INL/EXT-21-64525

Light Water Reactor Sustainability Program

An Adaptable Software Toolkit for Dynamic Human Reliability Analysis: Progress Toward HUNTER 2

September 2021

U.S. Department of Energy

Office of Nuclear Energy

INL/EXT-21-64525

An Adaptable Software Toolkit for Dynamic Human Reliability Analysis: Progress Toward HUNTER 2

Ronald Boring, Thomas Ulrich, Jooyoung Park, Torrey Mortenson, Jeeyea Ahn, and Roger Lew

September 2021

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

This page intentionally left blank.

ABSTRACT

In response to a review of the technology maturity of risk tools in the U.S. Department of Energy's Light Water Reactor Sustainability Project, 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. 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.

ACKNOWLEDGEMENTS

The authors wish to thank the Risk-Informed System Analysis (RISA) Pathway of the U.S. Department of Energy's Light Water Reactor Sustainability 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 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.

CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS iv

ACRONYMS ix

1. INTRODUCTION 1
   1.1 Project Background 1
   1.2 Benefits of HUNTER as a Dynamic HRA Framework 2
   1.3 Report Structure 3

2. PREVIOUS HUNTER EFFORTS 3
   2.1 HUNTER 1 Framework 3
   2.2 GOMS-HRA 4
   2.3 SPAR-H Autocalculation 6
   2.4 Dynamic Dependency 7

3. EXPANDED HUNTER FRAMEWORK 8
   3.1 HUNTER Technology Readiness Level 8
   3.2 HUNTER's Adaptable Design Philosophy 11
   3.3 HUNTER Conceptual Framework 14
      3.3.1 Overview 14
      3.3.2 HUNTER Modules 15
      3.3.3 HUNTER Classes 16
      3.3.4 Special Considerations 17
      3.3.5 Relationship Between HUNTER 1 and HUNTER 2 Frameworks 17

4. HUNTER SOFTWARE IMPLEMENTATION DETAILS 18
   4.1 Background 18
   4.2 Input Data Structure and Analyst Workflow 19
   4.3 Modules 23
      4.3.1 Scheduler Module 23
      4.3.2 Human Reliability Module 23
      4.3.3 Environment Module 23
      4.3.4 Task Module 23

5. DEMONSTRATION AND FINDINGS 26
   5.1 SGTR Scenario 26
      5.1.1 SGTR Description 26
      5.1.2 SGTR Procedures 28
   5.2 Relevant Findings 29
      5.2.1 International HRA Empirical Study at Halden Reactor Project 29
      5.2.2 Task Complexity Score 33
      5.2.3 Complexity Time Multiplier for TACOM 34
   5.3 Demonstration SGTR Results 34

5.3.1 Method 34
5.3.2 Results of Model Runs 38
5.3.3 Discussion 43

6. DISCUSSION AND NEXT STEPS 43
6.1 Limitations 43
6.2 Future Work 44
6.2.1 Performance Shaping Factors 44
6.2.2 Advanced Programming Interface and Coupling to External Codes 45
6.2.3 Cognitive Modeling 47
6.2.4 HEPs 48
6.2.5 Automatic Procedure Parsing 48
6.2.6 Additional Use Cases 48

7. REFERENCES 49

FIGURES

Figure 1. The original HUNTER framework (adapted from Boring et al. 2016).....4
Figure 2. GOMS-HRA cognitive model (from Boring, Ulrich, and Rasmussen 2018).....5
Figure 3. Technology readiness levels (from See and Handley 2019).....9
Figure 4. Conceptual modules (in black) and classes (in blue) of HUNTER 2.....15
Figure 5. Crosswalk of HUNTER 1 to HUNTER 2.....18
Figure 6. Region of the HUNTER input file demonstrating the use of dummy coding to represent additional elements under the same procedure step.....20
Figure 7. Procedure substep region of the HUNTER input file.....20
Figure 8. Point region of the HUNTER input file denoting names of plant parameters.....21
Figure 9. GOMS-HRA task-level primitives found in HUNTER input file.....21
Figure 10. Mapping of the individual, task, and environment modules to the classes in the HUNTER software implementation.....22
Figure 11. Assertion class in HUNTER acting (a) directly as a procedure step and (b) indirectly to trigger procedure substeps.....24
Figure 12. Transition path logic evaluated by the assertion and then stored in the results class.....25
Figure 13. Descriptive feature of steam generator with SGTR.....26
Figure 14. Example of generic Level 1 PRA ET for SGTR (adapted from NUREG-2195).....27
Figure 15. Performance time data for HFE-1, -2, and -3 for two SGTR cases for Crews A–N in the Halden study.....31
Figure 16. GOMS-HRA task-level primitive mapping to Procedure EOP-E0.....37
Figure 17. Log file depicting showing overall execution of the SGTR model in HUNTER.....38
Figure 18. Log file showing stepwise execution of the SGTR model in HUNTER.....39
Figure 19. Log showing substep execution of the SGTR model in HUNTER.....40
Figure 20. Screenshot of data file output generated by the HUNTER Python code.....41
Figure 21. Distribution of times for HUNTER SGTR model runs.....42
Figure 22. Extension of the PSF concept from static to dynamic HRA.....44

TABLES

Table 1. Time information for each task-level primitive (from Boring et al. 2016). 5
Table 2. HEP information for each task-level primitive (from Boring and Rasmussen 2016; Boring, Ulrich, and Rasmussen 2018). 6
Table 3. Dynamic functions that affect dependency (from Boring et al. 2016). 8
Table 4. Description of TRLs for RISA toolkit (from Choi [2021]). 10
Table 5. HUNTER framework technology maturity assessment result (from Choi [2020]). 11
Table 6. HUNTER objectives and modeling considerations. 14
Table 7. HFEs defined from the ET of the SGTR scenario. 28
Table 8. Mapping of HFEs to Procedures. 29
Table 9. HFEs for two SGTR scenarios from the Halden study. 30
Table 10. Identified negative driving factors for HFEs in the Halden study 32
Table 11. Measures of TACOM (from Park et al. 2007). 33
Table 12. Mean times for SGTR scenario runs for Halden and HUNTER. 42

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
CD	core damage
C_F	checking in field
CST	condensate storage tank
CSV	comma separate 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
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

AN ADAPTABLE SOFTWARE TOOLKIT FOR DYNAMIC HUMAN RELIABILITY ANALYSIS: PROGRESS TOWARD HUNTER 2

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 while simultaneously lowering the demands on maintenance staff (Agarwal et al. 2021).

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. The name HUNTER is meant as a counterpart to the various animal 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 (MOOSE) (Permann et al. 2020). These tool names playfully combine to become tools like RAVEN-HUNTER or MOOSE-HUNTER.[a]

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

[a] 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.

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 naamed 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, but additional refinement is planned in the future toward a releasable code for industry. 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 usability 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 as a result of different human responses. This capability is especially useful for emerging areas of interest in risk modeling, such as severe accidents, HRA for human interactions with advanced technologies like digital and automated human-system interfaces, balance-of-plant activities beyond the main control room that are the main focus of conventional HRA methods, 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.

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 HUNTER
- Section 5 provides background and a demonstration of an SGTR scenario
- Section 6 considers shortcomings and next steps for HUNTER development activities.

2. PREVIOUS HUNTER EFFORTS

2.1 HUNTER 1 Framework

The HUNTER framework is a computational HRA[b] 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 as well as 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 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.

[b] 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.

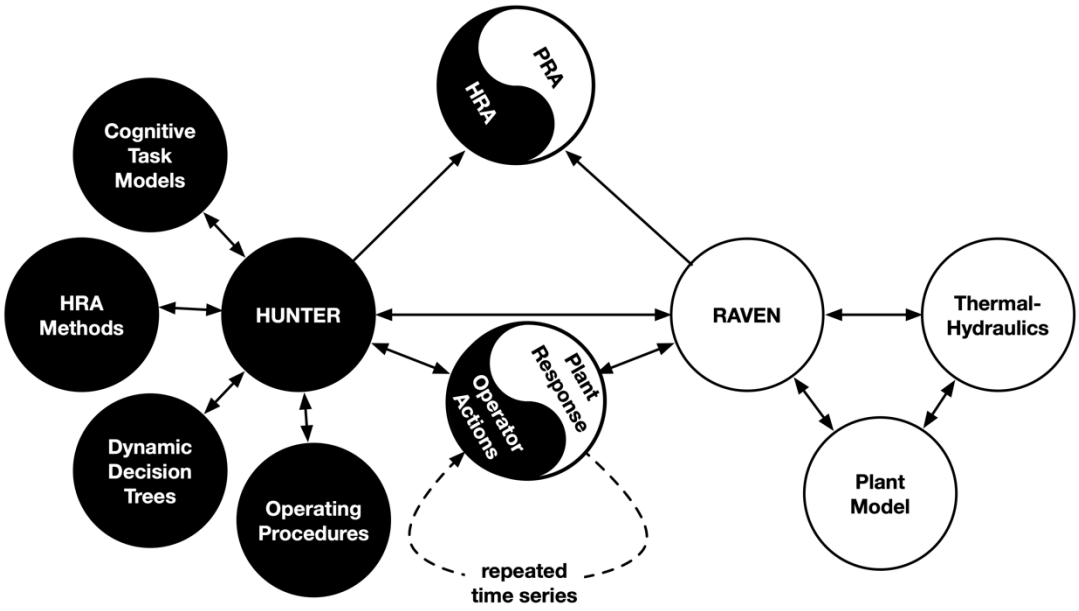


Figure 1. The original HUNTER framework (adapted from Boring et al. 2016).

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 the 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, the 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).

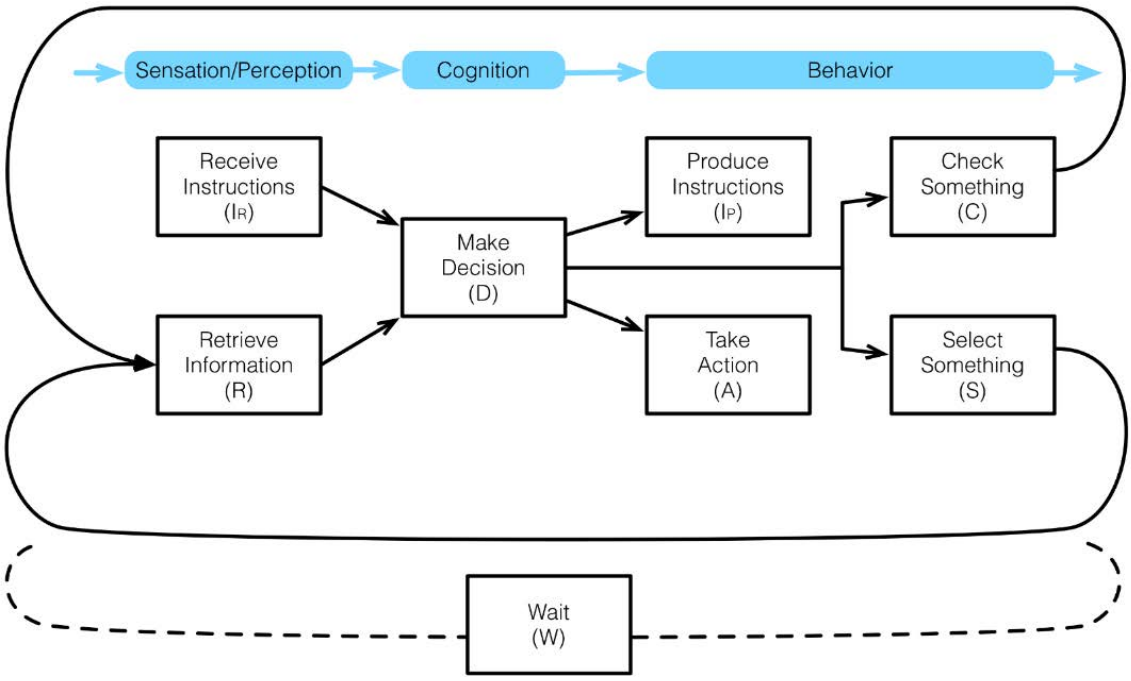


Figure 2. GOMS-HRA cognitive model (from Boring, Ulrich, and Rasmussen 2018).

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

Second, 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 and the Human Systems Simulation Laboratory (HSSL) at INL (Joe and Boring 2017). 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, Blackman, Marble, Byers, and Smith 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. It 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, Grobbelaar, Spiegel, and Rahn 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:

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

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, Dang, Zio, Baraldi, and Librizzi 2010; Swain and Guttmann 1983). Normally, dependency increases the overall HEP. Thus, it plays an important role in reasonably accounting for human actions in the context of PRAs and prevents PRA results from being estimated too optimistically based on the HRA results. 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, HRA 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 the 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. Table 3 shows dependency factors, including lag and linger.

Table 3. Dynamic functions that affect dependency (from Boring et al. 2016).

Dynamic Dependency 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	$PSF(t) = PSF(0) \quad for\ t \gg t_N$

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 4. 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.

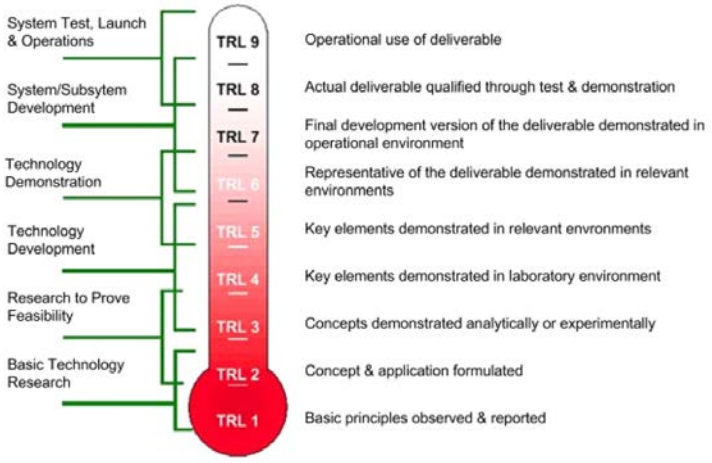


Figure 3. Technology readiness levels (from See and Handley 2019).

Table 5 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 QA and higher credibility
- Reorganize and review development roadmap to build standalone software
- Develop coupling module for physics-based RISA Toolkit (i.e., RELAP5-3D)
- Create necessary manuals and supporting documents for industrial deployment.

Table 4. 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 5. 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

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 Section 4 and a demonstration analysis reviewed in Section 5. 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. At the completion of this software development endeavor, future activities will begin to address the remaining requirements.

3.2 HUNTER's Adaptable Design Philosophy

As outlined in Section 2, the original HUNTER framework was a collection of various 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 acknowledgement 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 side 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 quicker 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 sequences 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 6.

Table 6. HUNTER objectives and modeling considerations.

OBJECTIVES	MODELING CONSIDERATIONS			
	Knowledge	**Fidelity**	**Efficiency**	**Purpose**
Flexibility **Modularity** **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, and the environment. This delineation of human activities was generally borne out in a review of Galyean (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 bundles of specific instances of tangible things, 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.

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 the conceptual model. 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 Section 4. The functional modules and classes are described next.

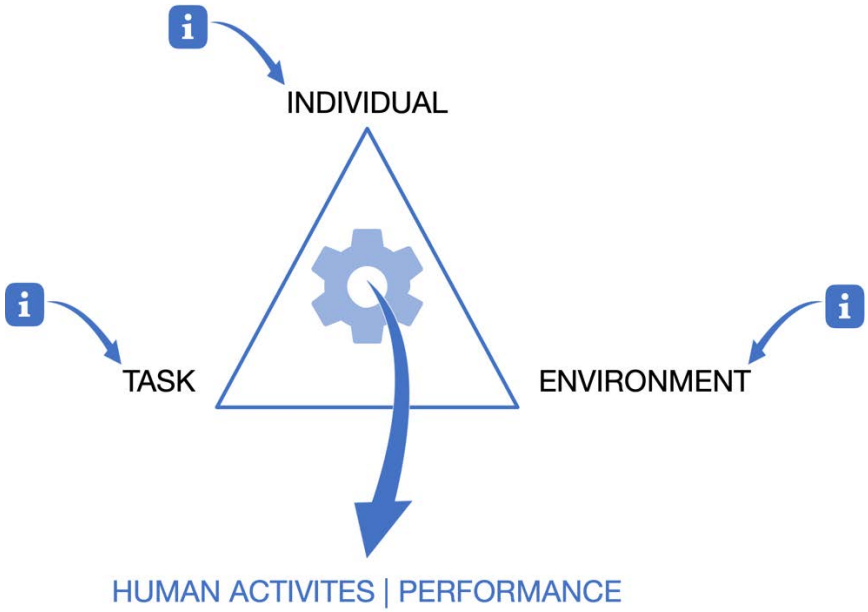


Figure 4. Conceptual modules (in black) and classes (in blue) of HUNTER 2.

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., ) 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 module 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 generic 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.

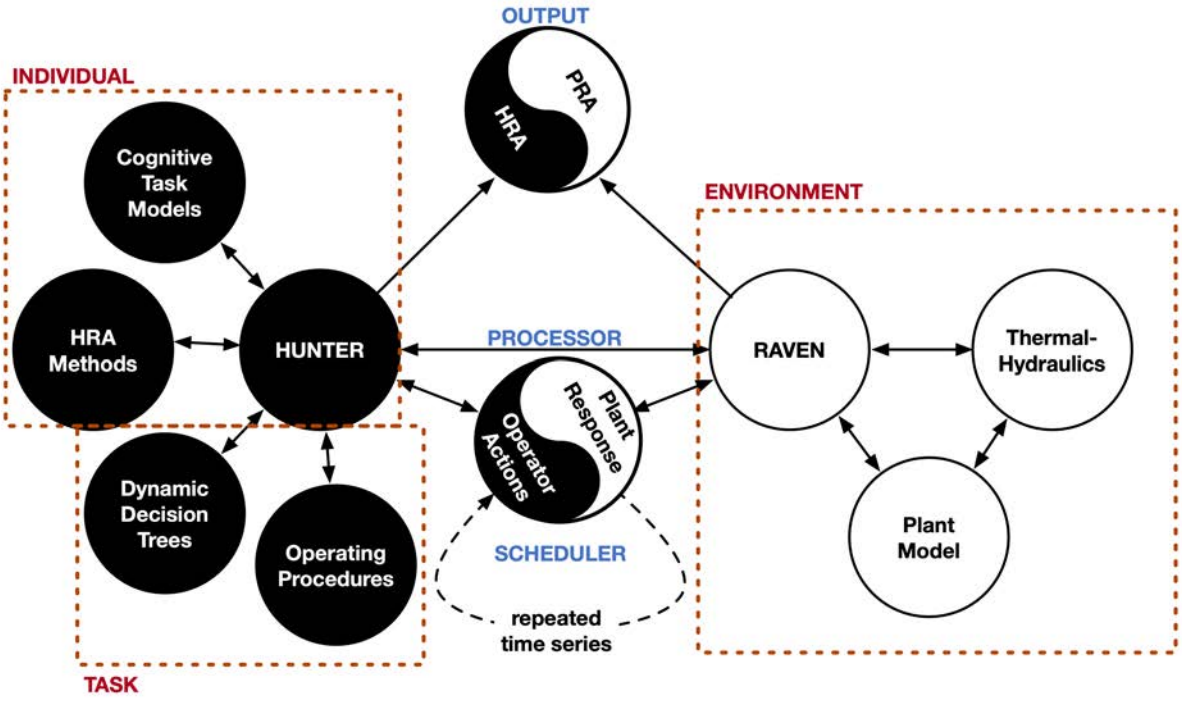


Figure 5. Crosswalk of HUNTER 1 to HUNTER 2.

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 section summarizes the first implementation of this framework as standalone software.

4. HUNTER SOFTWARE IMPLEMENTATION DETAILS

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, 1.0.0, the analyst must manually enter procedures, procedures steps, and substeps into the CSV file. 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 towards 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.

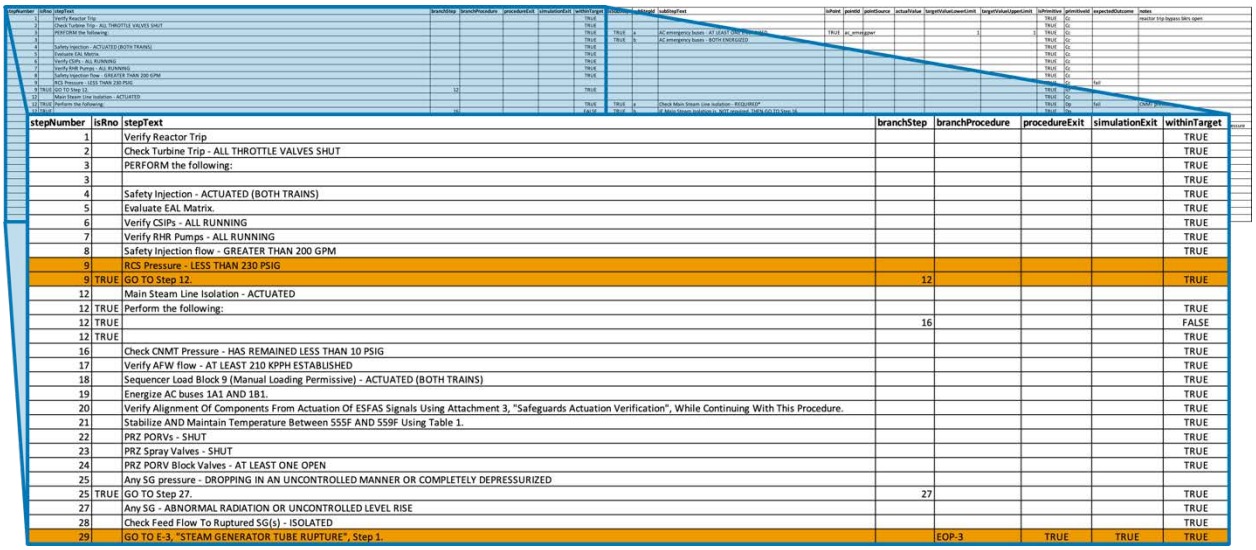
Each row of the input deck represents a procedure step-level item and must have a step number. This step number is used to group any additional related rows (e.g., all rows tagged as 12 are processed by the parser as a collection of sub-elements to be grouped within Step 12). As the number of items contained 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*.

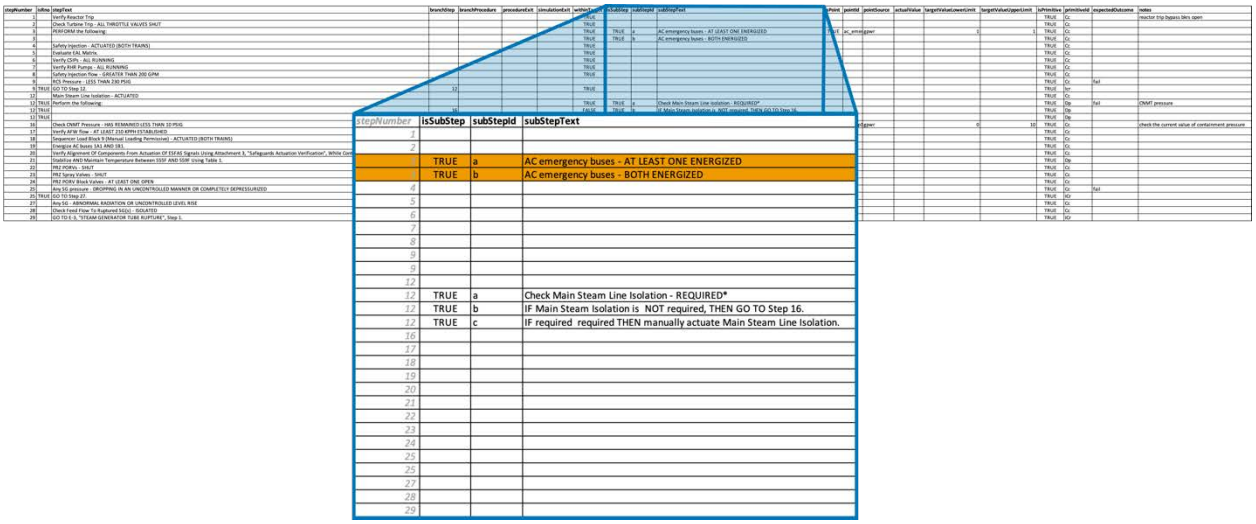
- *Point*—refers to addresses or names for parameters in external plant simulations. This information may also be dummy coded if no plant simulation is referenced.
- *Primitive*—refers to the GOMS-HRA task-level primitives, which the HUNTER code uses to determine time durations and nominal HEPs for tasks.



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.

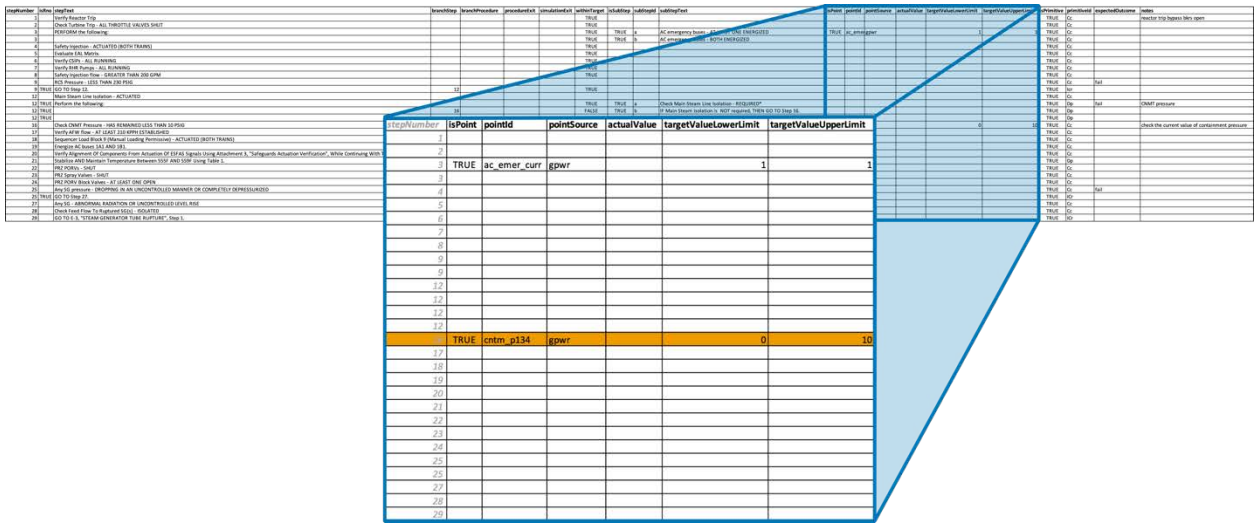


stepNumber	isSubStep	subStepId	subStepText
12	TRUE	a	Check Main Steam Line Isolation - REQUIRED*
12	TRUE	b	IF Main Steam Isolation is NOT required, THEN GO TO Step 16.
12	TRUE	c	IF required required THEN manually actuate Main Steam Line Isolation.

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

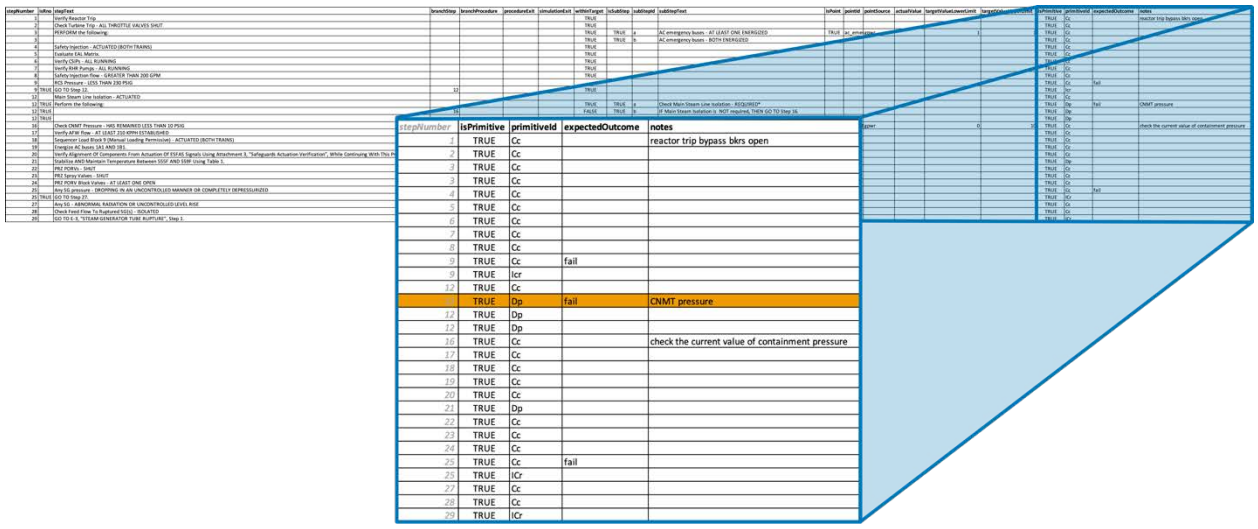
Figure 7. Procedure substep region of the HUNTER input file.

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
3	TRUE	ac_emer_curr	gpwr		1	1
10	TRUE	cntm_p134	gpwr		0	10

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		
10	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 labelled 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.

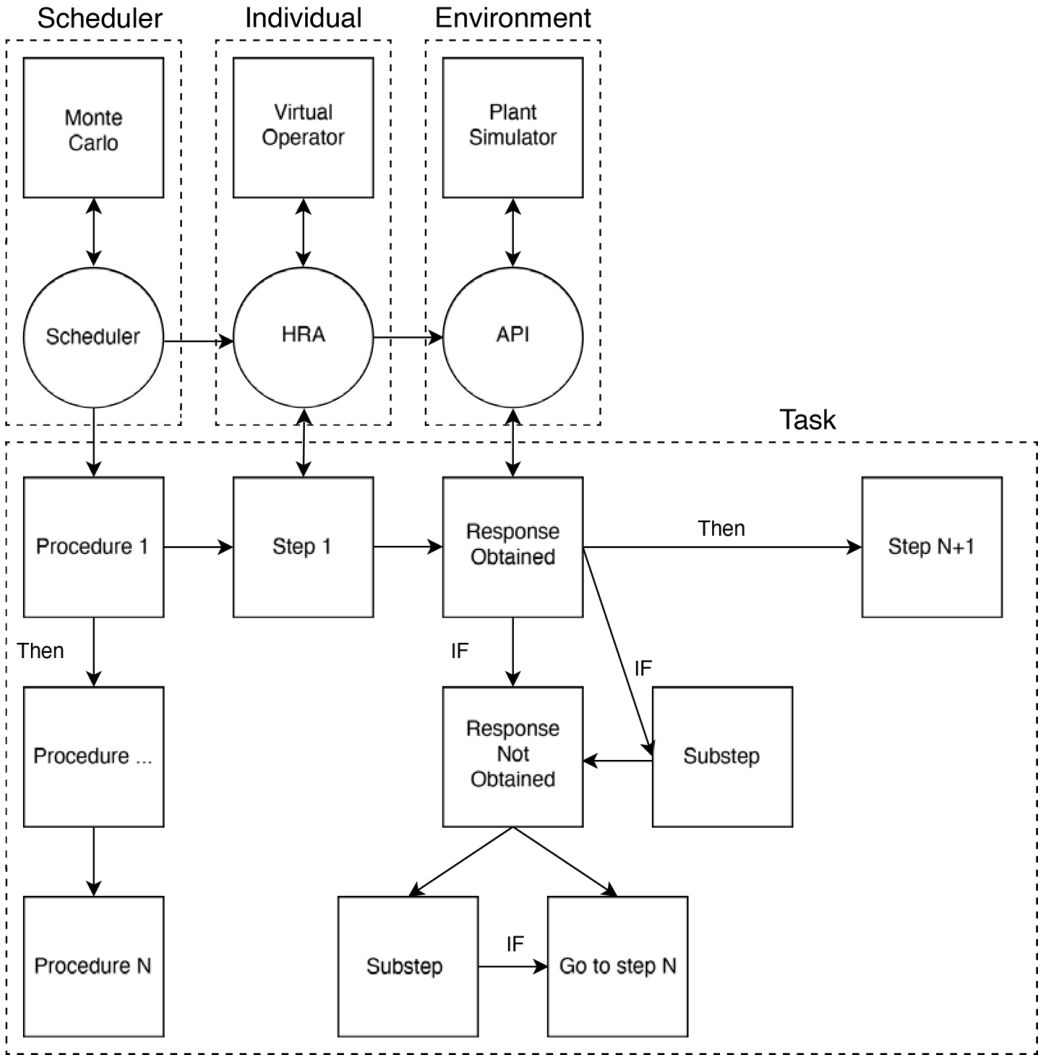


Figure 10. Mapping of the individual, task, and environment modules to the classes in the HUNTER software implementation.

4.3 Modules

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 each module and the Python classes contained within them. 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 Module

The scheduler module contains functionality defined through several classes to perform Monte Carlo based simulations of the task defined through the CSV input files. 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 Human Reliability Module

The HRA sampler class contains predefined (not by the analyst) GOMS-HRA (Boring and Rasmussen 2016) and SPAR-H PSF classes (Gertman et al. 2005). The GOMS-HRA class stores the identifier, HEP, and time distribution for each GOMS-HRA task-level primitive. The analyst can then reference any one of the GOMS-HRA task-level primitives simply by adding the identifier to each procedure step or substep. The PSF class stores the type, level, and multiplier values as defined in SPAR-H for its eight PSFs and can be used similarly by the analyst, though the PSFs also require setting the identifier and level for each procedure step or substep. Additionally, the sampler class contains functions that support random sampling from the time distributions to generate execution times for each step based on the GOMS-HRA task-level primitive assigned. A more in-depth description of the GOMS-HRA task-level primitive and SPAR-H PSF classes can be found in the subsequent section on the assertion class.

4.3.3 Environment Module

The environment module primary element is the API class, which contains the code that allows the scheduler to evaluate plant states for individual parameters defined in the point class. Similar to the GOMS-HRA task-level primitive and PSF identifier coding scheme for the procedure steps and substeps, the analyst can define simulator parameters as points. The point class stores the point name, the actual process value, the upper acceptable, and lower acceptable process values. The API class does not perform the evaluation itself (which is performed within the task module). It simply polls the plant representative simulator to determine the value of the parameter at the timepoint when the parameter is needed to evaluate the procedure step or substep success or failure.

4.3.4 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.

4.3.4.1 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.

4.3.4.2 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.

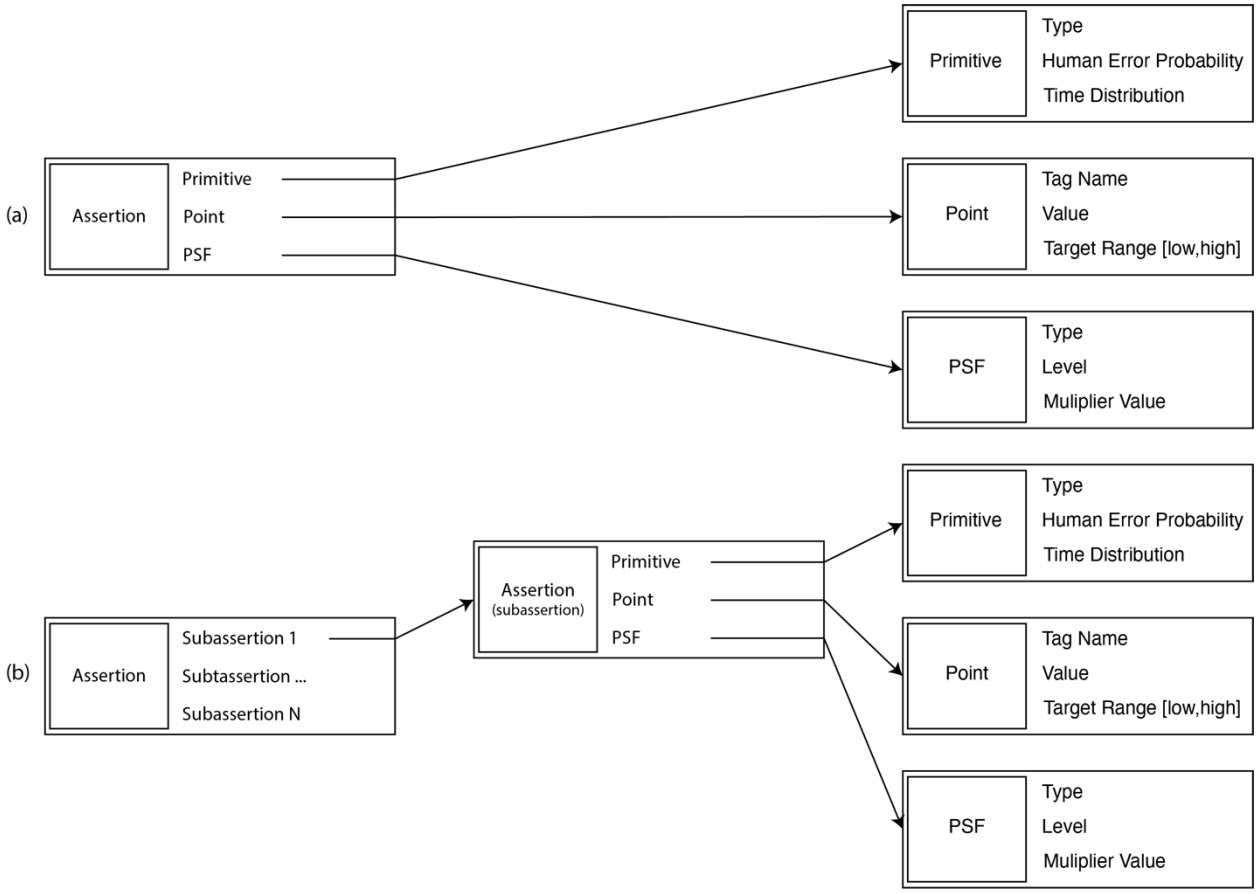


Figure 11. Assertion class in HUNTER acting (a) directly as a procedure step and (b) indirectly to trigger procedure substeps.

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. Furthermore, 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.

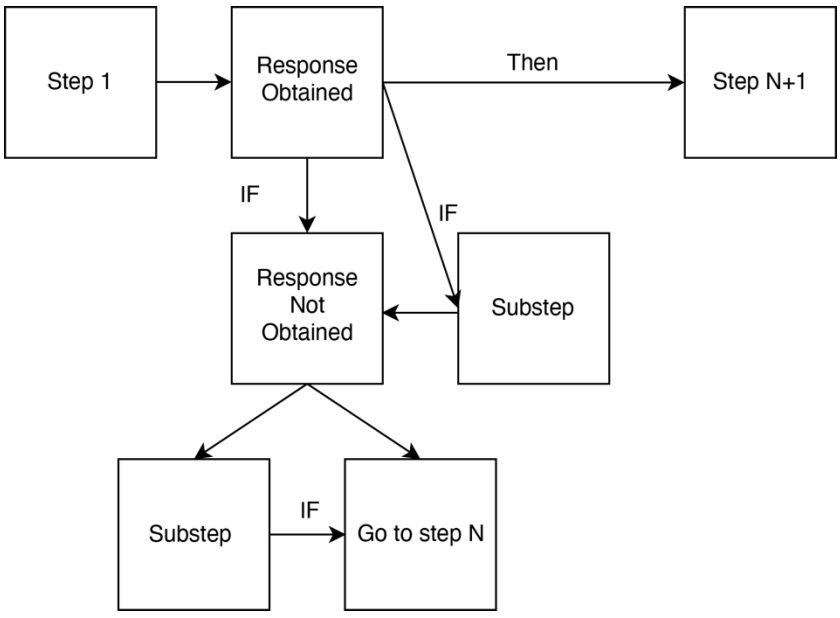


Figure 12. Transition path logic evaluated by the assertion and then stored in the results class.

4.3.4.3 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.

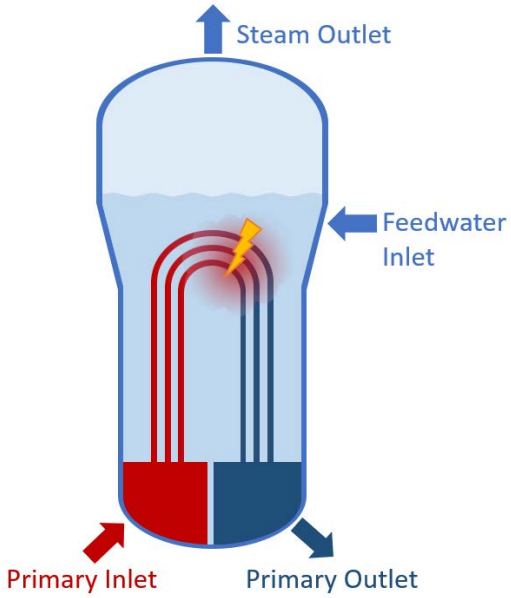


Figure 13. Descriptive feature of steam generator with SGTR.

5. DEMONSTRATION AND FINDINGS

5.1 SGTR Scenario

5.1.1 SGTR Description

An SGTR is portrayed in Figure 13. 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 14 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 7 shows examples of the possible HFEs, although specific modeling scenarios and specific plant PRAs may highlight different HFEs than this list.

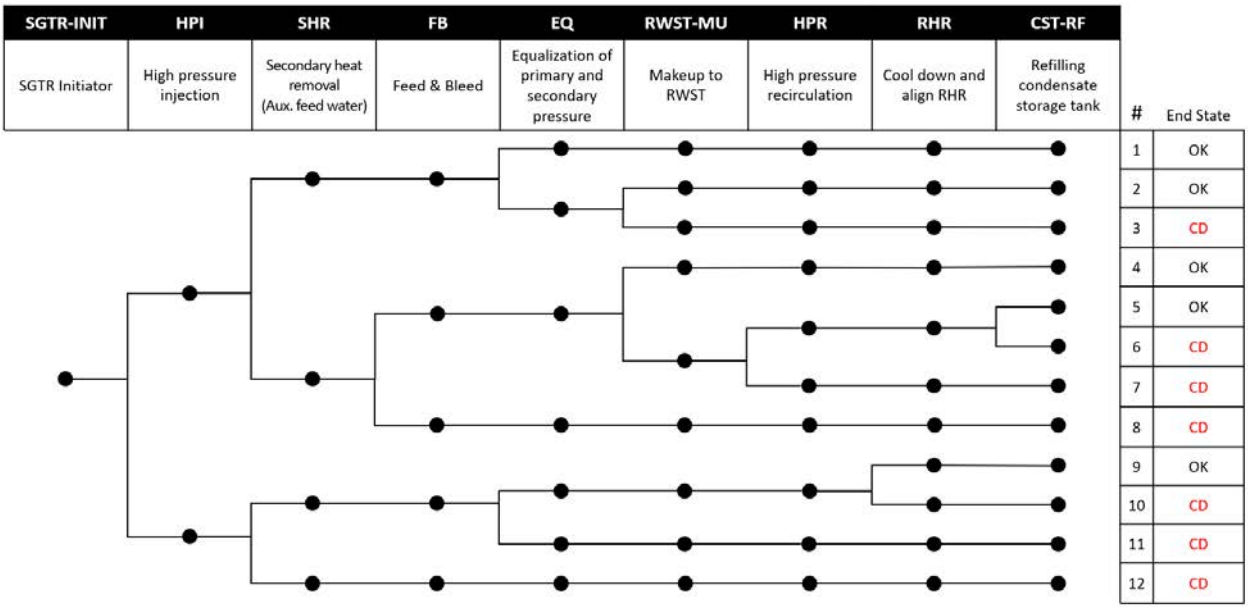


Figure 14. Example of generic Level 1 PRA ET for SGTR (adapted from NUREG-2195).

The following headings define the top branches for the event tree in Figure 14:

- 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 7. 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.

5.1.2 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 8 summarizes the procedure steps per each HFE.

Table 8. 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” 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” 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” Steps in OP-137 Section 8.1, “Auxiliary Feedwater System”

5.2 Relevant Findings

5.2.1 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 9. 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 9. 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 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 15). 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.

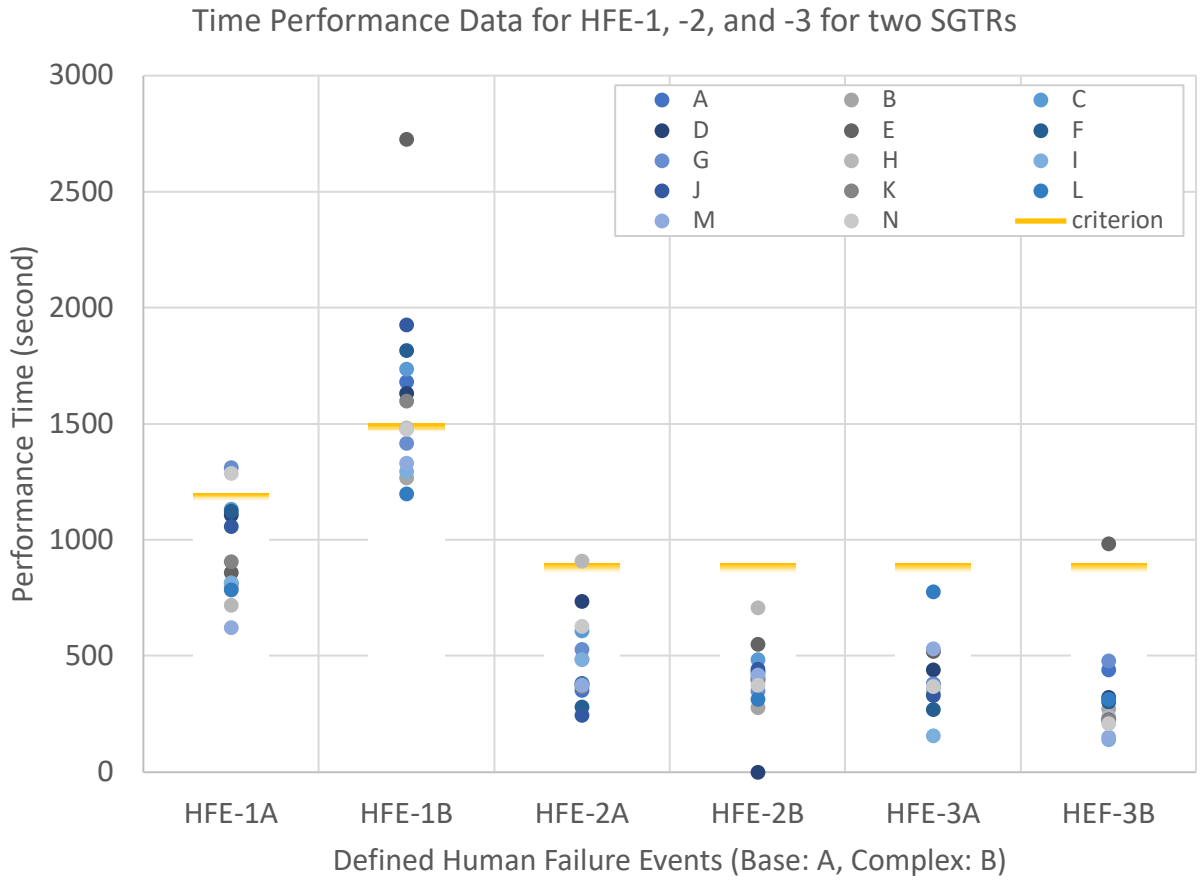


Figure 15. Performance time data for HFE-1, -2, and -3 for two SGTR cases for Crews A–N in the Halden study.

Table 10. Identified negative driving factors for HFEs in the Halden study.

Required Action	Base case	Complex case
HFE-1 (Faulted SG isolation)	- Execution complexity	– Scenario complexity – Procedural guidance – Execution complexity - Adequacy of time - Work processes
HFE-2 (RCS cooldown)	- Scenario complexity - Execution complexity - Procedural guidance - Team dynamics	- Stress - Scenario complexity - Execution complexity - Team dynamics
HFE-3 (RCS depressurization)	- Stress - Execution complexity - Team dynamics	- Stress - Scenario complexity - Execution complexity - 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 10). 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 15 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.

Table 11. 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	$R-TACOM = \sqrt{w_{TS} \times TS^2 + w_{TR} \times TR^2 + w_{TU} \times TU^2}$ $= \sqrt{0.621 \times TS^2 + 0.239 \times TR^2 + 0.140 \times TU^2}$ $where, \quad \begin{cases} TS = 0.716 \times SIC + 0.284 \times SSC \\ TR = 0.891 \times SLC + 0.109 \times AHC \\ TU = EDC \end{cases}$
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.

5.2.2 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 11 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.

5.2.3 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:

$$Response\ time = 1.340 \cdot e^{0.987 \cdot TACOM} \quad (2)$$

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

$$Time = 2.918 \cdot e^{(0.986 \cdot TACOM)} \quad \cdots (upper\ 95\%\ prediction\ limit) \quad (3)$$

$$Time = 0.617 \cdot e^{(0.986 \cdot TACOM)} \quad \cdots (lower\ 95\%\ prediction\ limit) \quad (4)$$

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.

5.3 Demonstration SGTR Results

5.3.1 Method

5.3.1.1 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.

5.3.1.2 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.

5.3.1.3 Manipulation of Complexity

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:

$$t_{complex(HFE1B)} = 1.664\, t_{base(HFE1A)} \tag{5}$$

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:

$$t_{complex(E0)} = 2.416\, t_{base(E0)} \quad (6)$$

5.3.1.4 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 16). 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.

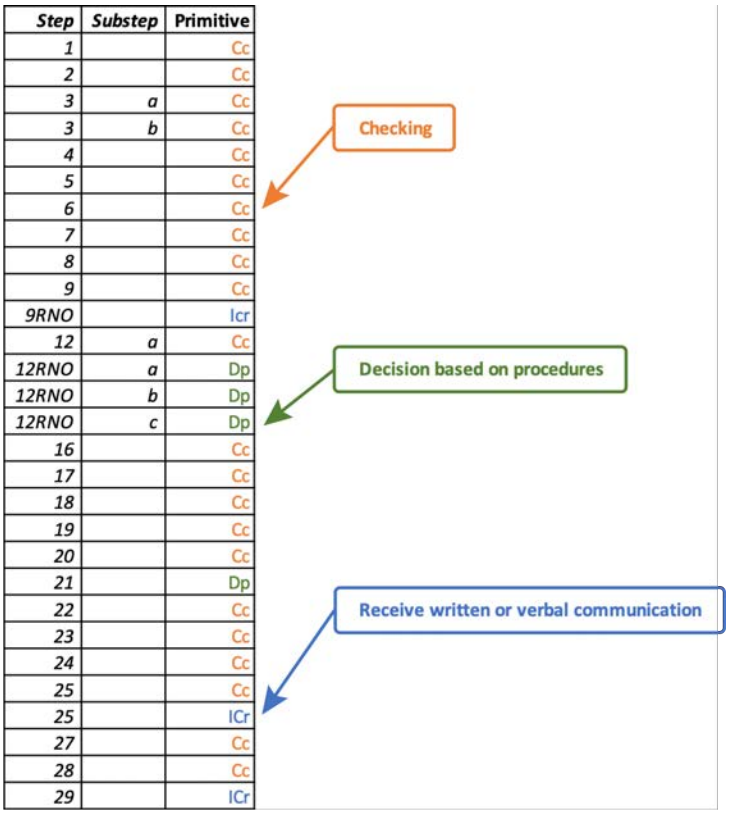


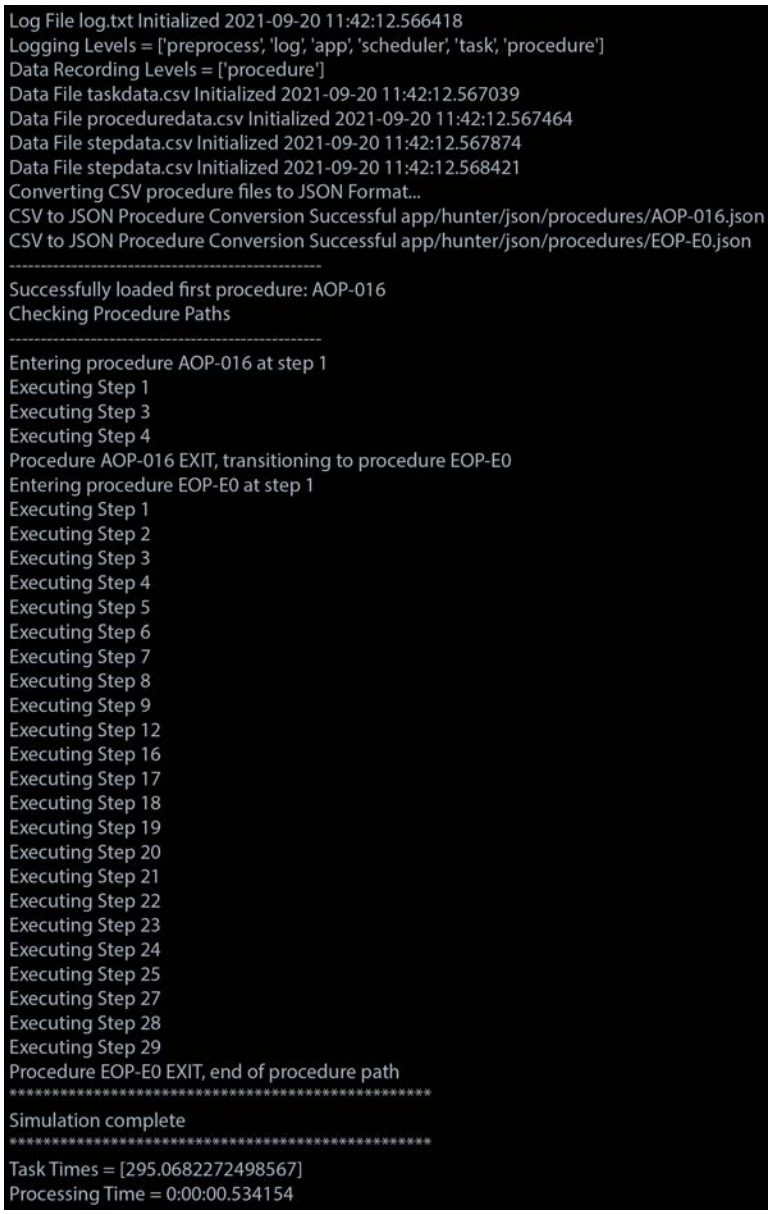
Figure 16. GOMS-HRA task-level primitive mapping to Procedure EOP-E0.

5.3.2 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.

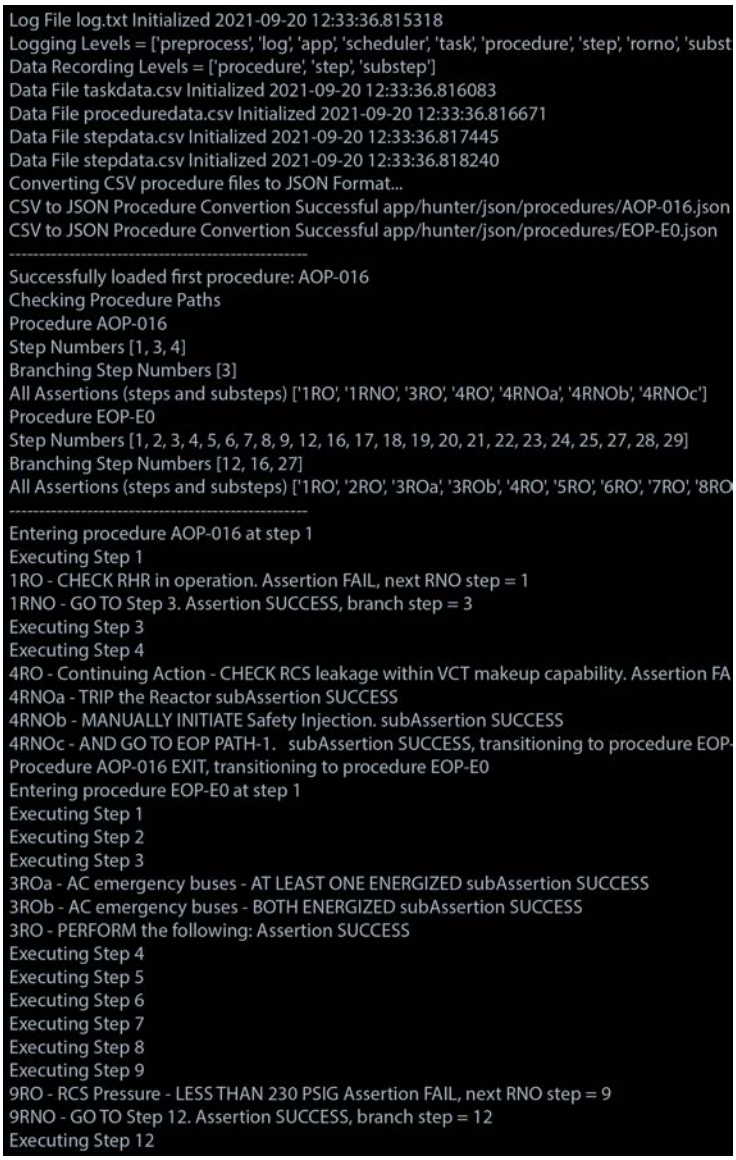
Note: No procedure, step, or substep level information is reported, but the overall completion times for each simulation run are recorded.

Figure 17. Log file depicting showing overall execution of the SGTR model in HUNTER.



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

Figure 18. Log file showing stepwise execution of the SGTR model in HUNTER.



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 19. Log showing substep execution of the SGTR model in HUNTER.

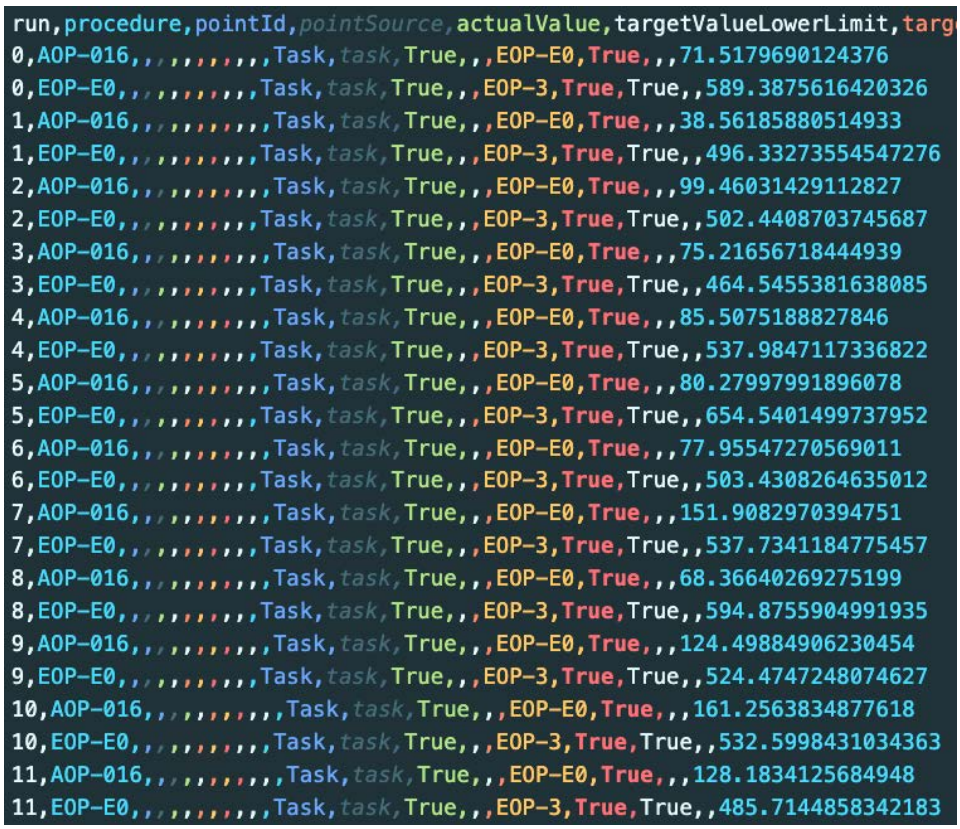
5.3.2.1 Data Output

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 17, 18, and 19 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.

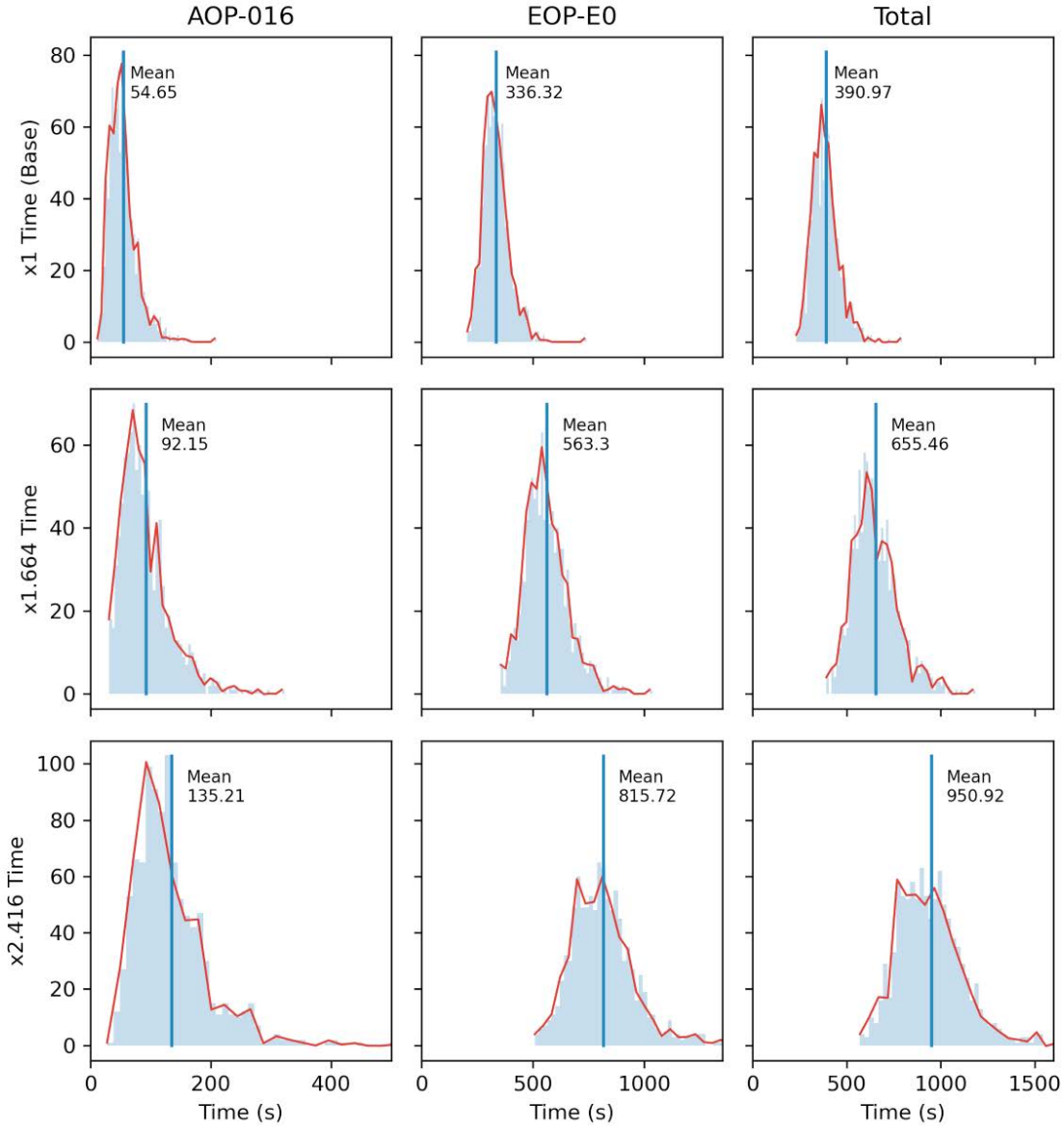
The data output can also be configured to record the simulation run. An example of data recorded at the simulation level can be seen in Figure 20, which depicts a portion of the actual data file that was used to generate the distributions shown in Figure 21. Each simulation run and each procedure within 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 20. 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 21. Distribution of times for HUNTER SGTR model runs.

Table 12. 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

5.3.2.2 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 21 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 12 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.

5.3.3 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.

6. DISCUSSION AND NEXT STEPS

6.1 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.

6.2 Future Work

6.2.1 Performance Shaping Factors

This project has researched how to model PSFs in the context of dynamic HRA and apply them within the HUNTER framework. In the HUNTER framework, the PSF module plays a role in reflecting PSF effects to outputs from GOMS-HRA (e.g., nominal HEPs or time), then providing the modified values to other HUNTER modules. Accordingly, the PSF module in the HUNTER framework should have two autocalculation functions: the PSF rating function and the PSF quantification function. The PSF rating function is responsible for automatically evaluating PSF levels based on information such as procedure instructions or plant response data from thermal hydraulic codes. How to extract useful information for PSF ratings using the given information and integrate them into a PSF level selection within the module is the key point of this function. In the PSF quantification function, the selected PSF levels are used for estimating final HEPs and time required for operators. This function needs to sufficiently reflect PSF characteristics that have not been considered in existing static HRA. For example, dynamic PSFs may be able to be extended so that PSFs affect the PSFs of the sequential human actions. Recall lag and linger effects from Table 3. Original PSFs in static HRA have only been considered for quantifying a single HFE as shown in Figure 22, not for carryover effects across HFEs.

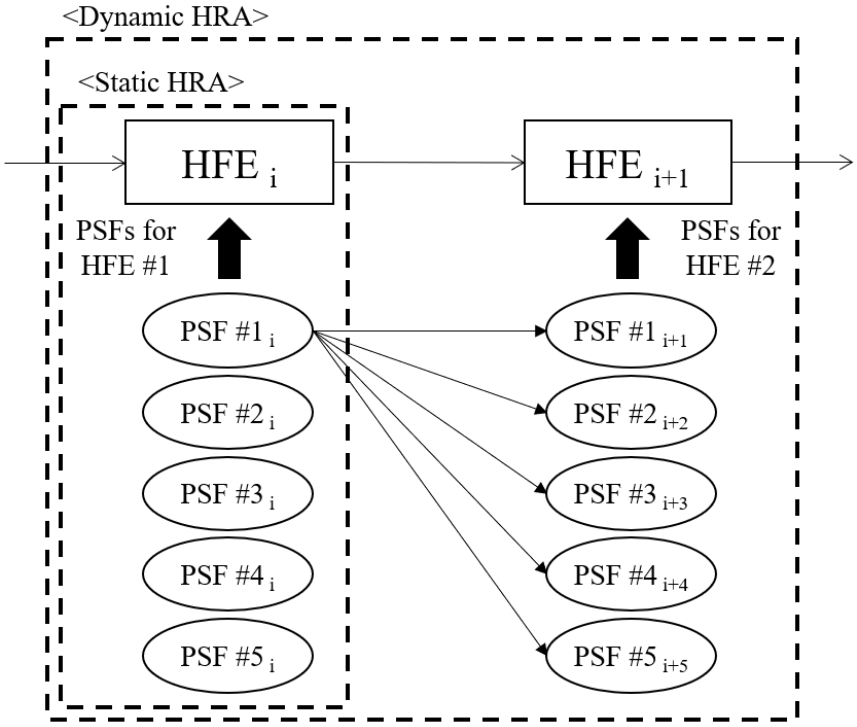


Figure 22. Extension of the PSF concept from static to dynamic HRA.

To date, our research team has studied PSFs in dynamic HRA based on the SPAR-H PSFs with the assumption that PSFs affect the PSFs of the sequential human actions. As a representative static HRA method, SPAR-H suggests eight PSFs: stress/stressor, available time, work processes, fitness for duty, complexity, ergonomics/human-system interface (HSI), experience/training, and procedures. In terms of dynamic HRA, we postulate that the SPAR-H PSFs are classified into two groups: PSF Type A (dependent PSFs) and PSF Type B (independent PSFs). The former group consists of four PSFs (i.e., stress/stressors, fitness for duty, work processes, and available time), which are dominant throughout an event scenario (or dependent on different HFEs). The latter group includes complexity, experience/training, ergonomics/HSI, and procedures that are dominant to the specific situation in which each human action is performed (or independent of different HFEs). For example, the stress level of an HFE may affect that of a subsequent HFE, while the procedure level of an HFE may not influence that of a subsequent HFE.

This research requires an ongoing effort on how to develop the PSF rating function and the PSF quantification function for each PSF in dynamic HRA. In particular, the effects of PSFs, not only on HEP calculations but also on other measures like time to complete tasks and decision-making outcomes, need to be modeled. Additionally, interdependencies between PSFs are important to understand and model. The relationship of PSFs across HFEs will greatly influence the calculated HEPs and will prove an important consideration in the aggregation of task-level data to HFEs. There is much foundational work to be done to model and validate the various effects of PSFs on human performance.

6.2.2 Advanced Programming Interface and Coupling to External Codes

A tenet of the HUNTER architecture is adaptability. This adaptability stems from the notion that the modules can be exchanged with other modules to achieve different goals like increased modeling efficiency or completeness. In the demonstration presented in Section 5, the environment module used full-scope simulator logs for particular parameters from previously run SGTR scenarios in the HSSL to drive the dynamic progression. This approach sufficed for the purposes of a proof-of-concept demonstration, but it did not fully realize the envisioned coupling of HUNTER to external simulation and modeling codes. Using presampled parameters does not allow flexible responses, and the model can only follow a predetermined route for which logged data are available.

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. 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. 2021). 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.
- *RELAP5-3D*—the Reactor Excursion and Leak Analysis Program (RELAP; Aumiller, Tomlinson, and Bauer 2001) is the foundational thermal hydraulic software that drives many full-scope simulators and supports modeling of advanced phenomena at existing and new NPPs. RELAP functions based on input decks that set up the model runs. This asynchronous mode of operation is not directly compatible with the bidirectional synchronous communication desired for HUNTER. The earlier incarnation of HUNTER used RAVEN (Rabiti et al. 2017) as middleware to serve as the API, first with the experimental RELAP7 and later with the released version of RELAP, namely RELAP5-3D. RAVEN used time estimates for tasks as inputs to RELAP in a Monte Carlo simulation. The limitation of this approach is that human action in an NPP is predicated by plant states, which are influenced by prior human actions. Plant operations are necessarily recursive, and it becomes challenging to model complex human-plant interactions a priori. Thus, the goal of coupling HUNTER with RELAP is to facilitate synchronous coupling, where human and plant models provide iterative feedback loops that drive the course of actions. An API between HUNTER and RAVEN may continue to use RAVEN as middleware if suitable; otherwise, a standalone API will be developed. The advantage of RELAP models to serve as the external environment module in HUNTER is the ability to customize the plant model and streamline for particular model applications.
- *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.

6.2.3 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.

Initial estimates of the cognitive modeling landscape have suggested that the Instance-Based Learning Theory (IBLT) shows promise in understanding the types of errors and decisions of greatest concern to HRA (Gonzalez et al. 2003). Additionally, IBLT has been used with success in various risk-focused

industries (Ben-Asher and Gonzalez 2015; Dutt, Ahn, and Gonzalez 2013; Gonzalez, Vanyukov, and Martin 2005). It is important to recall that a fully formed HRA method grounded on cognitive modeling has not yet been put forward. There may be some areas where the choice of IBLT may need to be adjusted as the models attempt to better capture human errors as they occur, and final predictive values are derived from those operations. However, the authors argue that this focus on decision-making and cognition is a positive step forward as the HUNTER framework continues to grow and develop.

6.2.4 HEPs

As discussed in Section 5, the present implementation of HUNTER is primarily concentrated with 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.

6.2.5 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.

6.2.6 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.

7. 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.

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.

Ben-Asher, N., & Gonzalez, C. (2015). Effects of cyber security knowledge on attack detection. *Comput. Human Behav., 48*, 51-61.

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. (2021). A microworld framework for advanced control room design. *Proceedings of the 12th Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies (NPIC&HMIT 2021)*, pp. 14-17.
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.
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. (2020). *Assessment of Verification and Validation Status—EMRALD and HUNTER, INL/EXT-20-59904.* 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.
Dougherty, E. M., Jr. (1990). Human reliability analysis—where shouldst thou turn? *Reliab. Eng. Syst. Saf., 29*(3), 283-299.
Dutt, V., Ahn, Y.-S., & Gonzalez, C. (2013). Cyber situation awareness: modeling detection of cyber attacks with instance-based learning theory. *Human. Factors, 55*(3), 605-618.
Galyean, W. (2006). Orthogonal PSF taxonomy for human reliability analysis. *Proceedings of the 8th International Conference on Probabilistic Safety Assessment and Management.*
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.
Gonzalez, C., Lerch, J.F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science, 27*(4), 591-635.
Gonzalez, C., Vanyukov, P., & Martin, M.K. (2005). The use of microworlds to study dynamic decision making. *Comput. Human Behav., 21*(2), 273-286.
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.

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.

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.

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)*.

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., 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

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.