Integrating Classical PRA Models Into Dynamic PRA

D. Mandelli[a*], C. Smith[a], and A. Alfonsi[a]
[a] Idaho National Laboratory (INL), 2525 N. Fremont Ave, 83415 Idaho Falls (ID), USA

Abstract: In this paper we present a series of methods designed to incorporate classical Probabilistic Risk Assessment (PRA) models such as Event-Trees (ETs) and Fault-Trees (FTs) into Dynamic PRA. In contrast to classical PRA, Dynamic PRA couples stochastic methods with safety analysis codes to determine the risk associated to complex systems such as nuclear plants. Compared to classical PRA methods, they can evaluate with higher resolution the safety impacts of timing and sequencing of events on the accident progression. As part of a Dynamic PRA analysis, it is not uncommon that some components of the system to be analyzed might not require a simulation model (which would be computationally expensive) due to its intrinsic characteristics but such components could be actually modeled by a classical PRA model (either ET or FT). In this paper, we investigate how we can integrate classical PRA models (either ET or FT) into a dynamic PRA: an "hybrid" PRA. We show how this integration has been performed within the the RAVEN statistical framework.

Keywords: Dynamic PRA, Event-Tree, Fault-Tree, Markov Model, Reliability Block Diagram

1. INTRODUCTION

Dynamic Probabilistic Risk Assessment (PRA) [1, 2] methods couple stochastic methods [3, 4, 5, 6, 7] (i.e., sampling methods) with system simulators (e.g., RELAP5-3D [8] and MELCOR [9]) to determine the risk associated to complex systems such as nuclear power plants. Compared to Classical PRA methods [10], they can evaluate with higher resolution the safety impacts of timing and sequencing of events on the accident progression without the need to introduce conservative modeling assumptions and success criteria [11].

As part of a Dynamic PRA analysis, it is not uncommon that some components of the system under consideration might not require a complex and computationally expensive simulation model due to its intrinsic characteristics (e.g., no time or physics dependency). From a modeling point of view, such components could be actually included in the analysis by employing simpler Classical PRA models such as Event Trees (ETs) or Fault Trees (FTs) [12].

This paper presents a set of methods to link Classical PRA models into a Dynamic PRA: a "hybrid" PRA. We consider not only ETs and FTs, but also Markov Models [13] and Reliability Block Diagrams (RBDs) [12] as possible modeling solutions (see Section 2). The linking (see Sections 3 and 4) has been performed within the RAVEN statistical framework [14] by creating a common data communication flow among: the sampled parameters, the linked Classical PRA models and the safety analysis code.

2. CLASSICAL PRA

PRA methods [12] have been employed in the nuclear industry after the publication of the NRC document NUREG-1150 [10]. Since then, each U.S. nuclear power plant has developed PRA models for each unit based on ETs, FTs, Markov models and RBDs.

* Corresponding author, diego.mandelli@inl.gov

ETs [12] inductively model the accident progression using a tree structure with the goal of depicting all possible accident sequences. Starting from an initiating event, the accident progression evolves and branches when a branching condition is encountered (i.e., the requested activation of a safety system). This branching condition creates two possible scenarios: the successful (upper branch) or the failed (lower branch) activation of the system. From there, the accident sequence progresses until a new branching condition is encountered.

A FT [12] instead deductively models the status of a system given the Boolean logic status of its components. Formally, system components are described by a set of Basic Events (which are boolean in nature, i.e., *True* or *False*) and the system status (i.e., the Top Event of the FT) is uniquely determined from the Basic Events through a series of logic gates (e.g., AND, OR).

A RBD [12, 15] models the dependencies among system components by employing a directed graph formalism. System components are represented as blocks and component dependencies are represented as directed links. System status is calculated by determining all possible communication flows among the set of input nodes and the set of output nodes. If there is communication flow then the system is failed. Each block is typically characterized by a simple reliability function that is used to compute the system reliability based on the structure of the graph.

Markov models (or Markov chains) [12, 13] are often employed to determine reliability/availability of systems characterized by multiple states, i.e., not only failed or operating states. These models consist of N mutually exclusive states which describe a specific status of the system (e.g., system operating, system under repair, system failed). Transitions among states are stochastic in nature and are described by a set of probability transition rates $M_{p,q}$ $(p,q = 1,\ldots,N)$. Mathematically, a Markov model can be described as [12] $\frac{dP(t)}{dt} = MP(t)$ where each element $P_i(t)$ of the vector $P(t) = [P_1(t),\ldots,P_N(t)]$ is the probability of being in state i at time t while $M = [M_{p,q}]$ contains all possible transition rates from state p to state q $(p,q = 1,\ldots,N)$. It is not uncommon that in PRA applications a sub set of state transitions $M_{p,q}$ are not stochastic (e.g., they have a fixed transition time) or they are time dependent (inhomogeneous Markov models: $M = M(t)$). In these cases, $\frac{dP(t)}{dt}$ cannot be solved neither analytically nor numerically using linear algebra solvers. Instead, these models can be solved numerically using a Monte-Carlo algorithm [13]: starting from an initial state, the algorithm determines a set of N_{HS} transition histories and it determines each element $P_i(t)$ as the ratio of the number of histories that ended in state i at time t over N_{HS}.

3. DYNAMIC PRA

In contrast to Classical PRA, Dynamic PRA approaches [16, 17] explicitly employ system simulation tools instead of complex logic structures (such as ETs and FTs). In fact, they can model in a single analysis: the thermal-hydraulic behavior of the plant [18, 19], external events [20] and operators responses to the accident scenario [21]. The probabilistic part of the analysis is performed by defining a set of stochastic parameters which dictates the time dependent accident progression [1, 2].

From a PRA perspective, a simulation run can be represented as a trajectory in the phase space:

$$\begin{cases} \dfrac{\partial \theta}{\partial t} = \Xi(\theta, c, s, t) \\ \dfrac{\partial c}{\partial t} = \Gamma(\theta, c, s, t) \end{cases} \tag{1}$$

where:

- $c = c(t)$ represents the status of components and systems of the simulator (e.g., status of emergency core cooling system, AC system)
- $\theta = \theta(t)$ represents the temporal evolution of physics-based variables. Each element of θ can be, for example, the temperature or pressure in a specific node of the simulator nodalization
- Ξ is the actual simulator code that describes how θ evolves in time
- Γ is the operator which describes how c evolves in time, i.e., the status of components and systems (system control logic)
- s is the set of stochastic parameters.

Starting from the system located in an initial state, $\theta(t = 0)$ and the set of stochastic parameters s (which are generally generated through a stochastic sampling process), the simulator Ξ determines the temporal evolution of $\theta(t)$. At the same time, the system control logic Γ determines the status of the system components $c(t)$.

Given this, a Dynamic PRA analysis is performed by:

1. Associating a probabilistic distribution function (pdf) to the set of parameters s (e.g., timing of events)
2. Performing stochastic sampling of the pdfs defined in Step 1
3. Performing a simulation run given s sampled in Step 2, i.e., solve the system of Equation 1
4. Repeating Steps 2 and 3 a large number of times and evaluating user defined stochastic parameters such as core damage probability.

This work focuses on extending Dynamic PRA capability to deal with multiple heterogeneous models: the objective is to show how the deterministic modeling of Dynamic PRA can be performed by considering multiple models, not only system simulators but also Classical PRA models. This paper is designed for a Dynamic PRA framework such as RAVEN [7] that allows multiple model to be linked together.

4. INTEGRATION

The idea behind Classical PRA integration into a Dynamic PRA is to replace an expensive physics or a control logic model with a simpler Classical PRA model such as a FT or an ET (see Section 2).

As an example, the control logic of the core safety injection system of a PWR could be modeled by an already available Classical PRA model (e.g., a FT) instead of coding a complex control logic in the RELAP5-3D [8] model. Analogously, the severe accident MELCOR [9] model could be substituted by a Classical PRA model (e.g., an ET).

The basic concept behind Dynamic PRA is that each of its models is characterized by a set of input and output variables and a set of equations that describe the relations between them. Thus, in order to create the integration of Classical PRA models into a Dynamic PRA, it is required to treat them as models described by a set of input variables, output variables and a set of constituent equations that map the two set of variables (see Table 1).

Model	Input Variables	Output Variables
ET	Branching conditions	Sequence, Outcome
FT	Basic Events	Top Event
Markov model	Initial state, End time	Final state
RBD	Block statuses	System status

Table 1: Input and Output variables for Classical PRA models.

The challenge resides in the fact that, in a Dynamic PRA, the timing of events is explicitly considered while Classical PRA models are defined only over Boolean logic variables (i.e., *True* or *False*). Thus, this integration requires that these models must be able to handle in input not only Boolean logic variables but also the time of occurrence of specific events. In Sections 4.1 through 4.4, the integration of Classical PRA models is described in detail for the four models (ETs, FTs, Markov models and RBDs).

Note that the following notation is employed: a Boolean logic variable set to *False* implies that the event related to that variable has not occurred while when the variable is set to *True*, the event related to that variable has occurred. In addition, an input variable with a time value implies that the event related to that variable occurred at that time.

4.1. FT Integration

As described in Section 2, a FT is a model that relates the Boolean logic status of the Top-Event (TE) to the set of S Basic Events $be_1, \ldots, be_S$: $TE = FT(be_1, \ldots, be_S)$ Thus, from a Dynamic PRA perspective, a FT is a model which accepts in input a Boolean logic value (*True* or *False*) for each Basic Event $be_1, \ldots, be_S$ and it generates a Boolean logic value for the output variable TE.

Now, the goal is to extend FT capabilities to accept in input not only Boolean logic values but also the time of occurrence of sub-set of Basic Events. An example of FT model evaluation for different combinations of input variables is shown in Tables 2 and 3 for the AND and OR gates respectively.

For the AND gate, if all Basic Events have logical values, the output of such gate is *True* only if all of them are *True*. Recalling that a time value for a Basic Event defines the occurrence time for such event (i.e., the transition from *False* to *True*), the AND gate (see Table 2) returns a time value if both Basic Events are time dependent ($be_1 = t_1$ and $be_2 = t_2$). Such value corresponds to the first time instant when both Basic Events are *True*: $max(t_1, t_2)$ (see middle table of Table 2). If the two Basic Events have mixed (time and logical) values, similar reasoning applies: if at least one Basic Event is *False* then the gate return a logical value: *False*. If $be_1 = True$ then the gate returns the first time instant when both Basic Events are *True*: t_2.

Similarly reasoning can be applied to the OR gate (see Table 3); if all Basic Events have logical values, the output of this gate is *True* if at least one of them is *True*. If both Basic Events are time dependent ($be_1 = t_1$ and $be_2 = t_2$), the OR gate returns a time value corresponding to the first time instant when at least one Basic Event is *True*: $min(t_1, t_2)$. If the two Basic Events have mixed (time and logical) values, similar reasoning applies: if at least one Basic Event is *True* then the gate return a logical value: *True*. If $be_1 = False$ then the gate returns the first time instant when one Basic Event is *True*: t_2.

In order to extend the FT model capabilities to accept both logical and time values, the evaluation of the Top Event has been performed as described in Algorithm 1. The main method, FTEVALUATION, uses

be_1	be_2	out
False	False	False
False	True	False
True	False	False
True	True	True

be_1	be_2	out
t_1	t_2	max(t_1,t_2)

be_1	be_2	out
False	t_2	False
True	t_2	t_2

Table 2: AND gate responses for different types of input basic events: Boolean logic (left), time valued (center) and mixed (right).

be_1	be_2	out
False	False	False
False	True	True
True	False	True
True	True	True

be_1	be_2	out
t_1	t_2	min(t_1,t_2)

be_1	be_2	out
False	t_2	t_2
True	t_2	True

Table 3: OR gate responses for different types of input basic events: Boolean logic (left), time values (center) and mixed (right).

two sub methods: EVALUATETIMEFT (which is described in Algorithm 1) and EVALUATELOGICFT. The latter one is not explicitly described here since it simply evaluates the Top Event of the FT given the logical status of the set of Basic Events. The method FTEVALUATION is structured as follows:

- If the values of all Basic Events are Boolean logic values, then evaluate the FT using EVALUATELOGICFT (return a Boolean logic value)
- If the values of all Basic Events contain at least a time value, then perform the following:
  1. Set all time values to False and evaluate the FT; if the Top Event is $True$ then return $True$
  2. Arrange time values in ascending order and set t_{th} equal to the first time $t_{th} = t_1$
  3. Set all time values to $True$ if their value is less than t_1 otherwise set them to $False$
  4. Evaluate the FT using EVALUATELOGICFT; if the Top Event is $True$ then return t_1, else set $t_{th} = t_2$
  5. Repeat Steps 3 and 4 over all time values.

Finally, note that the output value of the method FTEVALUATION can be either a logical or a time value.

4.2. ET Integration

As indicated in Section 2, an ET model relates the sequence number *seq* and the predicted outcome *out* of such sequence given the set of input variables, i.e. the set of R branching conditions $(BC_1, \ldots, BC_R)$: $(out, seq) = ET(BC_1, \ldots, BC_R)$

Similar to the FT case (see Section 4.1), the challenge on the integration of ETs in a Dynamic PRA perspective is that branching conditions $(BC_1, \ldots, BC_R)$ can have either a Boolean logic or time values. However, in contrast to deductive nature of FTs, ETs emulate accident progression through an inductive process. This implies that, provided a set of time-valued branching conditions, the inductive nature of

Algorithm 1 FT Evaluation Algorithm.

```
1: procedure FTEVALUATION
2: Input: S Basic Event be_s (s = 1,...,S)
3: Output: Top Event TE
4:
5: retrieve_time_input_values:
6:     Set time to empty array
7:     for s = 1 to S do
8:         if be_s != (True,False) then
9:             Add be_s to time
10:    Add 0. to time
11:    re-order time in ascending order
12:
13: evaluateFT:
14:    if size(time) = 1 then
15:        TE ← EVALUATELOGICFT(be_1,...,be_S)
16:        return TE
17:    else
18:        TE ← EVALUATETIMEFT(be_1,...,be_S,time)
19:        return TE
20:
21: procedure EVALUATETIMEFT
22: Input: S Basic Event be_s (s = 1,...,S)
23: Input: array of time values time
24: Output: Top Event TE
25:
26:    for all t in time do
27:        if t = 0 then
28:            Set all temporal Basic Events to False
29:            TE ← EVALUATELOGICFT(be_1,...,be_S)
30:            if TE = True then
31:                return TE
32:                break
33:        else
34:            Set temporal Basic Events be_s to False if be_s > t
35:            Set temporal Basic Events be_s to True if be_s < t
36:            out ← EVALUATELOGICFT(be_1,...,be_S)
37:            if out = True then
38:                TE ← t
39:                return TE
40:                break
```

the ET would require the generation of a temporal profile of the the variable *seq* and *out* which would not be meaningful or possible[1].

In order to overcome this issue, the evaluation of the ET provided a set of branching conditions BC_r ($r = 1, \ldots, R$) with either logical or time values can be performed by:

- Setting all time-valued branching conditions to $True$ (a time value implies the event has occurred)
- Determining *seq* and *out* for the corresponding combination of branching conditions

Algorithm 2 describes in more detail how the ET model is solved. The main method, $ETevaluation$, is employing a sub method EVALUATELOGICET which determines *seq* and *out* provided the logical values of the branching conditions.

Algorithm 2 ET Evaluation Algorithm.

```
1:  procedure ETEVALUATION
2:  Input: R Branching Conditions BC_r (r = 1,...,R)
3:  Output: ET Sequence seq
4:  Output: ET outcome out
5:
6:      for r = 1 to R do
7:          if BC_r != (True,False) then
8:              BC_r ← True
9:      (seq,out) ← EVALUATELOGICET(BC_1,...,BC_R)
10:     return (seq,out)
```

4.3. RBD Integration

The most effective way to include RBDs in a Dynamic PRA framework is to model them as graph models: the system is composed of a set of of nodes (i.e., corresponding to the blocks in a RBD) that are connected to each other using directional edges. System failure is determined if, given the status of each node, an information flow can be established between the input and the output nodes.

Note that a mapping between RBD and FT is always valid, i.e., it is possible to uniquely model a RBD by employing a FT. Given this, it is possible to extend the RBD capabilities to accept timing values (similarly to the FT case shown in Section 4.1) as indicated in Algorithm 3. The main method, RBDEVALUATION, uses two sub methods: EVALUATETIMERBD (which is described in Algorithm 3) and EVALUATELOGICRBD. The latter one is not explicitly described here since it simply solve the RBD given the logical status of its nodes.

4.4. Markov Models Integration

Integration of Markov models is straightforward since a Markov model (see Section 2) can be considered a model which accepts in input: the initial state S_{in} (i.e., the state at time $t = 0$), the time T_{end} at which we want determine the end state S_{end} and the transition matrix M. The output of this class of models is the state S_{end} at time T_{end}. The evaluation of this model is accomplished by performing transitions among states until T_{end} is reached [13] using a Monte-Carlo approach as described in Algorithm 4. The

[1] For example, a transition of *out* from OK to CD followed by transition from CD to OK could occur.

Algorithm 3 RBD Evaluation Algorithm.

```
1: procedure RBDEVALUATION
2: Input: M Nodes values Node_m with m = 1,...,M
3: Output: Graph Status GS
4:
5: retrieve_time_input_values:
6:     Set time to empty array
7:     for m = 1 to M do
8:         if Node_m != (True,False) then
9:             Add Node_m to time
10:    Add 0. to time
11:    re-order time in ascending order
12:
13: evaluateRBD:
14:    if size(time) = 1 then
15:        GS ← EVALUATELOGICRBD(Node_1,...,Node_M)
16:        return GS
17:    else
18:        re-order time in ascending order
19:        GS ← EVALUATETIMERBD(Node_1,...,Node_M,time)
20:        return GS
21:
22: procedure EVALUATETIMERBD
23: Input: M Nodes values Node_m with m = 1,...,M
24: Input: array of time values time
25: Output: Graph Status GS
26:
27:    for all t in time do
28:        if t = 0 then
29:            Set all temporal Nodes to False
30:            GS ← EVALUATELOGICRBD(Node_1,...,Node_M)
31:            if GS = True then
32:                return GS
33:                break
34:        else
35:            Set temporal Nodes Node_m to False if Node_m > t
36:            Set temporal Nodes Node_m to True if Node_m < t
37:            out ← EVALUATELOGICRBD(Node_1,...,Node_M)
38:            if out = True then
39:                GS ← t
40:                return GS
41:                break
```

sub method SAMPLETRANSITION, provided a generic state S_p ($p = 1, \ldots, N$) and M, randomly picks when the next transition occurs out of S_p and the arrival state S_q. Algorithm 4 continuously performs these transitions until T_{end} is reached.

Algorithm 4 Markov Evaluation Algorithm.

```
1:  procedure MARKOVEVALUATION
2:  Input: Initial state S_in
3:  Input: End time T_end
4:  Input: Transition matrix M
5:  Output: Final state S_end
6:
7:      actual_state ← S_in
8:      actual_time ← 0
9:      while true do
10:         (newState, transitionTime) = SAMPLETRANSITION(actual_state,M)
11:         actual_time ← actual_time + transitionTime
12:         if actual_time > T_end then
13:             S_end ← actual_state
14:             break
15:         else
16:             actual_state ← newState
17:     S_end ← actual_state
18:     return S_end
```

5. CONCLUSIONS

In this paper we have summarized a series of algorithms that can be employed to incorporate Classical PRA models into Dynamic PRA analyses. We have described in detail how it is possible to perform such integration with FTs, ETs, Markov models and RBDs. The objective is to model parts of the system not with advanced simulation tools but instead with Classical PRA models such as FTs or ETs.

Unlike Classical PRA, Dynamic PRA explicitly takes into account of timing of events and, thus, Classical PRA model capabilities needed to be extended in order to deal not only Boolean logic vales but also time values. This is in particular valid for models like FTs and DBDs. We have presented a set of algorithms that, provided an existing Classical PRA model, it creates a Dynamic PRA compatible model that can receive in input both Boolean logic or temporal values. In addition, by employing the EnsembleModel capability of the RAVEN code we were able to link such models to codes such as RELAP5-3D. The advantage of this modeling approach is that the implementation of a complex Dynamic PRA analysis can be strongly simplified since already available Classical PRA models can be integrated and connected to a Dynamic PRA simulation tools, not only codes (e.g., RELAP5-3D) but also ROMs.

REFERENCES

[1] J. Devooght, "Dynamic reliability," *Advances in Nuclear Science and Technology*, vol. 25, pp. 215–278, 1997.

[2] J. Devooght and C. Smidts, "Probabilistic reactor dynamics. the theory of continuous event trees," *Nuclear Science and Engineering*, vol. 111, pp. 229–240, 1992.

[3] B. Rutt, U. Catalyurek, A. Hakobyan, K. Metzroth, T. Aldemir, R. Denning, S. Dunagan, and D. Kunsman, "Distributed dynamic event tree generation for reliability and risk assessment," in *IEEE - Challenges of Large Applications in Distributed Environments*, pp. 61–70, 2006.

[4] E. Hofer, M. Kloos, B. Krzykacz-Hausmann, J. Peschke, and M. Woltereck, "An approximate epistemic uncertainty analysis approach in the presence of epistemic and aleatory uncertainties," *Reliability Engineering and System Safety*, vol. 77, no. 3, pp. 229–238, 2002.

[5] K. S. Hsueh and A. Mosleh, "The development and application of the accident dynamic simulator for dynamic probabilistic risk assessment of nuclear power plants," *Reliability Engineering and System Safety*, vol. 52, pp. 297–314, 1996.

[6] G. Cojazzi, "The DYLAM approach for the dynamic reliability analysis of systems," *Reliability Engineering and System Safety*, vol. 52, no. 3, pp. 279–296, 1996.

[7] A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, R. Kinoshita, and A. Naviglio, "RAVEN and dynamic probabilistic risk assessment: Software overview," in *Proceedings of European Safety and Reliability Conference (ESREL 2014), Wroklaw (Poland)*, 2014.

[8] RELAP5-3D Code Development Team, "RELAP5-3D code manual," tech. rep., Idaho National Laboratory Technical Report, 2005.

[9] R. O. Gauntt, *MELCOR Computer Code Manual, Version 1.8.5, Vol. 2, Rev. 2*. Sandia National Laboratories, NUREG/CR-6119, 2000.

[10] U. S. NRC, "Nurec/cr-1150: Severe accident risks: An assessment for five u.s. nuclear power plants final summary report," tech. rep., U.S. Nuclear Regulatory Commission, Washington DC, 2005.

[11] D. Mandelli, Z. Ma, C. Parisi, A. Alfonsi, and C. Smith, "Measuring risk importance in a dynamic PRA framework," in *Proceeding of Probabilistic Safety Assessment (PSA) Conference*, 2017.

[12] J. C. Lee and N. J. McCormick, *Risk and Safety Analysis of Nuclear Systems*. Wiley-Blackwell, 2011.

[13] T. Aldemir, "Utilization of the Cell-To-Cell Mapping Technique to construct Markov failure models for process control systems," in *Proceedings of Probabilistic Safety Assessment and Management: PSAM1*, pp. 1431–1436, Elsevier, New York, 1991.

[14] C. Rabiti, A. Alfonsi, J. Cogliati, D. Mandelli, and R. Kinoshita, "RAVEN, a new software for dynamic risk analysis," in *Proceedings of the Probabilistic Safety Assessment and Management (PSAM) 12*, 2014.

[15] S. Distefano and A. Puliafito, "Dependability evaluation with dynamic reliability block diagrams and dynamic fault trees," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 1, pp. 4–17, 2009.

[16] C. Acosta and N. Siu, "Dynamic event trees in accident sequence analysis: application to steam generator tube rupture," *Reliability Engineering and System Safety*, vol. 41, pp. 135–154, 1993.

[17] D. Mandelli, C. Smith, C. Rabiti, A. Alfonsi, R. Youngblood, V. Pascucci, B. Wang, D. Maljovec, P.-T. Bremer, T. Aldemir, A. Yilmaz, and D. Zamalieva, "Dynamic PRA: an overview of new

algorithms to generate, analyze and visualize data," in *Proceeding of American Nuclear Society (ANS), Washington (DC)*, 2013.

[18] D. Mandelli, C. Smith, T. Riley, J. Nielsen, A. Alfonsi, J. Cogliati, C. Rabiti, and J. Schroeder, "BWR station blackout: A RISMC analysis using RAVEN and RELAP5-3D," *Nuclear Technology*, vol. 193, no. 1, pp. 161–174, 2016.

[19] D. Maljovec, S. Liu, B. Wang, D. Mandelli, P. T. Bremer, V. Pascucci, and C. Smith, "Analyzing simulation-based PRA data through traditional and topological clustering: A BWR station blackout case study," *Reliability Engineering & System Safety*, vol. 145, no. 1, pp. 262–276, 2015.

[20] D. Mandelli, S. Prescott, C. Smith, A. Alfonsi, C. Rabiti, J. Cogliati, and R. Kinoshita, "Modeling of a flooding induced station blackout for a pressurized water reactor using the RISMC toolkit," in *Proceedings of PSA 2015 International Topical Meeting On Probabilistic Safety Assessment And Analysis*, 2015.

[21] R. L. Boring, R. B. Shirley, J. C. Joe, D. Mandelli, and C. Smith, "Simulation and non-simulation based human reliability analysis approaches," tech. rep., Idaho National Laboratory Technical Report: INL/EXT-14-33903, 2014.