executes a method on the view model of a specific unit by name.
- Parameters:

  unitNumber (int): The unit number to execute the method on.

  methodName (string, query): The name of the method to execute.
- Response: A success status if the method is executed successfully.

Simulator State Management

Set Simulator State

- Endpoint: POST /api/rancorsim/setstate/{simstate}
- Description: Sets the state of the simulator to either "freeze" or "run."
- Parameters:

  simstate (string, query): The state to set the simulator to. Must be either "freeze" or "run."
- Response: A success status if the state is set successfully.

Snap Simulator

- Endpoint: POST /api/rancorsim/snap
- Description: Saves the current state of the simulator to an initial conditions (IC) file.
- Response: A success status if the state is saved successfully.

Exit Simulator

- Endpoint: POST /api/rancorsim/exit/{pinCode:int}
- Description: Shuts down the simulator. A valid PIN code must be provided for security.
- Parameters:

  pinCode (int, query): The PIN code to authorize the simulator shutdown.
- Response: A success status if the simulator is shut down successfully.

Logging

Write to Log

- Endpoint: POST /log/baremetal
- Description: Logs messages directly to log files with preformatted text.
- Parameters:

  text (string, body): The text to log.

  filetype (string, query): The type of log file to write to.
- Response: A success status if the message is logged successfully.

Binary Data (MessagePack) Handling

MessagePack is an efficient binary serialization format that allows data to be serialized and deserialized in a compact, byte-efficient way. It is designed to be faster and more space-efficient than text-based formats like JSON, while still being versatile enough to support a wide range of data types.

Get Unit State (MessagePack)

- Endpoint: GET /messagepack/units/{unitNumber:int}
- Description: Retrieves the current state of the specified unit in MessagePack serialized binary format.
- Parameters:

  unitNumber (int): The unit number to retrieve the state for.
- Response: A binary file containing the serialized state of the specified unit.

Get Simulator State (MessagePack)

- Endpoint: GET /messagepack/rancorsim
- Description: Retrieves the current state of the simulator in MessagePack serialized binary format.
- Response: A binary file containing the serialized state of the simulator.

3.4.3 Revised Window Layouts

Previous versions of Rancor had limited window layout configurability. Rancor was limited to specifying the visibility and placement of a small set of windows as described in the layout tool in Section 3.3.2. For single-unit configuration, the built in layout tool is sufficient. However, for the multi-unit configuration only one indication and one control window could be displayed with a unit selector, making it impossible to view the HSIs for multiple units even if screen real estate was available. Therefore, a more complete layout configuration is possible with configuration files.

The revised window layouts allow for significantly increased flexibility. With multiple unit configurations each unit could have dedicated windows for indications and controls. Or the units could have dedicated windows for indications and a shared control window. The revised window layouts in conjunction with the server-client architecture allow for multi-unit multi-operator control room simulators to be configured with each operator having their own workstation to interact with the plant.

Figure 42 shows an example of a possible configuration in which two reactor units are displayed, each with the ability to toggle between different target units. In contrast, Figure 43 shows a similar two-unit configuration but with each window locked to a specific unit. Lastly, the revised window layout provides increased flexibility for extending the catalog and styles of available windows. Figure 44 depicts a restyled Rancor HSI made possible by selecting a different pre-defined style in the configuration file. Table 4 shows configuration files associated with the layouts of Figure 42 and Figure 43, respectively.

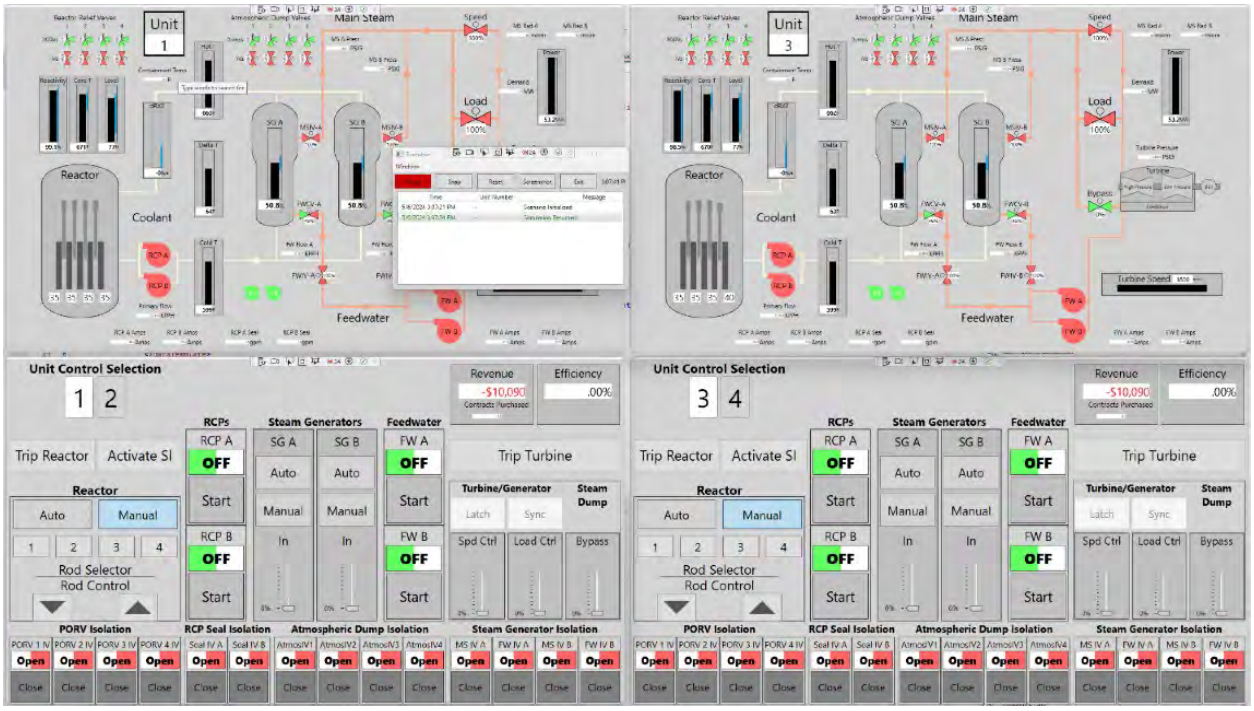


Figure 42. Two window groups with units 1 and 2 on the left and units 3 and 4 on the right

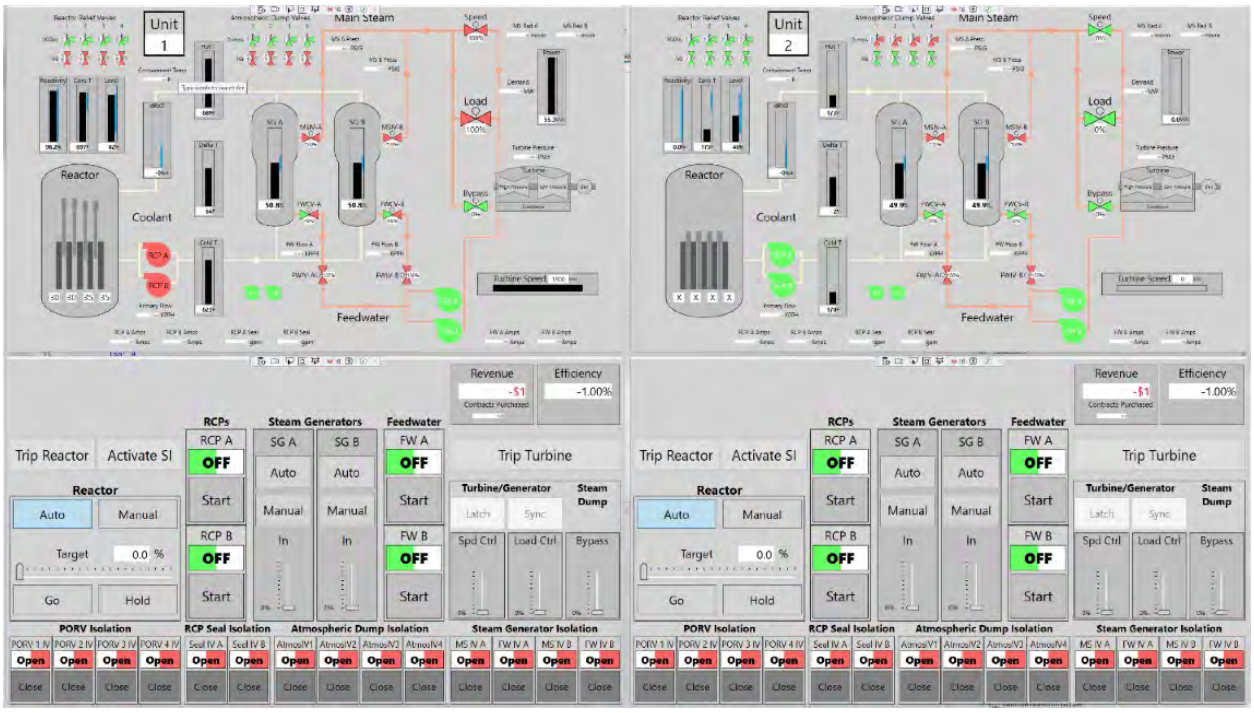


Figure 43. Two window groups each with a single unit

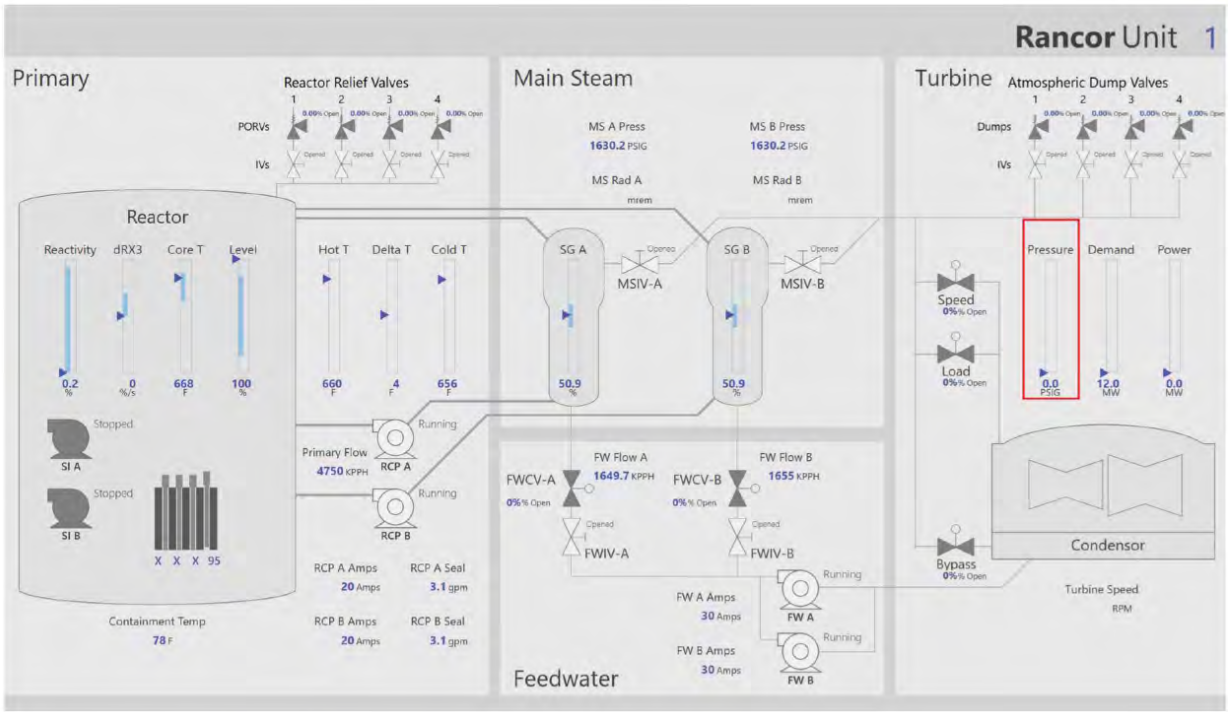


Figure 44. Example of alternate visualization for single unit layout

Table 4. Two configuration files for specifying different multi-unit displays in Rancor Version 3

Two-Unit Display with Selectable Units

```
PlantWindowLayouts:
  MainWindow:
    Left: 0
    Top: 0
    Width: 500
    Height: 300
    Show: true
    Resizable: true

UnitWindowGroups:
  - WindowGroupIdentifier: "Units1_2"
    UnitIdentifiers:
      - 1
      - 2
    WindowLayouts:
      PidWindow:
        Left: 0
        Top: 0
        Width: 1280
        Height: 720
        Show: true
        Resizable: true
      ControlWindow:
        Left: 0
        Top: 720
        Width: 1280
        Height: 720
        Show: true
        Resizable: true
  - WindowGroupIdentifier: "Units3_4"
    UnitIdentifiers:
      - 3
      - 4
    WindowLayouts:
      PidWindow:
        Left: 1280
        Top: 0
        Width: 1280
        Height: 720
        Show: true
        Resizable: true
      ControlWindow:
        Left: 1280
        Top: 720
        Width: 1280
        Height: 720
        Show: true
        Resizable: true
```

Two-Unit Display with Fixed Units

```
PlantWindowLayouts:
  MainWindow:
    Left: 0
    Top: 0
    Width: 500
    Height: 300
    Show: true
    Resizable: true

UnitWindowGroups:
  - WindowGroupIdentifier: "Units1"
    UnitIdentifiers:
      - 1
    WindowLayouts:
      PidWindow:
        Left: 0
        Top: 0
        Width: 1280
        Height: 720
        Show: true
        Resizable: true
      ControlWindow:
        Left: 0
        Top: 720
        Width: 1280
        Height: 720
        Show: true
        Resizable: true
  - WindowGroupIdentifier: "Units2"
    UnitIdentifiers:
      - 2
    WindowLayouts:
      PidWindow:
        Left: 1280
        Top: 0
        Width: 1280
        Height: 720
        Show: true
        Resizable: true
      ControlWindow:
        Left: 1280
        Top: 720
        Width: 1280
        Height: 720
        Show: true
        Resizable: true
```

3.4.4 Revised Implementation of English/SI Units

Previous versions of Rancor had interface for English and international standard (SI) units, e.g., Fahrenheit vs. Celsius temperature scales. The application implemented this by having properties for both units in the model and having separate displays for each. When Rancor launched, it would reference a configuration file to display the SI unit screens or the English unit screens. This has been simplified by building unit awareness into the indications (e.g., BarGauge, NumericIndicator) and binding to a Boolean property that specifies whether SI or English units should be displayed. This eliminates having the SI parameters in the model and manually updating them with the correct values and the second set of displays. This revision also allows SI or English units to be dynamically changed while the application is running through the Executive Units of Measure selection.

3.4.5 Low Level Console

For both server and client instances of Rancor a low level console shell is now provided, as shown in Figure 45. This provides real-time logging of simulator processes such as starting the Web API and client connections. It also logs faults and HUNTER states.

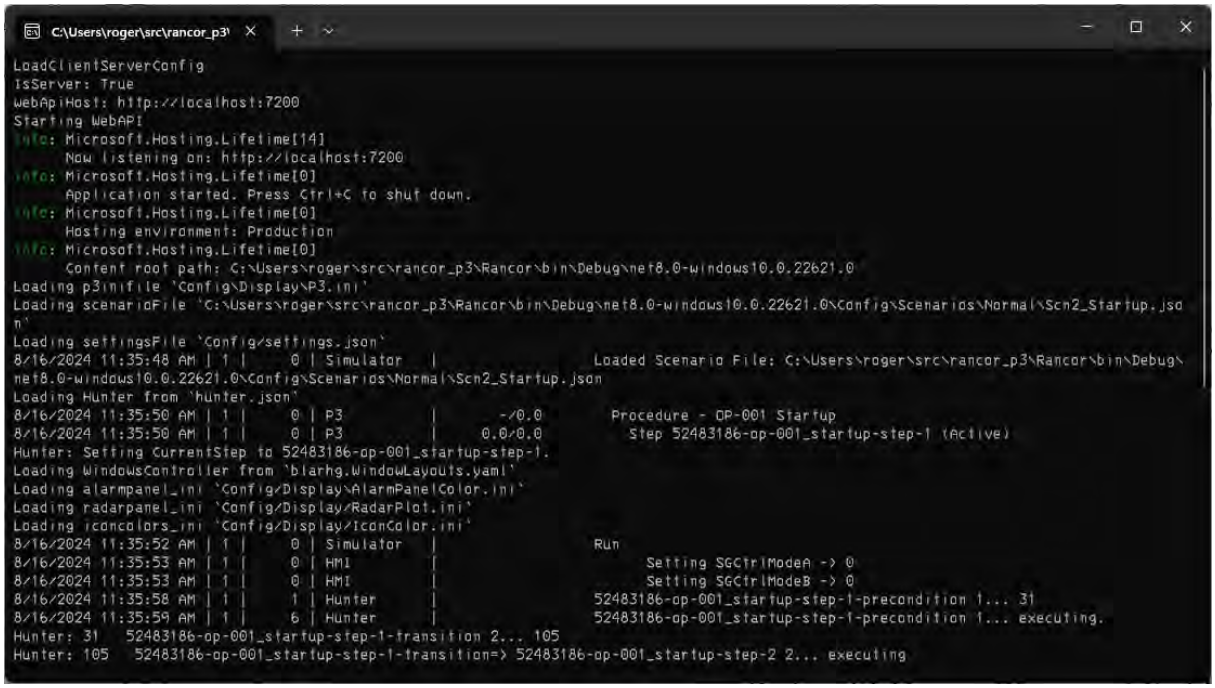


Figure 45. Rancor console

3.4.6 Digital Procedure System

Rancor includes a new Digital Procedure System that was developed to serve as a computer-based procedure system to allow users to navigate and execute prescribed interactions with Rancor. However, the Digital Procedure System is unique in that it incorporates features unlike traditional CBPs such as high resolution task-based data recordings that support experimental evaluations. To understand how this system functions it is important to understand how procedures are used traditionally within existing NPPs. A brief description of procedures in terms of their structures, formats, nomenclature, and usage is necessary to grasp the implementation of the Digital Procedure System within Rancor and has implications for HUNTER. Indubitably, HUNTER's core functionality can be achieved due to logic provided by the strong adherence to procedures observed by the commercial nuclear industry.

NPP control room operations are heavily reliant on procedures to provide prescriptive methods to manipulate the plant across normal and abnormal conditions. Existing U.S light water reactors use single and two-column paper-based procedures. Single-column procedures are the more intuitively understandable variant in that they provide a single course of actions within a well-known set of conditions that do not require the operator to deviate from a set path or sequence of procedure steps. Two-column procedures are more complicated because they contain textually explicit *and* implicit logic contained within the structure of the steps as they are arranged across the two columns. The two-column format is commonly adopted for pressurized water reactors.

The two-column format contains a Response Obtained (RO) and Response Not Obtained (RNO) column. Two-column procedures account for alternative paths applicable to different plant states (see Figure 46). The left column is the Response Obtained and the right column is the RNO. Operators evaluate the logic contained within the left column steps to determine if it is met or can be met with the prescribed actions. If the step logic is upheld by the plant state to yield the response obtained, the operator continues down the left column with the next sequential step. If the response is not obtained, the operator moves into the adjacent right column step, which can take on several different forms. In its simplest instantiation, the right column contains an action that allows the operator to achieve the desired response. After performing any actions and evaluating the RNO as successfully completed, the operator can then return to the next step prescribed in the left Response Obtained column. The right column can also contain a path prescription indicating what step the operator should move to based on the corresponding response obtained failure for that particular step. In this instance, the operator simply navigates to the indicated Response Obtained step. In the most complicated version of a RNO step an action is prescribed, the logic must be evaluated, and then a navigational instruction directs the operator to another procedure step. The Digital Procedure System in Rancor adopts this two-column format.

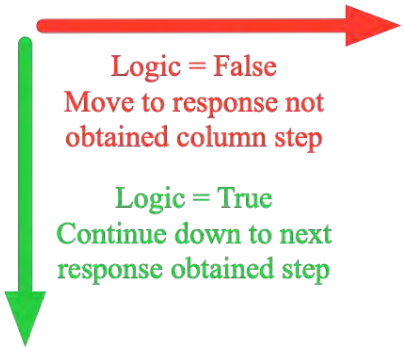
Response Obtained	**Response Not Obtained**
1. Step with instructions that may take the form of diagnosis **OR** actions **OR** both a. Logic statement to evaluate a plant parameter based on one or more criteria b. Action to manipulate a plant parameter c. Logic statement to evaluate the result of the action based on one or more plant parameters and criteria  	1. Step with instructions to take corrective action **OR** navigate to a different step in the procedure **OR** navigate to another procedure a. Navigate to… i. Another step within the same procedure ii. Another step within a different procedure b. Logic statement that dictates navigating to… i. Another step within the same procedure ii. Another step within a different procedure c. Action to manipulate a plant parameter [implicit return to next sequence step in response not obtained column] Ii. Explicit Instruction to evaluate the result of the action based on one or more plant parameter criteria and navigate (1) Another step within the same procedure (2) Another step within a different procedure

Figure 46. Two column procedure format depicting procedure instructions including navigational instructions to allow the procedures to be applicable across many different plant states.

The Digital Procedure System in Rancor provides a CBP to support the operators working through scenarios. The term Digital Procedure System is used intentionally here to differentiate this system from a standard CBP. Specifically, this is a research tool intended to collect data while users interact with Rancor. It is first necessary to understand basic CBPs before describing the additional functionality contained with the Digital Procedure System in Rancor.

The term CBP itself is archaic due to the ubiquitous nature of computers. However, the nuclear industry has a heavy reliance on paper-based procedures and, as such, the term effectively distinguishes a dichotic division. Computer-based procedures can be further classified based on the types of functionalities they support. The IEEE-Std-1786 *Guide for Human Factors Applications of Computerized Operating Procedure Systems (COPS) at Nuclear Power Generating Stations and Other Nuclear Facilities* (IEEE, 2021) provides a three-type type hierarchical classification system (see Table 5).

Table 5. Definitions for types of computer-based procedures

Type	Digital Text	Embedded Indicators	Embedded Controls
1	Yes	No	No
2	Yes	Yes	No
3	Yes	Yes	Yes

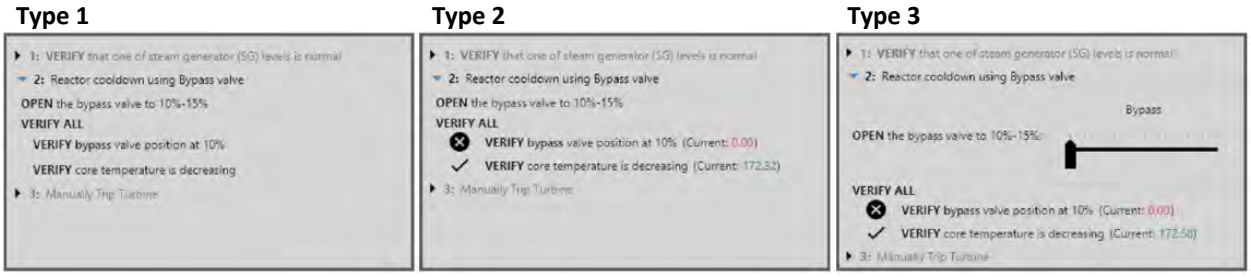


Figure 47. Three types of computer-based procedures in Rancor

Type 1 CBPs are like-for-like replications of paper procedures that introduce place keeping functionality. Type 2 CBPs introduce live parameter values. Type 3 CBPs introduce soft controls that allow the operator to directly act from within the procedure step itself. Using the Digital Procedure System, Rancor supports each of the three types of CBPs as defined in the IEEE-1786 classification. Type 1 is a digital representation of paper procedures, Type 2 is digital procedure with embedded indicators, and Type 3 is digital procedure with embedded soft controls. As shown in Figure 48, the Type 1 procedure contains only textual instructions. The operator must still monitor the parameter in the overview and P&ID displays and take any actions to control the components in the control panel display. Type 2 procedures include the indicator as live data value that updates as the plant state changes (see Figure 49). The operator can now read the logic of step instructions with the live data value embedded and highlighted in a color-coded scheme. The color coding is a form of automatic step resolution. The operator must still control the components outside of the procedure through the control panel display. As mentioned previously, Type 3 procedures build on these prior levels by adding soft controls that allow the operators to take action as prescribed by the step instruction text without the need for the P&ID *or* control panel display (see Figure 50. However, operators should consider the P&ID and controllers when making decisions, since an understanding of the system state beyond the parameters contained in the procedure text ensures the operator has at least attempted to achieve a level of situation awareness as opposed to simple rote procedure following.

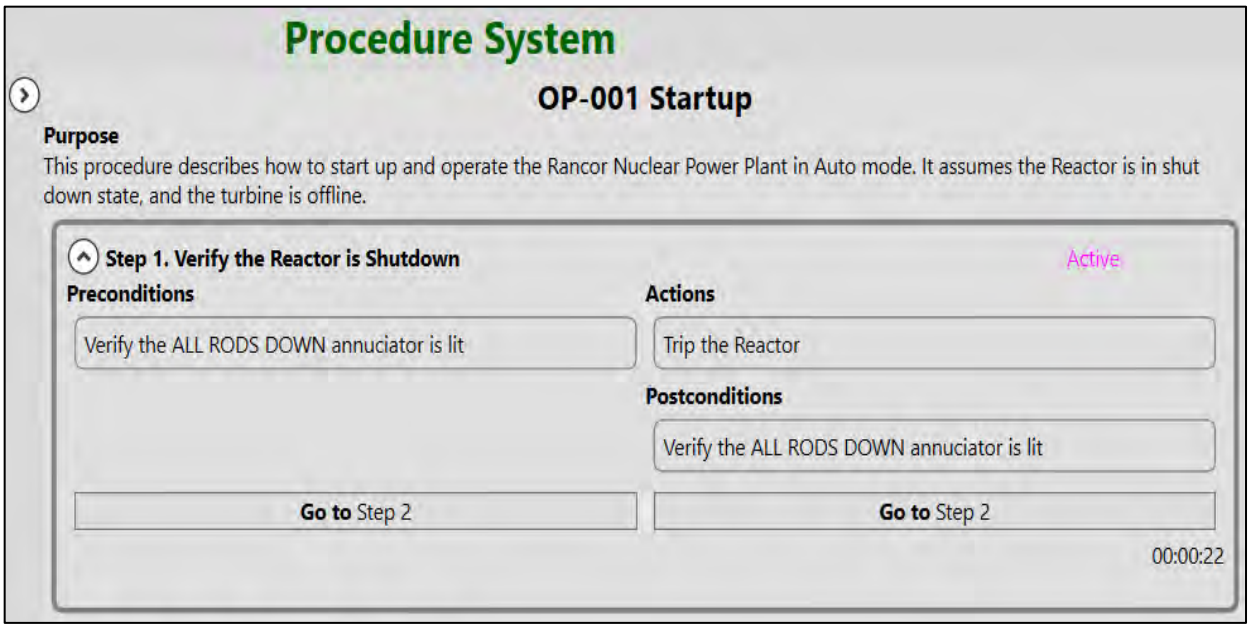


Figure 48. Type 1 computer-based procedure in Rancor

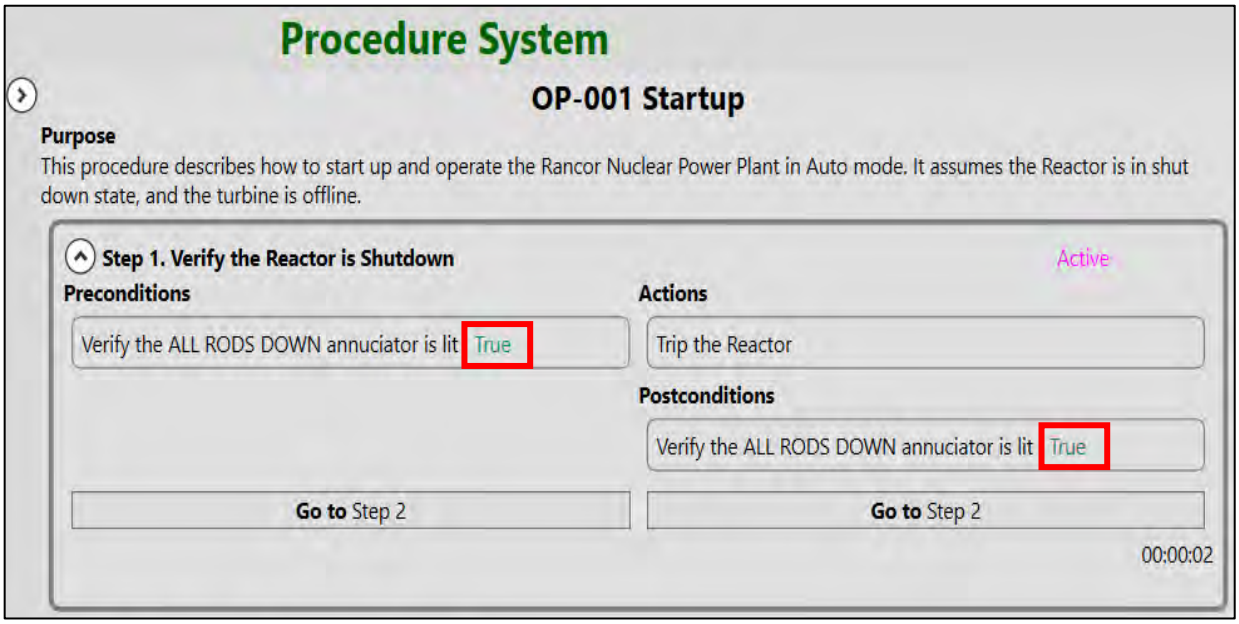


Figure 49. Type 2 computer-based procedure in Rancor

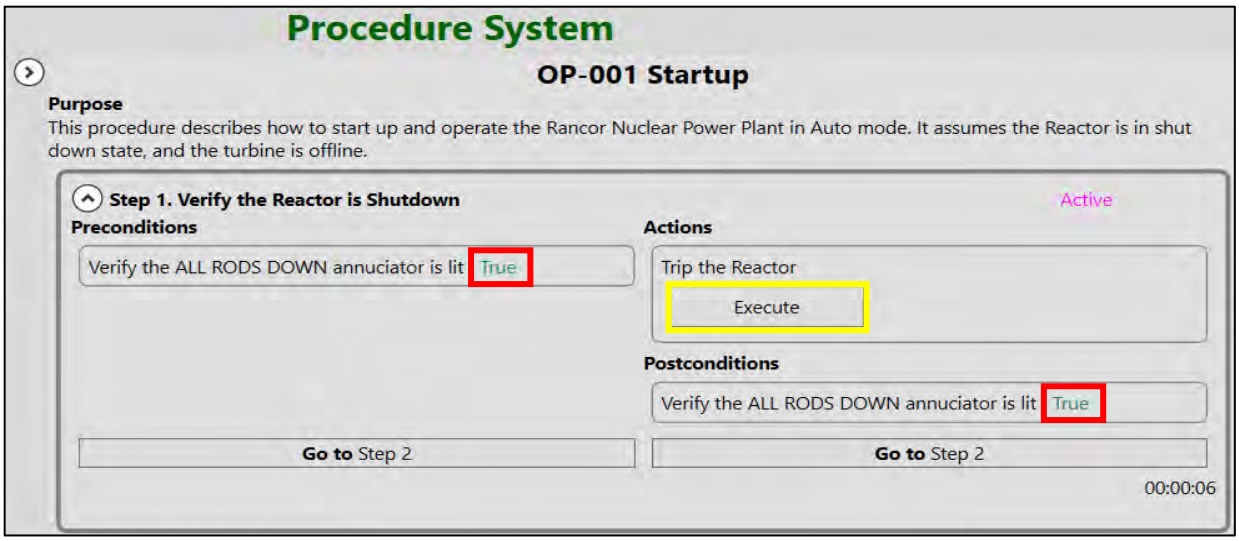


Figure 50. Type 3 computer-based procedure in Rancor

Due to all information being centralized into the Type 3 CBPs, the delineation between the control system and the procedures is blurred such that the procedures could be viewed as an extension of the control system itself. Figure 51 demonstrates how the entire procedure sequence can be executed with a few 'Execute' buttons when using Type 3 CBPs. This poses several issues for change management as procedures are revised at a greater frequency that the control systems. More research is needed to identify how to manage these changes to ensure fundamental control system functions are unchanged for safety critical aspects. Furthermore, much of this automation servers as a path limiter to funnel the operator towards the best course of action.

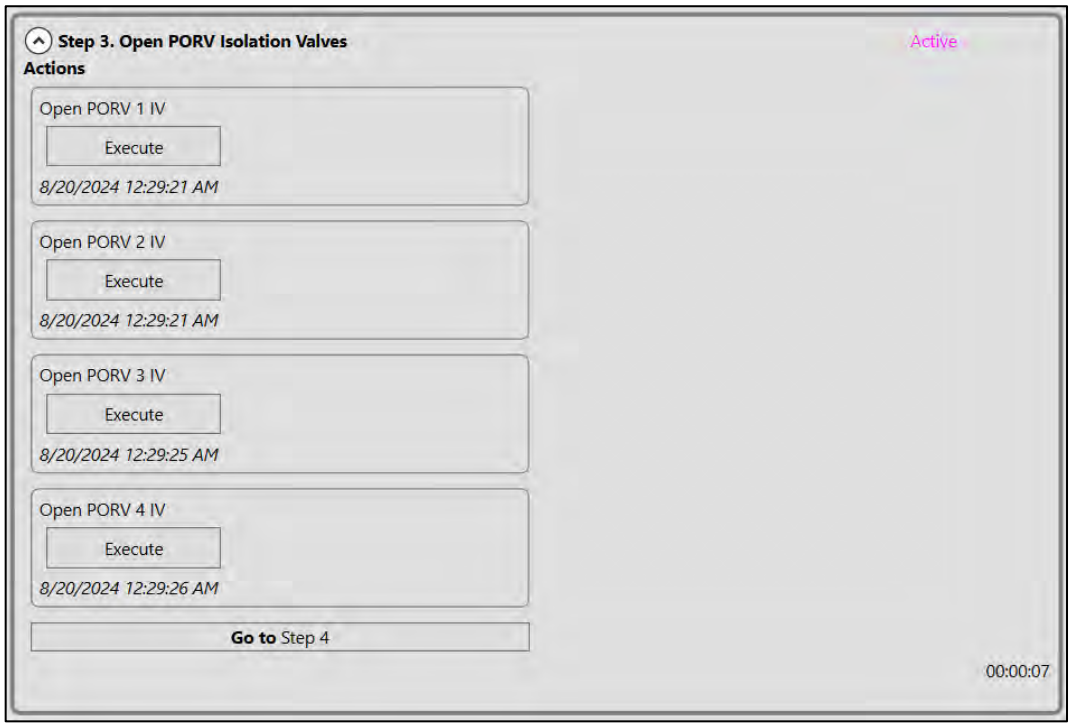


Figure 51. Execution of type 3 computer-based procedure in Rancor

Operators are uniquely capable of handling novel situations outside of the design considerations for the system and must retain the autonomy to maneuver through computer-based procedures when they fail to account for the plant state. Research is needed for developing the best methods to let operators "redline" the procedures, i.e. take a different course of action and record it in the system. As an experimental tool, the digital procedures provide the capability to toggle these path limiters on and off to determine how operators navigate when they are present and absent, which provides one avenue of exploration to identify how operators contend with deviations from prescribed actions as well as how they forge their own path when the path limiters are relaxed. The intent is to provide a means to identify the appropriate level of path limiters to prevent the uniquely manifesting guardrail driven out-of-the-loop performance decrement in which operators are rotely following the procedural instructions without proper understanding.

It is for these reasons that Rancor intentionally allows the operator to revisit steps or jump to nonsequential steps. In this format, Rancor serves as a task analysis procedure tool that is able to capture the plant state against operator goals as captured by the procedure selected. The operator goals may correctly or incorrectly align with the procedure. Therefore, the operator's navigation throughout the procedure and deviation from a prescribed path is an invaluable tool to understand how procedures should be structured for actual production systems in nuclear process control.

3.4.7 Headless Rancor for Simulations

Rancor has historically been used as a Human Factors and Human Reliability tool for real-time human in the loop scenario testing. The HUNTER virtual operator is now integrated in Rancor. The integration allows for HUNTER to monitor and plant parameters and perform control actions based on written procedures. HUNTER is a probabilistic Monte Carlo model. To support running HUNTER simulations Rancor can run in a headless simulation mode that overrides the normal update loop to run faster than real-time and runs without displaying the normal HSI windows. This mode also disables confirmation dialogs for exiting Rancor and disables the WebAPI. We refer to this as Headless Rancor. The headless option can be specified with the `--headless` flag to rancor.exe. The command line interfacing and configuration files allow for concurrent simulations to be performed from our Hunter-web tool.

3.5 Setting Up Rancor

3.5.1 Installation Requirements

Rancor has minimal hardware requirements and can run on commodity Windows 10 or later desktop and laptops. Operating Rancor with a single operator requires significant screen real estate to accommodate the different windows. A minimum of three windows are required (see Section 3.3.1) for the overview, P&ID, and control functionality. Optionally, a fourth CBP window may need to be displayed.

The minimum recommended display configuration is a single 30" or larger 4K (i.e., 3840 x 2160 pixel) monitor. The window layout is highly configurable, and the Rancor windows can be configured for alternate displays such as ultrawide layouts or multiple monitor displays.

Rancor can also be configured such that each unit of a multi-unit plant has dedicated screens and human input devices. This requires using the server-client functionality of Rancor as described in Section 3.4.1. The server and each client will need to run on a separate computer, and the computers should share the same local network. The network communication is localized between the server and client instances of Rancor. Rancor does not communicate to external servers at this time or require a connection to the internet.

3.5.2 Configuration Files

The configuration files can be found in the Config folder included with the Rancor software (see Figure 52). Some of the files are intended for direct manipulation by users, such as the display

configuration files, while others are intended to store data models. The Scenario folder contains JSON files that define the simulator parameters including the number of units, individual parameters for each unit in a particular configuration, and faults. These files also include some metadata information, such as the participant identification and scenario description and name. These files were not intended for direct user manipulation, as they represent a calculated plant state described by the key-value pairs of plant parameters and values. Advanced users with knowledge of JSON file formatting can manipulate the data models in the scenario files, but this is not recommended for typical users since knowledge of Rancor models is needed to ensure manipulations do not drive calculations to impossible values that could create a nonsensical plant state or even crash the simulator. The fault section of these files is something users may want to modify directly, but a setup utility is also included with Rancor to edit the files in a safer and more easily understandable format. A fault contains several key elements worth noting.

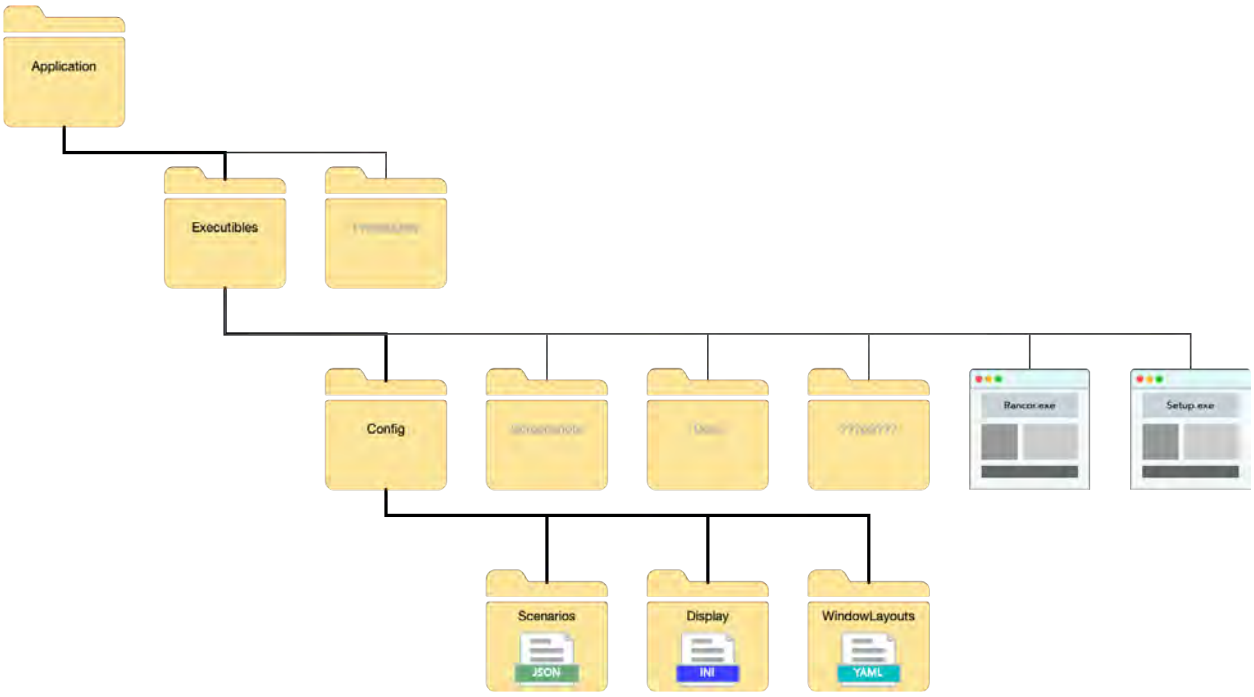


Figure 52. Configuration file locations and types

3.5.2.1 Window Layout File

As noted in Section 3.4.3, Rancor previously used a simple window layout file for configuring the visibility and placement of windows (e.g., overview, P&ID, and controls). The layout listed all the available windows and specified whether they were visible, their size and position, and whether they were resizable or in a fixed position.

The next type of configuration file can be found in the WindowsLayouts folder. These are Yet Another Markup Language (YAML) files that are human readable and easily editable. The revised window layout offers significantly more flexibility for laying out windows for multi-unit server-client configurations.

Each Rancor instance can be configured to use its own window layout configuration. The window layouts have two types of window groups. One set of window groups specifies plant windows. These are intended to show multi-unit overviews. The second type of window groups are unit window groups for individual reactor units. With the WPF application these window groups are distinguished by their DataContext. A WPF DataContext is essenatially a reference to a view model that provides

DependencyProperties and Commands that are bound to the view (i.e., the GUI window). The unit windows are bound to a unit view model to show indications and controls for a single unit. The plant windows have a plant view model as the DataContext. The plant view model has common DependencyProperties for the plant and simulation and contains a list of references to the unit view models. The plant view model facilitates generating common overview displays and controls.

Each instance of Rancor can have multiple window unit groups and multiple units can be assigned to each group. For instance, a four-unit Rancor simulator could be configured with a single large overview and two workstations for separate operators as represented in Figure 53. One workstation would operator Units 1 and 2 and other workstation would operate Units 3 and 4. The instructor could have a fourth machine for running the scenarios.

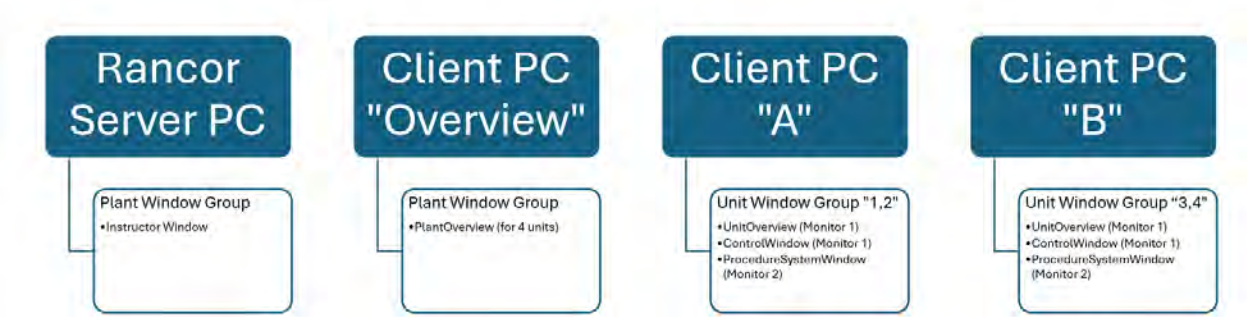


Figure 53. Example Rancor configuration for a 4-unit control room

Table 6. Example window layout configuration for a single unit in Rancor

```
HsiConfig:
  UseSI: false

PlantWindowLayouts:
  MainWindow:
    Left: 0
    Top: 0
    Width: 500
    Height: 300
    Show: true
    Resizable: true

UnitWindowGroups:
  - WindowGroupIdentifier:
"Unit1"
    UnitIdentifiers:
      - 1
    WindowLayouts:

HighPerfHMI.SingleUnitOverview:
        Left: 0
        Top: 0
        Width: 1280
        Height: 720
        Show: true
        Resizable: true
      ControlWindow:
        Left: 0
        Top: 720
        Width: 1280
        Height: 720
        Show: true
        Resizable: true
      ProcedureSystemWindow:
        Left: 1280
        Top: 720
        Width: 1280
        Height: 720
        Show: true
        Resizable: true
```

Table 6 provides a sample configuration for a single-unit plant in Rancor. It includes the following attributes:

- The HsiConfig:UseSI specifies whether the window should display SI units (true) or English units (false)
- The PlantWindowLayouts section specifies that only the MainWindow should be visible
- The UnitWindowGroups list has a single UnitWindowGroup. The name of the window group is Unit1
- The UnitIdentifiers list is a list of all the units in the UnitWindowGroup
- The UnitNumbers in this list correspond with the UnitNumbers in the scenario files
- The WindowLayouts section contains a list of Windows with window parameters.

3.5.2.2 Web API

The Web API specifies whether the Rancor instance is server or client. For client connections, it specifies the server host information to facilitate sharing information and parallel execution of multiple instances.

3.5.2.3 Scenario Configuration

Rancor uses JSON scenario files to define the initial conditions of the plant and faults that occur during the scenario. The PlantModel object contains a UnitModels property with a list of UnitModel states. Rancor requires that at least one UnitModel be included in the UnitModels list. The UnitModel definitions contain the UnitModel parameters for initializing the simulator and an integer UnitNumber that is used as a unique identifier. Each UnitModel in the UnitModels list should have a unique UnitNumber.

```
"Faults": [
  {
    "Description": "Malfunction for LOFW",
    "Trigger": {
      "$type": "Rancor.TimeTrigger, Rancor",
      "TriggerTime": 20
    },
    "Malfunctions": [
      {
        "$type": "Rancor.Malfunction, Rancor",
        "UnitNumber": 1,
        "ComponentId": "FeedWaterPumpA",
        "FinalValue": false,
        "Duration": null
      }
      {
        "$type": "Rancor.Malfunction, Rancor",
        "UnitNumber": 1,
        "ComponentId": "FeedWaterPumpB",
        "FinalValue": false,
        "Duration": null
      }
    ]
  }
]
```

Figure 54. The Faults object from a scenario file specifying a complete loss of feedwater from two simultaneous feedwater pump trips

The scenario files can optionally contain a list of faults under the FaultSettings property, as shown in Figure 54. Each Fault is specified by a description, a trigger, and a list of malfunctions. Rancor supports two types of triggers. The TimeTrigger specifies the simulation time in seconds when the malfunctions should be inserted. Rancor also supports EventTriggers. The EventTrigger specifies a UnitNumber, a

ComponentId, a comparison Operator, and a comparison Value. The trigger will monitor the ComponentID on the corresponding unit, and trigger the fault when the logical condition is met.

3.6 Rancor Logs

Rancor logging has evolved and expanded across different efforts since its original licensing in 2018. It should be noted that the impetus for its creation stems from the limited data available and high cost and effort to acquire data within HRA and human factors for nuclear process control. Rancor was explicitly developed to allow naïve participants to quickly train and begin collecting data to move beyond the small sample size issue that plagues full-scope simulator studies relying on a single operating crew completing scenarios. Instead, student operators could be used to gather data. Efforts to validate Rancor between student and licensed reactor operators have borne out the generalizability of the human performance data Rancor produces (Park et al., 2022). Rancor was also designed with the intent to capture data at a sub-scenario and sub-task level in the parlance of HRA. Traditional full-scope studies rely on expert observers to operator actions as human behaviors during the scenario. This represents a tedious and laborious process that limits the ability to *effectively* collect the necessary data at the appropriate task resolution.

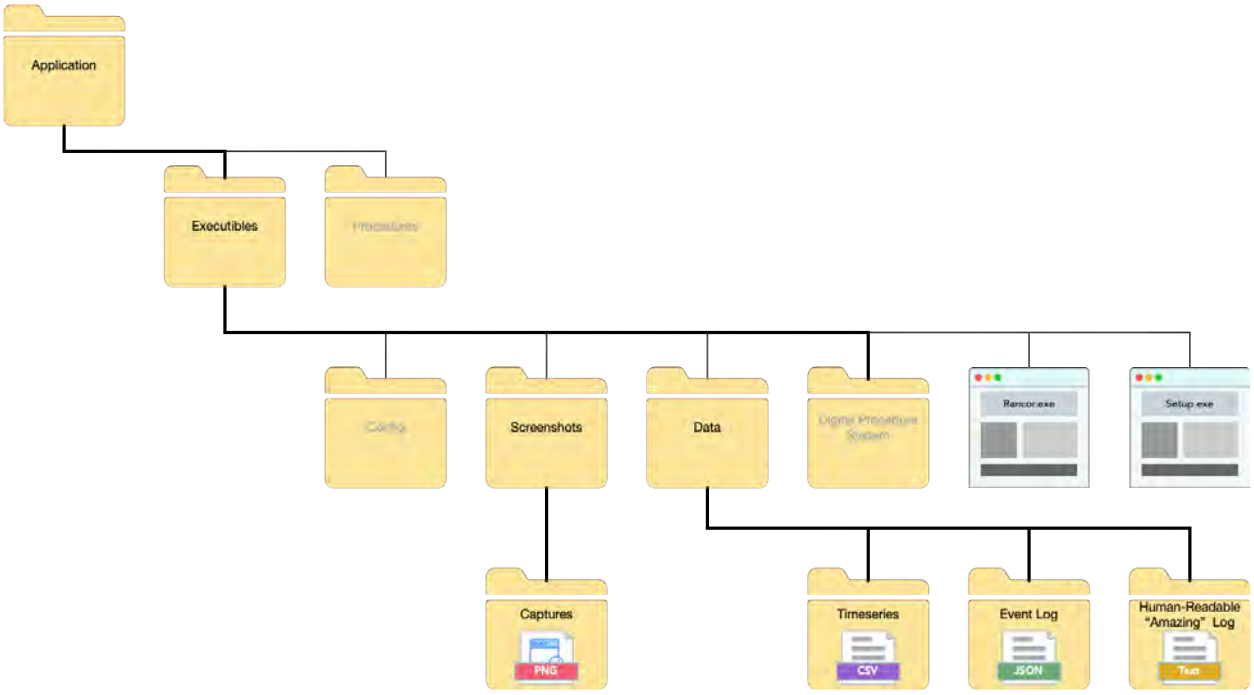


Figure 55. Rancor log files and locations

Several different formats of files are logged based on the type of data recorded as the type of data lends itself to different data structures. Figure 55 shows the four types of files that Rancor logs for later analysis and use.

- Rancor can screen capture interesting operational or plant phenomena as Portable Network Graphic (PNG) files
- Rancor generates time series data about plant parameters as CSV files

- Rancor generates event logs about things that are manipulated (such as fault insertions) as a JSON file
- Rancor generates human-readable logs that are aggregates of major plant state transitions and human actions, captured as text files.

The time series and human-readable log files are elaborated below.

3.6.1 Time Series Logs

As a data collection tool and simulation platform, Rancor collects all plant parameters at one-second time intervals in the timeseries. The file is named according to the participant ID, scenario, and computer-based procedure type.

- File name: [Participant ID*-Scenario*-cbp*-Date.timeseries.csv]

The parameters that are logged are summarized in

Table 7.

Table 7. Rancor simulator parameters that are logged in time series logs

- **ElapsedTime**: The time elapsed time [min:sec]
- **Paused**: [False] The simulator run; [True] The simulator paused
- **TimeMultiplier:** The time multiplier
- **ModeValue:** Mode of the plant
- **UnitNumber:** Unit designator used for multi-unit configuration
- **ControlsActive:** When the simulator is not in pause a covering mask is removed to allow interaction with the controls on the control display
- **GrossMW:** Total power in MW produced by the generator
- **NetMW:** Net power produced by the plant after accounting for house loads from recirculating pumps
- **RodCtrlMode:** [Reactivity] or [Temperature]
- **RodCtrlType:** Reactivity or temperature control mode
- **RX:** Process variable, i.e., actual reactivity [0-100%]
- **RXvesselTemperature:** Process variable, i.e. actual reactor temperature degrees F
- **RodCtrlGoHold:** [Go] the controller will attempt to maintain the temperature or reactivity setpoint, [Hold] the controller holds at the current temperature or reactivity
- **RodCtrlGo:** [True] controller go is enabled; [False] controller go is not enabled
- **RodCtrlHold: [**True] controller hold is enabled; [False] controller hold is not enabled
- **RodCtrlTargetRX**: Reactivity setpoint [0-100%]
- **RodCtrlTargetTemp:** Reactivity setpoint [0-100%]
- **Rod1 / Rod2 / Rod3 / Rod4**: Individual rod position in percent [0-100%]
- **dRX3**: Percentage in reactivity change over the last 3 seconds [%/s]. Reactivity rate fluctuations will induce a reactor trip
- **Tavg**: The average temperature of reactor coolant system [°F]
- **DeltaT:** The temperature difference between hot leg and cold leg [°F]
- **ReactorLevel**: The water level of reactor [%]
- **ReactorPressure**: The pressure of reactor [psig]
- **ContainmentTemperature:** The temperature of containment [°F]

- **RecircPumpsRunning**: Total number of recirculating pumps used to calculate flow through primary coolant loop [0-2]
- **RecircPumpA / RecircPumpB**: [True] Pump running [False] Pump stopped
- **RecircPumpAmpsA / RecircPumpAmpsB:** Current consumption of the recirculating pump [Amps]
- **HotLeg**: The temperature of RCS hot leg [°F]
- **ColdLeg**: The temperature of RCS cold leg [°F]
- **PrimaryFlow**: The flow of primary reactor coolant system [kpph]
- **FeedWaterPumpsRunning**: The number of running feedwater pumps
- **FeedWaterPumpA/ FeedWaterPumpB**: [True] Pump running [False] Pump stopped
- **FeedWaterPumpAIndicator/ FeedWaterPumpBIndicator**: Value shown in HSI
- **FeedWaterPumpASpoof / FeedWaterPumpBSpoof:** Null when no spoof
- **FeedWaterPumpAmpsA / FeedWaterPumpAmpsB**
- **IvRate:** The rate at which the isolation valves open and close
- **MSIVA / MSIVB**: The position of main steam isolation valve, [1] Open [0] Close
- **FWIVA / FWIVB**: The position of feedwater isolation valve, [1] Open [0] Close
- **MSIVA_TargetPosition / MSIVB_TargetPosition**: The target position of main steam isolation valve, [1] Open
- **FWIVA_TargetPosition / FWIVB_TargetPosition**: The target position of feedwater isolation valve, [1] Open
- **SGAin / SGBin:** Inlet valve position for steam generators
- **SGCtrlModeA / SGCtrlModeB:** [True] Automatic or manual steam generator control mode
- **SGCtrlAutoModeA / SGCtrlAutoModeB**: [True] SG control in auto mode; [False] SG control in manual mode
- **SGCtrlManualModeA / SGCtrlManualModeB**: [True] SG control in manual mode; [False] SG control in auto mode
- **GovernorValve**: The position of governor valve (speed control valve), [1] Open [0] Close
- **ControlValve**: The position of load control valve, [1] Open [0] Close
- **BypassValve**: The position of steam dump bypass valve, [1] Open [0] Close
- **TurbineRampRate**
- **ReadyToRoll**: The available status to roll the Latch, [True] Available [False] Not available
- **Latched**: The status of Latch, [True] Latched [False] Not latched
- **ReadyToSync**: The available status to synchronize, [True] Available [False] Not available
- **Synced**: The status of Synch, [True] Synced [False] Not Synced
- **TurbineSpeed**: The speed of turbine [rpm]
- **TurbineSpeedIndicator:** Speed of the turbine in RPM
- **TurbineSpeedSpoof:** Faultable value to simulate a spoofed signal for cybersecurity testing
- **TurbinePressure**: The pressure of turbine [psig]
- **GeneratedValue:** Dollar value profit generated by the plant
- **BonusValue:** Dollar value profit based on maintaining optimal generation based on demand within a configurable range error
- **CombinedValue:** Dollar value of total profit generated
- **Efficiency:** instantaneous thermal to electrical efficiency
- **FeedWaterAFlow / FeedWaterBFlow**: The flow of feedwater [kpph]
- **SteamA / SteamB**: The pressure of main steam [psig]
- **SGLevelA / SGLevelB**: The water level of steam generator [%]
- **SGLevelIndicatorA / SGLevelIndicatorB**

- **SGLevelSpoofA / SGLevelSpoofB**
- **MainSteamLeakRateA / MainSteamLeakRateB:** The leak rate of main steam line [kg/s]
- **SgLeakRateA / SgLeakRateB:** The leak rate of steam generator [kg/s]
- **RcpLeakRateA / RcpLeakRateB**: The leak rate of RCP [kg/s]
- **RcpSealLeakRateA / RcpSealLeakRateB**: The seal leak rate of RCP [kg/s]
- **RcpSealIVA1 / RcpSealIVB**: Valve position of RCP seal isolation valves
- **TimeOutsideDemandBand**: Elapsed time in which the generation was outside the allowed band based on the target demand of the grid
- **LowPrimaryCoolantFlow**: The alarm of the primary coolant flow Low, [True] Activated [False] Not activated
- **RxOverLimit**: The alarm of reactivity Overlimit, [True] Activated [False] Not activated
- **AllRodsDown**: The alarm that all rods are down, [True] Activated [False] Not activated
- **CoreSafetyInterlockEngaged:** Signal for reactor trip [True, False]
- **SafetyInjectionActive**: The alarm of safety injection, [True] Activated [False] Not activated
- **ManualSafetyInjectionActive**: The manual activation of safety injection, [True] Activated [False] Not activated
- **SafetyInjectionRunning**: The status of safety injection**,** [True] Activated [False] Not activated
- **TurbineTrip**: The alarm of turbine trip, [True] Tripped [False] Not Tripped
- **CoreHighTemp**: The alarm of core temperature High, [True] Activated [False] Not activated
- **CoreLowTemp** : The alarm of core temperature Low, [True] Activated [False] Not activated
- **SgAHighLevel / SgBHighLevel:** The alarm of SG water level High, [True] Activated [False] Not activated
- **SgALowLevel / SgBLowLevel**: The alarm of SG water level Low, [True] Activated [False] Not activated
- **CnmtRadMonitor**: The alarm of containment radiation monitoring, [True] Activated [False] Not activated
- **MsRadMonitor**: The alarm of main steam line radiation monitoring, [True] Activated [False] Not activated
- **AtmosDumpActive:** Armed
- **AtmosDump1Stuck / AtmosDump2Stuck / AtmosDump3Stuck / AtmosDump4Stuck**: The status of atmospheric dump valve Stuck malfunction states [nullable boolean]
- **AtmosDump1 / AtmosDump2 / AtmosDump3 / AtmosDump4**: The position of atmospheric dump valve, [1] Open [0] Close
- **AtmosDumpIV1 / AtmosDumpIV2 / AtmosDumpIV3 / AtmosDumpIV4** : The position of atmospheric dump isolation valves, [1] Open [0] Close
- **PorvDumpActive:** PORV vales opened
- **PorvDump1Stuck/ PorvDump2Stuck / PorvDump3Stuck / PorvDump4Stuck**: The status of pilot operated relief valve stuck malfunctions [nullable boolean]
- **PorvDump1 / PorvDump2 / PorvDump3 / PorvDump4**: The position of pilot operated relief valve, [1] Open [0] Close
- **PorvDumpIV1 / PorvDumpIV2 / PorvDumpIV3 / PorvDumpIV4**: The position of pilot operated relief isolation valve, [1] Open [0] Close
- **FwALowFlow / FwBLowFlow**: The alarm of feedwater flow Low, [True] Activated [False] Not activated
- **LowTurbinePressure**: The alarm of turbine pressure Low, [True] Activated [False] Not activated

3.6.2 Human-Readable Event Log

Another type of data recorded is a human-readable event log. The file is named according to the participant ID, scenario, and computer-based procedure type.

- File name: [Participant ID*-Scenario*-cbp*-Date.txt]

Each row of the human-readable event log includes DateTime, UnitNumber, SimulationStep, System, Step Time/Procedure time, and event message, separated by a vertical bar character (|).

- DateTime is recorded in real time
- UnitNumber is the simulator unit. UnitNumber is more relevant for the multi-unit rancor. When there is only operating a single unit, the UnitNumber will always be 1
- SimulationStep is a counter that steps up when the model updates. It's incremented roughly once per second
- System labels (i.e., P3 or HMI) differentiate if the logged line pertains procedurally vs visually. If the line pertains to a CBP, it's labelled 'P3.' If the line pertains to actions involving the GUI, it's labelled 'HMI.' P3 is a carryover term for HUNTER-P3, which HMI refers to human-machine interface, as synonym for HSI
- StepTime and ProcedureTime is the elapsed time to each step and procedure, and its unit is seconds to 1 decimal place. The two are delineated by a forward slash (/)
- Event message includes the status of the simulator for loading the scenario, pause, run, and exit. It allows the analyst to identify the action and time for entering the steps and procedures. The '/' denotes a closing tag. The ExecuteStepNavHandler in the event message logging records the element that was clicked and the link. It allows the analyst to know whether RO or RNO was used. The Steps also have 'Active', 'NotStarted', 'Completed', and 'Incomplete' statuses. Additionally, the analyst can verify the information needed to get error of omission (EOO) and error of commission (EOC) data based on the performance of the operator for each procedure step

Not every parameter is recorded for every log item.

For an example of the human readable event logs for loss of feedwater scenario, see Figure 55. When the simulator runs, the log is recorded as 'Run' in the event message as shown highlighted with the yellow line. Next, the malfunction is executed and logged, as highlighted with the blue line.

```
8/20/2024 2:33:21 PM | 1 |     0 | Simulator |              Run
8/20/2024 2:33:22 PM | 1 |     0 | Plant     |                Mode Changed -> Online
8/20/2024 2:33:41 PM | 1 |    20 | Fault     |                Malfunction Triggered: setting FeedWaterPumpA to False
```

Figure 56. The human readable event log for starting the scenario

The logs such as time and event messages continue to be recorded in order according to the task performed by the operator. As shown in Figure 57, entering the EOP procedure is recorded on the yellow line, and when exiting the procedure, '/' denotes closing the procedure. After entering the procedures, the operator should perform the action for each step. Active status of each step is recorded as shown in green, and the completion of the step is recorded as shown in orange. In Figure 58, analysts can identify EOO and EOC. For Step 3 of the AOP-001 procedure (Rapid shutdown), the operator should verify that turbine is tripped. If not, the operator should move to the RNO step. However, in the example, the operator didn't perform the RNO step and directly moved to Step 4. Thus, the event log recorded the EOO and EOC.

```
8/20/2024 2:33:45 PM | 1 |    24 | P3          |       -/0.0       Procedure - EOP-001 Loss of Feedwater

8/20/2024 2:33:45 PM | 1 |    24 | P3          |       0.0/0.0       Step 34957489-eop-001_loss_of_feedwater-step-1 (Active)
8/20/2024 2:33:47 PM | 1 |    26 | P3          |       2.0/2.0         ExecuteStepNavHandler 34957489-eop-001_loss_of_feedwater-step-1 -> 34957489-eop-001_loss_of_feedwater-step-2 (RO)
8/20/2024 2:33:47 PM | 1 |    26 | P3          |       2.0/2.0       /Step 34957489-eop-001_loss_of_feedwater-step-1 (Completed)

8/20/2024 2:33:47 PM | 1 |    26 | P3          |       0.0/2.0       Step 34957489-eop-001_loss_of_feedwater-step-2 (Active)
8/20/2024 2:33:50 PM | 1 |    28 | P3          |       2.0/4.0         OpAction - RancorEvent 'FeedWaterPumpA'
8/20/2024 2:33:50 PM | 1 |    28 | P3          |       2.0/4.0         Setting FeedWaterPumpA -> True
8/20/2024 2:33:51 PM | 1 |    29 | P3          |       3.0/5.0         ExecuteStepNavHandler 34957489-eop-001_loss_of_feedwater-substep-1.1-rno -> 34957489-eop-001_loss_of_feedwater-substep-2.2 ()
8/20/2024 2:33:51 PM | 1 |    29 | P3          |       3.0/5.0       /Step 34957489-eop-001_loss_of_feedwater-step-2 (Completed)

8/20/2024 2:33:51 PM | 1 |    29 | P3          |       0.0/5.0       Step 34957489-eop-001_loss_of_feedwater-step-2 (Active)
8/20/2024 2:33:54 PM | 1 |    33 | P3          |       3.0/8.0         ExecuteStepNavHandler 34957489-eop-001_loss_of_feedwater-substep-2.2 -> 34957489-eop-001_loss_of_feedwater-step-3 (RO)
8/20/2024 2:33:54 PM | 1 |    33 | P3          |       3.0/8.0       /Step 34957489-eop-001_loss_of_feedwater-step-2 (Completed)

8/20/2024 2:33:54 PM | 1 |    33 | P3          |       0.0/8.0       Step 34957489-eop-001_loss_of_feedwater-step-3 (Active)
8/20/2024 2:33:56 PM | 1 |    34 | P3          |       1.0/9.0         ExecuteStepNavHandler 34957489-eop-001_loss_of_feedwater-substep-3.1 -> 34957489-eop-001_loss_of_feedwater-substep-3.2 (RO)
8/20/2024 2:33:56 PM | 1 |    34 | P3          |       1.0/9.0       /Step 34957489-eop-001_loss_of_feedwater-step-3 (Completed)

8/20/2024 2:33:56 PM | 1 |    34 | P3          |       0.0/9.0       Step 34957489-eop-001_loss_of_feedwater-step-3 (Active)
8/20/2024 2:33:57 PM | 1 |    35 | P3          |      1.0/10.0         ExecuteStepNavHandler 34957489-eop-001_loss_of_feedwater-substep-3.2 -> 34957489-eop-001_loss_of_feedwater-step-4 (RO)
8/20/2024 2:33:57 PM | 1 |    35 | P3          |      1.0/10.0       /Step 34957489-eop-001_loss_of_feedwater-step-3 (Completed)

8/20/2024 2:33:57 PM | 1 |    35 | P3          |      0.0/10.0       Step 34957489-eop-001_loss_of_feedwater-step-4 (Active)
8/20/2024 2:34:05 PM | 1 |    43 | P3          |      8.1/18.1         ExecuteStepNavHandler 34957489-eop-001_loss_of_feedwater-substep-4.4-rno -> 34957489-aop-001_rapid_shutdown-step-1 ()
8/20/2024 2:34:05 PM | 1 |    43 | P3          |      8.1/18.1       /Step 34957489-eop-001_loss_of_feedwater-step-4 (Completed)

8/20/2024 2:34:05 PM | 1 |    43 | P3          |       -/18.1        Step  ()
8/20/2024 2:34:05 PM | 1 |    43 | P3          |       -/18.1      /Procedure - EOP-001 Loss of Feedwater
```

Figure 57. The human readable event log for stepping of the operator's action

```
8/20/2024 4:17:34 PM | 1 |     4 | P3          |       -/0.0       Procedure - AOP-001 Rapid Shutdown

8/20/2024 4:17:34 PM | 1 |     4 | P3          |     0.0/0.0         Step 34957489-aop-001_rapid_shutdown-step-1 (Active)
8/20/2024 4:17:36 PM | 1 |     6 | P3          |     2.0/2.0           ExecuteStepNavHandler 34957489-aop-001_rapid_shutdown-step-1 -> 34957489-aop-001_rapid_shutdown-step-2 (RO)
8/20/2024 4:17:36 PM | 1 |     6 | P3          |     2.0/2.0         /Step 34957489-aop-001_rapid_shutdown-step-1 (Completed)

8/20/2024 4:17:36 PM | 1 |     6 | P3          |     0.0/2.0         Step 34957489-aop-001_rapid_shutdown-step-2 (Active)
8/20/2024 4:17:40 PM | 1 |    10 | P3          |     4.0/6.0           OpAction - RancorEvent 'BypassValve'
8/20/2024 4:17:40 PM | 1 |    10 | HMI         |                       Setting BypassValve -> 0.1
8/20/2024 4:17:40 PM | 1 |    10 | P3          |     4.0/6.0           OpAction - RancorEvent 'BypassValve'
8/20/2024 4:17:40 PM | 1 |    10 | P3          |     4.0/6.0           Setting BypassValve -> 0.1
8/20/2024 4:17:41 PM | 1 |    11 | P3          |     4.0/6.0           ExecuteStepNavHandler 34957489-aop-001_rapid_shutdown-step-2 -> 34957489-aop-001_rapid_shutdown-step-3 (RO)
8/20/2024 4:17:41 PM | 1 |    11 | P3          |     4.0/6.0         /Step 34957489-aop-001_rapid_shutdown-step-2 (Completed)

8/20/2024 4:17:41 PM | 1 |    11 | P3          |     0.0/6.0         Step 34957489-aop-001_rapid_shutdown-step-3 (Active)
8/20/2024 4:17:44 PM | 1 |    14 | HRA         |     3.0/9.0  EOO      OpActionErrorOfOmission RNOAction=1 - Trip Turbine
8/20/2024 4:17:44 PM | 1 |    14 | P3          |     3.0/9.0  EOC      ExecuteStepNavHandler 34957489-aop-001_rapid_shutdown-step-3 -> 34957489-aop-001_rapid_shutdown-step-4 (RNO)
8/20/2024 4:17:44 PM | 1 |    14 | P3          |     3.0/9.0         /Step 34957489-aop-001_rapid_shutdown-step-3 (Completed)

8/20/2024 4:17:44 PM | 1 |    14 | P3          |     0.0/9.0         Step 34957489-aop-001_rapid_shutdown-step-4 (Active)
8/20/2024 4:17:46 PM | 1 |    15 | HMI         |                       Setting RodCtrlTargetRX -> 0
8/20/2024 4:17:46 PM | 1 |    15 | P3          |    1.0/10.0           OpAction - RancorEvent 'ScramReactor'
8/20/2024 4:17:46 PM | 1 |    15 | P3          |    1.0/10.0           Executing ScramReactor()
8/20/2024 4:17:47 PM | 1 |    17 | P3          |    3.0/12.0           Setting ManualSafetyInjectionActive -> True
8/20/2024 4:17:48 PM | 1 |    18 | P3          |    3.0/12.0           Setting ManualSafetyInjectionActive -> True
8/20/2024 4:17:49 PM | 1 |    19 | P3          |    4.0/13.0           ExecuteStepNavHandler 34957489-aop-001_rapid_shutdown-substep-4.4-rno -> 34957489-aop-001_rapid_shutdown-step-5 ()
8/20/2024 4:17:49 PM | 1 |    19 | P3          |    4.0/13.0         /Step 34957489-aop-001_rapid_shutdown-step-4 (Completed)

8/20/2024 4:17:49 PM | 1 |    19 | P3          |    0.0/13.0         Step 34957489-aop-001_rapid_shutdown-step-5 (Active)
8/20/2024 4:17:50 PM | 1 |    19 | P3          |    0.0/13.0           OpAction - RancorEvent 'ManualSafetyInjectionActive'
8/20/2024 4:17:50 PM | 1 |    19 | P3          |    0.0/13.0           Setting ManualSafetyInjectionActive -> True
8/20/2024 4:17:50 PM | 1 |    20 | P3          |    1.0/13.9           ExecuteStepNavHandler 34957489-aop-001_rapid_shutdown-step-5 -> 34957489-aop-001_rapid_shutdown-step-6 (RO)
8/20/2024 4:17:50 PM | 1 |    20 | P3          |    1.0/13.9         /Step 34957489-aop-001_rapid_shutdown-step-5 (Completed)

8/20/2024 4:17:50 PM | 1 |    20 | P3          |    0.0/13.9         Step 34957489-aop-001_rapid_shutdown-step-6 (Active)
8/20/2024 4:17:51 PM | 1 |    20 | HMI         |                       Setting ControlValve -> 0
8/20/2024 4:17:51 PM | 1 |    20 | HMI         |                       Setting GovernorValve -> 0
8/20/2024 4:17:51 PM | 1 |    20 | Fault       |                     Malfunction Triggered: setting FeedWaterPumpA to False
8/20/2024 4:17:52 PM | 1 |    21 | Plant       |                     Mode Changed -> Shutdown
8/20/2024 4:17:53 PM | 1 |    22 | HRA         |    2.0/15.9  EOO      OpActionErrorOfOmission ROAction=1 - Close PORV 1 IV
8/20/2024 4:17:53 PM | 1 |    22 | HRA         |    2.0/15.9  EOO      OpActionErrorOfOmission ROAction=2 - Close PORV 2 IV
8/20/2024 4:17:53 PM | 1 |    22 | HRA         |    2.0/15.9  EOO      OpActionErrorOfOmission ROAction=3 - Close PORV 3 IV
8/20/2024 4:17:53 PM | 1 |    22 | HRA         |    2.0/15.9  EOO      OpActionErrorOfOmission ROAction=4 - Close PORV 4 IV
8/20/2024 4:17:53 PM | 1 |    22 | P3          |    2.0/15.9           ExecuteStepNavHandler 34957489-aop-001_rapid_shutdown-step-6 -> 34957489-aop-001_rapid_shutdown-step-7 (RO)
8/20/2024 4:17:53 PM | 1 |    22 | P3          |    2.0/15.9         /Step 34957489-aop-001_rapid_shutdown-step-6 (Completed)

8/20/2024 4:17:53 PM | 1 |    22 | P3          |    0.0/15.9         Step 34957489-aop-001_rapid_shutdown-step-7 (Active)
8/20/2024 4:17:55 PM | 1 |    24 | P3          |    2.0/17.9           OpAction - RancorEvent 'AtmosDumpIV1'
8/20/2024 4:17:55 PM | 1 |    24 | P3          |    2.0/17.9           Setting AtmosDumpIV1 -> 0.0
8/20/2024 4:17:55 PM | 1 |    25 | P3          |    2.0/17.9           OpAction - RancorEvent 'AtmosDumpIV2'
8/20/2024 4:17:55 PM | 1 |    25 | P3          |    2.0/17.9           Setting AtmosDumpIV2 -> 0.0
8/20/2024 4:17:56 PM | 1 |    26 | P3          |    3.0/18.9           OpAction - RancorEvent 'AtmosDumpIV3'
8/20/2024 4:17:56 PM | 1 |    26 | P3          |    3.0/18.9           Setting AtmosDumpIV3 -> 0.0
8/20/2024 4:17:57 PM | 1 |    26 | P3          |    3.0/18.9           OpAction - RancorEvent 'AtmosDumpIV4'
8/20/2024 4:17:57 PM | 1 |    26 | P3          |    3.0/18.9           Setting AtmosDumpIV4 -> 0.0
8/20/2024 4:17:57 PM | 1 |    27 | P3          |    4.0/19.9           ExecuteStepNavHandler 34957489-aop-001_rapid_shutdown-step-7 -> 34957489-aop-001_rapid_shutdown-step-8 (RO)
8/20/2024 4:17:57 PM | 1 |    27 | P3          |    4.0/19.9         /Step 34957489-aop-001_rapid_shutdown-step-7 (Completed)

8/20/2024 4:17:57 PM | 1 |    27 | P3          |    0.0/19.9         Step 34957489-aop-001_rapid_shutdown-step-8 (Active)
8/20/2024 4:17:58 PM | 1 |    28 | P3          |    0.0/19.9           ExecuteStepNavHandler 34957489-aop-001_rapid_shutdown-substep-8.1 -> 34957489-aop-001_rapid_shutdown-substep-8.2 (RO)
8/20/2024 4:17:58 PM | 1 |    28 | P3          |    0.0/19.9         /Step 34957489-aop-001_rapid_shutdown-step-8 (Completed)

8/20/2024 4:17:58 PM | 1 |    28 | P3          |    0.0/19.9         Step 34957489-aop-001_rapid_shutdown-step-8 (Active)
8/20/2024 4:17:59 PM | 1 |    29 | P3          |    0.0/21.0           ExecuteStepNavHandler 34957489-aop-001_rapid_shutdown-substep-8.2 -> Exit (RO)
8/20/2024 4:17:59 PM | 1 |    29 | P3          |    0.0/21.0         /Step 34957489-aop-001_rapid_shutdown-step-8 (Completed)

8/20/2024 4:17:59 PM | 1 |    29 | P3          |      -/21.0         Step  ()
8/20/2024 4:17:59 PM | 1 |    29 | P3          |      -/21.0       /Procedure - AOP-001 Rapid Shutdown
```

Figure 58. The human readable event log demonstrating a skipped step

This page intentionally left blank

4 RANCOR-HUNTER (HUNTER 3.0)

4.1 Integration of Rancor and HUNTER

4.1.1 Background

Rancor was created to support the collection of human-in-the-loop data to assess attention in the context of a microworld simulator (Ulrich, 2017). Rancor was unique because it offered a microworld environment representative of real-world operations. The task was not contrived to support psychology experiments, and operators had an HSI with automatic controllers similar to those in operating plants. Rancor has gone through various iterations as described in Section 3.2.1.

With recent additions, Rancor is evolving from a microworld simulator to a simulation environment that supports not only human-in-the-loop testing and data collection, but also integrated human-machine performance modeling for HRA, synthetic data generation of plant parameters for nuclear engineering purposes, and prototyping interactive HSIs.

Rancor uses the Advanced Nuclear Interface Modeling Environment (ANIME) framework co-developed by INL and the University of Idaho for building HSI prototypes for nuclear power operations (Boring, Lew, and Ulrich, 2017). The ANIME framework has been used to emulate industrial standard HSIs like Honeywell Experion, High Performance HMI (Hollifield et al., 2008), and novel HSI concepts to nuclear power like neumorphic and skeumorphic controls (Hall et al., in press).

The ANIME framework has supported LWRS modernization and more recently LWRS Flexible Plant Operations and Generation (FPOG) Pathway. For both of these efforts we have developed functional HSI protoypes for full-scope simulators.

The increased generalization of Rancor's Application architecture is now at the point where it can become model agnostic. Future versions of Rancor are envisioned to use full-scope simulator APIs to communicate in real-time or faster than real-time with high-fidelity plant models. This would enable HSI human-in-the-loop testing, as well as HUNTER virtual operator integration with full-scope simulators.

4.1.2 System Integration Under the Hood

The HUNTER model was first implemented in the Python programming language. Early versions of HUNTER used loosely coupled model integration by following pre-defined paths generated from full-scope simulator data. The Python codebase had the dynamic fatigue time multiplier but did not have a full PSF implementation for calculating HEPs. Python is an interpreted, dynamically typed language. Python is known to be expressive and useful for rapid development. However, those features become a burden as complexity grows. Statically typed, compiled languages such as C#, C++, or Rust have compilers that can inform developers of potential errors at the time of compilation. The stricter boundaries make unexpected conditions less frequent and easier to deal with.

For these reasons, the Python codebase was migrated to C# in 2023 (Lew et al., 2023). This also supported integration with EMRALD, an INL dynamic PRA system. The C# version also implemented a dynamic Lag-Adapt-Linger model for modeling stress, a full PSF model for calculating HEPs, and a dynamic Fatigue-Speed-Accuracy for scaling elapsed task times with fatigue and decreasing accuracy with fatigue.

Rancor-HUNTER 3.0 uses the C# HUNTER library first developed to support EMRALD-HUNTER. The goal is to maintain a single code base that supports both EMRALD and HUNTER. The core component of HUNTER is the HRA Engine. The HRA engine represents an operator archetype (novice, expert, etc.) and operator state (time on shift, readiness, etc.). The HRA engine is agnostic to the schema of procedures. The HRA Engine simply models elapsed time and human errors for GOMS task level primitives. A HunterOperator Class in Rancor contains the logic to follow 2 column procedures and model procedure following using the HRA Engine.

A new feature of HUNTER 3.0 is the ability to concurrently perform a procedure and continuous actions. Some scenarios require the operator to monitor and adjust plant parameters as they continue further through a procedure. For example, the startup procedure requires the operator to maintain reactor temperature under 750 degrees Fahrenheit through rod control, bypass valve, or speed/load valve adjustments. If the reactor temperature exceeds 750 degrees, the reactor trips and the operator must return to the appropriate step in the startup procedure.

4.2 HUNTER-Web

With our current division of components, HUNTER is a library that contains parameterization for virtual operator archetypes and an HRA Engine for sampling task times and error rates of GOMS task level primitives with dynamic and static PSFs. The HUNTER library is integrated in EMRALD and Rancor. Rancor is evolving to become a simulation framework. With a single command line call, Rancor can simulate a plant scenario with specified initial conditions and HUNTER virtual operator parameters. The simulation can run faster than real time in a *headless* mode that logs data but does not render the HSI.

However, HUNTER modeling typically entails running several hundred simulation trials per scenario and not just a single run. A tool is needed to manage and conduct these simulation trials. HUNTER-Web is this tool. HUNTER-Web is a Python Flask app that uses JSON schemas and a Javascript JSON-Editor library for providing a web interface to author and maintain a database of JSON files. The database contains procedures, scenarios, and simulation trial parameters for Rancor-HUNTER. The HUNTER-Web application also provides a web frontend for running the simulation trials.

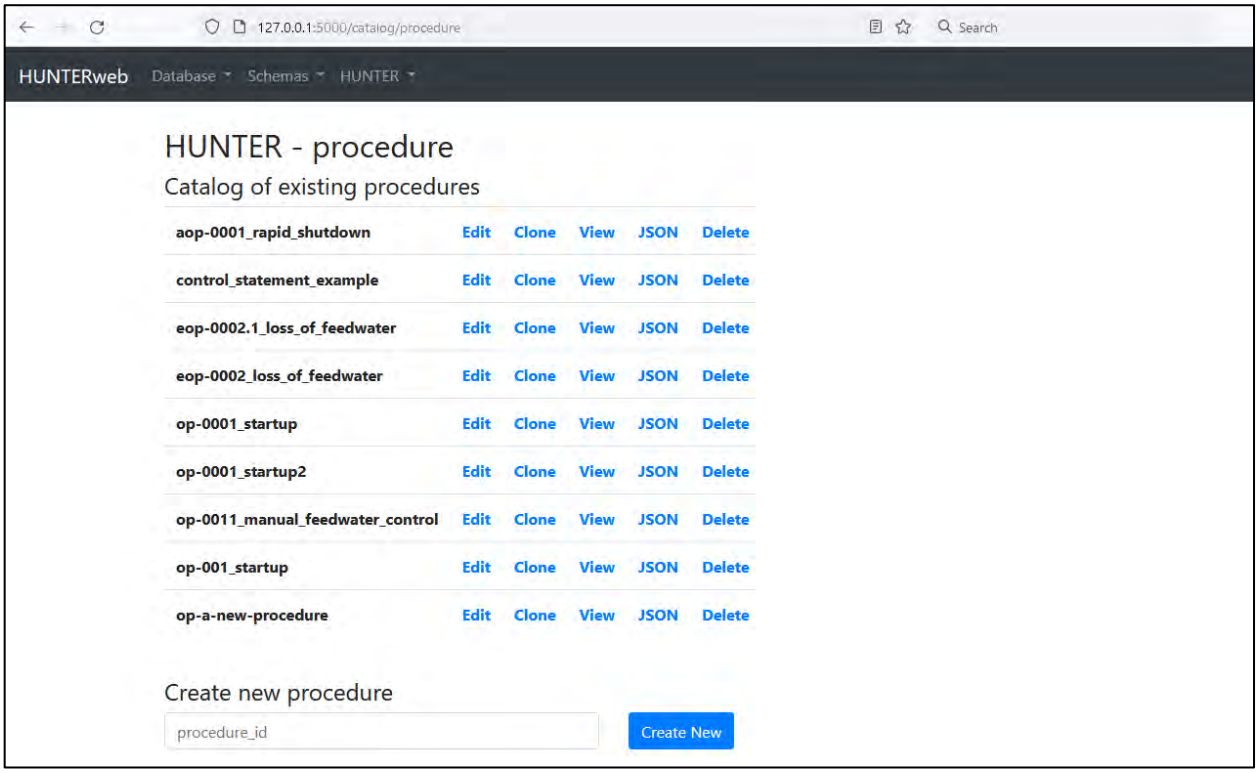


Figure 59. The HUNTER-Web interface for creating and editing procedures

Rancor Version 3 is packaged with HUNTER and HUNTER-Web. This allows users to locally install Rancor and interact with the Rancor simulator and procedure system to author procedures using the

HUNTER-Web tool. Figure 59 shows a screenshot of HUNTER-Web with the catalog of existing procedures. Figure 60 shows a screenshot of the startup procedure. Note that this tool allows creation of procedures for use with the Rancor Digital Procedure System or for use within HUNTER. The same procedures can be used to control the Rancor simulator by human or virtual operators.

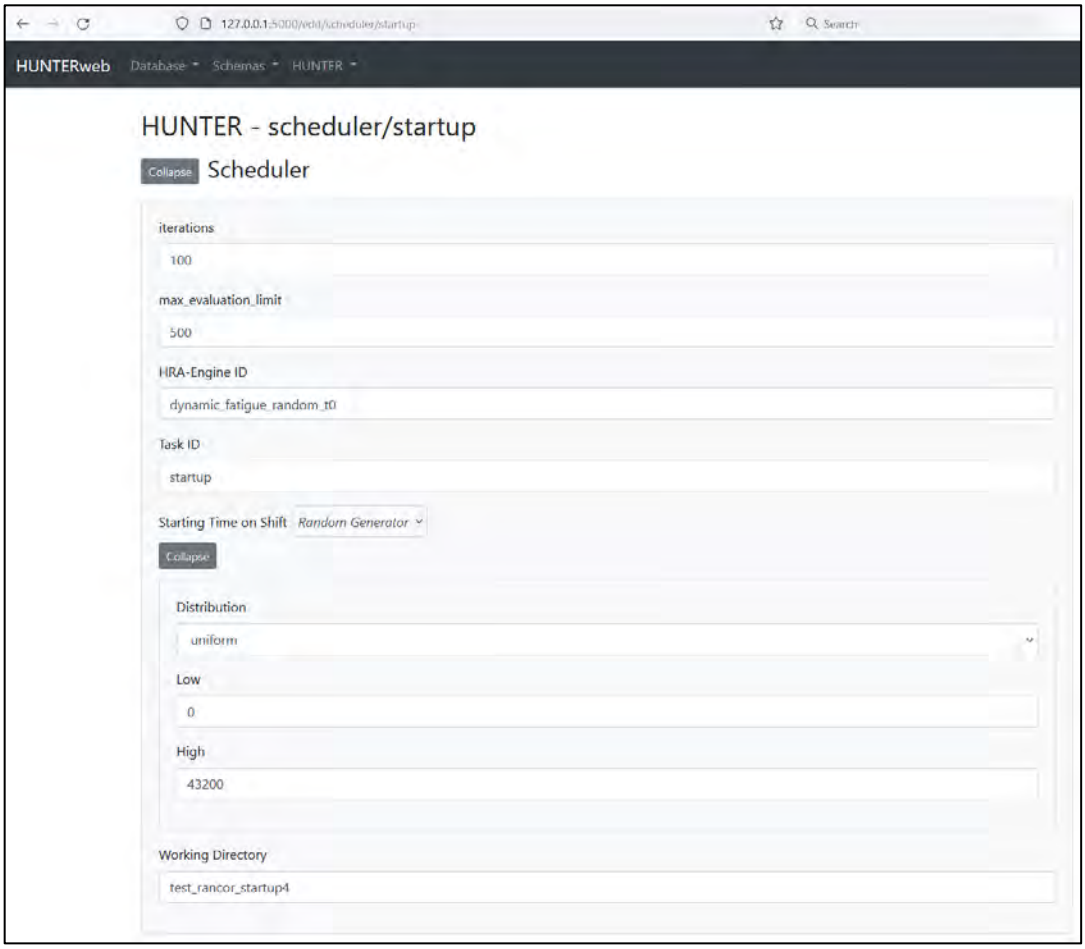


Figure 60. Screenshot of HUNTER-Web editor with a startup procedure

Users can also watch the HUNTER virtual operator follow procedures in real-time with the Rancor HSIs displayed. This provides a mechanism to validate HUNTER's proficiency for completing procedures before conducting human-in-the-loop validation runs. Once a user is satisfied that HUNTER is working, they can run HUNTER Monte Carlo simulations through the HUNTER-Web tool in the Rancor headless mode that runs simulations faster than real-time.

HUNTER-Web requires a Python installation with Flask. The Anaconda installer provides all the necessary Python libraries for running HUNTER-Web out of the box and for running the Rancor data compilation and analysis scripts. HUNTER-Web uses a Rancor Python API for running a single simulation run in headless mode or a set of Monte Carlo runs. Within the Rancor Python API, there are two functions that can be called: ***run_rancor*** (see Table 8) and **run_rancor_monte_carlo** (see Table 9).

Table 8. The run_rancor library for Web-HUNTER

*def **run_rancor***

This function runs a Rancor simulation using specified scenario, settings, and HUNTER files. It can optionally print the simulation output.

Args:

- **scenario_file** (str): The path to the scenario file to be used in the simulation.
- **settings_file** (str): The path to the settings file that contains configuration details for the simulation.
- **hunter_file** (str): The path to the hunter file, which defines the hunter's parameters and behavior.
- **participant_id** (str, optional): A unique identifier for the participant in the simulation. Defaults to 'x'.
- **verbose** (bool, optional): If set to True, the function will print the output of the Rancor simulation to the console. Defaults to False.

Returns:

- **bool**: Returns True if the Rancor simulation runs successfully. Returns False if an error occurs during the execution.

Table 9. The run_rancor_monte_carlo library for Web-HUNTER

*def **run_rancor_monte_carlo***

This function performs multiple Monte Carlo simulations using the Rancor simulator. Each simulation run is executed with a unique participant ID.

Args:

- **scenario_file** (str): The path to the scenario file to be used in each Monte Carlo simulation run.
- **settings_file** (str): The path to the settings file for configuration of each simulation.
- **hunter_file** (str): The path to the hunter file, defining hunter behavior and parameters for each simulation.
- **prefix** (str, optional): A prefix for the participant IDs used in each run. Participant IDs are generated in the form <prefix>001, <prefix>002, etc. Defaults to 'x'.
- **num_runs** (int, optional): The total number of Monte Carlo simulation runs to perform. Defaults to 10.
- **verbose** (bool, optional): If True, the output of each simulation run is printed to the console. Defaults to False.

Returns:

- **None**: This function does not return a value. It runs the specified number of simulations.

4.3 Human-in-the-Loop and Virtual Human-in-the-Loop Testing

One of the major challenges of any HRA approach is the need to validate the outputs of the method against actual human performance data (Kirwan, 1997). Rancor-HUNTER offers a first-of-a-kind platform in which the HRA method (i.e., HUNTER) is built into the simulator (i.e., Rancor). The environment module in HUNTER is Rancor. In other words, the data collected using human-in-the-loop studies with Rancor are directly applicable to informing the modeling of virtual operators in HUNTER. Or, data collected with Rancor can be used to validate HUNTER model predictions. This relationship is depicted in Figure 61.

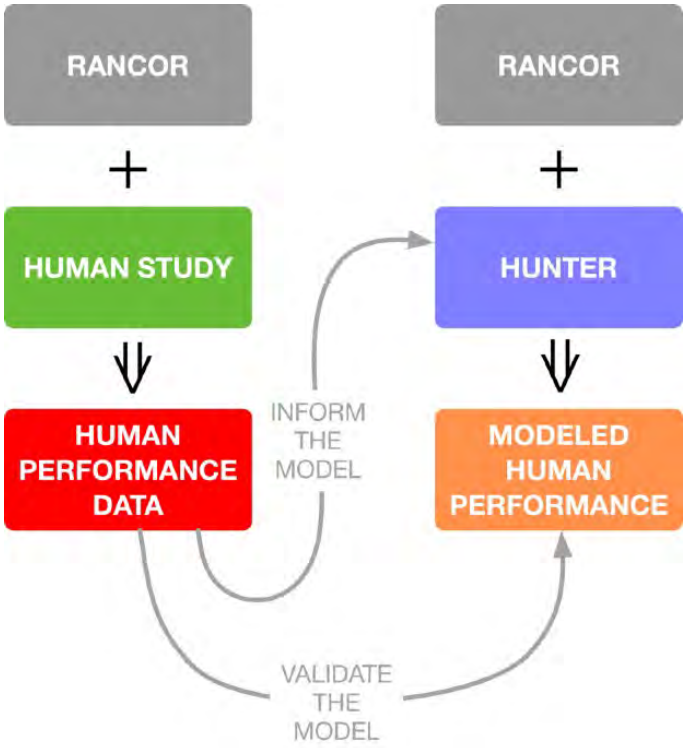


Figure 61. Informing and validating HUNTER with human-in-the-loop data from Rancor.

The degree to which human-in-the-loop data are needed for building the HUNTER model will vary. Typically, there will be insights derived from running scenarios in Rancor with student or professional operators such as:

- The degree of procedural adherence,
- The timing of particular types of tasks and talk level primitives, and
- The level of consequential PSFs like fatigue or stress.

Rancor-HUNTER does not rely on machine learning, and these types of data are not used as training data for the model. Instead, they are manual calibrations that the modeler can put into place to capture individual differences for the user sample being modeled. An example might be a well-trained student population using Rancor (Yang et al., 2023). They may exhibit faster execution times than actual trained operators, who may be more deliberate and cautious in taking actions due to a higher degree of self-checking. The modeler can incorporate different timing primitives in HUNTER to account for this population.

Rancor data may also be used to validate the model. If, for example, a model is built and run using Rancor-HUNTER, it is necessary to determine how well it matches actual human performance data. In such a case, output data from Rancor-HUNTER such as task error rates or run times are compared to empirical data from Rancor runs with human operators. If there is a disparity, the model builder should carefully review logs from Rancor-HUNTER runs against logs from Rancor human studies and adjust modeling characteristics accordingly. In some cases, Rancor-HUNTER may not be a good fit to actual human performance, and such findings should be noted so as to bound the use of Rancor-HUNTER and prevent overgeneralizing its results.

Model building and validation are separate exercises. A common technique is to use more than one scenario. For example, if there are data available from a Rancor study with LOFW and startup scenarios (Hall et al., 2023), one scenario (e.g., LOFW) might be used to help calibrate and inform the initial model in Rancor-HUNTER and then Rancor-HUNTER would run the second scenario (e.g., startup) independently of data inputs from the human performance for that scenario. After Rancor-HUNTER has run the second scenario, its outputs may be compared against those from the human study. If Rancor-HUNTER is a good match to the human performance data, the model may be considered reasonably validated. At such a time, it may be appropriate to start using Rancor-HUNTER to model novel scenarios for which human performance data had not been collected. The greater the availability of human performance data to inform the model, the greater the certainty surrounding the Rancor-HUNTER model predictions will be.

5 EXAMPLE STARTUP SCENARIO

5.1 Scenario Description

Startup is a scenario in which the operator increases reactivity to 100%. The initial condition of the nuclear power plant is shown in Figure 62. Reactivity is 0%, all rods are down, and the reactor coolant pumps and feedwater pumps are stopped. The Rancor operator follows the OP-001 (Startup) procedure. The operator opens several valves such as the pilot-operated relief valve, atmospheric dump isolation valve, feedwater isolation valve, and main steam isolation valve, and starts reactor coolant pumps and feedwater pumps. The rod control is checked for auto control mode, and the reactivity target is gradually increased. If the conditions are satisfied, such as the core temperature is higher than 400°F and the reactor is not scrammed, the turbine can be latched. After the turbine is latched, as a postcondition, the operator should check whether the NOT LATCHED annunciator is OFF. Then, the governor valve position is opened to 100% to increase the turbine speed. When the turbine speed satisfies 1800 RPM, the reactor is online, and the primary flow is in normal operation, the turbine can be synchronized. After the synchronization, the operator gradually increases the load and reactor target until they reach 100%. In this process, heat removal should be performed using the steam generator bypass valve to ensure that the core temperature does not exceed the limit. When the reactor is online, the plant state changes as shown in Figure 63.

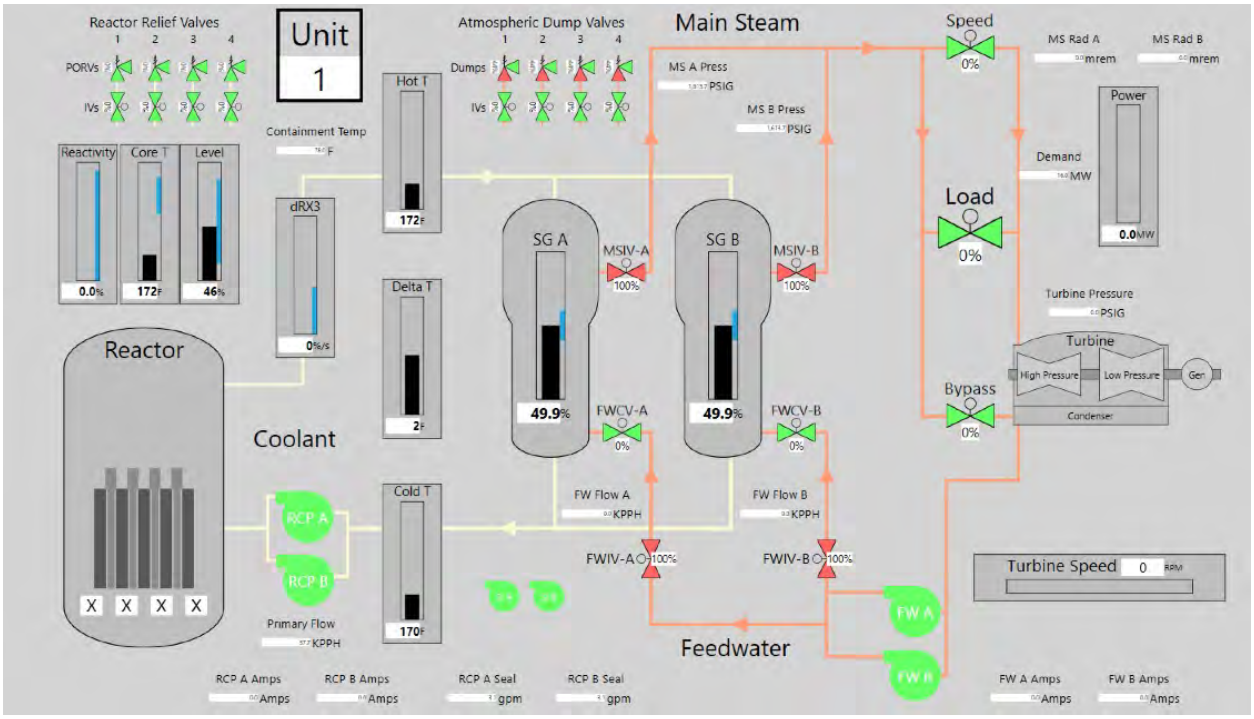


Figure 62. Initial condition for the start-up scenario

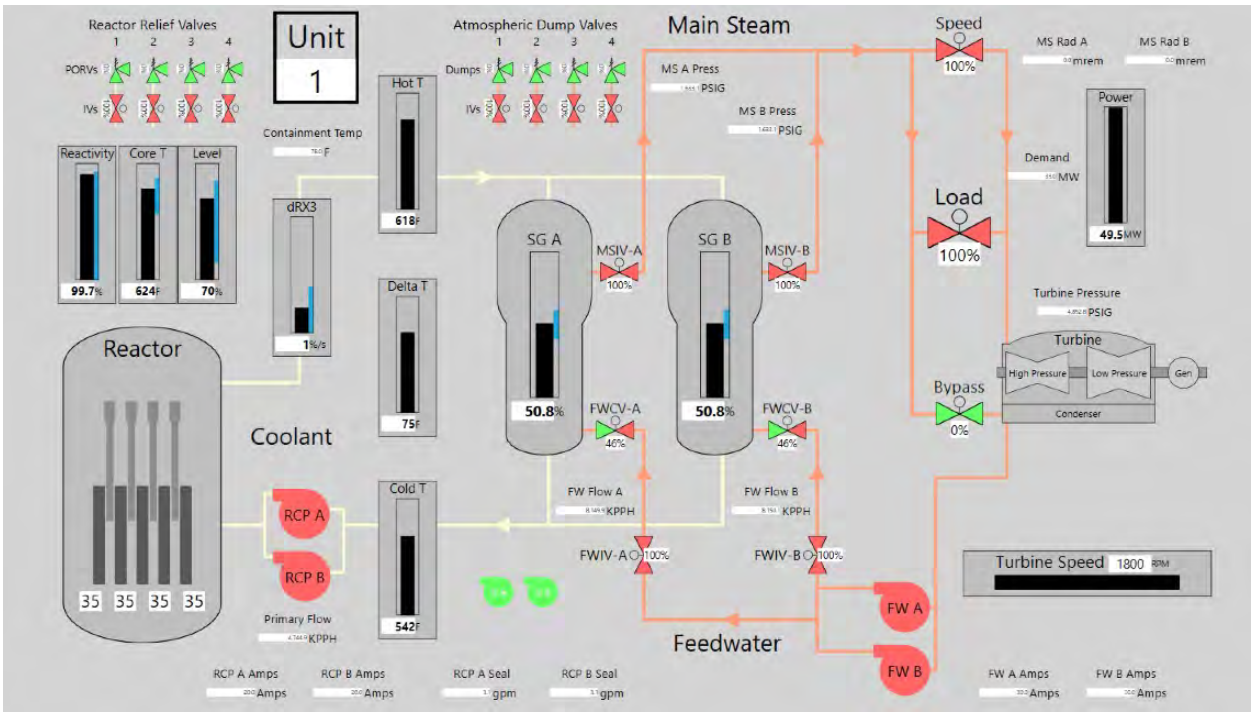


Figure 63. Start-up scenario

5.2 Rancor-HUNTER Runs

We selected the startup scenario described in the previous section to demonstrate the capabilities of Rancor-HUNTER. Previous runs of HUNTER have included station blackout (Boring et al., 2016), SGTR (Boring et al., 2022), LOFW (Lew et al., 2023), and startup (Lew et al., 2023), spanning normal and abnormal operations. We revisited the Startup to consider new contexts that might influence startup and because human-in-the-loop performance data using Rancor were available (Hall et al., 2023; Park et al., 2022). Here we considered 8 conditions in a 2 x 2 x 2 design corresponding to experience (novice vs. nominal), time pressure (present vs. not present), and continuous actions (present vs. not present), respectively. These conditions capture three PSFs, namely experience, stress, and workload.

Table 10 contains completion rates and reactor trip rates for the 8 conditions. The results of the HUNTER virtual operator model simulation provide insight into the performance differences based on operator experience, time pressure, and the use of continuous actions. The findings are summarized below, with all statistics presented as percentages and aggregated from 500 Monte Carlo runs per condition. Given the large number of simulations, the differences observed are considered statistically reliable.

The results show that the nominally (i.e., normally) experienced operators had substantially higher completion rates and fewer reactor trips compared to the low experienced operators. The data also show that the continuous actions reduce the number of reactor trips for the nominally experienced operator and for the low experienced operator when there was time pressure. The continuous actions did not help the low experienced operator when no time pressure was available. This suggests that the pace of the continuous actions was too slow for this condition. Conversely, when we look at the effect of time pressure on continuous actions for the nominally experienced operators, we see that the reactor trip rate increases slightly with time pressure. With time pressure, the pacing of the continuous actions procedure is likely too fast. The virtual operators are making target reactivity adjustments without letting the rod control system catch up. Real operators take into account several parameters, the situational context, and

the rate of change of parameters to decide what control actions should be taken during continuous actions. The HUNTER model is a very simple procedure following model that looks at a single parameter relative to threshold. With the low experienced operator, we can see that the completion rate is extremely low when no time pressure is present.

Table 10. Comparison of Rancor startup scenario resultss for experience, time pressure, and continuous action conditions

Nominal Experience Archetype		
	No Time Pressure	**Time Pressure**
No Continuous Actions	69.0% Completion 12.8% Turbine Trip	81.2% Completion 14.4% Turbine Trip
Continuous Actions	77.8% Completion 5.6% Turbine Trip	89.2% Completion 6.0% Turbine Trip
* All conditions had 500 Monte Carlo runs		

Low Experience Archetype		
	No Time Pressure	**Time Pressure**
No Continuous Actions	6.0% Completion 7.8% Turbine Trip	47.8% Completion 14.8% Turbine Trip
Continuous Actions	5.8% Completion 10.0% Turbine Trip	51.0% Completion 11.2% Turbine Trip
* All conditions had 500 Monte Carlo runs		

These rates are likely not reflective of actual performance, with lower completion and higher trip rates than would be expected at an actual plant. However, these values can be calibrated to human-in-the-loop performance data when available. While startup data generally exist for Rancor, the particular conditions are not captured in previous studies. As such, additional empirical studies using Rancor are required to validate and calibrate the findings. It is nonetheless useful to see initial model runs with HUNTER fully integrated with Rancor, which sets the stage for further scenario work in the future.

Figure 64 has histograms of completion times across the 8 conditions. We can see that the nominally experienced HUNTER operators is successful in a much tighter band than the low experienced operators.

From Figure 65 we can see that the mortality rate for progression through the procedure is most impacted by Step 11. In Step 11 the operator must check to see if the latch conditions are met, latch the turbine, then check to make sure the turbine latched. The high rate of failure is indicative of out of time errors due to not being able to complete all of these operations in a timely manner before the plant is no longer in the ready-to-latch state. If the reactor trips offline before the turbine is latched, the HUNTER model is currently not intelligent enough to go back and retry latching the turbine.

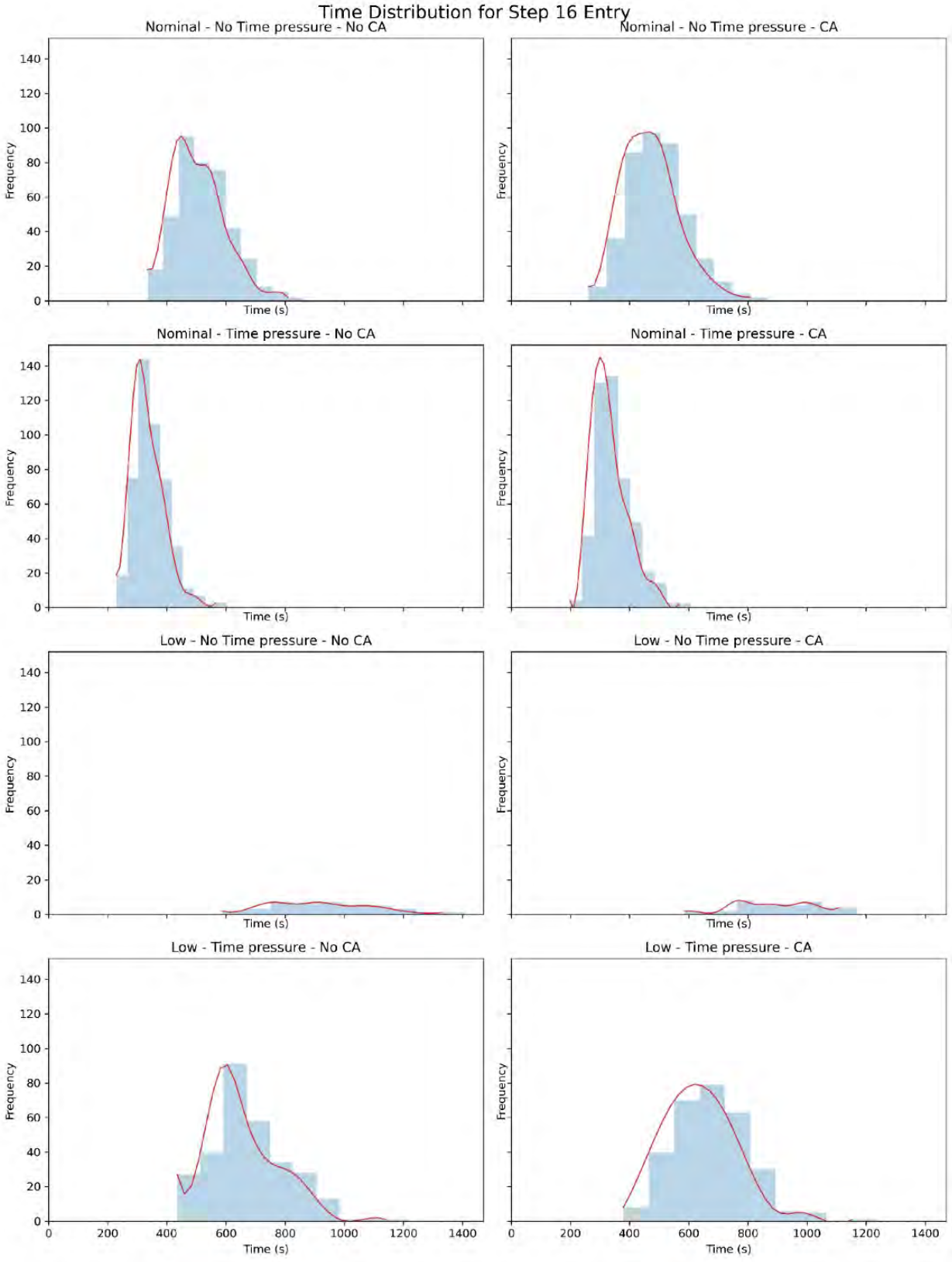


Figure 64. Graphical depiction of completion rates across the 3 factors considered in the startup scenario

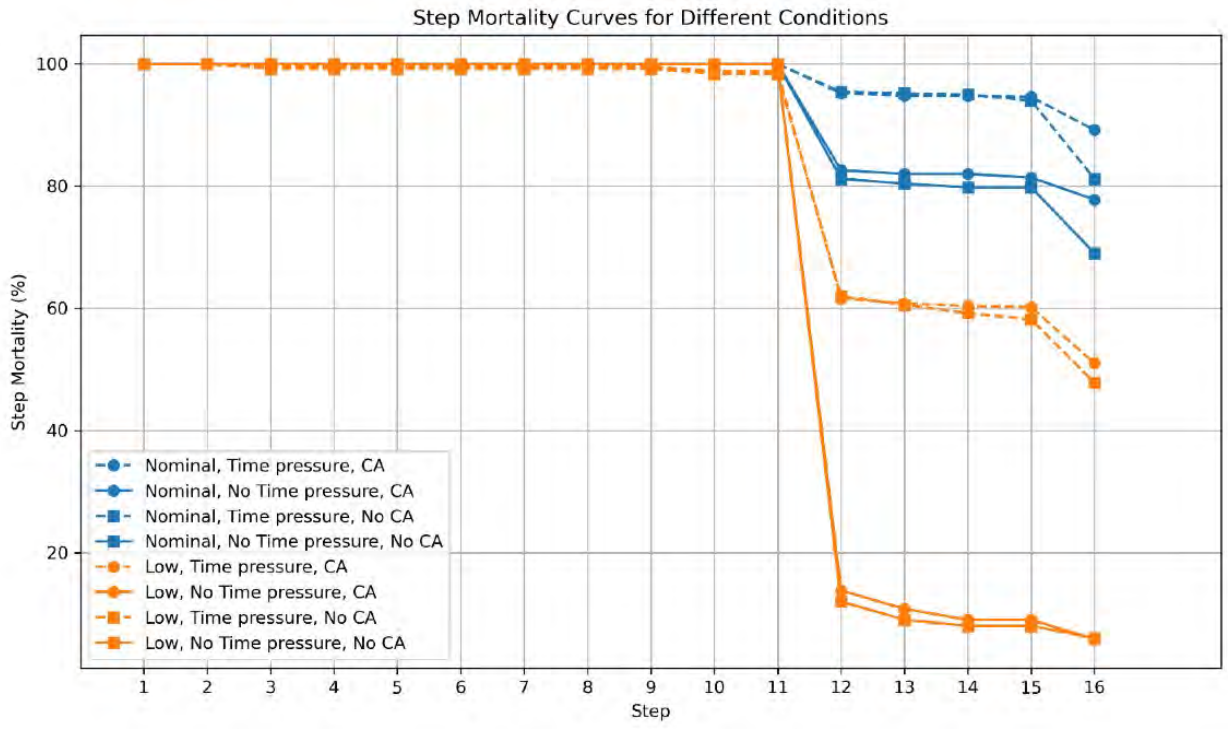


Figure 65. Step mortality for entry to each step of the startup procedure

This page intentionally left blank

6 DISCUSSION

6.1 HUNTER Integrated into Simulator Platforms

The most significant conclusion of this report is that HUNTER can be readily integrated with a simulator. In this case, the HUNTER codebase was integrated with the Rancor Microworld Simulator. While Rancor represents a simplified simulator model, it incorporates the essential functionality of a full-scope training simulator found at NPPs. The key linkage between HUNTER and Rancor is found in the Digital Procedure System, a computer-based procedure system that serves triple duty of logging human-in-the-loop usage data, controlling the simulator (especially in the Type 3 implementation), and providing a mechanism to couple HUNTER. Because HUNTER includes a task module that is essentially a procedure system, the integration with Rancor is seamless. HUNTER is able to act as a virtual operator to control the simulator, all while incorporating human contextual factors in the form of PSFs to shape the operator performance. This report demonstrates this capability and provides a user guide to enable analysts and researchers to perform their own simulations using Rancor-HUNTER.

6.2 Digital Procedure System

The original human reliability analysis method, THERP, made extensive use of expert estimations for quantification. THERP's creator, Alan Swain, later regretted that more data collection using human participants had not taken place (Boring, 2012). This trend has continued with many of the subsequent HRA methods, and a common criticism of HRA is that it is not adequately grounded in empirical human performance data.

Several important efforts have been undertaken to collect human performance data to inform HRA, including the NUCLARR, CORE Data, SACADA, and HuREX database efforts. Each has advanced the availability of HRA data, but each has also balanced trade-offs between ready and complex data collection. Generally, the more complex the data are to collect, the harder it is to populate a sample in a database. For example, an approach that requires extensive manual data collection by subject matter experts observing and coding performance requires large amounts of time and expertise, as it often cannot be accomplished in real time. The difficulties of collecting data are compounded by the relative infrequency of human errors. A common nominal human error probability is anchored at 1E-3. Such an error rate suggests we would only expect to see one error amid 1,000 instances, thereby requiring vast samples of human performance to realize a naturally occurring error.

In conjunction with efforts aimed at control room modernization for nuclear power plants at INL's Human Systems Simulation Laboratory, computer-based procedure prototypes were developed. One artefact of CBPs is that they capture the procedure step that operators are using at any given time. Without CBPs, if such data are needed, operator actions have to be logged manually, with an experimenter shadowing and note-taking procedure steps—a laborious and error-prone process due to the speed with which operators advance through procedure steps and communicate aspects of procedures between multiple crew members. With the advent of CBPs, procedure steps can be logged automatically. These logs can be joined with plant parameter logs, thereby allowing contextual linking between operator actions and plant evolutions.

In recent research by the authors (Boring et al., 2023a), it was found that each procedure step featured an implicit goal. When these goals are made explicit, it becomes possible to automatically determine if the task was completed successfully or not, thereby allowing one measure of automatic human error logging. These error logs are now being used to validate dependency quantification in terms of a task failure's effects on subsequent task performance. Additionally, these rich data logs from CBP and plant parameters are being used to calibrate and validate dynamic HRA models. This paper outlines existing progress using CBPs to inform HRA data, and outlines prospects for wider use.

6.3 Generating Synthetic Human Performance Data

While previous work has established the HUNTER framework and explored relevant use cases, a broader topic remains unaddressed: What are the outputs of HUNTER? Through Monte Carlo sampling, HUNTER provides: frequentist error rates, distributions for calculated human error probabilities, task and subtask durations, the gamut of path flows, and the evolution of PSFs and plant states. Beyond the nominal evolution of human actions, it is also possible to skew performance when desired, for example, if it is of interest to see the outcomes when the virtual operator is highly stressed or fatigued. This allows what-if modeling to stress-test performance under suboptimal conditions. This feature has been used in the HUNTER-P3 module to help inform the development of new operating procedures by screening for problem areas.

The outputs of dynamic HRA extend beyond the typical outputs of an HRA estimating method, and these outputs approach the level of data acquired from a human-in-the-loop (HITL) study using a plant simulator. An advantage of HUNTER is that it creates a virtual human in the loop (VHITL—pronounced "vittle"). While HUNTER VHITL is generally more complex to set up than an expert-based worksheet HRA method, it can be much simpler to execute than a HITL study and produces thousand-fold data beyond what would be possible with actual human participants. Thus, HUNTER is uniquely a source of synthetic data on human performance. With applications of machine learning such as predictive maintenance proliferating, there is an increasing need for human performance data to match plant operational data. HUNTER VHITL provides an important starting point for synthetic data that can, among other uses, be used to train machine learning applications.

Dynamic HRA approaches like HUNTER far exceed their value as a simple replacement for static HRA methods. Dynamic HRA produces a rich variety of data beyond simple human error probabilities. Indeed, HUNTER outputs can serve as synthetic data that enable emerging research and applications in artificial intelligence.

6.4 Next Steps

New dynamic HRA methods like HUNTER bring with them the promise of greater modeling fidelity and greater flexibility to explore what-if scenarios, which may prove especially useful to risk-informing novel designs such as plant upgrades or advanced reactor control rooms. However, dynamic HRA methods are not generally as easy to use as their static HRA forerunners. HUNTER was designed to streamline some of the process of modeling by using plant operating procedures and plant models. In this paper, we have reviewed some of these developments, with a particular focus on the importance of synchronous coupling between dynamic HRA modules. The true advantages of dynamic HRA may only be realized when there is a truly coupled interplay of multiple models working in tandem. HUNTER has demonstrated the value of dynamic feedback loops by coupling to the Rancor Microworld. This simplified simulator shows how a virtual operator can operate a virtual plant by following procedures in a manner that realistically reflects human performance including error tendencies.

Next steps include coupling HUNTER to full-scope simulators and using the operating procedures from actual plants to simulate human performance under a variety of scenarios, thereby validating HUNTER to actual operating experience. Scenarios will also include interactions with upgraded plants featuring new procedures. In this manner, HUNTER can be used in an unconventional manner to help anticipate error traps in new procedures before they are deployed at the plant.

7 REFERENCES

1. Boring, R. (2015). A dynamic approach to modeling dependence between human failure events. In Proceedings of the 2015 European Safety and Reliability (ESREL) Conference (pp. 2845-2851).
2. Boring, R. (2020). The first decade of the Human Systems Simulation Laboratory: A brief history of human factors research in support of nuclear power plants. In Advances in Artificial Intelligence, Software and Systems Engineering (pp. 528-535).
3. Boring, R. (2023). The spatial dimension in human reliability analysis. In Proceedings of the 33rd European Safety and Reliability Conference (ESREL 2023) (pp. 902-909).
4. Boring, R., Agarwal, V., Fitzgerald, K., Hugo, J., & Hallbert, B. (2013). Digital full-scope simulation of a conventional nuclear power plant control room, Phase 2: Installation of a reconfigurable simulator to support nuclear plant sustainability (INL/EXT-13-28432). Idaho Falls: Idaho National Laboratory.
5. 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. In Lecture Notes in Computer Science, 10293 (pp. 3-15).
6. 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.
7. Boring, R., Ulrich, T., Lew, R., Hall, A., Park, J., & Kim, J. (2023a). Preliminary empirical findings on dependency from a simulator study. In Proceedings of the 13th Nuclear Power Plant Instrumentation, Control & Human-Machine Interface Technologies (NPIC&HMIT) (pp. 195-204).
8. Boring, R., Ulrich, T., Lew, R., & Park, J. (2023b). HUNTER procedure performance predictor: Supporting new procedure development with a dynamic human reliability analysis. AHFE Open Access, 117, 29-38.
9. Boring, R. L. (2012). Fifty years of THERP and human reliability analysis. In Joint Probabilistic Safety Assessment and Management and European Safety and Reliability Conference (16B-Th3-5).
10. Boring, R. L., Shirley, R. B., Joe, J. C., Mandelli, D., & Smith, C. L. (2014). Simulation and non-simulation based human reliability analysis approaches (INL/EXT-14-33903). Idaho Falls: Idaho National Laboratory.
11. Boring, R. L., Ulrich, T. A., & Lew, R. (2023c). The next-generation control room: Using the Rancor Microworld Simulator to modernize digital control rooms. In Proceedings of the 13th Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies, NPIC and HMIT 2023 (pp. 677-684). American Nuclear Society.
12. Boring, R., Rasmussen, M., Smith, C., Mandelli, D., & Ewing, S. (2017). Dynamicizing the SPAR-H method: A simplified approach to computation-based human reliability analysis. In 2017 International Topical Meeting on Probabilistic Safety Assessment and Analysis, PSA 2017 (pp. 1024-1031).
13. 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.
14. duBois, Z., Lew, R., & Boring, R.L. (2023). Fail-safe automatic timed response protocol for cyber incident and fault management. In A. Moallem (Ed.), HCI for Cybersecurity, Privacy and Trust. HCII 2023. Lecture Notes in Computer Science, vol 14045. Springer, Cham.

15. Gideon, O., Ulrich, T., Lew, R., Barton, B., & Dubois, Z. (2024). Early-stage usability testing of thermal power dispatch simulator using novice operators. In J. Wright & D. Barber (Eds.), Human Factors and Simulation. AHFE 2024 International Conference. AHFE Open Access, vol 139. AHFE International, USA.
16. Hall, A., Alivisatos, C., Ulrich, T. A., Lew, R., Boring, R. L., & Poresky, C. (In press). Visual style elements in human–system interface design for nuclear power operations: Does style affect performance and preference? Human Factors.
17. Hall, A., Boring, R. L., Ulrich, T. A., Lew, R., Velazquez, M., Xing, J., Whiting, T., & Makrakis, G. M. (2023). A comparison of three types of computer-based procedures: An experiment using the Rancor Microworld Simulator. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 67(1), 2552-2557.
18. Hollifield, B., Oliver, D., Nimmo, I., & Habibi, E. (2008). The high performance HMI handbook: A comprehensive guide to designing, implementing and maintaining effective HMIs for industrial plant operations. Plant Automation Services.
19. IEEE. (2022). IEEE guide for human factors applications of computerized operating procedure systems (COPS) at nuclear power generating stations and other nuclear facilities (IEEE Std 1786-2022). IEEE.
20. Jung, W. D., Kang, D. I., & Kim, J. W. (2005). Development of a standard method for human reliability analysis (HRA) of nuclear power plants - Level 1 PSA full power internal HRA (KAERI/TR-2961/2005). Daejeon, Republic of Korea.
21. Kirwan, B. (1997). Validation of human reliability assessment techniques: Part 1—Validation issues. Safety Science, 27(1), 25-41.
22. Kwon, K.-C., Park, J.-C., Jung, C.-H., Lee, J.-S., & Kim, J.-Y. (1998). Compact nuclear simulator and its upgrade plan. IAEA.
23. Lew, R., Boring, R., & Ulrich, T. (2018). Transitioning nuclear power plant main control room from paper-based procedures to computer-based procedures. Proceedings of the Human Factors and Ergonomics Society.
24. Lew, R., & Ulrich, T. (2023). Rancor reduced order model nuclear power plant simulator for real-time simulations and hardware-in-the-loop testing. In Proceedings of the 13th Nuclear Power Plant Instrumentation, Control & Human-Machine Interface Technologies (NPIC&HMIT) (pp. 1688-1696).
25. Lew, R., Ulrich, T., & Boring, R. (2022). Human unimodel for nuclear technology to enhance reliability (HUNTER) demonstration: Part 2, Model runs of operational scenarios (INL/RPT-22-70076). Idaho Falls: Idaho National Laboratory.
26. Lew, R., Ulrich, T., & Boring, R. (2023). EMRALD-HUNTER: An embedded dynamic human reliability analysis module for probabilistic risk assessment. Idaho Falls: Idaho National Laboratory.
27. Nielsen, J. (1989, September). Usability engineering at a discount. In Proceedings of the third international conference on human-computer interaction on Designing and using human-computer interfaces and knowledge based systems (2nd ed., pp. 394-401). Elsevier Science Inc.
28. O'Hara, J., Higgins, J., Fleger, S., & Pieringer, P. (2012). Human factors engineering program review model (NUREG-0711, Revision 3). Washington, DC: U.S. Nuclear Regulatory Commission.
29. Park, J., & Boring, R. (2020). An approach to dependence assessment in human reliability analysis: Application of lag and linger effects. In The 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management Conference. Venice, Italy.

30. Park, J., Boring, R., & Kim, J. (2019). An identification of PSF lag and linger effects for dynamic human reliability analysis: Application of experimental data.

31. Park, J., Boring, R., Ulrich, T., & Lew, R. (2024). Use of time distributions to predict operator procedure performance in dynamic human reliability analysis. Idaho Falls: Idaho National Laboratory.

32. Park, J., Boring, R. L., Ulrich, T. A., Lew, R., Lee, S., Park, B., & Kim, J. (2022). A framework to collect human reliability analysis data for nuclear power plants using a simplified simulator and student operators. Reliability Engineering & System Safety, 221, 108326.

33. Park, J., Yang, T., Boring, R. L., Ulrich, T. A., & Kim, J. (2023). Analysis of human performance differences between students and operators when using the Rancor Microworld simulator. Annals of Nuclear Energy, 180, 109502.

34. Park, J., Boring, R. L., Ulrich, T. A., Yang, T., & Kim, J. (2022). 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.

35. Parry, G., Beare, A., Spurgin, A., & Moieni, P. (1992). An approach to the analysis of operator actions in probabilistic risk assessment (EPRI Rep. TR-100259). Electric Power Research Institute.

36. Podofillini, L., Dang, V., Zio, E., Baraldi, P., & Librizzi, M. (2010). Using expert models in human reliability analysis—a dependence assessment method based on fuzzy logic. Risk Analysis: An International Journal, 30(8), 1277-1297.

37. Prescott, S., Nevius, D., Ma, Z., & Lawrence, S. (2022). Using EMRALD to simplify and perform dynamic analysis with MAAP. Proceedings of the 16th International Conference on Probabilistic Safety Assessment and Management, PSAM 2022.

38. Prescott, S., Smith, C., & Vang, L. (2018). EMRALD, dynamic PRA for the traditional modeler. In Proceedings of the 14th International Probabilistic Safety Assessment and Management Conference. Los Angeles, CA.

39. Rabiti, C., Alfonsi, A., Mandelli, D., Cogliati, J. J., Wang, C., Talbot, P. W., Malijovec, D. P., et al. (2021). RAVEN user manual. Idaho Falls: Idaho National Laboratory.

40. Rasmussen, M., Standal, M., & Laumann, K. (2015). Task complexity as a performance shaping factor: A review and recommendations in Standardized Plant Analysis Risk-Human Reliability Analysis (SPAR-H) adaption. Safety Science, 76, 228-238.

41. Skjerve, A. B., & Bye, A. (2011). Simulator-based human factors studies across 25 years. London: Springer.

42. Stanton, N. A., Salmon, P. M., Rafferty, L. A., Walker, G. H., Baber, C., & Jenkins, D. P. (2017). Human factors methods: A practical guide for engineering and design. CRC Press.

43. Swain, A. D., & Guttmann, H. E. (1983). Handbook of human reliability analysis with emphasis on nuclear power plant applications (NUREG/CR-1278). Albuquerque, NM: Sandia National Laboratory.

44. Swain, A., & Guttmann, H. (1983). Handbook of human-reliability analysis with emphasis on nuclear power plant applications (NUREG/CR-1278). Albuquerque, NM: Sandia National Laboratory.

45. U.S. Nuclear Regulatory Commission. (2001). Nuclear power plant simulation facilities for use in operator training and license examinations (Rev. 3, Regulatory Guide 1.149). Washington, DC: US NRC.

46. Ulrich, T. A. (2017). The development and evaluation of attention and situation awareness measures in nuclear process control using the Rancor Microworld Environment. University of Idaho.

47. Ulrich, T. A., Werner, S., & Boring, R. L. (2015, September). Studying situation awareness on a shoestring budget: An example of an inexpensive simulation environment for theoretical research. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting (Vol. 59, No. 1, pp. 1520-1524). Sage CA: SAGE Publications.

48. Ulrich, T. A., Lew, R., Werner, S., & Boring, R. L. (2017, September). Rancor: A gamified microworld nuclear power plant simulation for engineering psychology research and process control applications. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting (Vol. 61, No. 1, pp. 398-402). Sage CA: SAGE Publications.

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

50. Ulrich, T. A., Lew, R., Hancock, S. G., Westover, T. L., Medema, H. D., Boring, R. L., & Minard, N. (2021). Dynamic human-in-the-loop simulated nuclear power plant thermal dispatch system demonstration and evaluation study (INL/EXT-21-64329-Rev000). Idaho Falls: Idaho National Laboratory.

51. Yang, T., Boring, R. L., Kim, J., & Park, J. (2023). An experimental investigation of students' learning effects when using a simplified nuclear simulator. In Proceedings of the 13th Nuclear Plant Instrumentation, Control & Human-Machine Interface Technologies (NPIC&HMIT) (pp. 1002-1011).

52. Yang, T., Boring, R. L., Kim, J., & Park, J. (2023). An experimental investigation of students' learning effects when using a simplified nuclear simulator (INL/CON-23-71715-Rev000). Idaho Falls: Idaho National Laboratory.

53. Yang, T., Boring, R. L., Pope, C., Kim, J., & Park, J. (2024). Analysis of human error and performance in correlation with simulator complexity. Annals of Nuclear Energy, 207, 110727.

APPENDIX A: HUNTER BIBLIOGRAPHY

1. Ahn, J., Heo, Y., Lee, S., Boring, R., & Park, J. (2022). Dynamic approach to dependency analysis in human reliability analysis: Application in a steam generator tube rupture scenario. Proceedings of the 13th International Conference on Applied Human Factors and Ergonomics, New York, July 24-28.
2. Ahn, J., Park, J., Boring, R., Ulrich, T., & Heo, Y. (2022). The HUNTER dynamic human reliability analysis tool: Graphical user interface. Proceedings of the Probabilistic Safety Assessment and Management (PSAM 16), Honolulu, Hawaii, June 26-July 1.
3. Boring, R. (2015). A dynamic approach to modeling dependence between human failure events. Proceedings of the 2015 European Safety and Reliability (ESREL) Conference, 2845-2851.
4. Boring, R. (2023). The spatial dimension in human reliability analysis. Proceedings of the 33rd European Safety and Reliability Conference (ESREL 2023), 902-909.
5. Boring, R., Joe, J.C., & Mandelli, D. (2015). Human performance modeling for dynamic human reliability analysis. Lecture Notes in Computer Science, 9184, 223-234.
6. Boring, R., Mandelli, D., Joe, J., Smith, C., & Groth, K. (2015). A research roadmap for computation-based human reliability analysis, INL/EXT-15-36051. Idaho Falls: Idaho National Laboratory.
7. Boring, R., Mandelli, D., Rasmussen, M., Herberger, S., Ulrich, T., Groth, K., & Smith, C. (2016). Human unimodel for nuclear technology to enhance reliability (HUNTER): A framework for computational-based human reliability analysis. Proceedings of the 13th International Conference on Probabilistic Safety Assessment and Management (PSAM 13), Paper A-531, 1-7.
8. 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.
9. Boring, R., Rasmussen, M., Smith, C., Mandelli, D., & Ewing, S. (2017). Dynamicizing the SPAR-H method: A simplified approach to computation-based human reliability analysis. Proceedings of the 2017 Probabilistic Safety Assessment Conference, 1024-1031.
10. Boring, R. L., Rasmussen, M., Ulrich, T., Ewing, S., & Mandelli, D. (2017). Task and procedure level primitives for modeling human error. Advances in Intelligent Systems and Computing, 589, 30-40.
11. Boring, R., Rasmussen, M., Ulrich, T., & Lybeck, N. (2018). Aggregation of autocalculated human error probabilities from tasks to human failure events in a dynamic human reliability analysis. Proceedings of Probabilistic Safety Assessment and Management.
12. Boring, R., & Herberger, S.M. (2016). Testing subtask quantification assumptions for dynamic human reliability analysis in the SPAR-H method. Proceedings of the Annual Meeting of the Human Factors and Ergonomics Society, 60, 1503-1507.
13. 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, 956-963.

14. Boring, R.L., Shirley, R.B., Joe, J.C., Mandelli, D., & Smith, C.L. (2014). Simulation and non-simulation based human reliability analysis approaches, INL/EXT-14-33903. Idaho Falls: Idaho National Laboratory.

15. Boring, R., Ulrich, T., Lew, R., Hall, A., Park, J., & Kim, J. (2023). Preliminary empirical findings on dependency from a simulator study. . Proceedings of the 13th Nuclear Power Plant Instrumentation, Control & Human-Machine Interface Technologies (NPIC&HMIT), pp. 195-204.

16. 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.

17. Boring, R., Ulrich, T., Park, J., Ahn, J., Heo, Y. (2022). The HUNTER Dynamic Human Reliability Analysis Tool: Overview of the Software Framework for Modeling Digital Human Twins, Probabilistic Safety Assessment and Management (PSAM 16), Honolulu, Hawaii, June 26-July 1.

18. Boring, R., Ulrich, T., Park, J., Ahn, J., & Heo, Y. (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.

19. Boring, R., Ulrich, T., Park, J., Mortenson, T., Ahn, J., & Lew, R. (2022). An adaptable software toolkit for dynamic human reliability analysis: Progress toward HUNTER 2, INL/EXT-21-64525. Idaho Falls: Idaho National Laboratory.

20. Boring, R., Ulrich, T., Lew, R., & Park, J. (2023a). HUNTER procedure performance predictor: Supporting new procedure development with a dynamic human reliability analysis. AHFE Open Access, 117, 29-38.

21. Boring, R., Ulrich, T., Lew, R., & Park, J. (2023b). Synchronous vs. asynchronous coupling in the HUNTER dynamic human reliability analysis framework. Proceedings of the 14th International Conference on Applied Human Factors and Ergonomics (AHFE 2023) and the Affiliated Conferences, San Francisco, California, USA.

22. Heo, Y., Ahn, J., Lee, S., Boring, R., & Park, J. (2022). Characterization of recovery human action mechanisms in nuclear power plants. Proceedings of the 13th International Conference on Applied Human Factors and Ergonomics, New York, July 24-28.

23. Heo, Y., Ulrich, T., Boring, R., Park, J., Ahn, J. (2022). The HUNTER Dynamic Human Reliability Analysis Tool: Coupling an External Plant Code, Probabilistic Safety Assessment and Management (PSAM 16), Honolulu, Hawaii, June 26-July 1.

24. Herberger, S.M., & Boring, R.L. (2016). Simulated human error probability and its application to dynamic human failure events. Proceedings of the 13th International Conference on Probabilistic Safety Assessment and Management (PSAM 13), Paper A-517, 1-8.

25. Joe, J.C., Boring, R.L., Herberger, S., Miyake, T., Mandelli, D., & Smith, C.L. (2015). Proof-of-concept demonstrations for computation-based human reliability analysis: Modeling operator performance during flooding scenarios, INL/EXT-15-36741. Idaho Falls: Idaho National Laboratory.

26. Kim, J., Boring, R., Mortenson, T., Ulrich, T., Park, J., Yang, T., & Kim, J. (2023). Dynamicizing SPAR-H: Generalized form for auto-calculating the performance shaping factor for experience and training. Proceedings of the 13th Nuclear Plant Instrumentation, Control & Human-Machine Interface Technologies (NPIC&HMIT 2023), Knoxville, TN, July 15-20.

27. Lew, R., Boring, R., & Ulrich, T. (2018). Applications of dynamic human reliability assessment (dHRA) for context aware operations. Advances in Intelligent Systems and Computing, 778, 1138-1150.

28. Lew, R., Kim, J., Park, J., Ulrich, T., Boring, R., Prescott, S., & Mortenson, T. (2023). EMRALD-HUNTER: An embedded dynamic human reliability analysis module for probabilistic risk assessment, INL/RPT-23-72783. Idaho Falls: Idaho National Laboratory.

29. Lew, R., Ulrich, T., Boring, R. (2022). Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER) Demonstration: Part 2, Model Runs of Operational Scenarios, INL/RPT-22-70076. Idaho Falls: Idaho National Laboratory.

30. Mandelli, D., Parisi, C., Alfonsi, A., Maljovec, D., St. Germain, S., Boring, R., Ewing, S., Smith, C., & Rabiti, C. (2017). Dynamic PRA of a multi-unit plant. Proceedings of the 2017 Probabilistic Safety Assessment Conference, 1061-1068.

31. Mandelli, D., Parisi, C., Smith, C., Wang, C., Alfonsi, Z.A., Prescott, S., Biersdorf, J., Boring, R., Ulrich, T., & St. Germain, S. (2019). Fully-coupled modeling of a multi-unit plant. Reliability Engineering and System Safety, 185, 303-317.

32. Park, J., & Boring, R. (2020). An approach to dependence assessment in human reliability analysis: Application of lag and linger effects. Proceedings of the 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management Conference, Venice, Italy, November 1–5.

33. Park, J., & Boring, R. (2021). Identification of performance shaping factors affecting subsequent human actions for dependence assessment in human reliability analysis. Proceedings of the 12th International Conference on Applied Human Factors and Ergonomics (AHFE 2021) and the Affiliated Conferences, Manhattan, New York, July 25-29.

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

35. Park, J., Boring, R., Kim, J., Ulrich, T., & Prescott, S. (2023). Analysis of tasks in autonomous systems using the EMRALD dynamic risk assessment tool. Proceedings of the 14th International Conference on Applied Human Factors and Ergonomics (AHFE 2023) and the Affiliated Conferences, San Francisco, California, USA.

36. Park, J., Boring, R., Ulrich, T. (2022). An approach to dynamic human reliability analysis using EMRALD dynamic risk assessment tool. Proceedings of Probabilistic Safety Assessment and Management (PSAM 16), Honolulu, Hawaii, June 26-July 1.

37. Park, J., Boring, R., Ulrich, T. (2022). 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.

38. Park, J., Boring, R., Ulrich, T., Lew, R. (2024). Use of time distributions to predict operator procedure performance in dynamic human reliability analysis, INL/RPT-24-78875. Idaho Falls: Idaho National Laboratory.

39. Park, J., Kim, J., Ulrich, T., Boring, R., Prescott, S. (2023). Analysis of tasks in autonomous systems using the EMRALD dynamic risk assessment tool, 14th International Conference on Applied Human Factors and Ergonomics (AHFE 2023) and the Affiliated Conferences, San Francisco, California, USA.

40. Park, J., Pope, C., Heo, Y., Boring, R., & Prescott, S. (2023). Simulation-based recovery action analysis using the EMRALD dynamic risk assessment tool. Proceedings of the 13th Nuclear Plant

Instrumentation, Control & Human-Machine Interface Technologies (NPIC&HMIT 2023), Knoxville, TN, July 15-20.

41. Park, J., Ulrich, T., Boring, R., Zhang, S., Ma, Z., & Zhang, H. (2021). Modeling FLEX human actions using the EMRALD dynamic risk assessment tool. Proceedings of the 2021 International Topical Meeting on Probabilistic Safety Assessment and Analysis (PSA 2021), Columbus, OH, November 7–12.
42. Park, J., Yang, T., Kim, J., Boring, R., & Pope, C. (2023). An investigation of time distributions for task primitives to support the HUNTER dynamic human reliability analysis. Proceedings of the Applied Human Factors and Ergonomics (AHFE 2023) Hawaii Edition, Hawaii, USA.
43. Rasmussen, M., & Boring, R.L. (2016). Implementation of complexity in computation-based human reliability analysis. Risk, Reliability and Safety: Innovating Theory and Practice, Proceedings of the European Safety and Reliability Conference, 972-977.
44. Rasmussen, M., Boring, R.L., Ulrich, T., & Ewing, S. (2017). The virtual human reliability analyst. Advances in Intelligent Systems and Computing, 589, 250-260.
45. Ulrich, T., Boring, R., L., Ewing, S., & Rasmussen, M. (2017). Operator timing of task level primitives for use in computation-based human reliability analysis. Advances in Intelligent Systems and Computing, 589, 41-49.
46. Ulrich, T., Boring, R., & Rasmussen, M. (2017). Operator timing of task level primitives for use in computation-based human reliability analysis. Advances in Intelligent Systems and Computing, 589, 41-49.
47. Ulrich, T., Boring, R., Ahn, J., Heo, Y., & Park, J. (2022). The HUNTER dynamic human reliability analysis tool: Procedurally driven operator simulation. Proceedings of the Probabilistic Safety Assessment and Management (PSAM 16), Honolulu, Hawaii, June 26-July 1.
48. Ulrich, T., Boring, R., Lew, R., & Rasmussen, M. (2018). Using microworlds to support dynamic human reliability analysis. Proceedings of Probabilistic Safety Assessment and Management.

APPENDIX B: RANCOR BIBLIOGRAPHY

1. Alivisatos, C., Poresky, C., Peterson, P. F., Boring, R. L., & Ulrich, T. A. (2021). Operator preferences in an advanced nuclear control operating room. Proceedings of the 12th Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies (NPIC&HMIT 2021), 25-35.
2. Boring, R. (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.
3. Boring, R., Kelly, D., Smidts, C., Mosleh, A., & Dyre, B. (2012). Microworlds, simulators, and simulation: Framework for a benchmark of human reliability data sources. Joint Probabilistic Safety Assessment and Management and European Safety and Reliability Conference, 16B-Tu5-5.
4. 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.
5. Boring, R. L., Ulrich, T. A., Lew, R., & Hall, A. (2021). A microworld framework for advanced control room design. Proceedings of the 12th Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies (NPIC&HMIT 2021), 14-17.
6. 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.
7. Boring, R., Ulrich, T., Lew, R., Hall, A., Park, J., & Kim, J. (2023). Preliminary empirical findings on dependency from a simulator study. . Proceedings of the 13th Nuclear Power Plant Instrumentation, Control & Human-Machine Interface Technologies (NPIC&HMIT), pp. 195-204.
8. Boring, R., Ulrich, T., Lew, R., Hall, A., & Park, J. (2023, July). Preliminary Empirical Findings on Dependency from a Simulator Study. In *13th Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies, NPIC and HMIT 2023* (pp. 195-204). American Nuclear Society.
9. Choi, J. H., Park, J., Boring PhD, R. L., Lee, S., Park, B., Park, C., & Kim, J. (2021). *An Experimental Investigation of Human Performance Differences Depending on Simulator Complexity* (No. INL/CON-21-62194-Rev000). Idaho National Laboratory (INL), Idaho Falls, ID (United States).
10. duBois, Z., Lew, R., Boring, R.L. (2023). Fail-Safe Automatic Timed Response Protocol for Cyber Incident and Fault Management. In: Moallem, A. (eds) HCI for Cybersecurity, Privacy and Trust. HCII 2023. Lecture Notes in Computer Science, vol 14045. Springer, Cham.
11. Dyre, B. P., Adamic, E. J., Werner, S., Lew, R., Gertman, D. I., & Boring, R. L. (2013). A microworld simulator for process control research and training. Proceedings of the Human Factors and Ergonomics Society 57th Annual Meeting, 1367-1371.
12. Gideon, O., & Ulrich, T. A. (2023, September). Simulator Feature Framework: Requirements to Support Training, Research, and Education. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 67, No. 1, pp. 1109-1115). Sage CA: Los Angeles, CA: SAGE Publications.
13. Gideon, O., & Boring, R. (2023). A graded approach to simulators: Feature requirements mapping to simulator types for nuclear plant control room research use cases. In T. Ahram & W. Karwowski (Eds.), Augmented, Virtual and Mixed Reality Simulation. AHFE (2023) International Conference. AHFE Open Access, 118. AHFE International, USA.

14. Hall, A., Alivisatos, C., Ulrich, T. A., Lew, R., Boring, R. L., & Poresky, C. (In press). Visual Style Elements in Human–System Interface Design for Nuclear Power Operations: Does Style Affect Performance and Preference? *Human Factors*.

15. Hall, A., Boring, R. L., Ulrich, T. A., Lew, R., Velazquez, M., Xing, J., Whiting, T., & Makrakis, G. M. (2023, September). A Comparison of Three Types of Computer-Based Procedures: An Experiment Using the Rancor Microworld Simulator. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 67, No. 1, pp. 2552-2557). Sage CA: Los Angeles, CA: SAGE Publications.

16. Jooyoung Park, Yang, T., Boring, R. L., Ulrich, T. A., & Kim, J. (2023). Analysis of human performance differences between students and operators when using the Rancor Microworld simulator. Annals of Nuclear Energy, 180, Article 109502.

17. Kim, J., Park, J., Boring, R. L., Ulrich, T. A., Lee, S., & Kim, J. (2019, October). An experimental design on the use of rancor microworld simulator: A comparison of human performances between actual operators and students. Transactions of the Korean Nuclear Society Autumn Meeting, Goyang, Korea, 24-25.

18. Kim, S., Kim, J., Park, J., & Boring PhD, R. L. (2020). *Comparison of Human Performance between Operators and Students Using Rancor Microworld Simulator: A Preliminary Result* (No. INL/CON-20-58001-Rev000). Idaho National Laboratory (INL), Idaho Falls, ID (United States).

19. Lew, R., Boring, R. L., & Ulrich, T. A. (2022). Applications of the gamified Rancor microworld simulator model for dynamic human reliability simulation. Proceedings of the 16th Probabilistic Safety Analysis and Management Conference, Paper RO162.

20. Lew, R., Ulrich, R., & Boring, R. (2020). Rancor hybrid energy system microworld. Proceedings of the 64th International Annual Meeting of the Human Factors and Ergonomics Society, 1760-1764.

21. Lew, R., Boring, R. L., & Joe, J. C. (2014). A flexible visual process control development environment for microworld and distributed control system prototyping. Proceedings of the International Symposium on Resilient Control Systems (Resilience Week).

22. Lew, R., & Ulrich, T. (2023). Rancor reduced order model nuclear power plant simulator for real-time simulations and hardware-in-the-loop testing. Proceedings of the 13th Nuclear Power Plant Instrumentation, Control & Human-Machine Interface Technologies (NPIC&HMIT), pp. 1688-1696.

23. Park, J., Boring, R. L., Ulrich, T. A., Lew, R., Lee, S., Park, B., & Kim, J. (2022). A framework to collect human reliability analysis data for nuclear power plants using a simplified simulator and student operators. Reliability Engineering & System Safety, 108326.

24. Park, J., Ulrich, T. A., & Boring, R. L. (2020). An empirical study on the use of the Rancor microworld simulator to support full-scope data collection. Proceedings of the European Safety and Reliability Conference.

25. Park, J., Ulrich, T. A., Boring, R. L., Lee, S., & Kim, J. (2020). Identification of collectible items in the Rancor microworld simulator compared to full-scope studies. Advances in Intelligent Systems and Computing, 1204, 362-367.

26. Park, J., Boring PhD, R. L., & Kim, J. (2021). *A Comparison of Human Error Probabilities Collected from HuREX and SHEEP Frameworks* (No. INL/CON-21-64816-Rev000). Idaho National Lab.(INL), Idaho Falls, ID (United States)

27. Rasmussen, M., Laumann, K., & Boring, R. (2018). Looking for additional data sources for HRA: Microworlds and beyond. Advances in Intelligent Systems and Computing, 778, 310-318.

28. Ulrich, T., Lew, R., & Boring, R. (2020). Simulation technologies for integrated energy systems engineering and operations. Advances in Artificial Intelligence, Software and Systems Engineering, 1213, 566-572.

29. Ulrich, T. A. (2017). The development and evaluation of attention and situation awareness measures in nuclear process control using the Rancor Microworld Simulator. (PhD Dissertation). University of Idaho.

30. Ulrich, T., Boring, R., & Lew, R. (2018). Extrapolating nuclear process control microworld simulation performance data from novices to experts: A preliminary analysis. Advances in Intelligent Systems and Computing, 778, 283-291.

31. Ulrich, T., Boring, R., Lew, R., & Rasmussen, M. (2018). Using microworlds to support dynamic human reliability analysis. Proceedings of Probabilistic Safety Assessment and Management.

32. Ulrich, T., Boring, R. L., & Lew, R. (2019). On the use of microworlds for an error seeding method to support human error analysis. Proceedings of IEEE Resilience Week 2019, 242-246.

33. Ulrich, T., Boring, R., Lew, R., & Rasmussen, M. (2018). Using microworlds to support dynamic human reliability analysis. Proceedings of Probabilistic Safety Assessment and Management.

34. Ulrich, T. A., Lew, R., Mortenson, T., Park, J., Medema, H., & Boring, R. (2020). An integrated energy systems prototype human-system interface for a steam extraction loop system to support joint electricity-hydrogen flexible operations. INL/EXT-20-57880. Idaho Falls: Idaho National Laboratory.

35. Ulrich, T. A., Lew, R., Hancock, S. G., Westover, T. L., Medema, H. D., Boring PhD, R. L., ... & Minard, N. (2021). Dynamic Human-in-the-Loop Simulated Nuclear Power Plant Thermal Dispatch System Demonstration and Evaluation Study (No. INL/EXT-21-64329-Rev000). Idaho National Lab.(INL), Idaho Falls, ID (United States).

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

37. Ulrich, T. A., Werner, S., & Boring, R. L. (2016). Change detection for measuring attention allocation: A new approach for capturing situation awareness. Proceedings of the 60th Annual Meeting of the Human Factors and Ergonomics Society, 1806-1810.

38. Ulrich, T. A., Werner, S., Lew, R., & Boring, R. L. (2016). COSSplay: Validating a computerized operator support system using a microworld simulator. Communications in Computer and Information Science, 617, 161-166.

39. Ulrich, T. A., Werner, S., & Boring, R. L. (2015). Studying situation awareness on a shoestring budget: An example of an inexpensive simulation environment for theoretical research. Proceedings of the 59th Annual Meeting of the Human Factors and Ergonomics Society, 1520-1524.

40. Xing, J., Liu, P., Tang, P., Yilmz, A., Boring, R., & Gibson Jr., G. (2022). Analyzing operation logs of nuclear power plants for safety and efficiency diagnosis of real-time operations. Proceedings of the 29th International Workshop on Intelligent Computing in Engineering (EG-ICE).

41. Yang, T., Park, J., Boring, R. L., & Kim, J. (2022). Human performance analysis depending on expertise and simulator complexity. Transactions of the American Nuclear Society, 126, 148-151.

42. Yang, T., Boring, R. L., Pope, C., Kim, J., & Park, J. (2024). Analysis of human error and performance in correlation with simulator complexity. *Annals of Nuclear Energy*, *207*, 110727.

43. Yang, T., Park, B., Lee, S., Choi, J. H., Park, J., Boring, R. L., & Kim, J. (2021, November). Experimental Analysis of the Effects of Simulator Complexity on Human Performance. In *2021 5th International Conference on System Reliability and Safety (ICSRS)* (pp. 85-91). IEEE.

44. Yang, T., Park, J., Boring, R., & Kim, J. (2023, June). Comparison of Error Rate Depending on Operator Expertise and Simulator Complexity. In *Transactions of the Korean Nuclear Society Spring Meeting*.

45. Yang, T., Boring PhD, R. L., Kim, J., & Park, J. (2023). *An Experimental Investigation of Students? Learning Effects When Using a Simplified Nuclear Simulator* (No. INL/CON-23-71715-Rev000). Idaho National Lab.(INL), Idaho Falls, ID (United States)