INL/RPT-22-70424
Revision 0

Light Water Reactor Sustainability Program

Dynamic and Classical PRA Coupling using EMRALD and SAPHIRE

December 2022

U.S. Department of Energy

Office of Nuclear Energy

DISCLAIMER

This information was prepared as an account of work sponsored by an agency of the U.S. Government. Neither the U.S. Government nor any agency thereof, nor any of their employees, makes any warranty, expressed 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 herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof.

INL/RPT-22-70424
Revision 0

Dynamic and Classical PRA Coupling using EMRALD and SAPHIRE

Steven Prescott
Ted Wood
Michael Ziccarelli

December 2022

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

ABSTRACT

Both classical and dynamic probabilistic risk assessment tools are valuable for different kinds of analysis. Typically, one or the other is used depending on the scenario driven by the limitations of the tool. Often the results of one are used as a parameter in the other. This research looks at the possible methods for combining classical and dynamic analyses by coupling EMRALD and SAPHIRE.

This was initial exploratory research to evaluate methods and determine how the tools could be coupled. A short background of SAPHIRE and its solving methods are provided, along with information on EMRALD to understand the correlation between the types of modeling. Finally, an overview of several methods is provided as well as the limitations or application of each.

CONTENTS

FIGURES

ACRONYMS

CDF	Core damage frequency
CPRA	Classical probabilistic risk assessment
DLL	Dynamic Linked Library
DPRA	Dynamic probabilistic risk assessment
EMRALD	Event Modeling Risk Assessment using Linked Diagrams
GUI	Graphical user interface
INL	Idaho National Laboratory
JSON	JavaScript Object Notation
LOSP	Loss of Off Site Power
NRC	Nuclear Regulatory Commission
PRA	Probabilistic risk assessment
RAVEN	Reactor Analysis and Virtual control Environment
SAPHIRE	Systems Analysis Programs for Hands-on Integrated Reliability Evaluations
SPAR	Standardized Plant Analysis Risk

DYNAMIC AND CLASICAL PRA COUPLING USING EMRALD AND SAPHIRE

1. INTRODUCTION

In the realm of probabilistic risk assessment (PRA), there has been a trend of renewed interest in dynamic PRA (DPRA) methods, as seen by the papers at conferences such as the American Nuclear Society's Probabilistic Safety Assessment and Analysis[a] (PSA) conference and the International Association for Probabilistic Safety Assessment and Management[b] (PSAM) conference and journal publications. This is likely due to the availability of more computational resources and the desire for more accurate models. However, there are limitations and costs to using DPRA methods. Classical PRA (CPRA) methods became prevalent in the nuclear industry following the release of the WASH-1400 report [1] and the Three Mile Island incident [**Error! Reference source not found.**]. While CPRA methods have a significant value to industry, because of their static nature, they have limitations as well. While DPRA is not likely to replace CPRA, it does have a valuable contribution to PRA, and the combination of DPRA and CPRA may become more prevalent as time progresses. This report reviews one DPRA method implemented in the Event Modeling Risk Assessment using Linked Diagrams (EMRALD) [**Error! Reference source not found.**] tool and evaluates methods of coupling between it and the CPRA tool Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) [4]. Both tools were developed at Idaho National Laboratory (INL). The goal of this research is to determine if there are coupling methods that can be implemented in EMRALD to help overcome DPRA limitations while still providing the key benefits.

2. CLASSICAL VS. DYNAMIC PRA

There are many features that distinguish DPRA from CPRA with the primary concept being dynamic methods capture behavior over time. This section reviews the two software tools used for this research along with some of the benefits and disadvantages of each.

2.1 SAPHIRE Overview

The development of SAPHIRE was done under funding from U.S. Nuclear Regulatory Commission (NRC). The primary use is for developing and maintaining the Standardized Plant Analysis Risk (SPAR) models. It is also available for public and academic use with NRC approval. SAPHIRE uses classical fault tree and event tree models to represent system reliability and event sequences that occur.

2.1.1 SAPHIRE Results

A fault tree consists of a Boolean logic tree with basic events on the leaf nodes. The basic events represent failure options of components and have a probability associated with them; the gates in the fault tree model demonstrate how those component failures effect the entire system. Fault trees can be solved to give the probability of the entire system failure over a given mission time. The results also show cuts sets, each made up of a set of basic events minimally required to cause a failure and a probability for that specific cut set.

An event tree models an initiating event and different outcomes that can occur called sequences. Each sequence is an evaluation of a fault tree. Solving an event tree results in a frequency for each sequence including the probability of the initiating event. Additional calculations can provide the importance of

[a] https://www.ans.org/meetings/psa2021/

[b] https://www.iapsam.org/

specific basic events or changes using measures such as Birnbaum Risk Increase or Achievement Worth (RAW), or Fussell-Vesley [4].

2.1.2 Classical PRA Benefits and Limitations

As the primary safety assessment method for most nuclear power plants, CPRA benefits are well known. Overall, CPRA models provide an easy way to represent complex systems and quickly perform fine resolution safety analysis. Another primary advantage of CPRA is facilities already have a model and understand its behavior. A common stated limitation of CPRA is it does not consider dynamic or time-related scenarios. While time is not explicitly modeled in CPRA, some modeling techniques are able to capture some scenario dynamics such as adding the probability of recovery within X hours or operator actions. There have been several techniques to capture dynamic behavior or time-related data into CPRA that are important for current industry PRA models [5]. Post-processing methods such as the convolution factors in SAPHIRE adjust the results to capture effects not capable of being captured directly in the model.

However even with these CPRA techniques, there are limitations due to dynamic aspects that cannot be captured. One key limitation is in the results; as stated, the results are a probability of failure within a specified mission time. Specifics on failures are bounded by the mission time. Second, the cut set shows the basic events that can cause the failure but nothing about the order of the events.

There are several modeling limitations; a few include:

- Fixed failure rates – basic events are fixed rates used in the quantification, some features such as phases allow for a different rate in different phases. An example use for adjustable failure rate would be if three pumps in parallel which normally run at 60% with a lower failure rate, but if one goes out, the two can handle the load but have a higher failure rate.
- Looping – CPRA cannot have looping behavior in fault trees or event trees. Many operator actions can have looping whether a step can be repeated if it failed the first time.
- Binning – Many time aspects require binning within 4 hours or X-Y gallons per minute. Bins do a good job for hard cutoff criteria but can limit the precision of the model.
- Early completion – In modeling human factors if there is a series of tasks to complete, the user cannot credit any time for completing an earlier task to give more time for a later task or make up time with faster, later tasks given a long initial task time.

2.1.3 SAPHIRE Programmatic Solve Options

Combining the results from SAPHIRE with EMRALD would require EMRALD to modify and solve SAPHIRE programmatically. There are a several options to do this. One option is modifying the model through Models and Results Database (MAR-D) input and running a SAPHIRE macro. The second option is to use the SAPHIRE remote solver and generate or modify an input file. The SAPHIRE macro option provides more solving and results options but also requires more resources. If repeated calls or solves are needed, the SAPHIRE macro option could cause a prohibitive overhead cost as the entire application must be started and project loaded.

2.1.3.1 MAR-D Input

MAR-D provides an interface to load or extract data files that define the PRA database. The files are in a "flat-file" or ASCII file format. These files can be constructed or modified by hand or programmatically and then used by the macro script to add to or modify a model. The MAR-D file text format and field descriptions are provided in SAPHIRE reference material [6].

Typical uses of MAR-D include:

- Transferring PRA information between databases.
- Extracting MAR-D files from one SAPHIRE project and loading them (via MAR-D) into another SAPHIRE project. (The SAPHIRE project may be a new one or a previously existing project.)
- Importing other PRA code information.
- Formatting the model information from another PRA code to use MAR-D file formats and creating a SAPHIRE project by loading the files via MAR-D.
- Editing PRA files using a text editor.
- Extracting MAR-D files from a SAPHIRE project, editing the files to make changes to the model or model descriptions, and loading those files (via MAR-D) back into the SAPHIRE project.
- Archiving PRA files.
- Saving the MAR-D files for long-term storage in a text format rather than the native binary SAPHIRE format.

Details on the MAR-D formatting rules are also found in the SAPHIRE reference material [6]. Several MAR-D input files are used to update the selected SAPHIRE project. There are several different file types, each one specifying data for different items in the SAPHIRE model. For example, Event Tree Names/Descriptions can be added with an *.ETD file. A list of the file types and what they specify in the SAPHIRE model can also be found in the SAPHIRE reference material [6]**Error! Reference source not found.**.

2.1.3.2 SAPHIRE Macros

Macros are scripts that can run automated SAPHIRE 8 routines. Macros can be imported, exported, and run through the user interface, or they can be placed on the command line as SAPHIRE is started. Various functions and reports can be automated using these macros. The macros are written in a language very similar to the XML language which defines classes, verbs, and parameters.

Classes correspond to PRA data objects like projects, fault trees, basic events, event trees, change sets, and end states. Verbs are functions that can be performed on the various data objects. Examples of verbs would be cut-set generation for event trees or fault trees, event trees linking, or uncertainty analysis on generated cut sets. Parameters are used to shape the functions being performed by defining a truncation value, report name, or the sample size of an uncertainty analysis. There are over 250 keywords that define these classes, verbs, and parameters.

Below is a collection of keywords, a macro, that tells SAPHIRE to select an event tree and link it to create sequences.

```
<event tree>                      Opens the class of PRA object (event tree) we are going to work
                                  with.
  <unmark></unmark>               Verb to make sure no event tree is marked.
  <mark mask>LOSP</mark mask>     Verb to select or mark the event tree (LOSP).
  <unlink></unlink>               Verb to remove all sequences of this event tree.
  <link></link>                   Verb to link the marked event tree.
 </event tree>                    Close the class of PRA object we worked with.
```

It will then solve those sequences with a designated truncation value and then produce a summary report showing the overall answer and number of cut sets for each sequence.

```
<sequence>                                                  Opens the class of PRA object (event tree) we
                                                            are going to work with.
  <unmark></unmark>                                         Verb to make sure no event tree is marked.
  <include>                                                 Super verb for explicit marking.
    <mark event tree mask>LOSP</mark event tree mask>       Verb to select the LOSP event tree.
    <mask operation>and</mask operation>                    Verb with parameter to define marking.
    <mark sequence mask>*</mark sequence mask>              Verb to select all sequences in selected event
                                                            tree.
    <mask operation>and</mask operation>                    Verb with parameter to define marking.
    <mark logic fault tree>*</mark logic fault tree>        Verb to select any sequences using the marked
                                                            defined fault trees.
  </include>                                                Close of super verb for explicit marking.
<solve>                                                     Open of verb to solve for cut sets and apply post-
                                                            processing rules.
    <with update></with update>                             Parameter to indicate a cut-set update should be
                                                            done.
    <truncation>0.00</truncation>                           Parameter to indicate the truncation to be
                                                            applied.
  </solve>                                                  Close of verb.
  <report>                                                  Open of verb to perform a report.
    <type>use title</type>                                  Parameter defining the report type—use title.
    <title>Sequence Results Compare</title>                 Parameter defining the report title.
    <file name>current_vs_base.html</file name>             Parameter defining the name of the report.
    <report format>html</report format>                     Parameter defining the output type (PDF,
                                                            HTML, etc.).
  </report>                                                 Close of report verb.
</sequence>                                                 Close the class of PRA object we worked with.
```

2.1.3.3 SAPHIRE Remote Solver

The SAPHIRE remote solver was developed to allow remote and distributed solving of SAPHIRE models. There is both a Dynamic Linked Library (DLL) and executable version that take a model input file and save a cut-set results file. The input model and cut-set result file format is a user-readable text-based model using JSON (JavaScript Object Notation).

2.2 EMRALD Overview

Several methods and tools can be used to perform DPRA; some, such as Analysis of Dynamic Accident Progression Tree (ADAPT) [7] and Reactor Analysis and Virtual control Environment (RAVEN) [8], use dynamic event trees that explore all branching in the model. Others such as EMRALD use discrete event simulation to sample the next event in time instead of having specific time steps. Each has different use cases and advantages. EMRALD was chosen for this research since its design is more focused toward industry use and has interface features that can make specifying the coupling links simpler for users.

EMRALD was initially developed at INL to research DPRA methods and then was updated to couple with external physics-based simulation tools for external hazard analysis. Primary goals of its development were to provide modeling concepts corresponding to traditional PRA methods, a simple interface to represent complex interactions, and coupling features needed to solve emerging DPRA problems within the industry [9]. EMRALD is based on a three-phase discrete event analysis, where the simulation jumps to the next event in time [10], which is ideal for simulations that can have varyingly large and small time intervals in their simulation space. Statistical results are determined by compiling the outcome of many Monte-Carlo simulation runs. EMRALD consists of two main pieces, the web-based model builder GUI (graphical user interface) and the solver. Sections 2.2.1 and 2.2.2 describe the basic features and workings of EMRALD that are relevant for possible coupling methods (see “EMRALD, Dynamic PRA for the Traditional Modeler” for more detail) [11].

2.2.1 EMRALD Model Builder Interface

The model building interface was designed to help visualize DPRA flow and make it as similar to traditional PRA as possible. There are many pieces that make up an EMRALD model, they are discussed in the following sub-sections.

2.2.1.1 Diagrams

EMRALD model consists of multiple diagrams. Each diagram represents a particular piece of the model and various conditions or states that this piece of the model can be in. These pieces correlate to aspects of traditional PRA modeling and range from small-scale components to a large scope of plant response and design. A diagram contains multiple states with events that can occur and actions that may be executed. These all define how the current states of the simulation may shift over time.

Additionally, some diagrams (component and state diagrams) can also be evaluated for a Boolean result depending on which state they are currently in. This is a critical feature that, when combined with a Component Logic Event, can greatly simplify a model and prevent exponential size growth that happens with typical Markov models. Unlike the more general diagrams, such as plant response, these diagrams are restricted to only be in one state at a time during the evaluation process as shown in **Error! Reference source not found.**. For example, a pump cannot be both active and fail at the same time.

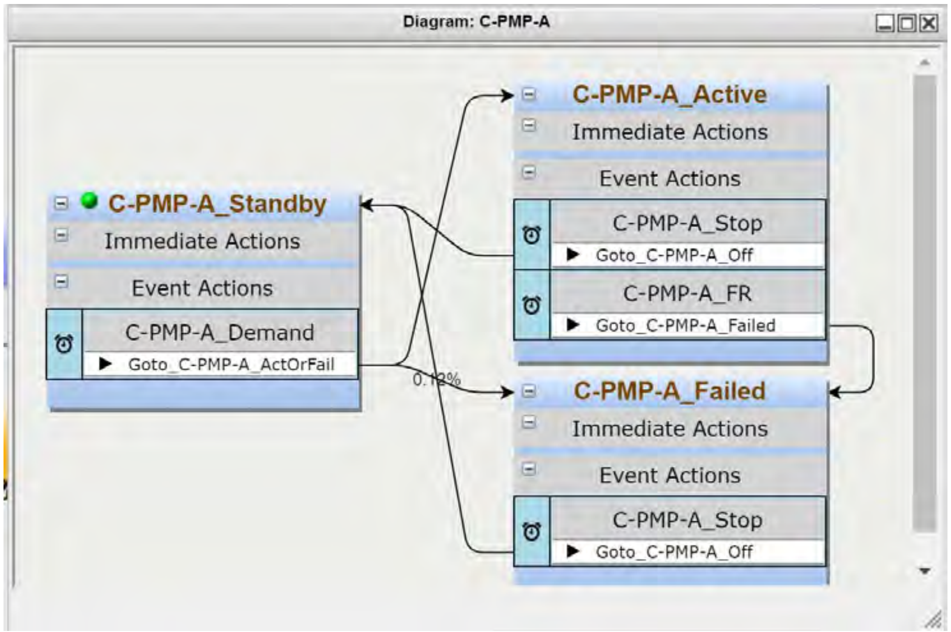


Figure 1. Example of a component diagram and the different states it can be in.

Component diagrams are critical for any type of coupling with other applications or with a CPRA. There can be a direct correlation between components in EMRALD and a corresponding CPRA model. In each EMRALD simulation, you know what state each component is in and can use the Boolean value to for driving other analysis.

2.2.1.2 State Events and Actions

States are a logical representation for a current condition in a diagram. Each state has or can have the following attributes:

- Type – Start, Standard, Key State, or Terminal, which establishes simulation behavior when setting up, running, or post-processing a simulation run
- Immediate Actions – Actions that are executed when entering the state
- Event Actions – Items to monitor when in this state which can then trigger one or more actions.

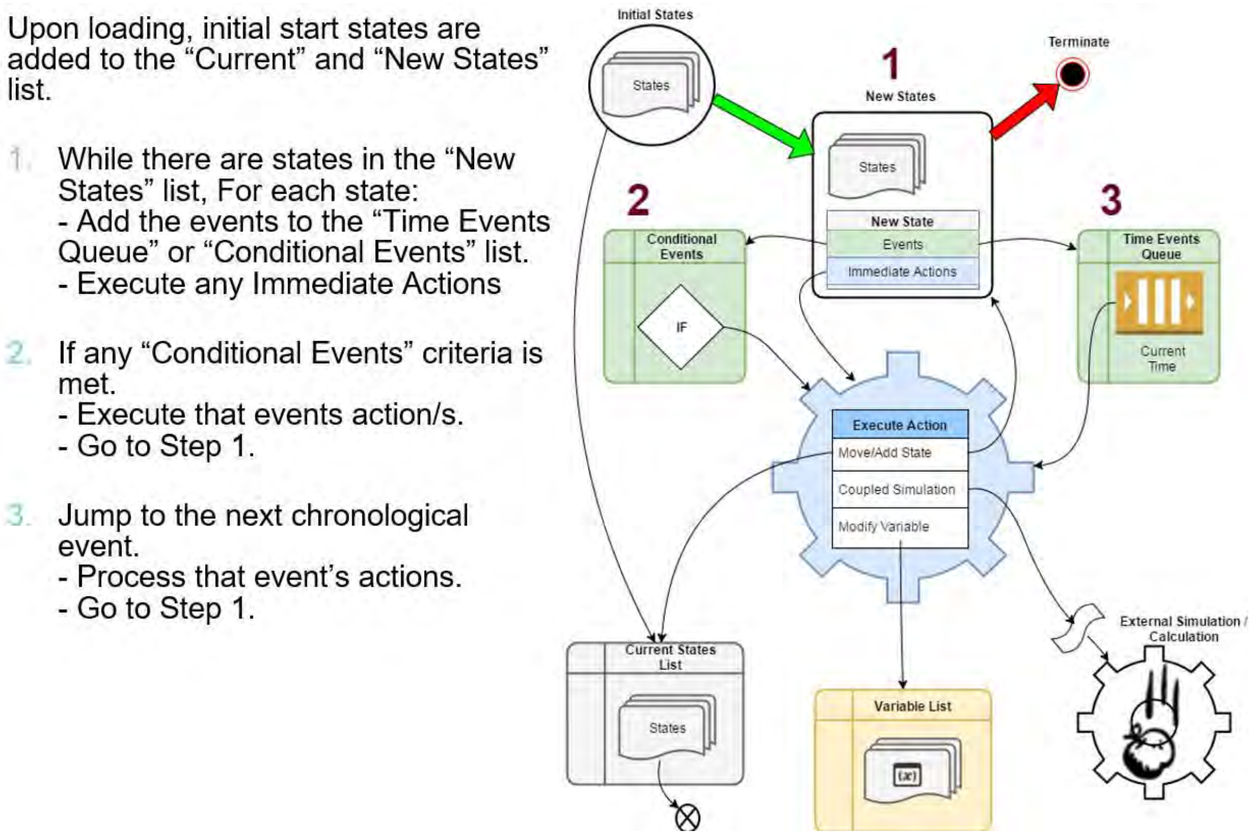
Events are what a state is looking for to trigger something. There are two categories of events: time based and condition based. Time-based events sample the next time to failure, such as a probability distribution. Condition events are evaluated every time something that could impact the event changes, such as evaluating if a variable is below a specified value. If a state is marked as a key state, then this state is tracked for results.

Actions are things that can be done in the simulation space, such as changing a variable value, transitioning to a different state, or running a thermal hydraulics code. Actions are taken when entering a state or are linked to an event and are executed after that event occurs.

Whenever EMRALD enters a new state during a simulation, the immediate actions are executed, and the events are either sampled and put in the time queue or added to an evaluation list to see if they are triggered. When exiting a state, all its events are no longer valid and are removed. Additional events or actions could be added to EMRALD to handle coupling with SAPHIRE and process results.

2.2.2 EMRALD Simulation

As stated, EMRALD is based on three-phase discrete event simulation with the evaluation steps shown in **Error! Reference source not found.**. There is no time stepping in a discrete event, instead it just jumps to the next event in time. This means that scenarios with both long and very long time durations can be modeled without a significant increase in computation time.



Upon loading, initial start states are added to the “Current” and “New States” list.

1. While there are states in the “New States” list, For each state:
   - Add the events to the “Time Events Queue” or “Conditional Events” list.
   - Execute any Immediate Actions

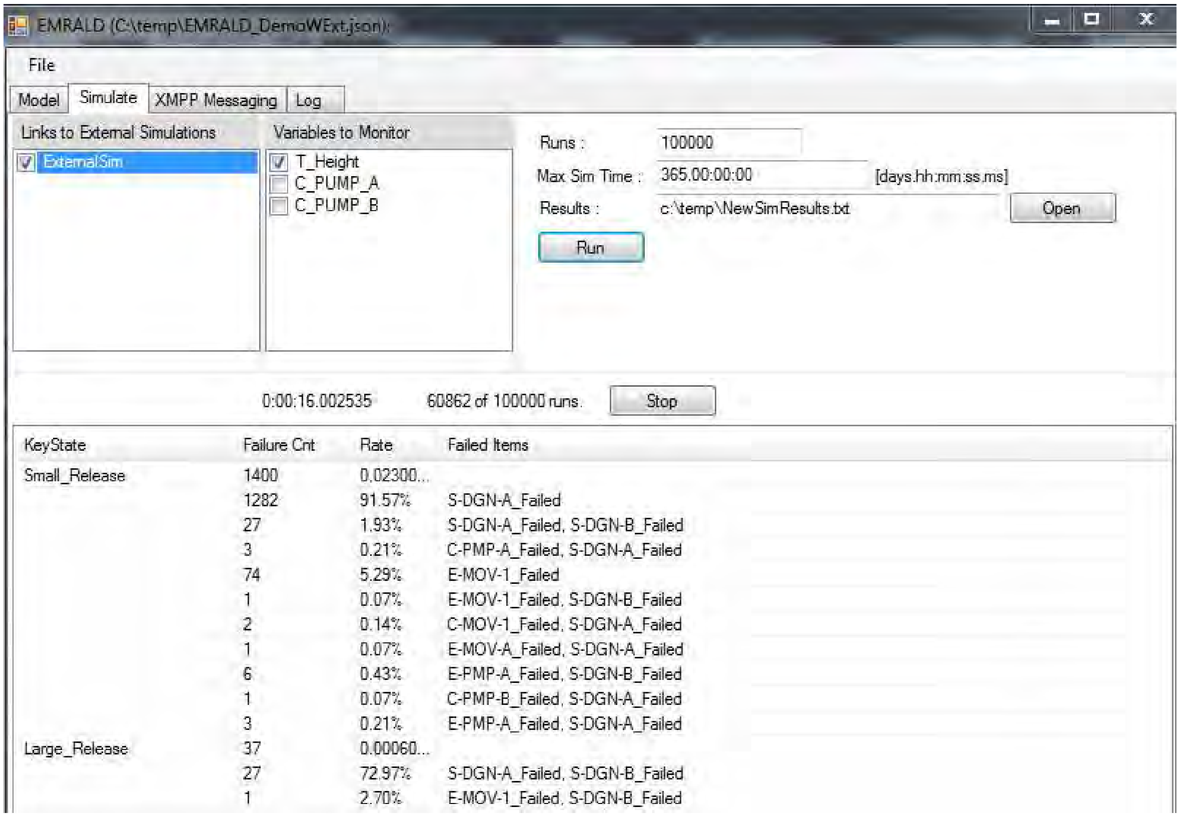
2. If any “Conditional Events” criteria is met.
   - Execute that events action/s.
   - Go to Step 1.

3. Jump to the next chronological event.
   - Process that event’s actions.
   - Go to Step 1.

Figure 2. EMRALD simulation process, based on three-phase discrete event simulation.

Each simulation loads the model and processes through the steps until a terminal state or the max simulation time (similar to a mission time in CPRA) is reached. When a simulation ends, any states that the simulation is currently in are saved for the results. The path and timing for how that state entered are saved. With multiple simulation runs, statistical data can determine the probability of ending in a key state vs. how many simulation runs were executed.

The solve engine shown in **Error! Reference source not found.**, is used to run the model. The user specifies the max simulation time and any variables they want to monitor. Compiling the results shows all the paths and timings that caused the model to end in one of the key states. Results can also show the different combinations of component states that were encountered for the key states, as shown in the bottom section of **Error! Reference source not found.**. This is similar to cut sets in CPRA, except that they are not minimal and may not directly contribute to the conditions causing the system reaching the key state. For example, just because a pump failed in a simulation run and shows in the result, it could have been the diesel generator and operator actions that caused the key state. Meaning the key state was entered because the simultaneous failure of the diesel generator and operator action and failed pump was just a random failure that has also occurred, but it has no effect on system’s transition to the key state.


EMRALD (C:\temp\EMRALD_DemoWExt.json)
File
Model
Simulate
XMPP Messaging
Log
Links to External Simulations
Variables to Monitor
Run
Open
Stop

☑ ExternalSim

☑ T_Height
□ C_PUMP_A
□ C_PUMP_B

Runs : 100000
Max Sim Time : 365.00:00:00 [days.hh:mm:ss.ms]
Results : c:\temp\NewSimResults.txt

0:00:16.002535 60862 of 100000 runs.

KeyState	Failure Cnt	Rate	Failed Items
Small_Release	1400	0.02300...	
	1282	91.57%	S-DGN-A_Failed
	27	1.93%	S-DGN-A_Failed, S-DGN-B_Failed
	3	0.21%	C-PMP-A_Failed, S-DGN-A_Failed
	74	5.29%	E-MOV-1_Failed
	1	0.07%	E-MOV-1_Failed, S-DGN-B_Failed
	2	0.14%	C-MOV-1_Failed, S-DGN-A_Failed
	1	0.07%	E-MOV-A_Failed, S-DGN-A_Failed
	6	0.43%	E-PMP-A_Failed, S-DGN-B_Failed
	1	0.07%	C-PMP-B_Failed, S-DGN-A_Failed
	3	0.21%	E-PMP-A_Failed, S-DGN-A_Failed
Large_Release	37	0.00060...	
	27	72.97%	S-DGN-A_Failed, S-DGN-B_Failed
	1	2.70%	E-MOV-1_Failed, S-DGN-B_Failed

Figure 3. EMRALD solve engine interface and summary results.

The solve engine and result calculations would be modified to handle coupling or execution CPRA and incorporate the results.

2.2.3 DPRA Features and Limitations

DPRA has many features that can be useful for risk analysis, but it also has limitations that keep it from having widespread usage.

- **Variables and Adjustable Failure Rate** – Depending on time or other conditions, the failure rate of a component or failure method could change. EMRALD makes this simple to model through the use of variables as parameters in events. If a variable changes, the event can be resampled or correctly adjusted [12] according to the new value. If evaluating a variable, that variable can be updated from an external simulation which would dynamically adjust the model.
- **Linked or Dependent Failures** – Special post-processing rules are needed to handle many things in CPRA, such as convolution factors [13]. DPRA modeling in EMRALD automatically captures this behavior.
- **Looping** – Feedback loops such as control logic or operators repeating an action if they fail the first time are easily modeled in EMRALD.
- **External Analysis Coupling** – There are often other analysis tools or complex calculation scripts that are used for analysis such as thermal hydraulics. EMRALD has methods to do both one- and two-way coupling. Variables can even be directly linked to an input or export file to be updated when the variable or file changes.
- **Path Results and Timing** – EMRALD results provide the path and timing for each scenario caused the outcome. This timing can show bottlenecks or critical areas in time that cause an outcome. A

time-based Sankey diagram currently in development shows the paths and statistics of time spent in each state.

- **Computational Costs** – A critical issue preventing widespread use of DPRA is the computational costs. There is really no limit on how long a model could take to run, depending how many runs were needed and the properties of the model. If events with very low probabilities are used, then more runs are needed to capture results that include those events.
- **Model Complexity and Errors** – While a good graphical interface can help visualize the model, it can be difficult to develop and debug complex models. Often errors are only encountered in a specific simulation run that makes a rare sample, and this can make it difficult to find all errors before a full simulation is run. EMRALD does have a debug log that can be turned on to track all the simulation steps for a specified run.

3. COMBINING DPRA AND CPRA

The goal of combining DPRA and CPRA is to gain the advantages of each while eliminating as many of the limitations as possible. For CPRA, we primarily want the use of existing models and the relatively fast solving speeds, along with importance measure calculations. On the DPRA side, we want the integrated time relationship along with the ability to model dynamic or complex scenarios while keeping the model simple and fast.

Ideally if you encounter a scenario that requires a dynamic model, you want to model the scenarios events and time dependent aspects in relation to the DPRA model, pulling in any dependent items from the CPRA model. The final result would be a combination of the DPRA results and CPRA results. This section goes over several options and trials in coupling and combining EMRALD and SAPHIRE models/results and the benefits and limitations or fallacies of each.

3.1 CPRA Fault Tree Link as DPRA Event

One idea would be to create a new type of event in EMRALD similar to a failure rate event—except instead of defining the value, you would link it to a SAPHIRE fault tree. When entering a state with the event in the EMRALD model, instead of EMRALD immediately sampling to determine the next time of the event, it would modify the SAPHIRE fault tree with any components currently failed and then call SAPHIRE to solve the modified tree. The returned probability would be converted from probability of failure to a rate using the cumulative failure distribution $F(t) = 1 - e^{-\lambda t}$ and solving for lambda, the value would then be sampled by EMRALD to determine the time of the event.

At first glance, it may seem like a good solution to model all the time dependent aspects pulling out the pieces needed from the CPRA and using the CPRA to quickly calculate the rest of the model. Technically, this could work but only for a model that did not have any fails-to-start events, all failure rates were memoryless, and the mission time of the SAPHIRE model is somehow modified to match the remaining time because the cumulative failure distribution, if not starting at time 0, is $e^{-\lambda t0} - e^{-\lambda t}$. In other words, this method would not work for any practical or useful model. Lenz explains these issues in depth in *Reliability Calculations for Complex Systems* [14].

The first problem is failure-to-start basic events in the CPRA model. In the EMRALD model, dynamically solving the fault tree could mean that you are already past these basic events. In this case, could all the failure-to-start events also be removed or ignored in the PRA? In another case, if the EMRALD model is already past the time where these basic events would normally occur, and it managed to negate them, then did the DPRA account the consequence of those events?

Failure distributions account for the second problem. If time has already passed in the EMRALD model, and we want to use the CPRA model, the distributions in the CPRA would have to account for the time already passed. If all the distributions are exponential and memoryless, this is not an issue; however, for others, they would need to be adjusted to account for this time.

Finally, there is the overall mission time. CPRA calculates the failure probability according to a specified mission time. In EMRALD, you can specify the total run time, similar to a mission time. In theory, this could be used to modify the CPRA mission time, but other events can also cause an end to a simulation run.

Testing in EMRALD was done to determine how to specify a link to a SAPHIRE fault tree and to modify basic event failure probabilities. Initial testing was done on the back end of EMRALD, using a new event type. However, this branch of work was stopped due to the unresolved issues described above. It is not feasible in most instances and not even be numerically possible to couple DPRA with CPRA through this methodology.

3.2 Hybrid Fault Tree / Event Tree

A recent paper by Mandelli outlines a method of linking CPRA with DPRA through a hybrid fault tree approach [15]. This method uses only the Boolean logic from the model, ignoring the probabilities. The failure times for components are added to the Boolean logic, and it is used to calculate if and when the entire system fails. This can be useful for some analysis if you just need the time of some system failure for a secondary process; in this paper's research, it was used to quickly determine when a cooling system has failed for running a thermal hydraulics analysis.

This method was shown using RAVEN, which uses a different method for DPRA than EMRALD. The key difference for the hybrid fault tree is that RAVEN samples all the distributions at the beginning of a run. EMRALD events are sampled when the simulation enters the state where they are relevant. EMRALD has no future knowledge past the simulations' current states and their relevant events. The next events in time are from the current states, and when a state is exited. those events are thrown away. Due to the discrete event process, the hybrid process outlined by Mandelli will not work for EMRALD.

3.3 Adjusting DPRA Results

While CPRA can calculate the overall risk or core damage frequency (CDF) of a plant, an EMRALD model typically models a specific dynamic scenario with relatively few states or events compared to CPRA. There could be other non-dynamic safety systems not included in the EMRALD model. Another method of coupling would be to multiply the probability of failing in the EMRALD model by the remaining CPRA model results. The end time of a simulation would need to match the mission time for a CPRA model.

As discussed in Section 2.2, there are two levels of results in EMRALD: the detailed time-based paths results and a basic probability of ending in a key state when the simulation ends. The key state probability can also be viewed in groups according to components that have failed. While these failures are not the same as CPRA cut sets, they do indicate what components have failed and the contribution that the unique set has to that key state probability. These failures can be used to modify the CPRA model and calculate a probability of failure given the failures of EMRALD model. The results would not be equivalent to the CPRA results but would be the probability of the dynamic event occurring and the other safety systems failing.

For example, you have the following cut sets from a CPRA result:

- CPRA Cut Sets = (A + B*C + B*D).

The DPRA model uses the component C in its model and C a failed event in the EMRALD key state, so that event is set to "True" in the CPRA model and solved:

- EMRALD Key State Failures (C)

- Modified CPRA Cut Set = (A +B).

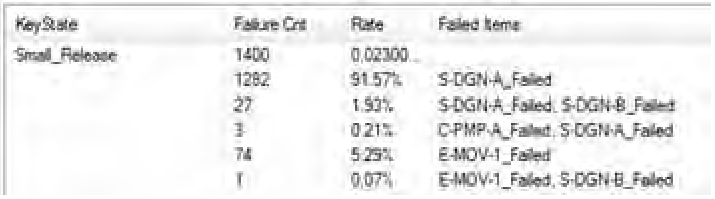
The combined result is the DPRA result probability times the modified CPRA results:

- Combined Results = (DPRA Prob) * (A + B).

The process for this coupling is to assign a fault tree or event tree to an EMRALD key state. Once the EMRALD simulation is complete, the following steps are done for each unique component failure combination (as seen in **Error! Reference source not found.** in the EMRALD results):

1. Modify the SAPHIRE model—set all common components to true or false, according to the EMRALD model
2. Calculate the SAPHIRE result
3. Adjust the EMRALD result by multiplying it with the SAPHIRE result.

The final key state result is the sum all the modified unique combinations results.

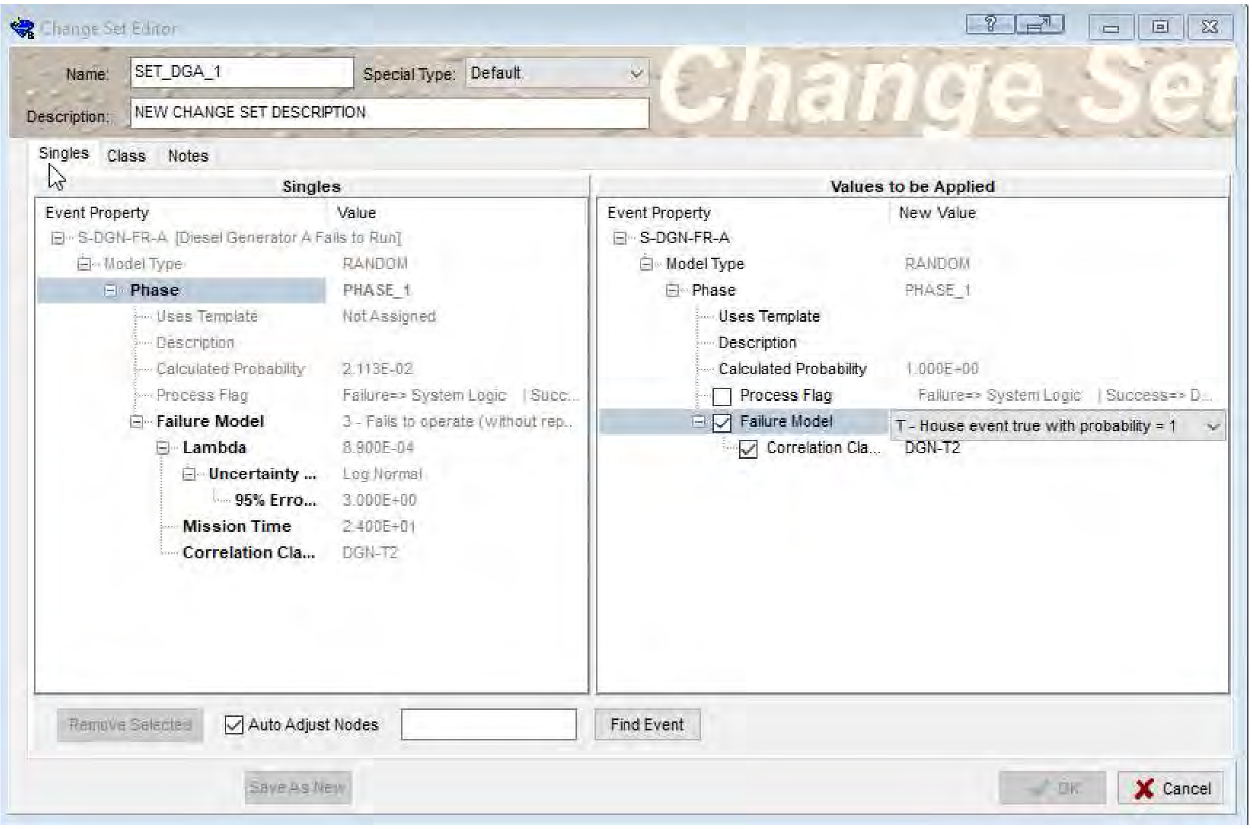


KeyState	Failure Cnt	Rate	Failed Items
Small_Release	1400	0.02300...	
	1282	91.57%	S-DGN-A_Failed
	27	1.93%	S-DGN-A_Failed, S-DGN-B_Failed
	3	0.21%	C-PMP-A_Failed, S-DGN-A_Failed
	74	5.29%	E-MOV-1_Failed
	1	0.07%	E-MOV-1_Failed, S-DGN-B_Failed

Figure 4. Figure 5. EMRALD results showing the failed component groups for the key state of Small_Release.

A SAPHIRE model can be modified to calculate similar results, minus any dynamic modeling, using the following steps (shown in Figure 6 to Figure 9):

1. Create a change set to set the common events to true or false (Figure 6)
2. Create an initiating event with the probability of the common events (Figure 7)
3. Create an event tree with the new initiating event and the applicable fault trees (Figure 8)
4. Solve the event tree with the change set applied (Figure 9).



Name: SET_DGA_1
Special Type: Default
Description: NEW CHANGE SET DESCRIPTION

Singles

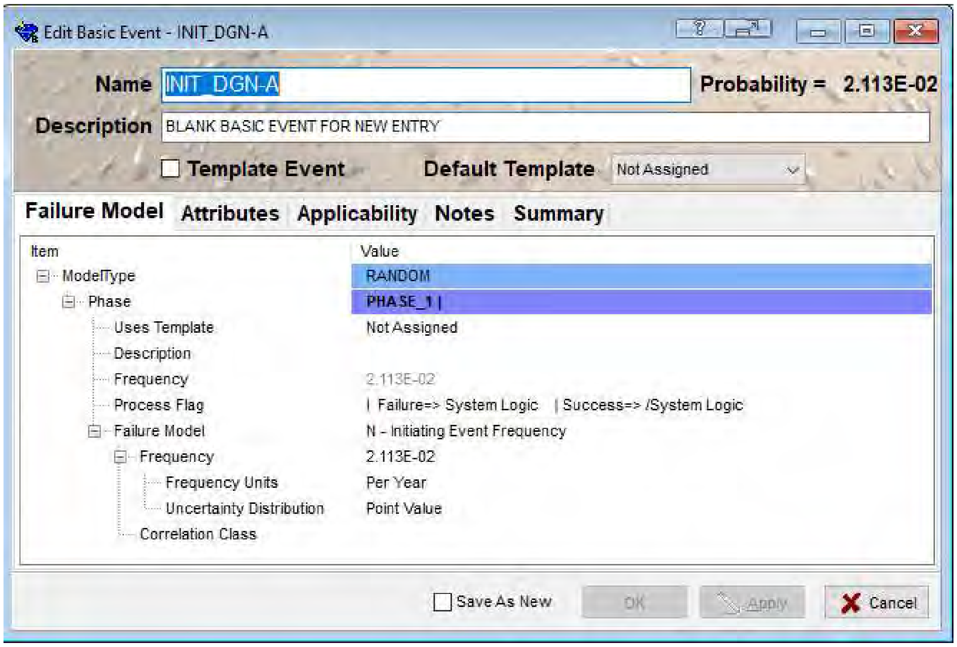
Event Property	Value
S-DGN-FR-A [Diesel Generator A Fails to Run]	
Model Type	RANDOM
Phase	PHASE_1
Uses Template	Not Assigned
Description	
Calculated Probability	2.113E-02
Process Flag	Failure=> System Logic | Succ...
Failure Model	3 - Fails to operate (without rep...
Lambda	8.900E-04
Uncertainty ...	Log Normal
95% Erro...	3.000E+00
Mission Time	2.400E+01
Correlation Cla...	DGN-T2

Values to be Applied

Event Property	New Value
S-DGN-FR-A	
Model Type	RANDOM
Phase	PHASE_1
Uses Template	
Description	
Calculated Probability	1.000E+00
□ Process Flag	Failure=> System Logic | Success=> D...
☑ Failure Model	T - House event true with probability = 1
☑ Correlation Cla...	DGN-T2

☑ Auto Adjust Nodes

Figure 6. A change set to modify the diesel generator A to True.



Name INIT_DGN-A
Probability = 2.113E-02
Description BLANK BASIC EVENT FOR NEW ENTRY
□ Template Event
Default Template Not Assigned

Item	Value
ModelType	RANDOM
Phase	PHASE_1 |
Uses Template	Not Assigned
Description	
Frequency	2.113E-02
Process Flag	| Failure=> System Logic | Success=> /System Logic
Failure Model	N - Initiating Event Frequency
Frequency	2.113E-02
Frequency Units	Per Year
Uncertainty Distribution	Point Value
Correlation Class	

□ Save As New

Figure 7. Initiating event with value from removed events.

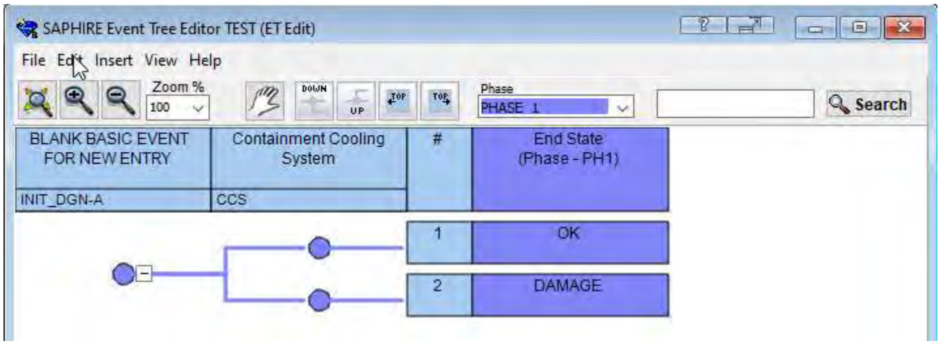
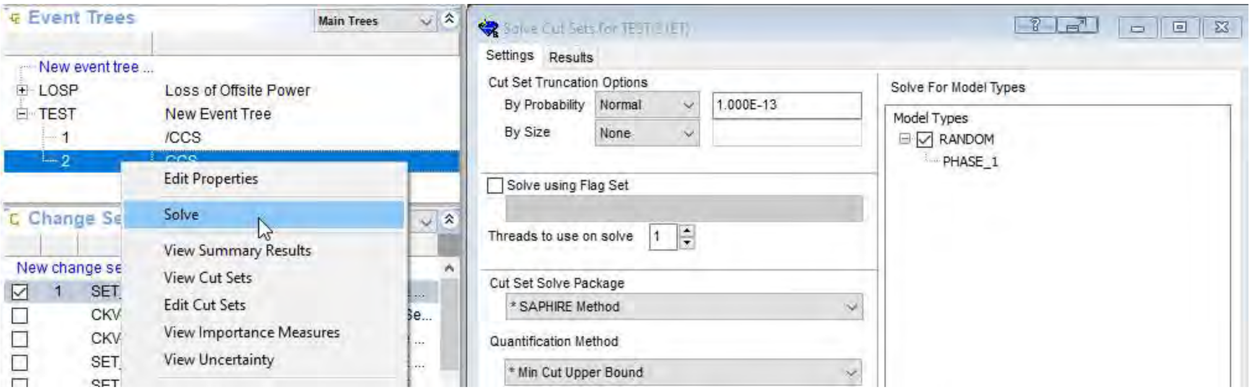

SAPHIRE Event Tree Editor TEST (ET Edit)
File Edit Insert View Help
Zoom %
100
DOWN
UP
TOP
Phase
PHASE_1
Search
BLANK BASIC EVENT FOR NEW ENTRY
INIT_DGN-A
Containment Cooling System
CCS
#
End State (Phase - PH1)
1
OK
2
DAMAGE

Figure 8. New SAPHIRE event tree with initiating event and affected fault trees.



Cut Set Truncation Options

By Probability Normal 1.000E-13

By Size None

□ Solve using Flag Set

Threads to use on solve 1

Cut Set Solve Package * SAPHIRE Method

Quantification Method * Min Cut Upper Bound

Model Types ☑ RANDOM PHASE_1

Figure 9. Solve of the event tree damage sequence applying the change set.

Besides the normal DPRA benefits, this coupling method would allow the analyst to evaluate the probability the dynamic scenario and the rest of the systems failing. This capability would be ideal for evaluating external hazards that have operator action and other dynamic pieces. However, this is limited to specific scenarios and cannot be used for more general or overall PRA analysis. A method for modifying and solving a SAPHIRE fault tree was developed in EMRALD. This allows EMRALD to get the SAPHIRE result that needs to be applied for this method. This was done in back-end testing only, and work needs to be done to allow the user to link a specified fault tree and component models to basic events.

3.4 Merging Model Results

Given the direct coupling restrictions described in the previous sub sections, a merging method external to both EMRALD and SAPHIRE could be ideal. Users may need results from both a DPRA and CPRA model. In some cases, using a dynamic model may be better to determine the initiating event needed in a CPRA event tree. An example of this would be the loss of service water initiating event. Here there are two pumps, each running 50% of the time. Switching between items and items with operator procedures can be modeled more realistically with DPRA. In other modeling cases, you may have exactly opposite, where the user wants to apply an initiating event frequency to key state results in EMRALD. An example of this would be in the case of a rare initiating event where if the user adds it to the EMRALD model directly they would have to run orders of magnitude more simulation runs to have a good convergence or low uncertainty in the results.

In cases where there is no overlapping between the CPRA and DPRA model, results could be just a specific combination of probabilities from each. In more complex scenarios, some methods can be used to apply differences from a DPRA to restructure the CPRA model, augmenting the results. Several methods for specific applications are discussed in detail in *Mutual Integration of Classical and Dynamic PRA*

[**Error! Reference source not found.**]. These methods focus on applying DPRA results back into the CPRA.

A possible issue in effectively merging models is the link between the models must be maintained to make sure models stay synced. Without a link, changes to either model could break the results merging. As the use of DPRA in industry is relatively low, common use cases and needs must be established before a sound process or tool for merging model results could be effectively developed.

4. DEVELOPMENT

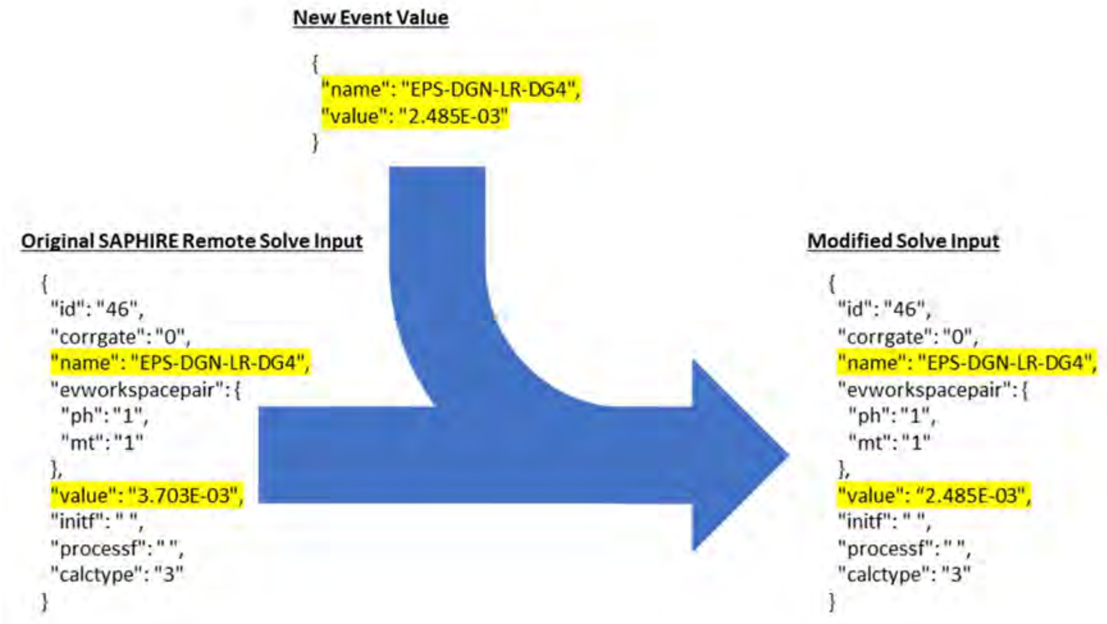
No specific development was done in SAPHIRE, and minimal effort is expected for future work. The macro options provide in-depth methods for modifying and solving; however, retrieving results through this method is limited, and simple results output may need added. A SAPHIRE macro using commands shown in Section **Error! Reference source not found.** was created and used to do some initial testing, and it took several minutes to run basic execution of some fault trees using a generic reactor model.

The remote solver provides basic solve functionality without the time costs of the macro option overhead. It reads a custom data input in a JSON format called a JSInp file; this provides a simple way to modify input and get results. However, using the remote solver requires the model to be saved inside the EMRALD model, so that it can be modified and sent to the solver. Generating a remote solve input file for a specific fault tree or event tree requires the user to solve it inside SAPHIRE and retrieve the file from a temporary location. Using a macro the EMRALD model could reference a general SAPHIRE model to generate results.

4.1 Solving in SAPHIRE

Several features were added to EMRALD to enable coupling testing and solving of a SAPHIRE model. For all the coupling methods, the SAPHIRE remote solver was used, and the executable was included in the EMRALD application directory. To link EMRALD pieces to SAPHIRE fault tree models, the JSON file was imbedded as part of the EMRALD model to be used as the template or base for modifications. The SAPHIRE fault tree model contains a list of events, with each event containing JSON tokens for each specific detail. For the purpose of solving a model modified with EMRALD, the "name" and "value" tokens are of particular interest.

For EMRALD, a simple find-and-replace algorithm using JSON tools is used to update event values. The algorithm takes a list of key-value pairs, shown as “New Event Value” in Figure 10, where the key is the event name, and the value is the new value to be associated with that event. Using this list of event names and values, the algorithm uses a given SAPHIRE input file, finds references to each event name in the list and, andupdates that event's value. This is shown in the following figure.


New Event Value

```
{
 "name": "EPS-DGN-LR-DG4",
 "value": "2.485E-03"
}
```

Original SAPHIRE Remote Solve Input

```
{
 "id": "46",
 "corrgate": "0",
 "name": "EPS-DGN-LR-DG4",
 "evworkspacepair": {
   "ph": "1",
   "mt": "1"
 },
 "value": "3.703E-03",
 "initf": " ",
 "processf": " ",
 "calctype": "3"
}
```

Modified Solve Input

```
{
 "id": "46",
 "corrgate": "0",
 "name": "EPS-DGN-LR-DG4",
 "evworkspacepair": {
   "ph": "1",
   "mt": "1"
 },
 "value": “2.485E-03",
 "initf": " ",
 "processf": " ",
 "calctype": "3"
}
```

Figure 10. Example of modification method to modify SAPHIRE remote solve input file.

Using this method, EMRALD modifies the SAPHIRE model, saves it, and passes it as a parameter to the remote solver. The remote solver results are also a JSON file, and the document link variables in EMRALD can connect directly to the cut-set result value, or the results are read by the solve engine to be applied as needed.

No development was done on the model builder web interface for this testing. To implement a coupling method, further development will be needed to add options to the model builder and produce results in the solve engine.

5. RESULTS AND INSIGHTS

The goal of this research was to determine coupling methods that would make DPRA more feasible for industry use by providing coupling methods to CPRA tools and existing models. This research resulted in the evaluation of several coupling methods. Both a CPRA link as an EMRALD event and hybrid fault tree methods would not work numerically for useful analysis in EMRALD. A merging of model results would be a large endeavor requiring new software or extensive modifications to both EMRALD and SAPHIRE. The only method currently feasible is the adjust results method, which will be useful for improving specific scenarios results. One large benefit of implementing this method could come from the external hazards area where operator actions and the time between component failures and tasks would be critical. EMRALD has been used for modeling external hazards and operator actions [17] [18] directly incorporating the CPRA thus making these methods more applicable for industry use.

5.1 Future Development

This initial researched focused on methods and feasibility of implementing those methods using EMRALD and SAPHIRE. Back-end software development was done for the research, but no user interface or optimization work was done. The next step is the design and implementation for the adjusted-results coupling option in EMRALD. This would include options in the web interface for attaching SAPHIRE model pieces to key states in EMRALD and changes to the solve engine for coupled solving and results. After the capabilities are added, demonstration cases using an external hazard and a generic plant model will be developed to show the potential benefits of DPRA and CPRA coupling.

Industry use of DPRA is increasing, but more data on valuable scenarios or use cases must be established before determining what coupling or combined results are valuable. EMRALD simplifies many aspects of DPRA, and this is key to general DPRA use by industry. During this research, several features were identified to assist in simpler development or features to make EMRALD's DPRA more effective:

- Key State Initiating Value – Add an option to apply an initiating value to a key state. This would be similar to an initiating event frequency in CPRA. This would allow the EMRALD model to run more simulations hitting critical events while still providing an accurate overall result.
- Uncertainty – Add uncertainty distribution options to failure rate events.
- Auto Create Component Pieces – Allow the user to select default states and events for simple component diagrams. Such as "Standby," "Running," and "Failed" states, along with fails-to-start and fails-to-run events.
- Model Level Mission Time – Allow simple setting of the mission time in the model development not just in the solver.

The "Key State Initiating Value" feature in the first bullet is a very simple way of initial coupling. As described in Section 3.4, this could be linked to a SAPHIRE or other CPRA model so that this value is up to date. Solving the EMRALD model would automatically use the latest value from CPRA model.

6. REFERENCES

1. Bartel, R. 2016. "WASH-1400 The Reactor Safety Study The Introduction of Risk Assessment to the Regulation of Nuclear Reactors." NUREG/KM-0010, U.S. Nuclear Regulatory Commission (U.S. NRC). https://www.nrc.gov/docs/ML1622/ML16225A002.pdf.
2. Keller, W. and M. Modarres. 2005. "A historical overview of probabilistic risk assessment development and its use in the nuclear power industry: a tribute to the late Professor Norman Carl Rasmussen." *Reliability Engineering & System Safety* 89 (3): 271–285. https://doi.org/10.1016/j.ress.2004.08.022.
3. Idaho National Laboratory. n.d. "EMRALD." Accessed November 30, 2022. https://emrald.inl.gov/SitePages/Overview.aspx.
4. U.S. NRC. 2021. "Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 8." NUREG/CR-7039, U.S. NRC. https://www.nrc.gov/reading-rm/doccollections/nuregs/contract/cr7039/index.html.
5. Bley, D. C., D. R. Buttemer, and J. W. Stetkar, "Light water reactor sequence timing: its significance to probabilistic safety assessment modeling," Reliability Engineering and System Safety, 22, 27-60, 1988.
6. U.S. NRC. "Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 8: Data Loading." (June 2011). https://www.nrc.gov/reading-rm/doc-collections/nuregs/contract/cr7039/v7/index.html.
7. Jankovsky, Z., M. Denman, and T. Aldemir, "Recent Analysis and Capability Enhancements to the ADAPT Dynamic Event Tree Driver," in *Probabilistic Safety Assessment and Management*, Los Angeles, CA.
8. Alfonsi, A., C. Rabiti, D. Mandelli, J. J. Cogliati, and R. A. Kinoshita. 2013. "RAVEN as a Tool for Dynamic Probabilistic Risk Assessment: Software Overview." INL/CON-13-28291, Idaho National Laboratory.

9. Prescott, S., C. Smith, L. Vang. 2018. “EMRALD, Dynamic PRA for the Traditional Modeler,” presented at the Probabilistic Safety Assessment and Management (PSAM14), Los Angeles, CA, Sep. 2018, p. 12. http://www.iapsam.org/psam14/proceedings/paper/paper_76_1.pdf.

10. Pidd, M. 1995. “Object-Orientation, Discrete Simulation and the Three-Phase Approach,” *The Journal of the Operational Research Society* 46 (3): 362. http://doi.org/10.2307/2584330.

11. “EMRALD, Dynamic PRA for the Traditional Modeler.” PSAM 14 http://www.iapsam.org/psam14/proceedings/paper/paper_76_1.pdf.

12. Prescott, S., C. Smith, and L. Vang. 2018. "Probabilistic Methods for Cyclical and Coupled Systems with Changing Failure Rates" Presented at Probabilistic Safety Assessment and Management PSAM 14, Los Angeles, CA, September 2018. https://www.iapsam.org/PSAM16/slides/CO73-SLIDES-PSAM16.pdf.

13. Smith, C., T. Wood, J. Knudsen, and Z. Ma. "Overview of the SAPHIRE Probabilistic Risk Analysis Software" INL/CON-16-39539, Idaho National Laboratory. https://www.osti.gov/servlets/purl/1358392.

14. Lenz, M. and J. Rhodin. 2011. “Reliability calculations for complex systems.” LiTH-ISY-EX--11/4441--SE, Institutionen för systemteknik. http://liu.diva-portal.org/smash/get/diva2:433387/FULLTEXT01.pdf.

15. Mandelli, D., et al. 2020. “Linking classical PRA models to a dynamic PRA.” *Annals of Nuclear Energy* 149: 107746. https://www.sciencedirect.com/science/article/pii/S0306454920304448.

16. Mandelli, D. 2020. "Mutual Integration of Classical and Dynamic PRA." *Nuclear Technology* 207 (3): 363-375. https://doi.org/10.1080/00295450.2020.1776030.

17. " Ulrich, T.2020. "Dynamic Modeling of Field Operators in Human Reliability Analysis: An EMRALD and GOMS-HRA Dynamic Model of FLEX Operator Actions" In Advances in Safety Management and Human Performance. New York: Springer International Publishing. https://www.springerprofessional.de/en/dynamic-modeling-of-field-operators-in-human-reliability-analysi/18134622.

18. "Multi-Hazard Advanced Seismic Probabilistic Risk Assessment Tools and Applications" INL/EXT-16- 40055, Idaho National Laboratory.