INL/RPT-23-72783

Light Water Reactor Sustainability Program

EMRALD-HUNTER: An Embedded Dynamic Human Reliability Analysis Module for Probabilistic Risk Assessment

May 2023

U.S. Department of Energy

Office of Nuclear Energy

INL/RPT-23-72783

EMRALD-HUNTER: An Embedded Dynamic Human Reliability Analysis Module for Probabilistic Risk Assessment

Roger Lew,[1] Jisuk Kim,[2,3] Jooyoung Park,[3] Thomas Ulrich,[3]
Ronald Boring,[3] Steven Prescott,[3] Torrey Mortenson[3]

[1]University of Idaho
[2]Korea Nuclear International Cooperation Foundation
[3]Idaho National Laboratory

May 2023

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

This page intentionally left blank for pagination purposes

ABSTRACT

This report presents the integration of two dynamic risk tools recently developed under the U.S. Department of Energy's Light Water Reactor Sustainability Program. Event Modeling Risk Assessment using Linked Diagrams (EMRALD) is a software package for modeling and running dynamic probabilistic risk assessment. Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER) is a software tool for dynamically modeling human reliability analysis. These two software tools have been integrated as EMRALD-HUNTER to allow streamlined risk modeling of both plant system and human operator reliability. Select features of HUNTER have been embedded in EMRALD to facilitate better incorporation of human reliability analysis in dynamic probabilistic risk assessment models. The integration represents one of the first software-based efforts to reconcile dynamic human reliability analysis with probabilistic risk assessment models. Challenges and implementation solutions encountered while developing the integration of the software are included through two demonstration scenarios of the integrated tools: steam generator tube rupture and loss of feedwater events. The EMRALD-HUNTER demonstration scenarios successfully benchmarked against previous HUNTER runs and against empirical data from simulator studies. This report concludes with selection criteria for when to use EMRALD-HUNTER and when to use the standalone version of HUNTER.

ACKNOWLEDGEMENTS

The authors wish to thank the Risk-Informed System Analysis (RISA) Pathway of the U.S. Department of Energy's Light Water Reactor Sustainability (LWRS) Program for its support of the research activities presented in this report. In particular, we thank Svetlana Lawrence, pathway lead for RISA, for championing the demonstrations found in this report. We also gratefully acknowledge the postdoctoral support provided to the second author (Dr. Jisuk Kim) by the Korea Nuclear International Cooperation Foundation.

CONTENTS

FIGURES

TABLES

ACRONYMS

AFW	auxiliary feedwater
ASEP	Accident Sequence Evaluation Program
ATWS	anticipated transient without scram
BDBE	beyond design basis event
CNS	Compact Nuclear Simulator
CPR	cardiopulmonary resuscitation
CoBHRA	computation-based human reliability analysis
CRA	computational risk assessment
CREAM	Cognitive Reliability and Error Analysis Method
CSI	control of safety injection
DBA	design basis accident
DOE	Department of Energy
ECA	emergency contingency action
EMRALD	Event Modeling Risk Assessment using Linked Diagrams
FAB	feed and bleed
FLEX	flexible coping strategy
FRP	functional recovery procedure
GOMS	Goals-Operator-Method-Selection rules
HEP	human error probability
HFE	human failure event
HPI	high-pressure injection
HPR	high-pressure recirculation
HRA	human reliability analysis
HUNTER	Human Unimodel for Nuclear Technology the Enhance Reliability
HuREX	Human Reliability Data Extraction
IE	initiating event
INL	Idaho National Laboratory
JSON	JavaScript Object Notation
LOCA	loss of coolant accident
LOFW	loss of feedwater
LOOP	loss of offsite power
LOSC	loss of seal cooling
LWRS	Light Water Reactor Sustainability
MAAP	Modular Accident Analysis Program
MSIV	main steam isolation valve
NPP	nuclear power plant
PORV	pilot operated relief valve PORV
PRA	probabilistic risk assessment
PRIME	Procedure-based Risk Investigation Method
PRIMERA	Procedure-based Investigation Method of EMRALD Risk Assessment
PSF	performance shaping factor
PWR	pressurized water reactor
Rancor	Rancor Microworld Simulator
RCP	reactor coolant pump
RCS	reactor coolant system
RELAP	Reactor Excursion and Leak Analysis Program
RHR	residual heat removal

RISA	Risk-Informed Systems Analysis
RTS	reactor trip system
RWST	refueling water storage tank
SBO	station blackout
SGI	steam generator isolation
SGTR	steam generator tube rupture
SIAS	safety injections automation system
SPAR-H	Standard Plant Analysis Risk-Human
SSC	secondary side cooling
SSCR	secondary side cooling recovery
TDP	turbine-driven pump
THERP	Technique for Human Error Rate Prediction
TLP	task level primitive
TMI	Three Mile Island
U.S.	United States

EMRALD-HUNTER: AN EMBEDDED DYNAMIC HUMAN RELIABILITY ANALYSIS MODULE FOR PROBABILISTIC RISK ASSESSMENT

1. INTRODUCTION

Under the United States (U.S.) Department of Energy's (DOE) Light Water Reactor Sustainability (LWRS) Program, the Risk-Informed Systems Analysis (RISA) pathway has funded ongoing research and development activities in support of dynamic risk assessment methods. Legacy risk assessment methods are widely used in the nuclear industry and help ensure the overall safety and reliability of the U.S. commercial operating fleet of nuclear power plants (NPPs). However, these methods are largely static approaches to risk, meaning they operate on a fixed set of plant and operational conditions. Dynamic risk assessment uses Monte Carlo techniques to explore a wider range of outcomes, enabling what-if modeling. The approach is fundamentally different from static risk, since dynamic risk models component relationships and possible values. Randomized samples of model values through the Monte Carlo technique reveal emergent outcomes that could otherwise be challenging to foresee. Such modeling is especially important in the context of plant upgrades, novel plant operation strategies (e.g., use of new mitigating strategies to cope with beyond design bases events [BDBEs]), and advanced reactors, where there is not yet a large base of operating experience to understand system interdependencies or new operational contexts. The shift from risk for as-built systems to new systems provides the perfect basis for ensuring that risk assessment tools can be used to ensure the safety of these new systems. The purpose of developing dynamic risk assessment tools is to ensure the completeness and accuracy of modeling for new systems while also achieving efficiencies for the analysts.

Two recent tools have been developed under RISA to support emerging needs for dynamic risk assessment:

- Event Modeling Risk Assessment using Linked Diagrams (EMRALD; Prescott, Smith, and Vang) is a dynamic probabilistic risk assessment (PRA) software tool to help model causes and mitigations for hardware failures.
- Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER; Boring et al., 2022) is a dynamic human reliability analysis (HRA) software tool to help model operator performance including human errors.

Both of these tools were recently programmatically reviewed in LWRS for their deployment readiness, and both tools were recommended for further development to better support industry risk assessment needs (Choi, 2020). Since the time of the review, 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, Wood, and Ziccarelli, 2022). HUNTER has evolved from a collection of disparate dynamic HRA models into a standalone, integrated software tool (Boring et al., 2022) that also includes an embedded plant simulation to facilitate accurate human-technology interactions (Lew et al., 2022).

The purpose of the research effort captured in this report is to integrate these dynamic PRA and HRA tools to enable both human and plant risk modeling in a single tool. Such a tool would benefit analysts by providing a one-stop tool to cover hardware and operational risk. This report presents the integration of HUNTER within EMRALD to create EMRALD-HUNTER. 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.

This report overviews the development considerations and provides proof-of-concept demonstrations of EMRALD-HUNTER for two scenarios. The report is structured as follows:

- Chapter 1 (this chapter)—introduces the integration of EMRALD and HUNTER
- Chapter 2—provides background on the standalone version of HUNTER
- Chapter 3—explains the existing EMRALD implementation
- Chapter 4—provides a detailed rationale for integrating EMRALD and HUNTER
- Chapter 5—describes the software architecture and implementation details for EMRALD-HUNTER
- Chapter 6—overviews the treatment of dynamic performance shaping factors in EMRALD-HUNTER
- Chapter 7—provides sample analyses for steam generator tube rupture and loss of feedwater events
- Chapter 8—concludes the report with a comparison of the sample analyses to available empirical data and provides brief selection guidance for when to use EMRALD-HUNTER vs. standalone HUNTER.

2. INTRODUCTION TO HUNTER

2.1 What is HUNTER?

HUNTER (Boring et al., 2016 and 2022) has evolved through several iterations into a capable solution for dynamic HRA. HUNTER was born from efforts to adapt a simplified static HRA method like Standard Plant Analysis Risk-Human (SPAR-H; Gertman et al., 2005) into a dynamic HRA framework, essentially as a proof of concept that could be scaled up to more complex HRA methods (Boring et al., 2017). Additionally, HUNTER was scoped to model dynamic risk scenarios such as flexible coping strategy (FLEX) scenarios involving BDBEs and other risk contexts that may be less control room-centric than typical previous efforts (Joe et al., 2015). An additional benefit to the dynamic nature of HUNTER is that it enables human error to be treated as dynamic risk rather than as a static function, the latter being more common in most HRA approaches. Human performance is variable and depends on many factors that can shift and change in seconds based on the evolving surrounding context of the human activities. HUNTER's dynamic approach thus allows for a more realistic assessment of human error risks and a wider array of contexts and scenarios to model, including consideration of what-if modeling that is difficult to perform with traditional static HRA methods.

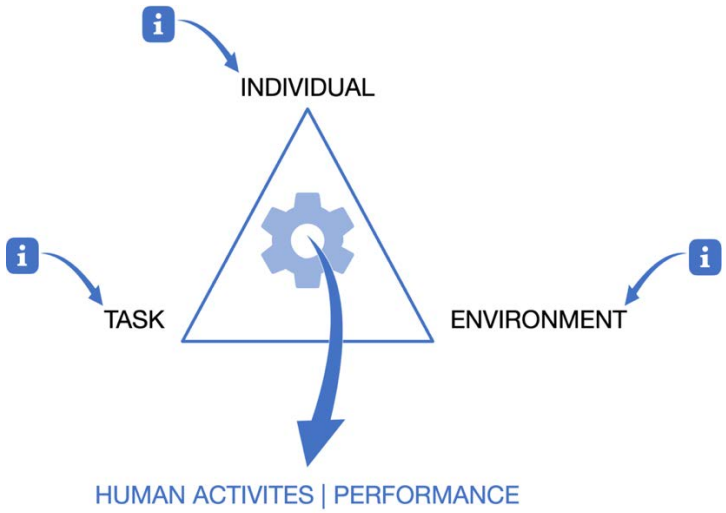


Figure 1. HUNTER conceptual components

HUNTER includes three primary conceptual components—the Individual, Task, and Environment—as depicted in Figure 1.

- The Individual Component models those aspects that affect human performance such as performance shaping factors (PSFs)
- The Task Component models what the human is doing and is primarily defined by operating procedures
- The Environment Component models the technological system the human is using such as a simulation of an NPP that provides the environment context for the other two conceptual components.

Each of these constructs capture various parts of the HUNTER architecture and include subcomponents in the software necessary to drive the scenario. For example, an overall Scheduler Model tracks the Monte

Carlo simulation runs and initiates each task while tracking the global time for the simulation as the task is executed. The Task Module handles the interchange between the Individual and the Environment as it advances along a timeline. Each of these constructs will be captured in more detail in later sections, however, it is important to note that this type of foundation allows HUNTER to capture models across three common areas of variability in risk modeling. The Individual allows for modeling different cognitive models, states, PSFs, or internal factors to human agents. The Task allows for the modeling of many different types of tasks from control room procedures to field operations. Lastly, the Environment allows for the integration of many different models of the "environment." Within HUNTER, the environment can be truly the actual physical environment, it can be a plant model (more commonly used) which represents a simulation of a nuclear power plant, or a BDBE model. All of these mean that HUNTER can support immense variation in modeling needs and provide a dynamic modeling structure for human error in these spaces.

2.2 Background

This section captures a brief overview of key background concepts of the HUNTER method, specifically the separation between static and dynamic HRA methods, an overview of the Goals-Operator-Method-Selection rules (GOMS)-HRA process used in HUNTER, and the past evolutions of the HUNTER method.

2.2.1 Static versus Dynamic HRA

HRA methods have often been described as either static or dynamic methods, with the former being far more common. Static HRA is associated with paper-based worksheet methods where analysts manually complete worksheets (or software equivalents) to arrive at the human error probability (HEP). While the paper nature of these analyses is indicative of the static characterization, this characterization goes further. Static HRA methods are also captured by a fixed calculation of an HEP for any given human failure event (HFE) scenario. What this means is that the assessment of an HEP for any HFE is fixed and is not modified by shifting situations or timing in the overall risk assessment, unless another worksheet is completed for a deviation scenario. This nominal course of human actions is a challenge in terms of ecological validity, because human performance is variable and does respond to changing timing or conditions. Therefore, static HRA methods are more limited in terms of realistically capturing human error. Of course, static HRA methods will capture context (e.g., by using different multipliers associated with the effects of PSFs), but these methods do not readily account for changing contexts within HFEs.

Dynamic HRA is commonly characterized as including modeling methods to capture different time scales and varying conditions to capture a distribution of HEPs rather than a fixed HEP. Dynamic HRA can use Monte Carlo modeling methods, cognitive modeling, or other simulation platforms to capture the overall progression of any scenario and a constantly shifting environment to more realistically model human performance. Dynamic HRA methods can also provide additional metrics of human error beyond the HEP. Metrics such as task time duration, step duration, or even sub-step level parameters can help increase the depth of our understanding of human performance in these contexts. This allows for more analysis of interaction of these factors to task performance. By grounding modeling on the temporal aspects of performance we can better understand how PSFs and overall time executions can affect or predict performance. For example, if plant states mean that the operator cannot physically complete the necessary tasks before some safety system triggers, then the subsequent failure is not a human error. If the agent cannot possibly succeed then the failure was not due to human error. However, as human agents will have an unclear perception of time available before system faults, the stress response of feeling hurried may lead to human errors in the attempt. These errors can be better captured with the temporal foundation in the HUNTER method. This dynamic nature adds complexity but brings HRA closer to reality, which could be extremely valuable to capture error-likely situations and identify likely recoveries and mitigations.

2.2.2 Dynamic versus Computational HRA

HUNTER has shown the utility of dynamic HRA in terms of considering the temporal dimension of human activities. While conventional HRA produces an HEP, HUNTER is able to generate additional quantitative outputs such as task time duration. HUNTER works by coupling a virtual operator model (i.e., a digital human twin) with a virtual plant model (i.e., a simulator or digital twin). The tight coupling of the operator and plant models allows exploration of the nuances of how events unfold through the feedback loop of an operator action and a plant response, leading to further operator responses leading to changes in the plant states, and so on (see Figure 2).

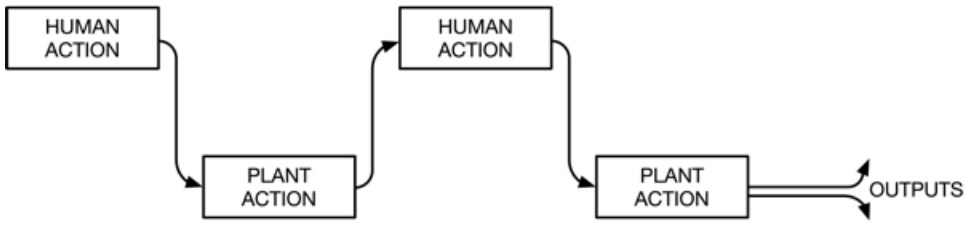


Figure 2. Tight coupling and interactive feedback loop of operator and plant in HUNTER

In accounting for the interplay between the human and the plant, an accurate estimate of time can be produced. The importance of time has been understood from early time reliability methods of HRA, since one of the ultimate measures of successful or failed events is whether or not required actions can be completed within a specific time window. This time window is a reflection of changing plant states (e.g., core heating resulting from insufficient cooling), which may lead to an escalation of the event, including core damage in rare cases. Plant parameters like core temperature evolve in a fashion to where there is a threshold of damage. The time window reflects when that threshold is crossed and provides an upper limit on how long the human has to complete certain actions. Dynamic HRA methods like HUNTER provide a simulation capable of estimating the duration of tasks and determining when safety time thresholds are crossed. Human error, in such cases, is not just the probability that the operator fails to complete a task; it is also the time it takes to complete tasks vs. the time available.

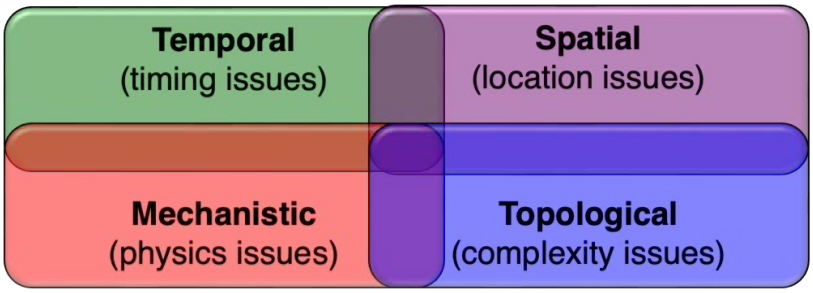


Figure 3. New facets of risk possible with computational risk assessment (courtesy of Curtis Smith)

HUNTER has been framed as a computation-based HRA (CoBHRA) approach (Rasmussen and Boring, 2016), which is the human-centered aspect of computational risk assessment (CRA; Sezen et al., 2019). CRA and CoBHRA make heavy use of computational tools like simulation and simulators to model plant or human performance. These tools may make use of multiple model codes in parallel, e.g.,

HUNTER makes use of both a virtual operator and a simulator to represent the NPP. With multiple codes, there is also the opportunity to consider different aspects of risk than has been the case in historical applications of PRA and HRA. Some of the roadmap topic for CRA to tackle include temporal, spatial, mechanistic, and topological, related to timing, location, physics, and complexity issues, respectively (see Figure 3). Within CoBHRA, mechanistic issues might be considered psychological rather than physical phenomena.

The potential for CoBHRA can be framed by considering the interrogative *wh*-words in English, including who, what, when, why, and how (Koutsoudas, 1968). Conventional static HRA can be considered to answer *who* and *what* as qualitative elements and *how often* as the quantitative element. As discussed in the previous section, dynamic HRA addresses temporal concerns—*when* and *how* long—that were not previously adequately considered in static HRA. Additional CRA elements are being introduced in future versions of HUNTER.

Table 1. GOMS-HRA task level primitives

Primitive	Description
A_C	Performing required physical actions on the control boards
A_F	Performing required physical actions in the field
C_C	Looking for required information on the control boards
C_F	Looking for required information in the field
R_C	Obtaining required information on the control boards
R_F	Obtaining required information in the field
I_P	Producing verbal or written instructions
I_R	Receiving verbal or written instructions
S_S	Selecting or setting a value on the control boards
S_F	Selecting or setting a value in the field
D_P	Making a decision based on procedures
D_W	Making a decision without available procedures
W	Waiting

2.2.3 GOMS-HRA

HUNTER decomposes procedure tasks into task level primitives (TLPs) according to the GOMS-HRA method (Boring and Rasmussen, 2016; see Table 1). Where applicable, the TLPs feature separate primitives for control room and field operations. For example, the *Check* (*C*) TLP can be divided into checking functions within the control room (C_C) or in the field (C_F). These TLPs follow a typical information processing framework adopted in cognitive psychology (Boring et al., 2018) as depicted in Figure 4, representing the most basic human activities delineated as sensation or perception, cognition or decision making, and behavior activities. Each of the TLPs also has associated task level errors, which identify the most likely types of errors to occur for each activity. The TLPs are mapped to nominal HEPs derived from static HRA methods. Additionally, the TLPs for control room activities feature timing data derived from simulator studies (Ulrich et al., 2017; see Table 2). Using nominal HEPs and timing data for the TLPs allows HUNTER to produce both HEP and duration outputs.

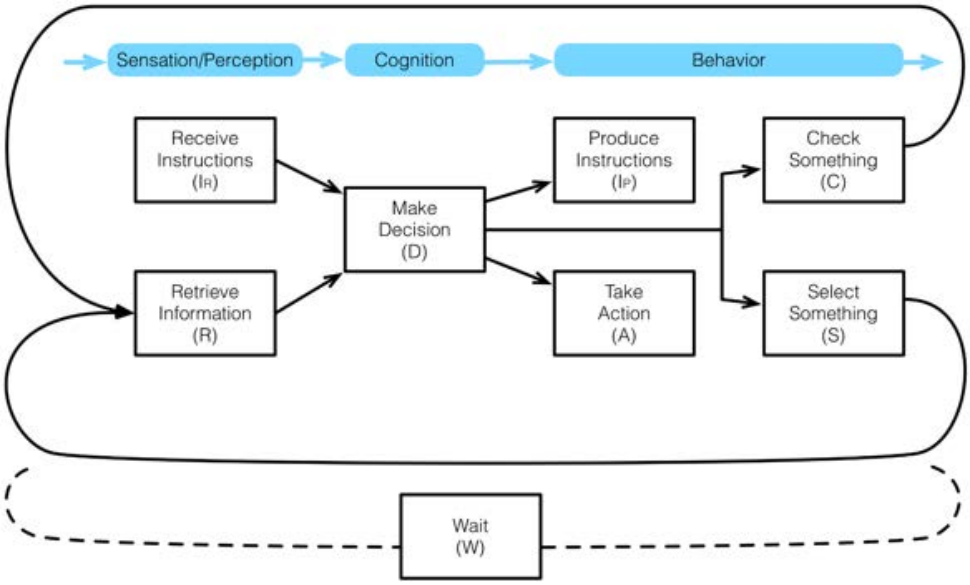


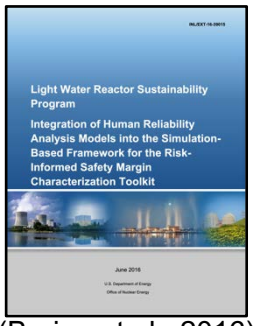
Figure 4. Cognitive model of GOMS-HRA task level primitives

Table 2. GOMS-HRA task level primitives

Task Level Primitive	5th %tile	Time (seconds)	95th %tile
A_C	1.32	18.75	65.26
C_C	2.44	11.41	29.88
D_P	2.62	51	152.78
I_P	3.35	15.56	40.66
I_R	1.47	10.59	31.84
R_C	3.08	9.81	21.90
S_C	3.01	34.48	115.57
W	1.79	14.28	113.61

HUNTER 1

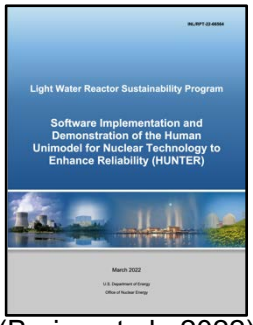
Initial framework and demonstration of HUNTER concepts



(Boring et al., 2016)

HUNTER 2

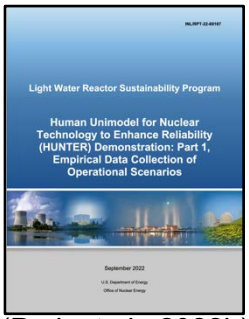
Initial standalone software demonstration of HUNTER



(Boring et al., 2022)

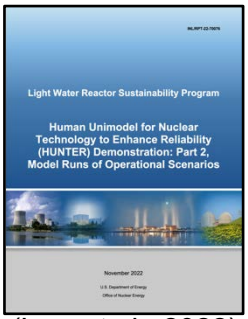
HUNTER 2.1

Data collection of human operators



(Park et al., 2022b)

New scenarios and simulator coupling



(Lew et al., 2022)

Figure 5. Evolution of HUNTER

An important feature of the HUNTER method 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. The PSFs impact human performance differently as time during a task progresses; so, including this timing structure can help understand more precisely when human error is more likely. The HUNTER method 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 making an error, it also provides a critical contextual data point which 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.

2.3 Previous Instances of HUNTER

Previous iterations of HUNTER were initially focused on scaffolding out a framework for dynamic HRA or integrating diverse tools into a common software platform (see Figure 5). A modular philosophy of integrating different modeling tools to create a functional dynamic HRA approach was key to the early instantiations of HUNTER and persists to the current instantiation. For example, while HUNTER is aligned to the SPAR-H HRA method, the architecture deliberately allows for use of different HRA methods that may prove better fits to particular analysis needs. Similarly, HUNTER exists in versions with different plant models used for the Environment module:

- A dummy-coded version that allows running without a plant model (HUNTER 2; Boring et al., 2022)
- A version linked to the Reactor Excursion and Leak Analysis Program (RELAP5-3D) thermal hydraulics code (HUNTER 2; Boring et al., 2022)
- A version linked to a simplified simulator for easier model development (HUNTER 2.1; Lew et al., 2022)

Fundamentally, the core architecture of HUNTER follows the conceptual framework established with HUNTER 2 (Boring et al., 2022), namely the three conceptual modules: the Individual, Task, and Environment. The Individual module serves to contain the various parts of the user which need to be modeled. This can include a cognitive model, PSFs, and other features of the specific human operator that is being modeled. Similarly, the Task module contains the aspects of the task that the human operator is to perform. This can contain the specific scenario that is being modeled, e.g., steam generator tube rupture, the specific procedures that need to be called, the steps of said procedures, and the GOMS-HRA primitives for the task. An important part of navigating tasks is determining deviations between work as intended and work as done (Ashour, Ashcroft, and Phipps, 2021), meaning HUNTER should model when humans correctly follow procedures and when they fail to follow procedures. The Environment module serves to contain the relevant plant model or system in which the operator is performing, external PSFs, any specific time limits, and other information relevant to the system context.

These modules serve as the high-level perspective of HUNTER. Specific software implementations and versions of HUNTER may shift the implementational details of HUNTER to suit particular analysis needs. Implementation modules go beyond the conceptual to the practical and logistics of the software itself. This report highlights a new implementation of HUNTER as an embedded tool to support PRAs using EMRALD. This implementation streamlines some features of HUNTER to allow incorporation within EMRALD's architecture and to facilitate ease of HRA model development for EMRALD users. It must be noted that EMRALD-HUNTER represents a subset of HUNTER features, and it is intended that further development will continue on the standalone version of HUNTER.

3. INTRODUCTION TO EMRALD

3.1 What is EMRALD?

EMRALD (Prescott, Smith, and Vang, 2018; Prescott, Nevius, Ma, and Lawrence, 2022) is a dynamic PRA tool developed by INL. EMRALD is a tool developed to model and analyze the sequence and timing of events that lead to specific outcomes in the context of dynamic PRA. It provides a simplified modeling process similar to existing static PRA modeling approaches but with a web-based graphical user interface that makes it easy for users to model and visualize complex interactions in dynamic PRA scenarios. Figure 6 shows an example of a model diagram in EMRALD with its easy-to-use flow diagram user interface. In addition to lowering the entry threshold for carrying out dynamic PRA, this tool allows users to couple the EMRALD models with other external codes.

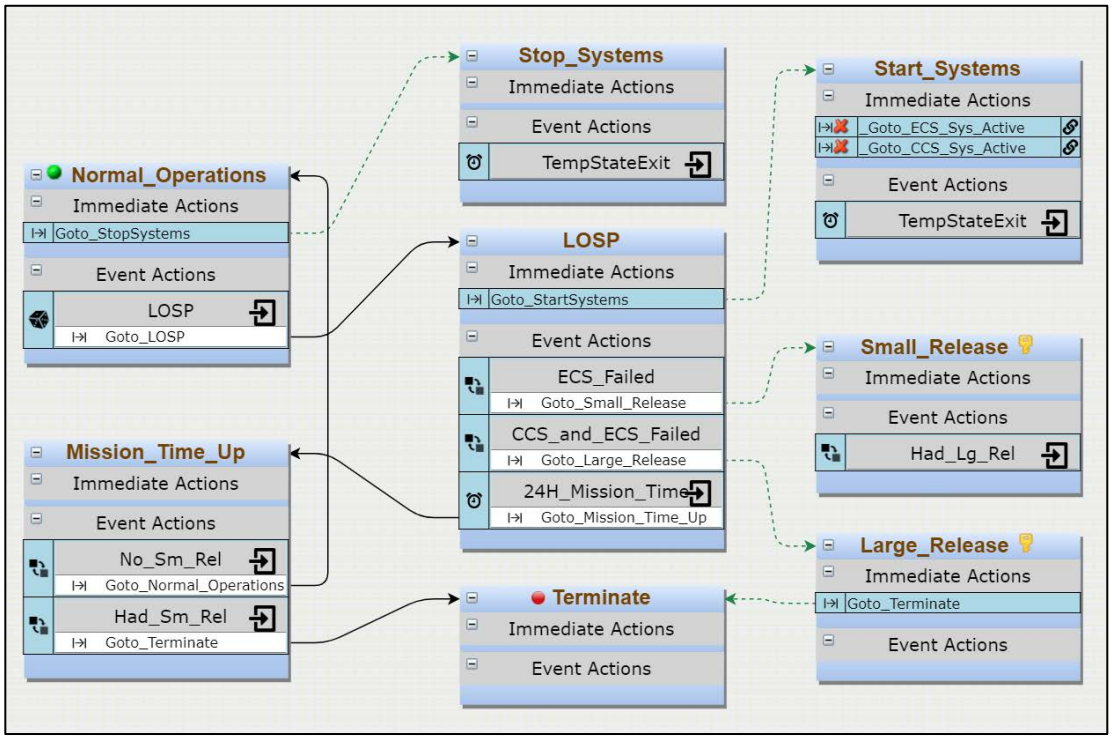


Figure 6. An example of the EMRALD model diagram

3.2 HRA Using EMRALD

EMRALD was developed primarily to support coupling PRA with physics-based tools to support time-based hazard scenarios such as flooding, but the EMRALD tool also includes useful functions suitable for implementing and exploring dynamic HRA. First, EMRALD supports dynamic modeling of human actions as they would actually be performed at NPPs in response to the changing conext of the plant. It simultaneously models the specific moment at which the action is performed, the time it takes to perform the action, and the failure probability of that action. Second, EMRALD allows evaluating timeline uncertainties of human actions based on Monte Carlo random sampling. Whether human actions can be completed within time constraints can be simulated and counted as overtime failure within EMRALD (Park et al., 2021; Park, Boring, and Ulrich, 2022).

There have been previous efforts to implement HRA models within the EMRALD tool. At the early stage of HRA research using EMRALD, two different approaches were suggested as summarized in Table 3. Procedure-based EMRALD modeling relies on procedural steps that describe the actions operators or plant personnel must perform in a given situation, while PRA/HRA-based EMRALD modeling makes the most of concepts and techniques that have been used in existing PRA and HRA. However, these two approaches feature a couple of limitations.

- Procedure-based EMRALD modeling does not communicate with PRA elements such as information on equipment status (i.e., operational or failed). In actual situations, the required operator actions may vary, depending on whether certain pieces of equipment remain operational. If the approach fails to consider components in PRA fault trees, it may be highly limited for evaluating various scenarios that lead to failure.
- For PRA/HRA-based EMRALD modeling, understanding how to assume time required for each basic event (i.e., HFE) and how to specifically model certain major HRA concepts (e.g., recovery opportunities) can be challenging and require more complex modeling or bloating of the model.
- Furthermore, the methods were tested using only a small subset of procedures. A method of modeling a larger collection of procedural steps that could be used in a scenario is not explicitly suggested.
- In addition, these modeling approaches do not fully consider PSFs, which influence human performance and are used to highlight error contributors and adjust basic HEPs. Adding PSF influences would require expert knowledge by the modeler and over complicate the model beyond what may be reasonable to expect of a probabilistic risk analyst without specific expertise in the nuances of human performance.

Table 3. Characteristics of two different EMRALD modeling approaches to dynamic HRA

	Procedure-based EMRALD Modeling (Ulrich et al., 2020)	**PRA/HRA-based EMRALD Modeling (Park et al., 2021)**
Description	Specifically models procedural contexts	Models basic events and HFEs already considered in PRA and HRA
Characteristics	Useful in accounting for context uncertainties that complicate HEP determinations	Within PRA/HRA modeling, it could be used to validate timeline uncertainties not covered in existing PRA/HRA

To handle the challenges posed by each approach and suggest a more structured and systemic method of analyzing human actions in the dynamic context, Procedure-based Risk Investigation Method – HRA (PRIME-HRA) and the Procedure-based Investigation Method of EMRALD Risk Assessment – HRA (PRIMERA-HRA) have been suggested (Park et al., 2021). The PRIME-HRA method provides guidance on implementing dynamic HRA, while the PRIMERA-HRA is the application of PRIME-HRA specifically within the EMRALD software. PRIME-HRA has contributed to the technical basis of the HUNTER tool, while PRIMERA-HRA has been used for understanding needs and requirements in dynamic HRA. Figure 7 summarizes the PRIME-HRA framework, which consists of four areas: (1) procedure-based task analysis, (2) task unit analysis for procedures applied to a given scenario, (3)

development of simulation models using dynamic risk assessment tools such as EMRALD and HUNTER, and (4) model analysis and integration into the PRA model.

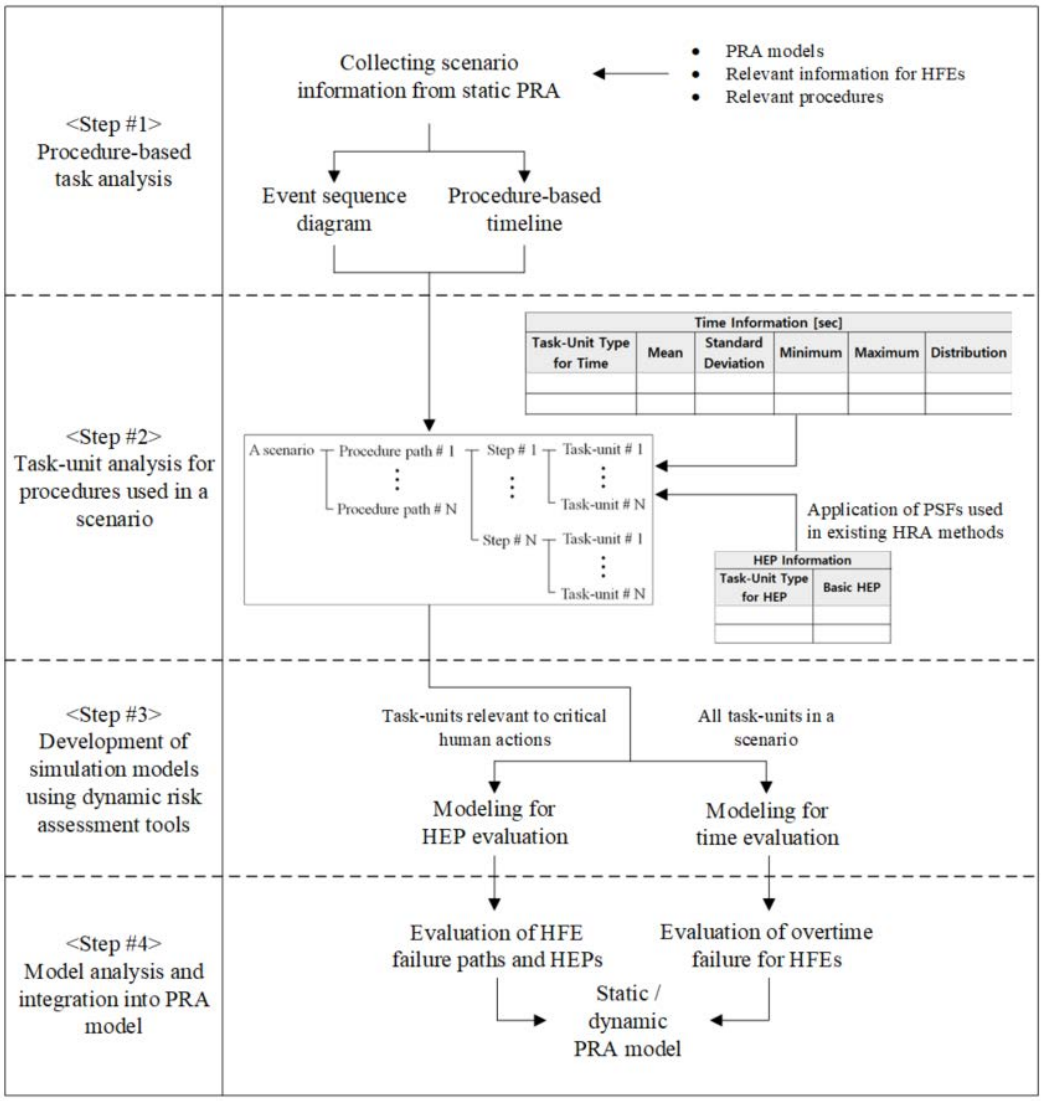


Figure 7. The PRIME-HRA framework

Regarding the first step, task analysis is the process of collecting and analyzing task-related information necessary for performing HRA. In this step, we collect the input data required for modeling procedures and implementing dynamic HRA. These data include static PRA models, information (e.g., PSF data) related to HFEs, and relevant procedures. We then develop an event sequence diagram and identify its actual timeline.

In the second step, the procedure paths in the event sequence diagram are decomposed to the task unit level. Basically, a procedure path consists of a couple of procedures that, in turn, include many procedural steps, each of which is comprised of a couple of task units. The task unit represents the procedure task type, as defined in the Human Reliability Data Extraction (HuREX; Jung et al., 2020) framework and

GOMS-HRA (Boring and Rasmussen, 2016). Time and HEP information are assigned for each task unit. In GOMS-HRA, the time information is assumed to follow a statistical time distribution whose mean value, standard deviation, and 5th and 95th percentile values are dependent on the particular task unit involved (Ulrich et al., 2017). The time data were collected through experiments involving actual operators at INL's Human Systems Simulation Laboratory (Boring, 2020), which was designed to conduct critical safety-focused human factors research. Depending on the general approach suggested in existing HRAs, HEPs are calculated based on the relationship between a basic HEP and the PSF multiplier values. In the present report, basic HEPs for task units were derived from the HuREX database. PSFs suggested by the SPAR-H method (Gertman et al., 2005) method were employed.

In the third step, simulation models are developed using dynamic risk assessment tools such as EMRALD and HUNTER. PRIMERA-HRA suggests detailed guidelines on how to develop simulation models using EMRALD. The simulation models developed based on these tools include all the information obtained from the previous steps, and are used for evaluating HEPs and time information for HFEs. Only task units relevant to critical human actions are used in the HEP evaluations, whereas the time information is evaluated for every task unit modeled in a given scenario.

In the final step, HFE failure paths, HEPs, and overtime failures for HFEs are evaluated. Those HFE failure paths that are based on cutsets generated from simulation logs explain why a given scenario is considered failed. These can be used to correct modeling errors in dynamic HRA tools. The HEPs generated are provided to the HFEs considered in static PRA models, or to account for human errors in dynamic PRA models. Evaluation of overtime failures for HFEs addresses whether the HFEs are completed within their allotted time windows. If not, this is considered a guaranteed failure (i.e., HEP = 1.0).

4. CONSIDERATIONS FOR INTEGRATING EMRALD AND HUNTER

4.1 Need for Integration

To properly evaluate human actions within a dynamic HRA model, it is important to provide sufficient contextual information describing the specific system or environment state at the moment that human actions are performed to understand the nature of human failures that may occur. In traditional static HRAs, information required for HRA processes is collected from procedures, structured interviews with knowledgeable experts, PRAs, and thermal hydraulics models. Procedures include detailed guidance on what operators or personnel need to do in most of situations that can happen in NPPs. Structured interviews are carried out to ask questions on things difficult for HRA analysts to understand or to collect plant-specific information such as time required to perform a human action. From the PRA side, event trees and fault trees provide specific scenarios where human actions are required, success criteria on human actions, or availability of systems in given scenarios. Lastly, thermal hydraulics analysis provides time windows of human actions (i.e., time constraints that operators need to finish their actions before plant states become irreversible) or plant parameters used by human reliability analysts to understand operational aspects in scenarios.

The mutual cooperation and communication between the HRA data providers above are necessary to successfully perform dynamic HRA. However, EMRALD does not have all of these required functions to support a more realistic representation of human error within dynamic HRA. While the current EMRALD allows for generic modeling, the tool mainly focuses on plant behavior, general operator actions and linking in external codes rather than HRA, because EMRALD was not originally developed for HRA. For this reason, our research team has coupled HUNTER and EMRALD to make them mutually complement one another in terms of dynamic HRA functions. This coupling adds simple methods for important HRA functions in EMRALD without extensive HRA background on behalf of the analyst. Equally importantly, it provides a streamlined approach to use HRA functions from the standalone version of HUNTER without some of the complexities involved with detailed dynamic HRA modeling. This approach proves more efficient than newly developing all dynamic HRA functions within EMRALD while ensuring PRA analysts can reasonably consider HRA without the need to master new tools like HUNTER. This combination enables small or simple HRA models to be combined under a simple EMRALD model, capturing a more realistic and detailed dynamic PRA and HRA.

4.2 Dynamic HRA Functions Currently Available in HUNTER and EMRALD

To couple EMRALD and HUNTER, it is necessary to understand what dynamic HRA functions can be included in each tool. Accordingly, this report investigates what functionalities are required for dynamic HRA and whether HUNTER and EMRALD can handle each function. Table 4 summarizes the list of necessary functions in dynamic HRA. The current availability of the functions within HUNTER and EMRALD is also summarized in the table. In the list, there are thirteen functions depending on the three areas of consideration, i.e., HRA, PRA, and thermal hydraulics codes and simulators. These functions are organically connected to reasonably generate outputs of dynamic HRA such as HEPs over time.

The HRA section consists of three subsections, i.e., task analysis, qualitative analysis, and quantitative analysis. Task analysis includes two functions, i.e., 1) analyzing and modeling human actions within the tool and 2) inputting data required for simulation. These functions are available in both HUNTER and EMRALD.

- Regarding the first function, HUNTER has a module (i.e., the Task module) for loading written procedures and automatically converting them into visual diagrams representing tasks written in procedures via the HUNTER user interface. Users can manually add or delete the diagrams or change the relationships through the interface. EMRALD also has its own user interface, which enables manually defining human actions into diagrams and relationship between the diagrams.

Table 4. List of requirements for dynamic HRA crosswalked to HUNTER and EMRALD

Section	Subsection	Requirement	HUNTER	EMRALD
HRA	Task analysis	Analyzing and modeling human actions within the tool	Yes	Yes
		Entering data required for simulation	Yes	Yes
	Qualitative analysis	Determination of a set of PSFs	Yes	No
		Evaluation of PSF levels	Yes	No
	Quantitative analysis	Calculation of HEPs	Yes	No
		Consideration of PSFs in dynamic context	Yes	No
		Time multiplier application	Yes*	No
		Evaluation of overtime failure	No	Yes
PRA	Event tree analysis	Modeling event sequences depending on success and failure of systems	No	Yes
	Fault tree analysis	Modeling failure of systems	No	Yes
		Entering failure data	No	Yes
Thermal hydraulics codes and simulators	Plant parameters	Coupling plant parameters with functions in the tool	Yes	Yes*

*Partially implemented or under development

- For the second function, HUNTER assigns nominal HEPs and time required for human actions depending on task types suggested in the GOMS-HRA method. In EMRALD, there is a function for entering probabilities in a diagram to move between diagrams. Using this function, users can use HEP values to model the progression probabilistically from one diagram to other diagrams. Analysts can use HEP values to model the success and failure of human actions. Also, EMRALD enables users to add a time distribution into each diagram. Time values obtained from the time distribution via the Monte Carlo sampling can represent time required for human actions.

Second, the quantitative analysis includes two functions, i.e., 1) determination of a set of PSFs and 2) evaluation of PSF levels.

- The HUNTER tool basically uses the eight PSFs suggested in the SPAR-H method, and users can select PSF levels for the SPAR-H PSFs via the HUNTER interface. HUNTER also includes provision for auto-calculating PSF levels.
- EMRALD does not provide any function to determine a set of PSFs and evaluate PSF levels within the tool.

Third, the quantitative analysis contains four functions, i.e., 1) calculation of HEPs, 2) consideration of PSFs in dynamic context, 3) time multiplier application, and 4) evaluation of overtime failure.

- The current HUNTER tool implements the first and second functions. Specifically, for the HEP calculation, HUNTER basically estimates HEPs by multiplying nominal HEPs assigned from the GOMS-HRA method and PSF multipliers on PSF levels evaluated in the qualitative analysis. Regarding the PSF application, HUNTER provides an option on how to apply PSFs in the calculation. Users can select the same approach to existing static HRA or mathematical models implementing PSF multiplier changes over time in the HUNTER interface. In addition, recent HUNTER research identifies the increase of time required for human actions depending on PSF levels. The time multiplier concept is currently under development.
- In EMRALD, evaluation of overtime failure is the only function available. This function is not currently implemented in the standalone version of HUNTER. Overtime failure refers to the failure caused when human actions are not completed within time constraints. If time required for human actions takes longer than their time constraints, it is assumed that human actions are guaranteed to fail. EMRALD enables users to add logic for evaluating overtime failure, then count it as a result of simulation.

The PRA section is composed of two subsections, i.e., event tree and fault tree analyses, which are the major modeling approaches in existing static PRA. These also play an important role in dynamic HRA and require three functions for implementing dynamic HRA.

- The first function related to event tree analysis is modeling event sequences depending on success and failure of systems. This function plays a role in defining specific environments and conditions at the moment human actions are carried out. In other words, it is used to differentiate scenarios, which require different mitigation strategies and human actions. Depending on scenarios, what human action needs to be analyzed in the dynamic context is determined. For example, after an initiating event, if a mitigation strategy is successful, the backup strategies would not need to be executed in the scenario. It means human actions for the first mitigation strategy are all needed to be analyzed, while other human actions included in the backup strategies do not need to be analyzed.
- The second function is modeling failure of systems. As a function stemming from fault tree analysis, it specifically models various causes leading to failure of systems such as failure to open a valve or run a pump. This includes entering failure data into the causes that have been modeled.

EMRALD supports such event sequence and fault tree modeling, while HUNTER does not provide these functions.

In the thermal hydraulics codes and simulator section, coupling plant parameters with the HRA and PRA functions is a main function required in dynamic HRA. This function plays a role in providing cues for human actions or determining entry conditions of mitigation strategies. Previous research regarding dynamic HRA (Boring et al., 2016) attempted to estimate effect of the complexity PSF based on plant parameters. As noted in Section 2.3, HUNTER includes several implementations of the Environment

module and readily supports coupling to plant simulations. EMRALD also supports coupling to thermal hydraulic simulations but does this on a model-by-model basis and does not automatically include such simulations.

4.3 Limitations of EMRALD for HRA

As introduced above, EMRALD is specialized in dynamic PRA, coupling plant behavior, general operator actions, and external simulation results, but only handles the functions related to HRA in a limited fashion. Some challenges of using EMRALD for HRA include:

- Users have to manually generate diagrams representing human actions and enter HEPs and time information in the diagrams. A procedure step includes multiple human actions, while a scenario requires a lot of procedure steps. Accordingly, modeling human actions in a scenario may need a lot of diagrams in EMRALD, causing additional model complexity.
- Unlike HUNTER, EMRALD does not have a function for automatically parsing procedures to model human actions. If there are many diagrams generated in an editing window, manipulating diagrams or loading the window in the interface slows down and makes HRA modeling difficult and complex.
- PSF level evaluation or HEP calculation are not available in EMRALD. Therefore, these need to be performed outside of EMRALD then added into EMRALD. Accordingly, within EMRALD, it is difficult to trace how HEPs and PSFs entered in diagrams have been evaluated.

These challenges should not be considered shortcomings of EMRALD, because EMRALD was not originally designed for detailed HRA applications. The purpose of this report and accompanying research is to redress such challenges and enable a more seamless handling of HRA within EMRALD.

4.4 Limitations of Standalone HUNTER for PRA

The current standalone version of HUNTER mainly concentrates on the functions in the HRA section in Table 4 but does not provide the functions related to PRA. As mentioned in the previous section, the PRA functions can be used for defining different scenarios and determining human actions needed to be analyzed.

- HUNTER mainly works on procedures, and the PRA functions can help HUNTER to navigate different scenarios like whether virtual operators keep performing the current procedure or whether they perform a contingency action transferring to other procedures.
- If you have scenarios with multiple operator procedures that affect each other and depend on plant conditions, you need a complex HUNTER model made by someone with expert knowledge in HRA and PRA.
- HUNTER does not provide a ready way to visualize event paths such as through event trees commonly used in PRA.

The current effort helps integrate HUNTER into broader PRA applications.

4.5 Advantages and Disadvantages of Integration

The biggest advantage of integrating HUNTER and EMRALD is to complement missing functions required for dynamic HRA and PRA. HUNTER can use dynamic PRA functions via EMRALD, while EMRALD acquires HRA support from HUNTER. This integration would be more efficient rather than newly adding the duplicated functions within HUNTER and EMRALD, respectively. This integration also allows a human reliability analyst to develop simple models for individual tasks, assuming required inputs are provided. Libraries of tasks can even be modeled and given to a probabilistic risk analyst. This

allows the analyst to do the overall modeling and just pull HRA pieces as needed, assigning the inputs. This also enables simpler modeling and validation of HRA pieces before use.

On the other hand, there may be some disadvantages of the integration such as running time increases due to the two-way communication between the codes or the inconvenience stemming from setting up a general software environment where both codes are simultaneously available. However, these may not be problems depending on how the codes are integrated. To minimize such disadvantages, the current effort focuses on embedding HUNTER functions into EMRALD. A tradeoff of this embedding is that the full suite of HUNTER functions are not transferred to EMRALD. However, these features are retained in the standalone HUNTER. Embedding HUNTER into EMRALD is an effective way to handle routine dynamic PRA applications needing HRA or specific EMRALD applications requiring greater HRA fidelity. For novel applications such as unexampled events or human factors design tasks that require a more in-depth understanding of human operational phenomena, standalone HUNTER remains the preferred analysis tool. Standalone HUNTER is designed for detailed HRAs and human performance modeling and retains those features where greater fidelity is required.

5. EMRALD-HUNTER IMPLEMENTATION

5.1 Embedding HUNTER into EMRALD

To perform a HUNTER HRA evaluation in EMRALD, a new type of state object needed to be added to EMRALD. An EMRALD model consists of diagrams made of different states. States have different types of events and actions that can occur depending on Monte Carlo sampling or conditions, but a full HRA event model is more than just an event or an action. This section explains the dynamic PRA simulation process in EMRALD and how HUNTER was integrated into it.

EMRALD has a process to execute an external code as an action called an External Sim Action, shown in Figure 8. This action allows the user to specify two scripts. The first script modifies or sets parameters for the code to execute. The second script can read results and shift EMRALD states depending on those results, driving the model. This is an advanced feature of EMRALD and requires considerable skill from the user. If there is a commonly used code EMRALD provides a way to simplify this process by adding a custom form in the user interface. For this, someone with web development skills and knowledge about the software to be executed can make a form that allows the user to simply drag-and-drop or select parameters to run the software. This was done for the Modular Accident Analysis Program (MAAP) thermal hydraulics tool MAAP (Prescott et al., 2022) and is shown in Figure 9. This option was first considered as a way to integrate HUNTER. However, a major benefit from HUNTER is the calculated time to perform an action. So, running HUNTER externally is not optimal because of the need to pass resultant values between HUNTER and EMRALD, a form of tight coupling described in Boring et al. (2023).

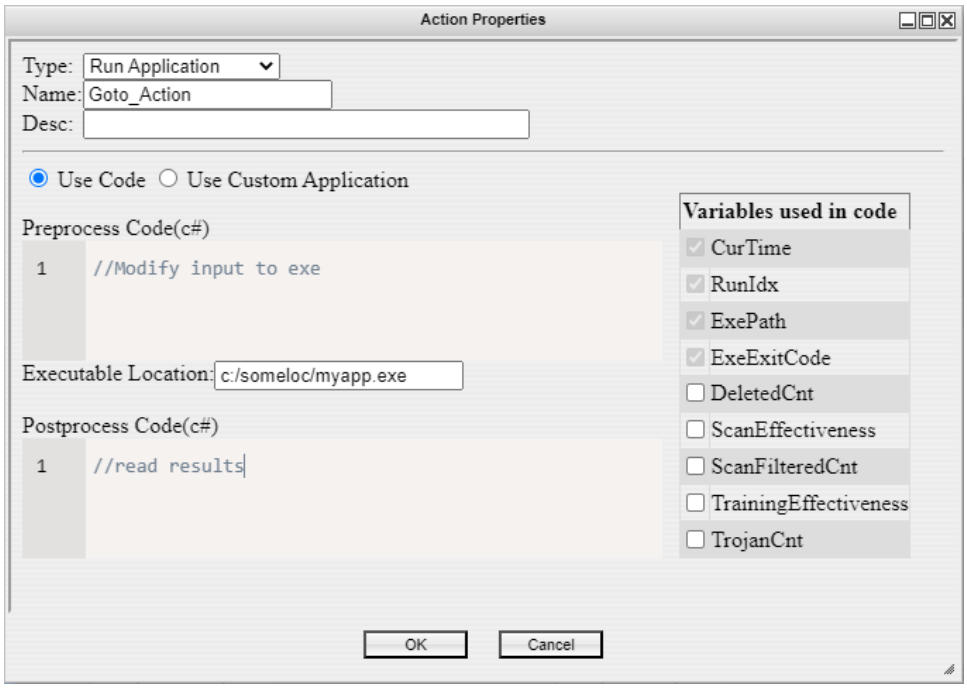


Figure 8. External code execution in EMRALD

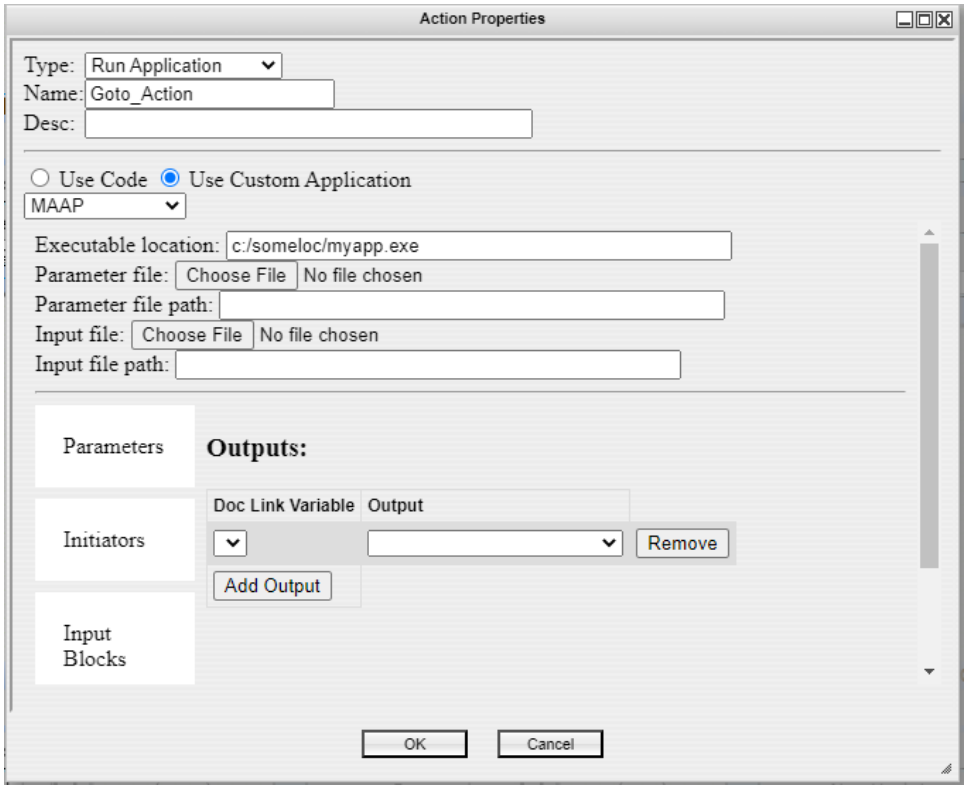


Figure 9. MAAP custom form for running an application in EMRALD

A more fitting approach is a HUNTER specific event in EMRALD. When an EMRALD model is being simulated, there are states that the simulation moves in and out of. As the simulation moves into a specific state, it knows the events that the state cares about. If those events occur while the simulation is in that state, then it executes the actions for that event. If that state is exited, then those events are no longer relevant and are no longer monitored. There are two categories of events in EMRALD. The first are condition-based events, where if that condition occurs then the event is immediately triggered, such as the value of a variable being true or false. The second are time-based events, where a Monte Carlo sample picks the time that event will occur, such as a distribution event.

```
public override TimeSpan NextTime(TimeSpan curTime)
{
  //Assign any data from the model before running the HRA code
  if(_hunterModel == null)
  {
    throw new Exception("Missing the hunter model");
  }

  //Setup the contextVarables
  _hunterModel._engine.SetContext(
    _contextVariables.ToDictionary(
      kvp => kvp.Key,
      kvp => kvp.Value.GetValue() as object
  ));

  //Call the HRA library to determine the time of the event
  TimeSpan retVal = _hunterModel._engine.EvaluateSteps(_hunterModel.Task.ProcedureCatalog, _procedureName, _startStep, _endStep);
```

Figure 10. HRA event code in EMRALD to call HUNTER

The return value of HUNTER would be similar to a Monte Carlo sampling of a distribution. This led to the creation of a new type of event in EMRALD. Each time-based event in EMRALD has a function that samples the time for the event. The standalone HUNTER software is written in the Python programming language, while EMRALD is written in the C# programming language. Therefore, the starting point for embedding HUNTER into EMRALD was porting the relevant subset of HUNTER code to C# so that it is compatible with the EMRALD codebase. For the HUNTER HRA event, EMRALD just calls the embedded HUNTER C# functions and returns the time and is shown in Figure 10.

We have embedded the HRA Engine of HUNTER into EMRALD as a .NET 6.0 library in C#. The EMRALD environment is entirely in .NET, and embedding HUNTER as a .NET library provides distinct benefits:

- Seamless integration—By porting the Python code to C#, HUNTER can achieve seamless integration with the existing EMRALD codebase, which allows for smoother communication between the modules.
- Better performance—Native C# code may perform better than Python code, especially if there are computationally intensive tasks or multiple interactions between the modules, because Python is interpreted code, while C# compiles natively as a binary application.
- Improved debugging—Debugging and stack tracing are much easier with a unified codebase, as HUNTER can use the same debugging tools and techniques as EMRALD.
- Reduced latency—Porting the code to C# eliminates the need for inter-process or network communication between the modules, which can reduce latency.

Implementing the HRA Engine in C# also provides an opportunity to re-work and re-think implementation strategies to reduce the overall complexity of the codebase while increasing some capabilities and maintainability of the codebase.

Running the HUNTER model also requires the EMRALD model to either contain or have a reference to the HUNTER model. It was decided to include the HUNTER model at the end of the EMRALD model, which is in JavaScript Object Notation (JSON) format, to simplify maintainability of the model. The HUNTER event also needs data to specify the procedure name, steps, and context links for adjusting the PSFs. An example of the model data format and values are shown in Figure 11.

It was also determined that the HUNTER HRA process would need to be able to specify which action would be taken when the event occurred. For example, the operator failing to perform the task is different than just the task having an infinite completion time. This means that an HRA outcome type must be linked to the desired action. Typically in EMRALD, an event triggers all the actions to occur that are under it. Another modification to EMRALD was needed to limit the actions according to the event results. To do this a new option was added to EMRALD simulation engine to allow an event to pick the actions available for execution.

It is important to note that the scope of the integration does not include all the original HUNTER functionality. Instead, a manageable set of HUNTER functionality was extracted and modified to achieve the coupling. As a result, the functionality documented here is not identical to that of the larger HUNTER effort behind the standalone tool and described in Boring et al. (2022) and Lew et al. (2022).

```
{
  "Event": {
    "id": 2,
    "name": "HRAProcedure",
    "desc": "",
    "mainItem": false,
    "evType": "etHRAEval",
    "moveFromCurrent": true,
    "procedureName": "sgtr",
    "startStep": 1,
    "endStep": 3,
    "contextLinks": [
      {
        "contextName": "StartShiftTimeH",
        "simVar": "StartTimeOnShiftH"
      },
      {
        "contextName": "ShiftTimeH",
        "simVar": "TimeOnShiftH"
      },
      {
        "contextName": "TaskAvailableTimeM",
        "simVar": "AvailableTimeM"
      },
      {
        "contextName": "Stress",
        "simVar": "Psf.Stress.Extreme"
      }
    ]
  }
},
```


Figure 11. A section of an EMRALD model in the JSON format specifying the HUNTER HRA event

5.2 Conceptual HUNTER Module Integration

The previous section provided background on how HUNTER was embedded into the EMRALD framework, including some changes to EMRALD to accommodate unique characteristics of human actions. Now, let us take a step back to effectively understand how HUNTER integrates into the encompassing EMRALD framework. It is first important to establish at a high, conceptual level how the context of the EMRALD model interacts with the HUNTER module. As can be seen in Figure 12 and Figure 13, two simplistic EMRALD models were constructed to represent loss of feedwater and steam generator tube rupture scenarios. The two EMRALD models for these scenarios are quite similar in that an initiating event leads to an HFE[a] for diagnosing the issue, and then if that human task is successful, the mitigation actions HFE is initiated to restore function for the affected plant components. EMRALD continues processing the encompassing model for the remainder of the simulation run. This is a simplistic model intended to show how HUNTER integrates with EMRALD, but in practice a more complicated EMRALD model would encounter failures related to the specific HFE in addition to other failure modes. In these more complicated models, the initial context conditions may be drastically different, and these varying contexts are passed to HUNTER as opposed to simple distribution sampling of time that occurs in simple models. Details for the HUNTER module integration are described next.

[a] Note that a human failure event is a scenario, not a guaranteed failure. An HFE has the opportunity for failure or success.

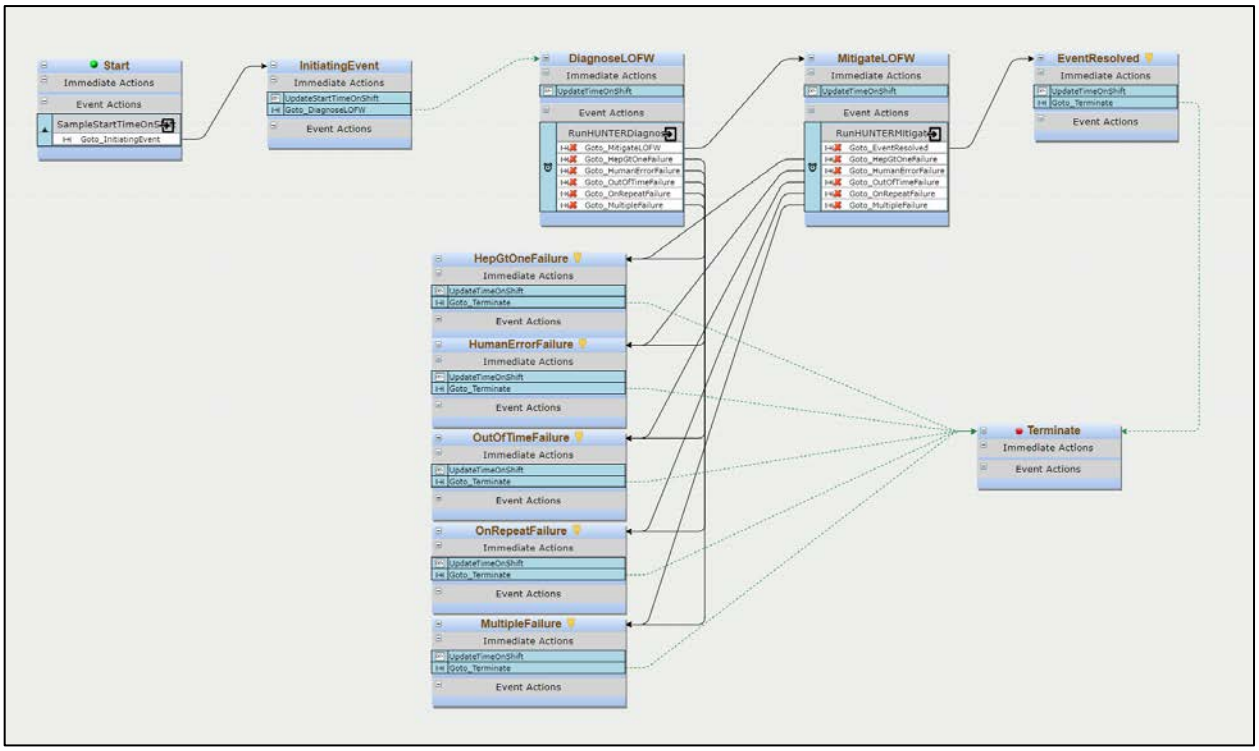


Figure 12. Loss of feedwater EMRALD mode

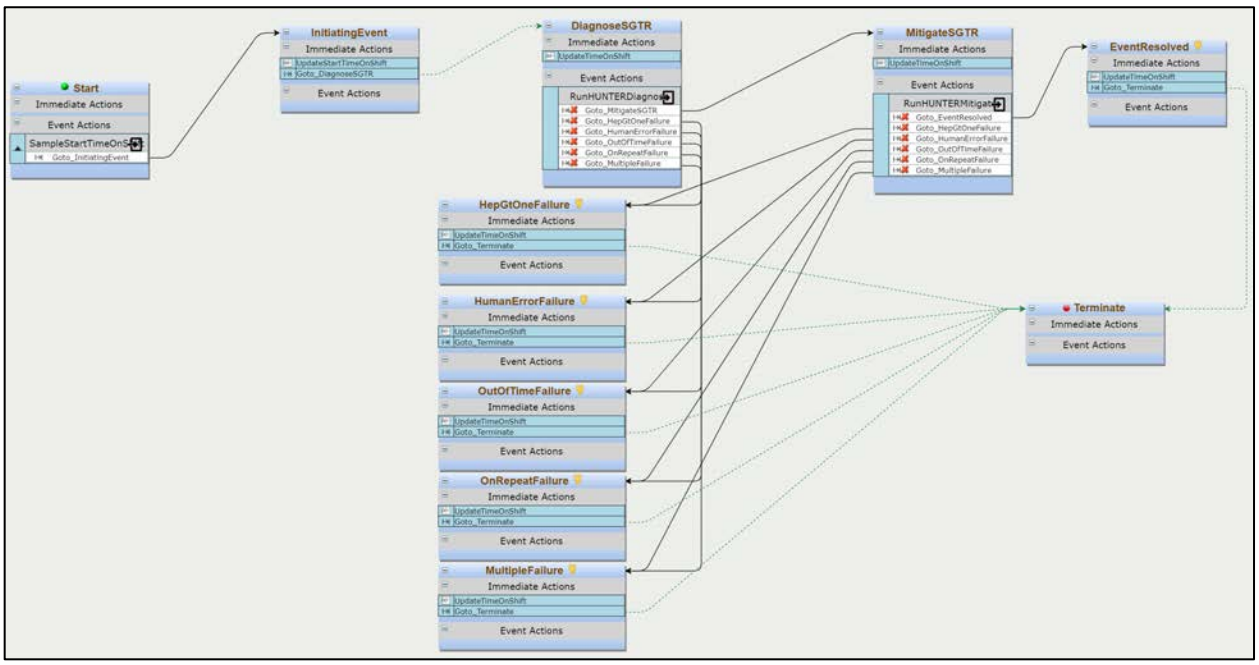
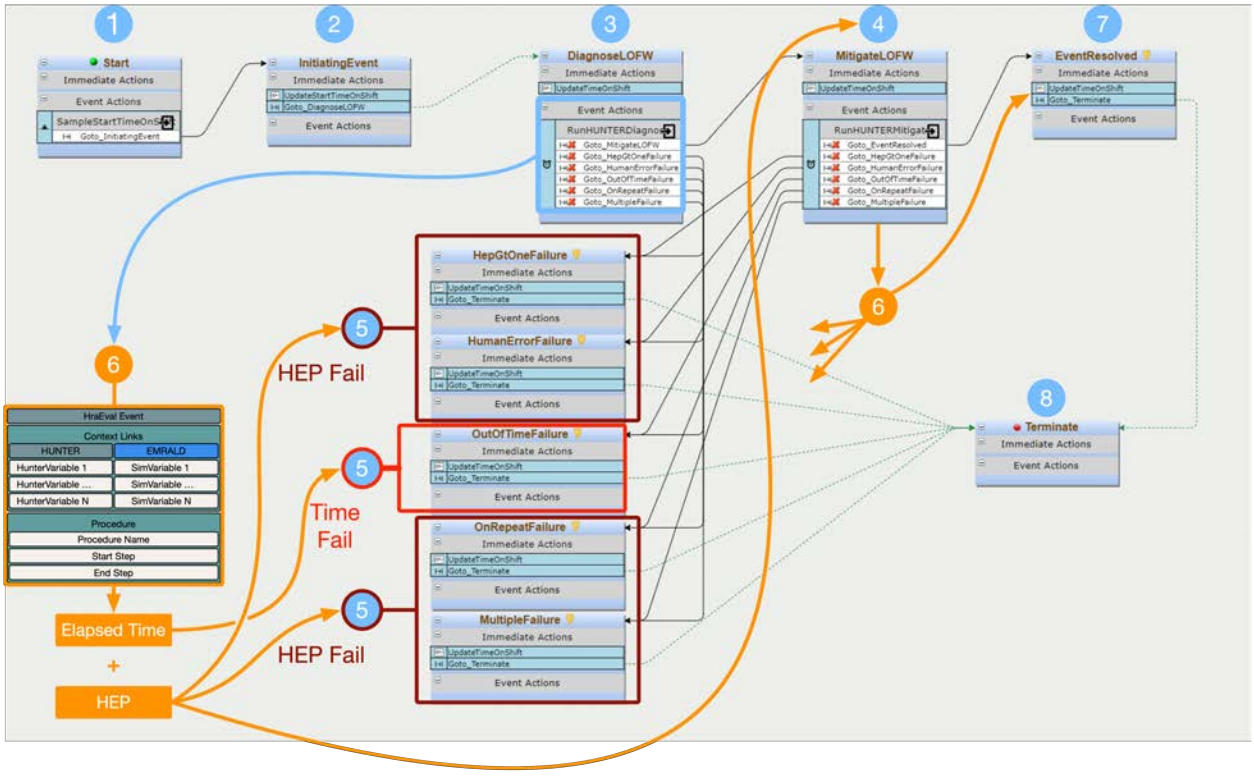


Figure 13. Steam generator tube rupture EMRALD model

Figure 14 contains the same loss of feedwater EMRALD model as depicted in Figure 12. However, Figure 14 also contains the additional custom time event objects developed to support HUNTER. A subsequent section will provide details on this new HRAEval event object created for EMRALD and its internal logic used to evaluate the HFE. For present purposes, it is sufficient simply to understand that this object represents the HUNTER module. This section will provide a detailed walkthrough of the model to explain the overall integration and will make use of the numbered items in Figure 14 to explain the flow of the model.

As with all EMRALD models, the simulation begins with a start state denoted with a 1 in Figure 14. In practice this start state would likely be in another higher-level model, and this other model would simply call the initiating event state labeled with a 2. The initiating event state samples an elapsed shift start time that is used in the HUNTER module to calculate fatigue and fitness for duty. Figure 14 numbered states 3 and 4 are the HFEs to *diagnose* and *mitigate* the loss of feedwater, which contain the actual HRAEval event object that executes the HUNTER module simulation. These two states are identical in structure but represent two distinct human tasks that must be completed in order, since the virtual operator must first diagnose the event as a loss of feedwater before the mitigation of the loss of feedwater can occur.



Numbers with blue backgrounds represent existing EMRALD elements, while numbers with orange backgrounds denote new objects developed to support the HUNTER module integration. The two shades of red depict HEP and time failure states calculated by the HRAEval event.

Figure 14. Loss of feedwater EMRALD model with HUNTER module elements overlaid

The HRAEval event logic is executed by the HUNTER module to return an elapsed time and an HEP of several possible types. The HRAEval event serially completes individual tasks contained within the specified procedure to generate a running total elapsed time. This process is explained in more detail later, but in general each GOMS-HRA primitive is evaluated to determine the elapsed time and HEP. Each primitive's elapsed time is added to the total task elapsed time, which is checked against the available time provided by the EMRALD model to determine if an overtime failure has occurred. It is important to note that this occurs at the GOMS-HRA primitive level. As each primitive's elapsed time is calculated, it is added to the total and compared against the available time *before* proceeding to the next primitive.

Each GOMS-HRA primitive is evaluated for task completion success or failure in Monte Carlo fashion by comparing the calculated HEP value to a randomly generated number between 0 and 1. If the sampled number is less than the calculated HEP value, the task is designated and coded as a human failure. The calculated HEP serves as an upper bound on the error rate. There are multiple ways in which a failure could manifest, which is why there are four possible HEP failure states represented by the states labeled with the blue background number 5 in Figure 14:

- PSF multipliers can result in the HEP exceeding a value of 1, which mathematically guarantees an HEP failure result with the state labelled "HEPGtOneFailure" or, in plain EMRALD influenced English, "HEP Go To One Failure". The standard HEP failure in which the calculated HEP exceeds the random generated number between 0 and 1 is the state labeled "HumanErrorFailure."
- Overtime failure in which the overall task time exceeds the time available is represented by number 5 state labeled "OutOfTimeFailure."
- HUNTER can be configured to allow each primitive to be repeated to represent a failed but recoverable action in which the virtual operator can repeat the GOMS-HRA primitive multiple times if the current execution is a failure. For example, this could represent the operator attempting to look for a particular value, elapsing the corresponding time, not finding it, and then looking again to find it. In previous HUNTER documentation, this has been referred to as the "time-debt" repeat method to represent one form of human error. With this mode enabled, the analyst can also designate the repetition limit for the GOMS-HRA primitives, with a repeat limit of three being the value selected for the data presented in the evaluation portion of this report. If the calculated HEP comparison to random number generated between 0 and 1 fails three times, then the transition to "OnRepeatFailure" state action is triggered.
- The final possible result state is labelled "MultipleFailure." The multiple failure can result from several different situations in which a combination of the three possible HEP failures occur in sequence. This can only occur with the repeat mode enabled; otherwise HUNTER would simply return with either "HEPGtOneFailure" or "HumanErrorFailure." It is important to note that HEP failure does not connotate an overtime failure, which will always be captured by the "OutOfTimeFailure" state. Future versions may be able to remove the "MultipleFailure" state outcome, but to ensure closure and HUNTER outcome resolutions it was included in this initial development effort as a catch all for potentially unforeseen modelling outcomes.

If the HRAEval event results in both an elapsed time within the available time limit *and* a successful HEP outcome occurs, then the action to transition to the next HFE to mitigate the loss of feedwater event is triggered. The mitigate loss of feedwater ("MitigateLOFW") state, denoted by a blue background number 4 in Figure 14, contains the HRAEval event, which is then executed in the same manner as was described above. The configuration of the mitigate event is different, and in turn HUNTER follows a different HFE, but the process is identical. The overall EMRALD model can fail here. Or, if the elapsed time is within the available time limit and the HEP outcome is successful, then the action to transition to

the “EventResolved” state, denoted by a blue background number 7, is triggered. “EventResolved” or any of the failure state outcomes triggered by the HRAEval event object result in updating the time on shift variable that captures the overall elapsed time for the model and immediately terminates by transition to the terminate state labeled as the blue background number 8 item.

The next section will describe the HRAEval event in more detail to explain how the context of encompassing EMRALD model and task configuration for an HFE is represented so that it can be passed to the HUNTER module for execution.

5.3 HRAEval Event Object

EMRALD has a number of events that analysts can use to build PRA models for a given scenario. However, none of the existing events provide the functionality required for the EMRALD-HUNTER interface. Therefore, a custom time event, termed the HRAEval event, was developed to accommodate the unique requirements for integrated HUNTER (see Figure 15). Specifically, this new custom HRA event provides several key features and functions that are described in the following section to support the EMRALD-HUNTER integration. The corresponding JSON file was previously shown in Figure 11.

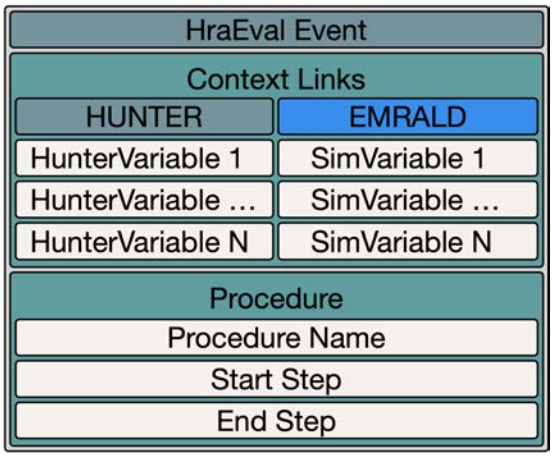


Figure 15. Conceptual representation of the EMRALD HRAEval event that provides the interfacing functionality between an EMRALD model and the HUNTER module

5.3.1 HRAEval Event Variable Exchange Functionality

The HRAEval event supports the ability to pass EMRALD variables to the HUNTER HRA engine through a context link dictionary that can be seen in Figure 15. The names of the variables are analyst defined and specific to a given EMRALD model. Therefore, the context link dictionary object provides the method to map these variables to the standard set of HUNTER HRA engine variables used internally to calculate PSFs that are applied to the GOMS-HRA primitive time distributions and HEPs. In practice, this is achieved by the analyst tagging each EMRALD model variable with an appropriate HUNTER HRA engine variable.

Conceptually, the variables can represent anything supported by EMRALD. HUNTER-relevant variables are envisioned to come from external plant model simulation parameters acquired from external simulation calls made through other event model objects in EMRALD. The initial effort presented in this report does not include an external plant model, due to ensuring simplicity and usability of the implementation. Instead, it defines HUNTER relevant variables within the EMRALD model as

EMRALD sampled variables. There are nine EMRALD variables exchanged with the HUNTER module as can be seen below in Table 5.

The outcome variables calculated by HUNTER are passed back to EMRALD by leveraging the action object within EMRALD. The HUNTER module simply triggers an action to transition to the appropriate outcome state. These outcome states can then be used as a standard EMRALD object to call other events.

In addition to serving as the conduit to exchange variables, the HRAEval event also serves as the mechanism for the analyst to define the HFE task in the form of a procedure name string, a starting step, and an ending step. The HRA module contains a small suite of prepopulated procedures. For flexibility, the analyst specifies a starting and ending step so that only the relevant sections of the procedure are performed. In its current form, the EMRALD-HUNTER codebase does not include the standalone HUNTER functionality to evaluate the logic within the procedure and adjust course through the procedure. The logic of the procedure is not evaluated, though prior versions of HUNTER have the functionality to allow the state of the plant in conjunction with the outcome of the virtual operator performing tasks to dictate the path through the procedure. Instead, in this instantiation, the procedure is treated simply as an ordered sequence of steps, with each step containing one or more GOMS-HRA primitives representing the actual tasks performed by the virtual operator to achieve the goals of each individual step and procedure. An example of the procedure represented a JSON object can be seen in the next section.

Table 5. Variables exchanged by EMRALD and HUNTER

Variable	HUNTER Input	Hunter Output	Description
AvailableTime	Yes		Available Time to perform the EMRALD run before failure
TimeRequired	Yes		Time needed for operator to complete task (used to calculate Available Time PSF)
StartTimeOnShift	Yes		The time span an operator has been on shift used to calculate fatigue and fitness for duty PSFs
TimeOnShift	Yes	Yes	EMRALD current simulation time including HRAEval time
HepGtOneFailure		Yes	Calculated HEP > 1 ensuring task failure
HumanErrorFailure		Yes	Calculated HEP > random generated sample resulting in task failure; only occurs if repeat mode is disable
OutOfTimeFailure		Yes	While executing HUNTER simulation the available time was exceeded
OnRepeatFailure		Yes	The maximum number of repeats was attempted and all failed; only occurs if repeat mode is enabled
MultipleFailure		Yes	A combination of failures above were triggered

5.3.2 HRAEval Event HRA Engine Logic

As a starting place, the procedure is represented simply as a sequence of steps, which must all be executed in series to successfully resolve the initiating event. In their current form these steps directly contain GOMS-HRA primitive designators in the form of two letter string codes (see Table 1 in Section 2.2.3). The primitive designators come from the HUNTER framework and conceptually can be viewed as a collection of the smallest basic human tasks such as acquiring information, reading a procedure step, and executing a control actuation. For the purposes of this development and evaluation effort, the GOMS-HRA primitives can simply be viewed as the nominal time and HEP distributions associated with each step of the procedure. For each GOMS-HRA primitive in the model, the nominal time and HEP are sampled and then modified by associated PSFs. Figure 16 below shows how the procedure is executed.

```
1. function HraEval(ProcedureName, StartStep, EndStep)
2.     Procedure = GetProcedure(ProcedureName)
3.     ElapsedTime = 0
4.     foreach (Step in Procedure[StartStep, EndStep])
5.         foreach (GOM in Step)
6.             RelevantPSFs = GetRelevantPSFs(GOM)
7.             ElapsedTime += FatigueMultipler +
                   RelevantPSFs.TimeMultiplier * GOM.SampledTime()
8.             if ElapsedTime > AvailableTime
9.                 return ElapsedTime, EvalState.OutOfTimeFailure
10.            AdjustedHEP = RelevantPSFs.AggregateMultiplier * GOM.Nominal
11.            if AdjustedHEP > 1
12.                return ElapsedTime, EvalState.HepGtOneFailure
13.            Success = Random() > AdjustedHEP
14.            if not Success and RepeatMode
15.                for i in range(MaxRepeatCount)
16.                    ElapsedTime += FatigueMultipler +
                          RelevantPSFs.TimeMultiplier * GOM.SampledTime()
17.                    if ElapsedTime > AvailableTime
18.                        return ElapsedTime, EvalState.OutOfTimeFailure
19.                    AdjustedHEP = PSFs.AggregateMultiplier * GOM.Nominal
20.                    Success = Random() > AdjustedHEP
21.                    if Success
22.                        break
23.            if not Success
24.                return ElapsedTime, EvalState.OnRepeatFailure
25.    if Success
26.        return ElapsedTime, EvalState.Success
27.    else
28.        return ElapsedTime, EvalState.HepFailure
```

Figure 16. Pseudocode for the procedure execution representing the central functionality of the HUNTER HRA engine used to calculate human success or failure and the elapsed time for each task

HRAEval is intended to allow users to specify partial tasks within EMRALD like diagnosing a steam generator tube rupture or taking mitigating actions. The HRAEval event allows users to specify a procedure name, the starting step, and an ending step so that users can decide the level of granularity they need in their model down to a single step (see Figure 17). HRAEval also allows users to define contextual variables including available time, time required, time on shift, and static PSF variables to be used by the HRA Engine for calculating PSF time and HEP multipliers and ultimately elapsed time for the event and success and failure of the event.

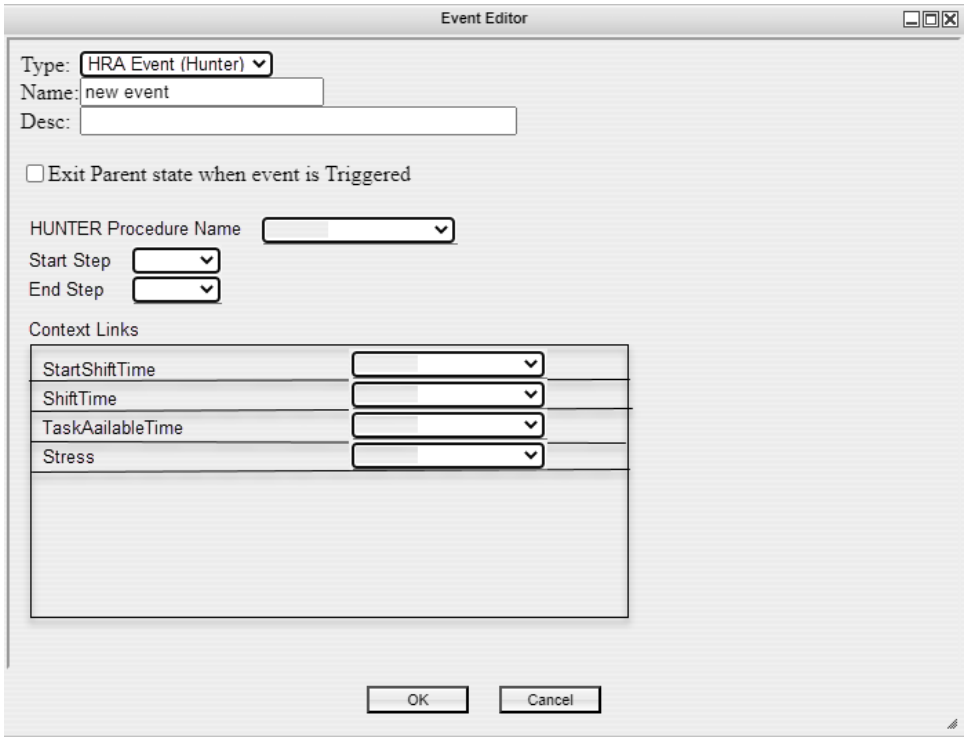


Figure 17. EMRALD-HUNTER interface for predefined procedures

6. DYNAMIC PERFORMANCE SHAPING FACTORS IN EMRALD-HUNTER

6.1 Introduction

HUNTER was originally conceived as a dynamic implementation of the static SPAR-H HRA method (Boring et al, 2017). The HUNTER framework has since made great strides in its ability to use PSFs. Particularly innovative is a first-of-a-kind use of PSFs to influence not only the HEP but also task time. Previous versions of HUNTER (e.g., Boring et al., 2022) had some experimental work for using PSFs for task duration but did not in practice use PSFs for calculating time. With HUNTER embedded in EMRALD, we have a fully implemented version of the eight SPAR-H PSFs. We model four PSFS—Fitness for Duty, Stress, Available Time, and Experience and Training—dynamically. In Section 6.3 we describe Fitness for Duty; in Section 6.4, Stress; and in Section 6.5, Experience and Training. First, in Section 6.2, we discuss the dynamic treatment of the PSF for Available Time. In the interest of a parsimonious proof of concept for making the SPAR-H PSFs dynamic, we sought to restrict modeling to the original eight SPAR-H PSFs. However, as might be expected within dynamic HRA and the treatment of time, there remain some concepts that do not transfer between static and dynamic HRA. Thus, the next section introduces Time Pressure as an extension to the original PSF list. The remaining four PSFs are treated as static PSFs that can be specified by the analyst. Each of these PSFs has a Diagnosis and Action variant, corresponding to the separate treatments in SPAR-H for cognitive vs. behavioral tasks, respectively. The levels of these variants are consistent across the factors, but sometimes the multipliers for Diagnosis factors are higher than their Action counterparts (Boring and Blackman, 2007).

Figure 18 outlines the types of dynamic PSF configurations that are possible in HUNTER. PSFs may be either manually assigned (akin to static HRA) or automatically assigned for their initial state. They may then experience a dynamic progression such as adjusting for lag and linger (i.e., delay and decay functions) over time (Boring, 2015; Park, Boring, and Kim, 2019). They may also be automatically calculated based on emerging conditions (Boring et al., 2017). Figure 18 shows an example where PSFs are automatically assigned at the onset of the HFE scenario run and then updated dynamically for context. In practice, in EMRALD-HUNTER, PSFs are assigned manually by the analyst at the scenario outset and then updated for lag and linger functions. Table 6 shows how PSFs are determined in EMRALD-HUNTER, while Table 7 shows how PSFs map to GOMS-HRA primitives, since every primitive is not relevant to every PSF.

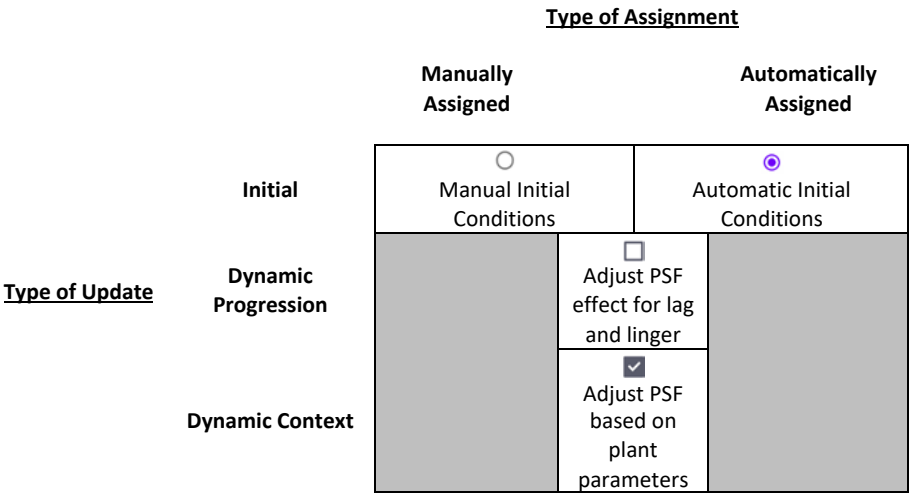


Figure 18. Types of PSF assignments possible in HUNTER

Table 6. Treatment of PSFs in EMRALD-HUNTER

Factor	Operation	ID	Levels (Multipliers)	Static
AvailableTime	Action	ATa	InadequateTime (9999), BarelyAdequateTime (10), NominalTime (1), ExtraTime (0.1), ExpansiveTime (0.01)	
AvailableTime	Diagnosis	ATd	InadequateTime (9999), BarelyAdequateTime (10), NominalTime (1), ExtraTime (0.1), ExpansiveTime (0.01)	
Complexity	Action	Ca	HighlyComplex (50), ModeratelyComplex (20), Nominal (1), ObviousDiagnosis (0.01)	Yes
Complexity	Diagnosis	Cd	HighlyComplex (5), ModeratelyComplex (2), Nominal (1), ObviousDiagnosis (0.001)	Yes
ErgonomicsHMI	Action	Ea	MissingOrMisleading (50), Poor (20), Nominal (1), Good (0.1)	Yes
ErgonomicsHMI	Diagnosis	Ed	MissingOrMisleading (5), Poor (2), Nominal (1), Good (0.01)	Yes
ExperienceAndTraining	Action	EaTa	Low (10), Nominal (1), High (0.5)	
ExperienceAndTraining	Diagnosis	EaTd	Low (3), Nominal (1), High (0.5)	
FitnessForDuty	Action	FfDa	-	
FitnessForDuty	Diagnosis	FfDd	-	
Procedures	Action	Pa	NotAvailable (100), Incomplete (50), AvailableButPoor (20), Nominal (1), DiagnosticOrSymptomOriented (0.1)	Yes
Procedures	Diagnosis	Pd	NotAvailable (10), Incomplete (5), AvailableButPoor (2), Nominal (1), DiagnosticOrSymptomOriented (0.01)	Yes
Stress	Action	Sa	Extreme (50), High (20), Nominal (1)	
Stress	Diagnosis	Sd	Extreme (5), High (2), Nominal (1)	
WorkProcesses	Action	WPa	Poor (5), Nominal (1), Good (0.05)	Yes
WorkProcesses	Diagnosis	WPd	Poor (5), Nominal (1), Good (0.05)	Yes

Table 7. Relationships between GOMS-HRA operations and PSFs in EMRALD-HUNTER

Operation	Sub Operation	Relevant PSFs
Action	controlRoom	Sa, ATa, Ca, Ea, EaTa, FfDa, Pa, WPa, TPa
Action	field	Sa, ATa, Ca, Ea, EaTa, FfDa, Pa, WPa, TPa
Checking	controlRoom	Sa, ATa, Ca, Ea, EaTa, FfDa, Pa, WPa, TPa
Checking	field	Sa, ATa, Ca, Ea, EaTa, FfDa, Pa, WPa, TPa
Retrieval	controlRoom	Sa, ATa, Ca, Ea, EaTa, FfDa, Pa, WPa, TPa
Retrieval	field	Sa, ATa, Ca, Ea, EaTa, FfDa, Pa, WPa, TPa
InstructionCommunication	produceWrittenOrVerbal	Sa, ATa, Ca, Ea, EaTa, FfDa, Pa, WPa, TPa
InstructionCommunication	receiveWrittenOrVerbal	Sa, ATa, Ca, Ea, EaTa, FfDa, Pa, WPa, TPa
Selection	controlRoom	Sa, ATa, Ca, Ea, EaTa, FfDa, Pa, WPa, TPa
Selection	field	Sa, ATa, Ca, Ea, EaTa, FfDa, Pa, WPa, TPa
Decision	basedOnProcedures	Sd, ATd, Cd, Ed, EaTd, FfDd, Pd, WPd, TPd
Decision	withoutProcedures	Sd, ATd, Cd, Ed, EaTd, FfDd, WPd, TPd

6.2 Dynamic PSF for Available Time and Time Pressure

The level for the Available Time PSF is calculated dynamically using the traditional approach of SPAR-H when both time available and time required are specified. When these attributes are not provided, the Available Time PSF is set to nominal. However, it is important to note that having a dynamic HRA system that calculates traditional categorical PSF levels for Available Time should be used with caution. If we step back, the purpose of the Available Time PSF is to capture failures due to running out of time. In fact, in SPAR-H, when there is inadequate time, the overall HEP is set to 1.0 (see Table 8). However, with HUNTER we explicitly track time with every run, and if Available Time is specified, HUNTER has internal logic to fail the task when the task duration (i.e., Time Required) exceeds Available Time. Therefore, care should be taken to avoid double penalizing tasks by setting Time Tequired equal to or greater than Available Time. We have elected to not co-opt the common notion of the Available Time PSF. Available Time is implemented and calculated when Available Time and Time Required are provided to HRAEval events as part of the HFE context.

Table 8. Available time PSF levels and multipliers in SPAR-H

Task Type	Available Time PSF Level	Multiplier Value
Diagnosis	Inadequate Time	HEP = 1.0
	Barely Adequate Time	10
	Nominal Time	1
	Extra Time	0.1
	Expansive Time	0.01
Action	Inadequate Time	HEP = 1.0
	Time Available = Time Required	10
	Nominal Time	1
	Time Available ≥ 5 x Time Required·	0.1
	Time Available ≥ 50 x Time Required·	0.01

Empirical data have shown that licensed operators can perform procedures in a slow and cautious manner as well as more expedient but careful manner. For this reason the ability to provide Time Pressure to expedite the pace of procedures is necessary to match empirical data. We have elected to include a ninth PSF for Time Pressure. The Time Pressure PSF only affects the aggregate PSF time multiplier and does not affect the HEP multiplier. Table 9 below lists the PSF factors that impact time.

Table 9. PSFs with time multipliers in HUNTER

Factor	Operation	ID	Level (Time Multiplier)	Static
ExperienceAndTraining	Action	EaTa	Low (3)	Yes
ExperienceAndTraining	Diagnosis	EaTd	Low (3)	Yes
TimePressure	Action	TPa	High (0.5), Nominal (1)	Yes
TimePressure	Diagnosis	TPd	High (0.5), Nominal (1)	Yes

Time Pressure is needed to calibrate the pacing of HUNTER. In a previous effort (Lew et al., 2022), we used HUNTER to model task completion times for a loss of feedwater scenario and startup scenario with the Rancor Microworld Simulator (Ulrich et al., 2017). The estimated task times for startup, a normal operating procedure, where very close between HUNTER and Rancor. However, HUNTER took nearly twice as long as operators in completing the loss of feedwater scenario. This suggests that operators are capable of expediting their pace when necessary. This effect is captured with the Time Pressure PSF.

6.3 Dynamic PSF for Fitness for Duty

6.3.1 Introduction

SPAR-H (Gertman et al., p. 25) defines Fitness for Duty as:

> whether or not the individual performing the task is physically and mentally fit to perform the task at the time. Things that may affect fitness include fatigue, sickness, drug use (legal or illegal), overconfidence, personal problems, and distractions. Fitness for duty includes factors associated with individuals, but not related to training, experience, or stress.

For the purposes of HUNTER, the dynamic treatment of the Fitness for Duty PSF is calculated according to two dimensions—Fatigue and general Fitness for Duty—which highlight physical and mental decrements, respectively.

6.3.2 Fatigue

As discussed, the Experience and Training and Time Pressure PSFs have levels with time multipliers. In aggregate these are used to scale the time sampled from the GOMS Task Primitive distributions. Time is also influenced by operator fatigue. In previous iterations of HUNTER we used the third order polynomial function presented in Figure 19. A curve-fitted equation was presented based on the data of Folkard (1997). Equation 1 can evaluate the relative fatigue index according to the hour on duty.

$$y = 0.0054\ x^3 - 0.0939x^2 + 0.4271x + 0.599\ (R^2 = 0.6912) \quad (1)$$

where y is the fatigue index and x is the number of hours on duty.

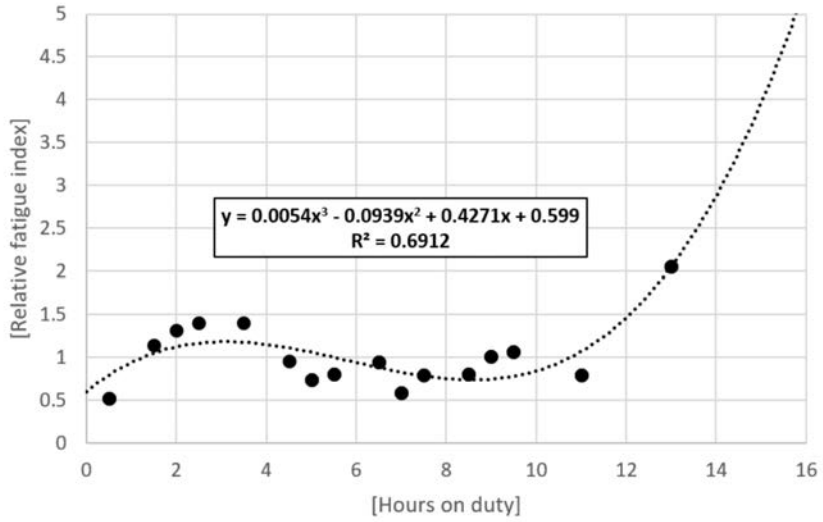


Figure 19. Time dependent 3rd order polynomial fatigue index based on Folkard (1997)

The fatigue index is used as a time multiplier in HUNTER. A notable shortcoming of this model is the exponential growth after 10 or so hours on shift. The last fitted data point is at 13 hours. If left unabated, task time estimates are 9.36 times slower if operators are 18 hours on shift. While 18 hours is an unreasonably long shift duration, if that were to occur, the time decrement is abnormally large and not empirically justified. Generally speaking, fitted models, especially those that grow exponentially, shouldn't be trusted outside of their range. The fatigue index is meta analytic modeled from datasets examining fatigue (Folkard, 1997). Studies have shown that fatigue is influenced by numerous factors like shift duration, average duty cycle of shifts (e.g., working 3 consecutive 12-hour shifts), cumulative components, and time working on shift (Boring et al., 2020). Fatigue is also influenced by circadian arousal cycles. The data illustrate that the circadian cycle has a peak amplitude of around 50%. The data also illustrate that on average fatigue is 100% (relative fatigue index of 2) slower at 13 hours. The relative fatigue then roughly doubles over 2 hours (see Figure 19). When we take a closer look at the Folkard (1997) dataset we see standard errors of +/- 0.25, suggesting a fair amount of individual variation (see Figure 20).

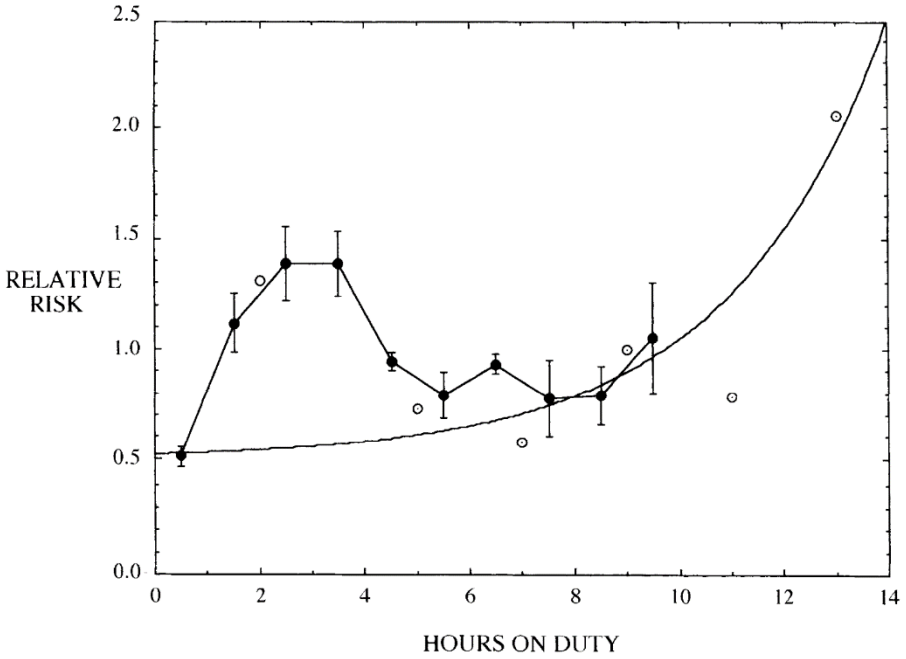


Figure 20. Fatigue risk rates with standard error from Folkard (1997)

To accommodate for individual variability, the polynomial model has been adapted to a factorial model that simulates a circadian phase, followed by a baseline phase, and finally a fatigue onset phase in Figure 21. The revised model has an R-squared over the same datapoints of 0.81 compared to the third order polynomial model in Equation 1, with $R^2 = 0.6912$. But, more importantly, the revised model randomizes baseline fatigue, peak fatigue, time to fatigue onset, fatigue transition time, circadian amplitude, and circadian phase by sampling normal distributions to mimic human variability.

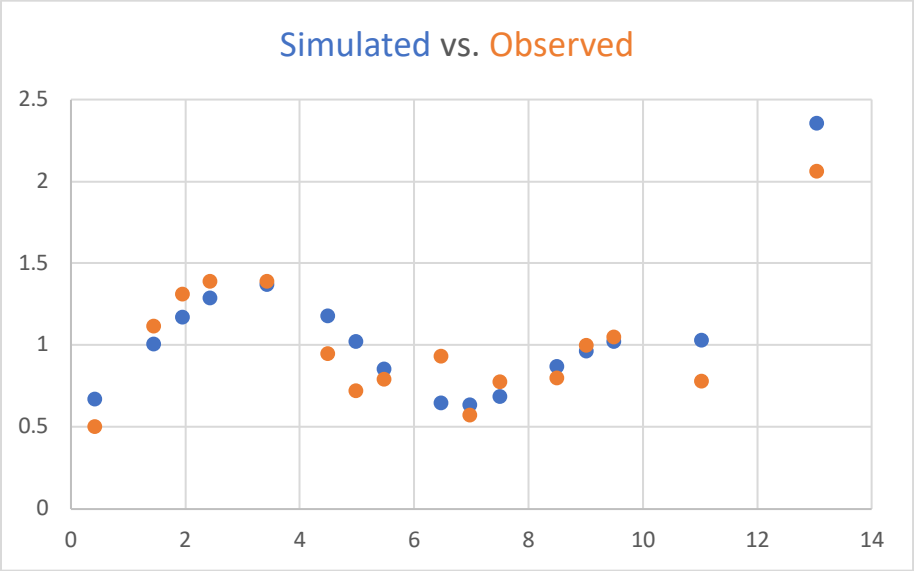


Figure 21. Simulated fatigue index values from revised factorial fatigue index model in blue and observed values from Folkard (1997) in red

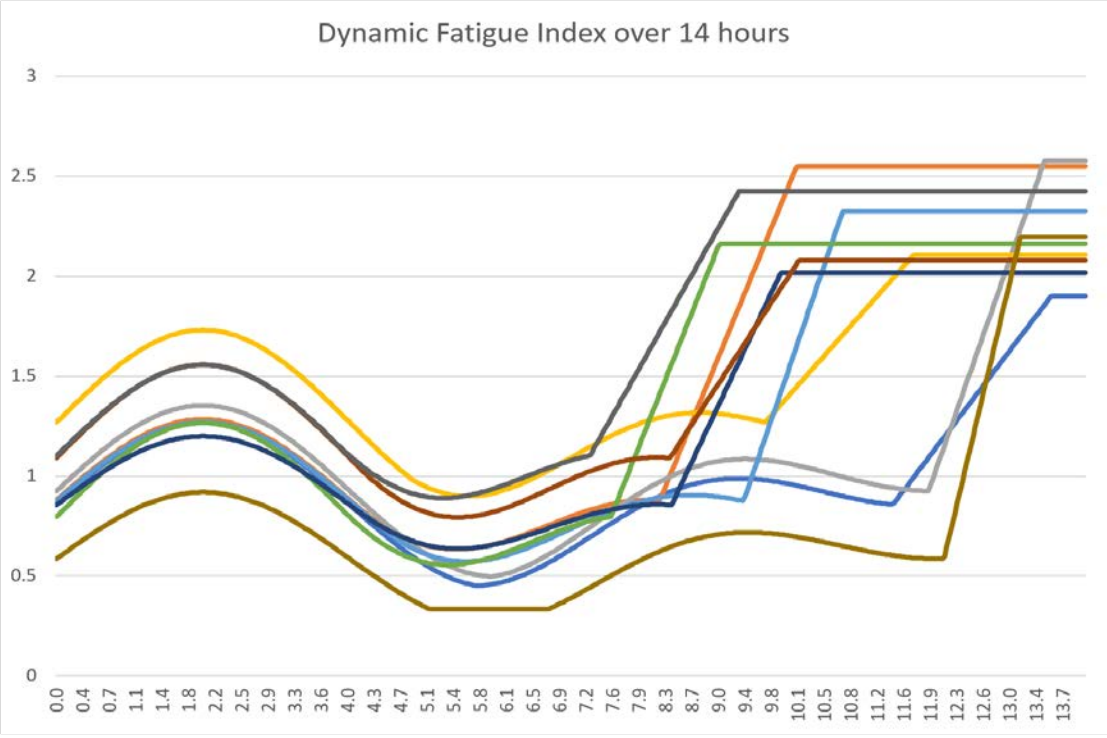


Figure 22. Ensemble plot of dynamic fatigue curves generated from stochastically setting revised fatigue model parameters

Figure 22 has an ensemble of ten randomly generated fatigue models. The models have varied fatigue baselines, fatigue onset times, as well as peak fatigue levels after which the model holds steady. The multiplier is bounded by 0.3333 to prevent abnormally fast execution (3x normal speed). The C# implementation of the revised model is in Figure 23. In summary, time in HUNTER is influenced by both PSF time multipliers and the fatigue index.

```
private double CalculateAdjustedTime(Primitive primitive, double sampled_time)
{
    double adjusted_time = sampled_time;

    if (psfCollection != null)
    {
        double psf_time_multiplier = psfCollection.TimeMultiplier(primitive) ?? 1.0;
        adjusted_time *= psf_time_multiplier;
    }

    if (TimeOnShiftFatigueEnabled)
    {
        adjusted_time *= FatigueIndex;
    }

    return adjusted_time;
}
```

Figure 23. Function that calculates adjusted time in HUNTER as a function of fatigue

6.3.3 Fitness for Duty

The fatigue index attempts to model decrements in the time required to complete tasks, but does not directly account for accuracy or cognitive effects. If operators do have to work beyond 14 hours how is their cognition likely to be impacted? A well-documented psychological phenomenon is the speed-accuracy tradeoff (Heitz, 2014), whereby as speed increases, accuracy declines. In terms of fatigue, accuracy reflects the cognitive effects. From Belenky (1994) we know that accuracy does decline with sleep deprivation, but at gradual pace out to 70 hours. Figure 24 depicts the decline in speed-accuracy per Belenky, and Figure 25 provides an ensemble plot of speed-accuracy from HUNTER. The fatigue index is inverse speed. So, by multiplying speed-accuracy by the fatigue index, we can obtain accuracy. Accuracy is the inverse of error rate, which is exactly what PSF multipliers are estimating. This means we

can model accuracy—the inverse of human error—using the fatigue index and the speed-accuracy. First, we fit a second order polynomial fit to the decay curve. Then a dynamic Fitness for Duty PSF multiplier can be calculated as shown in Figure 26.

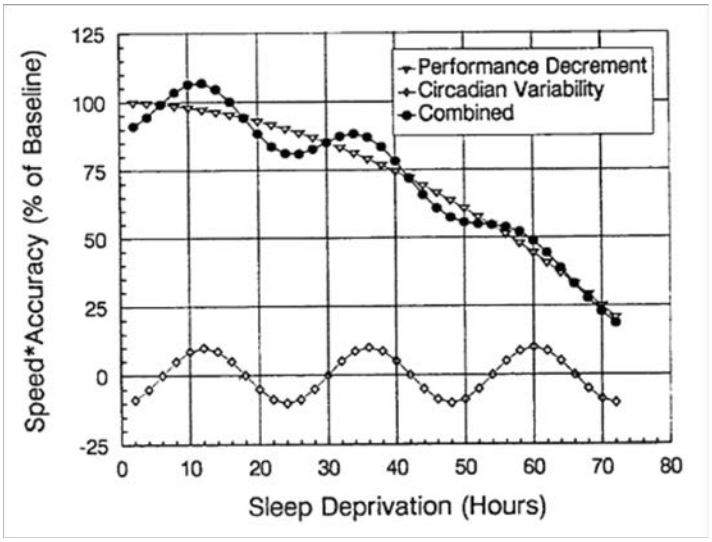


Figure 24. The impact of sleep deprivation on cognitive performance out to 72 hours (from Belenky, 1994)

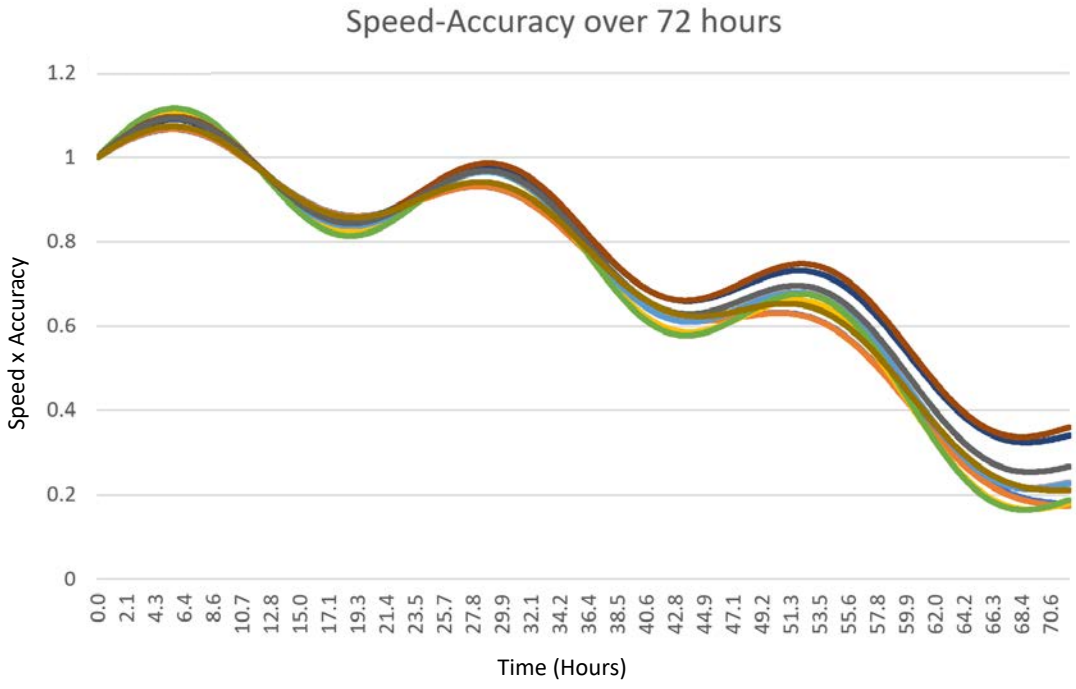


Figure 25. Ensemble speed-accuracy curves from HUNTER with stochastically final accuracy

```
3 references | 2/2 passing
public double GetMultiplier(double t)
{
    double speedAccuracy = GetSpeedAccuracy(t);
    double speedMultiplier = GetValue(t);
    double accuracy = speedAccuracy * speedMultiplier;
    return 1 / accuracy;
}
```

Figure 26. C# code to capture the relationship between speed, accuracy, and a fitness for duty PSF multiplier

Figure 27 depicts ensemble trends of dynamic Fitness for Duty over a duration of 72 hours. In the HRAEval event a fatigue-speed-accuracy model has been implemented to replace the polynomial fatigue index function with dynamic factorial model of fatigue index with stochastically generated parameters. Note that the peak multiplier around 5 hours is representative of the after-lunch fatigue phenomenon. In conjunction with the speed-accuracy component of the model, the dynamic Fitness for Duty PSF is calculated as an inverse of the accuracy estimate.

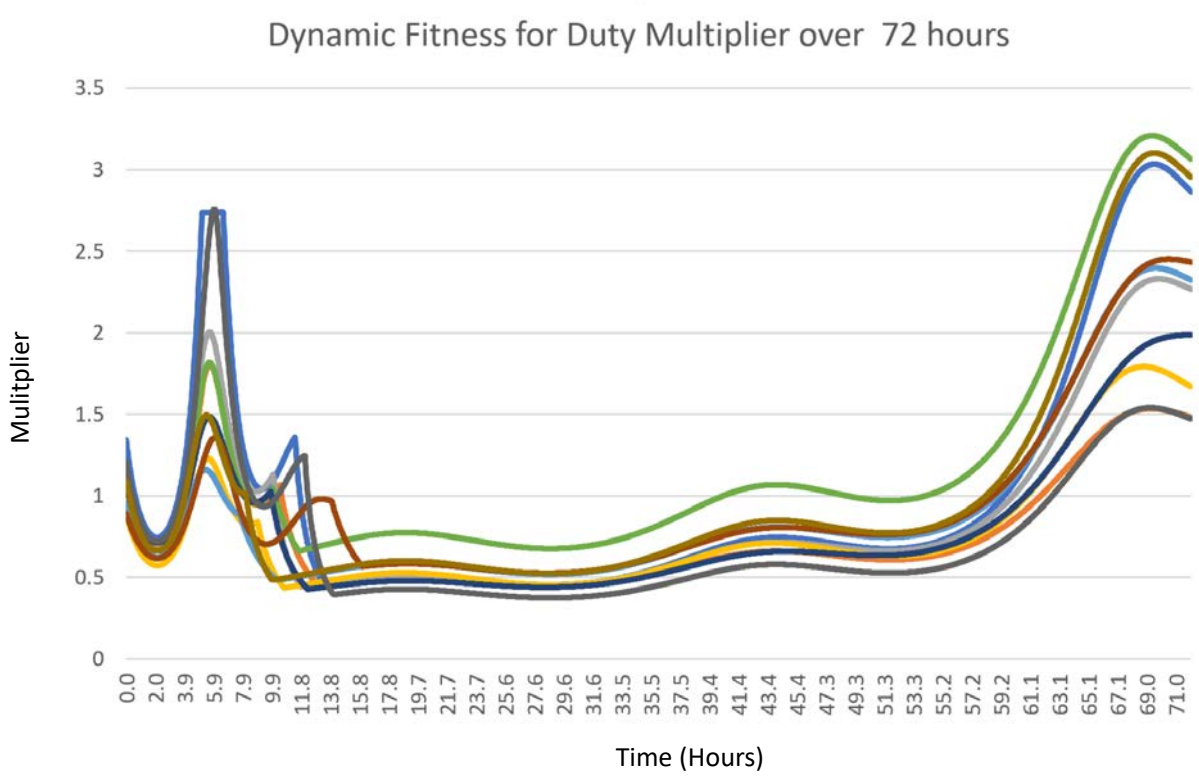


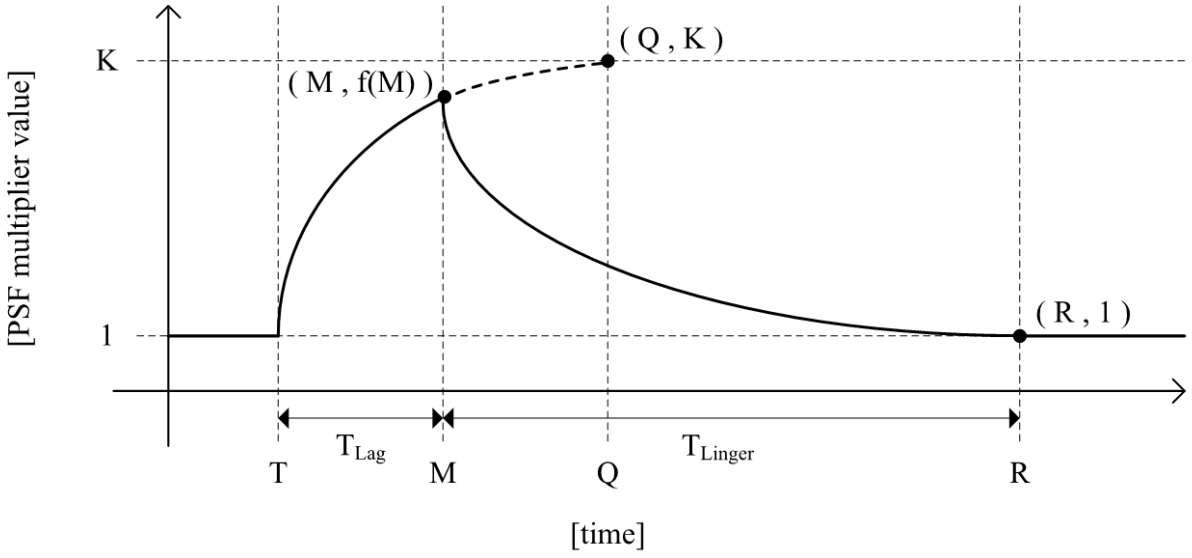
Figure 27. Ensemble plots for the dynamic Fitness for Duty multiplier over 72 hours

6.4 Dynamic PSF for Stress

6.4.1 Introduction

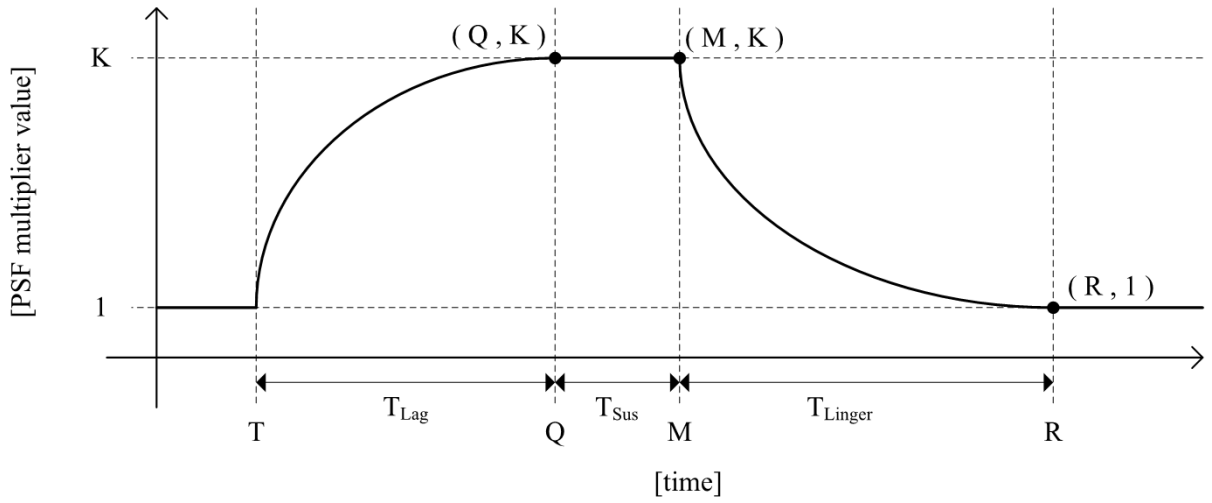
Stress refers to internal factors that cause mental tension and affect the ability of a person to focus and carry out activities. Stressors are a similar mental tension caused by external factors. Together, Stress and Stressors (commonly referred to simply as Stress) form a PSF that can have positive as well as negative effects. However, in SPAR-H, this PSF represents the level of undesired conditions that operators face while performing tasks, such as mental stress and excessive workload. SPAR-H classifies Stress and Stressors into three levels: Extreme, High, and Nominal. The Extreme level is when the person is subjected to disruptive stress, such as when it is sudden and sustained for a long time, and a multiplier of 5 is imposed. The High level defines a stress level higher than the nominal level due to factors like unexpected alarms, sustained noise, etc., and a multiplier of 2 is assigned. The Nominal level is a stress level conducive to good performance, and the multiplier is set to 1. In the absence of information, an error probability of 1 is charged for Stress and Stressors

The dynamic PSF for Stress and Stressors has been described in detail previously (Park, Boring, and Kim, 2019). First, stress increases dramatically until it reaches a maximum level (Dorin er al., 2012) and returns exponentially to a normal state after a certain period (Vitousek et al., 2018). Boring et al. (2022) provide mathematical models of the stress PSF that consider both lag of stress kicking in and the lingering impact of stress over time. These models are shown in Figure 28 when the task execution time is under 60 minutes, and in Figure 29when the task execution time is after 60 minutes.



$$\begin{cases} y = 1 & [x < T] \\ y = \dfrac{K-1}{\ln(Q-T+1)}\ln(x-T+1)+1 & [T \le X < Q] \\ y = \exp\left(-\dfrac{\ln(K)}{R-M}(x-R)\right) & [M \le x < R] \\ y = 1 & [R \le x] \end{cases}$$

Figure 28. A mathematical model of the stress PSF when the time to perform a task is less than 60 minutes (from Boring et al. 2022)



$$\begin{cases} y = 1 & [x < T] \\ y = \dfrac{K-1}{\ln(Q-T+1)}\ln(x-T+1)+1 & [T \le X < Q] \\ y = K & [Q \le x < M] \\ y = \exp\left(-\dfrac{\ln(K)}{R-M}(x-R)\right) & [M \le x < R] \\ y = 1 & [R \le x] \end{cases}$$

Figure 29. A mathematical model of the stress PSF when the time to perform a task is greater than 60 minutes (Boring et al. 2022)

For both Figure 28 and Figure 29, T is the starting time of a task, Q is time to reach a maximum HEP value in a task, and R is time to return to nominal HEP level. M is the time to finish a task, which is the sum of the time required and the time to start of a task. K is a multiplier value from HUNTER, which is generally assigned as 1 (Nominal), 2 (High), or 5 (Extreme) in SPAR-H. *f*(*M*) is a PSF level limited by the lag effect when the time to perform a task is less than 60 minutes.

Stress also impacts cognitive decision making. Stressors can come from a variety of undesirable conditions and circumstances and imped operators from optimally performing a task. Here we build on the lag and linger model of stress (Boring 2015; Park, Boring, and Kim 2019). The lag and linger model of stress is based on physiologically observed cortisol levels in humans in response to stress stimuli. After a stimulus is applied, there is a lag in the time it takes for the stress to reach peak levels. This lag is typically around 1 hour. Similarly, after a stressor has been removed the effects of stress and cortisol levels linger until they return to normal after approximately 3 hours. When a stress stimulus remains after 1 hour, the stress level is sustained with previous lag and linger model. The trigger to remove stress is when the task has been completed or the required time required has expired. But, what if the task is never completed? What if the task cannot be completed or the time required to complete the time is several days or weeks? Physiologically humans adapt over to stressful stimuli and will eventually become exhausted.

Here we have refined the lag and linger model to include adaptation. We refer to this model as Lag-Adapt-Linger (see Figure 30). The revised model can dynamically simulate the presentation and removal of stress events and is presented in Figure 31. Figure 32 shows the ensemble model put into practice with stress applied and removed across ten samples.

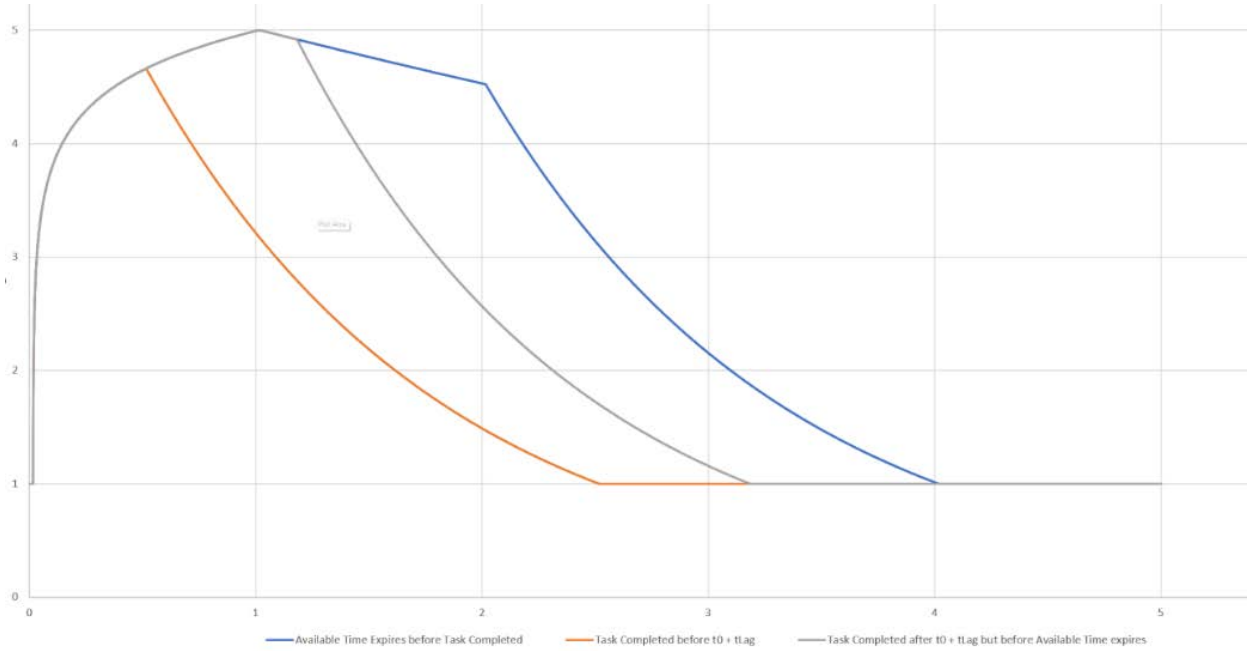


Figure 30. Idealized lag, adapt, and linger curves from when the task is completed before the peak level has been reached (orange), before the lag period (gray), and before available time expires (blue)

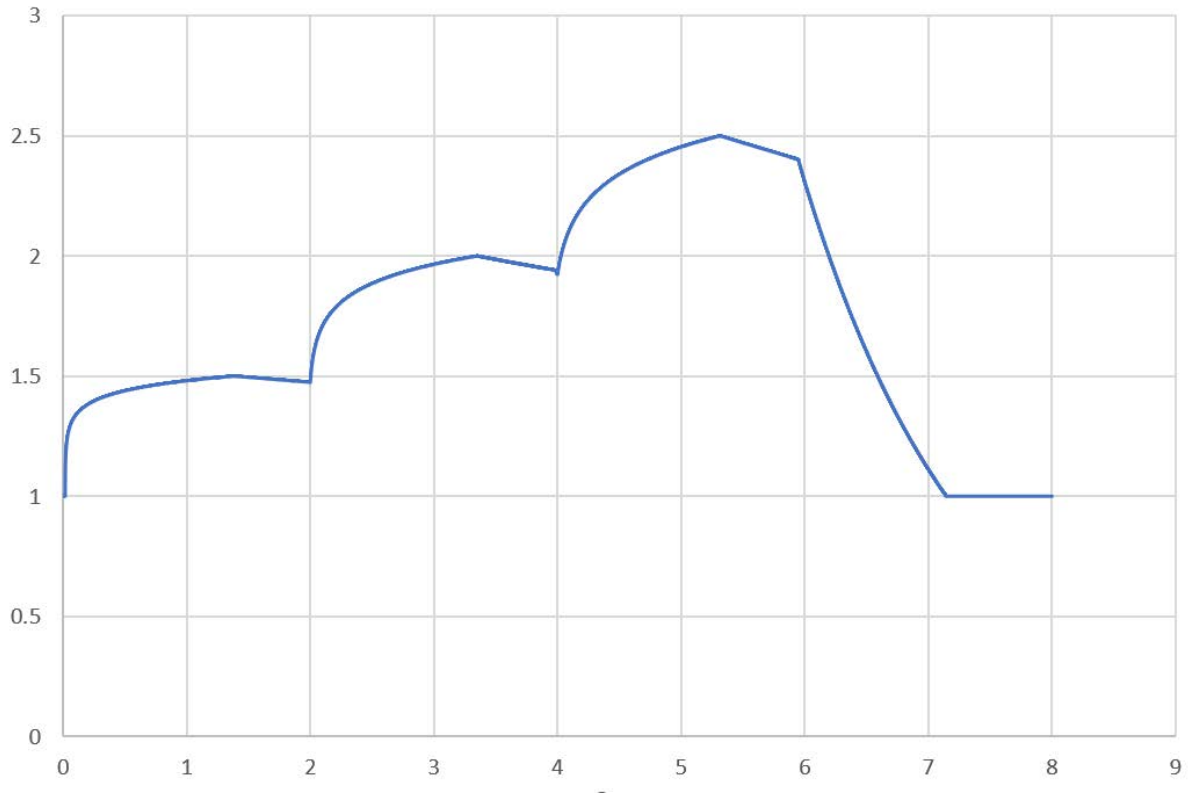


Figure 31. Idealized combined effect lag-adapt-linger curves from when the task is completed before the peak level has been reached, before the lag period, and before available time expires

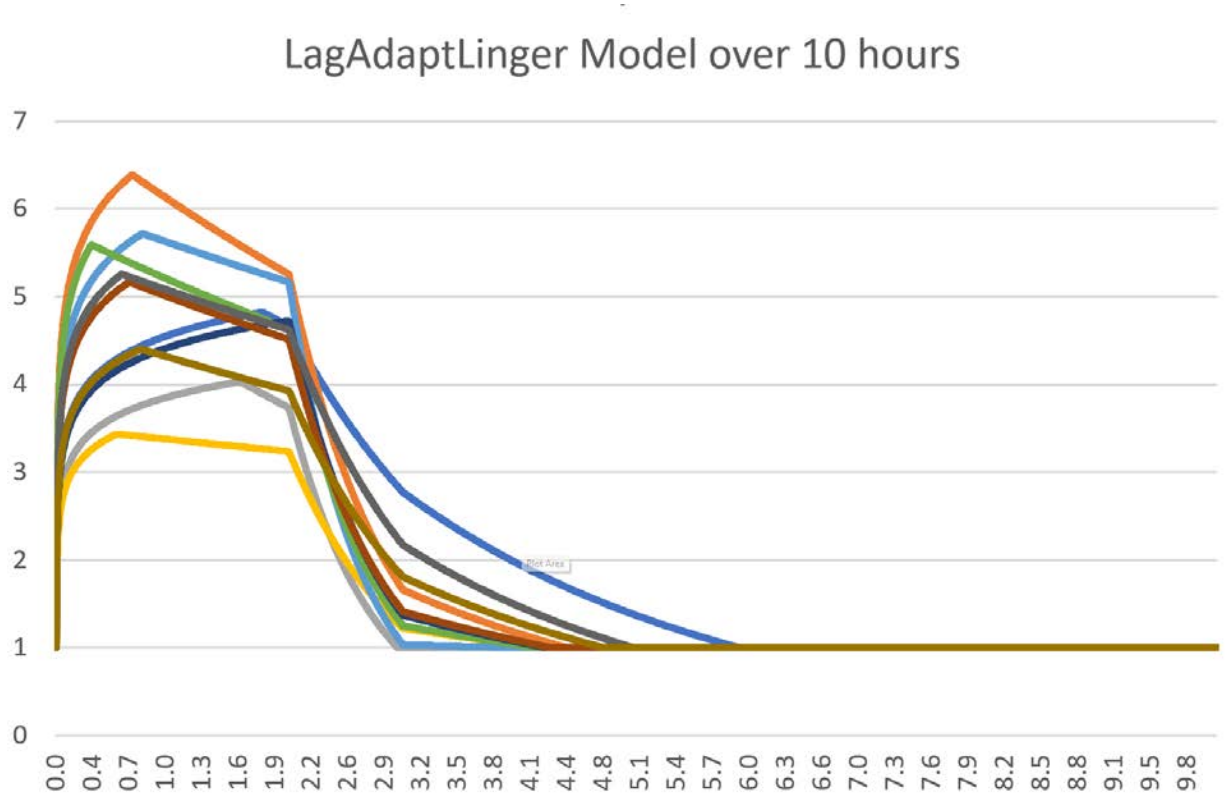


Figure 32. Ensemble of ten lag-adapt-linger models with stressor introduced at 1 minute and removed at 3 hours

6.4.2 Context Parameters

Three key time parameters can be specified as context links for HRAEval events. These specify ShiftTime (the time the operator has been on shift), AvailableTime (the time the operator has to complete the EMRALD Run), and TimeRequired (the time required for an operator to complete the tasking). Each of these parameters can be specified in seconds, minutes, or hours. For example, ShiftTimeH will specify time on shift in hours, and AvailableTimeM will specify the time available to complete the run in minutes. These variables should specify EMRALD SimVariables defined as doubles. (Note: If TimeRequiredM = 30 and TimeRequiredH = 1.5 are both specified, HUNTER will sum these and internally assign TimeRequired to 2 hours).

In addition to these time parameters, static PSF levels can be set as context parameters. The context name should be the PSF (e.g., Stress). For convenience, the PSF levels pre-load into EMRALD's global SimVariables. For example, Stress can be specified as Extreme as follows:

```
{
    "contextName": "Stress",
    "simVar": "Psf.Stress.Extreme"
}
```

6.5 Dynamic PSF for Experience and Training

6.5.1 Existing Treatment of Experience and Training as a PSF

Experience and training are among the key factors used to prevent or mitigate human error. In all countries that operate nuclear power plants, a formal educational system is provided to ensure reliable operation of nuclear reactors, and systematic training is conducted. Full scope simulator training (Swaton et al., 1987), which includes the reactor core and coolant systems, helps operators not only comprehend the nuclear power plant system, but also practice recognizing problems in case of an accident, making decisions, and taking appropriate actions. Through experience and training, it is possible to identify potential human errors and enhance the performance of operators.

One challenge in conducting HRA data collection is the extensive amount of training found in nuclear power plants. Reactor operators and other plant personnel are highly specialized and in demand. Yet, the high skill makes it difficult to perform research on participants with lower experience and training. Due to the constraints of cost, time, and its complexity, performing full scope simulator studies to collect HRA data can be challenging. However, limited functions or data collection on human error can be practiced with simplified simulators such as the Rancor Microworld Simulator (Rancor) and Compact Nuclear Simulator (CNS) (Park et al., 2023; Park et al., 2021), using either student or professional reactor operators. One such study relevant to experience and training will be detailed later in this section.

In existing HRA methods, the level of the Experience and Training PSF is determined based on expert judgment, and a multiplier is applied to the nominal HEP. The higher the PSF multiplier, the greater the HEP. The Experience and Training PSF, like all PSFs in SPAR-H (Gertman et al., 2005), distinguishes between Diagnosis and Action and is divided into three levels. For Diagnosis, the multiplier is 10 for the Low level (signifying a 10x increase in error when experience and training are low), 0.5 for the High level (signifying a ½ decrease to credit experience and training), and 1 for nominal or insufficient information cases (signifying no change over the nominal error rate). For Action, the multiplier is 3 for the Low level, 0.5 for the High level, and 1 for nominal or insufficient information cases. The original HRA method, Technique for Human Error Rate Prediction (THERP; Swain et al., 1983), categorizes the experience level as skilled and novice with the multiplier ranging from 1 to 2. Novice includes operators who have less than 6 months of experience with a reactor operator (RO) license, auxiliary operators (AO), maintainers, and technicians. As it is generally accepted that full performance capability requires about 6 months of experience, the training year is also taken into account (Swain et al., 1983). The Accident Sequence Evaluation Program (ASEP) method (Swain et al., 1987), a simplified version of THERP, divides into the cases where training is not considered and cases where well-known and practiced events are handled. In these cases, the multipliers of 10 for upper bound and 0.1 for lower bound are applied. Otherwise, the nominal HEP is applied. Cognitive Reliability and Error Analysis Method (CREAM; Hollnagel et al., 1998) evaluates the adequacy of training and preparation, taking into account the readiness of the work or familiarization. The PSF has three levels, with the multipliers ranging from 0.8 to 2.

Experience and training PSFs define several parameters, including the time elapsed since training or the period of requalification training, the quality of training, and existence of training. If there has been a lack of requalification training or a significant amount of time has elapsed, the PSF level is evaluated as Low. On the other hand, if the crew has just completed 10 days of refresher training, the PSF level is assumed to be Nominal or High. When considering the quality of training, inadequate training such as reluctance to use water to extinguish a fire or relying on incorrect guidance is assumed to correspond to a Low level of the PSF. Additionally, a general lack of training is evaluated as a Low level, while simulator training is regarded as a High level of the PSF (Gertman et al., 2005).

6.5.2 Experience and Training PSF Based on Objective Parameters

A brief overview of the relevant cognitive mechanisms known to govern experience and training provides the rationale for defining the experience and training PSF in HUNTER. The human information processing model people use to make decisions is depicted in Figure 33. This model comprises short term memory, working memory, long term memory, recall, and response (Campbell et al., 2002). Short term memory retains information over a short period of time. As shown in Figure 34 (left), recall performance for short term memory diminishes over time. Long term memory stores repetitive or long-standing memories, such as semantic memory or episodic memory. However, long term memory also has its limitations, and its performance can be improved through overlearning and increasing the recall number, as shown in Figure 34 (right). Working memory is a process of converting information from short term and long term memories into cognitive and physical actions such as decision, and includes some processing mechanisms such as chunking to enhance the memories (Cowan et al., 2008).

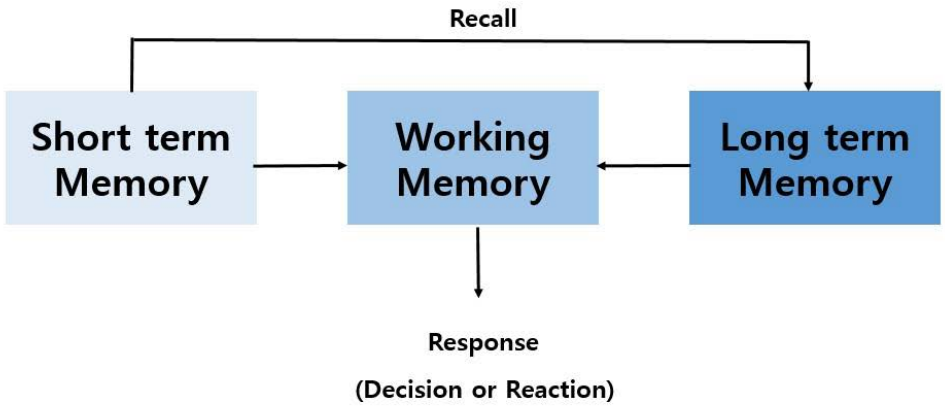


Figure 33. Human information processing model (from Campbell et al., 2002)

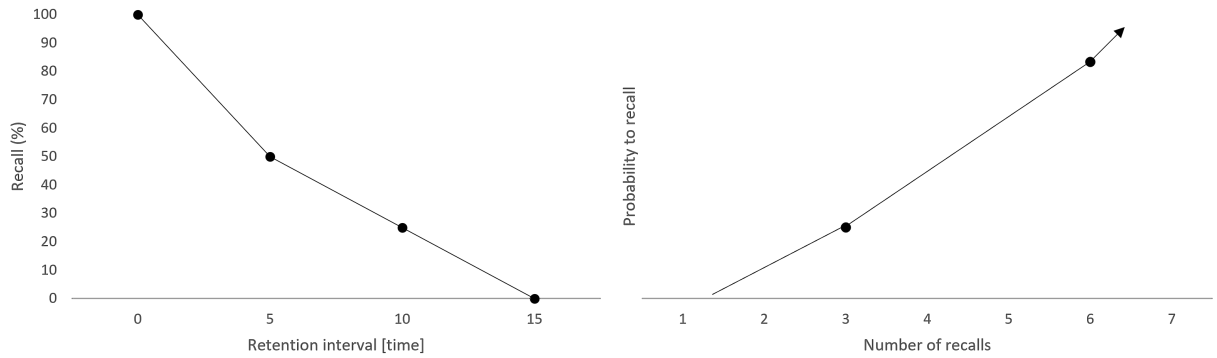


Figure 34. Recall probability depending on retention interval for short term memory (left) and recall ability depending on the number of recalls (right) (from Campbell et al., 2002)

Short and long term memories can both contribute to deteriorated human performance (Swain et al., 1983), but training can help memory capacity. In a dynamic flight emergency, a pilot's intuition is critical when making decisions. This is an unconscious process from memories stored through experience. To strengthen this ability, it is recommended to enhance training, expand experience, and repeat it (Manurung et al., 2022). Swaton et al. (1987) also suggests that operators can enhance their performance by continuing training with retraining to update and expand their knowledge and skills. In cardiopulmonary resuscitation (CPR) experiments (Curry et al., 1987), the training effect showed improved performance, but this improvement cannot be sustained for more than 6 months. Thus, the importance of regular training programs is suggested to maintain good performance.

However, there is no specific analysis of the effect over specific parameters such as time in existing HRA methods, except for the classification of skilled and novice operators based on six months of experience in THERP (Swain et al., 1983). In SPAR-H, one of the factors for evaluating the Experience and Training PSF is time elapsed since training or periodic requalification training (German et al., 2005). As time elapses, the effect of experience and training on positive performance decreases due to the deterioration of memory over time, which can be represented by a forgetting curve. On the other hand, as the amount of experience and training increases, the effect of the experience and training performance may increase, indicating the capacity for long term memory.

The forgetting curve depicts the relationship between memory retention and time elapsed. Ebbinghaus (Wixted et al., 1991, Murre et al., 2015) was the first to propose the forgetting curve, which is fitted with a power function or a logarithmic function, indicating a rapid decrease in memory retention over time. Although Murre et al. (2015) successfully replicated Ebbinghaus' experiment, there is no universally agreed upon form of the forgetting curve among researchers (White et al., 2001; Jaber et al., 2004). Nevertheless, it is generally accepted that the curve illustrates a decline in memory performance over time, with performance improving as additional review is implemented.

The Experience and Training PSF can be dependent on the time elapsed since training and the amount of training, as illustrated by the forgetting curve. It is possible to predict human performance related to the PSF based on the trend depicted in Figure 35 (cf. Kim et al., 2021).

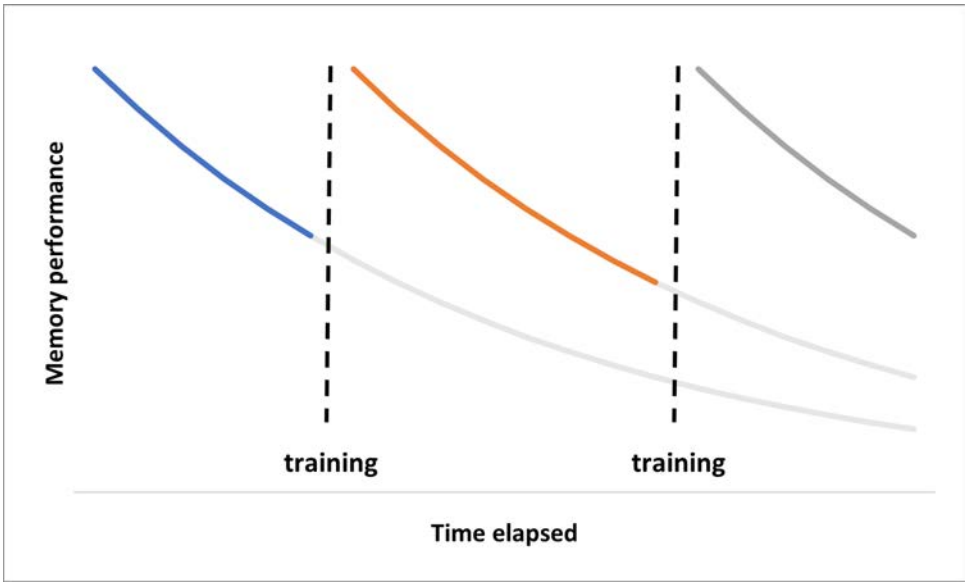


Figure 35. Predicted memory performance depending on time elapsed

6.5.3 Experience and Training Effects Evaluated from a Simplified Simulator Study

The data from a recent study (Kim et al., 2023) evaluating the use of simplified simulators to collect human performance data to inform HRA methods was further analyzed to examine experience and training effects. The study was structured such that the student participants completed four sessions of simulator scenarios, each separated by approximately two weeks. These data provided the opportunity to evaluate experience and training effects longitudinally.

6.5.3.1 Methods for Study

Rancor is a simplified nuclear power plant simulator developed by INL and University of Idaho (Ulrich et al, 2017). Using Rancor, an experimental participant can identify the status of components and systems in both normal and emergency situations and practice simulated operations with simplified procedures. In this study, the participants were students who lacked extensive experience and knowledge about operating nuclear power plants. The study design is specifically structured to observe the outcomes of training and experience over time with participants who had little prior knowledge or experience in nuclear power plant operations.

The study was conducted with 16 students majoring in nuclear energy at Chosun University in South Korea (Kim et al., 2023). They performed 10 simplified scenarios including start up, shut down, manual rod control during startup, manual feedwater flow control during startup, failure of a reactor coolant pump under full power operation, failure of a control rod under full power operation, failure of a feedwater pump under full power operation, turbine failure under full power operation, and steam generator tube rupture. Four trial sessions of experiments were scheduled, with each trial featuring four scenarios. Different, randomly selected scenarios were performed in each trial, although there are some overlapping scenarios across all the trials. While analysis within the same scenario across multiple trials would have been ideal, the simplified scenarios and procedures ensure a reasonable approximation. There was an average interval of 14 days between rounds.

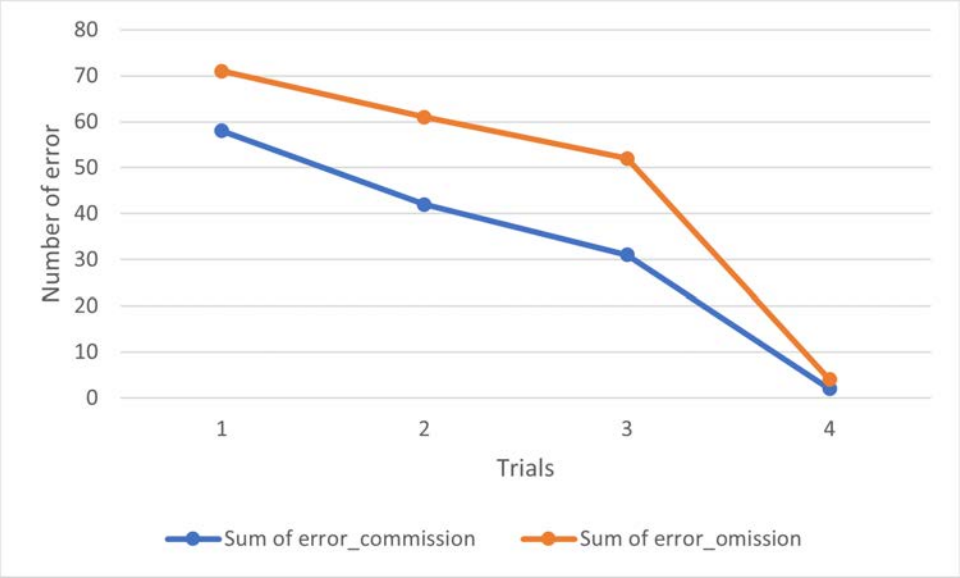


Figure 36. Number of errors across trials

6.5.3.2 *Results of Study*

The first analysis examines errors of commission—in which participants fail to take appropriate actions—and errors of omission—in which they omit the procedures while following the designated procedures. The result, as depicted in Figure 36, indicates decreases in the number of commission and omission errors as the number of trials increases. The reduction in errors can be attributed to the training effect, which enhances the participants' experience and training performance over successive trials.

The second analysis aimed to determine the impact of training on human performance by measuring the average time to complete a task and the error rate. Figure 37 shows the distributions of these variables based on the number of trials. As the number of trials increases, the means of these distributions decrease, indicating an improvement in human performance with increased training.

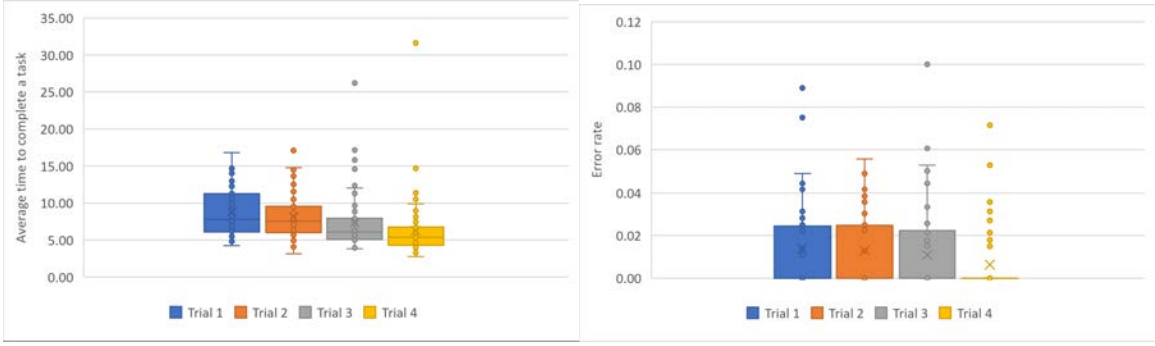


Figure 37. Distribution of average time to complete a task (left) and error rate (right) depending on trials

Furthermore, the results of the experiment also indicate that within the same experiment, the average time to complete a task and the error rate both decrease as the number of trials increase, as shown in Figure 38. However, when another experiment is conducted again after a period of time, there is a decrease in performance, which is then followed by an improvement as the number of trials increases. This suggests that the effectiveness of training is not permanent and may decay over time, but can be regained through additional training. This finding mirrors the crests and troughs of the forgetting curves over time as shown in Figure 35.

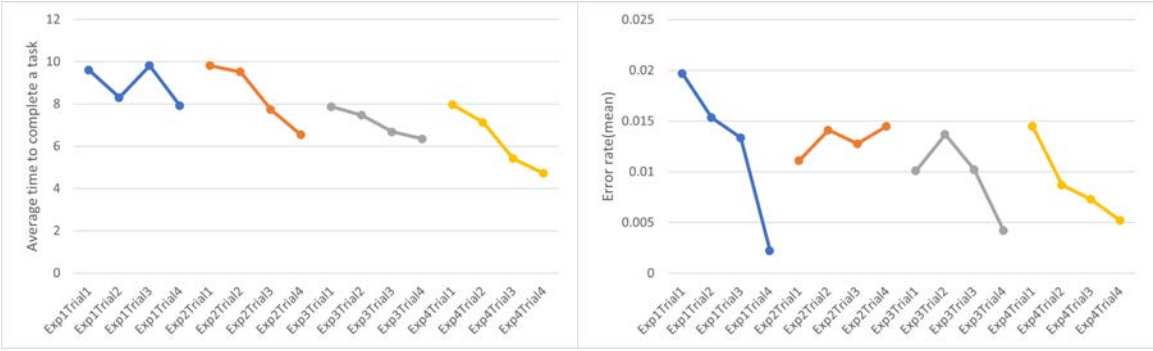


Figure 38. Means of average time to complete a task (left) and error rate (right) depending on experimental rounds and trials

6.5.4 Proposed General Form of Experience and Training PSF

While there is no generalized form of experience and training that can be derived absent training context, it is possible to calibrate high levels of training (indicated by repeated training trials) to the multiplier levels in SPAR-H for the Experience and Training PSF. Additionally, it is possible to model a decay curve as a function of elapsed time since last training. This forgetting curve is reset with refresher training. However, over time with additional training and experience, the forgetting curve is not as strong, indicating a shallower slope. Additional empirical data points are necessary to calibrate the function, but it generally takes the form:

$$X_t = X_0 \, e^{-L_1 N + L_2 T} \tag{2}$$

where:

- X_t is the multiplier for training and experience in SPAR-H at given time t,
- X_0 is the initial experience and training,
- e is the base of the natural logarithm,
- L_1 is the decay constant for the number of trainings, a positive value related to the decay over time,
- L_2 is the growth constant for the time elapsed since training, a positive value related to the growth over time,
- N is the number of trainings, and
- T is the total time elapsed.

This form of the equation only accounts for nominal or negative influences of Experience and Training as denoted by PSF multipliers ≥ 1. Because SPAR-H Action tasks would have a range of 1-3, the maximum value is 3 and minimum value is 1 in Equation 2. Equation 2 decreases exponentially with respect to the number of trainings and increases exponentially with respect to the time elapsed since training. Currently, the constants, L_1 and L_2, are unknown, but the constant, L_1, for the number of trainings would be a small number for highly skilled individuals, indicating a slow decay, and a large number for less skilled individuals, indicating a fast decay. When refresher training is administered, it restarts the function, with Experience and Training at a high level, as denoted by a low PSF multiplier. The constant, L_2, for the time elapsed since training is related to the effect of growth, so it would be less than the constant for the number of trainings.

Note that X is a multiplier that is inversely related to experience and training. The lower the experience and training, the higher the actual Experience and Training PSF is, and the higher the error probability will be. Conversely, high levels of experience and training would result in a low multiplier. Thus, the decay function of forgetting results in an increase in X, effectively making it a growth function for the HEP.

The study discusses the effects of training on the Experience and Training PSF multiplier with the assumptions made in Equation 2. The values of L_1 and L_2 are assumed to be 0.01 and 0.001, respectively, and the initial multiplier is assumed to be 5. The modeling assumes that the multiplier has a range of 1 to 10, although in actual SPAR-H, it has a range of 1-3. The results show that the multiplier decreases with the number of trainings, as shown in Figure 39, and increases with time elapsed since training, as shown in Figure 40. The study assumes a larger decay constant than the growth constant, resulting in a faster rate of decrease.

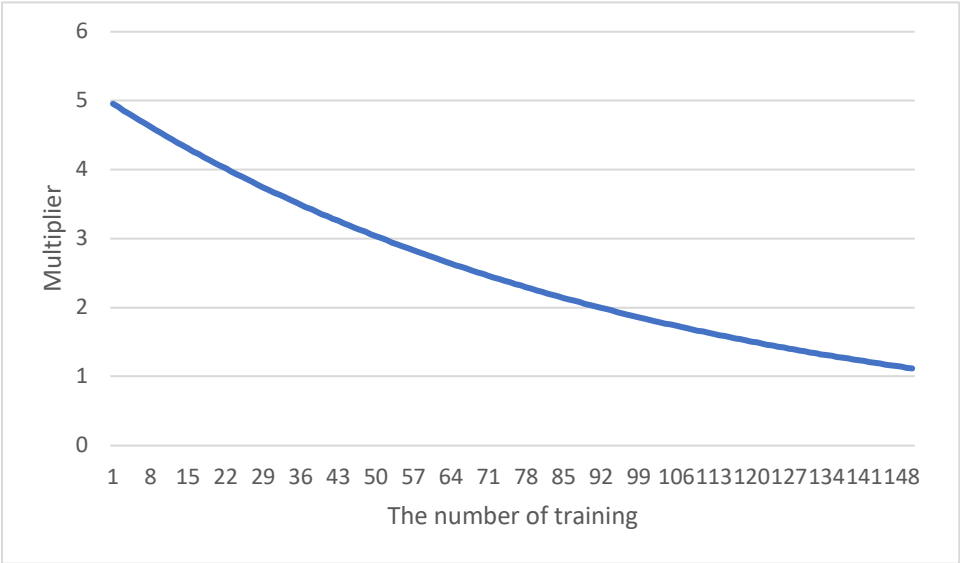


Figure 39. Experience and Training PSF multiplier decreasing depending on the number of trainings (L_1 = 0.01, L_2 = 0.001)

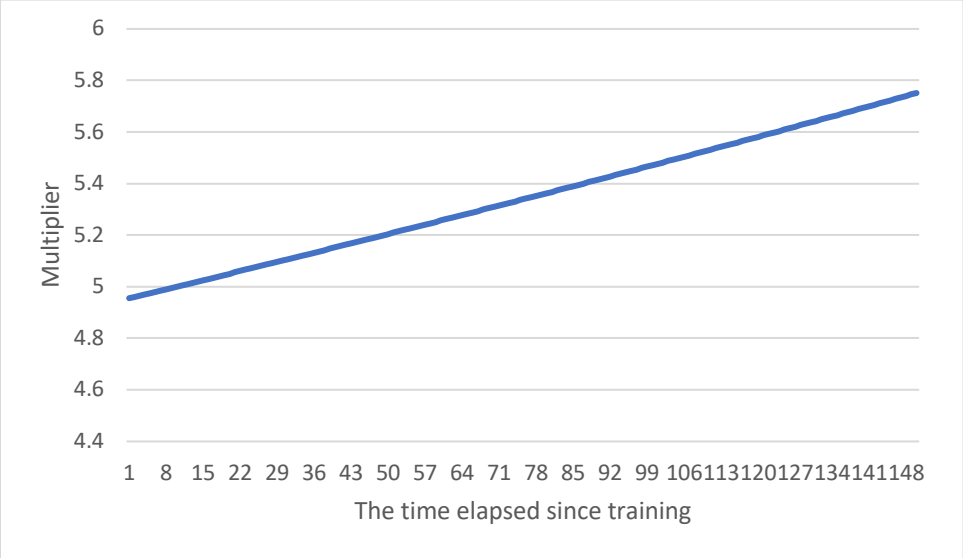


Figure 40. Experience and Training PSF multiplier increasing depending on the time elapsed since trainings (L_1 = 0.01, L_2 = 0.001)

Additionally, the study shows the effect of training cycle on the multiplier, as shown in Figure 41 and Figure 42, which demonstrate the changes in the Experience and Training PSF based on both of the number of trainings and the training cycle. Figure 41 shows the case of training for 10 days in an interval of 40 days, and Figure 42 shows the case of training for 5 days in an interval of 3 months. However, it should be noted that the constant values used in the study are assumptions, and training for 5 days in an actual 3-month cycle would not necessarily degrade performance.

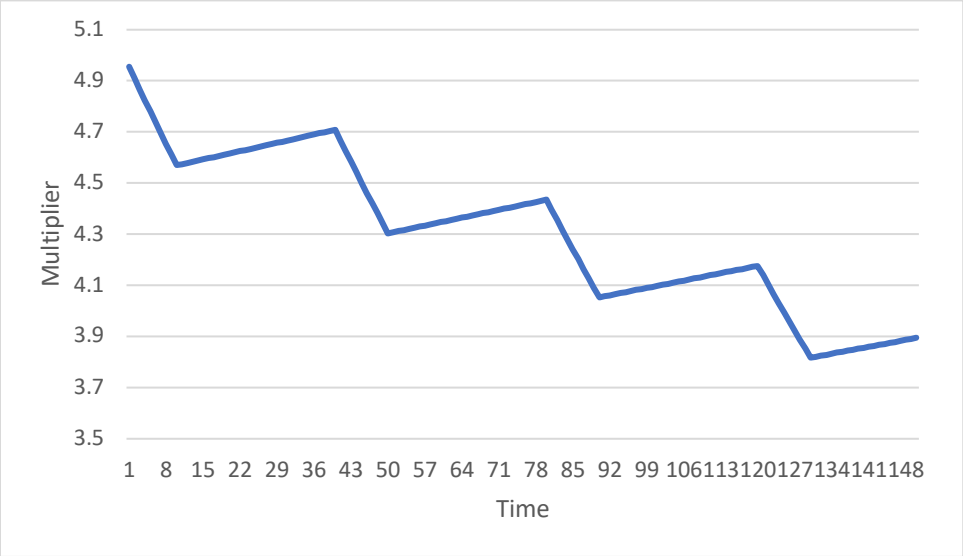


Figure 41. Experience and Training PSF multiplier for 10 days of training in a 40-day cycle ($L_1 = 0.01$, $L_2 = 0.001$)

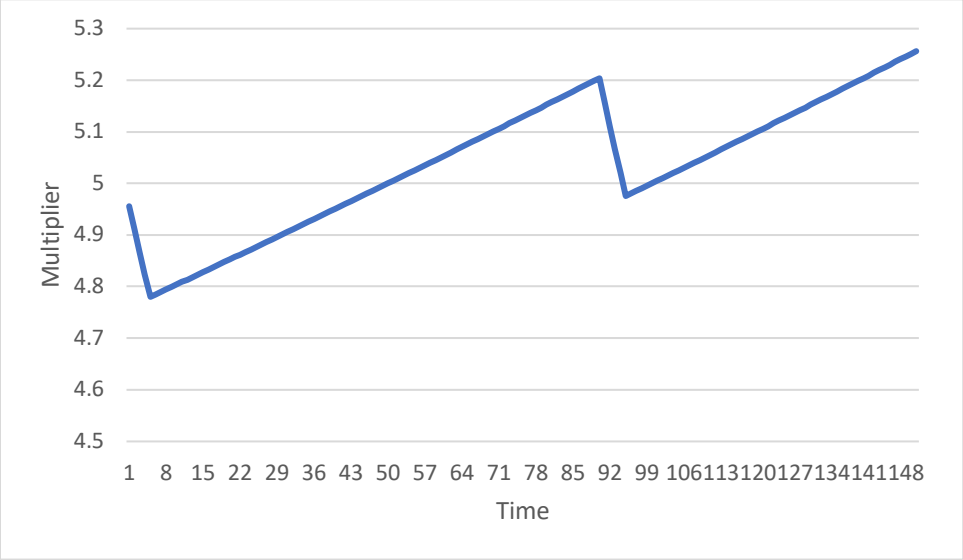


Figure 42. Experience and Training PSF multiplier for 5 days of training in a 3-months cycle ($L_1 = 0.01$, $L_2 = 0.001$)

7. SAMPLE ANALYSES

7.1 Introduction

To test and demonstrate the EMRALD-HUNTER coupling, two scenarios were modeled and run. The outputs of the EMRALD-HUNTER models were benchmarked against outputs from the standalone version of HUNTER for steam generator tube rupture (SGTR; Boring et al., 2022) and loss of feedwater (LOFW; Lew et al., 2022) scenarios.

7.2 Steam Generator Tube Rupture (SGTR)

7.2.1 SGTR Description

In a pressurized water reactor (PWR), the steam generator plays a crucial role in the transfer of heat to produce steam for the secondary side, which in turn drives the generator to generate electricity. Moreover, after the reactor is shut down, heat transfer serves to remove decay heat. The steam generator is also essential in isolating the primary side, which may contain radioactive material, from the secondary side. If one or more tube ruptures occur in the steam generator, the flow can leak from the primary side to the secondary side due to the higher pressure on the primary side. As a result, the core level in the primary side decreases, and a charging pump and safety injection system are required to recover the coolant. It is also important to isolate the defective steam generator to prevent any radioactive material from leaking into the secondary side, which may release into the environment.

In the past, several SGTR accidents have occurred, but they have not caused significant doses to the public as reported in NUREG/CR-6365 (MacDonald et al., 1996). Some historic SGTRs involving operator performance include:

- An SGTR accident occurred in 1991 in steam generator A of Mihanma Unit 2 due to high cycle fatigue. When the air ejector high radiation alarm and the secondary steam blowdown radiation monitor alarmed, the operator promptly started the charging pump and reduced the reactor power to shut it down. The reactor automatically tripped, and the turbine also tripped. The safety injection pump was automatically started due to the low level of the pressurizer and low pressure of the reactor coolant system (RCS). The operator identified the defective steam generator and isolated the main steam line isolation. However, a manual closing action was required as the valve did not close properly. Subsequently, the operator opened the steam relief valve to cool the RCS in the intact steam generator and used the pressurizer auxiliary spray to depressurize the RCS. As the RCS pressure lowered, the leak flow was reduced, and the pressurizer level was restored. Then, the operator stopped the two safety injection pumps. In this accident, the radiation alarm was the first indication of SGTR, and the operator's prompt response prevented the escalation of the accident.
- In 1993, tube rupture occurred in steam generator 2 of a unit at Palo Verde Nuclear Generating Station due to outside diameter stress corrosion cracking from tube-to-tube crevice formation. The pressure and level in the pressurizer decreased, and the operator immediately started the charging pump and energized the pressurizer heater to restore the level and pressure. However, the level and pressure continued to decrease, leading to de-energization of the pressurizer heater. The operator manually tripped the reactor, and the turbine automatically tripped. Because of pressurizer low pressure, several safety systems such as safety injection actuation system (SIAS) and the containment isolation actuation system were actuated. All of charging and safety injection systems restored the pressurizer level and pressure. An SGTR was suspected, but it could not be diagnosed immediately. The entry condition of the SGTR procedure was not satisfied, so the operator entered the

functional recovery procedure (FRP). The procedure focused only on the current situation rather than previous trends, and alarms and indicator alarms for SGTR were also not presented, making it more confusing for the operator. After the indicator alarm for SGTR, which was isolated when SIAS was restored, the operator performed the SGTR procedure and successfully diagnosed defective steam generator and isolated it. Although the power plant was safely stabilized, there was a slow response due to the operator actions.

To mitigate an SGTR, operator actions are necessary to minimize leakage from the primary system to the secondary coolant system and to maintain primary coolant subcooling. The operator should identify the point of leakage within an appropriate time through radiation alarms, diagnose the SGTR, determine the defective steam generator, and isolate it. Additionally, the operator should take action for RCS cooling, such as dumping steam. In case of automatic system failure, such as the radiation alarm for helping to diagnose SGTR or the safety injection system for recovering coolant, the operator should restore or manually start the system. Furthermore, while safety injection is necessary to recover the coolant, an operator action to stop the safety injection system is also required later to depressurize for reducing the break flow. These accidents highlight the critical importance of considering human factors in nuclear power plant operations.

7.2.2 SGTR PRA Modeling

SGTR is classified as an event where the break flow leaked from the primary to the secondary coolant exceeds the normal charging flow capacity (U.S. Nuclear Regulatory Commission, 1988). An event tree for SGTR has been developed by INL and is presented in Figure 43 (Ma et al., 2019). The event tree consists of an initiating event and 11 event tree headings listed in Table 10, and 22 scenarios are analyzed based on these headings. The event tree is designed to show how these scenarios can lead to either a stable state or core damage. The event tree provides a useful tool for analyzing the potential scenarios of SGTR.

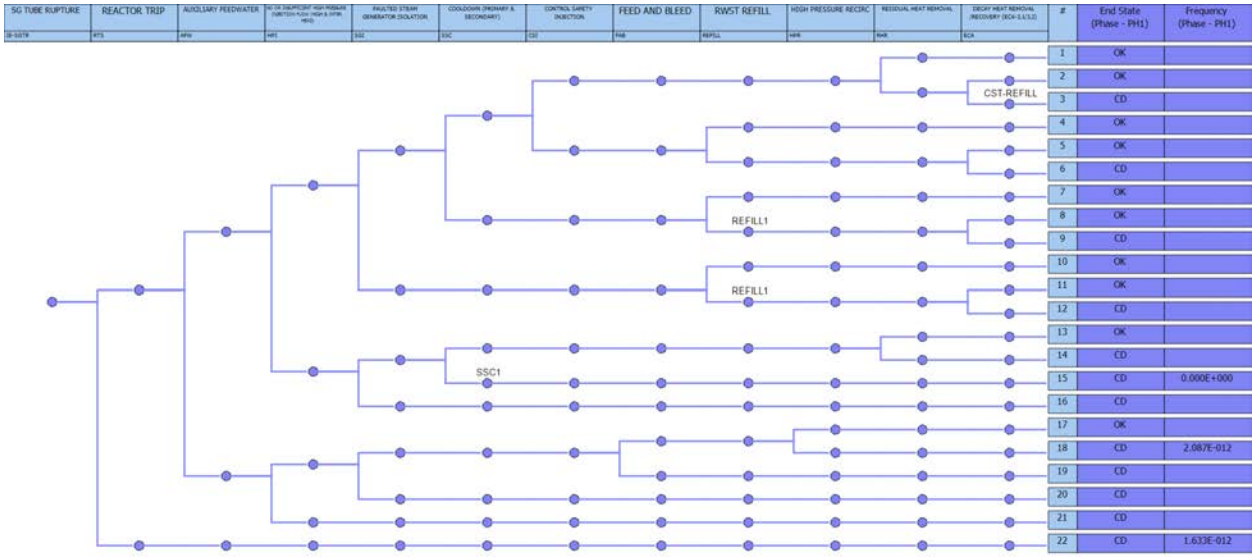


Figure 43. Generic SGTR event tree (from Ma et al., 2019)

Table 10. Event tree headings for SGTR (from Ma et al., 2019)

Heading	Description
IE-SGTR	Initiating event of steam generator tube rupture
RTS	Reactor trip
AFW	Supply of the auxiliary Feedwater system to steam generator
HPI	Injection of high-pressure safety injection system
SGI	Isolation of ruptured steam generator
SSC	Cooldown of primary & secondary sides
CSI	Termination or Control of high-pressure injection system
FAB	Feed and bleed operation
REFILL	Refill of refueling water storage tank (RWST)
HPR	High pressure recirculation operation
RHR	Removal of residual heat
ECA	Depressurization of the primary and secondary for decay heat removal/recovery

In the event tree for SGTR, the first scenario leads to a stable state where auxiliary feedwater can be successfully supplied to the steam generator, HPI is successful, ruptured steam generator is isolated, primary and secondary sides are cooled down, HPI is terminated, and residual heat is removed (Scenario 1). Even though the residual heat removal fails, it is still possible to lead to a stable state through successful depressurization of primary and secondary sides and alignment for RHR (i.e., ECA) (Scenario 2). However, if both RHR and ECA fail, it results in core damage (Scenario 3).

If HPI fails to terminate after cooldown, primary and secondary sides cannot be cooled down or the steam generator cannot be isolated, either RWST refill or ECA is required. If RWST fails to be refilled with ECA failure, it leads to core damage (scenarios 6, 9, and 12 respectively). However, if either RWST refill or ECA succeeds, it results in a stable state (scenarios 4 - 5, 7 - 8, 10 - 11).

When auxiliary feedwater is operational but HPI fails to operate, it is necessary to isolate the defective steam generator, cool down the primary and secondary sides, and remove the residual heat (Scenario 13). If any of these mitigations fails, it may result in core damage (Scenario 14 - 16). On the other hand, if auxiliary feedwater fails to operate, it is necessary to operate HPI, isolate the ruptured steam generator, and perform feed and bleed as well as high pressure recirculation to mitigate the accident (Scenario 17). If any of these mitigations fails, it may lead to core damage (Scenario 18 - 21). Finally, in the case where the reactor trip fails, it is analyzed as an anticipated transient without scram (ATWS) scenario (Scenario 18-22).

Table 11 provides several HFEs considered in SGTR (Ma et al., 2019). In addition to diagnosis of the SGTR, the operation actions to mitigate the accident include feed and bleed, RHR operation, and control or termination of safety injection flow. Additionally, the operator's response is also included when the automatic system such as the reactor protection system fails. By considering these HFEs, it is possible to assess the extent to which an operator's actions may have contributed to the accident or identify any

weakness in the system. This information can be used to improve the design of the system, as well as the training and procedures, with the aim of the reducing the risk of the accident.

Table 11. Generic human failure events in SGTR

Index	Description of Human Failure Events
1	Operators fail to diagnose SGTR and start procedures.
2	Operators fail to respond with reactor protection system signal present.
3	Operators fail to maintain pump suction.
4	Operators fail to control auxiliary feedwater turbine-driven pump after battery depletion; Non-Station Blackout.
5	Operators fail to initiate feed and bleed cooling.
6	Operators fail to start high pressure recirculation.
7	Operator fails to refill the refueling water storage tank.
8	Operators fail to control/terminate safety injection flow.
9	Operators fail to initiate residual heat removal.
10	Operators fail to recover offsite power in 1 hr.
11	Operators fail to align AC power given non-Loss of offsite power.
12	Operators fail to depressurize RCS/secondary side.
13	Operators fail to depressurize RCS/secondary side (Rapid).
14	Operators fail to implement SGTR procedure ECA 3.1 & 3.2.

7.2.3 EMRALD-HUNTER SGTR

This section describes an SGTR model developed in EMRALD with embedded HUNTER functionality. Figure 44 (repeated from Figure 13 for purposes of illustrating the current explanation) shows the EMRALD-HUNTER SGTR model. The model includes eleven states:

1) “Start”
2) “InitiatingEvent”
3) “DiagnoseSGTR”
4) “MitigateSGTR”
5) “HepGTOneFailure”
6) “HumanErrorFailure”
7) “OutofTimeFailure”

8) “OnRepeatFailure”

9) “MultipleFailure”

10) “EventResolved”, and

11) “Terminate.”

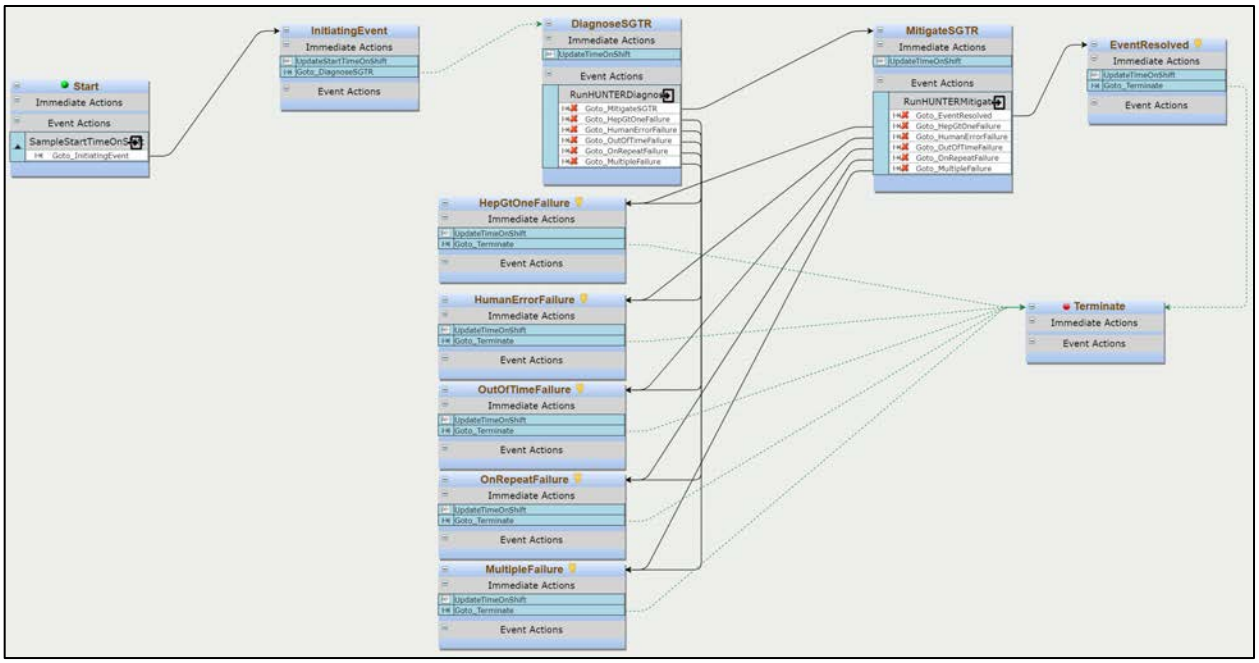


Figure 44. The EMRALD-HUNTER SGTR model

The “Start” state indicates the start of simulation in EMRALD-HUNTER. In the “Start” state, the event “SampleStartTimeOnShift” randomly samples the start time on shift from a time distribution added by users, then leads to the “InitiatingEvent” state. The start time on shift and overall time on shift modeled in states are used for estimating values in dynamic PSF calculationa for Fitness for Duty. The “InitiatingEvent” state declares the start of the SGTR, updates time on shift, then leads to the “DiagnoseSGTR” state via the two immediate actions modeled in the state. The “DiagnoseSGTR” and “MitigateSGTR” events are HRAEval events that use HUNTER. These states load the HUNTER functions for implementing SGTR procedures modeled in HUNTER (see Figure 45 and Figure 46), which include procedure steps and GOMS-HRA primitives, then simulate human actions relevant to diagnosing and mitigating the SGTR. These events calculate an elapsed time required to execute a procedure and have a list of actions. The “HepGTOneFailure”, “ HumanErrorFailure”, “OutofTimeFailure”, “OnRepeatFailure”, “MultipleFailure”, and “Success” actions represent different outcomes from the HUNTER simulation. Recall these outcomes from Section 5.2. A brief explanation on each state is repeated below:

- “HepGTOneFailure”—An HEP is equal to or greater than 1.0 as a result of HUNTER simulation and cannot be completed
- “HumanErrorFailure”—An HEP less than 1.0 was calculated, but the tasked failed due to chance
- “OutofTimeFailure”—Human actions are not completed within the time window

- “OnRepeatFailure”—An HEP less than 1.0 was calculated and the task was repeated up to the MaxRepeat count but failed due to chance.
- “MultipleFailure”—More than one failure type occurs.
- “Success”—The success of diagnosis and mitigation of SGTR.

In the EMRALD model each of these actions point to a key state followed by “Terminate” which ends the simulation run.

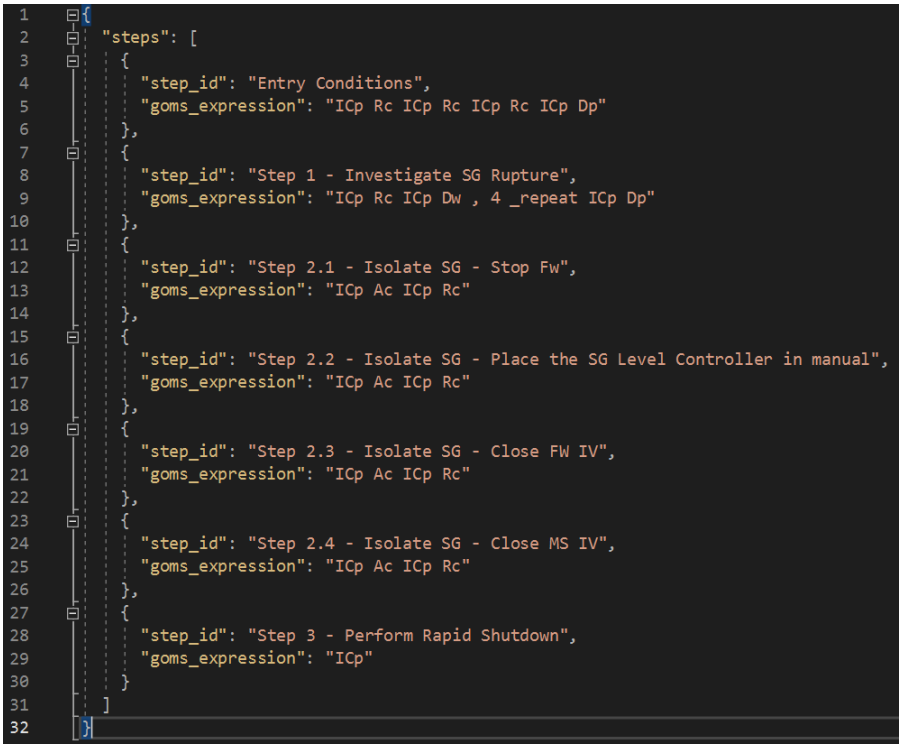
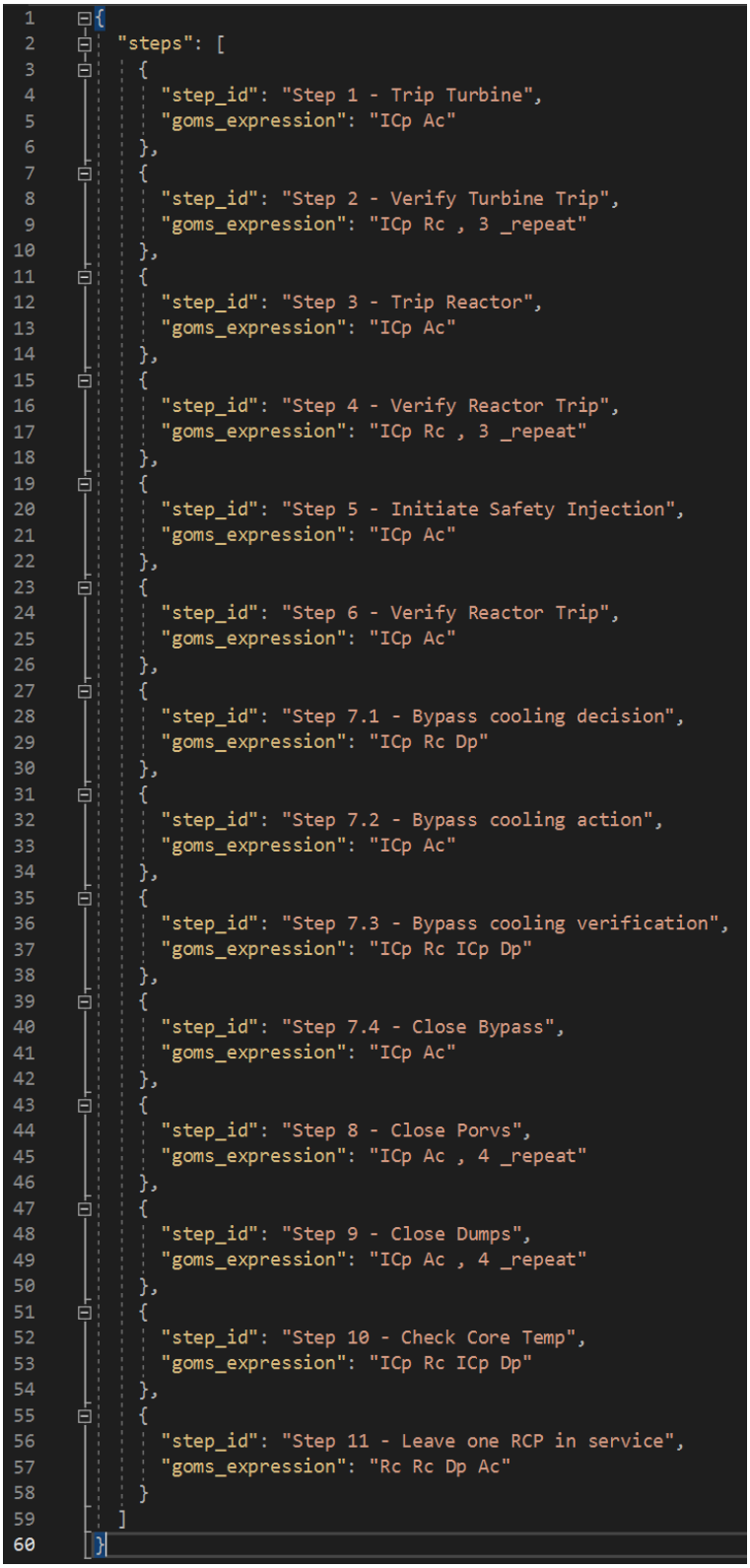


Figure 45. Procedure contents coded for diagnosing SGTR within EMRALD-HUNTER



```
{
  "steps": [
    {
      "step_id": "Step 1 - Trip Turbine",
      "goms_expression": "ICp Ac"
    },
    {
      "step_id": "Step 2 - Verify Turbine Trip",
      "goms_expression": "ICp Rc , 3 _repeat"
    },
    {
      "step_id": "Step 3 - Trip Reactor",
      "goms_expression": "ICp Ac"
    },
    {
      "step_id": "Step 4 - Verify Reactor Trip",
      "goms_expression": "ICp Rc , 3 _repeat"
    },
    {
      "step_id": "Step 5 - Initiate Safety Injection",
      "goms_expression": "ICp Ac"
    },
    {
      "step_id": "Step 6 - Verify Reactor Trip",
      "goms_expression": "ICp Ac"
    },
    {
      "step_id": "Step 7.1 - Bypass cooling decision",
      "goms_expression": "ICp Rc Dp"
    },
    {
      "step_id": "Step 7.2 - Bypass cooling action",
      "goms_expression": "ICp Ac"
    },
    {
      "step_id": "Step 7.3 - Bypass cooling verification",
      "goms_expression": "ICp Rc ICp Dp"
    },
    {
      "step_id": "Step 7.4 - Close Bypass",
      "goms_expression": "ICp Ac"
    },
    {
      "step_id": "Step 8 - Close Porvs",
      "goms_expression": "ICp Ac , 4 _repeat"
    },
    {
      "step_id": "Step 9 - Close Dumps",
      "goms_expression": "ICp Ac , 4 _repeat"
    },
    {
      "step_id": "Step 10 - Check Core Temp",
      "goms_expression": "ICp Rc ICp Dp"
    },
    {
      "step_id": "Step 11 - Leave one RCP in service",
      "goms_expression": "Rc Rc Dp Ac"
    }
  ]
}
```

Figure 46. The procedure contents coded for mitigating SGTR within EMRALD-HUNTER

Figure 47 shows an example of simulation result of EMRALD-HUNTER SGTR model. All the runs discussed had a sampled starting time on shift with a mean of 4 hours and a standard deviation of 2 hours. With 1,000 trials, 962 cases (i.e., “EventResolved”) were successfully mitigated from SGTR, while 12 cases resulted in failed scenarios (i.e., “OnRepeatFailure”) and 26 cases of overtime-based failure scenarios (i.e., “OutOfTimeFailure”) were observed. The failed scenarios refer to the failure cases caused by failure of GOMS-HRA primitive, while the overtime-based failure scenarios mean the failure cases that human actions are not finished within the time window.

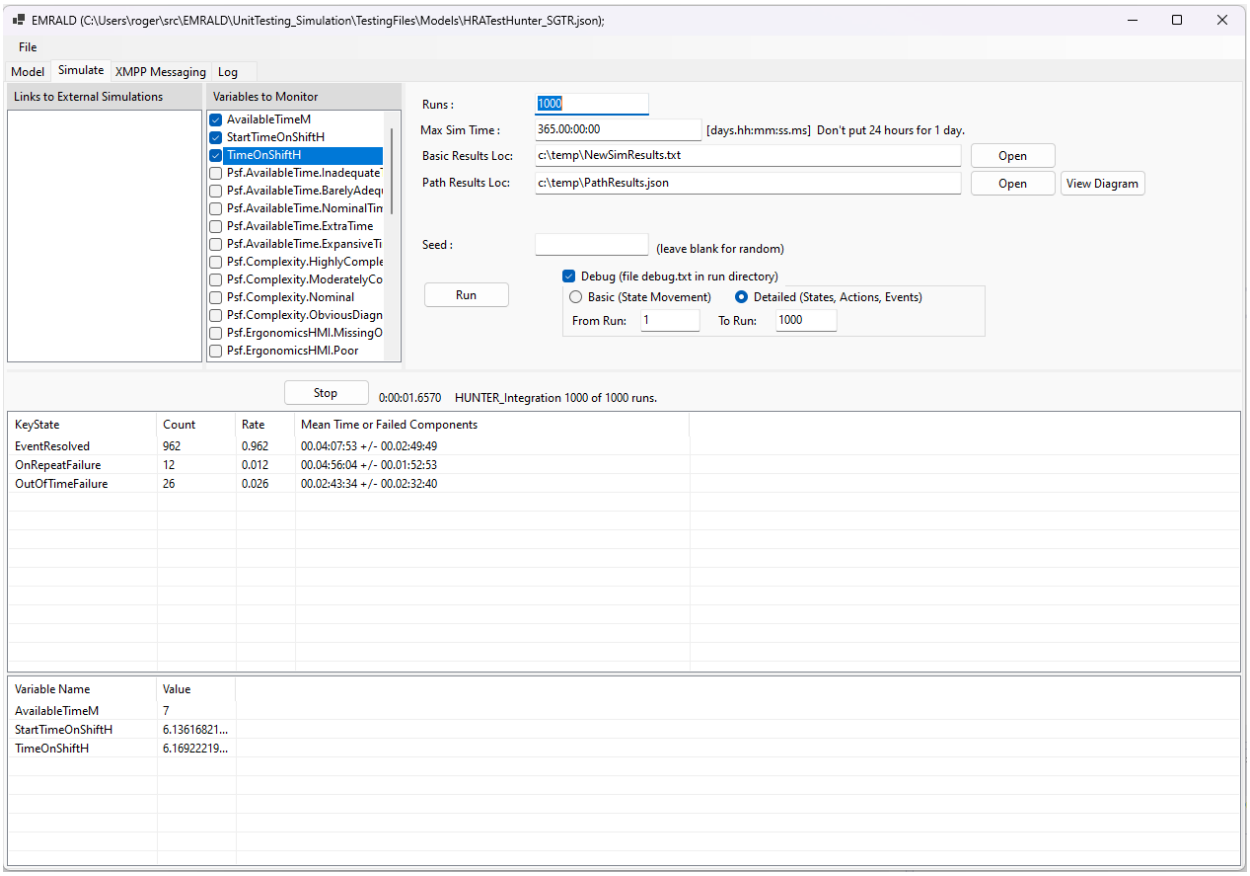


Figure 47. An example simulation result of the EMRALD-HUNTER SGTR model

Table 12 summarizes the simulation outputs of the EMRALD-HUNTER SGTR model depending on stress and time pressure. The stress levels are used as inputs for the dynamic Stress PSF evaluation for the HEP, while Time Pressure affects the time calculation only. In the table, there are three major outputs from the model, i.e., the number of failed scenarios, HEPs, and overtime failure counts. HEPs are calculated by dividing the number of failed scenarios by the number of scenarios (i.e., the number of trials). For the number of failed scenarios and HEPs, these values increase for higher stress level. In contrast, the overtime failure counts increase much less than the number of failed scenarios and HEPs depending on the higher stress level. Regarding Time Pressure, it is mainly relevant to the elapsed time. Figure 48 shows the average elapsed time on Stress and Time Pressure across SGTR scenarios. The figure indicates that the Time Pressure option dominates elapsed time, while the Stress level affects it less. In Table 12, if the Time Pressure option is applied in the simulation, overtime failure results in a relatively

low value. The reduced elapsed time Time Pressure prevents errors due to overtime. On the other hand, if Time Pressure is not applied, the overtime failure count increases.

Table 12. Simulation outputs of the EMRALD-HUNTER SGTR model for stress and time pressure

Stress	Time Pressure	The Number of Failed Scenarios	HEPs (The Number of Failed Scenarios / The Number of Scenarios)	Overtime Failure Count
Nominal	Yes	38	3.800e-2	26
	No	322	3.220e-1	321
High	Yes	162	1.620e-1	28
	No	382	3.820e-1	335
Extreme	Yes	403	4.030e-1	32
	No	512	5.120e-1	512

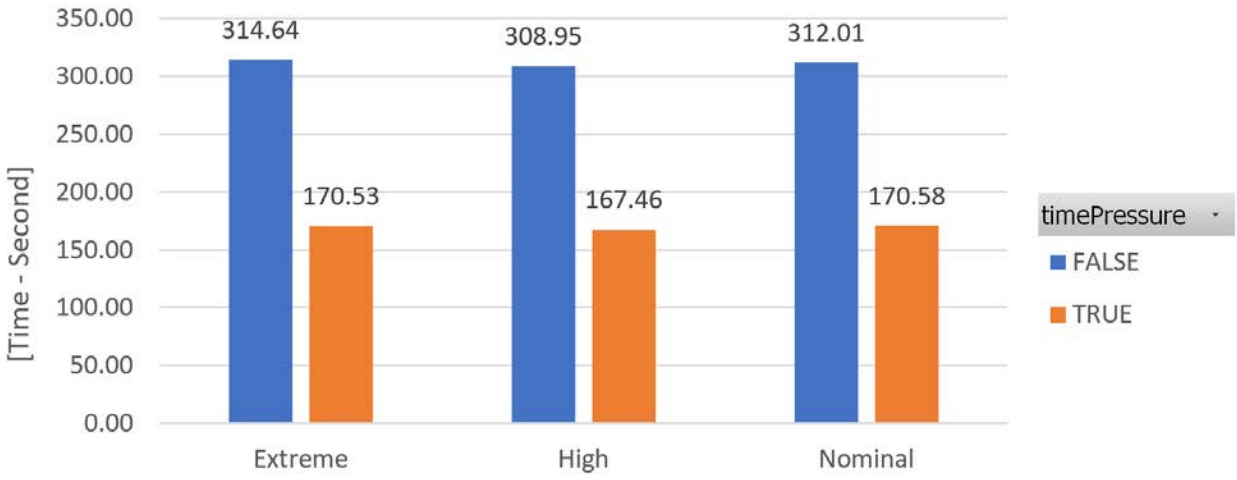


Figure 48. The elapsed time on stress level and time pressure in the SGTR scenarios

7.3 Loss of Feedwater (LOFW) Scenario

7.3.1 LOFW Description

In a PWR, the feedwater of the secondary side plays an essential role in producing electricity. The feedwater receives heat from the primary side to produce steam that drives the generator. It is also important to ensure that the feedwater on the secondary side is available for decay heat removal after the reactor shuts down. However, failure to supply the feedwater can reduce heat transfer to the secondary

side from serving as a heat sink, which can lead to increased pressure and temperature of the RCS. Thus, the reactor is eventually tripped with the reactor high pressure or steam generator low level. To address this issue, most PWRs are equipped with an auxiliary feedwater system. An accident resulting from the failure to supply feedwater to the steam generator is known as loss of feedwater, which is one of the design basis accidents (DBA).

In 1985, Davis-Besse Nuclear Power Plant experienced an actual LOFW accident at 90% operating power (U.S. Nuclear Regulatory Commission, 1985). At the time, one of two main feedwater pumps was in automatic control, and the other was in manual control. However, the automatic control pump stopped due to overspeed, leaving only the manually controlled pump operational. Furthermore, due to a spurious closure of the main steam isolation valve (MSIV), the turbine-driven main feedwater pump was unable to receive steam supply, rendering the redundant pump unusable as well. As a result, the plant experienced a loss of main feedwater.

Upon detecting a reduction in the steam generator water level, the operator expected the auxiliary feedwater supply system to automatically activate. However, the operator manually operated it before the steam generator low-level setpoint was reached. Unfortunately, during this process, the operator mistakenly pressed the valve to isolate the auxiliary feedwater supply system. As a result, the auxiliary feedwater pumps also stopped due to overspeed, leading to a total loss of feedwater. In general, it was advisable to initiate automatic actuation manually when failure is imminent, but this case shows the occurrence of operator errors cannot be ignored. Such errors may result from a lack of understanding of the plant’s state or mistakes in performance.

In the event of a potential boil dry situation caused by the LOFW, it was critical for the operator to quickly activate the auxiliary feedwater system. Fortunately, the operator responded promptly by resetting the control system and correcting the earlier error. However, the auxiliary feedwater valve that should have automatically reopened failed to open. Despite attempting to operate it manually from the main control panel, the valve remained unresponsive. At this critical juncture, the operator made the crucial decision to activate the startup feed pump. As it is motor-driven pump that does not require steam from the steam generator, it is a more reliable system for supplying feedwater. By supplying feedwater through the startup feed pump, the plant’s condition was stabilized.

The operator was supposed to perform feed and bleed operation according to procedure. However, the operator deviated from the procedure by recovering the auxiliary feedwater system instead of performing feed and bleed. Although the operator should have followed the procedure, it was later confirmed this approach may be more cost-effective. Furthermore, as the pressure increased due to the reactor coolant system overheating and steam generator boiling dry, the pressurizer pilot operated relief valve (PORV) opened and closed twice without the operator’s knowledge. The PORV did not close completely, resulting in a section where the pressure rapidly decreased, which the operator failed to notice. Fortunately, the PORV was eventually closed properly, and no further problems occurred. Nevertheless, the operator's lack of awareness could have been a significant contributing factor to the incident. It was evident that several operator actions are necessary in the LOFW, highlighting the role of human factors.

Similarly, the well-known Three Mile Island (TMI) accident was also triggered by LOFW, which led to loss of coolant Accident (LOCA) with a series of related complex events (U.S. Nuclear Regulatory Commission, 2022). The failure of both the main and auxiliary feedwater systems prevented the secondary side from being cooled, leading to automatic tripping of the turbine generator and the reactor, making heat removal difficult and increasing pressure in the primary system. In this case, the stuck open PORV caused the LOCA accident. These incidents also demonstrate the critical importance of understanding and addressing the role of operators in NPP events.

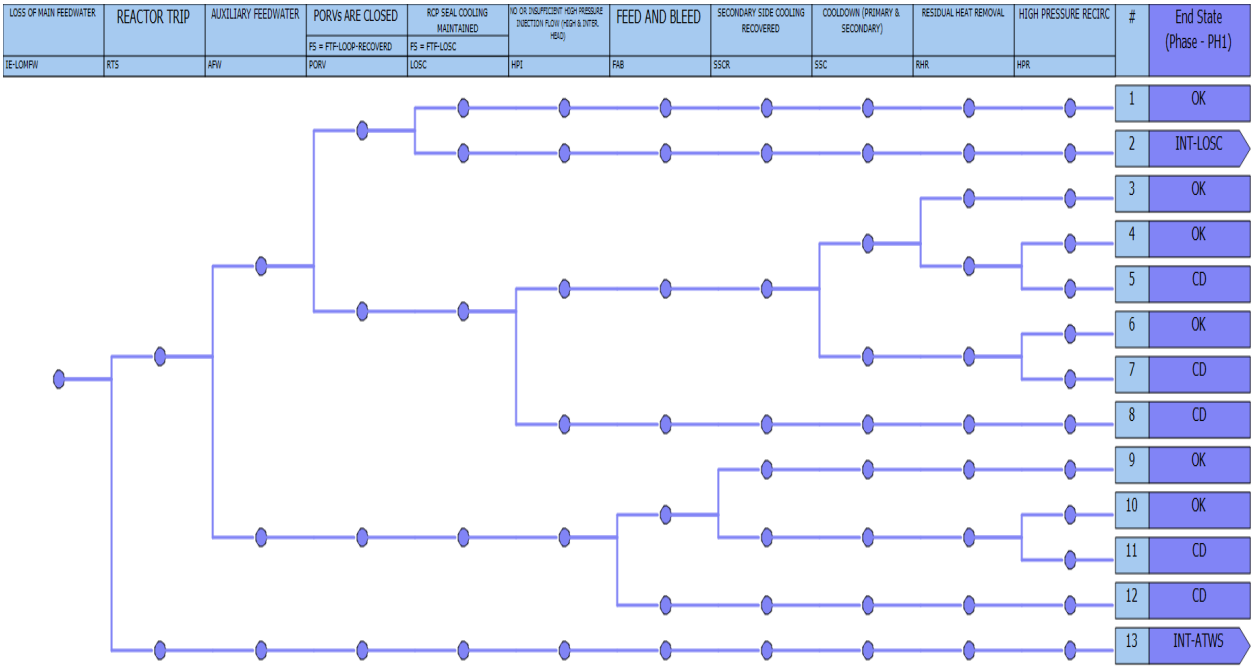


Figure 49. Generic LOFW event tree (from Ma et al., 2019)

Table 13. Event tree headings for LOFW (from Ma et al., 2019)

Heading	**Description**
IE-LOFW	Initiating event of loss of feedwater
RTS	Reactor trip
AFW	Supply of the auxiliary Feedwater system to steam generator
PORV	Close of pressurizer pilot operated relief valve (PORV)s
LOSC	Maintain of reactor coolant pump seal cooling
HPI	Injection of high-pressure safety injection system
FAB	Feed and bleed operation
SSCR	Recover of secondary side cooling
SSC	Cooldown of primary & secondary sides
RHR	Removal of residual heat
HPR	High pressure recirculation operation

7.3.2 LOFW PRA Modeling

Figure 49 depicts the event tree for LOFW (Ma et al., 2019). The event tree comprises the initiating event and 10 event tree headings as shown in Table 13, and 13 sequences are analyzed based on these headings. The event tree illustrates how these scenarios can lead to either a stable state or core damage. Table 14 provides several HFEs considered in LOFW.

Table 14. Generic human failure events in LOFW

Index	Description of Human Failure Events
1	Operators fail to respond with RPS signal present.
2	Operators fail to manually initiate AFW.
3	Operators fail to trip reactor coolant pumps.
4	Operators fail to depressurize RCS/secondary side (Rapid).
5	Operators fail to initiate emergency boration.
6	Operators fail to initiate feed and bleed cooling.
7	Operators fail to initiate feed and bleed cooling (Depend).
8	Operators fail to start high pressure recirculation.
9	Operators fail to restore HTX 1A after test or maintenance.
10	Operators fail to restore HTX 1B after test or maintenance.
11	Operators fail to restore train P1A after test or maintenance.
12	Operators fail to restore train P1B after test or maintenance.
13	Operators fail to recover offsite power in 1 hr.
14	Operators fail to align AC power given non-loss of offsite power LOOP IE.
15	Operators fail to control AFW turbine-driven pump (TDP) after battery depletion; Non-station blackout (SBO).

The first scenario of the event tree is defined as a stable state where AFW can be successfully supplied to the steam generator, the PORVs are properly closed, and the reactor coolant pumps (RCPs) seal cooling is maintained after reactor trip (Scenario 1). Scenario 2 defines a situation where the PORVs are properly closed, but RCP seal cooling fails, leading to a LOCA. Scenarios 3 - 8 require high-pressure injection (HPI) due to an improperly open PORV and possible coolant leakage. If HPI fails, coolant leakage continues, resulting in core damage (scenario 8). If HPI is successful, the plant's stability depends on whether the primary and secondary side cooling systems can be successfully cooled down. If cooldown fails, but high-pressure recirculation (HPR) is successful, the plant may be stable (scenario 6). If HPR fails, the plant is defined as core damaged (scenario 7). If either the primary or secondary side cooldown is successful, with or without residual heat removal, the plant is considered stable (scenarios 3 and 4).

In the event of LOFW, if the AFW system fails to supply feedwater, heat removal can still be achieved through feed and bleed operation. After successful feed and bleed operation, if neither secondary side cooling recovers and HPR fails, the plant is deemed core damaged (scenario 11). If either one succeeds, the plant is defined as stable state (scenarios 9 and 10). If the feed and bleed operation also fails, it results in core damage (scenario 12). In case the reactor trip fails, it is analyzed as at ATWS scenario.

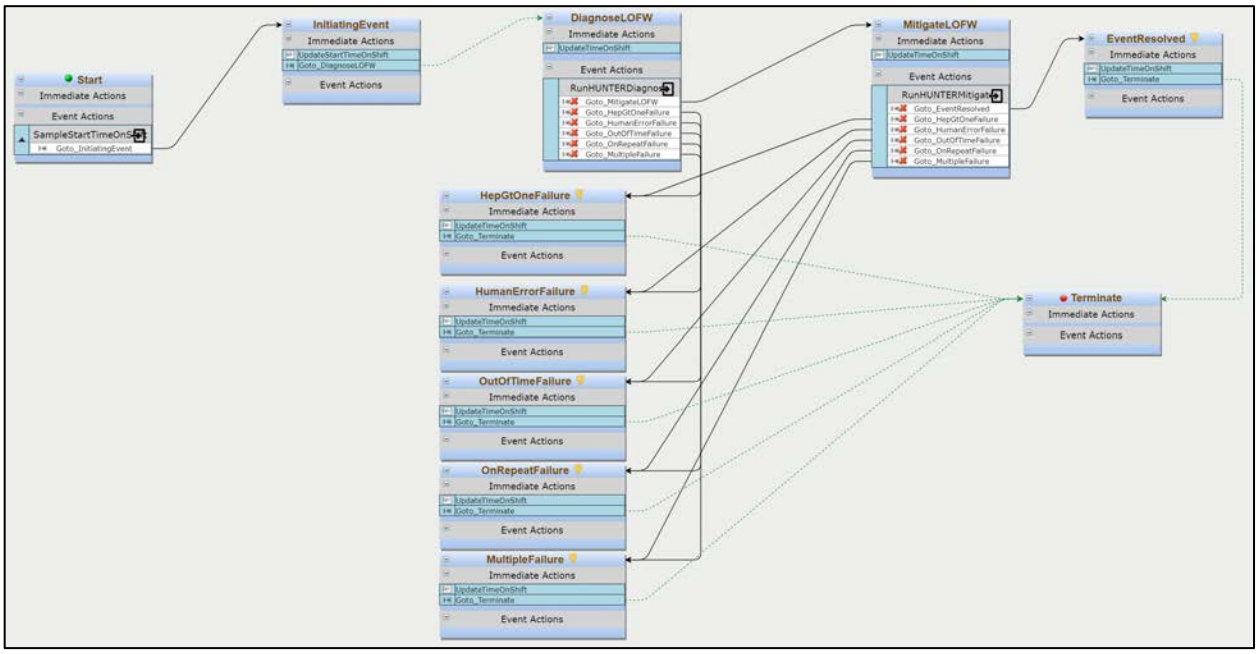


Figure 50. The EMRALD-HUNTER LOFW model

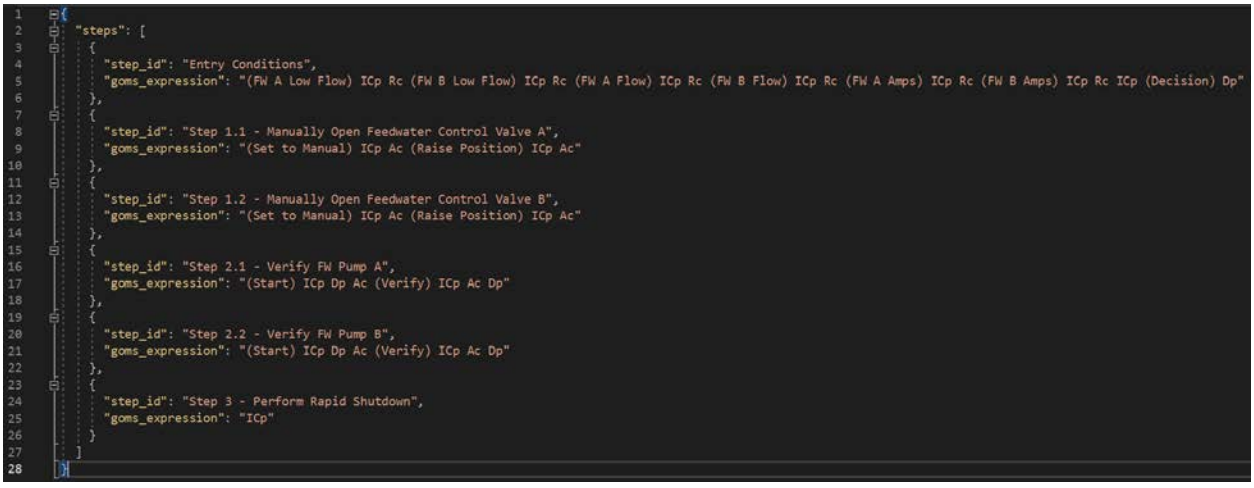


Figure 51. The procedure contents coded for diagnosing LOFW within EMRALD-HUNTER

7.3.3 EMRALD-HUNTER LOFW Model

This section introduces a LOFW model developed under EMRALD-HUNTER. Figure 50 (repeated from Figure 12) shows the EMRALD-HUNTER LOFW model. The diagram follows a similar scheme to the SGTR scenario described in Section 7.2.2. In the simulation for LOFW model, the file depicted in Figure 51 is used to define the human actions for each step for diagnosing LOFW, while the rapid shutdown process similar to SGTR and represented earlier in Figure 46 is carried out to mitigate the LOFW.

Table 15 summarizes the simulation outputs of the EMRALD-HUNTER LOFW model depending on stress and time pressure. Figure 52 indicates the average elapsed time on stress and time pressure in LOFW scenarios. These show similar tendencies in the outputs depending on the Stress level and Time Pressure option.

Table 15. The simulation outputs of the EMRALD-HUNTER LOFW model depending on stress and time pressure

Stress	Time Pressure	The Number of Failed Scenarios	HEPs (The Number of Failed Scenarios / The Number of Scenarios)	Overtime Failure Count
Nominal	Yes	11	1.100e-2	8
	No	192	1.920e-1	192
High	Yes	73	7.300e-2	6
	No	227	2.270e-1	216
Extreme	Yes	272	2.720e-1	5
	No	346	3.460e-1	244

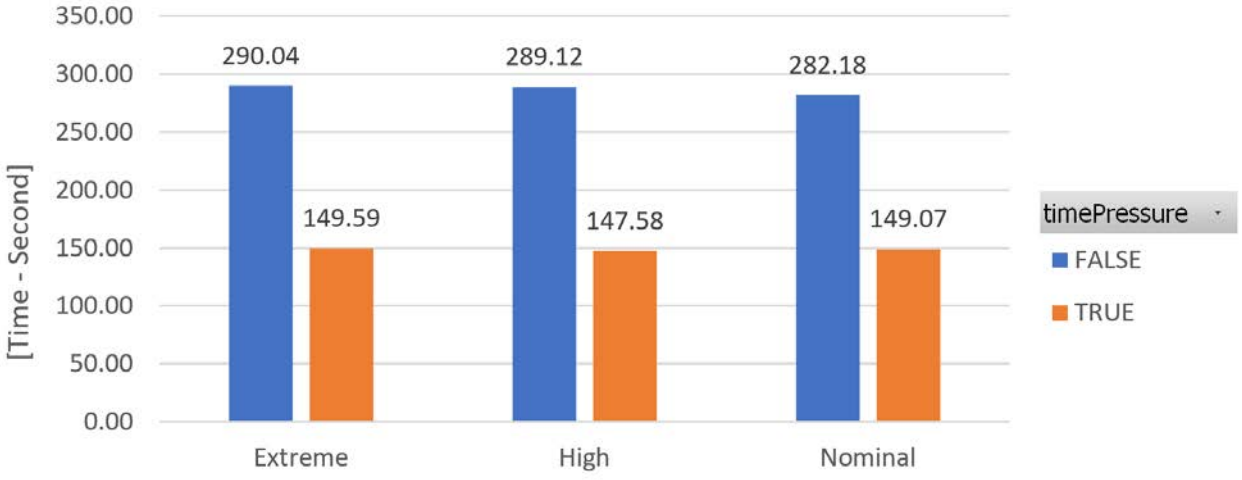


Figure 52. The elapsed time on stress level and time pressure in LOFW scenarios

8. DISCUSSION

This report has captured the coupling of EMRALD and HUNTER, providing an embedded tool for HRA to be used with dynamic PRA in EMRALD. The embedded version of HUNTER demonstrated the ability to simplify key features of HUNTER such as decision logic in the Task module or the linked plant model in the Environment module to allow it to function within the dynamic event scheduling provided by EMRALD. HUNTER embedded within HUNTER was able to use the existing interface in EMRALD with only minor additions, making HUNTER a seamlessly integrated addition to EMRALD.

This report demonstrated the utility of EMRALD-HUNTER with two accident scenarios, SGTR and LOFW, which had previously also been run with the standalone version of HUNTER. The Monte Carlo runs in EMRALD readily produced HEPs and task durations. These can be compared to the results from the *International HRA Empirical Study*, published as several volumes of NUREG/IA-0216. NUREG/IA-0216, Volume 2 (Bye et al., 2011) reviews the results from a large-scale simulator study for SGTR. The scenarios for diagnosing and mitigating the SGTR in EMRALD-HUNTER correspond to HFE-1A and HFE-2A as presented in Figure 53. The predicted HEPs from EMRALD-HUNTER for nominal Stress levels are shown as an orange rectangle and fall within the confidence bounds for the empirical data. These data suggest that EMRALD-HUNTER does a good job of predicting the HEPs within the modeled scenarios for SGTR. Note that NUREG/IA-0216 does not break down task durations in a way that allows comparison between the empirical data and the predicted times from EMRALD-HUNTER.

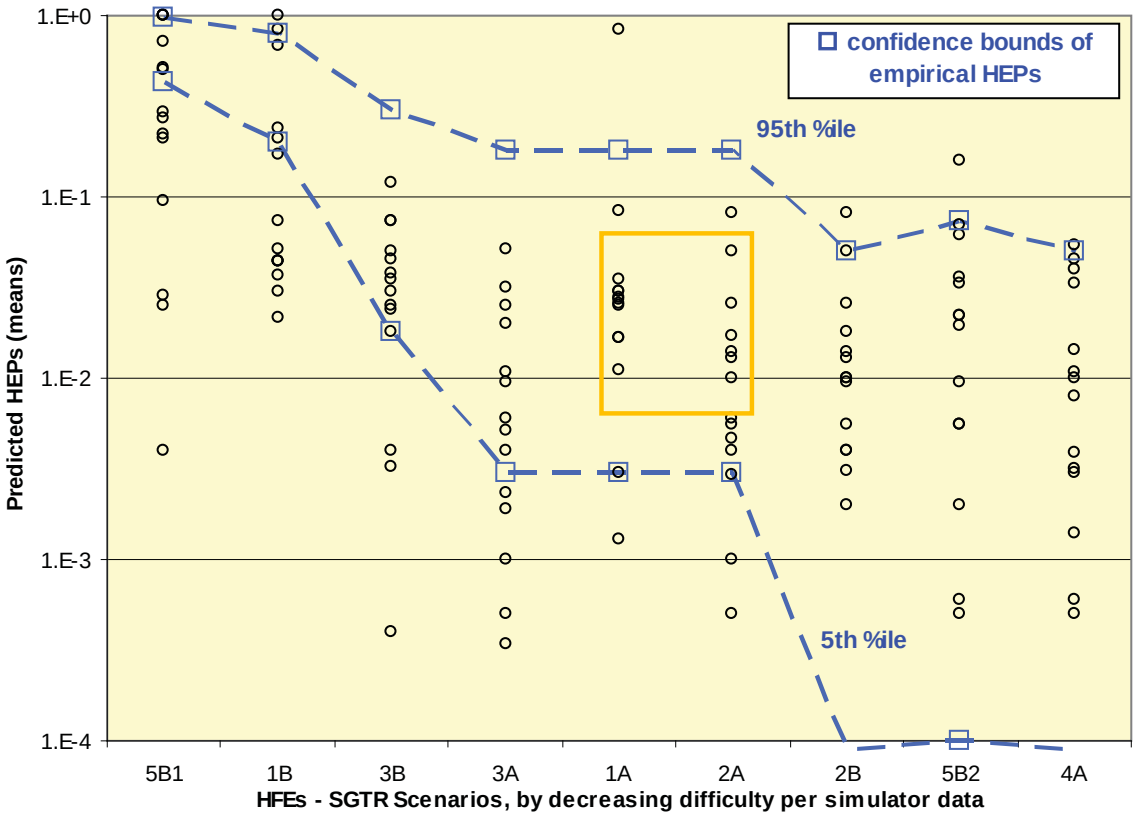


Figure 53. Empirical data for HEPs for SGTR overlaid with predicted HEPs from EMRALD-HUNTER

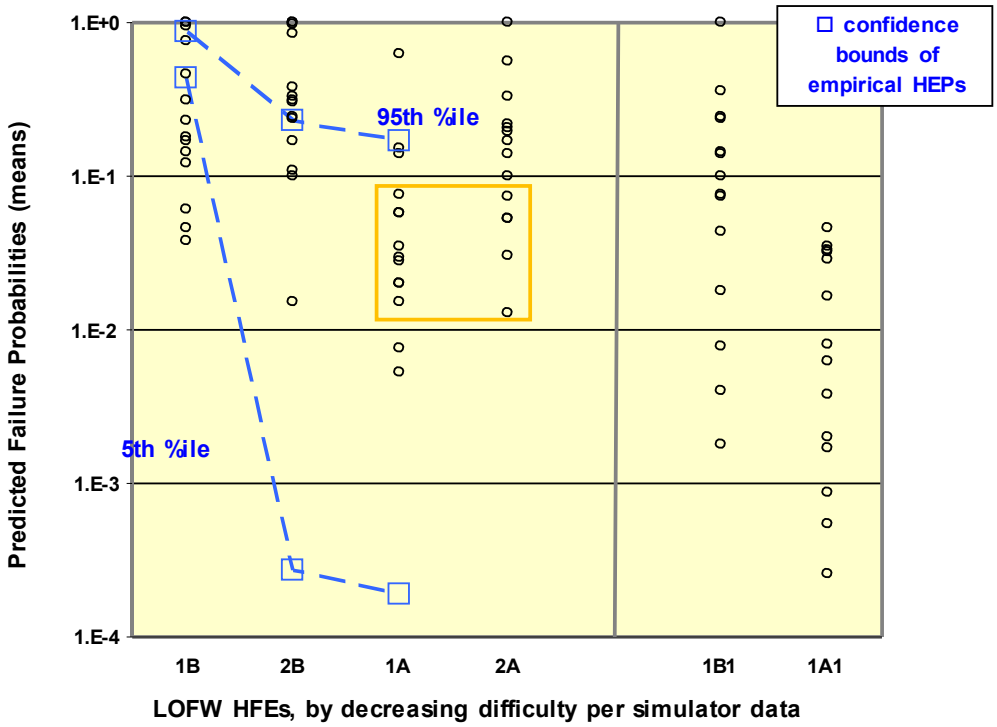


Figure 54. Empirical data for HEPs for SGTR overlaid with predicted HEPs from EMRALD-HUNTER

NUREG/IA-0216, Volume 3 (Dang et al., 2012) reports of the results of a related simulator study for LOFW. The scenarios modeled in EMRALD-HUNTER correspond to HFE-1A and HFE-2A shown in Figure 54. Due to a data constraint, the confidence bounds could not be determined for HFE-2A in NUREG/IA-0216, Volume 3. Nonetheless, as can be seen by the orange box in the figure, the predicted HEPs for LOFW with nominal Stress fall within the observed levels from the empirical data. The data suggest that EMRALD-HUNTER accurately predicts HEPs within the modeled scenarios for LOFW.

While HUNTER provides new HRA functions to support EMRALD modeling, it must be noted that the embedded HUNTER represents a simplified variant of HUNTER. Development of the standalone version of HUNTER will continue to serve modeling efforts that:

- Involve more realistic human performance modeling at the subtask level than is provided in the embedded version of HUNTER
- Have a considerable number of procedures that would benefit from the procedure authoring tools in the full version of HUNTER
- Consider the effects of deviations between work as imagined and work as done in procedure following
- Require complex branching logic in procedures beyond what is supported in EMRALD-HUNTER

- Benefit from additional PSFs and modeling of additional effects of PSFs beyond task duration and HEPs
- Use more frequently added new features including exploratory modeling aspects of HRA
- Need an included plant simulation for modeling plant phenomena.

At the same time, the embedded version of HUNTER affords numerous advantages, especially when looking for a streamlined version of HRA to add to dynamic PRA. The two versions of HUNTER can readily co-exist. Figure 55 provides a comparison table to assist in understanding when EMRALD-HUNTER vs. standalone HUNTER might be most appropriate.

Use EMRALD-HUNTER	Use standalone HUNTER
• When you wish to add HRA to EMRALD models • When you are analyzing human actions at the HFE level • When you are primarily interested in task duration and error for human actions • When you want to use external libraries and features linked to EMRALD	• When you are exploring psychological phenomena behind human actions • When you are analyzing human actions at the task or subtask level • When you are interested in performance measures beyond task duration and error • When you are developing human event sequences for later use in EMRALD • When you are analyzing and optimizing procedures for human event sequences

Figure 55. Guidance on when to use EMRALD-HUNTER vs. the standalone version of HUNTER

This report chronicles initial features of HUNTER embedded into EMRALD and the successful demonstration of EMRALD-HUNTER. Additional testing and demonstration of EMRALD-HUNTER is planned beyond the SGTR and LOFW scenarios presented in this report. For example, significant modeling work has already been done using EMRALD for HRA modeling with FLEX emergency mitigation (Park et al., 2021) and physical security (Christian et al., 2023). The integration of HUNTER into EMRALD is informed by these earlier efforts and the challenges that were incurred in modeling HRA efficiently without a specific HRA module in EMRALD. Revisiting earlier analyses with the embedded HUNTER functionality would be a good place to see the benefits of the new approach. As additional use cases and demonstrations are explored, the development team represented in this report will consider desirable new features to incorporate into future versions of EMRALD-HUNTER.

9. REFERENCES

Ashour, A., Ashcroft, D. M., & Phiipps, D. L. (2021). Mind the gap: Examining work-as-imagined and work-as-done when dispensing medication in the community pharmacy setting. Applied Ergonomics, 93, Article 103372.

Belenky, G. (1994). The Effects of Sleep Deprivation on Performance During Continuous Combat Operations. In Institute of Medicine (Ed.), Food Components to Enhance Performance: An Evaluation of Potential Performance-Enhancing Food Components for Operational Rations (pp. 127-136). The National Academies Press.

Boring, R. (2010). How many performance shaping factors are necessary for human reliability analysis. In Proceedings of the 10th International Probabilistic Safety Assessment and Management Conference.

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

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

Boring, R.L., & Blackman, H.S. (2007). The origins of the SPAR-H method's performance shaping factor multipliers. Official Proceedings of the Joint 8th IEEE Conference on Human Factors and Power Plants and the 13th Annual Workshop on Human Performance/Root Cause/Trending/Operating Experience/Self-Assessment (pp. 177-184).

Boring, R., Lew, R., Ulrich, T., & Park, J. (2023, in press). Synchronous vs. asynchronous coupling the HUNTER dynamic human reliability analysis framework. Proceedings of Applied Human Factors and Ergonomics (AHFE) 2023.

Boring, R.L., & Rasmussen, M. (2016). GOMS-HRA: A method for treating subtasks in dynamic human reliability analysis. Proceedings of the European Safety and Reliability Conference (pp. 956-963).

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 (pp. 1024-1031).

Boring, R., Ulrich, T., Mortenson, T., & Gertman, D. (2020). Fatigue as a performance shaping factor in human reliability analysis for long-duration spaceflight. Proceedings of the 64th International Annual Meeting of the Human Factors and Ergonomics Society (pp. 1681-1685).

Boring, R.L., Ulrich, T.A., 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 National Laboratory.

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 Conference (pp. 433-439).

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

Campbell, R.D., & Bagshaw, M. (2002). Human Performance and Limitation in Aviation (3rd ed.). Blackwell Science.

Choi, Y.-J. (2020). Assessment of verification and validation status—EMRALD and HUNTER (INL/EXT-20-59904). Idaho National Laboratory.

Christian, R., Yadav, V., Prescott, S.R., & St Germain, S.W. (2022). A dynamic risk framework for the optimization of physical security posture of nuclear power plants. Proceedings of the Probabilistic Safety Assessment and Management Conference (PSAM 16), Paper RO316.

Cowan, N. (2008). What are the differences between long-term, short-term, and working memory? Progress in Brain Research, 169.

Curry, L., & Gass, D. (1987). Effects of training in cardiopulmonary resuscitation on competence and patient outcome. Canadian Medical Association Journal, 137.

Dang, V.N., Forester, J., Boring, R., Broberg, H., Sassaiu, S., Julius, J., Männistö, I., Nelson, P., Lois, E., and Bye, A. (2012). International HRA Empirical Study—Phase 3 Report—Results from Comparing HRA Method Predictions to Simulator Data on LOFW Scenarios (NUREG/IA-0216, Vol. 3). U.S. Nuclear Regulatory Commission.

Dhillon, B. S. (2007). Human Reliability and Error in Transportation Systems. Springer London.

Dorin, R.I., Qiao, Z., Qualls, C.R., & Urban III, F.K. (2012). Estimation of maximal cortisol secretion rate in healthy humans. The Journal of Clinical Endocrinology & Metabolism, 97(4), 1285-1293.

Folkard, S. (1997). Black times: Temporal determinants of transport safety. Accident Analysis & Prevention, 29(4), 417-430.

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

Groth, K., & Mosleh, A. (2009). A Data-informed Model of Performance Shaping Factors for Use in Human Reliability Analysis. Center for Risk and Reliability, University of Maryland, College Park.

Heitz, R.P. (2014). The speed-accuracy tradeoff: History, physiology, methodology, and behavior. Frontiers in Neuroscience, 8, Article 150.

Hollnagel, E. (1998). Cognitive Reliability and Error Analysis Method (CREAM). Elsevier.

Jaber, M.Y., & Sikstrom, S. (2004). A numerical comparison of three potential learning and forgetting models. International Journal of Production Economics, 92.

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 National Laboratory.

Jung, W., Kim, J., Park, J., Jeoung, K., Kang, D., & Ha, J. (2005). Development of a Standard Human Reliability Analysis Method of Nuclear Power Plants (KAERI/TR-2961/2005). Korea Atomic Energy Research Institute.

Kim, J.T., Kim, J., Seong, P.H., & Park, J. (2021). Quantitative resilience evaluation on recovery from emergency situations in nuclear power plants. Annals of Nuclear Energy, 156, Article 108220.

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

Kolaczkowski, A., Forester, J., Lois, E., & Cooper, S. (2005). Good Practices for Implementing Human Reliability Analysis (NUREG-1792). U.S. Nuclear Regulatory Commission.

Ma, Z, Parisi, C., Davis, C., Boring, R.L., Zhang, H., & Park, J. (2019). Risk-Informed Analysis for an Enhanced Resilient PWR with ATF, FLEX, and Passive Cooling (INL/EXT-19-53556). Idaho National Laboratory.

MacDonald, P.E., Shah, V.N., Ward, L.W, & Elliosn, P.G. (1996). Steam Generator Tube Failures (NUREG/CR-6365). U.S. Nuclear Regulatory Commission.

Manurung, A.D.R., Shanti, I., & Mardhatillah, A. (2022). In flight emergency decision-making process: Does Intuition Matter? Acta Medica Philippina, 56.

Murre, J.M.J., & Dros, J.D. (2015). Replication and analysis of Ebbinghaus's forgetting curve. PLOS ONE.

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

Park, B., Lee, S., Yang, T., Choi, J. H., Park, J., Arigi, A., Boring, R. L., & Kim, J. (2021). An experimental analysis on the CNS simulator comparing human performance between operators and students. In Korean Nuclear Society Virtual Autumn Meeting.

Park, J., Ulrich, T.A., Boring, R.L., Zhang, S., Ma, Z., & Zhang, H. (2021). Modeling FLEX human actions using the EMRALD dynamic risk assessment tool. In International Topical Meeting on Probabilistic Safety Assessment and Analysis (PSA 2021) (pp. 1171-1180).

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, Article 109502.

Prescott, S., Nevius, D., Ma, Z., & Lawrence, S. (2022). Using EMRALD to Simplify and Perform Dynamic Analysis with MAAP. PSAM 16, Honolulu, Hawaii, June 26 - July 1, 2022.

Prescott, S., Smith, C., & Vang, L. (2018). EMRALD, Dynamic PRA for the Traditional Modeler. PSAM 14, Los Angeles, CA, Sep. 2018.

Prescott, S., Wood, T., & Ziccarelli, M.. (2022). Dynamic and Classical PRA Coupling Using EMRALD and SAPHIRE (INL/RPT-22-70424). Idaho National Laboratory.

Rasmussen, M., & Boring, R. L. (2016). Implementation of complexity in computation-based human reliability analysis. Proceedings of the European Safety and Reliability Conference (pp. 972-977).

Sezen, H., Hur, J., Smith, C., Aldemir, T., & Denning, R. (2019). A computational risk assessment approach to the integration of seismic flooding hazards with internal hazards. Nuclear Engineering and Design, 355, Paper 110341.

Spencer, M., Robertson, K., & Folkard, S. (2006). The Development of a Fatigue/Risk Index for Shift Workers. Health and Safety Executive Report, 446.

Swain, A. D. (1987). Accident Sequence Evaluation Program Human Reliability Analysis Procedure (NUREG/CR-4772). U.S Nuclear Regulatory Commission.

Swain, A.D., & Guttmann, H.E. (1983). Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications (NUREG/CR-1278). U.S Nuclear Regulatory Commission.

Swaton, E., Neboyan, V., & Lederman, L. (1987). Human factors in the operation of nuclear power plants. IAEA Bulletin.

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.

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.

U.S. Nuclear Regulatory Commission. (1985). Loss of Main and Auxiliary Feedwater Event at the Davis-Besse Plant on June 9, 1985 (NUREG-1154). U.S. Nuclear Regulatory Commission.

U.S. Nuclear Regulatory Commission. (1988). NRC Integrated Program for the Resolution of Unresolved Safety Issues A-3, A-4, and A-5 Regarding Steam Generator Tube Integrity, Final Report (NUREG-0844). U.S. Nuclear Regulatory Commission.

U.S. Nuclear Regulatory Commission. (2022). Backgrounder on the Three Mile Island Accident. Fact Sheet.

Vitousek, M.N., Taff, C.C., Ardia, D.R., Stedman, J.M., Zimmer, C., Salzman, T.C., & Winkler, D.W. (2018). The lingering impact of stress: Brief acute glucocorticoid exposure has sustained, dose dependent effects on reproduction. Proceedings of the Royal Society B: Biological Sciences, 285(1882), 20180722.

White, K.G. (2001). Forgetting functions. Animal Learning & Behavior, 29.

Wixted, J.T., & Ebbensen, E.B. (1991). On the form of forgetting. Psychological Science, 2