INL/RPT-22-66564

Light Water Reactor Sustainability Program

# Software Implementation and Demonstration of the Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER)

March 2022

U.S. Department of Energy

Office of Nuclear Energy

INL/RPT-22-66564

# Software Implementation and Demonstration of the Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER)

Ronald Boring, Thomas Ulrich, Jeeyea Ahn, Yunyeong Heo, Jooyoung Park

March 2022

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

This page intentionally left blank.

## ABSTRACT

The Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER) framework is implemented as a standalone software tool to support dynamic human reliability analysis. This report documents an updated conceptual framework, called HUNTER 2, that stresses the flexibility, modularity, and scalability of the software tool to allow adaptability to a wide range of dynamic human risk modeling scenarios. The conceptual framework is implemented as a Python executable library. The Task module is based on operating procedures at nuclear power plants and serves as the scheduler. The Individual module represents influences on individual performance that are captured through dynamically calculated performance shaping factors. The Environment module captures the interaction with a physical world model, in this case thermal-hydraulic modeling using RELAP5-3D. As a proof-of-concept demonstration, base and complex variants of a steam generator tube rupture scenario are modeled in HUNTER and validated against empirical data. A unique aspect of the HUNTER software is its ability to support calculations beyond human error probabilities, such as human action durations. The report concludes with a discussion of future activities. One area of considerable focus is ensuring additional scenarios are modeled to support emerging industry needs for human reliability analysis.

## ACKNOWLEDGMENTS

The authors wish to thank the Risk-Informed System Analysis (RISA) Pathway of the U.S. Department of Energy's Light Water Reactor Sustainability Program for its support of the research activities presented in this report. In particular, we thank Svetlana Lawrence, pathway lead for RISA, for her contributions to shaping the industry focus of the software development; Jong-Yoon Choi, for his careful review of the original HUNTER framework and recommendations for software implementation; and Curtis Smith, director for the Nuclear Safety and Regulatory Research division at Idaho National Laboratory, for his championing of dynamic and computation-based human reliability analysis approaches. Jeeyea Ahn and Yunyeong Heo were visiting students from Ulsan National Institute of Science and Technology. We gratefully acknowledge their support through the Korea Nuclear International Cooperation Foundation (KONICOF).

# CONTENTS

## FIGURES

## TABLES

# ACRONYMS

| | |
|---|---|
| $A_C$ | actions in control room |
| $A_F$ | actions in field |
| AFW | auxiliary feedwater |
| AHC | abstraction hierarchy complexity |
| AOP | abnormal operating procedure |
| API | advanced programming interface |
| ATWS | anticipated transient without scram |
| $C_C$ | checking in control room |
| $C_F$ | checking in field |
| CST | condensate storage tank |
| CSV | comma separated value |
| DBA | design-basis accident |
| $D_P$ | decisions with procedures |
| $D_W$ | decisions without procedures |
| EDC | engineering decision complexity |
| EMRALD | Event Modeling Risk Assessment using Link Diagrams |
| EOP | emergency operating procedure |
| EQ | equalization |
| ES | emergency supplement |
| ET | event tree |
| FB | feed and bleed |
| FLEX | flexible plant operations |
| GBWR | Generic Boiling Water Reactor |
| GOMS | Goals, Operators, Methods, Selection rules |
| GPWR | Generic Pressurized Water Reactor |
| HAMMLAB | Halden Human-Machine Laboratory |
| HEP | human error probability |
| HFE | human failure event |
| HPI | high-pressure injection |
| HPR | high-pressure recirculation |
| HRA | human reliability analysis |
| HSI | human-system interaction |
| HSSL | Human Systems Simulation Laboratory |
| HUNTER | Human Unimodel for Nulcear Technology to Enhance Reliability |
| INIT | initiator |
| INL | Idaho National Laboratory |
| $I_P$ | producing instructions |
| $I_R$ | receiving instructions |
| JSON | JavaScript Object Notation |
| KONICOF | Korea Nuclear International Cooperation Foundation |
| LOB | loss of battery |
| LODG | loss of diesel generator |
| LOOP | loss of offsite power |
| LWR | light-water reactor |
| LWRS | Light Water Reactor Sustainability |
| MFW | main feedwater |
| MOOSE | Multiphysics Object-Oriented Simulation Environment |

| | |
|---|---|
| MSIV | main steam isolation valve |
| MU | makeup |
| NPP | nuclear power plant |
| NRC | Nuclear Regulatory Commission |
| NUREG | Nuclear Regulatory Commission Report |
| OP | operating procedure |
| PORV | pressure operated relief valve |
| PRA | probabilistic risk assessment |
| PSF | performance shaping factor |
| QA | quality assurance |
| RAVEN | Risk Analysis Virtual Code Environment |
| $R_C$ | retrieve information in control room |
| RCS | reactor coolant system |
| RELAP | Reactor Excursion and Leak Analysis Program |
| $R_F$ | retrieve information in field |
| RF | refilling |
| RHR | residual heat removal |
| RISA | Risk Informed System Analysis |
| RNO | response not obtained |
| RPS | reactor protection system |
| R-TACOM | Revised Task Complexity |
| RWST | reactor water storage tank |
| $S_C$ | selecting on control boards |
| $S_F$ | selecting in field |
| SG | steam generator |
| SGTR | steam generator tube rupture |
| SHERPA | Systematic Human Error Reduction and Prediction Approach |
| SHR | secondary heat removal |
| SI | safety injection |
| SIC | step information complexity |
| SLC | step logic complexity |
| SPAR-H | Standardize Plant Analysis Risk-Human |
| SSC | (1) secondary side cooling *OR* (2) step size complexity |
| TACOM | Task Complexity |
| TEJUN | Task Engine for Job and User Notification |
| THERP | Technique for Human Error Rate Prediction |
| TR | task structure |
| TRL | Technology Readiness Level |
| TS | task scope |
| TU | task uncertainty |
| U.S. | United States |
| V&V | verification and validation |
| W | wait |

# SOFTWARE IMPLEMENTATION AND DEMONSTRATION OF THE HUMAN UNIMODEL FOR NUCLEAR TECHNOLOGY TO ENHANCE RELIABILITY (HUNTER)[a]

# 1. INTRODUCTION

## 1.1 Project Background

The U.S. Department of Energy's Light Water Reactor Sustainability (LWRS) Program was established to enable the long-term operation of existing domestic nuclear power plants (NPPs). The viability of the fleet of commercial reactors is enhanced through innovative approaches to improve the economics of light-water reactors (LWRs). LWRS consists of five research pathways (LWRS 2021):

- *Plant Modernization*, focused on enabling plant efficiency improvements through long-term modernization
- *Flexible Plant Operation and Generation*, focused on enabling diversification through non-electrical products
- *Risk-Informed System Analysis (RISA)*, focused on developing analysis methods and tools to optimize safety and economics
- *Materials Research*, focused on understanding and predicting the behavior of materials
- *Physical Security*, focused on developing technologies to optimize physical security.

With the exception of the Materials Research pathway, the LWRS pathways have projects that consider the human contribution to plants. The projects look for ways to achieve efficiencies in plant staffing while still maintaining the same or better performance, safety, and security of existing plants. Efficiencies may come in the form of developing tools and processes that allow staff to perform existing tasking better. For example, creating analytic tools that predict maintenance needs rather than following a conservative prescriptive maintenance schedule can potentially save the equipment cost of prematurely replacing equipment (Agarwal et al. 2021). Such tools have the added human benefit of lowering the demands on maintenance staff.

Human reliability analysis (HRA) is the study of human error, specifically how to identify sources and contributors of human error and how to quantify that error (Boring 2009). The RISA pathway sponsors a number of HRA-related projects that aim to create better tools to support industry risk assessment needs. One such project is the Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER) project (Boring et al. 2016). HUNTER is a framework to support the dynamic modeling of human error in conjunction with other modeling tools. HUNTER creates a virtual operator or, potentially, a human digital twin (Shengli 2021) as a human operations counterpart to plant hardware modeling and simulation. The name HUNTER is meant as a counterpart to the various animal-named modeling tools developed at Idaho National Laboratory (INL), such as Risk Analysis Virtual Code ENvironment (RAVEN; Rabiti et al. 2017) and Multiphysics Object-Oriented Simulation Environment

[a] This report expands an earlier version of this report published as INL/EXT-21-64525, *An Adaptable Software Toolkit for Dynamic Human Reliability Analysis: Progress Toward HUNTER 2* (Boring et al. 2021b).

(MOOSE; Permann et al. 2020). These tool names playfully combine to become tools like RAVEN-HUNTER or MOOSE-HUNTER.[b]

The theoretical underpinnings of HUNTER were developed previously (Boring et al. 2016). We refer to these earliest efforts as HUNTER 1. However, little effort had been devoted to making HUNTER a software tool that could be integrated into industry efforts. It remained a research framework for dynamic HRA efforts, but these efforts did not combine into a single solution. As the HUNTER project continues, the main goals are now twofold:

- Create a usable and adaptable standalone software tool
- Develop example applications and use cases to meet industry HRA needs.

This report addresses the first goal, namely the development of the HUNTER software, here named HUNTER 2 to disambiguate it from the earlier efforts. The disparate elements of the HUNTER 1 framework have now been formalized and implemented as an executable Python library. The initial version is a fully functional proof of concept, including a graphical user interface to simplify the process of model building and editing. These refinements go hand in hand with the planned development of additional use cases and accident scenario models. This report includes a sample model using the HUNTER software for a steam generator tube rupture (SGTR) scenario. As additional scenarios are modeled, new software features and modeling functions will be included to facilitate a greater utility of HUNTER for a wide range of applications.

## 1.2 Benefits of HUNTER as a Dynamic HRA Framework

Historically, HRA was developed as a worksheet solution, suitable for supporting static probabilistic risk assessments (PRAs). Static HRAs and PRAs review a particular snapshot of possible outcomes, but they do not model a dynamic event progression. The changing event progression—the defining characteristic of dynamic HRAs and PRAs—allows the modeling of the range of activities and outcomes as well as the consideration of a variety of what-if scenarios that would prove onerous to perform manually with static methods. Dynamic HRA can also be used to model scenarios for which there is minimal operational experience to explore what outcomes may emerge because of different human responses. This capability is especially useful for emerging areas of interest in risk modeling, such as severe accidents, HRA for human interactions with advanced technologies like digital and automated human-system interfaces, balance-of-plant activities beyond the main control room, and specialized areas like flexible equipment use and physical security modeling. As work on developing sample analyses in HUNTER continues, it is important to demonstrate the additional risk insights afforded by dynamic modeling that would not be possible with conventional static methods. An easy-to-use software tool that can help bring new risk insights is essential for industry as it supports new risk requirements.

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

[b] While a literal "hunter" is usually antithetical to the longevity of the animals with which it associates, the HUNTER framework here is meant to complement the capabilities of the animal modeling methods and thereby ensure their long lifespan.

Dynamic HRA—and, by extension, HUNTER—will succeed as risk tools only if they provide true benefits to the risk analysts who use them. Dynamic HRA offers the potential to provide deeper modeling fidelity; opportunities for exploring the ranges of human performance; the ability to extrapolate HRA to new scenarios, technologies, and domains; and the prospect to model output types beyond HEPs. However, dynamic HRA does not accomplish these advantages over static HRA without costs. Dynamic HRA can be considerably more complex to set up and model. As such, HUNTER strives to strike a balance by creating a uniquely simple and adaptable software tool that may be readily used by risk analysts to model phenomena of interest.

### 1.3 Report Structure

The remainder of this report is structured as follows:

- Section 2 outlines previous efforts at developing HUNTER and supporting HRA tools
- Section 3 presents an updated conceptual framework for HUNTER to support implementation as standalone software
- Section 4 discusses the current software implementation of the HUNTER Task module
- Section 5 provides the technical basis behind the HUNTER Individual module implementation
- Section 6 overviews the implementation of the HUNTER Environment module
- Section 7 demonstrates the features and implementation of the HUNTER graphical user interface
- Section 8 provides background and a demonstration of an SGTR scenario
- Section 9 considers shortcomings and next steps for HUNTER software development and model development activities.

# 2. PREVIOUS HUNTER EFFORTS

## 2.1 HUNTER 1 Framework

The HUNTER framework is a computational HRA[c] approach to dynamically model human cognition and actions as well as incorporate these respective elements into a PRA framework (Boring et al. 2016; Joe et al. 2015). Many researchers (Boring, Joe, and Mandelli 2015; Boring et al. 2014; Coyne and Mosleh 2018) have emphasized the importance of simulation and human performance modeling in HRA. The HUNTER framework was developed to overcome some challenges with existing static HRA and to more realistically and accurately evaluate human-induced risks in NPPs. It was also conceived to offer a simple-to-use modeling approach that builds on well-established static HRA approaches while adding new dynamic modeling features. During the brief tenure of the HUNTER project, there have been several efforts to model varieties of human behaviors, produce an error rate over a denominator of repeated trials, dynamically compute performance shaping factor (PSF) levels to arrive at HEPs for any given point in time, and present the decision points that operators make while engaging with the plant.

The original HUNTER project was not intended to produce a standalone HRA method but rather a framework that combines a variety of methods and tools required for dynamic HRA. Figure 1 shows the

[c] While *dynamic* HRA is the commonly used term, *computational* HRA emphasizes that the modeling extends beyond the temporal dynamics (Joe et al. 2015). Time is only one dimension is the event progression, which may unfold spatially and across multiple systems. Simulation tools are used to model these phenomena.

original HUNTER framework, here called HUNTER 1. There are two important considerations in this early model of HUNTER. First, the HUNTER framework was designed to interact with other dynamic risk analysis tools like RAVEN (Rabiti et al. 2017). Previous HUNTER efforts focused on connecting HUNTER with RAVEN (Boring et al. 2016). As the HRA counterpart to RAVEN, HUNTER was used to quantify HEPs for operator actions in a station blackout scenario based on time-dependent plant response data and operator actions. Second, the existing HUNTER framework has considered three major concepts—cognitive models, PSFs, and data sources—for analyzing dynamic operator actions. In the previous HUNTER efforts, the Goals, Operators, Methods, and Selection rules (GOMS) - HRA (Boring and Rasmussen 2016), the Standardized Plant Analysis Risk-HRA (SPAR-H) autocalculation (Boring, Rasmussen, Smith, Mandelli, and Ewing 2017), and dynamic dependency (Boring 2015b) approaches were developed to implement the concepts within the HUNTER framework. These are described in the next subsections.

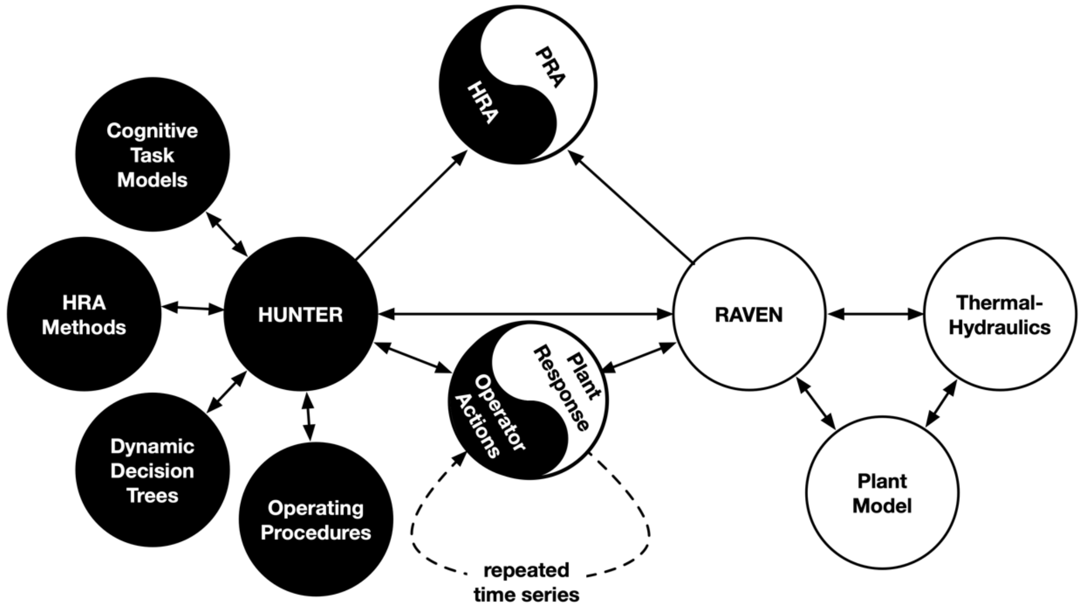


Figure 1. The original HUNTER framework (adapted from Boring et al. 2016).

## 2.2 GOMS-HRA

GOMS-HRA (Boring et al. 2016; Boring and Rasmussen 2016) was developed to provide cognition-based time and HEP information for dynamic HRA calculation in the HUNTER framework. It is theoretically derived from the GOMS method, which has been used to model proceduralized activities and evaluate user interactions with human-computer interfaces in human factors research (Card, Moran, and Newell 2018). As a predictive method, GOMS-HRA is well-equipped to simulate human actions under specific circumstances in a scenario. The basic approach of GOMS-HRA consists of three steps: (1) breaking human actions into a series of task-level primitives, (2) allocating time and error values to each task-level primitive, then (3) predicting human actions or task durations.

In GOMS-HRA, human actions are broken into task-level primitives, consisting of the most elemental types of human activities. GOMS-HRA uses six types of task-level primitives defined in the Systematic Human Error Reduction and Prediction Approach (SHERPA; Torres, Nadeau, and Landau 2021). The following are the SHERPA error types:

- *Actions* (A)—Performing required physical actions on the control boards ($A_C$) or in the field ($A_F$)
- *Checking* (C)—Looking for required information on the control boards ($C_C$) or in the field ($C_F$)
- *Retrieval* (R)—Obtaining required information on the control boards ($R_C$) or in the field ($R_F$)
- *Instruction Communication* (I)—Producing verbal or written instructions ($I_P$) or receiving verbal or written instructions ($I_R$)
- *Selection* (S)—Selecting or setting a value on the control boards ($S_C$) or in the field ($S_F$)
- *Decisions* (D)—Making a decision based on procedures ($D_P$) or without available procedures ($D_W$)

This GOMS-HRA taxonomy is captured in a cognitive model, as depicted in Figure 2, with an added element for time spent in waiting (W). This figure shows how tasks are aligned to stages of information processing, beginning with sensation and perception, progressing to cognition, and culminating in behavioral actions. Note that items like Instructions ($I_R$ and $I_P$) can be either verbal (e.g., communication between shift supervisor and reactor operator) or written (e.g., use of printed operating procedures).

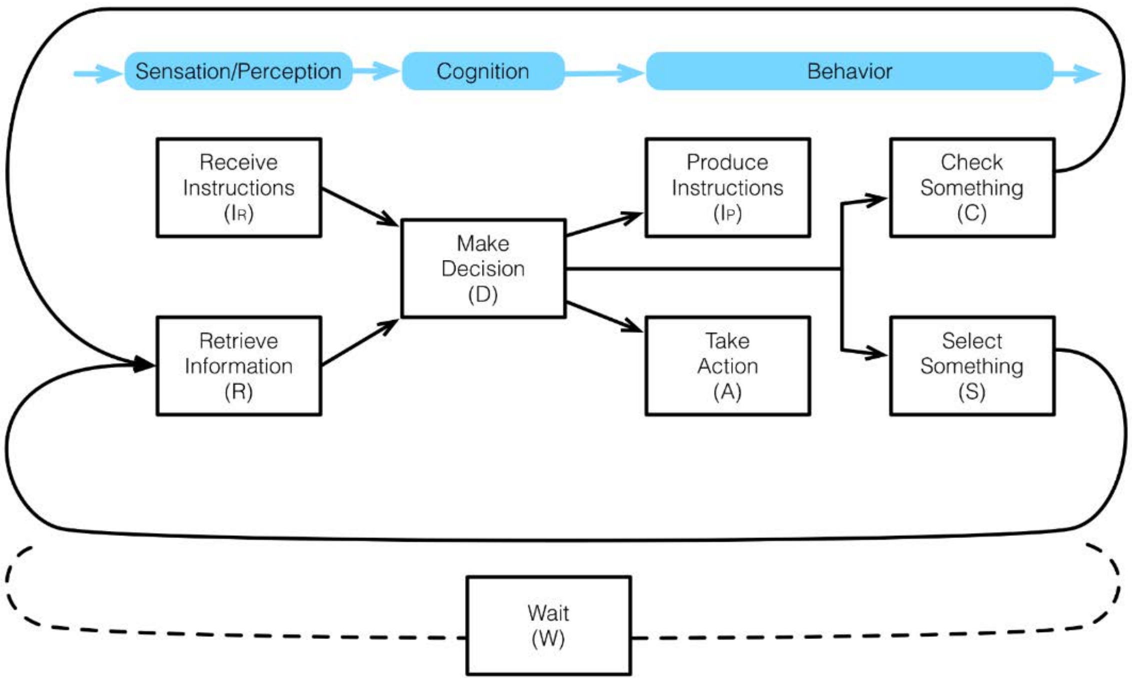


Figure 2. GOMS-HRA cognitive model (from Boring, Ulrich, and Rasmussen 2018).

Table 1. Time information for each task-level primitive (from Boring et al. 2016).

| **Task-Level Primitive** | **Distribution** | **Mean (log scale)** | **Standard Deviation (log scale)** | **5th Percentile** | **95th Percentile** |
|---|---|---|---|---|---|
| $A_C$ | Lognormal | 2.23 | 1.18 | 1.32 | 65.30 |
| $C_C$ | Lognormal | 2.14 | 0.76 | 2.44 | 29.90 |
| $D_P$ | Exponential | 0.02 | N/A | 2.62 | 152.80 |
| $I_P$ | Lognormal | 2.46 | 0.76 | 3.35 | 40.70 |
| $I_R$ | Lognormal | 1.92 | 0.93 | 1.47 | 31.80 |
| $R_C$ | Lognormal | 2.11 | 0.60 | 3.08 | 21.90 |
| $S_C$ | Lognormal | 2.93 | 1.11 | 3.01 | 115.60 |
| W | Lognormal | 2.66 | 1.26 | 1.79 | 113.60 |

The time and error values are allocated for task-level primitives of human actions analyzed in the first step. Table 1 and Table 2 summarize the time and HEP information for each task-level primitive. The time information includes the statistical distribution, mean, standard deviation, 5th and 95th percentile, which have been derived from the time data collected through experiments using actual operators in the Human Systems Simulation Laboratory (HSSL) at INL (Joe and Boring 2017; Ulrich et al. 2017a). For the HEP information, these are assumed based on data suggested in the Technique for Human Error Rate Prediction (THERP; Swain and Guttmann 1983) method.

Table 2. HEP information for each task-level primitive (from Boring and Rasmussen 2016; Boring, Ulrich, and Rasmussen 2018).

| Task-Level Primitive | Description | Nominal HEP | THERP Source | Notes |
|---|---|---|---|---|
| $A_C$ | Performing required physical actions on the control boards | 0.001 | 20-12 (3) | Assume well-delineated controls |
| $A_F$ | Performing required physical actions in the field | 0.008 | 20-13 (4) | Assume series of controls |
| $C_C$ | Looking for required information on the control boards | 0.001 | 20-9 (3) | Assume well-delineated indicators |
| $C_F$ | Looking for required information in the field | 0.01 | 20-14 (4) | Assume unclear indication |
| $R_C$ | Obtaining required information on the control boards | 0.001 | 20-9 (3) | Assume well-delineated indicators |
| $R_F$ | Obtaining required information in the field | 0.01 | 20-14 (4) | Assume unclear indication |
| $I_P$ | Producing verbal or written instructions | 0.003 | 20-5 (1) | Assume omit a step |
| $I_R$ | Receiving verbal or written instructions | 0.001 | 20-8 (1) | Assume recall one item |
| $S_C$ | Selecting or setting a value on the control boards | 0.001 | 20-12 (9) | Assume rotary style control |
| $S_F$ | Selecting or setting a value in the field | 0.008 | 20-13 (4) | Assume series of controls |
| $D_P$ | Making a decision based on procedures | 0.001 | 20-3 (4) | Assume 30-minute rule |
| $D_W$ | Making a decision without available procedures | 0.01 | 20-1 (4) | Assume 30-minute rule |

## 2.3 SPAR-H Autocalculation

The earlier HUNTER work investigated how to adapt the existing static SPAR-H to a dynamic framework. The SPAR-H Method (Gertman et al. 2005) is an easy-to-use HRA method developed by INL and published by the U.S. Nuclear Regulatory Commission (U.S. NRC). The approach focuses on the quantification of HEPs on the basis of PSF multipliers. SPAR-H has been widely used by both industry and regulators in its intended area of supporting PRAs for NPPs, but it is also finding use in other industries, such as oil and gas (Boring 2015a; Rasmussen, Standal, and Laumann 2015). In traditional static HRA approaches like SPAR-H, human actions are manually analyzed by human reliability analysts using tools like the Electric Power Research Institute's HRA Calculator (Julius et al. 2005). Specifically, for the HEP calculation, the analysts need to allocate a nominal HEP (i.e., a default error rate that serves as the starting value for HRA quantification) for a human failure event (HFE) or a smaller task-unit, rate a variety of PSF levels representing contextual impacts, and then modify the nominal HEP by applying the multiplier values for PSFs. In contrast, in the dynamic HRA version, the multiplier is calculated automatically without analyst inputs. In this case, the Complexity PSF multiplier is derived entirely from plant parameters. In HUNTER, GOMS-HRA provided the nominal HEPs. The details on the SPAR-H autocalculation approach are well described in Boring et al. (2017). The basic form of the equation for the Complexity PSF is found below:

$$\begin{aligned} \mathit{Normalized\ Complexity} &= 1.26754 \times LOOP + 1.26753 \times LODG + 1.26753 \times LOB \\ &- 0.00025 \times temperature - 0.00507 \times power + 1.65116 \end{aligned} \tag{1}$$

where LOOP represents a Boolean (i.e., true or false) variable for loss of offsite power, LODG represents a Boolean variable for loss of diesel generator, and LOB represents a Boolean variable for loss of battery. The temperature and power parameters represent plant parameters. The equation is generated dynamically in response to the evolving scenario and is normalized to the multiplier range found in the static form of SPAR-H.

## 2.4 Dynamic Dependency

Dependency analysis in HRA is a method of adjusting the failure probability of a given action by considering the impact of the action preceding it (Podofillini et al. 2010; Swain and Guttmann 1983). Normally, dependency increases the overall HEP, representing the noting that error begets error. Thus, dependency plays an important role in reasonably accounting for human actions in the context of PRAs and prevents PRA results from being estimated too optimistically based on the HRA results. Dependency analysis has been known to significantly affect the overall result of PRAs. If the results of dependency analyses are inaccurate, they could prove unconvincing for explaining the human failures in the context of PRA. In other words, risk metrics such as core damage frequency can be significantly underestimated in cut sets or sequences containing multiple HFEs if dependency is not considered.

One of the major benefits of transitioning from static to dynamic HRA is that dynamic HRA makes it possible to model operator actions over time as well as straightforwardly analyze dependencies between these actions. Existing static HRA methods mostly do not consider human performance changes over time or during the event progression, nor do they provide a truly dynamic account of human actions (Park, Boring, and Kim 2019). Accordingly, human reliability analysts have mostly performed dependency analysis by relying on static PRA and HRA information. Dynamic HRA, on the other hand, considers human actions dynamically and models types of activities and events, even where the human role is not clearly understood or predicted (i.e., unexampled events such as severe accidents). Furthermore, a dynamic simulation represents a sequence of operator actions, which make it easier to identify dependency candidates with contextual impacts. The authors previously considered how to treat dependency in dynamic HRA. Representatively, Boring (2015b) conceptually suggested PSF *lag* and *linger* effects as an option to treat dependence between operator actions. PSF lag indicates that the effect of the PSF on performance does not immediately psychologically or physically appear, while PSF linger means that the influence of PSFs on previous operator actions is unfinished after the actions, resulting in residual effects on the next operator actions. Park et al. (2019) and Park and Boring (2020) validated the effects on the basis of experimental data and applied the concept to the dynamic dependency analysis.

# 3. EXPANDED HUNTER FRAMEWORK

## 3.1 HUNTER Technology Readiness Level

As described in the previous section, the original HUNTER framework was a useful but disparate collection of solutions without a common software wrapper to tie it together. Aspects like PSF autocalculation were linked together for specific demonstrations, but the HUNTER framework remained primarily a research tool of specialized applications, and it did not present a coherent solution ready for industry application. This low level of software maturity was noted in a recent review by Choi (2020). RISA software tools, including HUNTER, are being evaluated according to their capability to support industry PRA. The criteria for evaluation are identified and discussed below.

Development maturity is captured specifically in terms of Technology Readiness Level (TRL; Government Accountability Office 2020). TRLs were originally developed and applied by the National Aeronautics and Space Administration but were later widely adopted by the U.S. Department of Defense and other agencies. TRLs depict how close to deployment a technology is, with higher numbers (up to TRL 9 on the scale) representing higher readiness for deployment. TRLs are depicted in Figure 3. A crosswalk between TRLs and RISA pathway goals is found in Table 3. TRLs are especially useful for gauging the maturity of research, which starts conceptually but may fail to reach deployment if not aligned to a systematic development process. High-value technologies should not languish at low TRLs, and the review of RISA tools identifies tools that would benefit industry through deployment and prioritizes this process. A goal of the RISA pathway is that tools should be usable for industry demonstrations, suggesting a TRL 7 or higher. Tools that fall below this TRL should be brought to a higher TRL. Of course, technology maturation is not an overnight process, and it is not necessarily possible to leapfrog multiple levels in a short time. Elevating TRLs serves as a goal to drive the systematic advancement of capabilities and maintain advancement momentum over the necessary development life cycle.

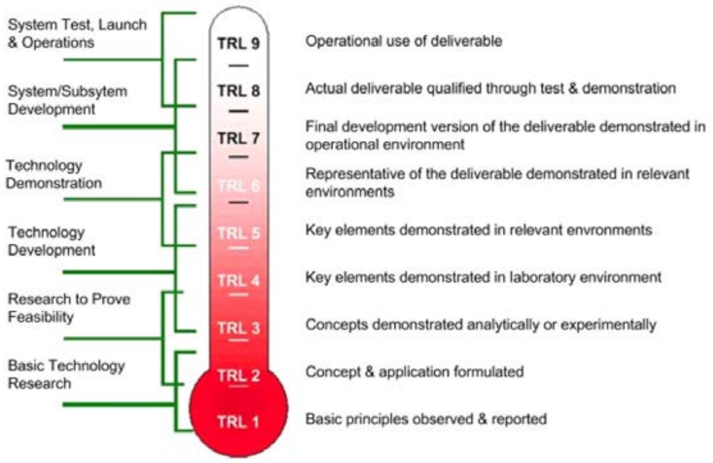

System Test, Launch & Operations
System/Subsytem Development
Technology Demonstration
Technology Development
Research to Prove Feasibility
Basic Technology Research
TRL 9
TRL 8
TRL 7
TRL 6
TRL 5
TRL 4
TRL 3
TRL 2
TRL 1
Operational use of deliverable
Actual deliverable qualified through test & demonstration
Final development version of the deliverable demonstrated in operational environment
Representative of the deliverable demonstrated in relevant environments
Key elements demonstrated in relevant envronments
Key elements demonstrated in laboratory environment
Concepts demonstrated analytically or experimentally
Concept & application formulated
Basic principles observed & reported

Figure 3. Technology readiness levels (from See and Handley 2019).

Table 4 summarizes the assessment of the original HUNTER. Only the first three criteria in the technology maturity assessment could be evaluated in terms of TRLs for HUNTER. These are:

- *Development level*—deemed of high importance but with a TRL of 3. The technical basis was in place, but the standalone software was not.
- *Use of proven technology*—deemed of high importance but with a TRL of 5. The HRA theories and methods underlying HUNTER were developed and ready for more complete demonstrations.
- *PRA capability and applicability*—deemed of high important and with a TRL of 7. HUNTER was designed for HRA and PRA use but needs further demonstrations and refinements.

The remaining criteria were not evaluated in terms of TRL, because information was not readily available. These included documentation, clear system requirements, easy installation, a graphical user interface, version control, verification and validation (V&V), quality assurance (QA), a tool web page, user support, a training program, and a software license. In the effort now to develop HUNTER as a more technologically mature software tool to support PRA and HRA, these criteria serve as requirements for future development.

Choi (2020) concluded the evaluation of HUNTER to date with the following concrete recommendations:

- Immediately develop necessary standalone software
- Review INL's software development guidance to prepare future quality assurance and higher credibility
- Reorganize and review development roadmap to build standalone software
- Develop coupling module for physics-based RISA toolkit (i.e., Reactor Excursion and Leak Analysis Program [RELAP5-3D])
- Create necessary manuals and supporting documents for industrial deployment.

The first task at hand to satisfy these requirements is to develop HUNTER into a software tool. This goal is the primary work covered in this report, with details of the software implementation found in Sections 4 – 7 and a demonstration analysis reviewed in Section 8. The HUNTER framework was expanded into a software tool capable of modeling dynamic HRA activities. This process has involved rethinking some aspects of HUNTER (explained in the next subsection) to allow it to support industry needs more readily. The effort documented in this report has developed the standalone software, including the coupling module to the physics-based toolkit. Ongoing development efforts will continue to refine the standalone software to higher TRLs. At the completion of this software development endeavor, future activities will begin to address the remaining requirements. User guidance will follow in the next step toward industrial deployment, including developing example dynamic HRA model linked to industry applications.

Table 3. Description of TRLs for RISA toolkit (from Choi [2021]).

| Level of Technology Development | Technology Readiness Level | TRL Definition | Application to RISA Toolkit |
|---|---|---|---|
| System Operation | TRL 9 | Actual system operated over the full range of expected conditions. | The toolkit technology is in its final form and routinely used under the full range of industrial purpose. |
| System Development | TRL 8 | Actual system completed and qualified through test and demonstration. | The toolkit has been proven to operate in its final form and under expected conditions. In almost all cases, this TRL represents the completion of R&D. Entire planned and proposed V&V of the toolkit is finalized by developer/user. |
| | TRL 7 | Full-scale, similar (prototypical) system demonstrated in relevant environment | Full-scale demonstration of actual/prototypical technology/toolkit in relevant operation environment. Full-scale experiment and validation are performed. Development of the toolkit is virtually complete. |
| Technology Demonstration | TRL 6 | Engineering/pilot-scale, similar (prototypical) system validation in relevant environment | Engineering-scale models or prototypes are tested in a relevant environment. Experiment and validation in engineering/pilot-scale environment includes scaling effect testing, which can support operational system, design. This level represents the completion of technology development for operational demonstration, and the prototype toolkit is ready for test. The prototype toolkit will have the capability of performing all the functions that will be required from an actual operational system. The operating environment for the testing should closely represent the actual operating environment. Major difference from TRL 5 is the scale-up from laboratory to engineering size. |
| Technology Development | TRL 5 | Laboratory scale, similar system validation in relevant environment | The basis of technology is fully integrated into the toolkit and ready for larger scale demonstration and validation. This level will include results from the laboratory-scale test, analysis of the differences between the laboratory and actual operating system/environment, and analysis of experiments/demonstrations results for actual operating system/ environment application. The major difference between TRL 4 and 5 is the increase of the toolkit fidelity and environment to the actual application. Verification is complete and the toolkit development level is close to prototypical. |
| | TRL 4 | Component and/or system validation in laboratory environment | The basis of technology for toolkit is partly integrated and can be applied for component level demonstration. This is relatively "low fidelity" compared with the actual level of the toolkit completion. The expected maturity of this level includes the integrated experiments and validation, examination of scaling effect and actual application. Verification and regression test could be included. TRL 4-6 represents the bridge from scientific research to engineering. TRL 4 is the first step in determining whether the basic modeling will work in the toolkit. |
| Research to Prove Feasibility | TRL 3 | Analytical and experimental critical function and/or characteristic proof of concept | Actual R&D is started for toolkit development. This includes analytical studies and laboratory-scale studies to validate the phenomena of separate technology. This level will have the results of laboratory tests performed to measure parameters of interest and comparison to analytical predictions for critical toolkit functions. At TRL 3, actual R&D progresses to experiments and verifications. Validation could be done for part of the toolkit development, but system level validation is not yet initiated. |
| Basic Technology Research | TRL 2 | Technology concept and/or application formulated | Progressed from TRL 1, technical options may be developed in TRL 2. However, still no activity was performed to prove assumptions and concept. Literature studies will outline the toolkit development concept. Most of the activity in this level is analytical research and paper studies to understand goal of the R&D. Related experiments and V&V works could be designed during this level. |
| | TRL 1 | Basic principles observed and reported | This is the lowest level of technology readiness. Scientific research begins to be translated into applied R&D. Available information includes published research or other references that identify the principles that underline the technology. No actual R&D started. |

Table 4. HUNTER framework technology maturity assessment result (from Choi [2020]).

| Requirements | Importance | Description | Technology Readiness Level (TRL) |
|---|---|---|---|
| Development level | High | Technical bases are well studied and developed. However, still needs to develop standalone software. | 3 |
| Use of proven technology | High | Mature theories are used as basis. No information available for code itself. | 5 |
| PRA capability/applicability | High | HUNTER framework is designed for PRA use. Need more demonstration. | 7 |
| Documentation | Medium | No information | N/A |
| System requirements | Low | No information | N/A |
| Easy installation | Medium | No information | N/A |
| Graphic user interface (GUI) | Medium | No information | N/A |
| Version control | Medium | No information | N/A |
| V&V activity/history | Medium | No information | N/A |
| QA program | High | No information | N/A |
| Web page | High | No information | N/A |
| User support | High | No information | N/A |
| Training program | Medium | No information | N/A |
| License | Medium | No information | N/A |

## 3.2 HUNTER's Adaptable Design Philosophy

As outlined in Section 2, the original HUNTER framework was a collection of dynamic HRA tools that were not contained in a single software application. Intrinsic to the earliest conceptualizations of HUNTER was the idea that some aspects of the modeling could be exchanged for different modules. For example, while the initial framework focused on making the SPAR-H PSFs dynamic (Boring et al. 2017), there was an acknowledgment that more comprehensive or nuanced PSF treatments should also be possible in HUNTER. In other words, while the initial proof-of-concept demonstration may focus on simplified parts of dynamic HRA, this simplification should not prove the limiting factor of HUNTER, and there should be opportunities to support more comprehensive modeling.

The details of how HUNTER should accomplish this shift from simplified models when convenient to detailed models when needed were not previously articulated. With the technology maturity assessment came the crucial realization that not all models that form HUNTER will have equally high TRLs. There may be advantages to having some models more at a research level and some ready for deployment.

Further, there is no one-size-fits-all implementation solution. Different analyses will have different requirements, which will require different models at different TRLs. Therefore, HUNTER must maintain this inherent consideration of adaptable instantiations for different modeling scenarios.

The process of translating HUNTER from a collection of research models into a standalone, integrated software tool necessitated the central goal of adaptability. As a result, HUNTER incorporates a flexible, modular, and scalable software architecture. This trio of concepts refer to the underlying objectives for HUNTER deployment. The three concepts overlap somewhat but are not fully interchangeable:

- *Flexible*—aligns with the ability of the HUNTER software to model a variety of applications. Most conventional HRA, for example, models reactor operator crews in main control rooms. This type of HRA is well understood and may not immediately benefit from the added functionality of dynamic HRA. However, the ability to create a virtual operator model that can be used for both main control room and balance-of-plant activities gives HUNTER the plasticity to model a diverse range of scenarios. Importantly, the value of HUNTER for industry may reside foremost in its ability to model emerging scenarios that are not well understood or for which there is no modeling precedence in conventional HRA.
- *Modular*—refers to the notion that parts of HUNTER can be interchanged. Modularity means that the part of the software code for modeling PSFs, for example, could be exchanged for another module. The PSF code, currently anchored in SPAR-H, could be switched for a different methodological treatment of PSFs. The emphasis in HUNTER becomes specifying how the module will communicate with the rest of the software, while providing fully functional default modules that can be used for the most common modeling applications.
- *Scalable*—means functions and features can be added on to the base software. For example, a cognitive modeling architecture might be added to the basic HUNTER model to influence decision outcomes during scenario runs. Scalability may mean that more complex modules may be used for certain analyses to increase modeling fidelity (often at the cost of modeling efficiency or outcome transparency). Scalability also means that some features may be excluded. For example, if particular modeling scenarios do not have information to drive some HUNTER features, these features may be toggled off when needed.

The adaptability objectives of flexibility, modularity, and scalability are influenced by a variety of modeling considerations. Most notably:

- *Knowledge*—our understanding of particular phenomena, specifically psychological aspects of operations in given contexts, will drive how modeling is deployed in HUNTER. Certain modeling approaches may be well validated (i.e., have higher TRLs), while other modeling approaches are more theoretical (i.e., are earlier in development and with lower TRLs). The analyst deploying HUNTER may opt for well-understood models for novel contexts to gain higher confidence in the results, or they may use less mature modeling for exploratory purposes to understand the range of possible phenomena rather than the most common course of action.
- *Fidelity*—the degree to which the modeling should accurately reflect human performance may shape how features are instantiated. For example, a severe accident modeling scenario may need to deploy a higher fidelity decision-making algorithm given the importance of operator expertise in navigating such contexts. This contrasts with more routine operations, which closely follow written procedures and may not require the same level of decision-making by operators. The former may require a sophisticated cognitive modeling architecture that can weigh goals and tradeoffs. In contrast, the latter may simply deploy a procedural script for the operator modeling component. Both should be possible in HUNTER (i.e.,

modeling flexibility), but they will affect which modules are selected (i.e., modeling modularity) and which features are invoked during simulation runs (i.e., modeling scalability).

- *Efficiency*—this consideration comprises how quickly the model may be set up and how quickly simulations may be run. Unless model building is automated by the PRA and other tools already at the analysts' disposal, the simulation model must be built for each scenario. The model development time is driven by constraints, such as the amount of time to complete the task, which is a direct reflection of the urgency of the analysis. For example, rapid-response modeling required after an incident may have a much shorter development timeline than a more routine version update of existing models over a multiyear timeline. This urgency may drive the need for a simpler model. On the other end of the equation, a simulation that is part of an extensive, multi-scenario analysis, such as in support of a whole plant PRA, may focus on the execution time of the model. HFEs in the PRA that are deemed of low risk significance may not warrant the luxury of waiting for a richly modeled scenario to complete. Instead, simple and quick modeling may suffice for such purposes.
- *Purpose*—specifies how the analysis will be used. While the purpose shapes some of the other modeling considerations, it is most useful as a concept to define the output of the analysis. For example, conventionally, HRA is used to calculate HEPs. Individual HEPs may then be substituted into an overall risk model to see the effect of human performance on the outcome of an event sequence. As noted in Boring et al. (2018), there remain some challenges with aligning dynamically calculated HEPs to those produced by static HRA methods. This stems from the unit of analysis, whereby most static HRA looks at the whole sequence of actions wrapped as an HFE, while dynamic HRA typically considers actions at the step or task level. The aggregation from step to HFE is not clearly understood, and calculated HEPs at the step or task must undergo some further conversion to achieve comparability with HEPs for HFEs. Further, there remain other outputs that may prove just as informative to HRA and have not been the purview of conventional HRA methods. For example, dynamic HRA can calculate time estimates for particular tasks. Often, the criterion for success or failure is not the overt commission of an error but the timing-out of an activity expected to be completed in a specific time window. With the exception of some early time reliability efforts like the time reliability estimation found in the Human Cognitive Reliability method (Parry et al. 1992), most HRA does not inform how long tasks involving human performers require. HRA may need this information as an input to the analysis, but most HRA methods do not provide explicit guidance to estimate time durations. The HUNTER framework can readily calculate probabilistic estimates of how long tasks take, thus providing a different type of output and purpose for risk analyses. Additionally, HUNTER has the ability to provide qualitative outputs, such as the state of dynamically calculated PSFs. Such analytic outputs could be informative to a hybrid static-dynamic HRA approach, for example, in which dynamic modeling is used to derive insights on operator performance that are subsequently used by human analysts to complete the HRA.

A crosswalk of objectives and modeling considerations for HUNTER may be found in Table 5.

Table 5. HUNTER objectives and modeling considerations.

| **OBJECTIVES** | **MODELING CONSIDERATIONS** | | | |
|---|---|---|---|---|
| | **Knowledge** | **Fidelity** | **Efficiency** | **Purpose** |
| **Flexibility**<br><br>**Modularity**<br><br>**Scalability** | Novel modeling scenarios mean less knowledge about performance outcomes. This may require generalizing known models, incorporating new modules that incorporate more known aspects of the modeling scenarios, or developing new features necessary to represent modeling nuances. | Some modeling contexts need less fidelity, while others—particularly risk-significant scenarios—may require more detailed fidelity. The model should adjust according to the demands for fidelity. Some modules may not be necessary for all contexts, while richer modules may be required for higher fidelity, and these modules may be turned on or off for particular analyses. | As with fidelity, some analyses may have different requirements. A dynamic HRA that's part of a larger PRA may need to emphasize computational efficiency, requiring simpler models. Modules may be optimized for speed with a reasonable approximation of operator performance, allowing quicker computation times when running the models. | The outputs of the dynamic modeling may vary—from autocalculated HEPs, to time required by human personnel, to the evolution of performance shaping factors. HRA may, in other words, be used for different purposes, and the software should accommodate these different purposes. |

## 3.3 HUNTER Conceptual Framework

### 3.3.1 Overview

With the inherent adaptability of HUNTER in mind, what are the essential modules of the HUNTER 2 framework? One critique of the trend to build increasingly complex models of human performance in HRA was leveraged by Galyean (2006). Galyean suggested that most human performance could be accounted for simply by looking at three factors:

- The individual
- The organization
- The environment.

This delineation of human activities was generally borne out in a review of Galyean by Boring (2010), with the conclusion that this general framework holds but may need more nuances for predicting the range of human actions. Only a very crude account of human performance is possible if these three factors alone are considered. While this three-factor model is useful, human performance may be understood with greater precision by considering a finer granularity of factors.

Inherent in Galyean's three-factor model is the idea of the individual and the context (i.e., organization and environment). This characterization may however fail to take proper account of the nature of the task the individual is performing, which provides an additional degree of context. This refinement of the three-factor model nears the model of constraints to action used in biomechanics (Newell, van Emmerik, and McDonald 1989). In that model, bodily coordination and control are influenced by individual, task, and environment factors. The focus of the model of constraints to action is clearly on physical movement, with constraints being individual physical capabilities of the organism, the nature of the movement task itself, and environmental influences that impinge or encourage that movement. Despite its focus on physical movement, the model readily generalizes to all human activities, including both physical actions and mental endeavors like decision-making. This basic model and its three factors as shown in Figure 4 serve as the software pillars for the new implementation of HUNTER, whereby each pillar serves as a module in the architecture. Joining the modules in the figure are classes, which are depicted in blue. For the present purposes, modules describe the basic elements of human behavior, while classes are the functions that enable the modules to work. Put another way, modules represent *who* (individual), *what* (task), and *where* (environment). Classes represent *how, why,* and *when* activities occur within the modules. Modules are the figurative nouns and verbs of HUNTER, while classes are the adjectives and adverbs.

This definition differs somewhat from the formal definition of module and class used in many software programming environments, but it nonetheless captures the fundamental structure of HUNTER. The modules and classes described in this section should be seen as describing the functional nature of the HUNTER framework, while the specific software implementation may consolidate or expand the specific modules and classes. The net effect remains the same: functionally and conceptually, the HUNTER framework consists at a high level of these basic elements. Further details on the actual software implementation follow in Sections 4 - 7. The functional modules and classes are described next.

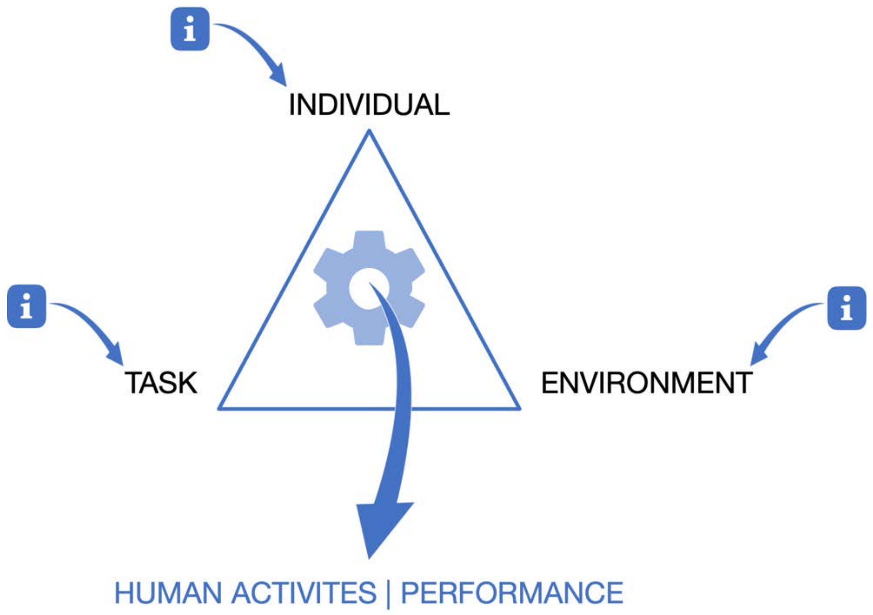


Figure 4. Conceptual modules (in black) and classes (in blue) of HUNTER 2.

### 3.3.2 HUNTER Modules

The three modules, depicted as corner nodes in black text in Figure 4, are briefly noted at a conceptual level here. We use the example of a virtual control room operator model for illustration here, but HUNTER is not limited to only this representation of plant personnel.

- *Individual Module*—this is the representation of the human performing the activity, sometimes referred to as a "virtual operator." It incorporates relevant characteristics of the individual that impact that individual's performance. Such factors could be considered internal PSFs, which are the psychological considerations—like internal stress, experience, knowledge, and fitness for duty—that the individual brings to the task. These factors may contribute directly to error rates (e.g., stress causes poor decision-making) or indirectly (e.g., performance is slowed when fatigued). The individual module may, when so configured, include a cognitive model that accounts for crucial aspects of performance like decision-making.

- *Task Module*—this is the representation of what activity the human is performing. The human follows a course of action, whether guided by an operating procedure, a mental schema, or decision-making according to emergent stimuli and strategic goals. In the simplest form of the task module, the task is represented by a script that mirrors procedures. The task advances step by step, responding to a set of if-then queries to plant states. For example, if a high-priority alarm sounds, the script will direct a specific response by the virtual operator. In a simple model, the operator's ability to perform that task may be influenced by factors contained in the individual and environment modules, but the operator does not deviate from the script. Of course, actual reactor operators are not merely automata, and they will weigh in on the suitability of scripts and even improvise when appropriate. A richer model of the task would include provisions for skill of the craft and acting outside of rote script following. A yet richer model would incorporate tradeoffs and decision-making, including decision heuristics indicative of operator expertise.

- *Environment Module*—this is the representation of the world in which the human is acting. In this sense, the "world" consists of the systems and tools the human uses. It is the virtual world counterpart to the virtual operator represented in the individual module. For most NPP modeling, this world model corresponds to a plant simulation. The environment may often only encompass the immediate environment and not necessarily consider the broader environment such as the natural setting of the plant if that is not central to the task at hand. Level 1, 2, and 3 HRAs correspond to modeling scenarios involving design-basis plant functioning, plant damage, and impacts beyond the plant, respectively (St Germain et al. 2016). The level of the risk modeling determines whether the environment is modeled at the micro-, meso-, or macrolevel. The environment module considers the external PSFs like the availability of procedures, the quality of the HMI, and the complexity and difficulty of the plant conditions. These may be derived from plant parameters provided by the plant simulation (e.g., Boring et al. 2017).

### 3.3.3 HUNTER Classes

There are four classes of the HUNTER framework illustrated in blue in Figure 4. They are:

- *Input Class*—the context is set by the scenario at hand. This is shown in Figure 4 as an input (i.e., i) into each of the modules, representing the influences that feed into the scenario. A preprocessor sets the context—the initial configuration for the individual, the task at hand, and the state of the plant—in which HUNTER operates.

- *Scheduler Class*—the glue that holds the other modules together. In the figure, this is signified by the lines of the triangle. It coordinates the interactions between different modules and also paces the progression of the event. Modules may complete their calculations at different rates, and the scheduler synchronizes the inputs, outputs, and interdependencies to a common time scale.

- *Processor Class*—the processing that occurs at each step of the task, which is depicted by a gear in the center of the Figure 4. A step occurs when all modules have completed their modeling refresh cycle and exchanged information. For example, the environment has advanced a time step, updating plant parameters, which have been perceived by the virtual operator (individual), who has responded by activating a virtual switch (task). This task may be driven by a procedure, which must meet certain requirements to advance. The processor class determines the point of advancement to the next task. The processor may include logical assertions, such as actions predicated on conditions met, branching points, and operator decisions.

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

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

### 3.3.4 Special Considerations

This conceptual representation is necessarily simplified, and it should be noted that the modules may employ additional classes and supporting tools to accomplish their functionality. For example, if the environment module is a full-scope simulator, it needs a software binder or advanced programming interface (API) to allow communication between the simulator and HUNTER. This API may be quite different between simulators, but the basic conceptual function remains the same, namely to facilitate the exchange of information between the environment module and other entities in the HUNTER software. Alternately, the API may consist of lookup tables of prescripted runs, inputs from physical test loops, or even dummy values, depending on the needs of the risk model.

The adaptability aspects of HUNTER outlined in Section 3.2 mean that the specific software implementation for each module or class can be changed depending on the modeling requirements. The processor class, for example, may have hooks for procedures and decision-making. The default configuration deployed at this time does not yet incorporate a decision-making subclass. As such, this function is simply turned off in the software, and modeling assumes rote procedure following. The HUNTER architecture allows a subclass to be linked and activated as it becomes available and is needed by the modeling community. Similarly, HUNTER uses a simplified list of PSFs for proof of concept. This does not prevent a more comprehensive model of PSFs to be inserted as a subclass when one is developed. This concept of adaptability from simple to complex modeling in HUNTER is accomplished through turning functions on or off and by allowing the capability to link to more complex modeling tools as needed.

One of the primary advantages of dynamic HRA comes from the ability to consider the range of outcomes and trajectories that are possible—something that is difficult and extremely time-consuming to be performed manually using current static HRA and PRA tools. The range of outcomes is accomplished by the ability to run each modeled scenario multiple times, covering both the bounds of expected human performance (i.e., from worst to best performance, and everything in between) and the addition of uncertainty to the model. Model runs such as Markov Chain Monte Carlo iterations are guided by a combination of the classes. The scheduler class may track not just individual tasks within a model run but also overall repeats of model runs. The input class may change conditions slightly (e.g., varying the effects of certain PSFs) at the restart of each run. The processor class may direct activities along different branch points to see consequences of different simulated operator actions. Finally, the output class may log the relevant results from each model run and aggregate them in a meaningful way for understanding trends, frequencies of particular operational paths, and significant outcomes.

### 3.3.5 Relationship Between HUNTER 1 and HUNTER 2 Frameworks

Superficially, Figure 4 for HUNTER 2 may seem much more abstracted than Figure 1 for HUNTER 1. In fact, the two approaches are not that different. The framework shown for HUNTER 1 is actually a specific software architecture, while the framework for HUNTER 2 abstracts out to become a more general conceptual model. This change reflects the greater emphasis on adaptability in HUNTER 2 and the corresponding desire not to lock down the implementations for the modules and classes.

Figure 5 shows the original HUNTER framework from Figure 1 superimposed with the more generic modules and classes from HUNTER 2. As can be seen, there is a direct mapping of some elements. The modules, as would be expected, are comprised of multiple sub-elements, while the classes link these modules functionally. What's clearly missing from the original HUNTER framework is a way to account for inputs like setting the initial configuration of the model. The original HUNTER framework in Figure 1 represents more of an architectural snapshot. As such, the need to reflect the states of the model was not depicted but was implicitly accounted for in the model.

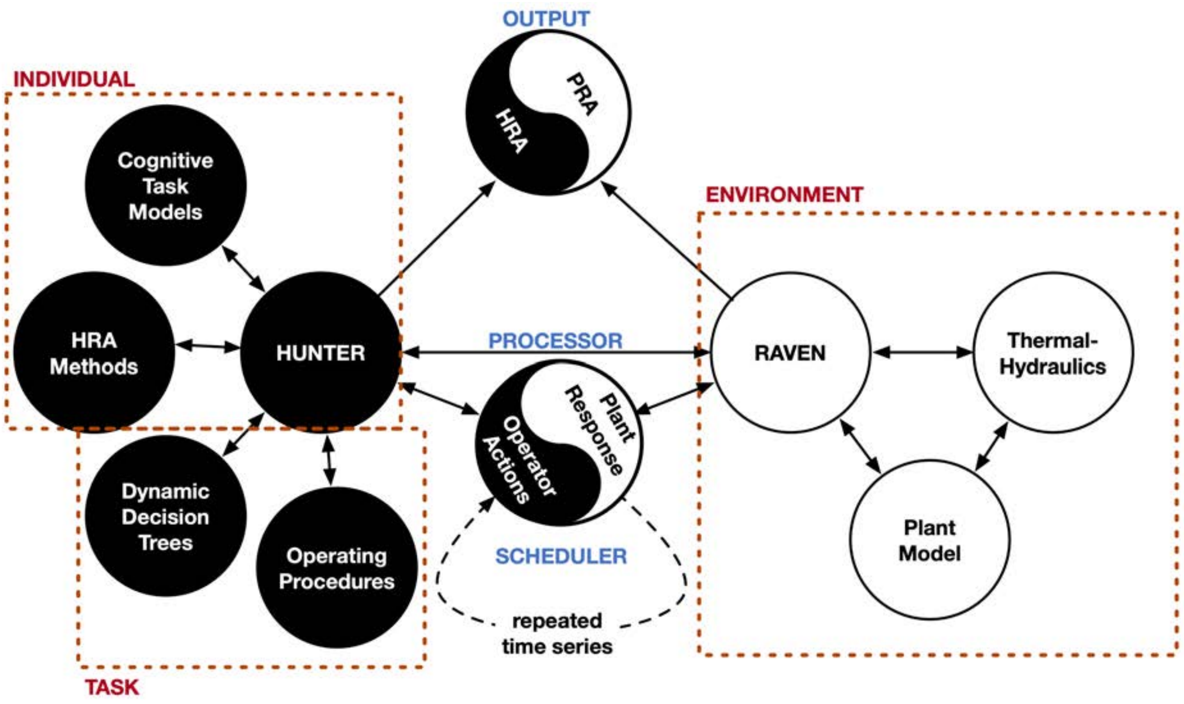


Figure 5. Crosswalk of HUNTER 1 to HUNTER 2.

HUNTER 2, as shown in the figure and documented in this report, is both an extension and generalization of the original HUNTER framework. The original HUNTER figure remains a useful sketch of a software implementation of HUNTER, while this section has cast a more conceptual framework. The next sections summarize the implementation of this framework as standalone software.

# 4. TASK MODULE IMPLEMENTATION

## 4.1 Background

The HUNTER framework was translated into a simulation application written in the Python programming language. The simulation code supports the ability to execute scenarios comprised of a set of procedures. The procedures are predefined as inputs to the application and contain all the necessary data elements to execute the human tasks based on the simulated nuclear plant state, evaluate the virtual operator human reliability context, and proceed down the procedure path appropriately. In the interest of reader understanding, an explanatory description of the HUNTER software implementation begins with the data input structure and format for the procedure files, since the simulation application is structured around these data as an input.

## 4.2 Input Data Structure and Analyst Workflow

The simulation requires a user-defined comma-separated value (CSV) file for each of the procedures used for simulation runs. The CSV format was intentionally selected to provide a non-proprietary data structure that is widely compatible across software tools. This format shares some similarity to input deck files for other codes such as RELAP5-3D (Aumiller, Tomlinson, and Bauer 2001). In the current application version, the analyst must manually enter procedures, procedures steps, and substeps into the CSV file, although a graphical skin to simplify this process is described in Section 7. Future iterations of the code are planned to automatically parse procedures to greatly simplify this process. The analyst must also populate details for the human reliability context and the virtual plant context for each procedure. Specifically, the analyst can prescriptively drive the procedure path by specifying plant parameters needed to support procedure logic at branch points. The code also supports defining multiple possible paths toward different outcomes. With the open-ended procedure construction, the analyst must select a supporting simulator and define specific parameters associated with each step that are evaluated at run time to determine the success or failure of a given step. Where external simulation codes are not available, information may be dummy coded to support HUNTER's progression through the procedures. In both the open-ended and dummy mode of procedure execution, the analyst can also define the human reliability parameters for each step to add the HRA layer onto the procedure execution. In this simulation mode, the procedure uses the provided context information to calculate an HEP for each step, which yields a success or failure outcome and can change the course of the simulated procedure path.

Each row of the input deck represents a procedure step-level item and must have a step number. This step number is used to group any additional related rows (e.g., all rows tagged as 12 are processed by the parser as a collection of sub-elements to be grouped within Step 12). As the number of items contained within a procedure step is variable, adding rows supports the ability to add as many elements as needed without complex reformatting of the input CSV file.

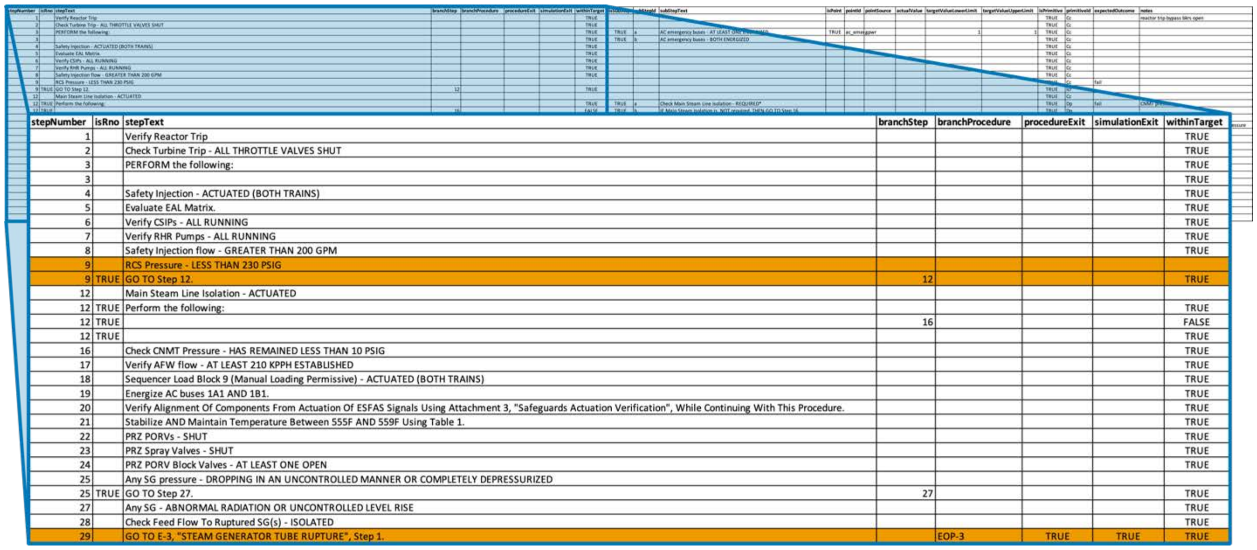
There is an intermediary process that occurs when launching the HUNTER application. A parser class, which is a subclass of the scheduler module, converts the CSV input file into a JavaScript Object Notation (JSON) format that is then directly consumed by the scheduler class. This parser class allows the analyst to work with the more easily human read and edited CSV file, while the JSON format is much easier for the application to consume. The CSV file can be edited and viewed as a spreadsheet table, meaning it has consistent fields mapped across a matrix. In contrast, a JSON file does not populate cells without information, and only fields that are actively changed are conveyed in the file. Thus, the structure of a JSON files is much less linear in appearance.

The parser file relies on dummy coding to process each row, or subsequent rows, as a part of a single procedure step. The input file can be divided into the four nesting levels of step, substep, point, and primitive as depicted in Figure 6 – Figure 9, respectively.

- *Step*—refers to the main step activities. In the common format used in procedures for Westinghouse pressurized-water reactors, there are two columns. The primary tasking occurs in the left column, meaning the task that is initially completed for that procedure step. There are also sometimes procedure steps in the right column, called the Response Not Obtained (RNO) column. If the tasking in the left column cannot be completed, the operator transfers to the RNO column to complete that tasking.
- *Substep*—refers to the secondary tasking that is performed within a main procedure step. Often a procedure step requires multiple substeps by operators to complete the desired tasking. While

steps are numbered, substeps are usually treated alphabetically, Step 1a, 1b, etc., meaning main Step 1 followed by substeps *a* and *b*.

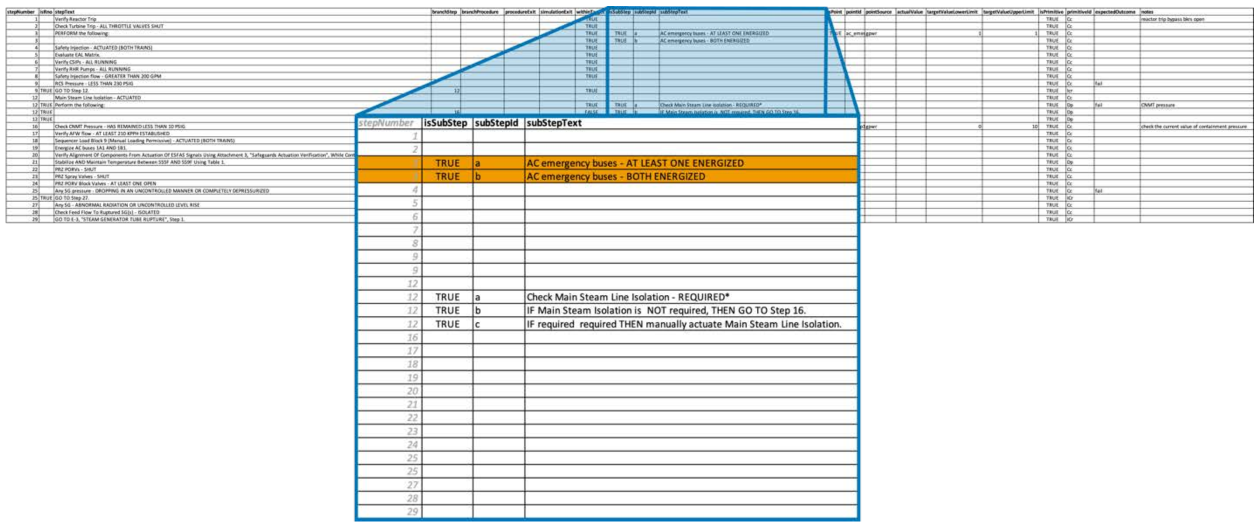
- *Point*—refers to addresses or names for parameters in external plant simulations. This information may also be dummy coded if no plant simulation is referenced.
- *Primitive*—refers to the GOMS-HRA task level primitives, which the HUNTER code uses to determine time durations and nominal HEPs for tasks. More than one task level primitive may be associated with a step.



| stepNumber | isRno | stepText | branchStep | branchProcedure | procedureExit | simulationExit | withinTarget |
|---|---|---|---|---|---|---|---|
| 1 | | Verify Reactor Trip | | | | | TRUE |
| 2 | | Check Turbine Trip - ALL THROTTLE VALVES SHUT | | | | | TRUE |
| 3 | | PERFORM the following: | | | | | TRUE |
| 3 | | | | | | | TRUE |
| 4 | | Safety Injection - ACTUATED (BOTH TRAINS) | | | | | TRUE |
| 5 | | Evaluate EAL Matrix. | | | | | TRUE |
| 6 | | Verify CSIPs - ALL RUNNING | | | | | TRUE |
| 7 | | Verify RHR Pumps - ALL RUNNING | | | | | TRUE |
| 8 | | Safety Injection flow - GREATER THAN 200 GPM | | | | | TRUE |
| 9 | | RCS Pressure - LESS THAN 230 PSIG | | | | | |
| 9 | TRUE | GO TO Step 12. | 12 | | | | TRUE |
| 12 | | Main Steam Line Isolation - ACTUATED | | | | | |
| 12 | TRUE | Perform the following: | | | | | TRUE |
| 12 | TRUE | | 16 | | | | FALSE |
| 12 | TRUE | | | | | | TRUE |
| 16 | | Check CNMT Pressure - HAS REMAINED LESS THAN 10 PSIG | | | | | TRUE |
| 17 | | Verify AFW flow - AT LEAST 210 KPPH ESTABLISHED | | | | | TRUE |
| 18 | | Sequencer Load Block 9 (Manual Loading Permissive) - ACTUATED (BOTH TRAINS) | | | | | TRUE |
| 19 | | Energize AC buses 1A1 AND 1B1. | | | | | TRUE |
| 20 | | Verify Alignment Of Components From Actuation Of ESFAS Signals Using Attachment 3, "Safeguards Actuation Verification", While Continuing With This Procedure. | | | | | TRUE |
| 21 | | Stabilize AND Maintain Temperature Between 555F AND 559F Using Table 1. | | | | | TRUE |
| 22 | | PRZ PORVs - SHUT | | | | | TRUE |
| 23 | | PRZ Spray Valves - SHUT | | | | | TRUE |
| 24 | | PRZ PORV Block Valves - AT LEAST ONE OPEN | | | | | TRUE |
| 25 | | Any SG pressure - DROPPING IN AN UNCONTROLLED MANNER OR COMPLETELY DEPRESSURIZED | | | | | |
| 25 | TRUE | GO TO Step 27. | 27 | | | | TRUE |
| 27 | | Any SG - ABNORMAL RADIATION OR UNCONTROLLED LEVEL RISE | | | | | TRUE |
| 28 | | Check Feed Flow To Ruptured SG(s) - ISOLATED | | | | | TRUE |
| 29 | | GO TO E-3, "STEAM GENERATOR TUBE RUPTURE", Step 1. | | EOP-3 | TRUE | TRUE | TRUE |

Note: Step 9 is highlighted to show the dummy coding used in the parser flag column to denote response obtained and RNO step types.

Figure 6. Region of the HUNTER input file demonstrating the use of dummy coding to represent additional elements under the same procedure step.



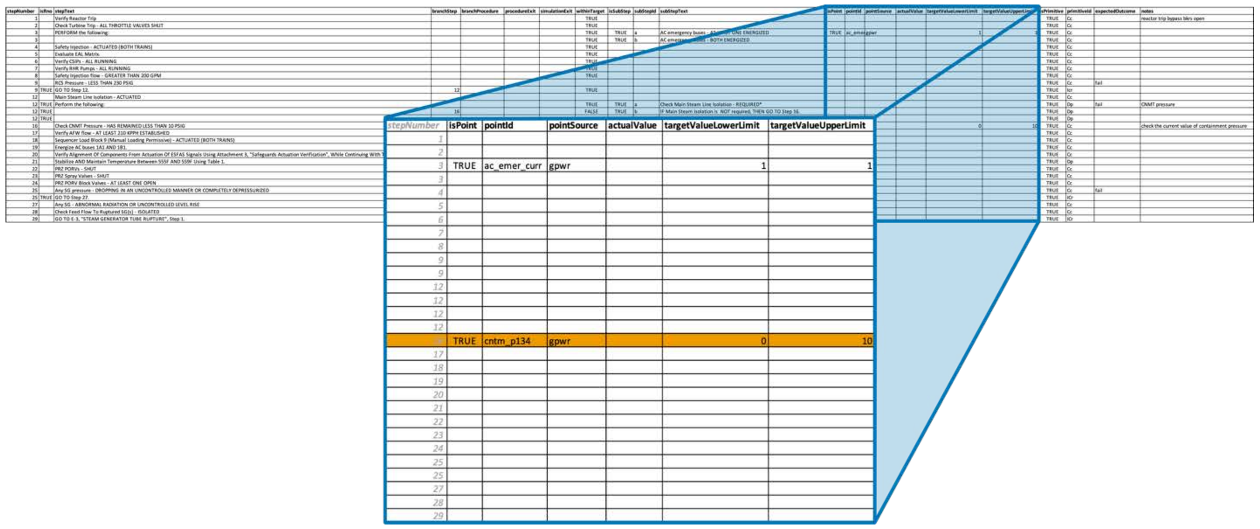
| stepNumber | isRno | stepText | branchStep | isPrimitive | primitiveId | expectedOutcome | notes |
|---|---|---|---|---|---|---|---|
| 1 | | Verify Reactor Trip | | TRUE | Cc | | reactor trip bypass bkrs open |
| 2 | | Check Turbine Trip - ALL THROTTLE VALVES SHUT | | TRUE | Cc | | |
| 3 | | PERFORM the following: | | TRUE | Cc | | |
| 3 | | | | TRUE | Cc | | |
| 4 | | Safety Injection - ACTUATED (BOTH TRAINS) | | TRUE | Cc | | |
| 5 | | Evaluate EAL Matrix. | | TRUE | Cc | | |
| 6 | | Verify CSIPs - ALL RUNNING | | TRUE | Cc | | |
| 7 | | Verify RHR Pumps - ALL RUNNING | | TRUE | Cc | | |
| 8 | | Safety Injection flow - GREATER THAN 200 GPM | | TRUE | Cc | | |
| 9 | | RCS Pressure - LESS THAN 230 PSIG | | TRUE | Cc | fail | |
| 9 | TRUE | GO TO Step 12. | 12 | TRUE | Icr | | |
| 12 | | Main Steam Line Isolation - ACTUATED | | TRUE | Cc | | |
| 12 | TRUE | Perform the following: | | TRUE | Dp | fail | CNMT pressure |
| 12 | TRUE | | 16 | TRUE | Dp | | |
| 12 | TRUE | | | TRUE | Dp | | |
| 16 | | Check CNMT Pressure - HAS REMAINED LESS THAN 10 PSIG | | TRUE | Cc | | check the current value of containment pressure |
| 17 | | Verify AFW flow - AT LEAST 210 KPPH ESTABLISHED | | TRUE | Cc | | |
| 18 | | Sequencer Load Block 9 (Manual Loading Permissive) - ACTUATED (BOTH TRAINS) | | TRUE | Cc | | |
| 19 | | Energize AC buses 1A1 AND 1B1. | | TRUE | Cc | | |
| 20 | | Verify Alignment Of Components From Actuation Of ESFAS Signals Using Attachment 3, "Safeguards Actuation Verification", While Cont | | TRUE | Cc | | |
| 21 | | Stabilize AND Maintain Temperature Between 555F AND 559F Using Table 1. | | TRUE | Dp | | |
| 22 | | PRZ PORVs - SHUT | | TRUE | Cc | | |
| 23 | | PRZ Spray Valves - SHUT | | TRUE | Cc | | |
| 24 | | PRZ PORV Block Valves - AT LEAST ONE OPEN | | TRUE | Cc | | |
| 25 | | Any SG pressure - DROPPING IN AN UNCONTROLLED MANNER OR COMPLETELY DEPRESSURIZED | | TRUE | Cc | fail | |
| 25 | TRUE | GO TO Step 27. | | TRUE | ICr | | |
| 27 | | Any SG - ABNORMAL RADIATION OR UNCONTROLLED LEVEL RISE | | TRUE | Cc | | |
| 28 | | Check Feed Flow To Ruptured SG(s) - ISOLATED | | TRUE | Cc | | |
| 29 | | GO TO E-3, "STEAM GENERATOR TUBE RUPTURE", Step 1. | | TRUE | ICr | | |

| stepNumber | isSubStep | subStepId | subStepText |
|---|---|---|---|
| 1 | | | |
| 2 | | | |
| 3 | TRUE | a | AC emergency buses - AT LEAST ONE ENERGIZED |
| 3 | TRUE | b | AC emergency buses - BOTH ENERGIZED |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |
| 9 | | | |
| 9 | | | |
| 12 | | | |
| 12 | TRUE | a | Check Main Steam Line Isolation - REQUIRED* |
| 12 | TRUE | b | IF Main Steam Isolation is NOT required, THEN GO TO Step 16. |
| 12 | TRUE | c | IF required required THEN manually actuate Main Steam Line Isolation. |
| 16 | | | |
| 17 | | | |
| 18 | | | |
| 19 | | | |
| 20 | | | |
| 21 | | | |
| 22 | | | |
| 23 | | | |
| 24 | | | |
| 25 | | | |
| 25 | | | |
| 27 | | | |
| 28 | | | |
| 29 | | | |

Note: Dummy coding used in the isSubStep parser flag column and the additional subStepId populated with the procedure substep identifier.

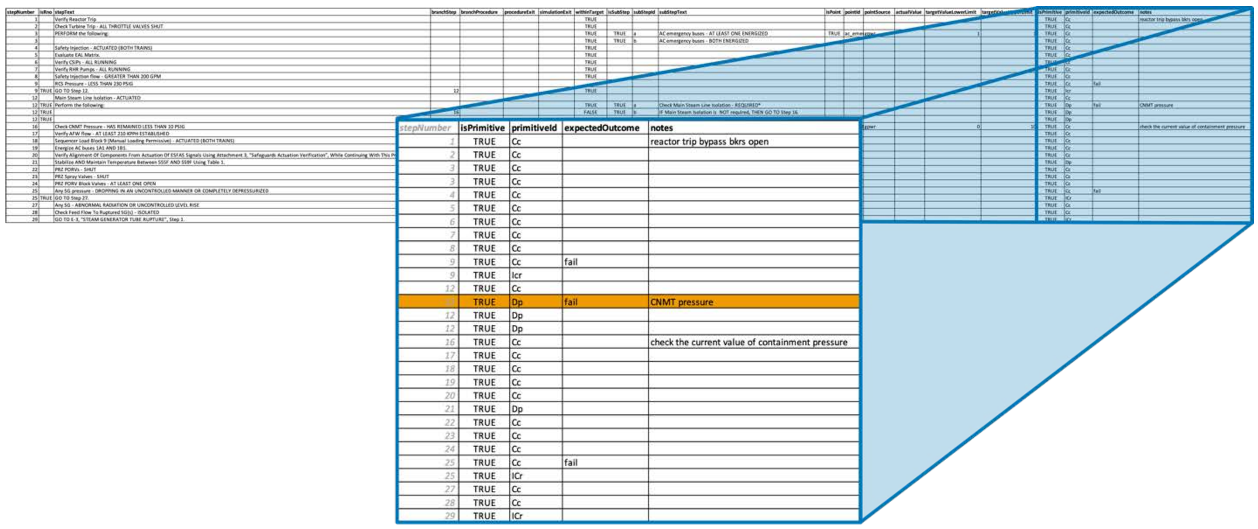
Figure 7. Procedure substep region of the HUNTER input file.

To clarify the above, a brief description of nuclear operating procedures helps to understand the representation of the procedures in the simulation. There are different types of procedures used in an NPP, but for the modeled scenario, the procedures are emergency operating procedures (EOPs) and abnormal operating procedures (AOPs). These types of procedures follow a two-column format in which each step is represented twice in the procedure, once in the left response obtained and once in the right RNO column. If the logic of the step described in the left column, referred to as the response obtained, is upheld, the operator moves to the next numerical step of the procedure. If the logic is not upheld, the operator moves to the right column, RNO, for that step. Furthermore, numerical procedure steps often include substeps denoted with letters (i.e., *a*, *b*).



| stepNumber | isPoint | pointId | pointSource | actualValue | targetValueLowerLimit | targetValueUpperLimit |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | TRUE | ac_emer_curr | gpwr | | 1 | 1 |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 9 | | | | | | |
| 12 | | | | | | |
| 12 | | | | | | |
| 12 | | | | | | |
| 12 | | | | | | |
| 16 | TRUE | cntm_p134 | gpwr | | 0 | 10 |
| 17 | | | | | | |
| 18 | | | | | | |
| 19 | | | | | | |
| 20 | | | | | | |
| 21 | | | | | | |
| 22 | | | | | | |
| 23 | | | | | | |
| 24 | | | | | | |
| 25 | | | | | | |
| 25 | | | | | | |
| 27 | | | | | | |
| 28 | | | | | | |
| 29 | | | | | | |

Figure 8. Point region of the HUNTER input file denoting names of plant parameters.



| stepNumber | isPrimitive | primitiveId | expectedOutcome | notes |
|---|---|---|---|---|
| 1 | TRUE | Cc | | reactor trip bypass bkrs open |
| 2 | TRUE | Cc | | |
| 3 | TRUE | Cc | | |
| 3 | TRUE | Cc | | |
| 4 | TRUE | Cc | | |
| 5 | TRUE | Cc | | |
| 6 | TRUE | Cc | | |
| 7 | TRUE | Cc | | |
| 8 | TRUE | Cc | | |
| 9 | TRUE | Cc | fail | |
| 9 | TRUE | Icr | | |
| 12 | TRUE | Cc | | |
| [illegible] | TRUE | Dp | fail | CNMT pressure |
| 12 | TRUE | Dp | | |
| 12 | TRUE | Dp | | |
| 16 | TRUE | Cc | | check the current value of containment pressure |
| 17 | TRUE | Cc | | |
| 18 | TRUE | Cc | | |
| 19 | TRUE | Cc | | |
| 20 | TRUE | Cc | | |
| 21 | TRUE | Dp | | |
| 22 | TRUE | Cc | | |
| 23 | TRUE | Cc | | |
| 24 | TRUE | Cc | | |
| 25 | TRUE | Cc | fail | |
| 25 | TRUE | ICr | | |
| 27 | TRUE | Cc | | |
| 28 | TRUE | Cc | | |
| 29 | TRUE | ICr | | |

Figure 9. GOMS-HRA task-level primitives found in HUNTER input file.

Populating any row of these four regions requires adding TRUE to the corresponding parser flag columns, which are labeled isRno, isSubStep, isPoint, and isPrimitive, based on their associated element type. For example, Figure 6 above shows the left region of the CSV input file for a procedure with the leftmost column indexing each step as a numerical number. The next column, isRno, is a parser flag column used to denote if the step is a response obtained or an RNO type of step. Step 9 in the figure, which is highlighted in orange, demonstrates the dummy coding for a step with a response obtained. The second row of Step 9 shows an RNO step. The second nesting region, the substep region shown in Figure 7, follows exactly the same organization but for substeps. Step 3 in the figure shows substeps *a* and *b* appearing as two sequential rows. Similar to adding an RNO element, adding substeps requires adding rows and tagging those rows with the step number. It is possible to combine RNO and substeps.

The third and fourth nesting regions, shown as Figure 8 and Figure 9, depict the point and primitive fields, respectively. These elements can be nested at the step or substep level. Therefore, any element added to this row must ensure that the isPoint and isPrimitive parser flag columns are set in addition to the isSubStep column if the point or primitive should be nested within a substep item.

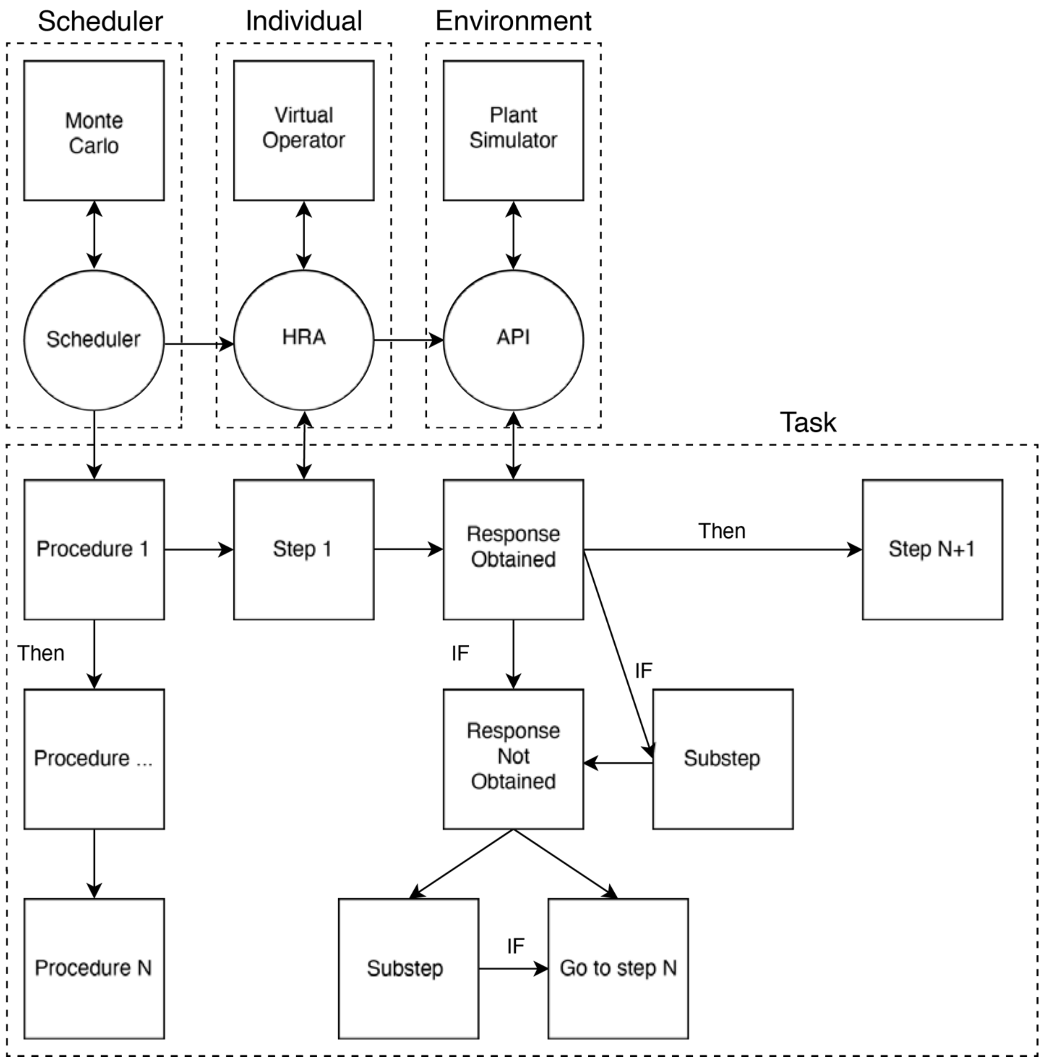


Figure 10. Mapping of the individual, task, and environment modules to the classes in the HUNTER software implementation.

## 4.3 Modules

As noted in earlier sections, the code is organized as a Python package with modules. Each module focuses on performing a subset of the functionality contained within the HUNTER framework. This section describes the closely linked Scheduler class and Task module. Note that, as previously described, adaptability can come from switching out modules or creating APIs for the modules to interface with external software. The modules and their interactions are illustrated in Figure 10.

### 4.3.1 Scheduler Class

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

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

### 4.3.2 Task Module

The Task module contains the classes that store and manipulate the activity executed during each simulation run. The Task module contains the procedures with their steps, following the two-column procedure format described in Section 4.2. While comparing the plant state against the logic in the left response obtained column, if at any time the logic for the substeps is not upheld, the operator moves to the right column and starts executing the numerical step for RNO from the beginning of the step, even if there are substeps listed under the main step.

#### *4.3.2.1 Step Class*

The step class is the top level object for each procedure step. It is a superclass of the assertion class that inherits the same properties but adds additional parameters. It stores the procedure step number and can store a response obtained and RNO typed assertion. At the least, it must contain a response obtained assertion, but it may also, and typically does, store an RNO assertion as well. The Task module executes the response obtained assertion and, if that is successful, it transitions to the next numerical procedure step. If the response obtained step fails, the RNO assertion is evaluated. The assertion class is defined in the following section and represents the core of the simulation.

#### *4.3.2.2 Assertion Class*

The assertion class (see Figure 11) is the central element of the entire simulation because it encapsulates each instance of the dynamic simulation that is evaluated. The assertion class is generically defined to represent a procedure step in the response obtained left column, RNO right column, or substep. The assertion class holds all the relevant simulation parameters for procedure logic evaluation at any given timepoint. Each assertion contains two types of subclasses to store information pertinent to the human reliability parameters and the plant parameters.

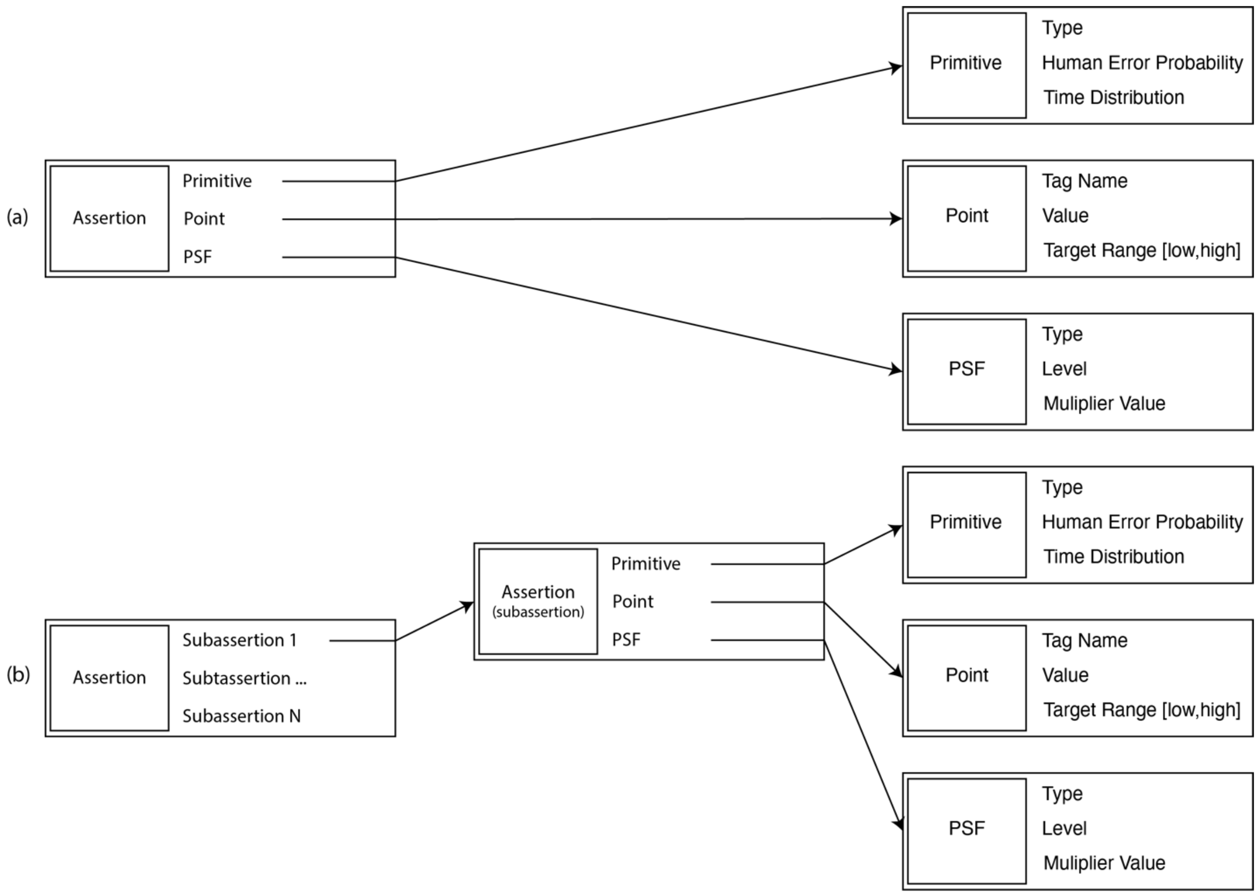


Figure 11. Assertion class in HUNTER acting (a) directly as a procedure step and (b) indirectly to trigger procedure substeps.

Several subclasses store the human reliability parameters within the assertion class.

- The GOMS-HRA task-level primitives are defined in the primitive subclass. Analysts can assign multiple primitives to each assertion if necessary, but in practice there should never be more than two based on our preliminary usage exploration. Ideally the analyst should attempt to define a single primitive for each assertion, but since an assertion can represent a step or substep, more complicated steps that do not contain any substeps may require two primitives to capture the intended tasks. An analyst simply assigns a primitive identifier, such as $C_C$ to denote the "check in the control room" task-level primitive. Each task-level primitive type has a predefined execution time distribution and HEP associated with it.
- The analyst can account for contexts surrounding the virtual operator with the PSF subclass. The analyst selects relevant PSFs from SPAR-H and assigns a default level, which has a corresponding multiplier that will be applied on top of the nominal HEP linked to the GOMS-HRA task-level for that assertion. Some PSFs may also be set to autocalculate, as covered in Chapter 5 on the Individual module. The analyst can also switch from the GOMS-HRA nominal HEP to use the nominal HEPs for diagnosis and action found in SPAR-H. However, in its current version, the GOMS-HRA task-level primitive must still be populated by the analyst to provide the timing information for the assertion. The nominal HEPs may be modified through the simulation run by autocalculated HEPs.

- The assertion class also contains NPP simulator parameters. Each assertion can hold multiple plant parameter objects defined by the point subclass. Each point object defines a component referenced in the step. The analyst defines the simulator tag name of the component along with acceptable limits for the process value required for an affirmative evaluation of the component's state.

The primitives and points do not need to be defined for each assertion. The simulation can run in a dummy mode in which the analyst can define the outcome for each assertion to define a prescribed path of interest to evaluate. This dummy mode of operation precludes the evaluation of the points within the simulator to determine the outcome for the step. The intent of this dummy mode is to support examining the human reliability variables along a prescribed path to examine known scenarios or validate to an empirically observed scenario.

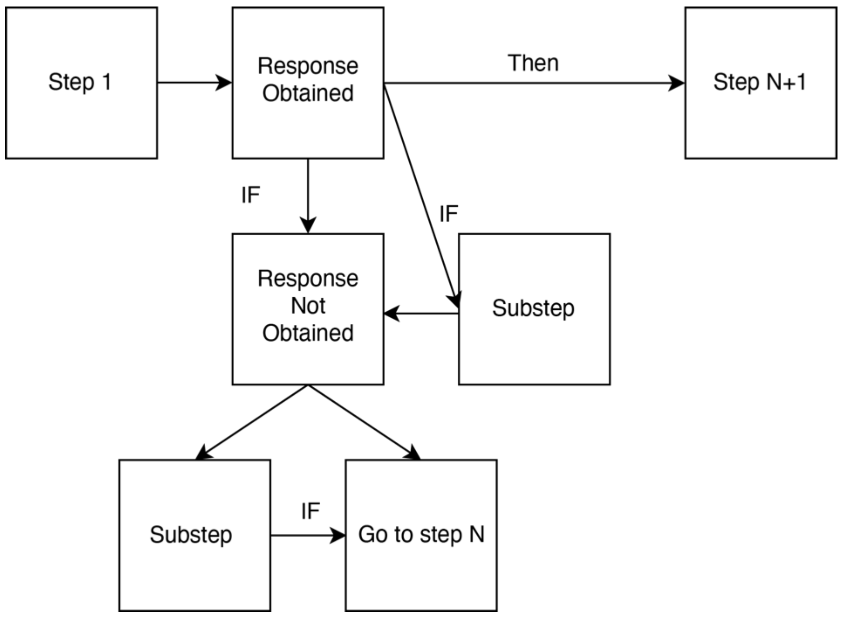


Figure 12. Transition path logic evaluated by the assertion and then stored in the results class.

#### 4.3.2.3 Results Class

Each assertion representing a step or substep is evaluated, and the outcome of that evaluation is stored in the results class. The results class stores the results of the point evaluation based on the state of the simulator at the time the assertion was evaluated. Additionally, the results of the HRA evaluation—comprised of the time elapsed for the execution of the step and the HEP—govern whether it was successful or not. The results also contain the transition information, which controls what step will be executed next within the task module. This transition can take many forms, as can be seen by the arrow denoting the transition paths between the procedure steps based on their execution in Figure 12. The results from each element are output to a CSV data log file and debug log for further analysis.

# 5. INDIVIDUAL MODULE IMPLEMENTATION

## 5.1 Introduction

In this section, we discuss the development of the Individual module, which is currently treated through PSFs. We use dynamic PSFs to model the contextual effects on individual performance within the HUNTER framework. The Individual module aims to qualitatively and quantitatively evaluate PSFs, then modify basic HEPs of tasks assigned from GOMS-HRA primitives (Boring and Rasmussen 2016) in the Task module. The basic strategy of this module uses the theoretical part of the existing static SPAR-H HRA method (Gertman et al. 2005) as is, but optionally suggests a relatively new way to account for PSF effects in a dynamic context. In other words, human reliability analysts can select an option to rate and quantify PSFs between either the static SPAR-H or the dynamic approaches suggested in this section. This strategy aims to reduce confusion that results from the transition from static to dynamic HRA as well as give an opportunity for human reliability analysts to select the better approach depending on the intended use.

To support our dynamic modeling using the eight SPAR-H PSFs, we reviewed human performance literature and developed data-based mathematical models to rate and quantify PSFs in the context of dynamic HRA. In the following subsections, the features of dynamic HRA considered in the HUNTER Individual module development, and how SPAR-H PSFs are treated in the Individual module, will be introduced in order.

Note there are other possible approaches to capturing context and quantifying HRA. The modular nature of HUNTER does not consider SPAR-H to the exclusion of other approaches. It is understood that SPAR-H is a simplified HRA method. Its value to dynamic HRA may be primarily as a proof of context rather than an exhaustive solution to meet all HRA needs. It is also important to note that there are many ways that SPAR-H may be operationalized in dynamic HRA. For example, earlier efforts to make SPAR-H dynamic within HUNTER focused on so-called external PSFs (Boring et al. 2017). External PSFs take contextual factors like plant parameters to build functions linking PSFs to the plant. In this earlier work, the level of the complexity PSF was autocalculated through a multiple regression equation with core temperature and power level as independent variables (Boring et al. 2016). The approach presented here treats PSFs as self-contained functions that do not rely on plant parameters. There is still value in exploring other forms of dynamic PSFs beyond the treatment in this section.

Additionally, this section highlights modifications to the HEP through dynamic PSFs. Research is ongoing to consider the effects of PSFs beyond error probabilities. For example, in a later section (see Section 8.4.1 and following sections) we briefly review the effects of a single PSF—complexity—as a multiplier on time to complete tasks. This novel treatment of PSFs as a time multiplier demonstrates some of the versatility of dynamic HRA. Whereas existing static HRA is limited to considering the effects of PSFs on error rates, dynamic HRA may be used to consider different outputs such as time to complete tasks or effects on decision-making strategies.

## 5.2 General Treatment of Performance Shaping Factors in the HUNTER Individual Module

The Individual module in the HUNTER software has been designed to represent dynamics, meaning human risks over time. Regarding PSFs, it is important to understand how PSF effects change over time and how PSFs interact with PSFs in other human actions. Existing static HRA does consider dynamics, which are often the basis of dependence. Dependence in HRA refers to a method for adjusting the failure probability for the following action by considering the impact of the preceding action (Park and Boring 2021). However, static dependence only considers linkages for human actions for factors such as the same crew, work performed closely in time, overlapping cues, and a shared location between actions (Gertman et al. 2005). This conventional treatment of dependence does not specifically try to see the quantitative influence over time in the relationship between PSFs. Dependence is treated as a correction factor for dynamics rather than as a true model for PSF changes over time.

Representatively, some PSFs such as stress, or fitness for duty are highly sensitive to the effects of time. For example, assume that operators perform many tasks that result in an extremely high stress level. If they do additional tasks right after finishing those stressful tasks, the operators' stress levels for the additional work will be higher by virtue of carryover stress from the previous tasks. On the other hand, if they perform additional tasks after a couple of hours, their stress levels will likely have dissipated and will likely not be affected by the previous stressful tasks. Thus, we have observed evidence of the time effect on PSFs. Nevertheless, it has not been specifically researched and treated in the HRA.

As noted in Boring (2015b), PSFs are subject to carryover effects from one point in time to the next. There are four dynamic memory effects identified in Boring et al. (2016):[d]

- *Lag* is a delay in the onset of the influence of a PSF. Some PSFs may only slowly swell to full effect, despite a series of triggers. Fatigue is an example of a PSF that rarely has sudden effects but rather builds gradually over time. Other PSFs may have little lag and kick in almost instantly. Stress is an example of a PSF that manifests quickly, often as a result of an automatic physiological "fight or flight" mechanism. It is beyond the scope of this paper to speculate on adaptive advantages to having different degrees of lag. Intuitively, the lag in fatigue onset ensures a graceful degradation of performance, thereby ensuring that short-term effects of fatigue do not overwhelm the ability of the individual to function effectively despite waning physical and mental faculties. Conversely, stress, which can help the individual have a rapid response to conditions, is a PSF that should have minimal lag because of its central role in ensuring timely emergency response to protect the individual in threatening situations.
- *Linger* is an effect that continues even after the underlying causes of the PSF cease. For example, cortisol is a stress hormone that rapidly increases the body and brain response to threat stimuli while suppressing certain non-essential functions like digestion and growth that are not germane to the immediate response required of the situation. Cortisol acts rapidly, serving the crucial function of supporting the immediate response needs. Elevated cortisol levels do not immediately dissipate. Even after the stress trigger has disappeared, it will take time for cortisol levels to return to normal in the body. Thus, stress lingers even after the stressor is gone.
- *Memory* (a.k.a., history or hysteresis), which is related to lag and linger, means that the PSF remains anchored in its previous states, preventing dramatic surges in the face of sudden plant upsets or sudden dropouts in the absence of direct influences on the PSF. A memory of previous states reflects the pace of physiological changes in many cases, notwithstanding a sudden onset threat stress (e.g., fight or flight) that dramatically overrides existing physical and mental states. Memory is treated mathematically as a cumulative moving average and serves to smooth the PSF to sudden changes.
- *Decay* is a type of diminution of the effect of the PSF. In the absence of fresh drivers on its performance, a PSF should return to the nominal state over time. For example, elevated stress caused by a plant upset will settle to a normal level after cessation of the event. There is significant linger in a PSF like stress; it does not simply abate when the source of the stress is removed. However, stress will not continue indefinitely, and it will eventually fade to a non-stress state. Thus, a decay function may be built into the basic function of the PSF to afford the gradual return to a predefined nominal state. Note that decay operates counter to linger—decay accelerates change to the PSF, while linger slows it.

These four dynamic PSF functions are summarized in Table 6. In Table 6, let *PSF*(*t*) be the shape function at time *t*. Assume events occur at times $t_i, i = 1, \dots, N$. Lag and linger together are basically a continuity statement that can be combined into the following equation:

$$PSF(t_{i+1}) = \lim_{t \to t_{i+1}} PSF(t) \tag{2}$$

[d] The discussion of PSF memory and decay are taken from Boring et al. (2016).

Table 6. Dynamic functions that may affect the general calculation of the PSF.

| Dynamic PSF Function | Effect on PSF | Notation |
|---|---|---|
| lag | A PSF will be slow to change at the outset of a new effect | $PSF(t_{i+1}) = \lim_{t \to t_{i+1}}{}^{-} PSF(t)$ |
| linger | A PSF will be slow to change at the termination of an existing effect | $PSF(t_{i+1}) = \lim_{t \to t_{i+1}}{}^{+} PSF(t)$ |
| memory | General form of lag and linger, denoting that the effect of the current PSF is a function of preceding values for that PSF | $PSF(t_{i+1}) = f(t_i)$ |
| decay | A PSF will settle to its original state over time | $PSF(t) = PSF(0) \quad for\ t \gg t_N$ |

To consider the effects over time on PSFs in the HUNTER Individual module, this report extends the PSF concept from static to dynamic HRA, as shown in Figure 13. In static HRA, PSFs are primarily used for quantifying at the task or human failure event (HFE) level. In other words, human reliability analysts evaluate PSFs for each task independently, while the relationship between PSFs across different tasks is rarely considered in the analysis. In the extended PSF concept for dynamic HRA (also shown in Figure 13), it is assumed that PSFs in a task affect those in other tasks performed after the task. There are two influences suggested in the concept:

- The intra-PSF influence of a PSF on the same PSF for the next tasks, and
- The inter-PSF influence of a PSF on different PSFs for the next tasks.
- The inter-PSF influence of a PSF on different PSFs during the same task.

An example of intra-PSF influence occurs when the stress PSF in Task 1 affects the stress PSF in Task 2. The case in which the complexity PSF in Task 1 influences the available time PSF in Task 2 shows the example of inter-PSF influence. In this report, we mainly focus on intra-PSF rather than inter-PSF influence. In this conceptualization of PSF influences, it is also assumed that dynamic modeling includes dependency effects between human actions. The third influence—inter-PSF influence during the same task—is primarily an artifact of the lack of independence between PSF definitions (Boring et al. 2006) and does not constitute a specific area of interest for dynamic HRA.

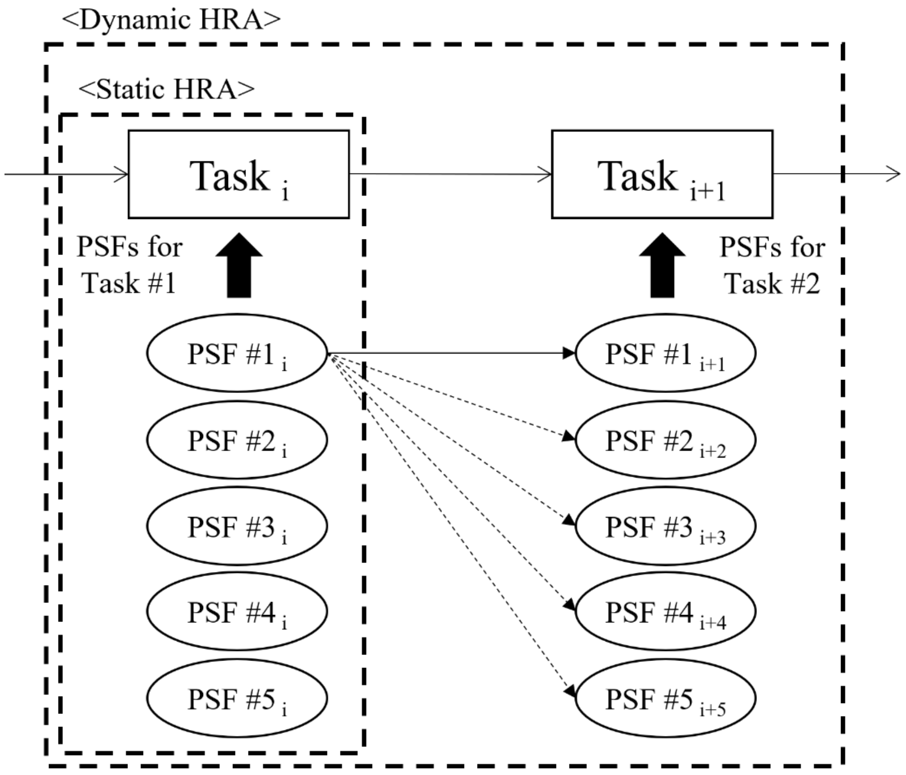


Figure 13. Extension of the PSF concept from static to dynamic HRA.

Table 7. Categorization of PSF qualification and quantification functions.

| | | **Qualification Function** | |
|---|---|---|---|
| | | **Manually Assigned** | **Automatically Assigned** |
| **Quantification Function** | **Static** | The level of the PSF and its multiplier are manually assigned in the model, equivalent to static HRA. | The level of the PSF is automatically assigned, and static (i.e., predefined) multipliers are applied for each level. |
| | **Dynamic** | The level of the PSF is manually assigned, but the multiplier is automatically calculated (e.g., adjusted for lag and linger effects). | The level of the PSF is automatically assigned and the PSF multiplier is autocalculated. |

## 5.3 Treatment of SPAR-H PSFs in the Individual Module

Table 7 summarizes how to treat the SPAR-H PSFs in the Individual module. The Individual module consists of two functions:

- The PSF qualification function, and
- The PSF quantification function.

The PSF *qualification* function is responsible for manually assigning PSF levels, similarly to how this is performed in the existing static SPAR-H method. Alternately, PSF levels may be assigned automatically based on information such as procedure instructions or plant response data from thermal-hydraulic codes or mathematical models. The PSF qualification function includes two categories, "Manually Assigned" and "Automatically Assigned," respectively.

In the PSF *quantification* function, HUNTER uses the selected PSF levels to estimate final HEPs based on PSF multiplier values suggested in the SPAR-H method, or automatically quantifies the values based on mathematical models. This function has two options, "Static" and "Dynamic." "Static" means that multiplier values suggested in the existing static SPAR-H method are used for allocating the multiplier value to the selected PSF level determined in the PSF qualification function. "Dynamic" estimates a PSF multiplier value based on experimental data-based mathematical models. All the options in the PSF qualification and quantification functions are selectable in the HUNTER interface. The details related to the interface are described in Chapter 7.

As mentioned at the beginning of this section, the HUNTER Individual module basically includes a way to use the existing static SPAR-H method with the "Manually Assigned" and "Static" options. Some PSFs, such as stress/stressor, fitness for duty, or available time, have an "Automatically Assigned" option in the PSF qualification function or a "Dynamic" option in the PSF quantification function. In the current version, these dynamic approaches are only available for stress/stressor, fitness for duty, and available time PSFs. However, this doesn't mean that other PSFs can't be automatically or dynamically assigned in future implementations. If there are adequate mathematical models or experimental data to implement dynamic approaches to other PSFs, they will be updated in the future. Table 8 cross-walks the different static and dynamic options for qualification and quantification functions for PSFs.

Table 8. A summary of how to treat the SPAR-H PSFs in the Individual module.

| **SPAR-H PSFs** | **PSF Qualification Function** | **PSF Quantification Function** |
|---|---|---|
| Stress/stressors | • "Manually Assigned" only | • "Static" or "Dynamic" |
| Fitness for duty | • "Manually Assigned"<br>• "Automatically Assigned" (if "Dynamic" is selected in the PSF quantification function) | • "Static" or "Dynamic" |
| Available time | • "Manually Assigned"<br>• "Automatically Assigned" (if "Dynamic" is selected in the PSF quantification function) | • "Static" or "Dynamic" |
| Work processes | • "Manually Assigned" only | • "Static" only |
| Experience/ training | • "Manually Assigned" only | • "Static" only |
| Complexity | • "Manually Assigned" only | • "Static" only |
| Ergonomics/ HMI | • "Manually Assigned" only | • "Static" only |
| Procedures | • "Manually Assigned" only | • "Static" only |

In the following subsections, the details on how to assign and quantify each PSF in the HUNTER Individual module are described.

### 5.3.1 Stress/Stressors

The SPAR-H method (Gertman et al. 2005) defines stress as the level of undesirable conditions and circumstances that impede operators from easily completing a task. Stress can include mental stress, excessive workload, or physical stressors such as that imposed by different environmental factors. Table 9 shows the stress PSF level and multiplier value in the SPAR-H method. In the existing static SPAR-H method, stress is assigned to one of three levels (i.e., Extreme, High, or Nominal). Based on this level assignment, the nominal HEP is then multiplied by one of the three multiplier values (i.e., 5, 2, or 1).

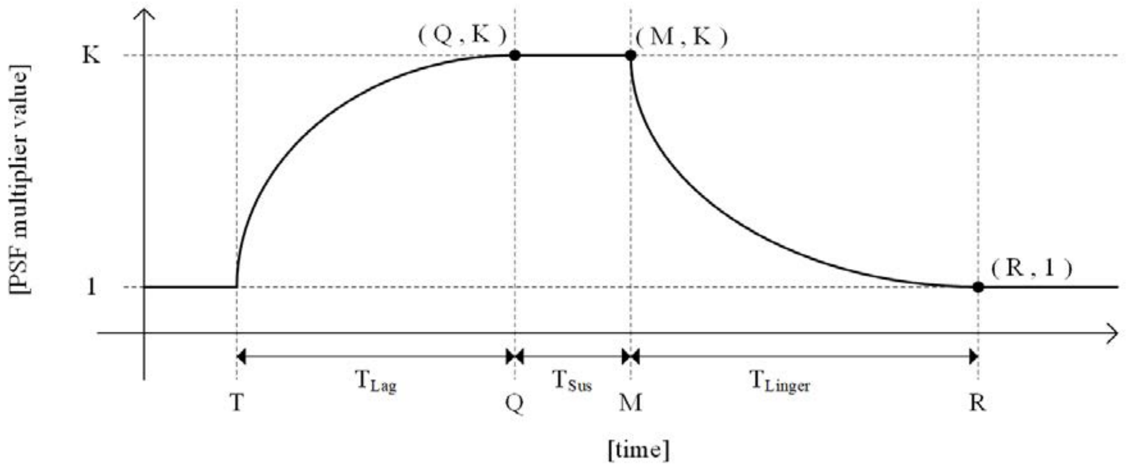
Table 9. Stress PSF level and multiplier value in SPAR-H (Gertman et al. 2005).

| Stress PSF Level | Multiplier Value |
|---|---|
| Extreme | 5 |
| High | 2 |
| Nominal | 1 |

The stress PSF has the "Dynamic" option in the quantification function. We use PSF lag and linger models (Boring 2015b; Park, Boring, and Kim 2019) to implement the dynamic option for stress in the HUNTER Individual module. PSF lag indicates that the effect of the PSF on performance does not immediately psychologically or physically appear, while PSF linger means that the influence of PSFs on previous human actions is not finished after the actions, resulting in residual effects on the next actions. The authors' previous research conceptually suggested PSF lag and linger effects as an option to treat dependence between operator actions in the dynamic context, and then developed mathematical models for PSF lag and linger effects.

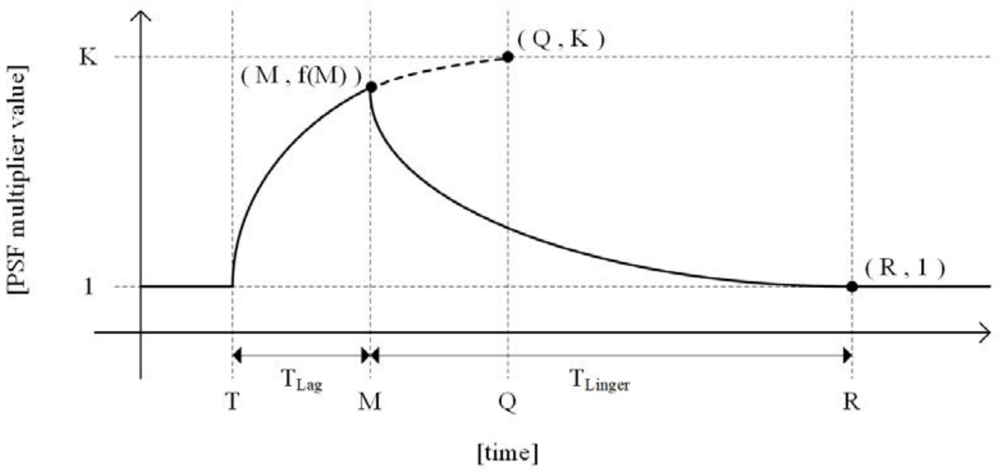
The lag and linger effects for the stress PSF are technically based on experimental research in the field of biology. Several studies demonstrated that secretion of hormones such as cortisol (Dorin et al. 2012) and corticosterone (Vitousek et al. 2018) affects the stress level. Based on the results of these studies, we developed mathematical models for PSF lag and linger effects. First, for modeling the PSF lag effect, we focused on the stress-increasing trend (i.e., how stress reaches a maximum level, and the time that it takes to reach that maximum value). In one of the relevant biology studies, Dorin et al. (2012) tried to estimate maximum cortisol secretion rates. Cortisol is a hormone that increases dramatically during adaptation to physiological stress. The results of their study included a trend that the cortisol concentrations reach maximum level (i.e., natural log function), and the time that it takes to reach the maximum value (i.e., 60 min). Second, we investigated Vitousek et al. (2018) to model the lingering impact of stress. This study experimentally investigated the decrease of corticosterone hormones, which represents the decrease of stress, depending on the change over time. In the results of their study, the concentration of the corticosterone hormone (i.e., stress) exponentially decreases and reaches a normal state after 180 minutes.

Figure 14 and Figure 15 show mathematical models of PSF lag and linger effects for a task when the time to perform a task is more than and less than 60 minutes, respectively. Explanations for the parameters in each figure are shown in Table 10. The biggest difference between the two models concerns PSF lag time. For the first model, the time to perform a task is more than 60 minutes, so the effect of the PSF is sustained at the end of the task. For the second model, the time to perform a task is less than 60 minutes. It is finished before the PSF value reaches its maximum level. In addition, this report only adopts PSF lag and linger effects on negative PSF multiplier values (e.g., ×2 or ×5), while positive PSF multiplier values (i.e., those values that enhance performance) are not considered.



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

Figure 14. A mathematical model of PSF lag and linger effects for a task when the time to perform a task is more than 60 minutes.



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

Figure 15. A mathematical model of PSF lag and linger effects for a task when the time to perform a task is less than 60 minutes.

Table 10. Definition of parameters in mathematical model in Figure 14 and Figure 15.

| Parameter | Definition |
|---|---|
| T | Starting time of a task |
| Q | Time that it takes to reach a maximum PSF value in a task |
| M | Time to finish a task |
| R | Time to return to nominal PSF level (i.e., time that the PSF effect of a task is totally finished) |
| K | A PSF value |
| f(M) | A PSF value limited by lag effect when the time to perform a task is less than 60 minutes |
| $T_{Lag}$ | Delayed time to arrive at a PSF value by the lag effect (maximum 60 minutes) |
| $T_{Sus}$ | Time that the effect of PSFs is sustained by end of the task |
| $T_{Linger}$ | Delayed time to return to nominal PSF level by the linger effect( i.e., 180 minutes) |

It is an established fact that stress is cumulative in the field of biology (Frodl and O'Keane 2013). Accordingly, our research assumes that stressors from different tasks can be temporarily accumulated. After the mathematical model is applied to each stress PSF in each task, we need to incorporate the stress effects into a value representing the stress level of each task with the cumulative effect. Figure 16 shows how the stress value in each task is integrated. The solid line in each timeline represents the task duration from the start to the end. As shown in the figure, the total stress (i.e., the cumulative stress) is calculated by product of stress values of previous and current tasks at the point. For example, the stress of Task $_{i+2}$ is calculated as the product of stress values of Task $_{i}$, Task $_{i+1}$, and Task $_{i+2}$. For Task $_{i}$ in this case, stress is already finished, but there is the lingering stress effect from the mathematical model.

$$\boxed{Total\ Stress\ of\ Task\ i = \prod_{n=1}^{i} Stress\ (Task\ n)}$$

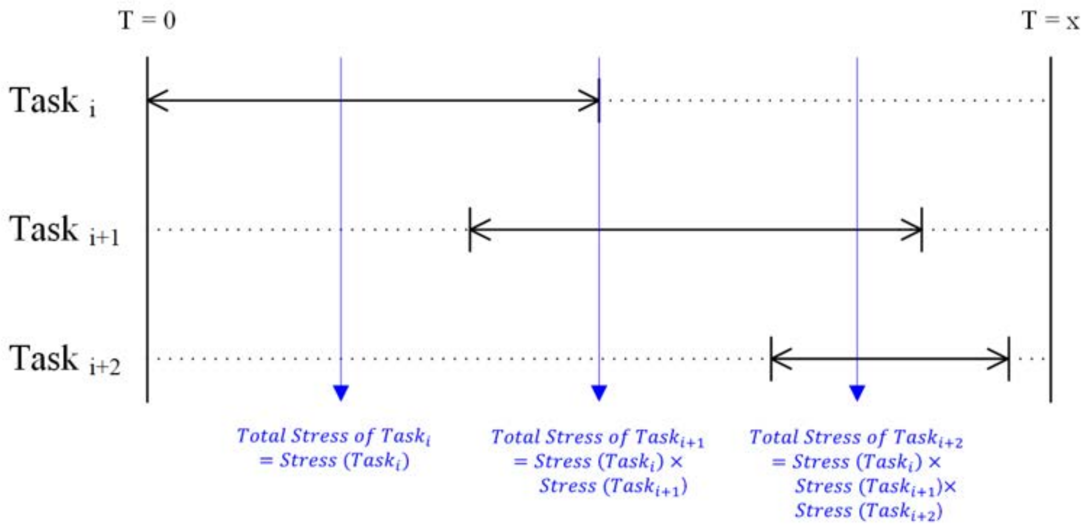

T = 0
T = x
Task $_i$
Task $_{i+1}$
Task $_{i+2}$
$Total\ Stress\ of\ Task_i = Stress\ (Task_i)$
$Total\ Stress\ of\ Task_{i+1} = Stress\ (Task_i) \times Stress\ (Task_{i+1})$
$Total\ Stress\ of\ Task_{i+2} = Stress\ (Task_i) \times Stress\ (Task_{i+1}) \times Stress\ (Task_{i+2})$

Figure 16. An example of stress effect integration.

The necessary input data for running PSF lag and linger models are the starting time of a task, time required, and PSF multiplier value. The starting time of a task can be obtained from thermal-hydraulic codes or simulator models interfaced to HUNTER. The time required is provided from GOMS-HRA. The PSF multiplier value is manually obtained from the HUNTER interface by human reliability analysts. Other PSFs are automatically calculated by equations assumed in the PSF lag and linger models.

Table 11 and Figure 17 show the input data needed and plots representing the result of dynamic calculation of the stress PSF. In the example, two tasks are assumed. The first task is performed between 50 seconds and 250 seconds with 200 seconds time required, while the second task is started at 200 seconds and finished at 800 seconds with 600 seconds time required. In the figure, there are three plots representing the individual stress effects of Tasks 1 and 2, along with the total stress meaning the cumulative stress. The total stress in the duration of Tasks 1 and 2 represents the final stress multiplier value over time with the cumulative effect.

Table 11. An example of input data for dynamically calculating the stress PSF

| **Task No.** | **Parameter** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Time required [sec]** | **T [sec] (Starting time of a task)** | **M [sec] (Time to finish a task)** | **Q [sec] (Time to reach the maximum PSF value)** | **R [sec] (Time to return to nominal PSF level)** | **K [sec] (A PSF value for a task)** | **f(M) [sec] (A PSF value limited by lag effect)** |
| #1 | 200 | 50 | 250 | 3,650 | 11,050 | 2 | 1.648 |
| #2 | 600 | 200 | 800 | 3,800 | 11,600 | 2 | 1.781 |

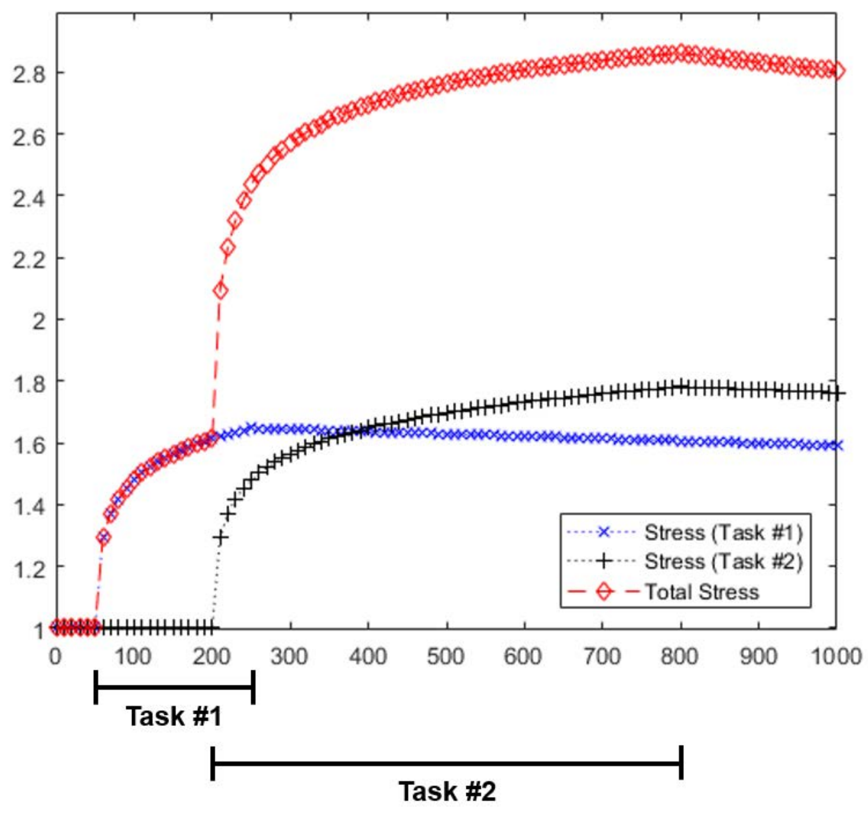


Figure 17. An example of plots from dynamic calculation for the stress PSF.

### 5.3.2 Fitness for Duty

The SPAR-H method (Gertman et al. 2005) describes fitness for duty as whether or not the individual performing the task is physically or mentally fit to perform the task at that time. Things that may affect fitness include fatigue, sickness, drug use, overconfidence, personal problems, and distractions. Table 12 shows the fitness for duty PSF level and multiplier value in the SPAR-H method. In the existing static SPAR-H method, there are three levels (i.e., Unfit, Degraded Fitness, or Normal), which modify nominal HEPs with the two multiplier values (i.e., 5 or 1) or set the task as guaranteed failure (i.e., HEP = 1.0) depending on the levels.

Table 12. Fitness for duty PSF level and multiplier value in SPAR-H (Gertman et al. 2005).

| Fitness for duty PSF Level | Multiplier Value |
|---|---|
| Unfit | P = 1.0 |
| Degraded Fitness | 5 |
| Normal | 1 |

To develop a mathematical model for the fitness for duty PSF, the literature regarding fatigue was reviewed more than the other factors such as sickness, drug use, overconfidence, personal problems, and distractions. Fatigue is a representative fitness factor that has been extensively researched in the field of cognitive engineering (Gersh, McKneely, and Remington 2005). Few studies have been performed to specifically consider other subfactors.

Health and Safety Executive (HSE), which is Britain's national regulator for workplace health and safety, has developed an approach to estimate fatigue index by considering six fatigue factors: (1) duty length, (2) rest length, (3) average duty per day, (4) cumulative component, (5) duty timing component, and (6) job type / breaks component (Health and Safety Executive 2014). The fatigue index trend has been researched for different lengths of shift (Spencer, Robertson, and Folkard 2006). Specifically, the fatigue index values have been calculated based on the four data sets shown in Table 13, while the relative fatigue values have been also estimated by dividing hourly fatigue values into the mean for the first 8 hours, then compared to see that the trends from different data sets are consistent. This study yielded a highly significant main effect of time on shift ($p<0.001$) through a repeated-measures analysis of variance based on relative fatigue values for the four data sets.

Table 13. A summary of the studies across hours on duty (Spencer et al. 2006).

| **Authors** | **Data** | **Measure** | **Total Number** |
|---|---|---|---|
| (Akerstedt 1995) | Sweden (1990/1991) | Lost time injuries (1+ days) | 160,000 |
| (Folkard 1997) | Various Transport Operations | Accidents or signals passed at danger | N/A |
| (Hänecke, Tiedemann, Nachreiner, and Grzech-Šukalo 1998) | Germany (1994) | Lost time injuries (> 3 days) | 1,200,000+ |
| (Nachreiner, Akkermann, & Haenecke 2000) | Germany (1994-1997) | Fatal injuries | 2,000+ |

To account for fitness for duty in HUNTER, we developed an equation representing the relative fatigue values from the data set included in the research introduced above (Spencer et al. 2006). Figure 18 shows the relative fatigue index over hours on duty. The curve-fitted equation is shown as a cubic equation in the figure. The *R*-square value of the equation (i.e., 0.69) shows a statistically adequate level. In the HUNTER Individual module, the equation is used for determining the fitness for duty PSF level as well as quantifying its multiplier value over time. If a shift is changed, the multiplier value starts from 0 again, while the maximum value is assumed as 5 according to the existing SPAR-H method. This equation is automatically applied if the human reliability analyst selects the "Dynamic" option in the HUNTER interface.

An interesting finding in the figure is that some fatigue levels between the second and the fifth hours are slightly higher. In regard to this phenomenon, Folkard (1997) suggested that this fleeting peak in fatigue may be due to the individuals' circadian rhythms, which means physical, mental, and behavioral changes following a 24-hour cycle. In fact, this kind of phenomenon has not been treated in HRA, with one exception in determining cumulative sleep deprivation effects as a PSF (Boring et al. 2020). Nevertheless, using the data-driven model over time may give us an opportunity to include a multimodal curve and realistically reflect the effect of the fitness of duty PSF in HRA.

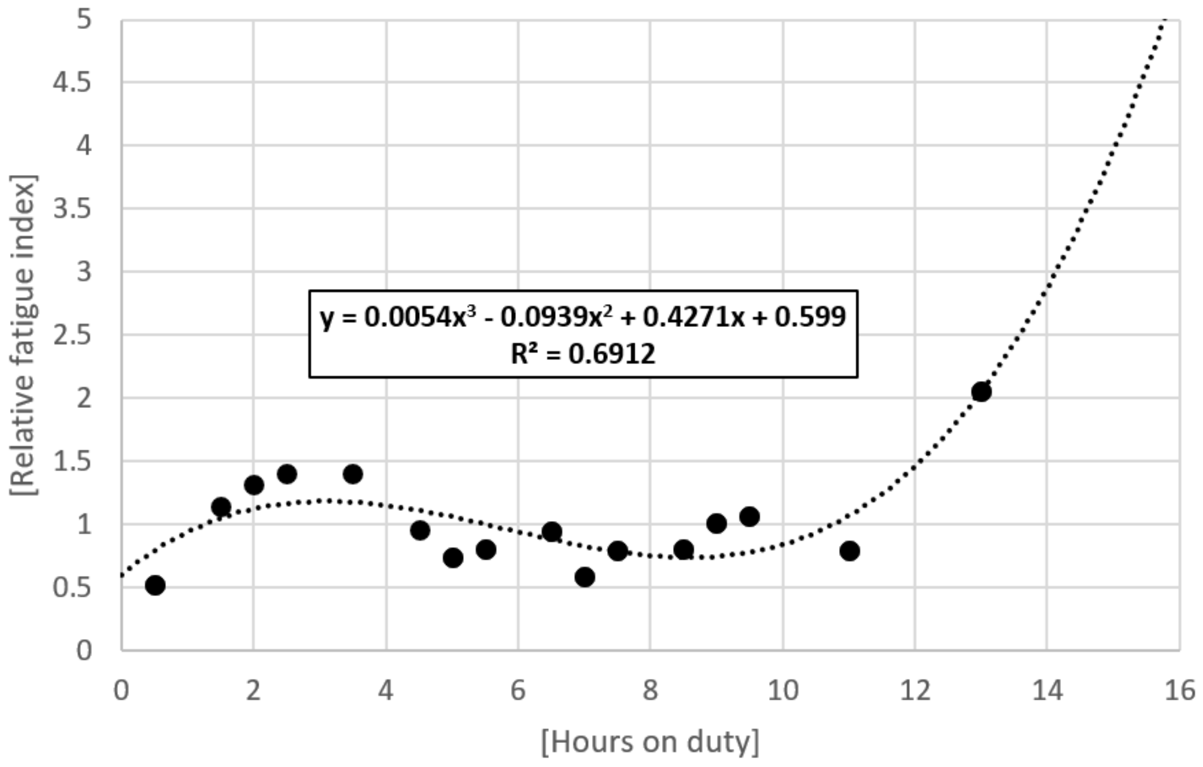


Figure 18. The relative fatigue index over hours on duty.

### 5.3.3 Available Time

Available time refers to the amount of time that an operator or a crew has to diagnose the situation and execute a task. Table 14 shows the available time PSF level and multiplier value in SPAR-H. To evaluate the PSF level for the available time PSF, it is necessary to obtain time information such as the system time window ($T_{SW}$), delay time ($T_{Delay}$), time available ($T_{Available}$), time required for diagnosis ($T_{Diagnosis}$), and time required for action or exectution[e] ($T_{Execution}$) as shown in Figure 19. The figure shows the timeline of existing HRA. The details on the timeline and its components are well described by the Electric Power Research Institute (2016).

The available time PSF has different criteria to select a PSF level depending on task types such as diagnosis or action. For example, if we see the same multiplier value (0.1) in the task types (extra time in diagnosis vs. time available ≥ 5 * time required in execution), the extra time in diagnosis is evaluated when time available is between 1–2 times greater than the nominal time required for diagnosis and is also greater than 30 minutes, while the PSF level for the execution is determined if the time available is 5 times higher than the time required for execution.

[e] The term *action* is used interchangeably with *execution* in HRA. SPAR-H prefers the term action.

Table 14. Available time PSF level and multiplier value in SPAR-H (Gertman et al. 2005).

| Task Type | Available Time PSF Level | Multiplier Value |
|---|---|---|
| **Diagnosis** | Inadequate time | P = 1.0 |
| | Barely adequate time | 10 |
| | Nominal time | 1 |
| | Extra time | 0.1 |
| | Expansive time | 0.01 |
| **Action** | Inadequate time | P = 1.0 |
| | Time available = time required | 10 |
| | Nominal time | 1 |
| | Time available >= 5 * time required | 0.1 |
| | Time available >= 50 * time required | 0.01 |

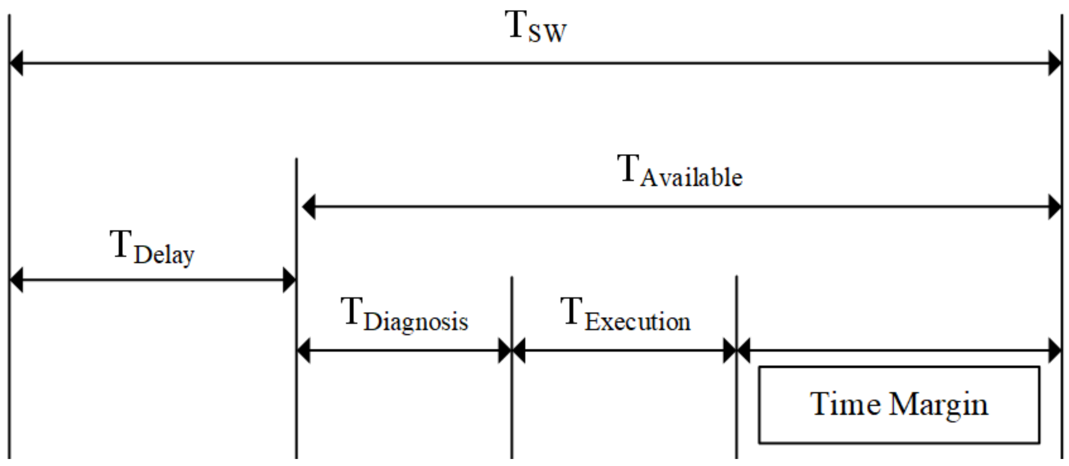


Figure 19. Time factors in existing HRA methods (Electic Power Research Institute 2016).

To design a function for automatically evaluating the PSF level of the available time PSF, we first classify the GOMS-HRA talk level primitive types into diagnosis and action as shown in Table 15.

Table 15. Classification of GOMS-HRA task-level primitive types.

| Task Level Primitive | Description | Task Type |
|---|---|---|
| $A_C$ | Performing required physical actions on the control boards | Action |
| $A_F$ | Performing required physical actions in the field | Action |
| $C_C$ | Looking for required information on the control boards | Action |
| $C_F$ | Looking for required information in the field | Action |
| $R_C$ | Obtaining required information on the control boards | Action |
| $R_F$ | Obtaining required information in the field | Action |
| $I_P$ | Producing verbal or written instructions | Action |
| $I_R$ | Receiving verbal or written instructions | Action |
| $S_C$ | Selecting or setting a value on the control boards | Action |
| $S_F$ | Selecting or setting a value in the field | Action |
| $D_P$ | Making a decision based on procedures | Diagnosis |
| $D_W$ | Making a decision without available procedures | Diagnosis |

Second, we modify the existing time factors in Figure 19 so that they are applicable to the HUNTER framework for dynamic HEP calculation. Figure 20 shows the HUNTER timeline, which includes the system time window ($T_{SW}$), time that a task is performed ($T_{Task}$), time to start a task ($T_{Start}$), and time available ($T_{Available}$). First, in the HUNTER software, the system time window ($T_{SW}$) is automatically provided from the connected thermal-hydraulic model. Second, the time that a task is performed ($T_{Task}$) is used on behalf of the time required for diagnosis and execution in the existing SPAR-H method. Third, the time to start a task ($T_{Start}$) is used instead of the delay time. The delay time refers to the duration of time it takes for an operator to acknowledge the cue from the start of an initiating event. It has been generally estimated using interviews with experts such as operators or instructors who have a lot of experience on NPP systems. However, in the dynamic context, the delay time may include delays provided by thermal-hydraulic codes or time values estimated from the GOMS-HRA task level primitive time distributions. Therefore, rather than using the delay time concept, it may be preferable to use the time to start a task ($T_{Start}$), which can be obtained from the HUNTER scheduler. Last, the definition of the time available ($T_{Available}$) is similar with the existing definition. It is calculated by $T_{Available}$ minus $T_{Start}$.

Finally, this report suggests the logic to determine a multiplier value of the available time PSF with the modified timeline and the existing logic. Table 16 shows the logic to determine a multiplier value of the available time PSF, while Table 17 shows an example of the available time PSF calculation. This logic is automatically applied if the human reliability analysts select the "Dynamic" option in the HUNTER interface.

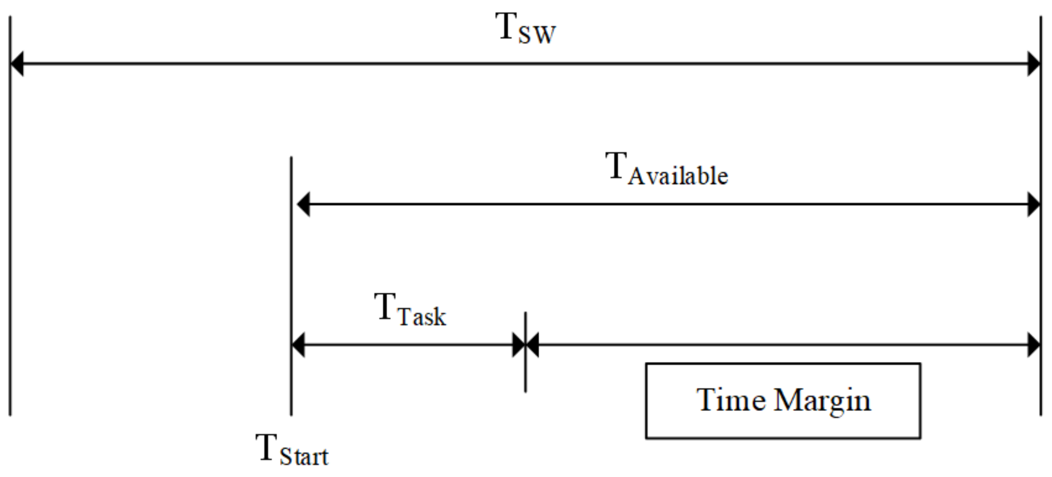


Figure 20. The HUNTER timeline.

Table 16. The logic to determine a multiplier value of the available time PSF

| **GOMS-HRA Primitive Type** | **Evaluation Logic** | **Multiplier Value** |
|---|---|---|
| **Diagnosis-based primitive** | ($T_{Available}$ > 30 mins) & ($T_{Available}$ / $T_{Task}$ >2) | 0.01 |
| | ($T_{Available}$ > 30 mins) & ($T_{Available}$ / $T_{Task}$ <=2) | 0.1 |
| | ($T_{Available}$ <=30mins) | 1 |
| | ($T_{Available}$ < $T_{Task}$) | Task Failed |
| **Action-based primitive** | ($T_{Available}$ / $T_{Task}$ >=50) | 0.01 |
| | (50>$T_{Available}$ / $T_{Task}$ >=5) | 0.1 |
| | (5>$T_{Available}$ / $T_{Task}$ >=1) | 1 |
| | ($T_{Available}$ < $T_{Task}$) | Task Failed |

Table 17. An example of the available time PSF calculation.

| $T_{Start}$ [sec] | $T_{SW}$ [sec] | $T_{Task}$ [sec] | $T_{Available}$ [sec] | The multiplier value for diagnosis-based primitives | The multiplier value for execution-based primitives |
|---|---|---|---|---|---|
| 0 | 3000 | 50 | 3000 | 0.01 | 0.01 |
| 100 | 3000 | 50 | 2900 | 0.01 | 0.01 |
| 200 | 3000 | 50 | 2800 | 0.01 | 0.01 |
| 300 | 3000 | 50 | 2700 | 0.01 | 0.01 |
| 400 | 3000 | 50 | 2600 | 0.01 | 0.01 |
| 500 | 3000 | 50 | 2500 | 0.01 | 0.01 |
| 600 | 3000 | 50 | 2400 | 0.01 | 0.1 |
| 700 | 3000 | 50 | 2300 | 0.01 | 0.1 |
| 800 | 3000 | 50 | 2200 | 0.01 | 0.1 |
| 900 | 3000 | 50 | 2100 | 0.01 | 0.1 |
| 1000 | 3000 | 50 | 2000 | 0.01 | 0.1 |
| 1100 | 3000 | 50 | 1900 | 0.01 | 0.1 |
| 1200 | 3000 | 50 | 1800 | 1 | 0.1 |
| 1300 | 3000 | 50 | 1700 | 1 | 0.1 |
| 1400 | 3000 | 50 | 1600 | 1 | 0.1 |
| 1500 | 3000 | 50 | 1500 | 1 | 0.1 |
| 1600 | 3000 | 50 | 1400 | 1 | 0.1 |
| 1700 | 3000 | 50 | 1300 | 1 | 0.1 |
| 1800 | 3000 | 50 | 1200 | 1 | 0.1 |
| 1900 | 3000 | 50 | 1100 | 1 | 0.1 |
| 2000 | 3000 | 50 | 1000 | 1 | 0.1 |
| 2100 | 3000 | 50 | 900 | 1 | 0.1 |
| 2200 | 3000 | 50 | 800 | 1 | 0.1 |
| 2300 | 3000 | 50 | 700 | 1 | 0.1 |
| 2400 | 3000 | 50 | 600 | 1 | 0.1 |
| 2500 | 3000 | 50 | 500 | 1 | 0.1 |
| 2600 | 3000 | 50 | 400 | 1 | 0.1 |
| 2700 | 3000 | 50 | 300 | 1 | 0.1 |
| 2800 | 3000 | 50 | 200 | 1 | 1 |
| 2900 | 3000 | 50 | 100 | 1 | 1 |
| 3000 | 3000 | 50 | 0 | Task Failed | Task Failed |

Note that available time is a unique PSF that requires inputs from the Task, Individual, and Environment modules in HUNTER. A particularly unique aspect of available time is the run-ahead function required of the Environment module, in which the system time window ($T_{SW}$) must be calculated without operator intervention. This approach runs counter to the tight model coupling described in Chapter 6, because it runs the thermal-hydraulic code in batch mode first to obtain additional values used in HUNTER. Nonetheless, a real-time generator of the system time window proves a valuable approach that demonstrates a unique value proposition of dynamic HRA in HUNTER.

### 5.3.4 Other SPAR-H PSFs

As shown in Table 8, the current version of the HUNTER Individual module only treats the static evaluation options for the other SPAR-H PSFs such as work processes, experience or training, complexity, and ergonomics or HMI, and procedures. In the current HUNTER software, most input information for the dynamic HRA relies on parameters from thermal-hydraulic codes and procedures.

Accordingly, HUNTER may not be able to treat all the PSFs based alone on these parameters. For this reason, the current version of HUNTER opted to evaluate these PSFs through static options, but it is too early to conclude that this is a limiting factor on the versatility of dynamic PSFs. We have found some literature and methods potentially useful dynamicize the PSFs, although these need to be modified to apply within HUNTER. Research related to the complexity PSF suggests an equation to quantify complexity (Boring et al. 2016) or an approach to quantify task complexity based on procedures (Jang, Kim, and Park 2021). As briefly mentioned in Section 5.3, there is ongoing research to find additional mathematical models, collect necessary data to develop the models applicable to NPPs, and validate whether the models are meaningful within the HRA.

# 6. ENVIRONMENT MODULE IMPLEMENTATION

## 6.1 Purpose

The Environment module in HUNTER represents the physical environment—in this case, a nuclear power plant—with which the virtual operator interacts. HUNTER's main purpose is to simulate virtual operations, analyze the virtual operating situation, and derive HRA results, so it is essential that HUNTER reflect the response of the plant in tandem with the operator performing the plant operations. Plant status and plant operation are mutually related, since procedure progress may change depending on plant status, and plant parameters may change according to operator actions. From a dynamic modeling point of view, coupling with a plant simulator is necessary to capture human operations that change depending on plant status or particular guidance offered at different steps in the procedures. In this section, we will describe the coupling of HUNTER with a separate software code capable of plant simulation. This coupling reflects the operatorss manipulations during plant simulation and includes the plant status needed for procedure progress. This chapter focuses specifically on the implementation of the Environment module using RELAP5-3D.

## 6.2 Coupling between HUNTER and a Plant Simulator

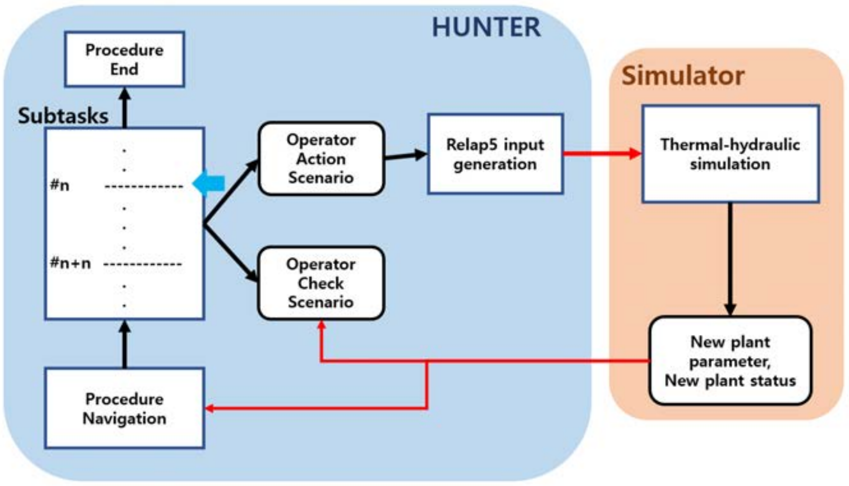


Figure 21. Coupling scheme.

To reflect the interplay between plant state and operator response, HUNTER's virtual operator is linked to an external plant model. In HUNTER, an operator manipulation is applied to the plant simulator or thermal-hydraulic code. It must be possible to perform actions (i.e., tasks) that directly affect plant operations, such as closing valves or operating pumps. In turn, the operator monitors parameters of the plant. Some parameters are monitored regularly to determine routine or abnormal plant evolutions. The HUNTER virtual operator, as guided by continuous and one-time tasks in the procedures, checks for parameter changes or compares plant parameters with specific values presented in the procedures. Additionally, when the operator affects a change on the plant, they confirm the changes occurred as desired.

Thus, the interface between the virtual operator in HUNTER and the virtual plant is a two-way feedback loop, as depicted in Figure 21. The operator must be able to receive parameters from the plant model, and they must be able to manipulate or control aspects of the plant. To accomplish this interaction, we have implemented a two-way synchronous coupling between HUNTER and a plant simulator:

- *Monitor (Plant → Operator):* HUNTER reads parameter or component information reflected in the plant simulation.
- *Control (Operator → Plant):* In turn, HUNTER can alter the state of the plant simulation through virtual operator actions.

These two functions operate in parallel in a dynamic fashion. The plant model progresses with or without virtual operator input. The plant parameters are available to the virtual operator during this progression. When the operator intercedes (e.g., by closing a valve), this is reflected in the plant model as a change. There is an iterative feedback loop between the HUNTER virtual operator and the plant model, progressing to a particular stopping point for the scenario. The human and plant models are synchronized through information exchange (i.e., parameter monitoring and operator actions) through the coupling mechanism. Hold points may be employed to wait for particular information exchanges. For example, the operator model may be in a wait-and-monitor mode awaiting a particular parameter level from the plant model. Similarly, the plant model may simulate plant functions for a particular interval up to a synchronization point and pause for potential input like operator actions from HUNTER. Coupling may occur at regular intervals or may be irregular depending on the type of plant activity being monitored or controlled.

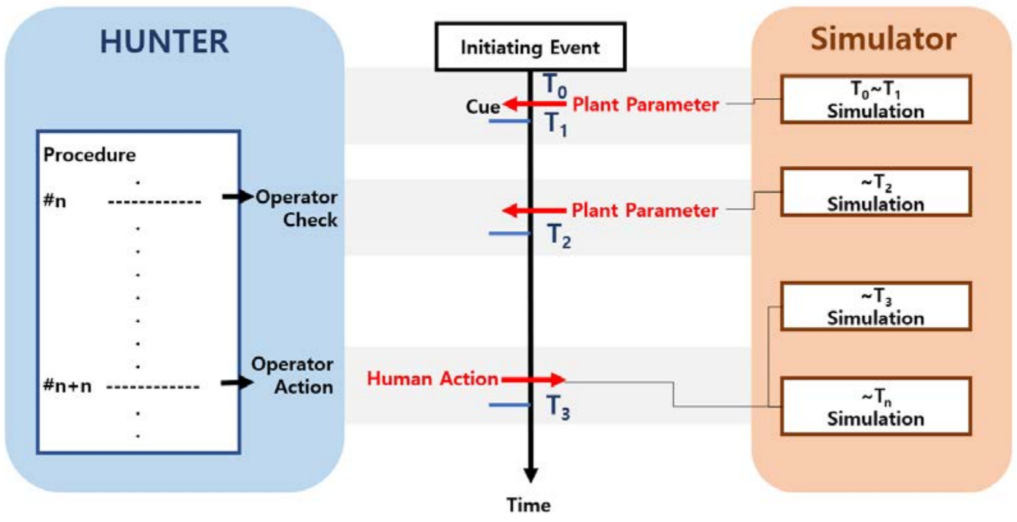


Figure 22. Concept idea of coupling between HUNTER and plant simulator.

An irregular interval coupling is depicted in Figure 22. The HUNTER virtual operator performs plant operations as specified in the procedure, each procedure step taking different times according to the GOMS-HRA primitives. The plant also changes over time, with or without actions from the operator, reflecting the natural progression of plant conditions. Of course, plants are designed to need control actions by the operators, and the desired evolution of the plant is determined by human actions. For this reason, the operator's actions should be reflected in the simulator or thermal-hydraulic code, and the plant status or plant response is transmitted to HUNTER. A typical course of action is:

- A procedure step directs the operator to take an action like closing a valve
- HUNTER transmits the valve's changed state to the thermal-hydraulic code
- The thermal-hydraulic code changes the plant state or model to reflect the closed valve and continues
- The next step in the procedure directs the operator to confirm that the valve is closed
- HUNTER retrieves the valve status from the thermal-hydraulic code
- HUNTER performs the logic check on the status of the valve and proceeds to the next step if the valve is closed or to alternate RNO steps to deal with the stuck-open valve.

HUNTER and the thermal-hydraulic or plant simulator should exchange information similar to how the operator checks plant parameters and makes decisions. Both operator and plant function independently, but they synchronize at various points. HUNTER uses task level primitives in GOMS to estimate how long each procedure step takes to perform. The thermal-hydraulic code will run to this point and then pause for synchronization, either a checking of the plant state or a manual action performed by the virtual operator on the plant model.

## 6.3 RELAP5-3D Coupling with HUNTER

The coupling task described in this report was performed based on a computer simulation software dedicated to nuclear power plant operational thermal-hydraulics analysis. RELAP5-3D was developed at INL for the analysis of transients and accidents in water–cooled NPPs and related systems as well as the analysis of advanced reactor designs (Choi 2019). The specific reason for selecting RELAP5-3D is that we can analyze various plants (assuming the specific plant models exist), including for established plant designs as well as for relatively recently developed plants such as small modular reactors. Recent developments in RELAP5-3D include implementation in INL's supercomputer cluster, called the High Performance Computing (HPC) environment. Using the massively parallel HPC, thermal-hydraulic code calculation can be achieved considerably faster than using other plant simulators. We have also developed codes dealing with RELAP5-3D using Risk Analysis and Virtual Control Environment (RAVEN) developed by INL (Choi et al. 2021). RAVEN has been coupled with RELAP5-3D for streamlining risk analysis using thermal hydraulics. The RAVEN interface simplifies updating RELAP5-3D input data and also makes it possible to conduct multiple simulation runs in parallel. RAVEN receives data and regenerates input decks for RELAP5-3D. Once the regenerated simulation is completed, RELAP5-3D provides data back to RAVEN for post-processing (Choi 2019; Choi et al. 2020; Choi et al. 2021; Choi 2021). Post-processing links the thermal-hydraulic output with time and saves the results in a CSV format that may be readily used by data analysis software. It provides the possibility for various analyses in the future by allowing use of distribution samples for uncertainty analysis.

In regular RELAP5-3D simulations, an analyst sets every sequential propagation of an initiating event before running. However, we have implemented a simulation in this coupling that leaves all possibilities open to reflect operator interaction with the plant. In HUNTER coupled with RELAP5-3D, as briefly described above, the simulation starts with the virtual operator monitoring the plant parameters. The virtual operator follows the procedure and performs an action on the plant according to the plant response. The RELAP5-3D simulation runs for the time duration determined by the operator activities in the procedures. It then pauses the simulation, updates the RELAP5-3D model, restarts the simulation, and continues. A detailed description of this coupling is given below.

If the target task is an action, the task can be terminated immediately by satisfying the procedure's criteria after the operator performs the action. If, on the other hand, operator manipulation involves performing an action to a certain point (e.g., opening a valve until level reaches a certain level), this process involves iteratively performing an action and checking the parameter until it satisfies the criterion. It should not be decided a priori that the operator will always complete this task successfully. The operator may fail to complete the task. For example, another task could distract the operator from successfully completing the task, or the operator may fail to actuate the valve properly over time. Coupling should be performed step by step between HUNTER and RELAP5-3D rather than assuming a best human outcome.

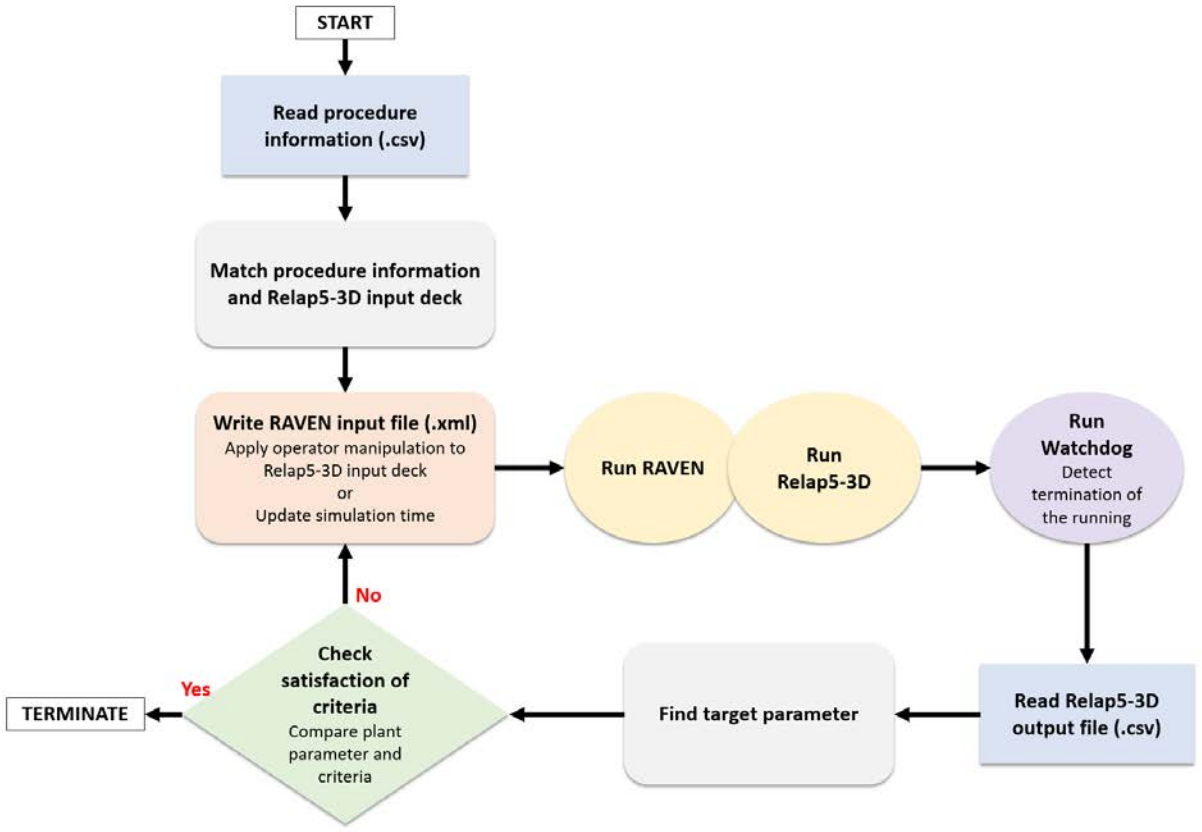

START
Read procedure information (.csv)
Match procedure information and Relap5-3D input deck
Write RAVEN input file (.xml)
Apply operator manipulation to Relap5-3D input deck
or
Update simulation time
Run RAVEN
Run Relap5-3D
Run Watchdog
Detect termination of the running
Read Relap5-3D output file (.csv)
Find target parameter
Check satisfaction of criteria
Compare plant parameter and criteria
No
Yes
TERMINATE

Figure 23. Loop logic for HUNTER and RELAP5-3D coupling.

The coupling between HUNTER and the Environment module is shown as a flow chart in Figure 23. First, HUNTER takes information for each procedure step. The procedure specifies three possible activities relative to the Environment module:

- Read a particular parameter from the RELAP5-3D plant model
- Change a characteristic (e.g., open a valve) in the RELAP5-3D plant model

- Wait a particular interval, during which there is no interaction between the plant model and the HUNTER.

HUNTER checks the operator manipulation and which parameter and component are the targets through this file. Second, RAVEN (as controlled by HUNTER) performs a mapping to the RELAP5-3D input deck to apply target information for the plant simulations. RAVEN connects each component and parameter to the card number defined in the RELAP5-3D input deck. Third, RAVEN updates the connected RELAP5-3D input deck card and runs the simulation. Last, HUNTER compares the derived results with the criteria provided by the procedure. If the criteria are not met, HUNTER increases the simulation clock and reruns the simulation. This synchronous back and forth is called loop logic.

### 6.3.1 Code Structure

We developed RAVEN and RELAP code to accomodate the overall coupling. Several scripts include a script to read the procedure CSV file provided by HUNTER, map procedure information and RELAP5-3D, and write and edit RAVEN input files. Moreover, we have made a script to monitor RELAP5-3D simulation, a script to read the results of the simulation, and a script that updates the simulation clock to re-run the simulation if it does not meet the criteria.

Mapping translates the component or plant parameters received from the procedure into RELAP5-3D information. This is completed by implementing operator manipulations in the RELAP5-3D model through procedure analysis and linking the scripts with procedure information. When the target plant is determined, analysts can easily add one-by-one mapping through the trip cards and the component card in the RELAP5-3D input deck. When reading procedure information, the HUNTER Environment module finds and reads specific information using parameters defined in the procedure file (see Chapter 4). When reading RELAP5-3D results, it finds and reads appropriate parameters and times suitable for loop logic, as shown in Figure 23. In addition, we developed subroutines that update the XML file to modify and create the RAVEN input file. This changes the plant variables by adjusting the input deck of RELAP5-3D. There is also a feature accounting for operator time to complete tasks. When comparing more than two plant parameters, this is accomplished in the library subroutines. Finally, since we cannot predict the required simulation time of RELAP5-3D, hold logic is implemented to synchronize HUNTER with RELAP5-3D. When the plant simulation result is saved as a CSV file, the file can be found and read. File generation using a server is performed in INL's HPC, but we have developed the coupling between both HPC and a computer running a local copy of RELAP5-3D. In the script for monitoring a typical desktop computer, we used the Watchdog function, a Python API and shell utility, to monitor file system events to detect every file and directory generated, modified, or deleted. The information is therefore exchanged through files such as input decks for RELAP5-3D and CSV parameter logs for HUNTER.

## 6.4 Proof-of-Concept Coupling Demonstration

### 6.4.1 Overview

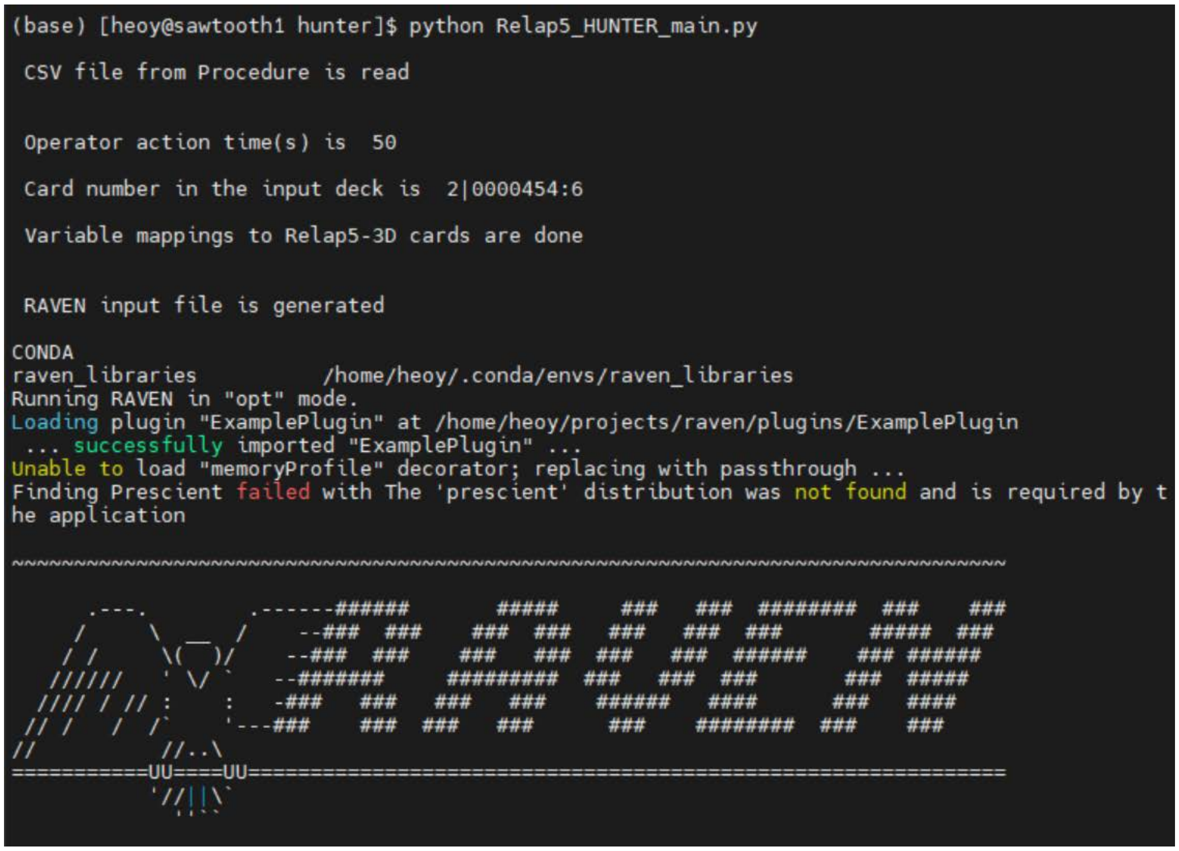
To simulate the two-way coupling between the virtual operator in HUNTER and the plant response in RELAP5-3D, we must perform an operator manipulation and check plant status. Therefore, like a virtual operator, we need to know how to imitate actions such as checking plant parameters and confirming changes or waiting for plant parameter changes to a specific set point. The code can perform plant simulation using loop logic to connect HUNTER and RELAP5-3D. To demonstrate this Environment module implementation, we present an arbitrary example. The arbitrary example was defined through the following modeling criteria:

1. Include an operator action
2. Include tasks that check plant status
3. Check whether the criteria are met by monitoring two or more parameters simultaneously

According to the above criteria, the example scenario for this demonstration is defined as follows:

1. Operators shut the main coolant pump in 50 seconds
2. Operators check the following plant parameter changes and confirm their criteria
    a. Criteria for parameter 1 (center core channel temperature): over 629
    b. Criteria for parameter 2 (lower plenum mass flow rate): under 818

We assumed that the operator stops the pump, checks the parameter change, checks the specific criteria to perform the next procedure task, then moves on to the next task. The RELAP5-3D model used is the INL Generic Pressurized Water Reactor (IGPWR), which is generic 3-loop Westinghouse pressurized water reactor model. Also, the model simulates SGTR (tube rupture occurs in 0 second), but when proceeding with the scenario, arbitrary operator actions independent of the procedure or event tree are simulated (Ma et al. 2021).



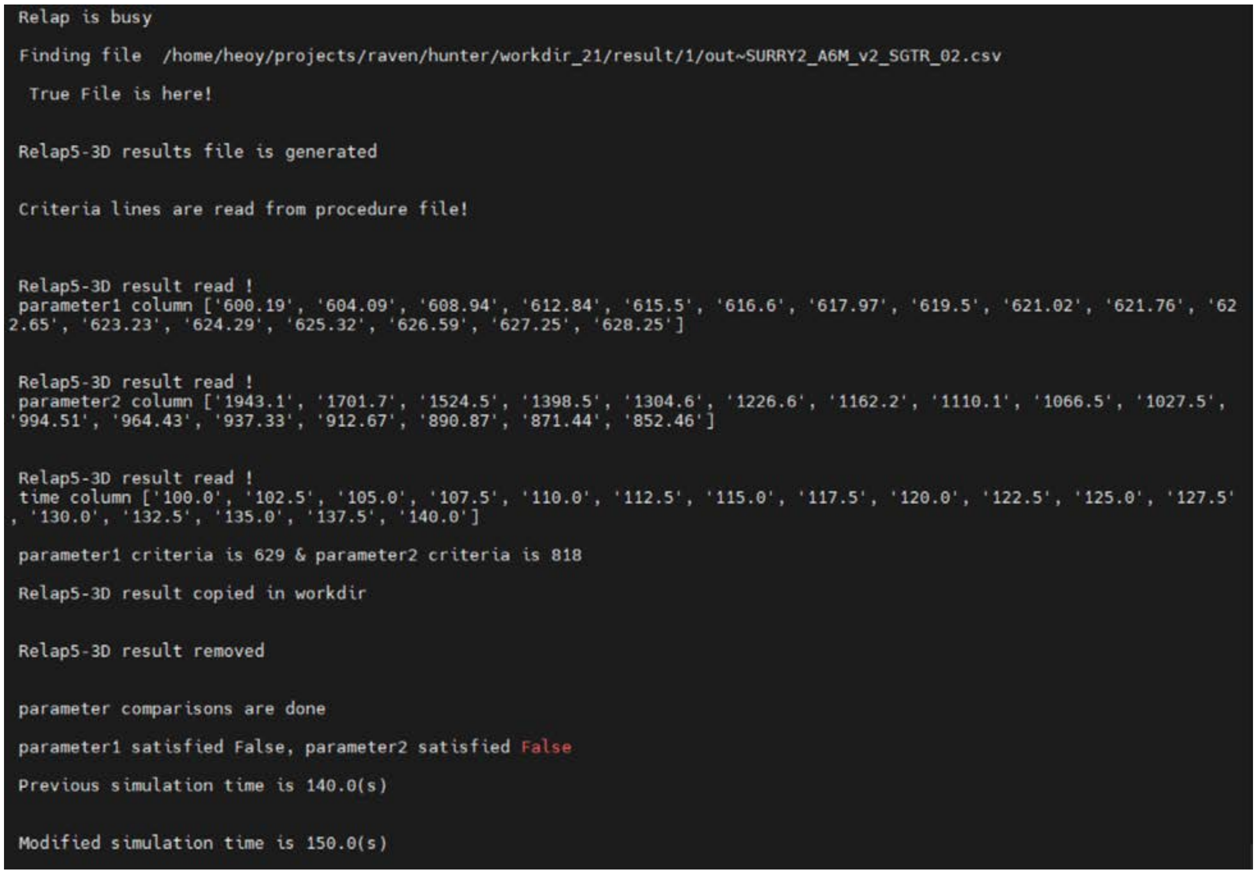
```
(base) [heoy@sawtooth1 hunter]$ python Relap5_HUNTER_main.py
 CSV file from Procedure is read


 Operator action time(s) is  50

 Card number in the input deck is  2|0000454:6

 Variable mappings to Relap5-3D cards are done


 RAVEN input file is generated

CONDA
raven_libraries          /home/heoy/.conda/envs/raven_libraries
Running RAVEN in "opt" mode.
Loading plugin "ExamplePlugin" at /home/heoy/projects/raven/plugins/ExamplePlugin
 ... successfully imported "ExamplePlugin" ...
Unable to load "memoryProfile" decorator; replacing with passthrough ...
Finding Prescient failed with The 'prescient' distribution was not found and is required by t
he application

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

      .---.        .------######       #####     ###   ###  ########  ###    ###
     /     \  __  /    --###  ###    ###  ###   ###   ###  ###       #####  ###
    / /     \(  )/    --###  ###    ###   ###  ###   ###  ######    ### ######
   //////   ' \/ `   --#######    #########  ###   ###  ###       ###  #####
  //// / // :    :   -###   ###   ###   ###   ######   ####      ###   ####
 // /   /  /`    '---###    ###  ###   ###     ###    ########  ###    ###
//          //..\
===========UU====UU=================================================================
            '//||\`
              ''``
```

Figure 24. Screenshot of running codes (1/4).

### 6.4.2 Results of Module Runs

As shown in Figure 24, the information is read from the procedure first when we run the main script using Python. This part receives procedure information from HUNTER. In addition, the information is matched directly to the RELAP5-3D input deck and can be checked by analysts. Moreover, it generates the RAVEN input file using the mapped information and runs RAVEN. In INL HPC, RAVEN executes

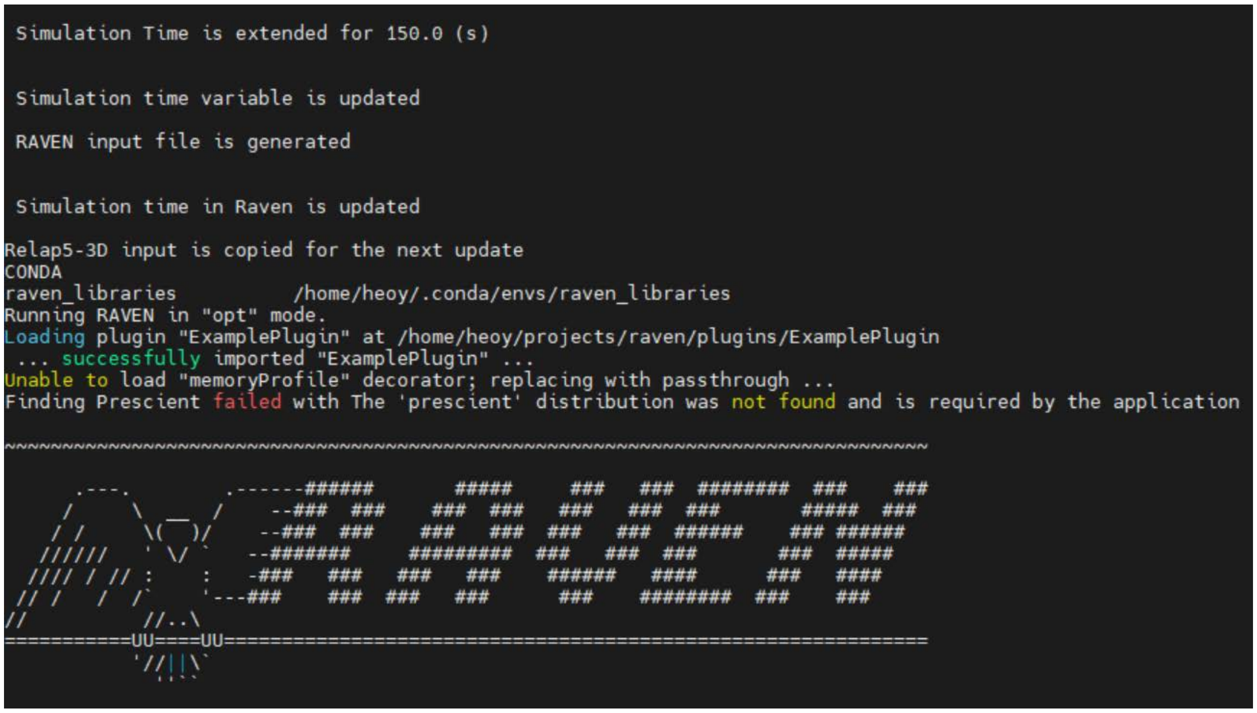
RELAP5-3D by sending jobs to the HPC server. When RELAP5-3D completes the plant simulation calculation, results are logged in a CSV file format. The input deck determines which parameters will be logged as well. Therefore, the analyst must either insert the parameters of interest into the RAVEN input file to be applied to the RELAP5-3D input deck or pre-calibrate the RELAP5-3D input deck.



```
 Relap is busy

 Finding file  /home/heoy/projects/raven/hunter/workdir_21/result/1/out~SURRY2_A6M_v2_SGTR_02.csv

  True File is here!


 Relap5-3D results file is generated


 Criteria lines are read from procedure file!



 Relap5-3D result read !
 parameter1 column ['600.19', '604.09', '608.94', '612.84', '615.5', '616.6', '617.97', '619.5', '621.02', '621.76', '62
2.65', '623.23', '624.29', '625.32', '626.59', '627.25', '628.25']


 Relap5-3D result read !
 parameter2 column ['1943.1', '1701.7', '1524.5', '1398.5', '1304.6', '1226.6', '1162.2', '1110.1', '1066.5', '1027.5',
'994.51', '964.43', '937.33', '912.67', '890.87', '871.44', '852.46']


 Relap5-3D result read !
 time column ['100.0', '102.5', '105.0', '107.5', '110.0', '112.5', '115.0', '117.5', '120.0', '122.5', '125.0', '127.5'
, '130.0', '132.5', '135.0', '137.5', '140.0']

 parameter1 criteria is 629 & parameter2 criteria is 818

 Relap5-3D result copied in workdir


 Relap5-3D result removed


 parameter comparisons are done

 parameter1 satisfied False, parameter2 satisfied False

 Previous simulation time is 140.0(s)


 Modified simulation time is 150.0(s)
```

Figure 25.Screenshot of running codes (2/4).

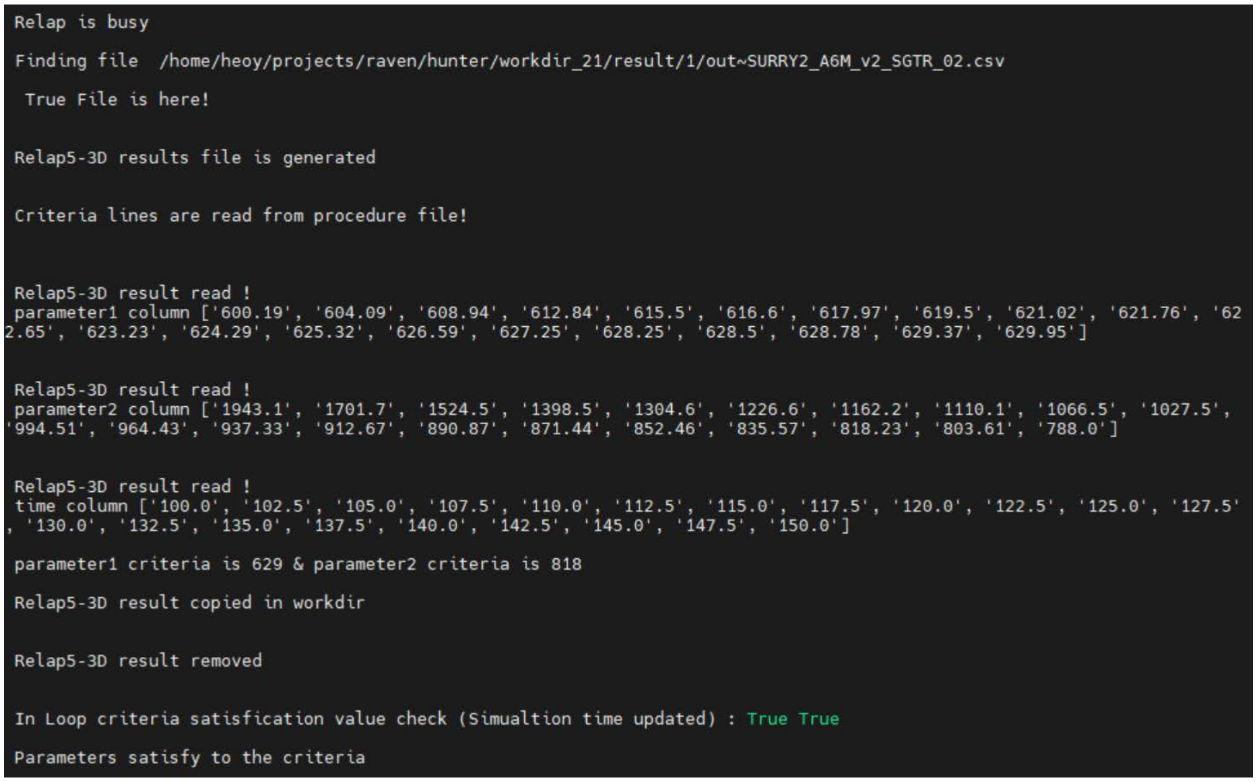
Next, it reads the generated RELAP5-3D result file and finds the criteria designated as flags in the procedure file. In Figure 25, two target parameters and a time variable are read and shown. The frequency of the parameter is logged at 2.5-second intervals as defined in the RELAP5-3D input deck. Then, the process of checking whether the variable of interest satisfies the criteria is performed. If it is not satisfied, it shows a False result and exits the loop. The simulation time must be increased to check the parameter change, so the simulation time is automatically updated, and **RELAP5-3D** performs the plant simulation again. In addition, as shown in Figure 25, the RELAP5-3D file is copied to the working directory for repeated calculation in the loop.



```
 Simulation Time is extended for 150.0 (s)


 Simulation time variable is updated

 RAVEN input file is generated


 Simulation time in Raven is updated

Relap5-3D input is copied for the next update
CONDA
raven_libraries          /home/heoy/.conda/envs/raven_libraries
Running RAVEN in "opt" mode.
Loading plugin "ExamplePlugin" at /home/heoy/projects/raven/plugins/ExamplePlugin
 ... successfully imported "ExamplePlugin" ...
Unable to load "memoryProfile" decorator; replacing with passthrough ...
Finding Prescient failed with The 'prescient' distribution was not found and is required by the application

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

      .---.        .------######       #####     ###   ###  ########  ###    ###
     /     \  __  /    --###  ###    ###  ###   ###   ###  ###       #####  ###
    / /     \(  )/    --###  ###    ###   ###  ###   ###  ######    ### ######
   //////   ' \/ `   --#######    #########  ###   ###  ###      ###  #####
  //// / // :    :   -###   ###  ###   ###   ######    ####      ###   ####
 // /   /  /`    '---###    ###  ###   ###    ###    ########  ###    ###
//          //..\
===========UU====UU=============================================================
           '//||\`
             ''``
```

Figure 26. Screenshot of running codes (3/4).

In this arbitrary example, we set the simulation time to increase by 10 seconds. Figure 26 shows that the new simulation clock is set to 150 seconds. RAVEN is re-run, which, in turn, runs RELAP5-3D. As if starting from the loop again, it modifies the RAVEN input file, changes the plant simulation simultaneously, and monitors the completion of the simulation to read the result file. After that, it is compared with the criteria again. If the target plant parameter in the loop satisfies the criteria, it exits the loop, ends the calculation, and saves the result. This process is shown in Figure 27.



```
Relap is busy

Finding file  /home/heoy/projects/raven/hunter/workdir_21/result/1/out~SURRY2_A6M_v2_SGTR_02.csv

 True File is here!


Relap5-3D results file is generated


Criteria lines are read from procedure file!



Relap5-3D result read !
parameter1 column ['600.19', '604.09', '608.94', '612.84', '615.5', '616.6', '617.97', '619.5', '621.02', '621.76', '62
2.65', '623.23', '624.29', '625.32', '626.59', '627.25', '628.25', '628.5', '628.78', '629.37', '629.95']


Relap5-3D result read !
parameter2 column ['1943.1', '1701.7', '1524.5', '1398.5', '1304.6', '1226.6', '1162.2', '1110.1', '1066.5', '1027.5',
'994.51', '964.43', '937.33', '912.67', '890.87', '871.44', '852.46', '835.57', '818.23', '803.61', '788.0']


Relap5-3D result read !
time column ['100.0', '102.5', '105.0', '107.5', '110.0', '112.5', '115.0', '117.5', '120.0', '122.5', '125.0', '127.5'
, '130.0', '132.5', '135.0', '137.5', '140.0', '142.5', '145.0', '147.5', '150.0']

parameter1 criteria is 629 & parameter2 criteria is 818

Relap5-3D result copied in workdir


Relap5-3D result removed


In Loop criteria satisfication value check (Simualtion time updated) : True True

Parameters satisfy to the criteria
```

Figure 27. Screenshot of running codes (4/4).

The coupling between HUNTER and RELAP5-3D was successful using RAVEN as an intermediary, as shown in the above result. The example demonstrates how the virtual operator is driving the plant through controls and monitoring. This process is simply scaled up when following a larger procedure, such as for the SGTR scenario covered later in this report.

## 6.5 Discussion

### 6.5.1 Limitations

As described in the above proof of concept, we successfully coupled HUNTER with RELAP5-3D. However, there are two limitations to this approach. First, if we have a RELAP5-3D plant model, we can simulate a variety of plants, but a RELAP5-3D code implementation is needed to represent operator manipulation accordingly. According to the operator's manipulations, this requires additional effort by the analyst for instantiating the initiating event in RELAP5-3D. Most RELAP5-3D models are simplified codes that do not represent the full spectrum of human actions.

Second, we need an additional code for deriving plant information required outside the plant response for HEP calculation or PSF determination in HUNTER. In many cases, a variable such as available time should be derived. As described in Chapter 5, Available time is a function of how long an action takes relative to how long is available for that action to be performed before plant conditions degrade. Available time may change according to the plant response when the operator's manipulation is performed. Therefore, simple plant simulation and multiple simulations through sampling must be performed simultaneously to determine the relevant variables. The operator actions take so-and-so long, depending on the plant evolution, and the plant evolution is a product of the operator actions. To derive available time, additional batch mode coupling should be developed using the between HUNTER, RAVEN, and RELAP5-3D.

### 6.5.2 Future Work

A challenge of using RELAP5-3D is the need to couple synchronously between the virtual operator in HUNTER and the thermal hydraulics in RELAP5-3D. This coupling has been demonstrated, but it represents linking several different pieces of code to achieve this functionality. For example, RELAP5-3D is fundamentally designed to run in batch mode, independent of starting and stopping. HUNTER makes use of RAVEN as a mediator, whereby RAVEN creates input decks for RELAP5-3D and outputs log files for use by HUNTER. RAVEN handles the starting and stopping, employing reruns with new configurations in response to operator manipulations. This is accomplished against a backdrop of RELAP5-3D being run by RAVEN in Monte Carlo repetitions according to a predefined distribution at the time of initiation. Such functionality is duplicative to the simulation run features built into HUNTER.

RELAP5-3D exists as variants optimized for real-time dynamics with operator inputs. For example, simulator vendor GSE Systems' RELAP5-HD provides real-time interactivity with RELAP5-3D (GSE Systems 2012). RELAP5-HD provides the backend for synchronizing plant parameters with real human operator actions, thus enabling a full-scope simulator for training purposes. A wide variety of human actions are included as initiating events in the plant models. Additionally, recent simulator implementations feature APIs to allow real-time monitoring and control interjection from third-party software applications like human-machine interfaces or, by extension, software such as HUNTER. Such functionality is not native to RELAP5-3D, which complicates efforts to model realistic human-centered scenarios in HUNTER.

Future work should seek to bring such functionality into the HUNTER-RELAP5-3D interface or to couple HUNTER specifically with software simulations like RELAP5-HD. Tools like RELAP5-HD may be more readily available at utilities where HRA is performed, enabling easier adoption of HUNTER in industry. A tradeoff of using a customized version of RELAP5-3D optimized for human integration is that the temporal synchronization may preclude faster-than-real-time analysis. Future work will strive to determine the best method for coupling HUNTER to the Environment module and provide guidance for real-time vs. faster-than-real-time applications of dynamic HRA.

# 7. GRAPHICAL USER INTERFACE

## 7.1 Introduction

A major goal of HUNTER is to function as an easy-to-use dynamic HRA tool. One barrier to the adoption of dynamic tools is the complexity of using them. The original HUNTER software was command-line based, and the information input/output was text-based through a terminal window. In addition, each module was written as separate code. This required running independent codes for the segmented functions, providing input manually, and manipulating the modules via command line. To simplify the analysis process, it is necessary to implement software that integrates the fragmented code into one wrapper and provides a user-friendly interface. To that end, we created a graphical user interface (GUI) to ease modeling functions. This GUI also links previously disparate pieces of code into a single standalone software framework.

The essential functions included in the HUNTER interface include:

1. Ability to create new models
2. Ability to load existing models
3. Ability to edit models according to HUNTER functions such as PSFs

4. Ability to save models
5. Ability to run the models with different modeling configurations
6. Ability to view the results of model runs
7. Ability to save and export model runs.

HUNTER's interface functionality is expected to evolve and mature with each successive version of the HUNTER software code. The current version captures the above functionality but does not implement a GUI for each different element. Elements 1–4, which correspond to model setup, feature a convenient GUI in which all functions can be performed using simple mouse clicks and inputs. Elements 5–7, which correspond to model runs, still largely make use of the command-line structure native to the underlying Python development environment. As additional use cases are identified and new models are created in HUNTER, it is expected that additional GUI functionality will be developed to support Elements 5–7.

The remainder of this section uses example screen shots to describe and explain the GUI. Because HUNTER was developed in Python, it is cross compatible with different operating systems.[f] The screens depicted in this section were captured from HUNTER as it is being operated using Microsoft Windows.

## 7.2 Starting Pages

### 7.2.1 Splash Screen

After the HUNTER software package is installed, the HUNTER application is started either by inputting `execute HUNTER.py` into the command line in Python, or by double-clicking on the HUNTER icon (see Figure 28). After launching the software, a splash screen pops up (see Figure 29) and remains until the HUNTER application is closed by clicking the appropriate close-window button in the titlebar, initiating a new HUNTER model, or opening an existing one.

Figure 28. HUNTER desktop icon.

[f] A limiting factor regarding HUNTER's cross-platform compatibility is the operating system used for the linked plant simulation code for the Environment module.

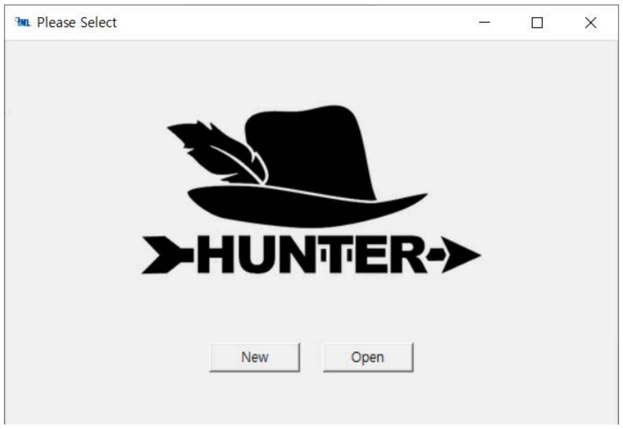
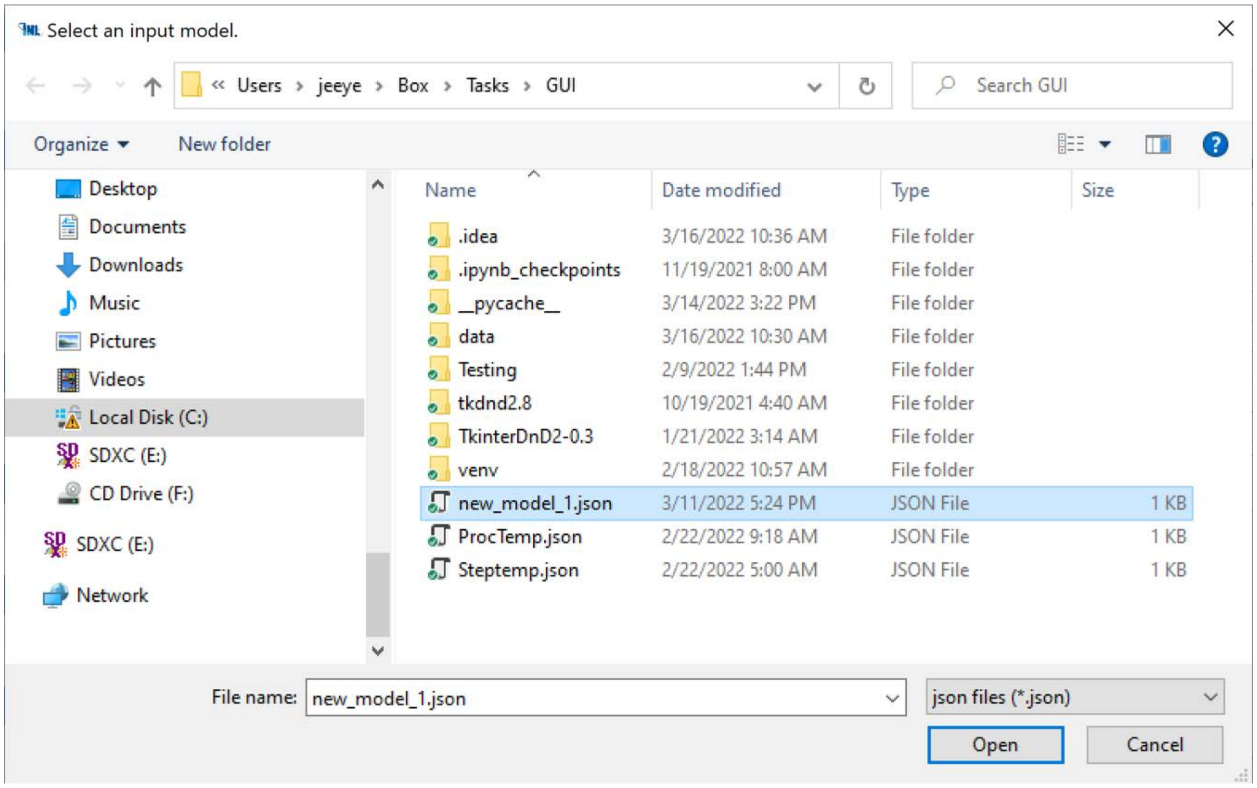


Figure 29. HUNTER splash screen.

### 7.2.2 New or Existing Models

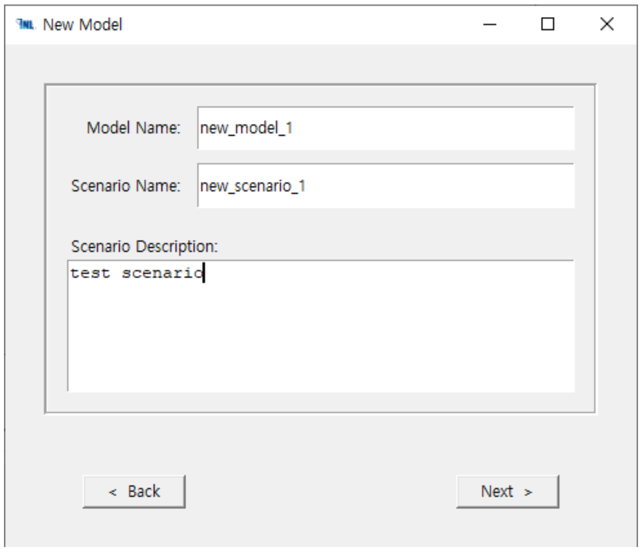
HUNTER scenarios are grouped into models, which link all the subsets files (e.g., procedures). If the user chooses to open an existing model, a dialog box appears, as shown in Figure 30, enabling the user to select a previously created model file. Models are stored as JSON files.



| Name | Date modified | Type | Size |
|---|---|---|---|
| .idea | 3/16/2022 10:36 AM | File folder | |
| .ipynb_checkpoints | 11/19/2021 8:00 AM | File folder | |
| __pycache__ | 3/14/2022 3:22 PM | File folder | |
| data | 3/16/2022 10:30 AM | File folder | |
| Testing | 2/9/2022 1:44 PM | File folder | |
| tkdnd2.8 | 10/19/2021 4:40 AM | File folder | |
| TkinterDnD2-0.3 | 1/21/2022 3:14 AM | File folder | |
| venv | 2/18/2022 10:57 AM | File folder | |
| new_model_1.json | 3/11/2022 5:24 PM | JSON File | 1 KB |
| ProcTemp.json | 2/22/2022 9:18 AM | JSON File | 1 KB |
| Steptemp.json | 2/22/2022 5:00 AM | JSON File | 1 KB |

File name: new_model_1.json

Figure 30. HUNTER file dialog for loading an existing model.

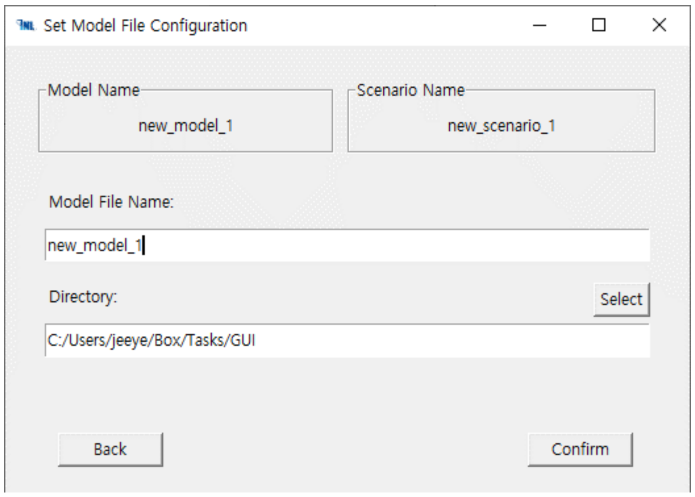


Model Name: new_model_1

Scenario Name: new_scenario_1

Scenario Description: test scenario

Figure 31. HUNTER screen for creating a new model.



Model Name: new_model_1

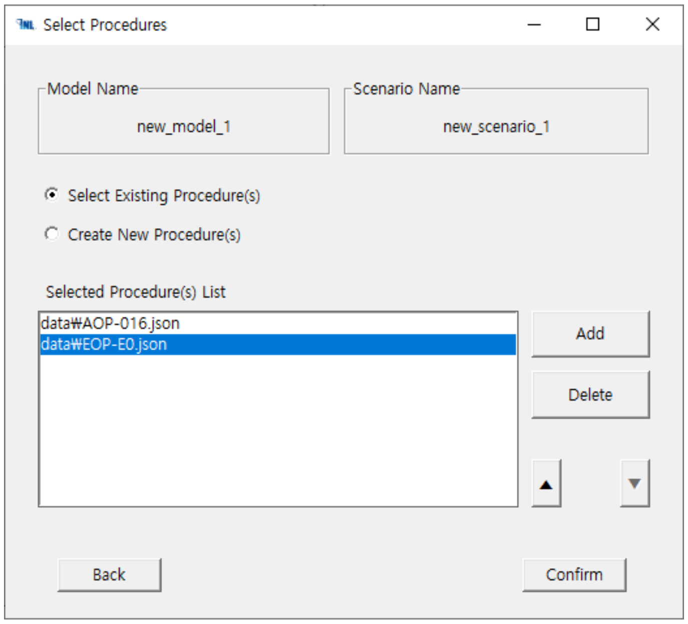
Scenario Name: new_scenario_1

Model File Name: new_model_1

Directory: C:/Users/jeeye/Box/Tasks/GUI

Figure 32. HUNTER dialog window for specifying a model file location.

If the user chooses to create a model, a window for entering the model name, scenario name, and description appears, as shown in Figure 31. A follow-on screen (see Figure 32) allows the user to specify the location of the model files so they can be saved for later reuse. After entering this required information and clicking onthe Next button, the following window (see Figure 33) allows the user to add or delete existing procedures from the model, or to create new models. Figure 34 shows the dialog box for selecting procedure files to add to the model. Procedure files are also stored as JSON files. Multiple procedure files can be selected at once, and the order of the selected procedure files can be changed (using the arrow buttons) or deleted (using the Delete button).


Select Procedures

Model Name: new_model_1

Scenario Name: new_scenario_1

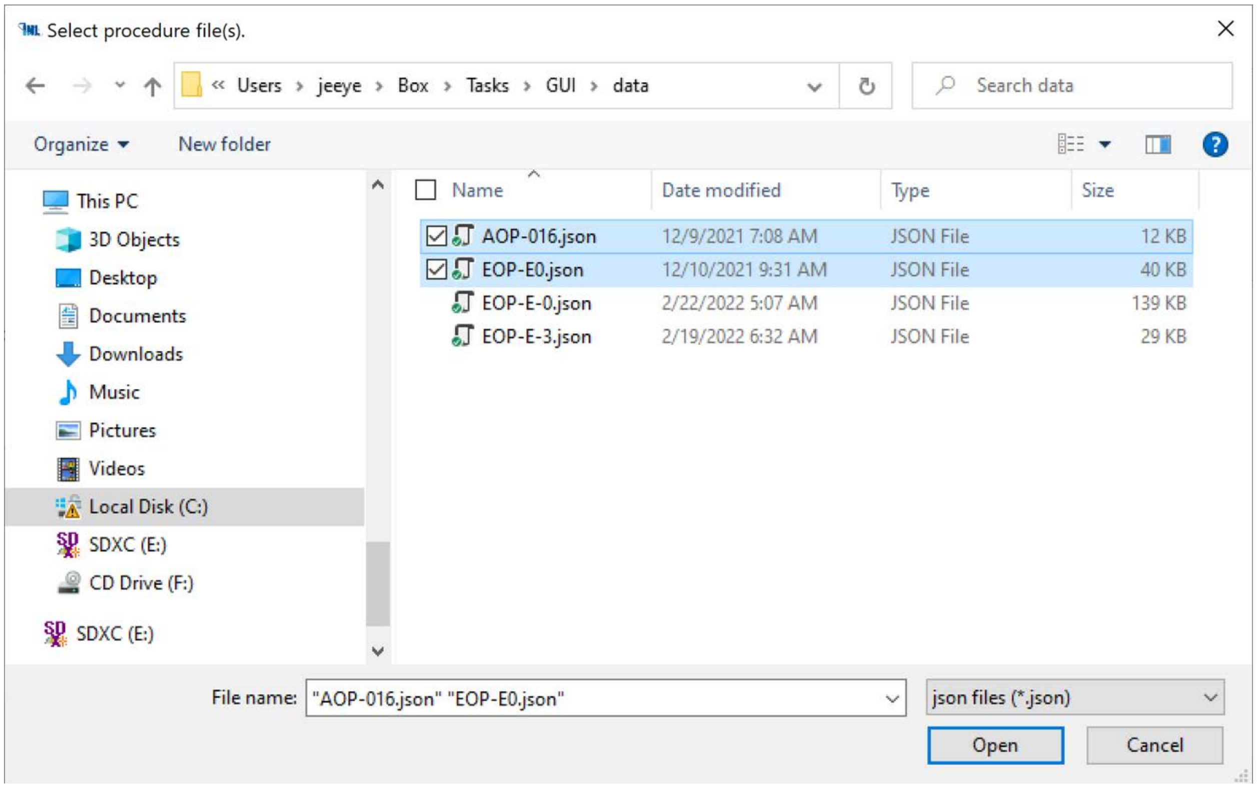
Select Existing Procedure(s) [✓]

Create New Procedure(s)

Selected Procedure(s) List

data₩AOP-016.json
data₩EOP-E0.json

Figure 33. HUNTER dialog window for adding, creating, or deleting existing procedures from the model.


Select procedure file(s).

« Users › jeeye › Box › Tasks › GUI › data

| Name | Date modified | Type | Size |
|---|---|---|---|
| ✓ AOP-016.json | 12/9/2021 7:08 AM | JSON File | 12 KB |
| ✓ EOP-E0.json | 12/10/2021 9:31 AM | JSON File | 40 KB |
| EOP-E-0.json | 2/22/2022 5:07 AM | JSON File | 139 KB |
| EOP-E-3.json | 2/19/2022 6:32 AM | JSON File | 29 KB |

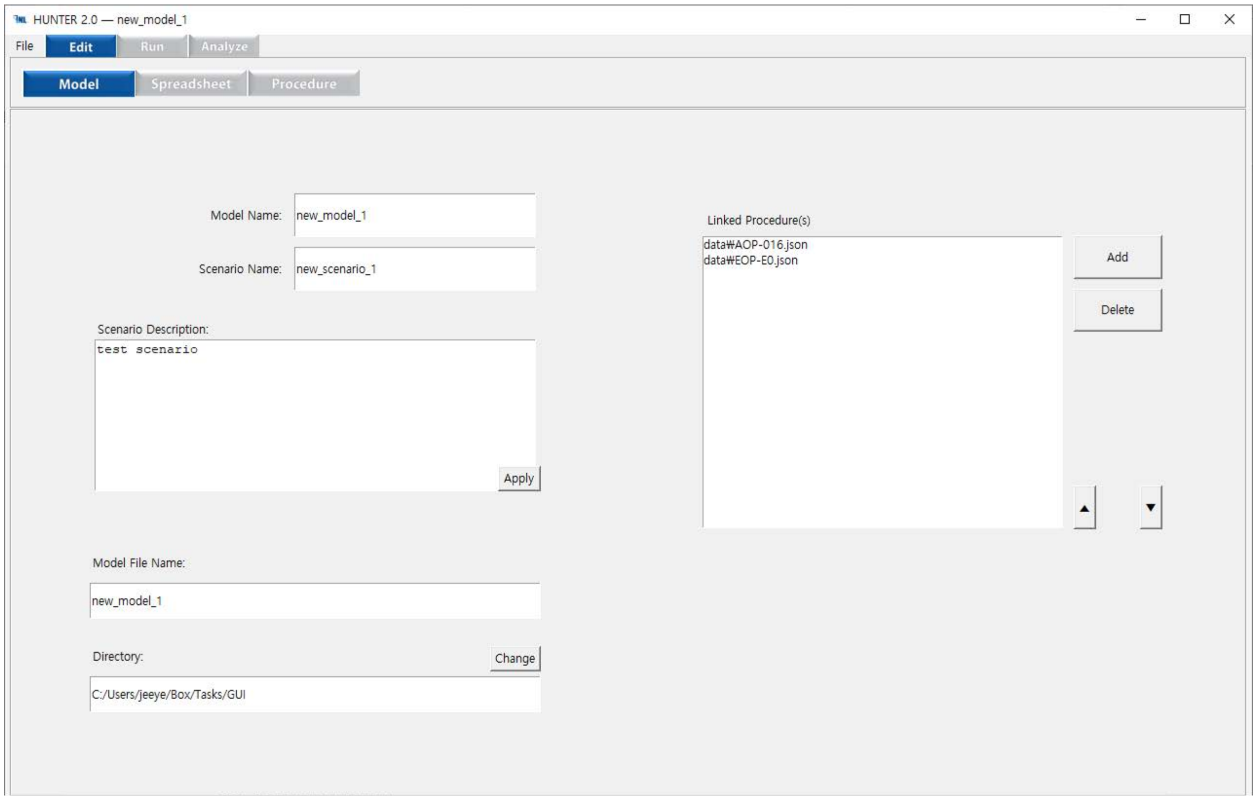
File name: "AOP-016.json" "EOP-E0.json"

json files (*.json)

Figure 34. HUNTER file dialog window for adding procedures.

If the user opts to manually create a new procedure, they are prompted to name the procedure and are then taken to the procedure window (described in the next section), where they can build the new procedure in step-by-step fashion.

## 7.3 Main Edit Window



Model Name: new_model_1

Scenario Name: new_scenario_1

Scenario Description:

```
test scenario
```

Linked Procedure(s)

data₩AOP-016.json
data₩EOP-E0.json

Model File Name:

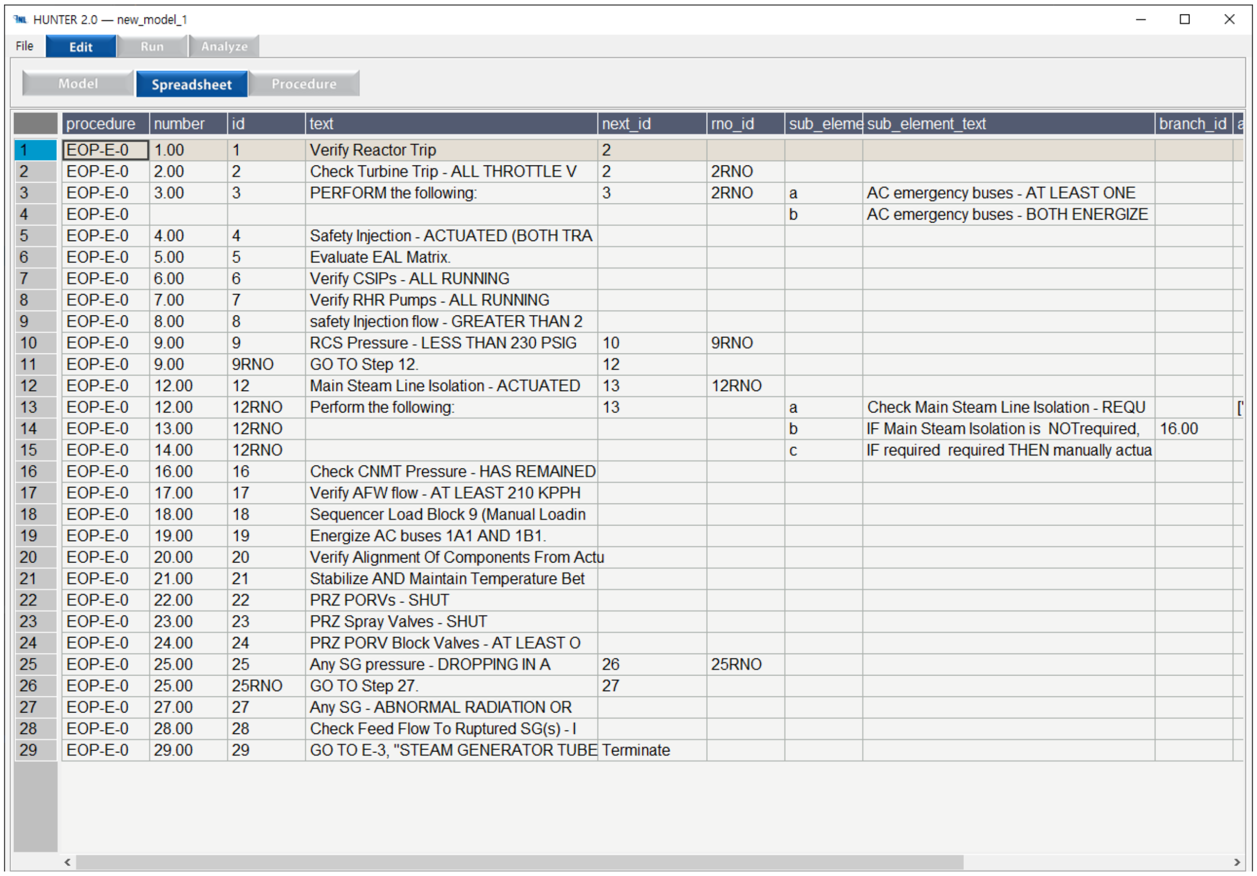
new_model_1

Directory:

C:/Users/jeeye/Box/Tasks/GUI

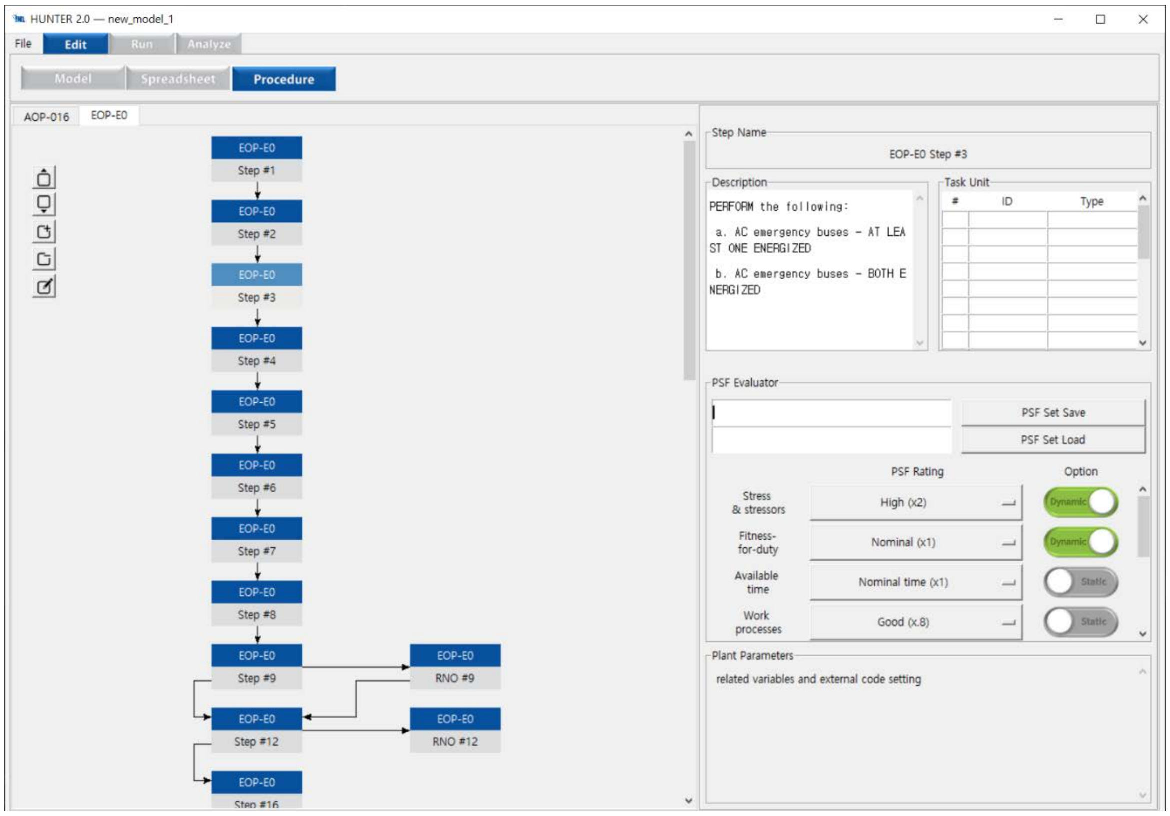
Figure 35. HUNTER model window view.

HUNTER has three main modes: Edit, Run, and Analyze, as denoted by selectable items in the menu bar (see Figure 35). The currently displayed item is highlighted with a blue background. Currently unavailable options have a dark gray background. A fourth option (i.e., the File menu) is always available and allows the user to load, save, and print models. Edit mode allows the user to define and edit different elements of the HUNTER model, such that the end user should be able to create complete HUNTER models from within these windows. Within Edit mode are three windows: Model, Spreadsheet, and Procedure. These are represented as buttons (the Model window is shown in Figure 35). When the initial setup is finished, the model settings are imported to the Model window, where the user can see and change them. The Spreadsheet window (see Figure 36) gives an overall view that can be used to check and change all the input values for HUNTER analysis at once. The Spreadsheet window gives the user a full view of the entire HUNTER taxonomy and is recommended only for power users. Lastly, there is the Procedure window, as seen in Figure 37.



| | procedure | number | id | text | next_id | rno_id | sub_eleme | sub_element_text | branch_id | a |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | EOP-E-0 | 1.00 | 1 | Verify Reactor Trip | 2 | | | | | |
| 2 | EOP-E-0 | 2.00 | 2 | Check Turbine Trip - ALL THROTTLE V | 2 | 2RNO | | | | |
| 3 | EOP-E-0 | 3.00 | 3 | PERFORM the following: | 3 | 2RNO | a | AC emergency buses - AT LEAST ONE | | |
| 4 | EOP-E-0 | | | | | | b | AC emergency buses - BOTH ENERGIZE | | |
| 5 | EOP-E-0 | 4.00 | 4 | Safety Injection - ACTUATED (BOTH TRA | | | | | | |
| 6 | EOP-E-0 | 5.00 | 5 | Evaluate EAL Matrix. | | | | | | |
| 7 | EOP-E-0 | 6.00 | 6 | Verify CSIPs - ALL RUNNING | | | | | | |
| 8 | EOP-E-0 | 7.00 | 7 | Verify RHR Pumps - ALL RUNNING | | | | | | |
| 9 | EOP-E-0 | 8.00 | 8 | safety Injection flow - GREATER THAN 2 | | | | | | |
| 10 | EOP-E-0 | 9.00 | 9 | RCS Pressure - LESS THAN 230 PSIG | 10 | 9RNO | | | | |
| 11 | EOP-E-0 | 9.00 | 9RNO | GO TO Step 12. | 12 | | | | | |
| 12 | EOP-E-0 | 12.00 | 12 | Main Steam Line Isolation - ACTUATED | 13 | 12RNO | | | | |
| 13 | EOP-E-0 | 12.00 | 12RNO | Perform the following: | 13 | | a | Check Main Steam Line Isolation - REQU | | [' |
| 14 | EOP-E-0 | 13.00 | 12RNO | | | | b | IF Main Steam Isolation is NOTrequired, | 16.00 | |
| 15 | EOP-E-0 | 14.00 | 12RNO | | | | c | IF required required THEN manually actua | | |
| 16 | EOP-E-0 | 16.00 | 16 | Check CNMT Pressure - HAS REMAINED | | | | | | |
| 17 | EOP-E-0 | 17.00 | 17 | Verify AFW flow - AT LEAST 210 KPPH | | | | | | |
| 18 | EOP-E-0 | 18.00 | 18 | Sequencer Load Block 9 (Manual Loadin | | | | | | |
| 19 | EOP-E-0 | 19.00 | 19 | Energize AC buses 1A1 AND 1B1. | | | | | | |
| 20 | EOP-E-0 | 20.00 | 20 | Verify Alignment Of Components From Actu | | | | | | |
| 21 | EOP-E-0 | 21.00 | 21 | Stabilize AND Maintain Temperature Bet | | | | | | |
| 22 | EOP-E-0 | 22.00 | 22 | PRZ PORVs - SHUT | | | | | | |
| 23 | EOP-E-0 | 23.00 | 23 | PRZ Spray Valves - SHUT | | | | | | |
| 24 | EOP-E-0 | 24.00 | 24 | PRZ PORV Block Valves - AT LEAST O | | | | | | |
| 25 | EOP-E-0 | 25.00 | 25 | Any SG pressure - DROPPING IN A | 26 | 25RNO | | | | |
| 26 | EOP-E-0 | 25.00 | 25RNO | GO TO Step 27. | 27 | | | | | |
| 27 | EOP-E-0 | 27.00 | 27 | Any SG - ABNORMAL RADIATION OR | | | | | | |
| 28 | EOP-E-0 | 28.00 | 28 | Check Feed Flow To Ruptured SG(s) - I | | | | | | |
| 29 | EOP-E-0 | 29.00 | 29 | GO TO E-3, "STEAM GENERATOR TUBE | Terminate | | | | | |

Figure 36. HUNTER Spreadsheet window view.



Step Name: EOP-E0 Step #3

Description:

```
PERFORM the following:

a. AC emergency buses - AT LEA
ST ONE ENERGIZED

b. AC emergency buses - BOTH E
NERGIZED
```

Task Unit

| # | ID | Type |
|---|---|---|

PSF Evaluator

| | PSF Rating | Option |
|---|---|---|
| Stress & stressors | High (x2) | Dynamic |
| Fitness-for-duty | Nominal (x1) | Dynamic |
| Available time | Nominal time (x1) | Static |
| Work processes | Good (x.8) | Static |

Plant Parameters

related variables and external code setting

Figure 37. HUNTER Procedure window view.

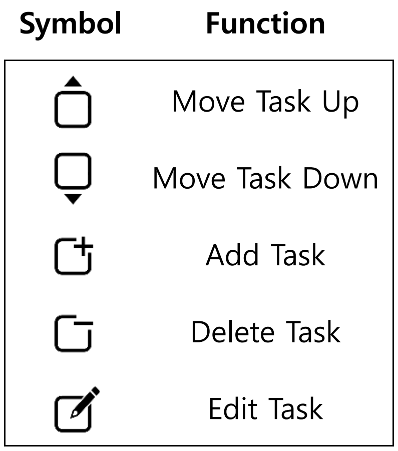
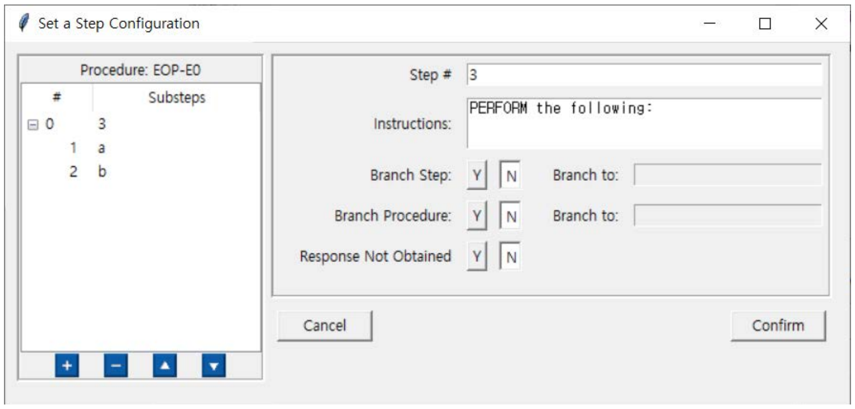

Symbol
Function
Move Task Up
Move Task Down
Add Task
Delete Task
Edit Task

Figure 38. HUNTER procedure editing tools.

The procedure window visualizes, in procedural step units, the procedure relevant to the model's scenario, and offers an interface that helps to easily define procedure step attributes. The procedure window consists of the graphical procedure model area (on the left) and the input setting panel (on the right). The left-side graphical area includes a tab for each procedure, the name of which is displayed at the top. Each tab employs a flowchart-like format to visualize all the steps of the procedure. If the user needs to change the contents or the order of the steps, they can edit the steps via the button panel at the top left (see Figure 38 for a legend).

Each step box in this diagram can be clicked on to highlight the selected step. Information on the selected step is displayed in the panel on the right. The user can select a step box and then press the or button on the left to move the step up or down in the step order. To change the contents of the step, the user can use the pop-up window that appears by pressing the button. The user can also select a step and then press the button to delete it. However, due to step linkages, the user can only delete RNO steps via the window that pops up by pressing the edit button (), and not by pressing the delete button ().

A user can add a new step by pressing the button. A step configuration pop-up window appears, and the settings inputted to that window are used to create a new step. Each newly created step is, by default, positioned last in the procedure, but the order can be changed by using the step move buttons.


Set a Step Configuration

Procedure: EOP-E0

| # | Substeps |
|---|---|
| 0 | 3 |
| 1 | a |
| 2 | b |

Step # 3

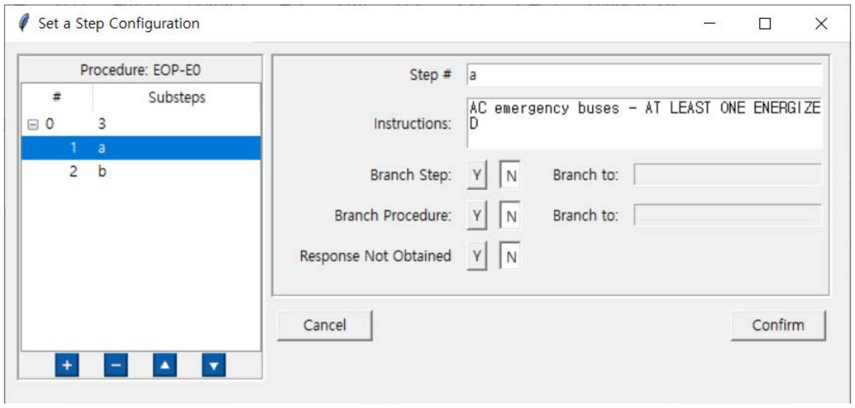
Instructions: PERFORM the following:

Branch Step: Y N Branch to:

Branch Procedure: Y N Branch to:

Response Not Obtained Y N

Figure 39. HUNTER procedure step configuration editor: normal step.


Set a Step Configuration

Procedure: EOP-E0

| # | Substeps |
|---|---|
| 0 | 3 |
| 1 | a |
| 2 | b |

Step # a

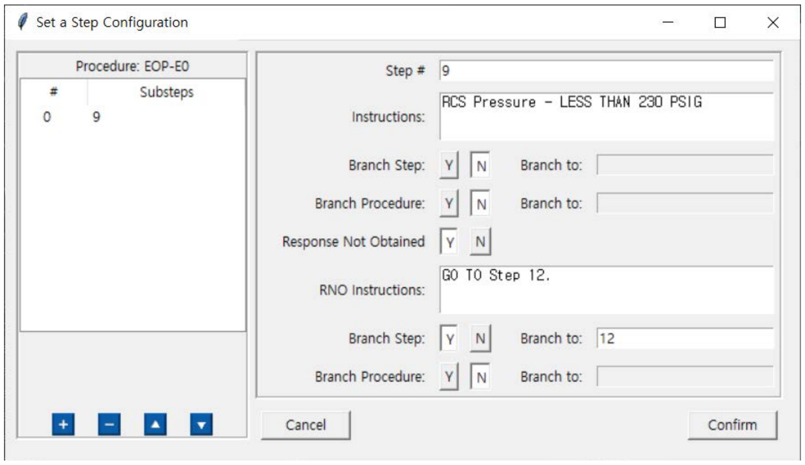
Instructions: AC emergency buses - AT LEAST ONE ENERGIZED

Branch Step: Y N Branch to:

Branch Procedure: Y N Branch to:

Response Not Obtained Y N

Figure 40. HUNTER procedure step configuration editor: substep.



Step # 9

Instructions: RCS Pressure - LESS THAN 230 PSIG

Branch Step: Y N Branch to:

Branch Procedure: Y N Branch to:

Response Not Obtained Y N

RNO Instructions: GO TO Step 12.

Branch Step: Y N Branch to: 12

Branch Procedure: Y N Branch to:

Figure 41. HUNTER procedure step configuration editor: Response not obtained.

The window in Figure 39 pops up when the edit or add buttons are clicked. The user can add a substep by clicking the + button at the bottom of the window (see Figure 40). To make a substep of a substep, the user selects the parent substep from the left-hand list box and presses +. To delete a specific substep, the user selects the target substep in the left-hand list box and presses -. The user can operate the up or down buttons to change the substep order. Information on the selected step/substep is displayed on the right. When the edit button is pressed, information on the selected step is displayed. If the corresponding step is RNO (see Figure 41), the user can check the N button of Response Not Obtained so that a space for adding RNO information appears. To delete an existing RNO step, the user clears the RNO option via this step configuration setting window.

The Procedure window (see Figure 37) contains information relevant to all aspects of HUNTER. Figure 42 shows a wireframe version of the Procedure window, in which the various types of HUNTER information are shown in different colors depicting the different modules. Yellow areas correspond to information used for the Task module. The green area information is used for the Environment module and contains definitions of key plant parameters employed by the steps in order to monitor and control the plant simulation, when applicable. Finally, the red area information is used by the Individual module to set the PSF fields. Note that each of these areas is linked to a specific procedure step or substep. Currently, there is no ready way of using the GUI functionality to set global parameters (e.g., shared PSF assignments) for all steps in the procedure. That functionality is best achieved by using the Spreadsheet window.

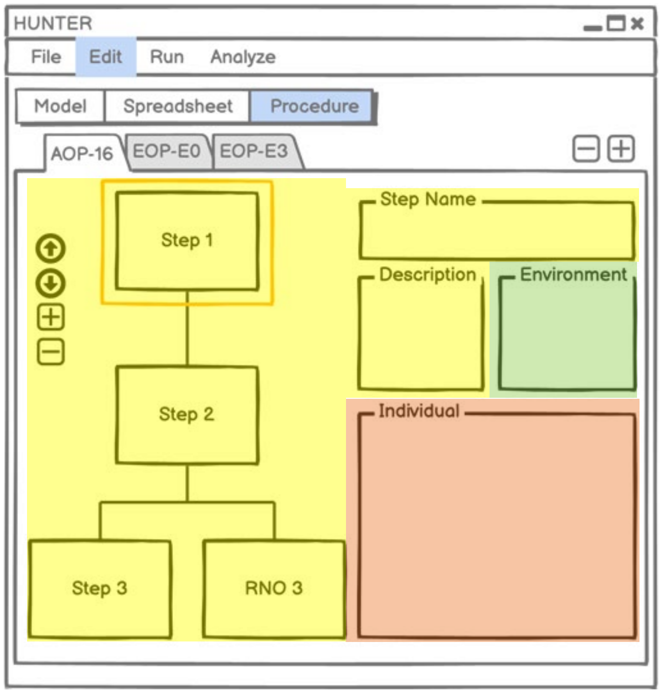


Figure 42. Wireframe rendering of the HUNTER procedure window, with color-coded sections depicting Task (yellow), Environment (green), and Individual (red) module information

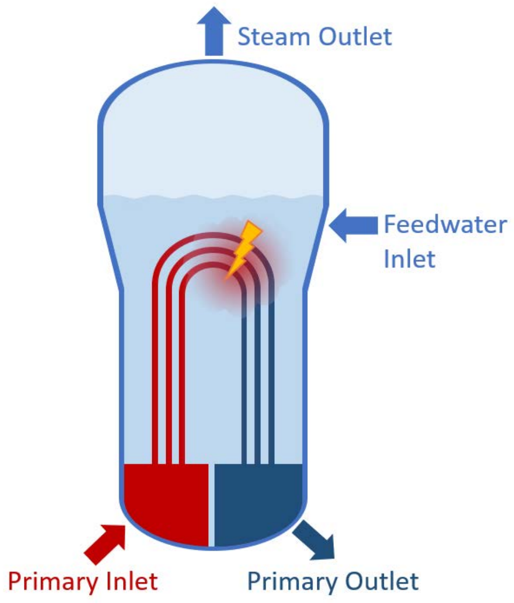


Figure 43. Steam generator tube rupture.

# 8. DEMONSTRATION AND FINDINGS

## 8.1 SGTR Scenario

### 8.1.1 SGTR Description

An SGTR is portrayed in Figure 43. An SGTR is one of design-basis accidents (DBAs) in which one or more heat transfer tubes are broken such that the coolant from the primary side leaks to the secondary side. In pressurized-water reactors, steam generators form the primary and secondary boundary. They serve to transfer the heat generated in the primary system to the secondary system. The primary coolant, which potentially contains radioactive materials, is separated from the secondary side water supply through the U-tubes of the steam generators, thus preventing radioactive leakage into the environment. The primary and secondary system boundary can be kept limited to the steam generators during a leak or rupture if appropriate action is taken by operators in a timely manner. To this end, operators must identify and isolate the damaged or ruptured steam generator to minimize the possibility of further radiation leakage. To successfully mitigate the SGTR accident, secondary heat removal and the depressurization of the reactor coolant system (RCS) are required, and the operation of the high-pressure safety injection system might also be required.

"Success" in responding to the SGTR scenario is to intervene at the right time to limit the release of radioactive material and prevent core damage. In this regard, there are generally four main tasks that operators must perform in SGTR scenarios: identify and isolate the damaged steam generator(s), perform RCS decompression using steam evacuation or pressurized spray and a pressure operated relief valve (PORV), terminate safety injection (SI) according to SI termination requirements, and enter into long-term cooling cycle operation. If cooling using the secondary side is not possible due to the inability to use the secondary feedwater system and auxiliary feedwater system, the RCS is cooled and depressurized through a feed and bleed (FB) operation.

From a PRA point of view, events to respond to the initial event can be classified according to its irreversible branching, which is expressed as headings in an event tree (ET). Figure 44 shows a generic ET for SGTR. HFEs—referring to those opportunities for humans to disrupt the successful function of the plant—can be defined from the ET of the SGTR. Possible HFEs can be defined by analyzing the effect of operator intervention and failure in the event shown in ET. Table 18 shows examples of the possible HFEs, although specific modeling scenarios and specific plant PRAs may highlight different HFEs than this list.

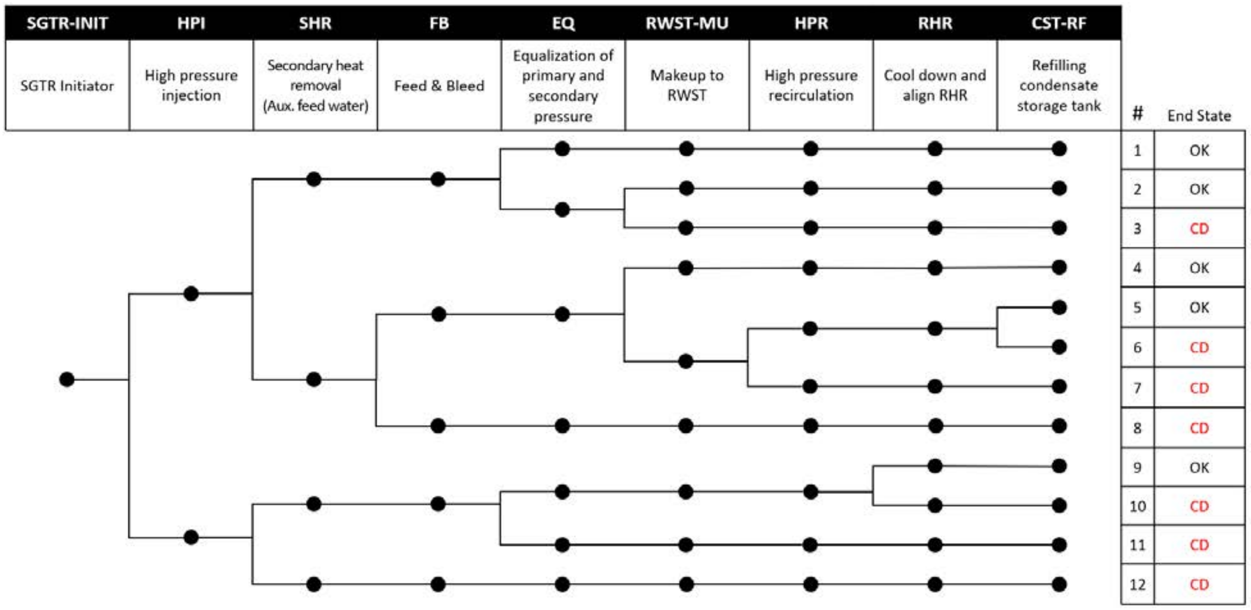


Figure 44. Example of generic Level 1 PRA ET for SGTR (adapted from NUREG-2195).

The following headings define the top branches for the event tree in Figure 44:

- SGTR-INIT: Initiating event induced SGTR from DBA events
- HPI: High-pressure injection (HPI) systems, both SI pumps and charging pumps, if applicable
- SHR: Secondary heat removal (SHR) system, either main feedwater (MFW) or auxiliary feedwater (AFW)
- FB: Feed and bleed operation and the supporting relief path
- EQ: Operator actions for equalization of primary and secondary pressure, which involves depressurization and control of primary pressure
- RWST-MU: Long-term makeup water to the reactor water storage tank (RWST)
- HPR: High-pressure recirculation (HPR) and the associated operator action
- RHR: Operator action to cool down to cold shutdown and align the residual heat removal (RHR) system
- CST-RF: Operator action based on refilling the condensate storage tank (CST) for long-term secondary side cooling using the AFW system

Table 18. HFEs defined from the ET of the SGTR scenario.

| HFE | HFE Description |
|---|---|
| 1 | Operator fails to respond with no reactor protection system (RPS) signal present. |
| 2 | Operator fails to respond with RPS signal present. |
| 3 | Operator fails to start and align SHR system. |
| 4 | Operator fails to identify SGTR and implement procedures. |
| 5 | Operator fails to isolate faulted steam generator(s). |
| 6 | Operator fails to depress RCS via secondary side cooling (SSC). |
| 7 | Operator fails to control/terminate SI. |
| 8 | Operator fails to initiate FB cooling. |
| 9 | Operator fails to refill RWST. |
| 10 | Operator fails to start recirculation mode. |
| 11 | Operator fails to refill CST. |

Table 19. Mapping of HFEs to Procedures.

| HFE | HFE Description | Procedure Mapping |
|---|---|---|
| 1 | Operator fails to respond with no RPS signal present. | Step #1 in EOP E-0, "Reactor Trip or Safety Injection"<br>Steps in FR-S. 1, "Response to Nuclear Power Generation/ATWS" |
| 2 | Operator fails to respond with RPS signal present. | |
| 3 | Operator fails to start and align SHR system. | Step #17 in EOP E-0, "Reactor Trip or Safety Injection" |
| 4 | Operator fails to identify SGTR and implement procedures. | Step #25 ~ #29 in EOP E-0, "Reactor Trip or Safety Injection" |
| 5 | Operator fails to isolate faulted steam generators (SGs). | Step #4 ~ #19 in EOP E-3, "Stream Generator Tube Rupture" |
| 6 | Operator fails to depress RCS via SSC. | Step #29 ~ #35 in EOP E-3, "Stream Generator Tube Rupture" |
| 7 | Operator fails to control/terminate SI. | Step #75 ~ #76 in EOP E-3, "Stream Generator Tube Rupture" |
| 8 | Operator fails to initiate FB cooling. | Steps in FR-S. 1, "Response to Nuclear Power Generation/ATWS" |
| 9 | Operator fails to refill RWST. | Foldout in EOP E-3, "Stream Generator Tube Rupture"<br>Steps in ES 1.3, "Transfer to Cold Leg Recirculation" |
| 10 | Operator fails to start recirculation mode. | Steps in ES 1.3, "Transfer to Cold Leg Recirculation" |
| 11 | Operator fails to refill CST. | Foldout in EOP E-3, "Stream Generator Tube Rupture"<br>Steps in OP-137 Section 8.1, "Auxiliary Feedwater System" |

### 8.1.2 SGTR Procedures

This section describes procedure steps related to HFEs defined in the SGTR scenario. This study considered the procedures for GSE Systems' Generic Pressurized Water Reactor (GPWR) for mapping the steps into each HFE. Table 19 summarizes the procedure steps per each HFE.

## 8.2 Relevant Findings

### 8.2.1 International HRA Empirical Study at Halden Reactor Project

The multilateral international HRA empirical study conducted at Halden Reactor Project (Lois et al. 2011) was designed to arrive at an empirically based understanding of the performance, strengths, and weaknesses of different HRA methods used to model human responses to accident sequences in PRAs. The empirical basis was developed through experiments performed at Halden Reactor Project's Halden Man-Machine Laboratory research simulator, with 14 licensed crews responding to accident situations similar to those modeled in PRAs. The study was divided into three phases, each covered in a separate volume of NUREG/IA-0216:

- *Phase 1*–the pilot phase consists of developing, testing, and revision the study's methodology and experimental design (Lois et al. 2011)
- *Phase 2*–consists of the comparison of HRA predictions for all human actions corresponding to SGTR (Bye et al. 2011)
- *Phase 3*–consists of the comparison of four loss-of-feedwater human actions (Dang et al. 2014).

Across Phases 2 and 3, HRA methods were compared with each other and with the empirical results in both qualitative and quantitative ways.

For Phase 2, the study included two types of scenarios for SGTR: the SGTR base scenario and the SGTR complex scenario. The SGTR base scenario consists of a rupture initiated in Steam Generator #1 to cause nearly immediate alarms of secondary side radiation and other abnormal indications/alarms. These conditions are not sufficient to cause an immediate automatic scram, but the status of the plant is degrading due to the rupture. The SGTR complex scenario is a complicated case of the base scenario with two main differences. The main differences are that the event starts off with a major steamline break with a nearly coincident SGTR in Steam Generator #1 that will cause an immediate automatic scram (and expectations that the crew will enter the EOP E-0 procedure for post-reactor-trip actions). The scenario features the autoclosure (as expected) of the main steam isolation valves in response to the steamline break along with the failure of any remaining secondary radiation indications (not immediately known nor expected by the crew) as part of the simulation design. The combination with the steamline break makes it considerably more difficult for the crew to diagnose the existence of the SGTR, especially in response the step in the EOP E-0 procedure concerning elevated radiation indications.

The specific HFEs used in the scenario are presented in Table 20. Success criteria for the events are typically determined by successfully avoiding irreversible changes to the plant state that affect the likelihood of core damage. To this end, the success criteria were determined on the basis of the expectations of the trainers for operator response in accordance with their training. The success/failure criteria included expected time windows for how long an activity was expected to take.

Table 20. HFEs for two SGTR scenarios from the Halden study.

| HFE | Descriptions | Base case | Complex case |
|---|---|---|---|
| HFE-1 | Failure to identify and isolate the ruptured SG | HFE-1A | HFE-1B |
| HFE-2 | Failure to cool down the RCS expeditiously | HFE-2A | HFE-2A |
| HFE-3 | Failure to depressurize the RCS expeditiously | HFE-3A | HFE-3B |
| HFE-4 | Failure to stop the safety injection (SI) | HFE-4A | N/A |
| HFE-5 | Failure to give a closing order to the PORV block valve | N/A | HFE-5B1<br>HFE-5B2 |

Overall, most crews successfully performed the required tasks, as would be expected for a well-trained DBA. The only challenging task proved to be the isolation of the faulted steam generator in the complex case (see Figure 45). All operators were well trained and very familiar with the base SGTR, since they participated in a period training program, which has the SGTR scenario trained twice every year. The complex SGTR scenario featured a compound fault that was not frequently trained.

The HFEs were ranked relative to their difficulty considering crew performance. The ranking process took into account: the number of "failing" crews and "near misses" for each HFE, the difficulty in operational terms, and the supplemental information provided to the HRA teams. The derived difficulty ranking of HFEs is: 5B1 > 1B > 3B > 3A > [1A, 2A, 2B] > 5B2 > 4A. In the case of HFE-2 and -3, the goal of the tasks is to control the temperature or pressure of the RCS, and as they have characteristics that change with time, they were greatly affected by the conditions at the start of the performance. When the automatic protection system was activated, it took more time to fully perform tasks, such as RCS cooldown due to its effect. In the task of decompressing the RCS, it was also observed that the timing of terminating the task occurred too early (e.g., ending without sufficiently decompressing the RCS). This is thought to be because the decompression rate was too fast, and there were several stopping conditions that had to be stopped or monitored before the decompression was completely achieved. This in turn led to slight deviations (slightly below) from the success criteria or insufficient performance.

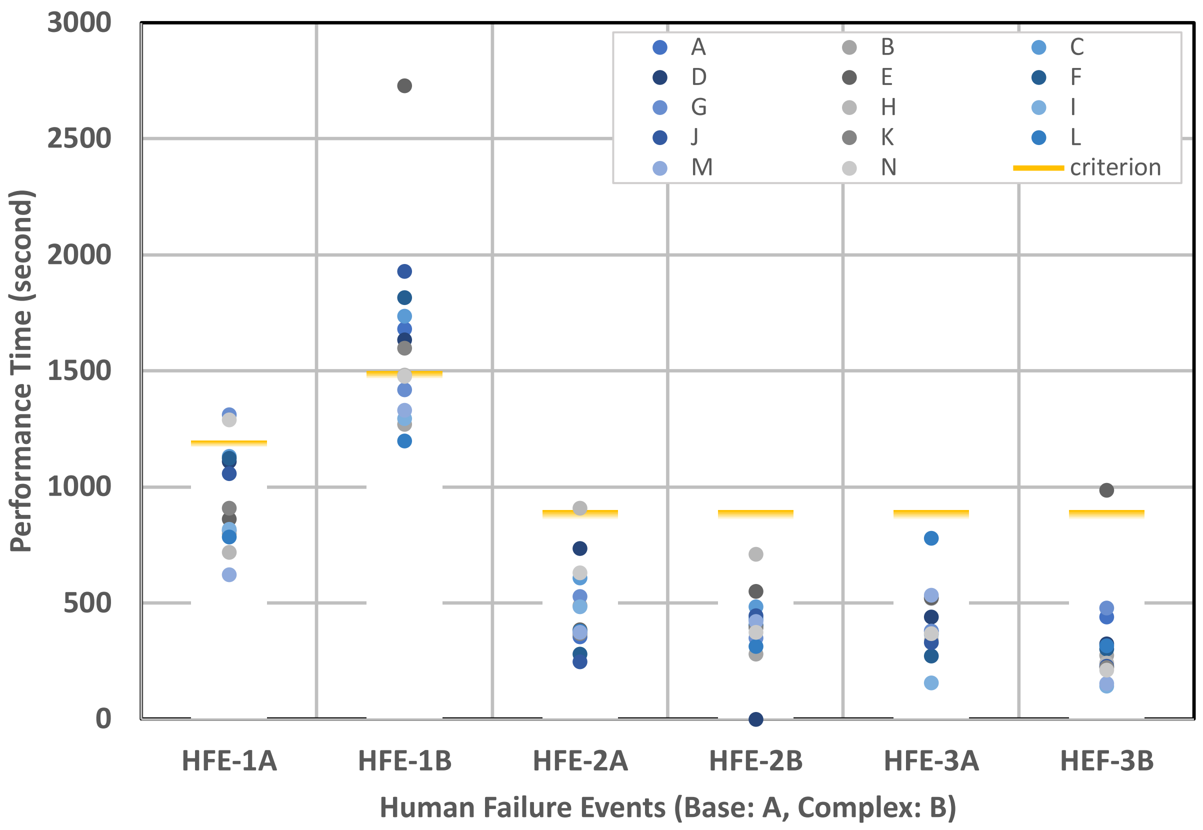


Figure 45. Performance time data for HFE-1, -2, and -3 for two SGTR cases for Crews A–N in the Halden study.

Table 21. Identified negative driving factors for HFEs in the Halden study.

| **Required Action** | **Base case** | **Complex case** |
|---|---|---|
| HFE-1<br>(Faulted SG isolation) | - Execution complexity | - Scenario complexity<br>- Procedural guidance<br>- Execution complexity<br>- Adequacy of time<br>- Work processes |
| HFE-2<br>(RCS cooldown) | - Scenario complexity<br>- Execution complexity<br>- Procedural guidance<br>- Team dynamics | - Stress<br>- Scenario complexity<br>- Execution complexity<br>- Team dynamics |
| HFE-3<br>(RCS depressurization) | - Stress<br>- Execution complexity<br>- Team dynamics | - Stress<br>- Scenario complexity<br>- Execution complexity<br>- Team dynamics |

Through the experimental results of the Halden study, the following insights can be obtained. The Halden study has three identical HFEs for base and complex SGTR scenarios, HFE-1 (faulted steam generator isolation), HFE-2 (RCS cooldown), and HFE-3 (RCS depressurization). HFE-4 and HFE-5 did not have corresponding conditions in the base and complex SGTR scenarios. The research team derived PSFs that negatively affect human performance on each HFE through qualitative analyses (see Table 21). From those results, complexity (both scenario complexity and execution complexity) acted as a negative driving factor in all cases.

Since the study used time data as one of the main sources of performance data, it seems that the relationship between PSFs and time performance can be derived from the results. Figure 45 shows the performance time of 14 crews for HEF -1, -2, and -3. Since the purpose of HFE-3 is to depressurize the RCS, crews that did not fully decompress and did not complete were excluded from this figure. In terms of a PRA analysis, the first HFE has different success criteria of performance time for the base and the complex scenario, 20 minutes and 25 minutes, respectively. The other HFEs have the same success criteria of performance time for both base and complex cases. When comparing the operator performance on HFE-1, it seems the scenario complexity makes the task take longer. For HFE-1, which allows a longer performance time window for the complex scenario, the average time to perform was 970 seconds for the base and 1614 seconds for the complex scenario. Half of the crews failed to isolate the faulted steam generator in the allowed time for the complex scenario, while most crews succeeded for the base scenario. The combination with another accident initiator (i.e., steamline break in the complex case) makes the task to identify and isolate the faulted steam generator much more complicated and difficult, as reflected in the longer time to complete the task.

However, the remaining HFEs showed the opposite effect. The performance time of HFE-2 was 512 seconds and 400 seconds, and HFE-3 took 441 seconds and 350 seconds for the base and the complex scenarios, respectively. The consecutive actions of RCS cooldown and depressurization take less time in the complex scenario. Even for tasks with the same high-level goal or success criteria in the base scenario and complex scenario, there were cases where different means must be used because the given situation is different (e.g., there were differences because of the operational status of the automatic system or the available equipment). The results suggest whether the scenario is complex does not affect performance time the same way for each scenario. In other words, task complexity in the complex case was not always higher than in the base case, so it is necessary to calculate the task complexity in a given situation. The operator performance can be affected by various PSFs beyond complexity; therefore, a decomposition analysis may be required.

### 8.2.2 Task Complexity Score

The Task Complexity (TACOM) score, proposed by Park et al. (2002), provides a quantification method for the complexity of procedural tasks performed by NPP main control room operators (Jung et al. 2007; Podofillini et al. 2013). The suitability of the measure has been validated by comparing TACOM scores with two different types of human performance data—response times and subjective workload scores—showing the number of human errors increases proportionally with an increase in the TACOM score. The early model of the TACOM score had five sub-measures: step information complexity (SIC), step logic complexity (SLC), step size complexity (SSC), abstraction hierarchy complexity (AHC), and engineering decision complexity (EDC). To eliminate the dependency between the sub-measures, the revised version of TACOM (or R-TACOM) has suggested three sub-measures (Park and Jung 2007): (1) task scope (TS), (2) task structure (TR), and (3) task uncertainty (TU). Table 22 shows definitions, meanings, and relational expressions for each measure and sub-measure of TACOM and R-TACOM. The authors performed additional analyses using empirical data from the Halden study, and the results are explained in the next section.

Table 22. Measures of TACOM (from Park et al. 2007).

| Designation | Definition / Meaning |
|---|---|
| TACOM | $TACOM = \sqrt{(\alpha \times SIC)^2 + (\beta \times SLC)^2 + (\gamma \times SSC)^2 + (\delta \times AHC)^2 + (\varepsilon \times EDC)^2}$ |
| SIC | Step information complexity (SIC) represents the complexity due to the amount of information to be processed by human operators. |
| SLC | Step logic complexity (SLC) represents the complexity due to the execution logic of prescribed actions to be sequenced by human operators. |
| SSC | Step size complexity (SSC) represents the complexity due to the amount of prescribed actions to be performed by human operators. |
| AHC | Abstraction hierarchy complexity (AHC) represents the complexity due to the amount of system knowledge that is necessary to identify the problem space of the required operations. |
| EDC | Engineering decision complexity (EDC) represents the complexity due to the amount of cognitive resources that is necessary to establish the proper decision criteria of the required operations. |
| $\alpha, \beta, \gamma, \delta, \varepsilon$ | Relative weights for SIC, SLC, SSC, AHC and EDC, respectively. |
| R-TACOM | $R - TACOM = \sqrt{w_{TS} \times TS^2 + w_{TR} \times TR^2 + w_{TU} \times TU^2} = \sqrt{0.621 \times TS^2 + 0.239 \times TR^2 + 0.140 \times TU^2}$ $where, \begin{cases} TS = 0.716 \times SIC + 0.284 \times SSC \\ TR = 0.891 \times SLC + 0.109 \times AHC \\ TU = EDC \end{cases}$ |
| TS | Task scope (TS) represents the breadth, extent, range, or general size of a task. |
| TR | Task structure (TR) indicates whether the sequence and relationships between subtasks are well-defined or well structured. |
| TU | Task uncertainty (TU) is related to the degree of a predictability or a confidence associated with a task. |

### 8.2.3 Complexity Time Multiplier for TACOM

As mentioned in Section 5.2.1, the Halden study data do not lend themselves to deriving a simple multiplier between overall scenario complexity and performance time. The reason is that various parallel (and seemingly confounding) factors affect human performance, and the complexity of the entire scenario does not equally affect the complexity of the subdivided tasks (i.e., there is a big difference between overall complexity and local complexity). However, through a more detailed analysis, the relationship between PSFs like task complexity and time can be derived. In the simulation analysis result of a study by Park (2014), the relational expression for execution time (or response time) according to the TACOM score was derived and presented (Park and Cho 2010). A regression analysis was used for this derivation, and the relation is as follows:

$$Response\ time = 1.340 \cdot e^{0.987 \cdot TACOM} \quad (3)$$

The TACOM time relational expressions according to the prediction limits of upper 95% and lower 95% are:

$$Time = 2.918 \cdot e^{(0.986 \cdot TACOM)} \ \cdots (upper\ 95\%\ prediction\ limit) \quad (4)$$

$$Time = 0.617 \cdot e^{(0.986 \cdot TACOM)} \ \cdots (lower\ 95\%\ prediction\ limit) \quad (5)$$

Referring to this result, it seems that the relationship between PSFs (e.g., task complexity) and time as performance data can be derived. However, the current HUNTER model does not calculate the TACOM sub-measures, and further work will be required to map TACOM to the parameters like GOMS-HRA task-level primitives used in HUNTER. As such, it is preferable within HUNTER to derive context-specific or local complexity-to-time measures when a complexity multiplier is needed.

## 8.3 Method

### 8.3.1 Scenario

The HUNTER software implementation outlined in Sections 3 and 4 was put into practice for an SGTR scenario. This demonstration serves as a proof of concept in terms of the method of modeling human activities and the execution of the software code. SGTR was chosen because it is well documented in the static HRA literature, as discussed earlier in this section. In addition, INL has experience running operators through SGTR scenarios in the HSSL. While the HSSL has mostly been used to validate digital human-system interfaces in support of control room modernization, SGTR was used as a warmup exercise for those studies involving pressurized-water reactors (Medema et al. 2021). As such, detailed operator performance data were available to supplement published data and help build the dynamic model. As noted in Section 2, such data were also previously used to build the times associated with the task-level primitives in GOMS-HRA (Ulrich et al. 2017a).

The SGTR runs in the HSSL did not include all HFEs associated with SGTR. The SGTR exercise served as a warmup activity to familiarize operators with the board layouts, align them with the correct version of the Westinghouse procedures, and ensure proper command-and-control dynamics between the shift supervisor and the reactor operators. These purposes were accomplished primarily in walking through the EOP E-0, which involves diagnosing the fault before transitioning to EOP E-3, the appropriate procedure for SGTR. As such, the SGTR simulator runs involved a period of normal operations, insertion of the fault, and the crew diagnosing the fault by working through EOP E-0. When a crew brief was initiated by the shift supervisor to transition to EOP E-3, the SGTR scenario was terminated.

The scenario modeled for the present purposes corresponds to the first part of HFE-1 found in the Halden study (Bye et al. 2011) described earlier in this section. The Halden study captured two elements of HFE-1—first the identification of the faulted steam generator and then its isolation. The first part corresponds to EOP E-0, while the latter corresponds to EOP E-3.

## 8.4 HUNTER Approach to Complexity Effects on Task Duration

### 8.4.1 Time Measures

Previous modeling using HUNTER involved using the GOMS-HRA approach in conjunction with autocalculated PSF multipliers to estimate dynamic HEPs (Ulrich et al. 2017a). However, as noted in Boring et al. (2018), the conversion of a dynamic time series of HEPs to the more conventional static HEPs for each HFE is problematic. There remains no definitive way to map or aggregate the dynamically calculated HEPs. The focus of the current efforts has been on developing the software platform and not on the mathematical underpinnings of dynamic HEP calculations. The most appropriate ways to use dynamic HEPs will be further investigated as the project progresses.

The Halden study reveals that HEPs are not only a result of overt errors committed by crews. Overt errors manifest as wrong paths taken by the operators, incorrect system activations, or the failure to activate needed systems to support the functions of the plant. Non-overt errors show up as delays in completing required actions. Some of these present as technical specification violations, which occur when activities must be completed within a certain amount of time and are not. Others occur as something being performed more slowly than the normal course of action. This may relate to a subset of activities required to be compliant with an overall technical specification time limit or more generically to the thirty-minute rule used at plants to specify how long it should take at most to resolve plant upset conditions by operators. In the case of the Halden study, the research team codified several time windows during which particular actions should be completed. If crews failed to complete the actions in the specified time windows, the performance on the task was treated as an overtime error. Overtime errors were used in the calculation of HEPs for the Halden study.

Because of the uncertainties of calculating overt error HEPs dynamically and conversely the prospective value in determining time windows to calculate overtime errors, the model runs presented here focused on calculating the duration of tasks associated with the SGTR scenario. The time to complete each task was calculated in HUNTER using the GOMS-HRA timing data for each task-level primitive. Each task was calculated according to a distribution, resulting in different times for each run within the uncertainty bounds provided by GOMS-HRA. HUNTER logs time on each procedure step as well as the overall time to complete the task across each run.

### 8.4.2 Derivation of Complexity PSF for Time

The HUNTER simulation runs feature two conditions, a base or normal condition and a complex condition, mirroring the Halden benchmark conditions for the SGTR study. Building on the work done earlier on PSF autocalculation, we manipulated the complexity PSF in terms of its time effect. The base condition featured a time multiplier of 1 (i.e., no effect) for complexity, while the complex case featured a time multiplier to denote the increased complexity. Note that the Halden scenarios actually differed for the base and complex cases, and it may not be a suitable generalization to treat the differences only in terms of a single PSF. The differences in scenarios were more prevalent after HFE-1 in the Halden study, and HFE-1 may arguably be suitably captured by the complexity PSF. This was, in fact, the treatment provided by several HRA teams when performing the analysis for HFE-1.

For the purposes of illustrating the effects of PSF manipulations for this demonstration in HUNTER, we ran two scenarios—one base condition with no complexity PSF time multiplier in effect and one complex condition with the complexity PSF time multiplier in effect. The time multiplier was applied for each GOMS-HRA primitive, meaning the time distribution for the multiplier affects each calculated step of the model run individually.

To determine the appropriate multiplier, we reviewed the time data for the Halden study. As mentioned, there is no consistent timing relationship for SGTR across the base HFEs and the complex HFEs. HFE-1, which mostly corresponds to what we've modeled in HUNTER, sees the duration 1.664× slower for the complex vs. base case. HFE-1A (i.e., base SGTR) has a mean time of 970 seconds, whereas HFE-1B (i.e., complex SGTR) has a mean time of 1614 seconds. The time range is 623–1312 seconds for the simple case and 1289–1928 seconds for the complex case. The time effect of complexity HFE-1A may be represented as follows:

$$t_{complex(HFE1B)} = 1.664\, t_{base(HFE1A)} \tag{6}$$

This formula may not prove a universal solution. Recall the problem is that the other HFEs in Halden, namely HFE-2 and HFE-3, actually see the complex case performing faster than the base case. This can be explained by the fact that there ended up being less ambiguity over the method of cooldown (HFE-2) and depressurization (HFE-3) in the complex case. As noted in the description for HFE-2A (i.e., base SGTR) (Bye et al. 2011):

> ...3 out of 4 crews which unwillingly activated the steam line protection system (which causes steamline isolation) used extra time for completion of the task. The unexpected event disrupted their plan and resulted in minor problems (e.g., discussions, SG PORVs settings) that required extra time to recover, with the result of approaching or exceeding the allotted time.

In essence, some crews were not actually doing the same task for the base case as the complex case for HFE-2 and HFE-3. It's therefore impossible to compare their timing performance directly. HFE-1, however, remains comparable between the base and complex SGTR in terms of tasks completed, allowing us to focus on extracting the time multiplier for the complexity PSF for this scenario. This represents local complexity.

It is important to calibrate the actual activities to arrive at the appropriate effect of complexity on time. The Halden study's HFE-1 represents additional activities to those modeled in the HSSL studies and in HUNTER. Halden's HFE-1 goes through to steam generator isolation about seven steps into EOP E-3, whereas the SGTR scenario modeled here stops at identification of the steam generator fault at the point where the operators transfer from EOP E-0 to E-3. So, the Halden HFE1 scenario has several actions that are not modeled in the HSSL and HUNTER renditions of the SGTR scenario.

Bye et al. (2011) observes that it took crews about 10 minutes (600 seconds) from the point of transfer to EOP E-3 to completion of the isolation, which triggers the segue to HFE-2. The report does not give the exact timeline but notes the range from entering EOP E-3 to completing HFE-1 was 06:15 minutes to 13:27 minutes (375 – 807 seconds) for the base case. In consideration of that, the average time for crews in the Halden study to reach EOP E-3 was 370 seconds (calculated as 970 seconds total time for HFE-1 minus 600 seconds in E-3).

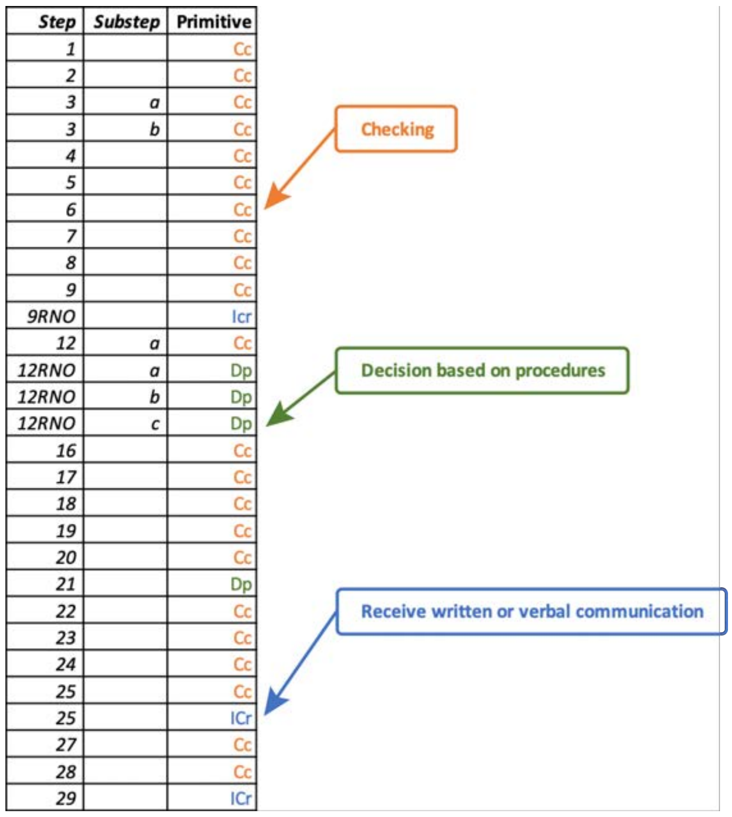
Notice that, for the complex case, Bye et al. (2011) state that the crews spent about 12 minutes (720 seconds) on average completing the EOP E-3 activities modeled in HFE-1, with a range of 8:36 minutes to 17:19 minutes (516 – 1039 seconds). The average time in HFE-1 before transfer to EOP E-3 for the complex scenario in the Halden study was 894 seconds (calculated as 1614 seconds total time for HFE-1 minus 720 seconds in E-3).

Thus, the time ratio for complex to basic scenarios for the EOP E-0 portion of HFE-1 in the Halden study is 894:370 seconds, which is 2.416. This ratio may be the preferred local complexity multiplier for time in EOP E-0:

$$t_{complex(E0)} = 2.416\ t_{base(E0)} \tag{7}$$

### 8.4.3 Model Runs

The HUNTER application was used to simulate a base and complex SGTR scenario. The input deck was populated for the initial response, AOP-16, and the emergency response, EOP E-0, to identify the ruptured steam generator. The simulation terminates once the faulted steam generator is identified. The base and complex scenarios were each run 1,000 times, with timing data calculated using the GOMS-HRA task-level primitives and adjusted by the complexity PSF timing multiplier for the complex scenario. The GOMS-HRA primitives were mapped onto the procedure steps and substeps for the AOP-016 and EOP E-0 (shown in Figure 46). Each of the primitives has an associated time distribution that was sampled during each of the simulated runs. The simulated runs used logged parameters obtained from previous runs with actual crews in the HSSL using GSE Systems' GPWR. Because of the availability of logged plant and operator data, the runs did not couple to a live data feed from the environment module, meaning a plant simulation code was not used interactively for the model run.

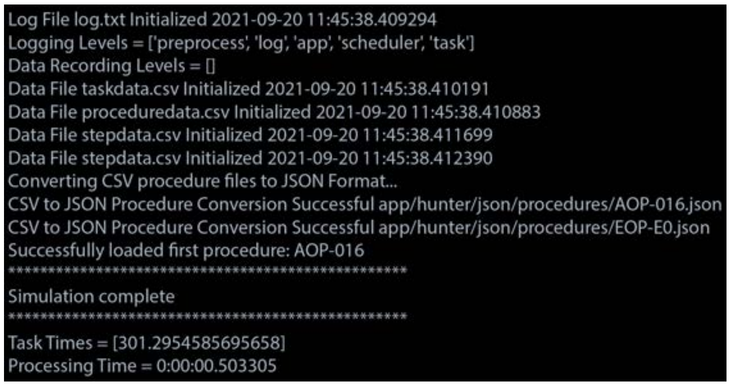

Checking
Decision based on procedures
Receive written or verbal communication

| Step | Substep | Primitive |
|---|---|---|
| 1 | | Cc |
| 2 | | Cc |
| 3 | a | Cc |
| 3 | b | Cc |
| 4 | | Cc |
| 5 | | Cc |
| 6 | | Cc |
| 7 | | Cc |
| 8 | | Cc |
| 9 | | Cc |
| 9RNO | | Icr |
| 12 | a | Cc |
| 12RNO | a | Dp |
| 12RNO | b | Dp |
| 12RNO | c | Dp |
| 16 | | Cc |
| 17 | | Cc |
| 18 | | Cc |
| 19 | | Cc |
| 20 | | Cc |
| 21 | | Dp |
| 22 | | Cc |
| 23 | | Cc |
| 24 | | Cc |
| 25 | | Cc |
| 25 | | ICr |
| 27 | | Cc |
| 28 | | Cc |
| 29 | | ICr |

Figure 46. GOMS-HRA task-level primitive mapping to Procedure EOP-E0.

### 8.4.4 Results of Model Runs

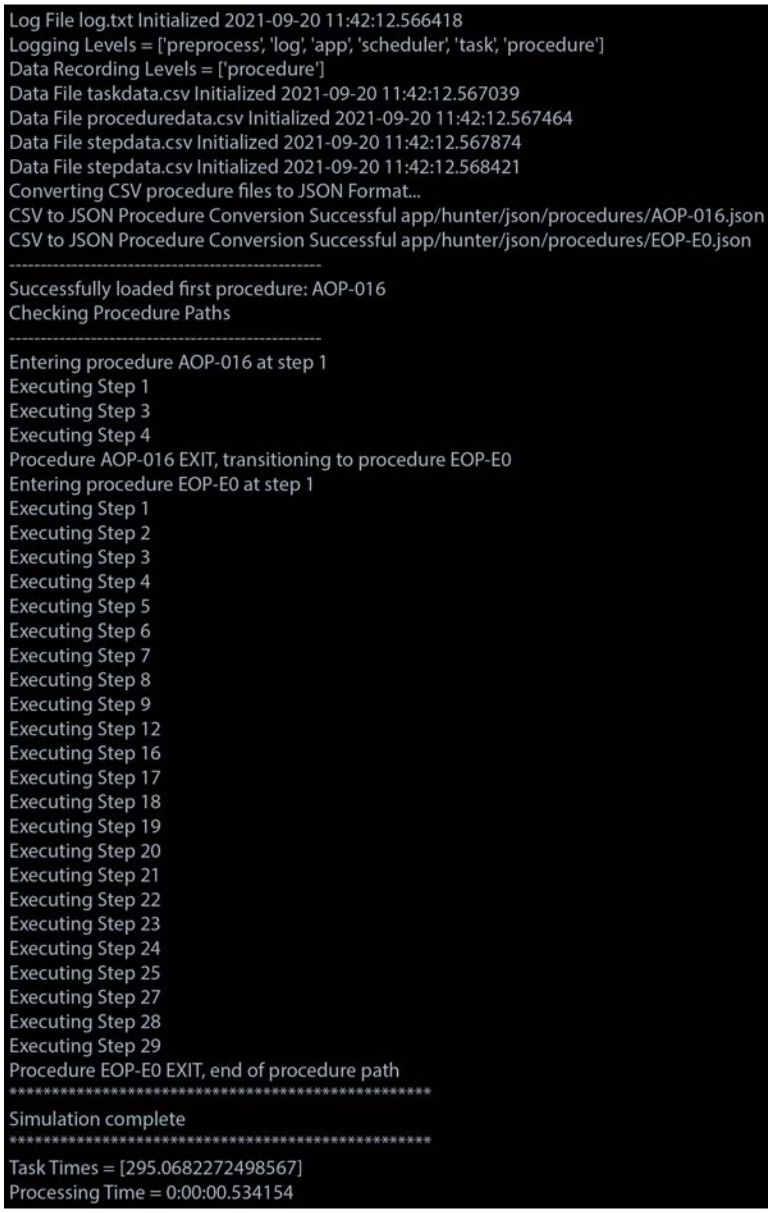
The results of the SGTR model run in a Monte Carlo simulation in HUNTER takes the form of log and data files that can then be statistically analyzed.



```
Log File log.txt Initialized 2021-09-20 11:45:38.409294
Logging Levels = ['preprocess', 'log', 'app', 'scheduler', 'task']
Data Recording Levels = []
Data File taskdata.csv Initialized 2021-09-20 11:45:38.410191
Data File proceduredata.csv Initialized 2021-09-20 11:45:38.410883
Data File stepdata.csv Initialized 2021-09-20 11:45:38.411699
Data File stepdata.csv Initialized 2021-09-20 11:45:38.412390
Converting CSV procedure files to JSON Format...
CSV to JSON Procedure Conversion Successful app/hunter/json/procedures/AOP-016.json
CSV to JSON Procedure Conversion Successful app/hunter/json/procedures/EOP-E0.json
Successfully loaded first procedure: AOP-016
**************************************************
Simulation complete
**************************************************
Task Times = [301.2954585695658]
Processing Time = 0:00:00.503305
```

Note: No procedure, step, or substep level information is reported, but the overall completion times for each simulation run are recorded.

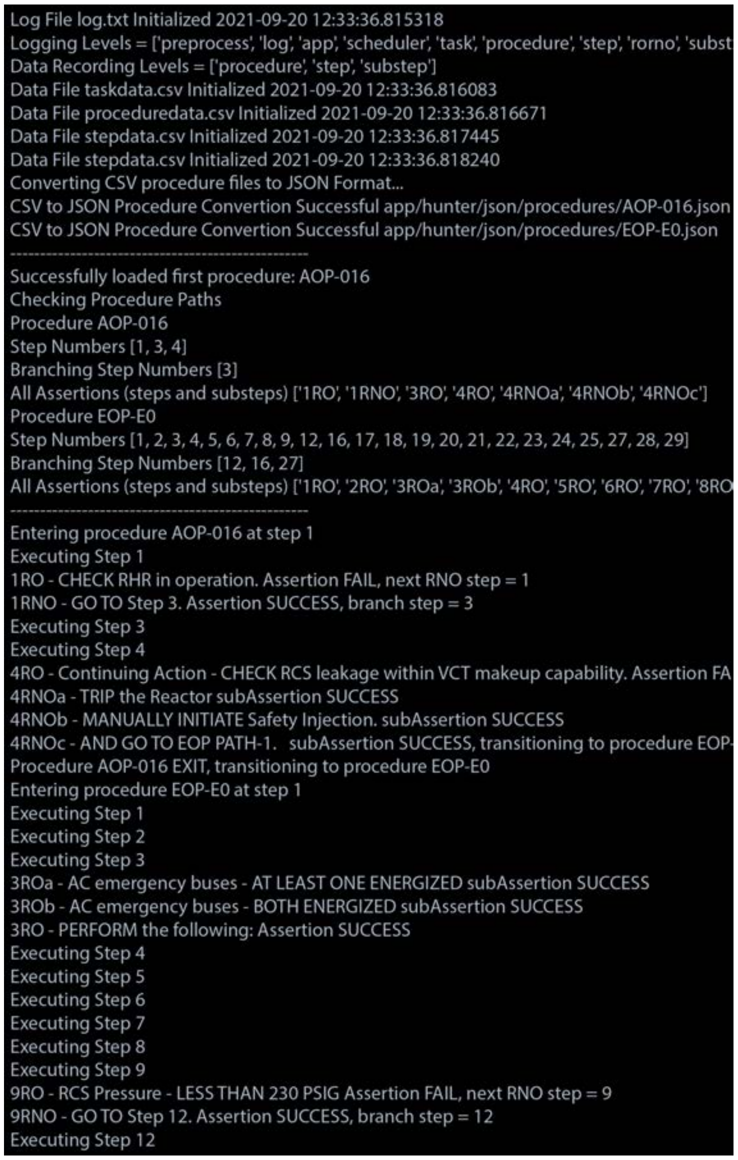
Figure 47. Log file depicting showing overall execution of the SGTR model in HUNTER.



```
Log File log.txt Initialized 2021-09-20 11:42:12.566418
Logging Levels = ['preprocess', 'log', 'app', 'scheduler', 'task', 'procedure']
Data Recording Levels = ['procedure']
Data File taskdata.csv Initialized 2021-09-20 11:42:12.567039
Data File proceduredata.csv Initialized 2021-09-20 11:42:12.567464
Data File stepdata.csv Initialized 2021-09-20 11:42:12.567874
Data File stepdata.csv Initialized 2021-09-20 11:42:12.568421
Converting CSV procedure files to JSON Format...
CSV to JSON Procedure Conversion Successful app/hunter/json/procedures/AOP-016.json
CSV to JSON Procedure Conversion Successful app/hunter/json/procedures/EOP-E0.json
--------------------------------------------------
Successfully loaded first procedure: AOP-016
Checking Procedure Paths
--------------------------------------------------
Entering procedure AOP-016 at step 1
Executing Step 1
Executing Step 3
Executing Step 4
Procedure AOP-016 EXIT, transitioning to procedure EOP-E0
Entering procedure EOP-E0 at step 1
Executing Step 1
Executing Step 2
Executing Step 3
Executing Step 4
Executing Step 5
Executing Step 6
Executing Step 7
Executing Step 8
Executing Step 9
Executing Step 12
Executing Step 16
Executing Step 17
Executing Step 18
Executing Step 19
Executing Step 20
Executing Step 21
Executing Step 22
Executing Step 23
Executing Step 24
Executing Step 25
Executing Step 27
Executing Step 28
Executing Step 29
Procedure EOP-E0 EXIT, end of procedure path
**************************************************
Simulation complete
**************************************************
Task Times = [295.0682272498567]
Processing Time = 0:00:00.534154
```

Note: This level of recording captures the execution of each procedure step along with their parent procedure. No substep level information is reported.

Figure 48. Log file showing stepwise execution of the SGTR model in HUNTER.



```
Log File log.txt Initialized 2021-09-20 12:33:36.815318
Logging Levels = ['preprocess', 'log', 'app', 'scheduler', 'task', 'procedure', 'step', 'rorno', 'subst
Data Recording Levels = ['procedure', 'step', 'substep']
Data File taskdata.csv Initialized 2021-09-20 12:33:36.816083
Data File proceduredata.csv Initialized 2021-09-20 12:33:36.816671
Data File stepdata.csv Initialized 2021-09-20 12:33:36.817445
Data File stepdata.csv Initialized 2021-09-20 12:33:36.818240
Converting CSV procedure files to JSON Format...
CSV to JSON Procedure Convertion Successful app/hunter/json/procedures/AOP-016.json
CSV to JSON Procedure Convertion Successful app/hunter/json/procedures/EOP-E0.json
------------------------------------------------
Successfully loaded first procedure: AOP-016
Checking Procedure Paths
Procedure AOP-016
Step Numbers [1, 3, 4]
Branching Step Numbers [3]
All Assertions (steps and substeps) ['1RO', '1RNO', '3RO', '4RO', '4RNOa', '4RNOb', '4RNOc']
Procedure EOP-E0
Step Numbers [1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29]
Branching Step Numbers [12, 16, 27]
All Assertions (steps and substeps) ['1RO', '2RO', '3ROa', '3ROb', '4RO', '5RO', '6RO', '7RO', '8RO
------------------------------------------------
Entering procedure AOP-016 at step 1
Executing Step 1
1RO - CHECK RHR in operation. Assertion FAIL, next RNO step = 1
1RNO - GO TO Step 3. Assertion SUCCESS, branch step = 3
Executing Step 3
Executing Step 4
4RO - Continuing Action - CHECK RCS leakage within VCT makeup capability. Assertion FA
4RNOa - TRIP the Reactor subAssertion SUCCESS
4RNOb - MANUALLY INITIATE Safety Injection. subAssertion SUCCESS
4RNOc - AND GO TO EOP PATH-1.   subAssertion SUCCESS, transitioning to procedure EOP-
Procedure AOP-016 EXIT, transitioning to procedure EOP-E0
Entering procedure EOP-E0 at step 1
Executing Step 1
Executing Step 2
Executing Step 3
3ROa - AC emergency buses - AT LEAST ONE ENERGIZED subAssertion SUCCESS
3ROb - AC emergency buses - BOTH ENERGIZED subAssertion SUCCESS
3RO - PERFORM the following: Assertion SUCCESS
Executing Step 4
Executing Step 5
Executing Step 6
Executing Step 7
Executing Step 8
Executing Step 9
9RO - RCS Pressure - LESS THAN 230 PSIG Assertion FAIL, next RNO step = 9
9RNO - GO TO Step 12. Assertion SUCCESS, branch step = 12
Executing Step 12
```

Note: This level of recording captures the execution of each procedure step and substep along with their parent procedure. No primitive or point results for each step or substep are reported.

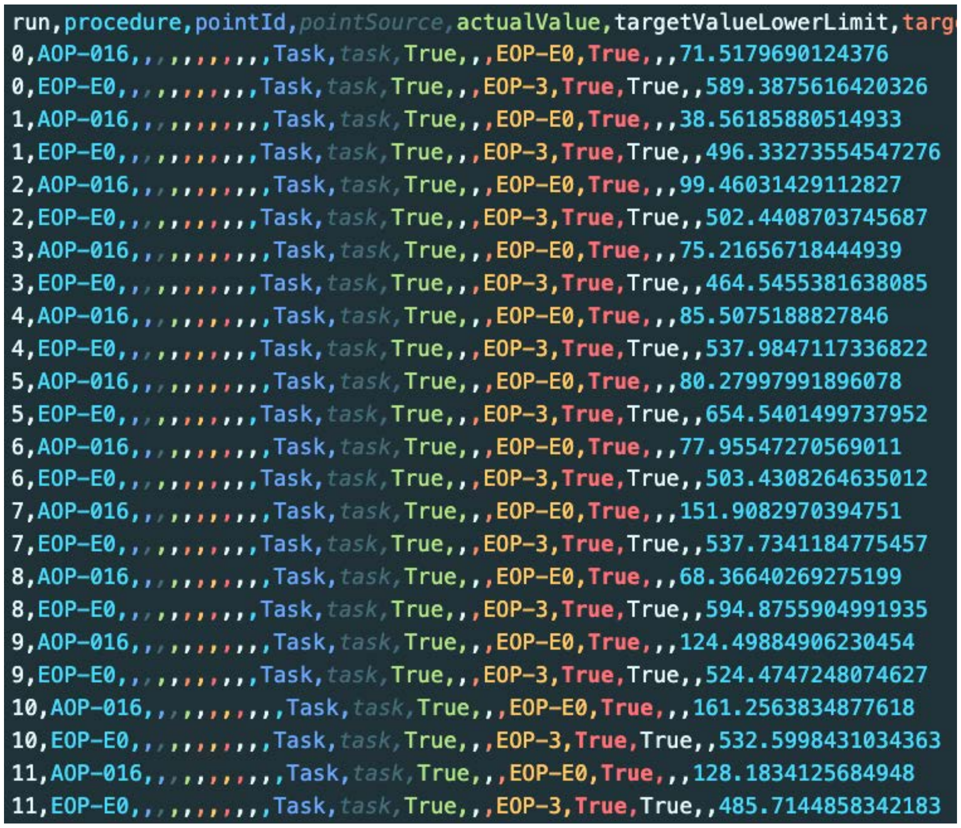
Figure 49. Log showing substep execution of the SGTR model in HUNTER.

### 8.4.5 Data Output

Running the SGTR model in HUNTER generates several output files that support the further analysis of the simulation results. Two types of files are output, and the analyst can define the granularity of the data recording and logging. The analyst can set the output granularity at the simulation, procedure, step, substep, primitive, and point levels (see Figure 47, 48, and 49 for examples of three different levels of logging). Depending on the purpose of the analysis, the analyst might require greater resolution in regard to the times and HEPs calculated for the procedures, steps, and substeps. If the analyst is only interested in the overall time required for a task, such as in the validation performed here against the Halden SGTR scenarios, the details of each step and substep are not necessary and the simulation run time can be reduced by recording and logging only what is of interest. Furthermore, data recording and logging at the appropriate level also eases analysis as the output data is readily aggregated at the appropriate granularity by precluding any preprocessing of raw data before performing statistical analyses of interest.

The primary difference between the data and log files is their formatting. The data files are formatted as CSV files to support their import into analysis tools, such as Excel or statistical software packages. The log data is in a more human readable format, which serves two purposes. First, it is much easier to view the execution path and provides valuable context for the quantitative data contained within the data files. Second, it serves as a debugging tool that allows the analyst to ensure the input files were read properly and executed as expected.

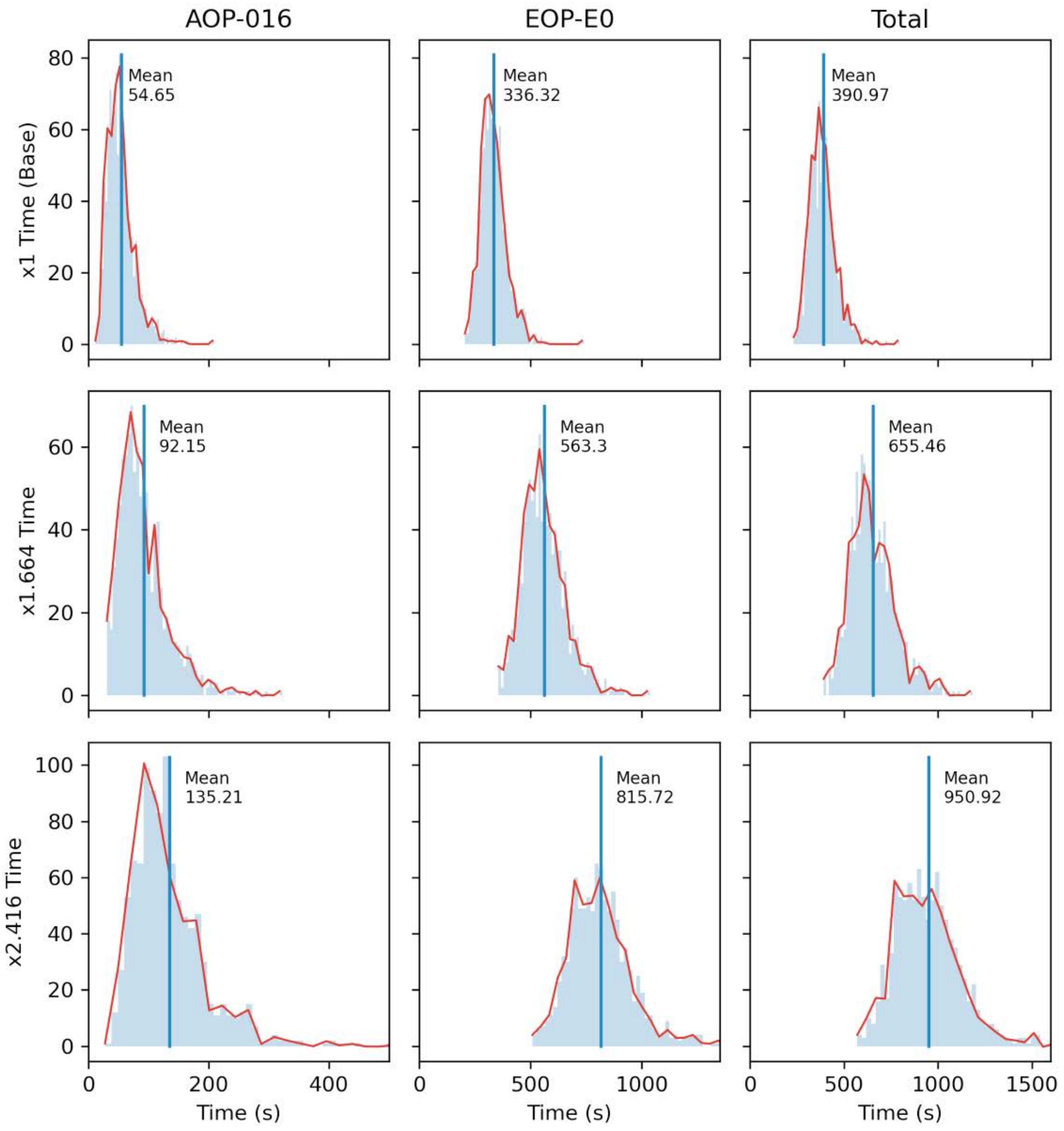
The data output can also be configured to record the simulation run. An example of data recorded at the simulation level can be seen in Figure 50, which depicts a portion of the actual data file that was used to generate the distributions shown in Figure 51. Each simulation run and each procedure within the run is recorded as a single line of data with various pieces of information, such as the elapsed time taken to complete that procedure.



```
run,procedure,pointId,pointSource,actualValue,targetValueLowerLimit,targ
0,AOP-016,,,,,,,,,,Task,task,True,,,EOP-E0,True,,,71.5179690124376
0,EOP-E0,,,,,,,,,,Task,task,True,,,EOP-3,True,True,,589.3875616420326
1,AOP-016,,,,,,,,,,Task,task,True,,,EOP-E0,True,,,38.56185880514933
1,EOP-E0,,,,,,,,,,Task,task,True,,,EOP-3,True,True,,496.33273554547276
2,AOP-016,,,,,,,,,,Task,task,True,,,EOP-E0,True,,,99.46031429112827
2,EOP-E0,,,,,,,,,,Task,task,True,,,EOP-3,True,True,,502.4408703745687
3,AOP-016,,,,,,,,,,Task,task,True,,,EOP-E0,True,,,75.21656718444939
3,EOP-E0,,,,,,,,,,Task,task,True,,,EOP-3,True,True,,464.5455381638085
4,AOP-016,,,,,,,,,,Task,task,True,,,EOP-E0,True,,,85.5075188827846
4,EOP-E0,,,,,,,,,,Task,task,True,,,EOP-3,True,True,,537.9847117336822
5,AOP-016,,,,,,,,,,Task,task,True,,,EOP-E0,True,,,80.27997991896078
5,EOP-E0,,,,,,,,,,Task,task,True,,,EOP-3,True,True,,654.5401499737952
6,AOP-016,,,,,,,,,,Task,task,True,,,EOP-E0,True,,,77.95547270569011
6,EOP-E0,,,,,,,,,,Task,task,True,,,EOP-3,True,True,,503.4308264635012
7,AOP-016,,,,,,,,,,Task,task,True,,,EOP-E0,True,,,151.9082970394751
7,EOP-E0,,,,,,,,,,Task,task,True,,,EOP-3,True,True,,537.7341184775457
8,AOP-016,,,,,,,,,,Task,task,True,,,EOP-E0,True,,,68.36640269275199
8,EOP-E0,,,,,,,,,,Task,task,True,,,EOP-3,True,True,,594.8755904991935
9,AOP-016,,,,,,,,,,Task,task,True,,,EOP-E0,True,,,124.49884906230454
9,EOP-E0,,,,,,,,,,Task,task,True,,,EOP-3,True,True,,524.4747248074627
10,AOP-016,,,,,,,,,,Task,task,True,,,EOP-E0,True,,,161.2563834877618
10,EOP-E0,,,,,,,,,,Task,task,True,,,EOP-3,True,True,,532.5998431034363
11,AOP-016,,,,,,,,,,Task,task,True,,,EOP-E0,True,,,128.1834125684948
11,EOP-E0,,,,,,,,,,Task,task,True,,,EOP-3,True,True,,485.7144858342183
```

Note: The data headings are truncated in this screenshot; however, some key information can be seen. The run number is the leftmost column, the adjacent column shows the procedure, and the far right column shows the elapsed time for that procedure and run.

Figure 50. Screenshot of data file output generated by the HUNTER Python code.



Note: Blue bars represents histogram plots of model runs, and red lines represent empirical density estimate of the distributions.

Figure 51. Distribution of times for HUNTER SGTR model runs.

Table 23. Mean times for SGTR scenario runs for Halden and HUNTER.

| | Mean Time (s) | | |
|---|---|---|---|
| | Basic | Complex | |
| **Halden** | 370 | 894 | |
| **HUNTER** | 391 | 655 | ×1.664 Time |
| | 391 | 951 | ×2.416 Time |

### 8.4.6 HUNTER Validation Results

To validate the SGTR model in the HUNTER Python code, three conditions for the SGTR model were simulated. A base condition consisted of the GOMS-HRA task-level primitives to calculate the time for each step. Two additional simulations were performed using the 1.664 and 2.416 multipliers from Equations 5 and 6, representing the respective overall and local added times for complexity derived from the Halden study. Figure 51 shows the distributions of the generated simulation times for the AOP-016, EOP E-0, and joint AOP-16 and EOP E-0 procedure portions of the SGTR. Table 23 shows the means of the simulation runs compared to the empirically measured Halden study completion times for the SGTR HFE-1 scenario up to entering EOP E-3. The HUNTER simulation closely replicated the total time for basic scenario. The complex scenario simulated in HUNTER using the time multipliers also approximated the empirical data from the Halden study. The local complexity time multiplier of 2.416 resulted in a distribution of completion times most similar to the Halden data than compared to the general complexity time multiplier of 1.664.

### 8.4.7 Discussion

The HUNTER simulations demonstrated good agreement in this initial validation. The base scenario simulation with a mean time of 391 seconds was only 16 seconds longer than the mean of the empirical data from the Halden study, which represents a 4.3% error in the average HUNTER SGTR model time estimate. Because the SGTR model in HUNTER was built independently of the Halden study, this is a promising result and demonstrates that the model of the SGTR scenario built from the GOMS-HRA task-level primitive timing data is a good approximation for a crew performing the standard SGTR plant response actions. The simulated complex condition also showed good agreement, with a difference of 57 seconds observed between the 951 seconds mean time for the distributions of simulated completion times and the mean 894 seconds from the Halden study. The 57 second difference is 6.4% and well within the bounds of a reasonable approximation.

This difference could also be minimized with a more refined method of complexity estimation. The time multiplier was applied uniformly across all GOMS-HRA task-level primitives, but a differential or even dynamically calculated multiplier is needed to be more realistic. Specifically, the complexity in the Halden study stems from an additional masking indication that requires additional steps for diagnosis. In the model here, complexity is treated simply as longer executions of the same steps, not as different steps. Therefore, a more accurate model of the Halden complex scenario should be developed and tested against the SGTR modeled in the HUNTER simulation to serve as additional validation. The simulation nonetheless provides promise for HUNTER as a platform for performing dynamic HRA related to time estimation.

# 9. DISCUSSION AND NEXT STEPS

## 9.1 Limitations

The research presented in this report is preliminary. The conceptual framework of HUNTER has been expanded to provide a more adaptable software architecture, and the software implementation incorporates many of these key features. But, there remains more development to be done on the modules and classes. For example, the current implementation uses logged simulator data instead of a simulation interactively driven by the virtual operator in HUNTER. Additionally, key features like the autocalculation of PSFs, decision-making, and HEP aggregation are not yet deployed in this early software version. These features will be expanded considerably prior to release of HUNTER as a standalone software tool for risk analysis. Likewise, the single scenario modeled—that of an SGTR—is but a brief demonstration of the types of scenarios that are of interest to industry.

The present version and application of HUNTER must be seen as an early proof of concept, with more complete development on the envisioned features still in the future. Nonetheless, the initial deployment of HUNTER and sample scenario run show the promise of the software to support dynamic HRA modeling needs in the future. Future software development will follow a twofold approach. First, the deficiencies noted in Section 2—such as a lack of standalone software or documentation—will continue to be completed to create a vetted and usable tool that supports industry needs. In parallel, additional features will be deployed. The conclusion of this report in the next section highlights some of the features that are being planned for future iterations of HUNTER.

## 9.2 Future Work

### 9.2.1 Advanced Programming Interface and Coupling to Additional External Codes

As described in Chapter 6, the initial software version of HUNTER includes copuling to RELAP5-3D. HUNTER already supports access to point data (i.e., parameters) in external software codes as depicted in Figure 4. The HUNTER model knows to look for specific inputs, which correspond to labels or locations in the appropriate configuration file. For example, a procedure provides opportunities for operators to look at specific plant parameters and to take control actions. In simulation code, these parameters are maintained in a database (i.e., the indicators and controls are represented by particular parameters that may be looked up or changed in the model database). Each step in the operating procedure that drives HUNTER progress has the opportunity for data inputs and outputs. The specific mechanism for exchanging data involves an API. A number of APIs are being considered for HUNTER to allow it a wider variety of modeling interactivity between the virtual plant and the virtual operator and ease the opportunity for industry to use HUNTER without extensive RELAP5-3D expertise. These APIs include code for data exchanges with the following external software codes:

- *GSE System's GPWR Simulator*—this commercially available full-scope simulator, adapted from the qualified training simulator at a pressurized-water reactor in the U.S., has been used extensively for control room modernization studies in the HSSL (Medema et al. 2021; Joe and Boring 2017). INL has previously developed a way for digital control prototypes to communicate with the simulator, both to read parameters for displays and to set control parameters for changing plant systems (e.g., starting a pump or closing a valve). GSE Systems provides a custom API called Gii to allow two-way communication between the plant simulator and outside software. INL has adapted Gii to work with Microsoft's .NET platform (for broad compatibility with Windows software) and with the Python programming language (for cross-code and cross-platform compatibility). The bridge API will work readily with HUNTER and can allow real-time operation of the virtual plant by the HUNTER virtual simulator. A related simulator called the

Generic Boiling Water Reactor is also available and could via similar means allow HUNTER to interface with a full-scope plant model for a boiling water reactor.

- *Rancor Microworld Simulator*—this simulator was developed as a simplified form of a full-scope simulator explicitly for research purposes (Ulrich et al. 2017b). By using fewer parameters and reduced-order models, Rancor has proven easy to learn and operate while still maintaining reasonable fidelity to actual plant performance. In fact, its ease of use makes it suitable for use by student operators, thereby allowing greater access to human-in-the-loop studies than is possible with full-scope simulators, which require highly skilled and comparatively less available professional reactor operators. Rancor supports a number of scenarios including SGTR (Park et al. in press), plus it has been adapted to novel applications like advanced reactors (Boring et al. 2021a). Currently, Rancor does not have a true API to support coupling with external software. However, the development team for Rancor is the same team that is developing the HUNTER software implementation. As such, creating an API to allow HUNTER to interface with Rancor would be a minor matter. The simplified nature of Rancor also makes it an ideal testbed for further proof-of-concept development, given the simpler modeling compared to the complexity of a full-scope plant model. Additionally, it is possible to pair Rancor and HUNTER in a multi-threaded faster-than-real-time manner for large-scale modeling efforts.
- *EMRALD*—the Event Modeling Risk Assessment using Linked Diagrams (EMRALD; Prescott, Smith, and Vang 2018) software is a discrete event simulation software that is easy to configure because of a graphical layout tool. EMRALD has been used recently to model physical security (Christian et al. 2020) and flexible plant operations (FLEX; Park et al. 2021), both of which involved creating custom interfaces to account for HRA. EMRALD does not currently directly support HRA functions. Rather than create possibly duplicative software tools to incorporate HRA, the goal is to create an API that allows EMRALD to invoke HUNTER in support of HRA modeling. While the details of this connection of software codes remain to be determined, it is likely that EMRALD would subsume the task and environment modules in HUNTER. HUNTER would benefit from using a tool that already accounts for these modules, while EMRALD would benefit from the human modeling that can inform human action probabilities, durations, and decisions on tasks.

The long-term goal is to provide APIs to link HUNTER to all the above software tools and flexibly incorporate additional software tools that arise. Coupling these software tools will likely require multiple years. The exact order of linking software tools remains to be determined.

### 9.2.2 Cognitive Modeling

As stated prior, the HUNTER framework's adaptability is an important advantage for HRA and PRA tasking, as the analyst needs only consider the key tasks or functionalities needed to complete the analysis at hand. One such module is an all-inclusive cognitive architecture or model that can better support the simulation of realistic and generalizable cognitive performance in human operators. A key concern in the HRA community going back several decades is the question of how well existing HRA methods truly represent the cognitive processes of humans and how any deficiencies may impact the analysis (Dougherty 1990; Rasmussen 1980; Boring 2015b; Hollnagel 2000).

Another challenge facing modern HRA is taking the next step into more dynamic HRA and the expectation that modern HRA models can handle shifting operational scenarios, such as FLEX, integrated energy systems, and novel environmental challenges facing nuclear power. The HUNTER framework's adaptability positions it well to grapple with these shifts; however, there is a need for something to handle cognitive task performance simulation of operators in such novel and emergent conditions where training or proceduralization may play less of a role in assuring high levels of performance. This is where the addition of a cognitive model pays dividends.

First, the increased modeling and simulation capacity that modern computational resources provide is extremely helpful in attempting to capture and model something as complex as human cognition. Second, a focus on a more dynamic and flexible incident progression is critical to any realistic assessment of the risk of human error. The HUNTER framework will make use of modern computational resources by integrating a cognitive model of decision-making into the human action process and will connect to an external simulated environment module such as RELAP5, with bidirectional communication through the event progression. This enables the physics simulation to provide the relevant environmental variables, which an operator uses to monitor the plant and make decisions. The need for dynamic models and the realities of shifting scenarios demonstrate the necessity for more dynamic models that could better represent the shifting contexts and fluidity of human action and the inclusion of more explicit psychological considerations. A dynamic plant model benefits little from a fixed or invariant human model.

The emphasis on a dynamic process is also a key step in creating a method that is more representative of actual human cognition and performance. Very rarely, if ever, do humans choose a path and march down it with a fixed and steady purpose. Rather, the more common characterization of human actions in this sense is the ever changing and adjusting of intention, effort, directionality, and goals. Even in highly proceduralized activities, there is the need to adapt to changing conditions. Humans are constantly in a state of flux in a continuously refreshed loop of cognition and environmental state changes. This reality can be a very difficult simulation challenge. By using a cognitive model connected to a representative world state simulation, HUNTER can loop through the specific aspects of the circumstances and update accordingly. This will give HRA teams more insight and information into how the humans are performing and where the errors are most likely to occur.

Additionally, the role of decision-making has been a difficult aspect to include or even handle within HRA methods. Decision-making is the source of many human errors of interest to HRA. The inclusion of a robust cognitive architecture will better capture the key nuances of human decision-making and highlight any sources of error that may otherwise be missed by traditional static HRA methods.

### 9.2.3 HEPs

As discussed in Section 8, the present implementation of HUNTER addresses time-related measures as a novel area to demonstrate added modeling capabilities in the transition from static to dynamic HRA. HRA typically focuses on how to estimate reasonable and acceptable HEPs and then provides them to PRA models (Park, Arigi, and Kim 2019; Park, Jung, and Kim 2020). In GOMS-HRA (see Table 2), the method provides nominal HEP information but doesn't tell us how to aggregate the HEPs. Accordingly, the authors' previous research (Boring et al. 2018) has suggested an approach to aggregate autocalculated HEPs from tasks to HFEs in a dynamic HRA implementation. Future research within HUNTER will address how to aggregate HEPs to allow better fits to industry-standard HFEs as units of analysis.

### 9.2.4 Automatic Procedure Parsing

Once the constituent software elements are in place to deploy a full-featured software version of HUNTER, one of the greatest challenges to modeling remains the effort required to build the input that drives the Task module and schedules the model runs. The majority of the other portions of the HUNTER software are reusable. PSFs, for example, rely on an underlying software that supports this functionality. The mapping of PSFs to particular parameters in the environment module is also a task that can be readily reused once completed the first time. However, each scenario that are run in HUNTER, as embodied in the Task module and driven by procedures, must be custom developed. While the remainder of HUNTER converges on a reusable library of software modules and classes, the Task module ends up being a bespoke representation that must be customized with each new analysis.

Because the HUNTER Task module is driven by procedures, one way to streamline this process would be to create tools that automate the process of translating the standard written procedure into the subtasks used to drive the model run forward. Such a tool might read a written procedure and parse the procedure into individual steps with accompanying assertions (i.e., procedure flow logic), GOMS-HRA task-level primitives, and inputs and outputs with the environment module. GOMS-HRA already features a mapping between procedures (i.e., procedure-level primitives) and task-level primitives (Boring et al. 2017) that could be expanded and automated.

Additional software tools created by the HUNTER development team include the Task Engine for Job and User Notification (TEJUN) (Lew et al. 2019). TEJUN is a computer-based procedure tool for interfacing with full-scope simulators. It provides a hybrid solution using a markdown language that could easily be used by the Task module in HUNTER. The markdown language used by TEJUN is currently created manually by translating paper-based procedures into a taxonomy that can be used for computer-based procedures. Whether fully automated or partially automated through a tool like TEJUN, the prospect of having an easier method to map procedures to the HUNTER Task module is the key to being able to use HUNTER easily for new analyses.

### 9.2.5 Additional Use Cases

The current demonstration only features a single scenario—SGTR. As noted, SGTR is an ideal scenario for building the proof-of-concept demonstration, because it is well documented in HRA publications. Additionally, the HUNTER development team has run the SGTR scenario multiple times in the HSSL and have detailed logs for plant parameters and operator performance available to them. As such, the modeling assumptions in HUNTER can be calibrated and modeling results can be validated to empirical findings.

Beyond this initial SGTR scenario, HUNTER is meant as a general-purpose risk software tool that will support a wide variety of human-centered scenarios. HUNTER must also be seen in the context of competing with well-established and widely used static HRA methods. These methods already have been used to model the most common risk scenarios required of PRAs. To truly benefit industry risk analysts, HUNTER must undertake modeling of two types of scenarios:

- *Existing scenarios already modeled in HRAs*—these analyses are necessarily duplicative to static HRA but allow analysts to transfer existing models to a dynamic structure. Additional aspects of the model such as the time windows may be useful to enhancing existing analyses that are reconsidered in HUNTER.
- *Novel scenarios that have not been covered in HRAs to date*—the benefit of HUNTER is in its ability to model phenomena for which there are no existing analyses and for which static methods may not be well suited. Emerging topics, as mentioned in Section 1.1, include severe accidents, advanced technologies like digital and automated human-system interfaces, and plant operations beyond the main control room. These analyses may be used for exploring phenomena not understood due to a lack of operational experience. By highlighting human performance areas of interest, dynamic modeling can serve to screen risk-significant phenomena that would benefit from empirical validation.

Going forward, HUNTER will seek to ensure the availability of analysis examples and templates that support industry's future needs in HRA. Once the initial software implementation of HUNTER is completed, novel modeling scenarios will also drive the addition of new features.

# 10. REFERENCES

Agarwal, V., Manjunatha, K.A., Gribok, A.V., Mortenson, T.J., Bao, H., Reese, R., Ulrich, T.A., Boring, R.L., & Palas, H. (2021). *Scalable Technologies Achieving Risk-Informed Condition-Based Predictive Maintenance Enhancing the Economic Performance of Operating Nuclear Power Plants*, *INL/EXT-21-64168.* Idaho Falls: Idaho National Laboratory.

Akerstedt, T. (1995). Work injuries and time of day—national data*. Shiftwork International Newsletter, 12(2).*

Aumiller, D.L., Tomlinson, E.T., & Bauer, R.C. (2001). A coupled RELAP5-3D/CFD methodology with a proof-of-principle calculation. *Nuclear Engineering and Design, 205,* 83-90.

Boring, R.L. (2009). Human reliability analysis in cognitive engineering. *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2008 Symposium* (pp. 103-110). Washington, DC: National Academy of Engineering.

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

Boring, R.L. (2015a). Defining human failure events for petroleum applications of human reliability analysis. *Procedia Manufacturing, 3*, 1335-1342.

Boring, R.L. (2015b). 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., Joe, J.C., and Mandelli, D. (2015). Human performance modeling for dynamic human reliability analysis. *Lecture Notes in Computer Science, 9184,* 223-234.

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

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

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

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

Boring, R., Rasmussen, M., Ulrich, T., & Lybeck, N. (2018). Aggregation of autocalculated human error probabilities from tasks to human failure events in a dynamic human reliability analysis. *Proceedings of Probabilistic Safety Assessment and Management.*

Boring, R.L., Shirley, R.B., Joe, J.C., Mandelli, D., and Smith, C.L. (2014). *Simulation and Non-Simulation Based Human Reliability Analysis Approaches, INL/EXT-14-33903.* Idaho Falls: Idaho National Laboratory.

Boring, R.L., Ulrich, T.A., Lew, R., & Hall, A. (2021a). A microworld framework for advanced control room design. *Proceedings of the 12th Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies (NPIC&HMIT 2021)*, pp. 14-17.

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,* 1681-1685.

Boring, R., Ulrich, T., Park, J., Mortenson, T., Ahn, J., & Lew, R. (2021b). *An Adaptable Toolkit for Dynamic Human Reliability Analysis: Progress Toward HUNTER 2, INL/EXT-21-64525*. Idaho Falls: Idaho National Laboratory.

Boring, R., Ulrich, T., & Rasmussen, M. (2018). Task level errors for human error prediction in GOMS-HRA. In *Safety and Reliability–Safe Societies in a Changing World* (pp. 433-439): CRC Press.

Boring, R.L., Whaley, A.M., Tran, T.Q., McCabe, P.H., Blackwood, L.G., & Buell, R.F. (2006). *Guidance on Performance Shaping Factor Assignments in SPAR-H, INL/EXT-06-11959.* Idaho Falls: Idaho National Laboratory.

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

Card, S. K., Moran, T. P., & Newell, A. (2018). *The Psychology of Human-Computer Interaction*: CRC Press.

Choi, Y.-J. (2019) *Assessment of Verification and Validation Status - RELAP5-3D and RAVEN, INL/EXT-19-56151.* Idaho Falls: Idaho National Laboratory.

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

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

Choi, Y.-J., Abdo, M., Mandelli, D., Epiney, A., Valeri, J., Gosdin, C., Frepoli, C., & Alfonsi, A. (2021). *Demonstration of the Plant Fuel Reload Process Optimization for an Operating PWR, INL/EXT-21-64549.* Idaho Falls: Idaho National Laboratory.

Choi, Y.-J. & Parisi C. (2020). *Risk-Informed Multi-Physics Best-Estimate Plus Uncertainties (BEPU) Application Development of RELAP5-3D Perturbation Model, INL-LTD-20-59594.* Idaho Falls: Idaho National Laboratory.

Christian, R., Prescott, S.R., Yadav, V., St Germain, S.W., & Weathersby, J. (2020*). Integration of FLEX Equipment and Operator Actions in Plant Force-on-Force Models with Dynamic Risk Assessment, INL/EXT-20-59510.* Idaho Falls: Idaho National Laboratory.

Coyne, K., & Mosleh, A. (2018). Dynamic Probabilistic Risk Assessment Model Validation and Application—Experience with ADS-IDAC, Version 2.0. In *Advanced Concepts in Nuclear Energy Risk Assessment and Management* (pp. 45-85): World Scientific.

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 Simualtor Data on LOFW Scenarios, NUREG/IA-0216, Vol. 3.* Washington, DC: U.S. Nuclear Regulatory Commission.

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.

Dougherty, E. M., Jr. (1990). Human reliability analysis—where shouldst thou turn? *Reliab. Eng. Syst. Saf., 29*(3), 283-299.

Electric Power Research Institute. (2016). *An Approach to Human Reliability Analysis for External Events with a Focus on Seismic, 3002008093.* Palo Alto: Electric Power Research Institute.

Folkard, S. (1997). Black times: Temporal determinants of transport safety. *Accident Analysis &*

*Prevention, 29*(4), 417-430.

Frodl, T., & O'Keane, V. (2013). How does the brain deal with cumulative stress? A review with focus on developmental stress, HPA axis function and hippocampal structure in humans. *Neurobiology of Disease, 52,* 24-37.

Galyean, W. (2006). Orthogonal PSF taxonomy for human reliability analysis. *Proceedings of the 8th International Conference on Probabilistic Safety Assessment and Management.*

Gersh, J. R., McKneely, J. A., & Remington, R. W. (2005). Cognitive engineering: Understanding human interaction with complex systems. *Johns Hopkins APL Technical Digest, 26*(4), 377-382.

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

Government Accountability Office. (2020). *Technology Readiness Assessment Guide: Best Practices for Evaluating the Readiness of a Technology for Use in Acquisition Programs and Projects, Report No. GAO-20-48G.* Washington, DC: Government Accountability Office.

GSE Systems. (2012). *RELAP5-HD: A high-definition RELAP5-3D application.* Sykesville, MD: GSE Systems.

Hänecke, K., Tiedemann, S., Nachreiner, F., & Grzech-Šukalo, H. (1998). Accident risk as a function of hour at work and time of day as determined from accident data and exposure models for the German working population. *Scandinavian Journal of Work, Environment & Health, 24*(Suppl. 3), 43-48.

Health and Safety Executive. (2014). *Fatigue and risk index calculator Ver 2.2 User Guidance*. Retrieved from https://www.yumpu.com/en/document/view/12949697/fatigue-and-risk-index-calculator-version-22-hse

Hollnagel, E. (2000). Looking for errors of omission and commission or The Hunting of the Snark revisited. *Reliab. Eng. Syst. Saf., 68*(2), 135-145.

Jang, I., Kim, Y., & Park, J. (2021). Investigating the Effect of Task Complexity on the Occurrence of Human Errors observed in a Nuclear Power Plant Full-Scope Simulator. *Reliability Engineering & System Safety, 214*, Article 107704.

Joe, J.C., & Boring, R.L. (2017). Using the Human Systems Simulation Laboratory at Idaho National Laboratory for safety focused research. *Advances in Intelligent Systems and Computing, 495,* 193-201.

Joe, J.C., Boring, R.L., Herberger, S., Miyake, T. , Mandelli, D., & Smith, C.L. (2015). *Proof-of-Concept Demonstrations for Computation-Based Human Reliability Analysis: Modeling Operator Performance During Flooding Scenarios, INL/EXT-15-36741.* Idaho Falls: Idaho National Laboratory.

Julius, J., Grobbelaar, J., Spiegel, D., & Rahn, F. (2005). *The EPRI HRA Calculator® User's Manual, version 3.0, TR-1008238*. Palo Alto: Electric Power Research Institute.

Jung, W., Park, J., Kim, J., & Ha, J. (2007). Analysis of an operators' performance time and its application to a human reliability analysis in nuclear power plants. *IEEE Transactions on Nuclear Science, 54*(5), 1801-1811.

Lew, R., Boring, R.L., & Ulrich, T.A. (2019). Task engine for job and user notification (TEJUN): A tool for prototyping computerized procedures. *Proceedings of the 11th Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies (NPIC&HMIT 2019),* pp. 932-940.

Light Water Reactor Sustainability Program. (2021). *Overview and Accomplishments: Sustaining*

*National Nuclear Interests.* Washington, DC: U.S. Department of Energy.

Lois, E., Dang, V.N., Forester, J., Broberg, H., Massaiu, S., Hildebrandt, M., Braarud, P.Ø., Parry, G., Julius, J., Boring, R., Männistö, I, and Bye, A. (2009). *International HRA Empirical study—Phase 1 Report, Description of Overall Approach and Pilot Phase Results from Comparing HRA methods to Simulator Performance Data, NUREG/IA-0216, Vol. 1.* Washington, DC: U.S. Nuclear Regulatory Commission.

Ma, Z., Parisi, C., Davis. C., Park, J., Boring, R., & Zhang, H. (2019). *Risk-Informed Analysis for an Enhanced Resilient PWR with ATF, FLEX, and Passive Cooling, INL/EXT-19-56151.* Idaho Falls: Idaho National Laboratory

Medema, H., Mohon, J., & Boring, R. (2021). Extracting human reliability findings from human factors studies in the Human Systems Simulation Laboratory. *2021 International Topical Meeting on Probabilistic Safety Assessment and Analysis (PSA 2021)*, pp. 98–107.

Nachreiner, F., Akkermann, S., & Haenecke, K. (2000). Fatal accident risk as a function of hours into work. *Arbeitswissenschaft in der betrieblichen Praxis, 17,* 19-24.

Newell, K.M, van Emmerik, R.E.A., & McDonald, P.V. (1989). Biomechanical constraints and action theory. *Human Movement Science, 8*, 403-409.

Park, J. (2014). Investigating the TACOM measure as a general tool for quantifying the complexity of procedure guided tasks. *Reliability Engineering & System Safety, 129,* 66-75.

Park, J., Arigi, A. M., & Kim, J. (2019). A comparison of the quantification aspects of human reliability analysis methods in nuclear power plants. *Annals of Nuclear Energy, 133*, 297-312.

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

Park, J., & Boring, R. L. (2021). Identification of performance shaping factors affecting subsequent human actions for dependence assessment in human reliability analysis. *Lecture Notes in Networks and Systems, 262,* 47–54.

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

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

Park, J., & Cho, S. (2010). Investigating the effect of task complexities on the response time of human operators to perform the emergency tasks of nuclear power plants. *Annals of Nuclear Energy, 37(9),* 1160-1171.

Park, J., & Jung, W. (2007). A study on the revision of the TACOM measure. *IEEE Transactions on Nuclear Science, 54*(6), 2666-2676.

Park, J., Jung, W., Ha, J., & Park, C. (2002). The step complexity measure for emergency operating procedures: Measure verification. *Reliability Engineering & System Safety, 77*(1), 45-59.

Park, J., Jung, W., & Kim, J. (2020). Inter-relationships between performance shaping factors for human reliability analysis of nuclear power plants. *Nuclear Engineering and Technology, 52*(1), 87-100.

Parry, G.W., Lydell, B.O.Y., Spurgin, A.J., Moieni, P., & Beare, A. (1992). *An Approach to the Analysis of Operator Actions in Probabilistic Risk Assessment, TR-100259.* Palo Alto: Electric Power

Research Institute.

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. *2021 International Topical Meeting on Probabilistic Safety Assessment and Analysis (PSA 2021).*

Permann, C.J., Gaston, D.R., Andrš, D., Carlsen, R.W., Kong, F., Lindsay, A.D., Miller, J.M., Peterson, J.W., Slaughter, A.E., Stonger, R.H., & Martineau, R.C. (2020). MOOSE: Enabling massively parallel multiphysics simulation. *SoftwareX, 11,* Article 100430.

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

Podofillini, L., Park, J., & Dang, V. N. (2013). Measuring the influence of task complexity on human error probability: an empirical evaluation. *Nuclear Engineering and Technology, 45*(2), 151-164.

Prescott, S., Smith, C., & Vang, L. (2018). EMRALD, Dynamic PRA for the traditional modeler. *Proceedings of the 14th International Probabilistic Safety Assessment and Management Conference.*

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

Rasmussen, J. (1980). Notes on human error analysis andp. In G. Apostolakis, S. Garribba, & G. Volta (Eds.), *Synthesis and Analysis Methods for Safety and Reliability Studies* (pp. 357-389). Boston, MA: Springer.

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

See, J. E., & Handley, (2019). *History and Current Status of Human Readiness Levels.* https://www.osti.gov/servlets/purl/164594

Shengli, W. (2021). Is human digital twin possible? *Computer Methods and Programs in Biomedicine Update, 1,* Article 100014.

Spencer, M., Robertson, K., & Folkard, S. (2006). The development of a fatigue/risk index for shiftworkers. *Health and Safety Executive Report, 446.*

St Germain, S., Boring, R., Banaseanu, G., Akl, Y., & Xu, M. (2016). Modification to the SPAR-H method to support HRA for Level 2 PSA. *13th International Conference on Probabilistic Safety Assessment and Management (PSAM 13)*, Paper A-112, pp. 1-9.

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

Torres, Y., Nadeau, S., & Landau, K. (2021). Application of SHERPA (Systematic Human Error Reduction and Prediction Approach) as an alternative to predict and prevent human error in manual assembly. *Congress of the 2021 International Ergonomics Association.*

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

Ulrich, T. A., Lew, R., Werner, S., & Boring, R. L. (2017b). 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.

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.