A Unified Approach to PSA Accident Sequence Model Quantification

Donald J. Wakefield and James C. Lin
ABSG Consulting Inc. (ABS Consulting), Irvine, CA, USA

Abstract: Existing, fault tree linking models and large, event tree linking models for nuclear power plants are so large that they challenge computer memory limits and/or require excessive run times to fully quantify at the frequency cut-offs required for convergence. In some software quantification tools, the amount of frequency cut-off is not known, and for others, the sheer size of the models becomes unwieldy. The conceptual approach described here is to make use of Monte Carlo simulation. The simulation is one which treats a series of initiating event challenges to the logic model as constants and each challenge assesses whether the logic model end states are true or not. The logic model may be a single fault tree, a single event tree with branch probabilities, or a combination of fault trees and event trees. The outcomes of each challenge are tallied at the end of the simulation to obtain conditional end state probabilities and then combined with the initiating event frequencies to obtain accident sequence frequencies. Quantification cut-offs are not used for this approach and there are no restrictions on the use of NOT gates. Convergence of the Monte Carlo simulation would be the main issue.

Keywords: PSA, Monte Carlo, Simulation, Sequences, Fault Trees.

1. BACKGROUND

Currently, linked fault tree models and linked event tree models have become so large that the time to quantify a large, nuclear plant probabilistic safety assessment (PSA) model has become excessive and/or challenges the available computer memory. Both of these issues limit the frequency truncation cut-offs that can be practically applied for accident sequence frequency quantification. Currently, the cut-offs that can be practically used are not sufficient to mathematically assure adequate convergence of the model end state frequencies.

The use of NOT logic in fault tree models is generally restricted by the features within the available quantification tools; e.g., a popular fault tree quantification engine FTREX [1] limits the level of NOT gates permitted to just 1. Large fault trees are used in both fault tree linking and event tree linking approaches. The advent of the Binary Decision Diagram approach [2] makes it possible to have many levels of NOT logic for modest size fault trees, but not for large fault tree linking models.

For maximum speed, it is desired to create a fully separable approach to sequence quantification to which multi-processors can be applied keeping each processor fully employed right up to the end of the quantification and without much penalty for assigning tasks to each processor. Assigning processors to separate initiating events is one way to accomplish this, but this approach is limited when only one or a small number of initiators account for most of the processing time.

Further, it is desired that both approaches to sequence modeling, be quantified using the same evaluation tool. The proposed simulation approach could be developed in such a way to accomplish this final objective.

2. OVERVIEW OF THE APPROACH

The proposed approach is to make use of Monte Carlo simulation. A simulation consists of a series of challenges to the logic model, where for each challenge we randomly sample the status of each probabilistic input to the model and assesses whether the logic model end states are true or not for that

challenge. The outcomes of many challenges are tallied at the end of the simulation to obtain the probabilities of each end state of the logic model given the type of challenge posed. Typically one type of a challenge would be posed for a given simulation. These challenge types would correspond to initiating event types such as a small loss of coolant accident, steam generator tube rupture, earthquake of a selected size, or a simple reactor trip. As many simulations would be run as there are initiating events posed. Since the rate of a specific challenge type occurs at a given frequency, this is like asking the outcome of millions of years of plant operation, where we assume that plant performance is represented by our logic models. The simulation envisioned here is not one that simulates many years of time directly, but rather simply many challenges from a constant frequency initiating event. The simulation approach is easily suited for parallel processing since the challenges can be evaluated independently and the results combined at the end of the simulation.

The evaluation of minimal cutsets and multiplication of basic events which make up each cutset (for fault tree linking models), or multiplication of split fraction probabilities to obtain a sequence probability conditional on the initiating event challenge (for large event tree linking models), is not what we mean here by assessing the logic model. Rather, we mean that given the true or false state of each logic model, probabilistic input for a given challenge, to assess whether the logic model top event or end states are true for that set of input states. Whether the logic model consists of a fault tree, an event tree, or some combination of the two, the logic model end state or end states are determined as true or false for each challenge.

The idea is that each logic model evaluation of a single challenge, as described above, should be very fast. The current logic model quantification approaches that use Boolean reduction and full event tree and fault tree walks with frequency truncation would not be necessary. However, in the proposed approach, many such challenges would have to be evaluated to complete a full simulation. The proposed approach does not rely on truncation, but does rely on Monte Carlo simulation and so is subject to convergence issues depending on the nature of the problem; i.e., high conditional failure probability outcomes should converge more quickly and with fewer challenges than those whose end state conditional probabilities are exceptionally rare events.

This approach has already been used in software developed by others for fault tree analysis; e.g., References [3] and [4]. Reference [3] is such a software tool that was developed in 2005. Tests of that code on a modern computer, and without crediting parallel processing, reveal that 300 million samples of an admittedly simple fault tree involving just 54 minimal cutsets can be performed in less than 5 seconds; and that quantification time includes the time to identify contributors and compute basic event importance measures as well as the probability of the fault tree top event probability. The development of Monte Carlo simulation approaches for fault trees in the past has been to handle more complex, often time-dependent gates that are not employed in most complex accident sequence models; e.g., for nuclear power plants.

3. ADVANTAGES OF A SIMULATION APPROACH

Before describing the proposed approach in more detail, we first list the advantages expected from the simulation approach.

1. For fault tree-linking models:
    a. There is no frequency truncation used to perform the simulations.
    b. There are no approximations required for cutset totaling.
    c. NOT logic gates can be accommodated easily and for many levels. This would permit exclusive event logic to be incorporated directly in the linked fault tree and also allow the incorporation of exclusive maintenance combinations and human action dependencies directly into the models.
    d. For fault tree linking models represented by a single top event, not only can the top event probability be reported but also the fractional importance of any gates the user chooses. Therefore, for models linking many individual sequences under a large OR

gate, the contribution of each sequence can also be evaluated, effectively allowing fault trees to compute many end states in a single pass.

e. Sequence groups can be defined in terms of intermediate gate states and importance measures computed for those groups as well.

2. For event tree linking models:
    a. It is possible to eliminate or at least minimize the split fraction development, associated frequency quantifications, and to avoid the need for complex split fraction assignment logic rules.
    b. This approach allows separate top events in the sequences to be explicitly dependent on the same basic events, while still preserving the time sequencing of the event tree tops.
    c. For large, event tree linked model evaluations, the split fraction representation of sequences and the basic event cutset contributors are saved, and importance measures for both split fractions and basic events may be computed directly.
3. Simulations can be developed in such a way to evaluate both fault tree linking and event tree linking models, allowing the best of both approaches to be obtained.
4. Parallel processing can be easily employed since each processor can be assigned to its own simulation batch of challenges and its own random number seed, and the simulation results then directly added at the end.
5. Partial correlation between probabilistic input states can be modeled efficiently by simulation techniques.
6. Additional gate types may be developed for complex accident sequence models that could prove useful; e.g., recovery gates which consider the timing of failures within the assumed mission time.

4. SIMULATION OF PROBABILISTIC INPUT STATES

The fault tree or event tree probabilistic inputs are either basic events, or split fractions, or both. We first need to sample whether these probabilistic inputs are true on a particular logic model challenge. The most brute force approach is to evaluate the status of every basic event or split fraction each challenge using a random number to decide which of the inputs are true for that challenge. This is one of the approaches adopted in Reference [3] for fault tree simulations. However, other approaches can greatly reduce the number of input state evaluations on any given challenge.

A second approach adopted in [3] first orders the probabilistic inputs and then uses their probabilities of being true to compute the first of the ordered inputs to be true for that challenge. This can be generalized to select the second event in the ordered list to be true and so on, minimizing the number of random numbers required to assess the entire list of ordered input states; i.e., those not evaluated to be true are then all false.

Another way to approach the evaluation of probabilistic inputs is to initially evaluate each probabilistic input as if it were to be repeatedly challenged and use a random number to determine which challenge number it is first to be assumed true. By applying this to all inputs, an ordered array by challenge number could list the inputs that are true for each challenge. All other inputs would be false. As the ordered list of challenges are evaluated in turn, new input state evaluations would only be performed for those input states that were true on the current challenge, the rest being known to be true only in later challenges. Note that in all these approaches while the input state probabilities are used in the calculation of which are true for a given challenge, they are not used in the logic model evaluation.

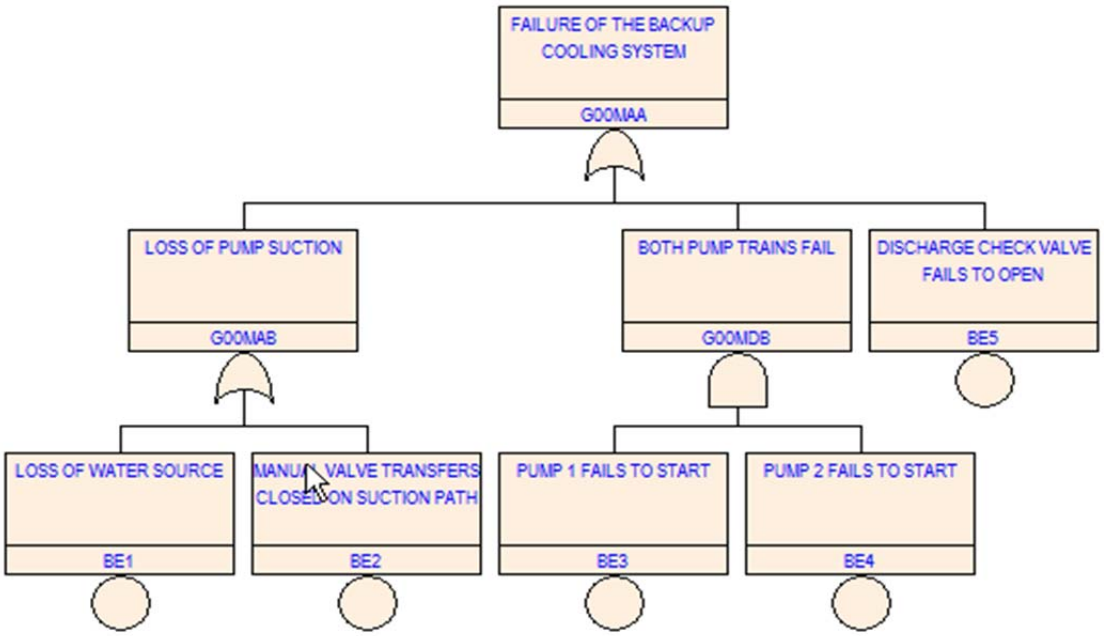
An obvious feature of this approach is that if there are challenges, in which there are no probabilistic states that are true, then the logic model solution for all states being false can be cited with no need to assess the status of the logic model end states again.

Additional heuristics are envisioned that would make this process more efficient and to limit the number of unique evaluations of the logic models.

5. SIMULATION APPLIED TO FAULT TREES

The determination of the status of the probabilistic events (i.e., basic events for fault trees) for each challenge was discussed in the previous section. Here we discuss the evaluation of the fault tree logic when it is known which basic events are true or false. It appears most efficient if the assessments used for one basic event change save intermediate information that could be used for the next basic event change in state. Of course if no basic events fail then there is no need to walk the tree logic because we already know the state of the fault tree assuming all basic events successful. Figure 1 provides an illustration.

Figure 1: Example Evaluation of a Fault Tree



Event #	1	2	3	4	5
P(BEx)	1.00E-05	1.00E-06	1.00E-03	1.00E-03	4.00E-04
RAND NO.	0.64	0.156	0.7	0.0005	0.52
COMPARE	>1E-5	>1E-6	>1E-3	<1E-3	>4E-4
TE STATE	FALSE	FALSE	FALSE	TRUE	FALSE

In this illustration, the brute force approach to determining the basic events' status for one challenge is used. The status of the fault tree with all inputs set to false is also false. Of the five events considered, only one, Basic Event 4, was found to be true for the challenge represented. Basic Event 4 only feeds the AND Gate G00MDB. Since the input from Basic Event 3 to the same gate is false, the AND gate is also false. There is no need to evaluate the higher levels of the fault tree because no further changes in gate states are possible. This is an important point, it will often be the case that the entire fault tree need not be evaluated because of the limited impacts of the basic events that are set to true each challenge; i.e., the higher level gates remain at their initial state. If no events are set to true, then no evaluation at all is required. Some fault tree pre-processing of the OR and AND gates to subsume unnecessary levels and to remove basic events whose values are 1.0 or 0, will also speed the fault tree evaluation process.

For the bottom-up fault tree walk, it would be useful to have a database of which gates each basic event enters since each occurrence of a failed basic event changes that gate's input states. Similarly, a

database of gates each gate enters and which gates and basic events enter each gate could be prepared prior to the start of the simulation.

It would appear straightforward to walk the fault tree logic one basic event state change to a given gate at a time. The cumulative effects of such gate state changes could then be determined as all basic events that change state and all the gates they feed are evaluated. Reference [3] implemented a clever bit-map approach for fault trees with just OR and AND gates. This would have to be generalized to k/n gates and to NOT logic for our purposes.

It is worth mentioning that some dependencies between basic events may best be considered using NOT logic directly in the fault trees. Other basic event dependencies, such as for mutually exclusive system alignments, may instead be accounted for by the manner in which the basic event states are sampled.

Contributors to the top event results can be saved if desired. Such failure combinations may not be minimal. We could determine if they are minimal by removing each basic event one at time from the cutset to see if the top event is still failed. If still failed then the counter for that basic event would not be incremented for that challenge after all. In the simulation approach, the computation of fractional basic event importance does not depend on whether the cutsets are minimal or not. In Reference [3], the cutsets identified by simulation are minimized at the end of the computation.

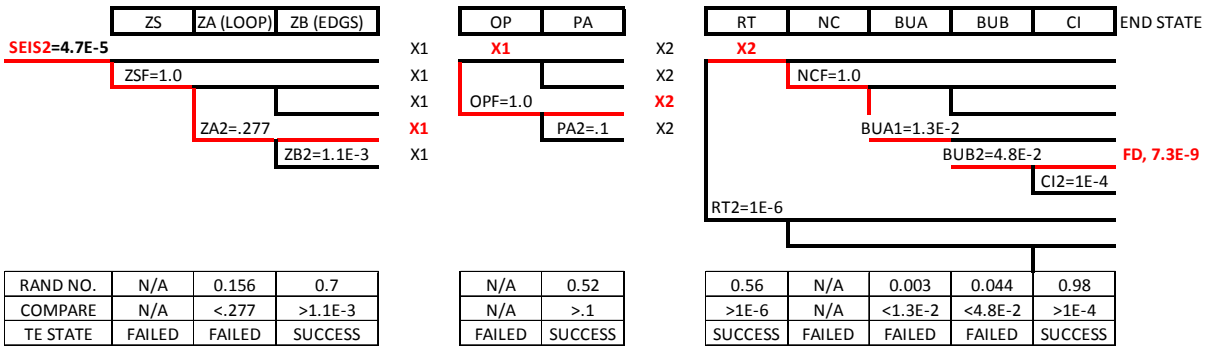
The cutset probabilities can be computed to determine which are to be saved. If the cutset probability is above a saved cutset cut-off specified by the user, then it would be saved for review. Since the highest probability cutsets may appear many times among all the challenges in the simulation, a process to remove duplicates is necessary.

For basic event importance, each basic event failed in a cutset that causes the top event to fail, could be tracked and a counter for each basic event incremented for each challenge that it fails and the top event is true. Unlike for the classical approach to fault tree solutions, where information about the status of gates is lost, for a simulation the approach to importance can be generalized to gates as well.

6. SIMULATION APPLIED TO EVENT TREES

For event tree logic models challenged by constant initiating event frequencies, the approach is similar to that for fault trees, but not exactly the same. The branch point probabilities, or split fractions, become the probabilistic input states and the challenge is to split fractions whose success or failure state is to be determined. However, for evaluation of event trees, just a single path through the event tree is evaluated; i.e., the plant has only one response to the challenging initiating event and its outcome is the end state assigned to that sequence. The logic evaluation would walk the event tree first deciding which split fraction applies at each event tree branch node, and then determine whether to choose the success or failure path from that node based on a random number selected for that top event. For split fractions that are always 1.0 or 0, the end state for that top event is always the same and no random number need be selected. A key difference from the fault tree evaluation approach discussed in Section 4, is that here, only the split fractions which are all pre-calculated and are actually used in the single sequence path, are then evaluated for that challenge; i.e., at most one random number for each of the top events in the linked event tree. Sequence frequency truncation would not be used; i.e., all sequences would be computed end-to-end regardless of their frequency. This simulation approach is then performed separately for each initiating event.

Figure 2: Simulation of Linked Event Trees



RAND NO.	N/A	0.156	0.7
COMPARE	N/A	<.277	>1.1E-3
TE STATE	FAILED	FAILED	SUCCESS

N/A	0.52
N/A	>.1
FAILED	SUCCESS

0.56	N/A	0.003	0.044	0.98
>1E-6	N/A	<1.3E-2	<4.8E-2	>1E-4
SUCCESS	FAILED	FAILED	FAILED	SUCCESS

Figure 2 illustrates the evaluation of linked event trees by this approach. Random numbers are rolled for each split fraction whose values are neither 1.0 nor 0 along the sampled path. These random numbers are compared with the split fraction values to determine the branch outcome; i.e., success or failure. In the single challenge example of Figure 2, three of the top events are guaranteed failed and three others fail probabilistically; i.e., a particularly unlucky day. The red highlighted path shows the sample sequence.

A very good feature of this approach is that it can be implemented on existing large event tree linking models developed in RISKMAN™ [5], without any modeling changes.

Similar to fault tree simulations, the sequence groups and split fraction importance information can be tracked for each of the split fractions along the single evaluated sequence, exactly as is done now within RISKMAN; i.e., as a table of split fraction names and a count of their status for each challenge evaluated. Here the counts stored would be for the total number of sampled sequences involving the success or failure of the split fractions to each sequence group, instead of the frequency of the sequences involving the split fractions to each bin or group.

A key issue is how to save the sequence representations without saving too many. Recall that since many simulations are performed, saving every sequence outcome would be excessive, even if just for a single initiator. In general, for RISKMAN sequences we only need to save the top “n” or so sequences to each end state; i.e., just those that are to be reviewed by the user. There are several things we can do to address this but the key is how to look for simulation outcomes that result in the same sequence, and to remove the second and other occurrences of that same sequence. One approach is to compute the sequence frequency for each sequence simulated (we already know the split fractions and their values at every node of the linked event tree sequence) assigned to a given end state. These can be compared against the sequences saved previously for that end state and only the highest ranked sequences would be retained. For sequences that have exactly the same frequency as those computed previously, these could be compared top event state by top event state to see if they are duplicate sequences, and the duplicates removed if a match of sequences is found. Since only one initiator is computed at a time, the comparison of sequences would not have to examine sequences starting with different initiators.

For multi-state top events, the approach is slightly different. For most problems the sum of the multi-state split fraction probabilities must sum to 1.0. Therefore we could sample a random number every challenge to determine the single path from among the multi-state paths of that top event to pursue. The split fraction values for the multi-state top event would determine the relative split to be sampled.

Note that no frequency truncation is used for this approach. Instead, it is the convergence of each end state frequency that is of interest. For nuclear power plant applications, the end states may be just success or core damage, or may include an extensive list of end states such as are required to define release categories in a Level 2 assessment.

The event tree logic model is evaluated conditional on the occurrence of the constant initiating event. To convert the challenge results to end state frequencies, one just multiplies the conditional end state probabilities evaluated in the simulation by the initiating event frequency to obtain the end state or more generally any sequence group frequencies for that initiating event.

Estimates of the error in the sample mean of each end state or sequence group can be made and are useful to demonstrate that sufficient simulations are performed. For the variation in an end state mean, one can assume the samples are distributed normally and use the estimates for the 95% confidence interval; i.e., between the 2.5% and 97.5%. The 95% confidence interval is then:

$$=2*1.96*SQRT\ [\mu*(1-\mu)/M]$$

where M is the number of trials and μ is the proportion of challenges with the end state true divided by the number of challenges M.

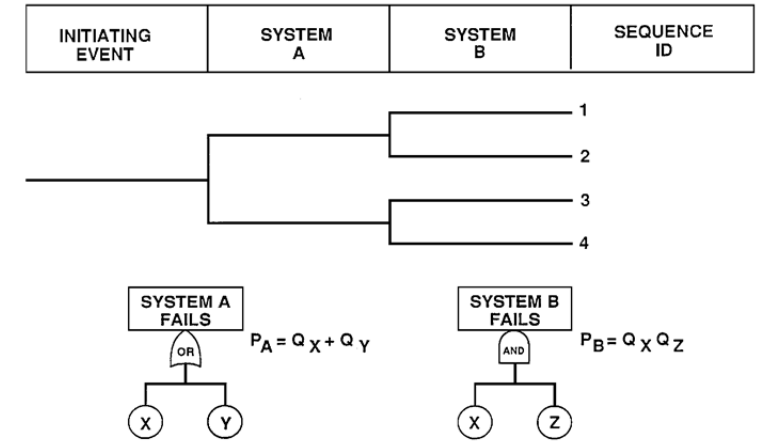
From the above, we see that for smaller values of μ larger values of M are required to achieve the same absolute value of the 95% confidence interval. This implies that the simulation approach works best when the conditional probabilities of the logic model end state are relatively high; so that fewer challenges are required for convergence. However, experience shows that the longest running RISKMAN sequences using classical techniques are precisely those which have high conditional probabilities of the end states.

7. SIMULATION APPLIED TO COMBINED EVENT TREE-FAULT TREE MODELS

Consider the simple example in Figure 3 which has two fault trees to represent the failure event combinations for System A and for System B. The simulation approach for logic models that use both event trees and fault tree models starts out the same as that used for just those with fault trees. The basic event states, used in any of the fault trees, are evaluated for each challenge using any of the approaches identified above in Section 4. The key here is that the basic event states that are determined for a given challenge are applicable to all the fault trees used in the event tree for that challenge.

We postulate an initiating event challenge and evaluate the fault tree logic under the first Top Event A in the event tree sequence. For the single challenge example in Figure 3, Event X is true and Events Y and Z are false on that challenge. From the fault tree for System A, Event X feeds an OR gate which must then be true; i.e., System A fails the first challenge. We are therefore on the down branch of the event tree under System A. System B is governed by an AND gate that requires both X and Z to be true to cause System B to fail. Event X is true but not Z. Therefore, System B does not fail and the three is walked on the upward, success, branch under System B to Sequence ID 3; i.e., the up branch under System B following failure of System A. This single path through the event trees is developed and the associated end state assigned for the first challenge. The simulation repeats this process for many challenges tracking the sequence counts ending in each of the end states and other results of interest. After multiplying by the initiator frequency, we can save the highest frequency sequences and also track the fractional importance of basic events, top event failures, and other gate states or event combinations as desired.

Figure 3: Simulation of Combined Event Tree-Fault Tree Models



Event #	X	Y	Z
P(BE)	1.00E-02	1.00E-03	1.50E-01
RAND NO.	0.0034	0.156	0.7
COMPARE	<1E-2	>1E-3	>.15
BE STATE	TRUE	FALSE	FALSE

For fault-tree linking models, the problem formulation is largely as presented in Figure 3. The fault tree may be the same for each node under a given top event, or may differ for each node. The fault trees for different systems often share basic events and even high level gates are repeated between the system models; for example to model common support systems. The evaluation of the individual fault trees are to be resolved only along the one sequence path taken for that challenge. These separate fault trees may be evaluated completely separately or their shared gates first evaluated and then the states of these shared gates introduced to the higher level fault trees for each top event. It is not necessary to evaluate these shared gates via parallel processing because the multi-processors are more efficiently assigned to their own set of challenges to integrate into the overall simulation.

For large event tree linking models, such as evaluated using RISKMAN [5], the event tree branch probabilities are evaluated conditionally on the status of earlier branches of the event tree. This requires many split fractions for each top event to be defined in terms of the states of earlier top events and each split fraction is then quantified separately ahead of time. The applicable split fraction values, or one minus their values along success branches, are then multiplied along each sequence to derive the individual sequence frequency.

For simulation of large event tree models, the classical approach to sequence frequency quantification would instead be dramatically changed. The fault tree top events could be used as is in the simulation without having to define split fractions, to define conditional split fractions on other split fractions, nor to quantify them ahead of time. Instead the states of preceding top events in the event tree would be associated with house events in the fault trees used for later top events. The states of all house events in a top event fault tree could then be resolved as part of the fault tree evaluation based on the earlier sequence path in the event tree. The states of such preceding top events can just be added to the list of basic event impacts assumed to be true for that challenge. In this way, the relatively small fault trees of the large linked event tree models are evaluated separately but only for the conditions defined by the specific sequence being walked for the given challenge. Since they are typically small fault trees, they would be expected to be resolved relatively quickly.

The large event trees in the event tree linking approach were designed to achieve finer granularity to the events in the accident sequences; i.e., to preserve event time sequencing and at a level of detail near the level of events seen in plant emergency procedures and hence what the operators are used to discussing. However, to accomplish this greater degree of granularity, additional top events have to be added to the sequence models to ensure that any shared equipment between top events appear

earlier as separate top events. The result of a properly constructed event tree linking model is that any basic event appears no more than once in the fault trees for top events describing an entire accident sequence. Further, trains of supporting systems must also be separated into their own top events so as to properly track the dependencies of other systems on these train separated support systems. The point of mentioning this is that with the simulation approach, it will no longer be necessary to add these top events. The status of all basic events in the model for each challenge is known prior to the event tree walk. Basic events can therefore appear in multiple top events without fear of double counting since we are only determining whether a top event is successful or failed and not what its occurrence probability is determined to be. The need for Boolean rules to choose the appropriate split fraction would also be eliminated. We can track the conditional probability of an end state or of individual sequences without having to compute the failure probabilities of individual top events. It is still possible, however, to determine the importance of individual basic events, top event failures, or gate failures to the end states and sequence groups of interest.

We see then that with the simulation approach, the large event tree linking and large fault tree linking methods for sequence quantification are coming closer together. Instead of combining the fault tree linking style models into a single top event, we could retain their relatively smaller event tree sequence models and evaluate the one event tree path per simulation just as for large event tree linking models. Whichever approach is used to build the top event node fault trees (i.e., use house events and evaluate them conditional on preceding top event states, or repeat the supporting logic explicitly within the top event fault tree itself) could be supported using simulation.

In fact, it becomes more a matter of modeling style as to how many top events to include in an event tree set for each initiating event. It is only conjecture at this point as to whether the evaluation of numerous small fault trees along an accident sequence, each depending on relatively few basic events, would be quicker than a smaller number of larger fault trees that are more typical of fault tree linking models.

8. CONCLUSIONS

In this paper, Monte Carlo simulation approaches to sequence frequency quantification have been discussed as may be applied to standard fault tree linking and large event tree linking accident sequence models for nuclear power plants. The flexibility of simulation techniques to more accurately and without frequency truncation evaluate such large logic models, including models that extensively utilize NOT logic; along with the ability to easily separate such problems for parallel processing, offers two powerful incentives to explore such techniques further. Convergence of the Monte Carlo simulation would be the main concern; i.e., can the processing time to evaluate the logic models for each challenge be made acceptable when the conditional probabilities of the end states are quite low?

We see that the proposed approach, lends itself to quantifying both linked fault tree models and linked event tree models. It does so in ways that bring these two sequence logic model constructs closer together, perhaps even allowing an easy model transition from one to the other, depending on the needs of the user.

Further, simulation approaches involving new gate types may possibly be formulated to address the timing of accident failures in ways that permit more affective modeling of recoveries.

Acknowledgements

The author would like to thank the encouragement and comments by the probabilistic risk assessment staff at ABS Consulting, headed by Dr. David H. Johnson, for their contributions to this paper.

References

[1] Electric Power Research Institute, "*Fault Tree Reliability Evaluation eXpert (FTREX 1.6)*", (2011).
[2] A. Rauzy, "*Aralia User Manual*", ARBoost Technologies, (2005).
[3] Formal Software Construction Limited, "*Open FTA User Manual Version 1.0*", (2005).
[4] K. Durga Rao, et al., "*Dynamic fault tree analysis using Monte Carlo simulation in probabilistic safety assessment*", Reliability Engineering and System Safety, Vol. 94, pp. 872–883, (2009).
[5] D.J. Wakefield, et al., "*RISKMAN, Celebrating 20+ Years of Excellence!*", ABS Consulting, presented at PSAM10, (2010), Seattle, Washington.