AHFE 2023: Vol. XX, 2023
doi: 10.54941/ahfeXXXX

Human error and performance modeling with virtual operator model (HUNTER) synchronously coupled to Rancor Microworld Nuclear Power Plant Simulator

Roger Lew[1], Ronald L. Boring[2], and Thomas A. Ulrich[2]

[1]University of Idaho, Moscow, ID 83843, USA
[2]Idaho National Laboratory, ID 83415, USA

ABSTRACT

The Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER) is a virtual nuclear power plant operator. The virtual operator can follow procedures the timing and reliability of their actions are tied to dynamic human performance modeling parameters (Boring et al., 2016). Unlike traditional (static) risk modeling HUNTER has a dynamic version of SPAR-H to calculate performance shaping factors (PSFs) based on evolving plant conditions. HUNTER models task level Goals-Operators-Methods-Selection rules (GOMS)-HRA as the operator walks through procedure steps. Here we describe how the HUNTER virtual operator model was tightly coupled with the Rancor Nuclear Power Plant Microworld as part of a suite of probabilistic risk assessment (PRA) tools.

Keywords: Human Reliability Analysis, Human Performance Modeling, Risked Informed Systems Analysis, Nuclear Power, Safety

INTRODUCTION

The Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER)-Rancor model is a virtual nuclear power plant operator capable of executing plant procedures and operating a simulated plant. HUNTER Rancor is an expansion of the current HUNTER 2 model (Boring, Lew, Ulrich, Park, 2023). Rancor is a simplified nuclear power plant model that was developed for human factors research for nuclear power operations. This revised model *tightly* couples with the Rancor Nuclear Power Plant Microworld. The tight-coupling synchronously the virtual operator model and rancor plant simulator. The virtual operator can check plant conditions and execute actions. The tight coupling allows provides higher fidelity and accurate time dynamics. HUNTER2-Rancor is part of a suite of probabilistic risk assessment (PRA) tools being developed under the RISMC pathway under LWRS.

RANCOR MICROWORLD SIMULATOR

The University of Idaho and INL jointly developed the Rancor Microworld Simulator, henceforth referred to simply as Rancor. Rancor is a simplified nuclear process control simulator developed to support human factors and human reliability analysis (Ulrich, 2017).

The primary implementation of Rancor is in Microsoft's Windows Presentation Foundation (WPF). The Rancor model was developed as a human

factors research tool for nuclear power operations with naïve. The plant model from this version was ported to Python. This implementation significantly reduces the complexity of integrating HUNTER with a simulator-in-the-loop by allowing the simulator to be a class instance within the HUNTER framework. Alternatively, the simulator would need to be run independently, and communication would need to occur through an API or a remote procedure call framework.

The Rancor Python also has a newly developed interactive User Interface implemented with Textual 0.2.0 (https://www.textualize.io/blog/posts/textual-0-point-2-point-0) to allow analysts and operators to run and interact with the model (see **Error! Reference source not found.**). This is a purpose-built monitoring interface for HUNTER that builds on the non-graphical version of Rancor shown earlier in **Error! Reference source not found.**.

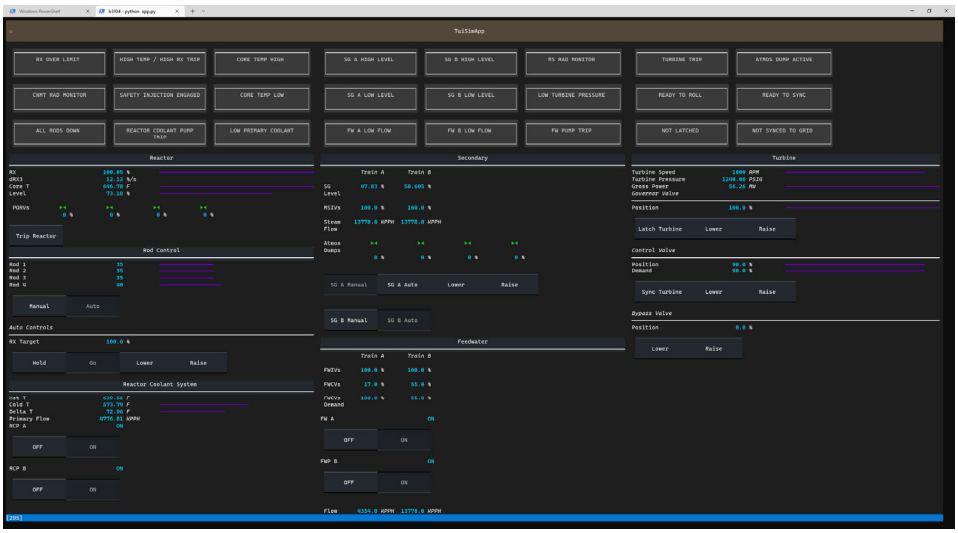

Figure 1: Textual user interface for Rancor Python model running in display terminal mode.

The reduced order model provides the ability to run faster than real-time. Python Rancor runs at 140x real time on a ten-year-old computer. This allows hundreds or even thousands of task runs to be modeled in a couple of hours. In contrast, full-scope simulators are typically locked at real-time or limited to 2x or 4x real-time.

IMPLEMENTATION OF HUNTER-RANCOR

HUNTER-Rancor is implemented in the Python programming language. Python's expressiveness and scientific software stack made it a good candidate for this type of project. The simulation code supports the ability to execute analyst-defined scenarios organized as tasks comprised of procedures or procedure sections. The procedures within the tasks are predefined as inputs to the application and contain all the necessary data elements to execute proceduralized tasks based on the simulated nuclear plant state. The state is used to evaluate the logic within each procedure step, and the virtual operator completion of each step is evaluated with a dynamic HRA module that uses the simulation context to calculate completion durations, HEPs, and success or failure:

1. ***Step***—refers to a small set of activities that represent the base organization unit. By design, all procedures attempt to be closed such that any step's success or failure branches to another step within the same procedure or transitions to a different procedure.
2. ***Preconditions***—refer to conditions that must be affirmed to proceed to the next element of the step. Preconditions can include aggregated individual conditions following different types of logic including any or all being affirmed. Preconditions can trigger a prescribed action, or they constitute the entirety of the step itself to represent diagnostic steps within a procedure.
3. ***Actions***—refer to operator action taken to manipulate the state of a component or system. Actions may have preconditions that must be met prior to their execution, but a step could contain only an action.
4. ***Postconditions***—refers to conditions that must be affirmed following an action taken within a step. The postconditions represent the plant response to an operator action. As such, postconditions must be affirmed before proceeding to the next serial step. When postconditions are not affirmed, a response is not obtained and the step is unsuccessful.
5. ***Substep***—refers to the secondary tasking that is performed within a main procedure step. Substeps are alternative model representations of serial elements and are included as an alternative means of coding a procedure step when the precondition, action, or postcondition is not suitable. Often a procedure step requires multiple substeps by operators to complete the desired tasking.
6. ***Logic***—each step or substep contains a statement that instructs the operator to act or determine a particular plant state. The logic contains the *object,* which is a component or more specifically a parameter of a component, and boolean *criteria* to evaluate the object's state. The result of the logic evaluation dictates whether the step *can* succeed or fail.

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

1. ***Primitive***—refers to the GOMS-HRA task level primitives (Boring et al., 2022), which the HUNTER code uses to determine time durations and nominal HEPs for tasks. More than one task level primitive may be associated with a step.
2. ***PSFs***—refer to performance shaping factors that are dynamically or statically calculated during the simulation and applied to the *primitive*. The nominal HEP and time duration sampled during the simulation are

multiplied by the combined PSFs to adjust the simulated time duration and the HEP value at each time point in the simulation.

3. ***Result***—for a given step to be executed successfully, the logic must be upheld and the primitive evaluation must be evaluated as within the available time and exceed the success criteria for the HEP.

HUNTER-Rancor Implementational Modules

The implementational modules include software code that supports the overall execution of the functional modules. In many cases these are transparent to the user of HUNTER and are therefore backend code required to make HUNTER run.

Plant Modules

Before describing the implementational modules, the plant model itself (a.k.a., the environment module) is worth describing. The plant model is not in itself a module, but it is a crucial aspect of the overall simulation and enables many of the dynamic capabilities provided by HUNTER. A plant model is run in tandem to track the plant state as the virtual operator manipulates components based on the procedure as well as the natural progression of the simulation without intervention, which is typically initiated with a fault and progresses towards a system failure state without operator intervention.

Schedule Module

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

Task Module

Within each simulation run, the Scheduler calls the Task module to execute the procedure steps. The Task module contains the classes that store and manipulate the activity executed during each simulation run. The Task module contains the procedures with their steps. The module can execute unlinked procedures in sequence, but in practice the task typically begins on a specific procedure at a specific step, and then the steps themselves contain all the information to guide the simulation since each step contains the appropriate transition to other procedures in the task.

HRA Module

The Task module relies on the HRA module to evaluate the plant context and performs two key functions. First, it calculates an elapsed time for each step or

substep activity and increments the simulated time for the main simulation *and* the plant simulation. This is important to progress the plant state in step with the time durations required to perform each step activity. Second, the HRA module calculates an instantaneous HEP, which is used to alter the course of the procedure path or incur a time debt.

Task Module

The Step module is responsible for executing each step, which entails evaluating the plant state against the logic defined within the step and evaluating the HRA context. The Step module contains a step class, which serves as the data structure to store parameter objects defining each step. One key parameter object is the logic which contains the plant model component parameter and the logic test used to evaluate the state of the component. The step class also contains several HRA parameter objects. GOMS-HRA task-level primitives are defined in the primitive subclass parameter object. Each step can contain multiple primitives to represent more complicated activities.

HUNTER-RANCOR RUNS

A HRA analysis configures HUNTER-RANCOR through a web tool that creates a database of JSON files. Analysts can author procedures and assign GOMS-primitives to the procedure elements. Then the analyst can specify scenarios consisting of a plant, initial condition for the plant, malfunctions, operator characteristics such as time on shift, and a set of procedures to follow.

Here we examined two scenarios were selected as development and demonstration scenarios. The two selected scenarios are *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. Results of HUNTER are compared to human operator data collected by Chosun University in Korea (Park et al., 2022).

Scenario 1: Loss of Feedwater

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

The virtual operator follows *EOP-0002 Loss of Feedwater* and *AOP-0001 Rapid Shutdown* to verify that feedwater flow has been lost and attempts to restore feedwater flow by manual turning both pumps back on. When the feedwater flow cannot be restored, the virtual operator rapidly shuts down the plant by placing the turbine on bypass and manually tripping the turbine and reactor. HUNTER was configured to run 500 iterations of the startup scenario

with starting time-on-shift between 0 and 12 hours. The time-on-shift affects the dynamic fatigue calculation. The loss of feedwater procedure was authored using the HUNTERweb procedure authoring interface, and Appendix A contains a rendered version of the procedure.

Results

The virtual operator was able to complete the *Loss of Feedwater Procedure* and *Rapid Shutdown Procedure* in all 500 simulated evolutions, as shown in Figure 2.

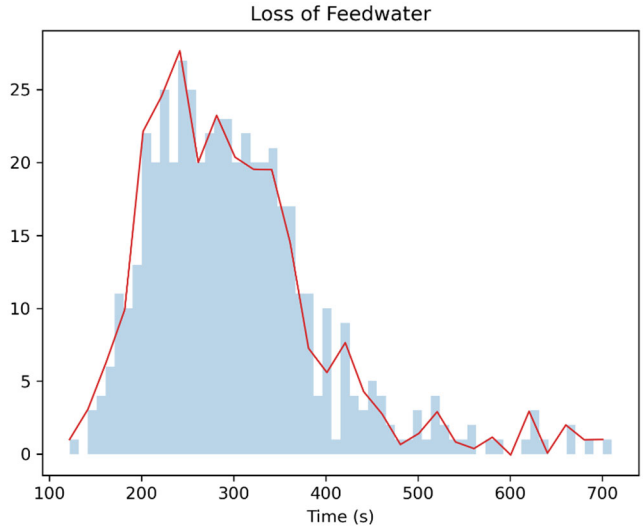

Figure 2: Distribution of times for HUNTER to complete loss of feedwater.

Discussion

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

Table 1. Comparison of HUNTER and human timing for the loss of feedwater scenario

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

Scenario 2: Startup

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

Rancor had a startup procedure that was adapted for use for HUNTER. The procedures for HUNTER need to be much more explicit than traditional procedures so that the virtual operator can complete the steps without having to have internal knowledge of the plant and operational controls. The startup procedure was also adapted to the procedure schema described in Section 4.2.

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

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

Results

The virtual operator was able to complete the startup procedure on 404 of 500 iterations. The average completion time was 733 seconds (12 minutes 13 seconds). The distribution of startup times is shown in Figure 3. Across the 500 iterations, the HUNTER virtual operator performed 8078 action attempts and had an error of commission rate of 0.001238 (i.e., failed 10 actions). The virtual operator checked 23,824 indicators with an error rate of 0.000839 (i.e., failed 20 checks).

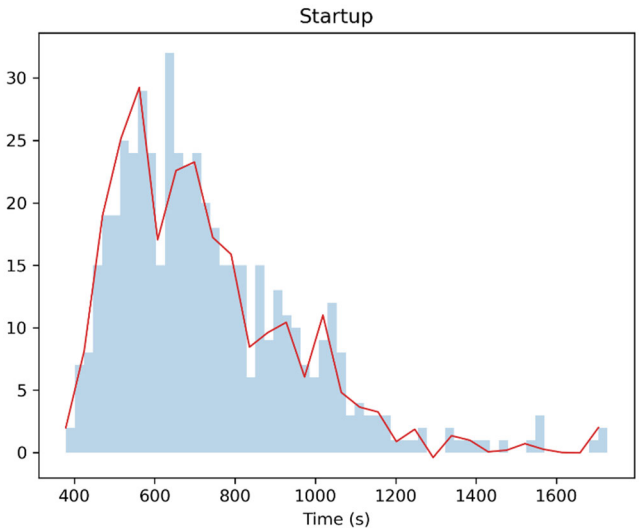

Figure 3: Distribution of times for HUNTER to complete startup.

Discussion

The virtual operator was able to complete the loss of feedwater scenario, and the timing and successful task completion is consistent with human operators (see Table 2) as reported in Part 1 of (Boring et al., 2022). The virtual operator had an average completion time of 302 seconds. The Chosun dataset had 10 students and 4 operators complete the Loss of Feedwater Scenario. The students took 195 seconds on average to complete the scenario, and the operators took an average of 154 seconds The virtual operator was able to successfully latch the turbine in 494 of 500 iterations and sync the turbine/generator in 498 of 500 iterations. However, the virtual operators were only able to bring the plant to 100% power in 404 of 500 iterations. During the ramping phase, actual operators manually monitor plant parameters and increase reactor power and load. The startup procedure for the virtual operator model implemented a strategy to accomplish ramping the plant but did so less successfully than real human operators do to the slow response time for retrieving and taking actions. During this phase of the evolution the human reactor operators did not need to reference the procedures and took faster actions. The HUNTER model could be improved by adding additional GOMS-HRA primitives for modeling manual control that does not rely on formally referencing written procedures for each control decision and action.

Table 2. Comparison of HUNTER and human timing for the startup scenario

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

Nineteen Chosun student participants performed the Rancor startup procedure and had comparatively more consistent and faster performance with an average time of 9 minutes and 40 seconds compared to the 12 minutes observed with the virtual operators (see Table 2). Nine operators completed the startup

procedure and completed the scenario in 12 minutes and 6 seconds. The timing of the HUNTER model is consistent with the operators from the Chosun study.

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

CONCLUSION

Here demonstrated that HUNTER can be synchronously coupled to Rancor. We conducted two scenarios and demonstrated that the virtual operator can complete task evolutions. This represents a significant demonstration of successfully coupling a virtual operator with a virtual plant model for the purposes of dynamic HRA. This approach shows promise, but it also demonstrates the need for further refinement in the modeling to better calibrate virtual operator to actual operator performance.

The authors wrote the code for both HUNTER and Rancor, and therefore have extensive experience with both code bases. During development, we could freely alter the models as needed to assist with the HUNTER-Rancor integration. The ability to debug from HUNTER to Rancor source code greatly eased development the complicated node-based procedure following. Because Rancor mimics the functionality of full-scope simulators, we are confident this coupling could be done with more complicated simulators. For the purposes of proof-of-concept demonstrations, HUNTER-Rancor provides an ideal, non-proprietary code base for developing and testing scenarios.

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

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

As noted, HUNTER does not currently support continuous action procedure steps. In human operators, continuous actions are not performed as part of

periodic surveys of the plant or in a continuous monitoring fashion, depending on the type of action. Time-slicing activities will likely require a virtual buffer of actions that will be affected by the overall workload of the operator. Continuous actions may take a second priority to immediately required actions, for example, and HUNTER will need a way to reflect the availability of the operator to shift and prioritize between actions. Another form of continuous action is alarm monitoring. Given the anecdotally reported complexities of operators filtering multiple alarms to determine which alarms to prioritize, more empirical work will be required before we are able to develop a HUNTER multitasking model that supports alarm response.

ACKNOWLEDGEMENT

This work of authorship was prepared as an account of work sponsored by Idaho National Laboratory (under Contract DE-AC07-05ID14517), an agency of the U.S. Government. Neither the U.S. Government, nor any agency thereof, nor any of their employees makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights.

REFERENCES

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

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

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

Boring, R., Lew, R., Ulrich, T., & Park, J. (2022). Synchronous vs. Asynchronous Coupling in the HUNTER Dynamic Human Reliability Analysis Framework. Human Error, Reliability, Resilience, and Performance AHFE 2023.

Park, J., Boring, R.L., Ulrich, T.A., Yahn, 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.

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

Ulrich, T., Boring, R., L., 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.