The HUNTER Dynamic Human Reliability Analysis Tool: Overview of the Enhanced Framework for Modeling Human Digital Twins

Ronald Boring, Thomas Ulrich, Jooyoung Park, Yunyeong Heo, and Jeeyea Ahn
Human Factors and Reliability Department, Idaho National Laboratory, Idaho Falls, Idaho, USA
{Ronald.Boring,Thomas.Ulrich,Jooyoung.Park,Yunyeong.Heo,Jeeyea.Ahn}@inl.gov

Abstract: The Human Unimodel for Nuclear Technology to Enhance Reliability (HUNTER) was previously developed as a simplified test case for dynamic human reliability analysis (HRA). HUNTER 1 paired a dynamicized version of the SPAR-H HRA method that autocalculated the effects of performance shaping factors (PSFs), an implementation of the GOMS-HRA method to compute time for modeled human tasks, and an interface between the RAVEN modeling environment and RELAP5 thermo-hydraulics code. In this manner, a simple implementation of a virtual operator was coupled to a virtual plant model. To mature this framework, HUNTER 2 has been initiated. HUNTER 2 seeks to scale the earlier proof-of-concept demonstration into a software toolkit that can be deployed to support industry needs for dynamic HRA.

1. INTRODUCTION

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 [1]. Within the U.S. Department of Energy's Light Water Reactor Sustainability Program, the Risk-Informed Systems Analysis (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 [2]. HUNTER is a framework to support the dynamic modeling of human error in conjunction with other (primarily hardware system) modeling tools. HUNTER creates a virtual operator or, potentially, a human digital twin [3] as a human operations counterpart to plant hardware modeling and simulation. The name HUNTER is meant as a counterpart to the various animal-named modeling tools developed at Idaho National Laboratory (INL), such as Risk Analysis Virtual Code ENvironment (RAVEN) [4] and Multiphysics Object-Oriented Simulation Environment (MOOSE) [5]. These tool names playfully combine to become tools like RAVEN-HUNTER or MOOSE-HUNTER.*

The theoretical underpinnings of HUNTER were developed previously [2]. We refer to these earliest efforts as HUNTER 1. However, as part of constructing the first principles for HUNTER, little effort was devoted to making HUNTER a software tool that could be integrated into industry efforts. It remained a research framework for dynamic HRA efforts, but these efforts did not combine into a single solution. As the HUNTER project continues, the main goals are now twofold:

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

This paper addresses the first goal, namely the development of the HUNTER software, here named HUNTER 2 to disambiguate it from the earlier efforts. The disparate elements of the HUNTER 1 framework have now been formalized and implemented as an executable Python library. The initial

* 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.

version is a fully functional proof of concept, including a graphical user interface (GUI) to simplify the process of model building and editing. These refinements go hand in hand with the planned development of additional use cases and accident scenario models. As additional scenarios are modeled, new software features and modeling functions will be included to facilitate a greater utility of HUNTER for a wide range of applications.

2. BENEFITS OF HUNTER AS A DYNAMIC H.R.A. 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, open-ended event progression. The shifting event progression—the defining characteristic of dynamic HRAs and PRAs—allows 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 in order to explore what outcomes may emerge because of different human responses. This capability is especially useful for emerging areas of interest in risk modeling, such as severe accidents, HRA for human interactions with advanced technologies like digital and automated human-system interfaces, balance-of-plant activities beyond the main control room, and specialized areas like flexible equipment use and physical security modeling. As work on developing sample analyses in HUNTER continues, it is important to demonstrate the additional risk insights afforded by dynamic modeling that would not be possible with conventional static methods. An easy-to-use software tool that can help bring new risk insights is essential for industry as it supports new risk requirements.

An additional benefit of dynamic HRA is that the tool can be used beyond simply producing a quantitative output of the human error probability (HEP). Dynamic HRA can provide qualitative insights into the types of activities plant personnel will perform in novel contexts. For example, dynamic HRA 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.

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.

3. PREVIOUS HUNTER EFFORTS

3.1 HUNTER 1 Framework

The HUNTER framework is an approach to dynamically model human cognition and actions as well as incorporate these respective elements into a PRA framework [2]. Many researchers [6-8] have emphasized the importance of simulation and human performance modeling in HRA. The HUNTER framework was developed to overcome some challenges with existing static HRA and to more realistically and accurately evaluate human-induced risks in nuclear power plants (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 1 framework. 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 [4]. 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 [2]. Second, the existing HUNTER framework has considered three major concepts—cognitive models, PSFs, and data sources—for analyzing dynamic operator actions. In the HUNTER 1 efforts, the Goals, Operators, Methods, and Selection rules (GOMS) – HRA [9], the Standardized Plant Analysis Risk-HRA (SPAR-H) autocalculation [10], and dynamic dependency [11] approaches were developed to implement the concepts within the HUNTER framework. These are described in the next subsections.

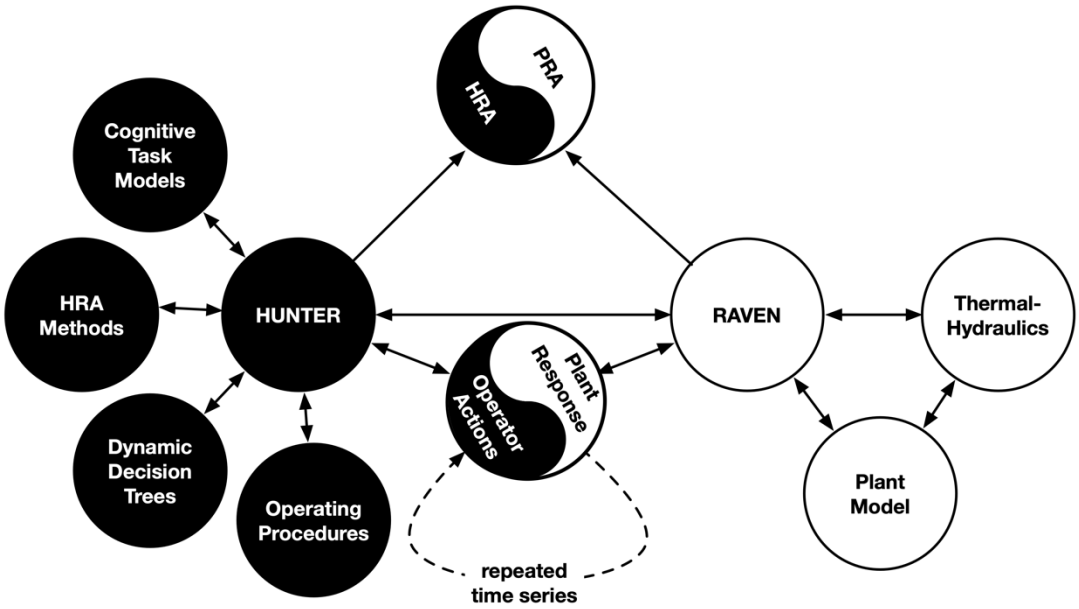

Figure 1: The Original HUNTER 1 Framework

3.2. GOMS-HRA

GOMS-HRA [9] was developed to provide cognition-based time and HEP information for dynamic HRA calculation in the HUNTER framework. It is theoretically derived from the GOMS method, which has been used to model proceduralized activities and evaluate user interactions with human-computer interfaces in human factors research [12]. As a predictive method, GOMS-HRA is well-equipped to simulate human actions under specific circumstances in a scenario. The basic approach of GOMS-HRA consists of three steps: (1) breaking human actions into a series of task-level primitives, (2) allocating time and error values to each task-level primitive, then (3) predicting human actions or task durations.

In GOMS-HRA, human actions are broken into task-level primitives, consisting of the most elemental types of human activities. GOMS-HRA uses six types of task-level primitives defined in the Systematic Human Error Reduction and Prediction Approach (SHERPA) [13]. 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 [14], with an added element for time spent in waiting (W). This figure shows how tasks are aligned to stages of information processing, beginning with sensation and perception, progressing to cognition, and culminating in behavioral actions. Note that items like Instructions (I_R and I_P) can be either verbal (e.g., communication between shift supervisor and reactor operator) or written (e.g., use of printed operating procedures).

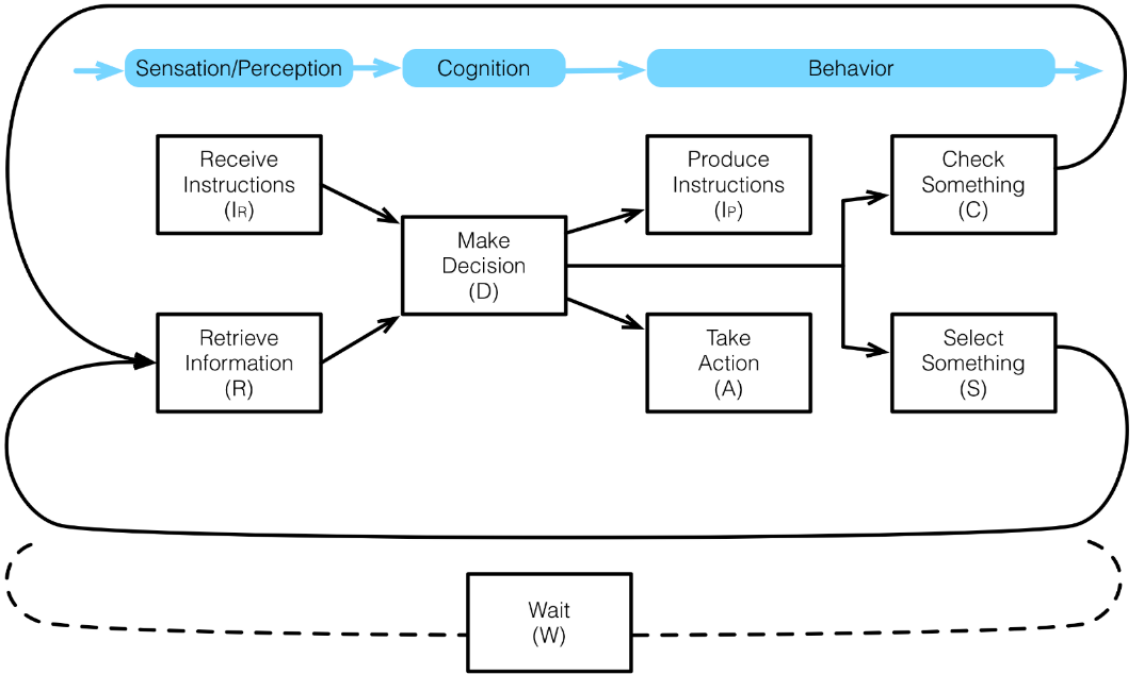

Figure 2: GOMS-HRA Cognitive Model

The GOMS-HRA primitives are treated at different levels of analysis, from Procedure-Level Primitives that readily map to operating procedures to Task-Level Primitives, which represent generic task types. The Task-Level Primitives include nominal error rates and nominal time-on-task estimates. The time information includes the statistical distribution, mean, standard deviation, 5th and 95th percentile, which have been derived from the time data collected through experiments using actual operators in the Human Systems Simulation Laboratory (HSSL) at INL [15].

3.3 SPAR-H Autocalculation

The earlier HUNTER work investigated how to adapt the existing static SPAR-H to a dynamic framework. The SPAR-H Method [16] is an easy-to-use HRA method developed by INL and published by the U.S. Nuclear Regulatory Commission. The approach focuses on the quantification of HEPs on the basis of PSF multipliers. SPAR-H has been widely used by both industry and regulators in its intended area of supporting PRAs for NPPs, but it is also finding use in other industries, such as oil and gas [17]. In traditional static HRA approaches like SPAR-H, human actions are manually reviewed by

human reliability analysts using tools like the Electric Power Research Institute's HRA Calculator. 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. The details on the SPAR-H autocalculation approach are well described in [18]. The basic form of the equation for the Complexity PSF is found below:

$$\textit{Normalized Complexity} = 1.26754 \times \textit{LOOP} + 1.26743 \times \textit{LODG} + 1.26753 \times \textit{LOB} - 0.00025 \times \textit{temperature} - 0.00507 \times \textit{power} + 1.65116 \quad (1)$$

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. The autocalculated PSF is multiplied by the nominal HEP for the Task-Level Primitive provided by a lookup table in GOMS-HRA.

3.4 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. Normally, dependency increases the overall HEP, representing the notion that error begets error. Thus, dependency plays an important role in reasonably accounting for human actions in the context of PRAs and prevents PRA results from being estimated too optimistically based on the HRA results. Dependency analysis has been known to significantly affect the overall result of PRAs. If the results of dependency analyses are inaccurate, they could prove unconvincing for explaining human errors 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 [19]. Accordingly, human reliability analysts have mostly performed dependency analysis by relying on static PRA and HRA information. Dynamic HRA, on the other hand, considers human actions dynamically and models types of activities and events, even where the human role is not clearly understood or predicted (i.e., unexampled events such as severe accidents). Furthermore, a dynamic simulation represents a sequence of operator actions, which make it easier to identify dependency candidates with contextual impacts. The authors previously considered how to treat dependency in dynamic HRA. Boring [11] 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. [19] validated the effects on the basis of experimental data and applied the concept to the dynamic dependency analysis.

4. EXPANDED HUNTER FRAMEWORK

As noted, the original HUNTER 1 framework was a collection of dynamic HRA tools that were not contained in a single software application. Intrinsic to the earliest conceptualizations of HUNTER was the idea that some aspects of the modeling could be exchanged for different modules. For example, while the initial framework focused on making the SPAR-H PSFs dynamic [18], there was an

acknowledgment that more comprehensive or nuanced PSF treatments should also be possible in HUNTER. In other words, while the initial proof-of-concept demonstration may focus on simplified parts of dynamic HRA, this simplification should not prove the limiting factor of HUNTER, and there should be opportunities to support more comprehensive modeling.

The 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 2 incorporates a flexible, modular, and scalable software architecture. This trio of concepts refers 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 human digital twin 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). 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. Similarly, if a detailed analysis is not warranted such as during screening, some comprehensive HUNTER features may be disabled to facilitate expedient analysis.

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 through practice, while other modeling approaches are more theoretical (i.e., are earlier in development with less well-understood phenomena). 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

Table 1: 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.

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 model, which is a direct reflection of the urgency of the analysis. For example, rapid-response modeling required after an incident may have a much shorter development timeline than a more routine version update of existing models over a multiyear timeline. This urgency may drive the need for a simpler model. On the other end of the equation, a simulation that is part of an extensive, multi-scenario analysis, such as in support of a whole plant PRA, may focus on the execution time of the model. HFEs in the PRA that are deemed of low risk significance may not warrant the luxury of waiting for a richly modeled scenario to complete. Instead, simple and quick modeling may suffice for such purposes.
- *Purpose*—specifies how the analysis will be used. While the purpose shapes some of the other modeling considerations, it is most useful as a concept to define the output of the analysis. For example, conventionally, HRA is used to calculate HEPs. Individual HEPs may then be substituted into an overall risk model to see the effect of human performance on the outcome of an event sequence. As noted in [20], there remain some challenges with aligning dynamically calculated HEPs to those produced by static HRA methods. This stems from the unit of analysis, whereby most static HRA looks at the whole sequence of actions wrapped as an HFE, while dynamic HRA typically considers actions at the step or task level. The aggregation from step to HFE is not clearly understood, and calculated HEPs at the step or task must undergo some further conversion to achieve comparability with HEPs for HFEs. Further, there remain other outputs that may prove just as informative to HRA and have not been the purview of conventional HRA methods. For example, dynamic HRA can calculate time estimates for particular tasks. Often, the criterion for success or failure is not the overt commission of an error but the timing-out of an activity expected to be completed in a specific time window. With the exception of some early time reliability curves, 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 can 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 1.

5. HUNTER SOFTWARE FRAMEWORK

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 [21], who suggested that most human performance could be accounted for simply by looking at three factors: individual, organization, and environment.

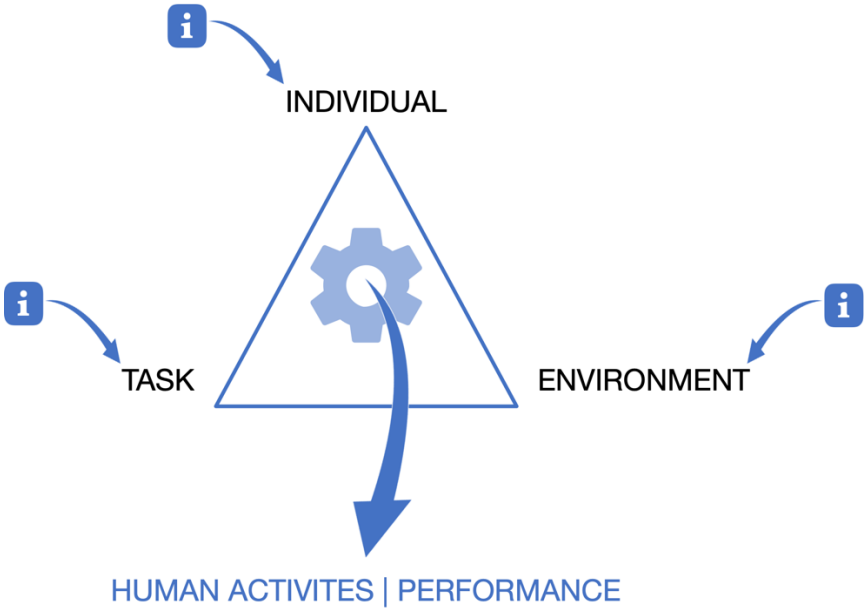

Figure 3: Conceptual Modules (in Black) and Classes (in Blue) of HUNTER 2

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 [22]. 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 3 serve as the software pillars for the new implementation of HUNTER, whereby each pillar serves as a module in the architecture. Joining the modules in the figure are classes, which are depicted in blue. For the present purposes, modules describe the basic elements of human behavior, while classes are the functions that enable the modules to work. Put another way, modules represent *who* (individual), *what* (task), and *where* (environment). Classes represent *how, why,* and *when* activities occur within the modules. Modules are the figurative nouns and verbs of HUNTER, while classes are the adjectives and adverbs.

6. HUNTER MODULES

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

- *Individual Module*—this is the representation of the human performing the activity, sometimes referred to as a "virtual operator." It 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—the hardware digital twin for the human digital twin—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. 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., [18]).

7. HUNTER CLASSES

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

- *Input Class*—the context is set by the scenario at hand. This is shown in Figure 3 as an input (i.e., ) into each of the modules, representing the influences that feed into the scenario. A preprocessor sets the context—the initial configuration for the individual, the task at hand, and the state of the plant—in which HUNTER operates.
- *Scheduler Class*—the glue that holds the other modules together. In the figure, this is signified by the lines of the triangle. It coordinates the interactions between different modules and also paces the progression of the event. Modules may complete their calculations at different rates, and the scheduler synchronizes the inputs, outputs, and interdependencies to a common time scale.

- *Processor Class*—the processing that occurs at each step of the task, which is depicted by a gear in the center of the Figure 3. A step occurs when all modules have completed their modeling refresh cycle and exchanged information. For example, the environment has advanced a time step, updating plant parameters, which have been perceived by the virtual operator (individual), who has responded by activating a virtual switch (task). This task may be driven by a procedure, which must meet certain requirements to advance. The processor class determines the point of advancement to the next task. The processor may include logical assertions, such as actions predicated on conditions met, branching points, and operator decisions.
- *Output Class*—the results of each incremental step in the model. Outputs are changes in the state of the model, which are logged as activities, parameter states, and human performance logs. The output class records the actual outputs, such as the calculated HEPs that allow HUNTER to be used as an HRA method.

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

8. CONCLUSIONS

This software 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 4 of this paper 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.

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, key features like the autocalculation of PSFs, decision-making, and HEP aggregation are not yet fully 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..

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 2 shows 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 such as a lack of software documentation, integration with existing PRA tools, and library of sample analyses will continue to be completed to create a vetted and usable tool that supports industry needs. In parallel, additional features will be deployed.

Acknowledgements

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

References

[1] 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.

[2] 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.

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

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] Boring, R.L. (2015). A dynamic approach to modeling dependence between human failure events. Proceedings of the 2015 European Safety and Reliability (ESREL) Conference, pp. 2845-2851.
[12] Card, S. K., Moran, T. P., & Newell, A. (2018). The Psychology of Human-Computer Interaction: CRC Press.
[13] 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.
[14] 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.
[15] Ulrich, T., Boring, R., L., Ewing, S., & Rasmussen, M. (2017). Operator timing of task level primitives for use in computation-based human reliability analysis. Advances in Intelligent Systems and Computing, 589, 41-49.
[16] 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.
[17] Boring, R.L. (2015). Defining human failure events for petroleum applications of human reliability analysis. *Procedia Manufacturing, 3*, 1335-1342.
[18] 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.
[19] 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.
[20] 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.
[21] Galyean, W. (2006). Orthogonal PSF taxonomy for human reliability analysis. Proceedings of the 8th International Conference on Probabilistic Safety Assessment and Management.
[22] Newell, K.M, van Emmerik, R.E.A., & McDonald, P.V. (1989). Biomechanical constraints and action theory. Human Movement Science, 8, 403-409.