INL/RPT-24-80303
Revision 0

Light Water Reactor Sustainability Program

An Evaluation of The Dynamic Physical Security Risk Assessment Methodology for Fleet-Wide Applications

September 2024

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, trademark, 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-24-80303
Revision 0

An Evaluation of The Dynamic Physical Security Risk Assessment Methodology for Fleet-Wide Applications

Steven R. Prescott
Robby Christian
Shawn W. St Germain
Vaibhav Yadav
Christopher P. Chwasz

September 2024

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

Page intentionally left blank

SUMMARY

The requirements for U.S. nuclear power plants to maintain a large onsite physical security force contribute to their high operational costs. The cost of maintaining the current physical security posture is approximately 10% of the overall operation and maintenance budget for commercial nuclear power plants. The goal of the Light Water Reactor Sustainability (LWRS) program's physical security pathway is to develop tools, methods, and technologies and provide the technical basis for an optimized physical security posture. The conservatisms built into current security postures may be analyzed and minimized to reduce security costs while still ensuring adequate security and operational safety. The research performed at Idaho National Laboratory within LWRS program's physical security pathway has successfully developed a dynamic force-on-force modeling framework using various computer simulation tools and integrating them with the dynamic assessment Event Modeling Risk Assessment using Linked Diagrams (EMRALD) tool. This integrated process for physical security analysis is named Modeling and Analysis for Safety Security using Dynamic EMRALD Framework (MASS-DEF).

This document provides an update on the progress in applying the MASS-DEF process to an operating commercial nuclear power plant as well as additional industry feedback regarding use of the tool for other physical security risk-informed topics. This report is only a summary of the progress and does not contain specific modeling results as those contain sensitive security information. Previous reports described how a user could integrate their plant-specific force-on-force models with the dynamic simulation tool EMRALD, model operator actions, and integrate with probabilistic risk assessment tools, such as CAFTA (Computer Aided Fault Tree Analysis System) or SAPHIRE (Systems Analysis Programs for Hands-on Integrated Reliability Evaluations), and with thermal-hydraulic tools, such as RELAP-5 or MAAP. Previous reports applied various combinations of available simulations codes with EMRALD using generic plant models to demonstrate how to perform the analysis.

This report is an update the progress of applying the dynamic computational framework to an actual nuclear facility using their security scenarios and timelines. This report also provides an update to the procedural guidance for the MASS-DEF process and an overview of the generic models available for use by utilities. This report does not contain any plant's sensitive information and/or safeguards information. This study's purpose was to verify that the results achieved using generic models are similar to actual plant results and refine our guidance on the use of the framework. This assessment enables further analysis, such as what-if scenarios and staff-reduction evaluation, thereby optimizing physical security at plants.

Page intentionally left blank

CONTENTS

FIGURES

Page intentionally left blank

ACRONYMS

BWR	boiling-water reactor
CD	core damage
DOE	Department of Energy
EMRALD	Event Modeling Risk Assessment using Linked Diagrams
EPRI	Electric Power Research Institute
FLEX	diverse and flexible mitigation capability
FoF	force-on-force
INL	Idaho National Laboratory
LWRS	Light Water Reactor Sustainability
MAAP	Modular Accident Analysis Program
MASS-DEF	Modeling and Analysis for Safety and Security using Dynamic EMRALD Framework
MODSIM	Modeling and Simulation
NPP	nuclear power plant
NRC	Nuclear Regulatory Commission
O&M	operation and management
PRA	probabilistic risk assessment
PSP	physical security plan
PWR	pressurized-water reactor
SAPHIRE	Systems Analysis Programs for Hands-on Integrated Reliability Evaluations
SGI	safeguards information

Page intentionally left blank

An Evaluation of the Dynamic Physical Security Risk Assessment Methodology for Fleet-Wide Applications

1. INTRODUCTION

Operation and maintenance (O&M) of several nuclear power plants (NPPs) in the United States have become financially burdensome to the point that the utilities may have to stop operation and retire their plants prior to the expiration of their operating license due to economic pressure. Moreover, the wholesale electricity prices have declined in some markets due to the increased penetration of renewables, such as wind and solar power, and the continued use of natural gas power. This phenomenon reduces NPPs' income from power generation. As a result, NPP operators aim to lower their O&M cost to ensure the plants can continue to produce electricity competitively.

The Department of Energy (DOE) has established the Light Water Reactor Sustainability (LWRS) program to assist NPP operators in sustaining their plant operations. The program has identified that the overall O&M cost to protect NPPs accounts for approximately 7% of the total cost of power generation, with labor accounting for half of this cost [1]. Within this overall labor cost, nearly 20% of it is needed to maintain the labor in physical security forces. The Nuclear Regulatory Commission's (NRC's) security requirements for commercial operating nuclear sites increased exponentially following the September 11th terrorist attacks resulting in a significant increase of onsite response force personnel across the nuclear industry [2]. The plant's response force includes the minimum number of armed responders, as required in 10 CFR 73, and security officers tasked with assigned duties, such as stationary observation/surveillance posts, foot-patrol, roving vehicle patrols, compensatory posts, and other duties as required [3]. Since labor costs continue to rise in the United States, any effort to reduce O&M costs needs to include a reduction in labor.

To support this mission, the LWRS program has established a pathway for physical security research. The physical security pathway aims to lower the cost of physical security through directed research into modeling and simulation, the application of advanced sensors, and the deployment of advanced weapons. These efforts are expected to reduce an NPP's dependency on labor work in the physical security area. Modeling and simulation are used to evaluate the margin inherent in many security postures and identify ways to maintain overall security effectiveness while lowering costs. Two areas identified for evaluation are taking credit for diverse and flexible mitigation capability (FLEX) equipment [4] and actions taken by operators to minimize the possibility of reactor damage during an attack scenario. While FLEX equipment was installed to support a plant's response to natural hazards, such as flooding or earthquakes, this equipment could also be used to provide reactor cooling in response to equipment damage caused by an attack on the plant. Initial studies have shown that it while it may not be cost effective to use actual FLEX equipment, other portable equipment dedicated to security response could be a cost-effective solution. Likewise, there are certain actions plant operators will take when an attack occurs to minimize the chance of core damage (CD). It will take modeling and simulating the reactor core and systems to evaluate the effect these operator actions may have on increasing the coping time of the reactor. This more inclusive process for physical security analysis is named Modeling and Analysis for Safety Security using Dynamic EMRALD (Event Modeling Risk Assessment using Linked Diagrams) Framework (MASS-DEF).

The LWRS research team has been working to apply the MASS-DEF methodology to actual NPPs in the United States [5]. This work extends the previous work by including more scenarios, making proposed changes to an actual plant security model, utilizing recent technical feedback received from the industry, and developing a generic boiling-water reactor (BWR) EMRALD model.

The nuclear industry needs to pursue an optimized plant security posture that considers efficiencies and innovative technologies to help reduce costs while meeting security requirements. Using portable equipment in the plant physical security posture has been identified as one area that holds the potential to optimize the security posture and reduce costs. Previous reports described the modeling and simulation capabilities developed to incorporate the deployment of portable equipment with force-on-force (FoF) modeling of a typical physical security posture at a generic light-water reactor plant. This report describes the lessons learned in applying these methods at a currently operating NPP using actual scenarios, plant models, and input from their plant staff. This report documents improvements made to guidance documents and generic plant models for both pressurized-water reactors (PWRs) and BWRs based on industry feedback to ensure usability and transferability to a variety of nuclear plant types, physical layouts, simulation capabilities, and organizational structures.

2. OVERVIEW

MASS-DEF aims to reduce conservatism in the design of physical protection systems. Traditionally, the effectiveness of a physical protection system is assessed by conducting multiple security simulations until the point at which intruders can penetrate critical areas known as target sets. This technique simplifies the functioning of the structures and systems being safeguarded, allowing the analyst to concentrate on the design of the physical protection system. Nevertheless, there are additional elements to consider both before and after the point of successful interference by attackers at a nuclear facility, such as calculating the time required for the plant to experience severe core damage, taking proactive safety measures within that period to avert such damage, or implementing initial safety steps before any disruption of a target by attackers. These considerations can be addressed by integrating them into the modeling and simulation framework alongside the physical security model. The result provides an additional margin that can be capitalized to optimize the physical protection system and reduce costs (e.g., by reducing and repositioning guard posts). The following section describes this integrated methodology in more details.

2.1 MASS-DEF Methodology

The combined safety and security measures can be modeled in a repetitive process to refine the number of guard stations. Theoretically, by easing the strict standards for physical protection failure and including preemptive safety measures, a surplus of protective margin can be achieved. This extra margin is systematically used to eliminate the least efficient guard station, using the outcomes of simulations as a guide, until the surplus is diminished to a level where the updated protection standard matches the original standard.

MASS-DEF incorporates simulations of attacks and factors in extra elements, like the timing of operator interventions and critical behaviors of safety systems, including the availability of pumps and the longevity of batteries. Dynamic modeling is adopted because the entire process of attack, response, operator actions, and plant responses is highly dynamic in that the timing and success or failures of one event affects the successes or failures of the next events. Therefore, the dynamic modeling tool called EMRALD designed to couple with other simulation tools is used in this methodology.

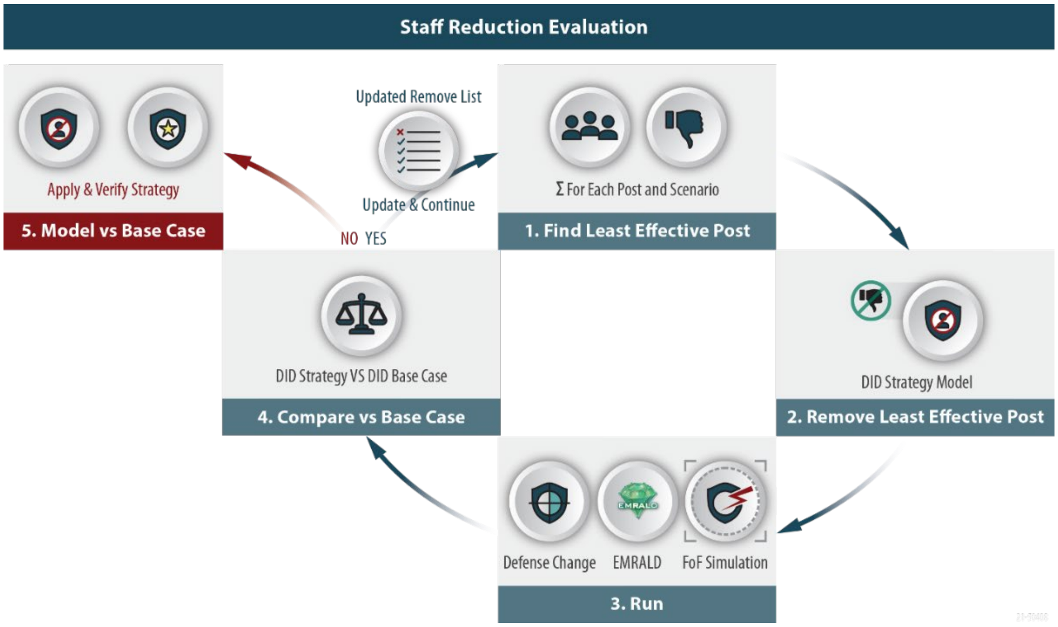


Figure 1. MASS-DEF general methodology.

Figure 1 outlines MASS-DEF primary steps to refine the number of armed responders, and these steps are detailed below:

1. Analyze the baseline security simulation results to identify the least effective guard post throughout the scenario.
2. Exclude the identified least effective guard post from the security scenarios in the altered model.
3. Run the integrated safety-security simulation considering the defense alterations and removal of post(s).
4. Evaluate the results against the initial simulation results:
    a. If the conditional core damage probability (CCDP) is equal to or better than the original model results, recommence the process from step 1.
    b. If the CCDP is inferior to the baseline model, then the security configuration from the previous iteration is kept and the process exits the reduction loop, proceeding to step 5.
5. Implement the removal list on the initial potential strategy model. Execute and confirm that the outcomes are less effective than the original base case model.

Figure 2 shows a previous study on the application of MASS-DEF to a hypothetical nuclear plant [6], where up to four guard posts were reduced without compromising the level of protection. However, note that these results may not be accurate because they do not reflect the complexities in actual NPPs. Therefore, MASS-DEF application on actual plants is studied and discussed in the next section.

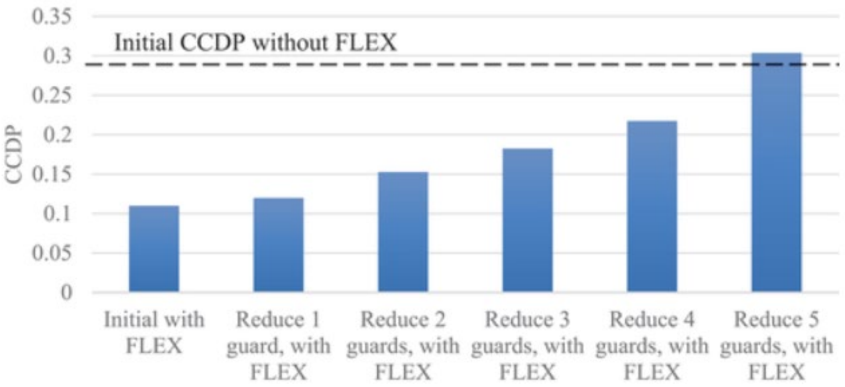


Figure 2. Guard post reduction process on a hypothetical nuclear plant [6].

2.2 Case Study

In Fiscal Years 2022 and 2023, this project was conducted with an industry collaborator using their facility and scenarios to evaluate the concepts of combining FoF simulation with modeling and simulation of alternative protection strategies and evaluating plant behavior for the outcome. The FoF simulation was done using Simajin software maintained by RhinoCorps. Transitioning from the basic testing of concepts to using real data developed this into the MASS-DEF process described in Section 2.1. Two phases in the case study were performed where the first analysis used previous scenarios evaluated for their existing protection strategy. INL, RhinoCorps, and the collaborating NPP reviewed the target set scenarios and identified several scenarios that could be prevented by using alternate cooling options such as a FLEX pump. The identified FoF simulation scenarios were exaggerated and had secondary targets added to them. This initial evaluation showed the potential for a 22% active response force reduction through operator actions of filling the steam generators on an attack detection and using a FLEX-like security pump. For detailed results, see the report *Plant-Specific Model and Data Analysis using Dynamic Security Modeling and Simulation* [7].

The initial analysis results and post reductions were presented to the facility, and they were pleased with the significant results but also identified several items of concern, such as auxiliary duties of some posts that were removed and the cost of adding an additional FLEX hookup location. Alternative options were discussed, including using the B5B connections [8] instead of an expensive plant modification for the additional FLEX location. They also wanted the analysis to include a planned set of security updates to see if these results would be as significant after their planned changes. The following sections go over the second phase of the case study.

2.2.1 Force-on-Force Simulation Model Changes

To perform a full analysis instead of just a few scenarios being analyzed as done in the first case, all the scenarios were reviewed and included in the analysis. About 50% of the scenarios could benefit from the new security strategy and were part of the post reduction, and the other scenarios are used in determining the post's importance for the reduction process. The new scenarios used for the planned physical security protection changes were used, including guard towers and adjusted post locations. Secondary targets of the FLEX connections and B5B connections were added as the easiest way to disable the use of the security pump. These were added after the primary target set items for most scenarios unless the attack path went near one of the locations then it was added as an in-route item.

2.2.2 EMRALD Model Changes

The only significant change needed for the EMRALD model was the addition of time for using the B5B connection. Since the procedure would take more time than the current FLEX connections, if the adversaries successfully hit the FLEX connection, then additional time was added to the procedure to capture that extra time.

2.2.3 Post Reduction

A MASS-DEF post reduction was performed by developing exaggerated scenarios. This turned out to be more difficult than just making the adversaries faster or decreasing the guard hit-to-kill ratio. To be accurate in showing the effectiveness of the alternative protection strategy, the exaggerated scenarios needed to make it more likely for the adversary to hit the primary targets but also be realistic after those targets are hit so the adversaries were not just easily getting both primary and secondary targets all the time. In some scenarios, this was sometimes done by just increasing the number of adversaries; in other scenarios, the hit-to-kill ratio was lowered until they reached a certain depth into the scenario. It turned out that multiple methods were needed, depending on the type of attack. This was an iterative process that was done until the adversaries successfully attacked 40–60% of the primary targets. In addition, for scenarios that also hit secondary targets most of the time, analysts needed to determine if their successes were valid or if they just hit the target because of the exaggerated conditions

Once the exaggerated scenarios were created, a list of least effective posts was created. Of these top candidates, any that were marked with special duties or were listed as high contributors for the scenarios not benefiting from the new strategies were removed as candidates. A set of posts were removed from the protection strategy and the simulations were run to determine the probability of success for the defense. This process was repeated, as described in Section 2.1, until the new strategy minus the removed posts was close to the same as the base case exaggerated scenarios.

2.2.4 Result Outcome

The exaggerated scenarios for the base case had an initial adversary success rate of 41.3% and the alternative protection strategies had an adversary success rate was 12.5%. This provided a 29% margin to be capitalized in the guard post reduction process. Several iterations of the post reduction were performed, and even without adjusting post assignments for the removed positions, almost 20% of the active guard posts could be supplanted with the alternative protection measures. This is a significant amount for the facility, and a conservative rough estimate without a cost analysis put recuperation costs within a year if they could perform the change under 10 CFR 50.54(p) [9].

3. INDUSTRY REVIEWS & FEEDBACK

The MASS-DEF process and case study were presented and/or discussed at several industry venues including PWROG meetings, LWRS stakeholder meetings, NEI events, and onsite at utilities. INL staff reviewed several facility targets sets to look at the potential use of the MASS-DEF process while also looking at other physical security change options such as RAPT categorization.

In general, industry was excited to see research into using risk-informed methods and alternative protection strategies for physical security because industry needs justify changes to security strategies and demonstrate that those changes will not negatively impact plant protection. Most expressed that the case study outcome was a significant result and a process to numerically show that equivalent protection level would be a huge step if it achieved regulatory acceptance.

The following questions and comments focused on regulatory concerns:

- Whether the NRC will accept operator actions used in the protection strategies within the current requirements, or if an alternative process be required.
- Concern that if these changes are submitted under 50.54(p), not requiring NRC review, the NPPs could get a citation during an inspection which would have a significant cost and negative impact. On the other hand, submitting these changes under 73.55(r) would have a significant upfront cost.

- Concern that even if the NRC accepts the methods, regional inspectors may have varying interpretation on regulatory compliance and disputing an inspector's decision can be difficult.
- A concern that the industry will be required to collect data from actual exercises involving operators and to track target hit times, procedure times, and then have a thermal hydraulics team analyze the outcome, which is time-consuming.
- A question on whether MASS-DEF benefit most PWR facilities.
- A question on the differences between MASS-DEF applications and benefits to BWR and PWR.
- A concern that anticipated regulatory changes may affect MASS-DEF implementation feasibility. There are several potential changes coming from the NRC with regard to physical security licensing options. The MASS-DEF process looks to be a great tool to develop risk informed decisions and to use as a justification for security program changes under the new, proposed rules. However, until these regulatory changes are finalized, there remains uncertainties on how best to proceed.

Industry's biggest desire was to use a pilot plant to demonstrate the security changes using MASS-DEF and then have the NRC review/inspect the results so they could follow the same process.

4. BWR GENERIC MODEL

One of the key feedback requests from industry was to have a generic BWR model like the generic PWR EMRALD model and determine if similar beneficial results are shown for these facilities as well. While BWRs do not have steam generators with inventory that can provide cooling while prevention methods are executed, they do have other safety systems and operator actions that could have similar outcomes. A task was added for this year to develop a generic BWR model for the MASS-DEF process with plans for a case study similar to the one describe in Section 2.2 in the next year.

A site volunteered to meet with INL to review their target set and discuss alternative prevention methods. These tasks were performed, and while no general on attack operator actions were identified, like filling steam generators for PWR's, there are many methods for alternative cooling that could be used for several scenarios. The results of this visit were used in the initial planning for the generic BWR physical security EMRALD model. The setup and many of the elements of BWR model are the same as the PWR. This model is discussed in detail in Appendix A.

5. SUMMARY

This report documents further progress on continuing work within the LWRS program's physical security pathway to develop methods and tools to support the optimization of physical security postures using modeling and simulation. Previous studies developed MASS-DEF as a dynamic computational framework that links results from commercially available FoF simulation tools, commercially available thermal-hydraulic tools, and the dynamic risk modeling tool EMRALD to model the complex nature of physical security scenarios and the time dependent nature of how CD could occur during these scenarios more accurately. Previous studies also developed and demonstrated the functional connection between the required applications using generic PWR physical security and reactor models. In a previous report, the MASS-DEF process was applied to an actual commercial NPP to verify that results obtained using generic models represented actual scenarios and to further refine the guidance to support future analysis by other utilities. As part of this research, a generic EMRALD model was developed to reduce the modeling effort that a utility would need to perform to replicate this type of analysis. The lessons learned from this study were used to work with industry to create a guidance document outlining a detailed process to perform this type of analysis. Additionally, a generic BWR physical security model was created to facilitate the use of the MASS-DEF process at commercial BWR reactor sites. Both generic physical security models are outlined in Appendix A.

6. REFERENCES

1. PG&E. 2018. “PG&E Company 2018 Nuclear Decommissioning Cost Triennial Proceeding Prepared Testimony – Volume 1.” 18-12 (U 39 E), Pacific Gas and Electric Company. https://analysis.nuclearenergyinsider.com/pge-seeks-decommissioning-head-start-cost-estimates-rise.
2. U.S. NRC. 2020. “Emergency Preparedness in Response to Terrorism.” About Emergency Preparedness, U.S. Nuclear Regulatory Commission. Last modified Nov. 13, 2020. https://www.nrc.gov/about-nrc/emerg-preparedness/about-emerg-preparedness/response-terrorism.html#one.
3. U.S. NRC. 2021. “PART 73—Physical Protection of Plants and Materials.” (NRC, 10 CFR), U.S. Nuclear Regulatory Commission. Last modified Mar. 24, 2021. https://www.nrc.gov/reading-rm/doc-collections/cfr/part073.
4. NEI. 2016. “Diverse and Flexible Coping Strategies (FLEX) Implementation Guide.” NEI 12-06, Rev. 4, Nuclear Energy Institute. https://www.nrc.gov/docs/ML1635/ML16354B421.pdf.
5. R. Christian, S. R. Prescott, V. Yadav, S. St. Germain, C. P. Chwasz. 2022. “Evaluation of Physical Security Risk for Potential Implementation of FLEX using Dynamic Simulation Methods.” INL/RPT 22 70315, Rev. 0, Idaho National Laboratory, Idaho Falls, ID. https://inldigitallibrary.inl.gov/sites/sti/sti/Sort_64424.pdf.
6. R. Christian, V. Yadav, S. R. Prescott, and S. W. St. Germain. 2022. “A Dynamic Risk Framework for the Physical Security of Nuclear Power Plants.” Nuclear Science and Engineering 197, no. 1 (2022): pp. S24 S44. https://doi.org/10.1080/00295639.2022.2112899.
7. S. R. Prescott, R. Christian, V. Yadav, S. W. St. Germain, C. P. Chwasz. 2022. “Plant-Specific Model and Data Analysis using Dynamic Security Modeling and Simulation.” INL/RPT-23-73490, Rev. 1, Idaho National Laboratory, Idaho Falls, ID. https://inldigitallibrary.inl.gov/sites/sti/sti/Sort_78183.pdf
8. U.S. NRC. 2011. “NRC Bulletin 2011-01: Mitigating Strategies.” U.S. Nuclear Regulatory Commission, May 11, 2021. https://www.nrc.gov/docs/ML1112/ML111250360.pdf
9. U.S. NRC. 2023. “§ 50.54 Conditions of licenses.” U.S. Nuclear Regulatory Commission, Last updated Dec. 18, 2023. https://www.nrc.gov/reading-rm/doc-collections/cfr/part050/part050-0054.html

Page intentionally left blank

Appendix A

Generic EMRALD Physical Security Model

The goal of the EMRALD model in the MASS-DEF process is to include operator actions and plant behavior into the security attack scenarios to provide an accurate statistical outcome. It combines with attack scenario or FoF simulation software tools and thermal hydraulic analysis, allowing the user to add missing pieces of operator action and plant design features while keeping track of the timing of events to generate a dynamic analysis. Both PWR and BWR generic models have been created, and while there are differences between BWR and PWR operation and safety systems, the main process for the models is the same. When there are differences in the model, these will be called out and explained.

Note there is no sensitive information in the generic EMRALD physical security models. No plant-specific information is in the models, and the plant behavior modeled is public information. Modifying a generic model to match a specific facility and adding any security procedures would make the model sensitive. The EMRALD UI is web-based but runs locally on the user's computer; there is no data sent online. It is recommended that anyone using a generic model make all non-sensitive information changes, save the model, and then use the offline EMRALD version to make any sensitive model changes on approved machines and locations. The solve engine to run the model should also be downloaded onto approved machines and run in a proper location.

A-1. Attack Scenario Changes

Typical FoF simulation attack scenarios consist of an attack force, the route being taken, barriers, guards, and targets the adversary is trying to take out. Traditionally, if the adversaries take out the targets, then the scenario is over and the adversaries win; if the adversaries are killed before the targets are hit, then the protection force wins. To perform the MASS-DEF analysis, another layer needs to be added. All the scenarios need to have additional targets added to them so the adversary can make sure that the site cannot use the additional prevention strategies. We will call the original targets primary targets and the additional targets secondary targets. Scenarios may have multiple prevention options that could be used to stop the adversary from causing core damage, and secondary targets for each applicable prevention option must be added to the scenario. The prevention strategies can also have multiple ways to be disabled, and the easiest secondary target(s) for the prevention options should be used for that scenario.

Once the adversaries hit the primary targets or on route to hit the primary targets, the adversaries will attempt to hit the secondary targets. The EMRALD model uses both primary and secondary target hit times. The goal is to neutralize the adversaries before the secondary targets are hit in scenarios where all the primary targets were previously hit. Only scenarios where all the primary targets are hit and not all the secondary targets are hit will contribute to a post reduction using the MASS-DEF process; see [6] for more details. Features in the generic EMRALD models focus on those scenarios, and the simulations quickly terminate if not all primary targets are hit or all primary and secondary targets are hit.

A-2. Generic Model Pieces and UI Locations

This explanation of the generic EMRALD physical security models assumes that the reader has a basic understanding of the EMRALD modeling tool, the EMRALD UI, and capabilities. Please refer to the EMRALD website for training and general use. The following sections go over the different modeling pieces and organization.

A-2.1 Diagrams

A model is broken up into small pieces called diagrams. Diagrams can represent individual components, system, procedures, etc. They capture the state things are in and what events and actions occur. For this model, there are four categories for the diagrams: Components, OpActions, Plant, and Other (see Figure A-1).

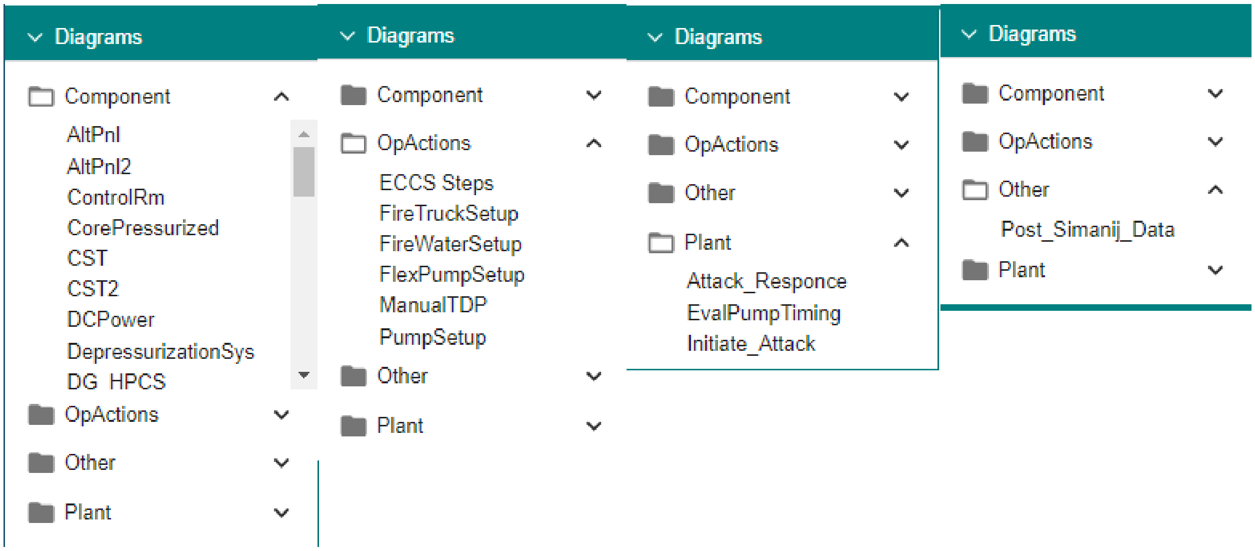


Figure A-1. Expansion of the different category diagrams used in the model.

A-2.1.1 “Component” Diagrams

These diagrams represent different components for the physical security model. Most are items that the adversary could target as part of a target set or are needed for modeling plant behavior. These diagrams are single state diagrams meaning they can only be in one state at a time and each state has a Boolean value associated with it; therefore, at any time in the simulation, it can be evaluated. Single state diagrams can be used as end nodes in an EMRALD Logic Tree; see Section A-2.2. The default value for the state can easily be seen in the diagram: green for true, red for false, and gray for ignore.

The residual heat removal pump shown in Figure A-2 is one of the items that could be an adversary target. The component starts in the standby state, and when the low pressure coolant injection system is started, it moves to the running state, and events can move it from the running state to failed or back to standby. Typically, all the components that can fail from an adversary have a “[name]_Timer” event. This event is the time the adversary has taken out the component for that simulation run. If they have not hit that component, then the number is very large, and the event will not trigger. Many components also have a “FailRt_[Name]” event. This event is the random probability that the component will fail when the component is starting; it also has a percentage change that it can fail to start. For most components, these values are set to “0” as physical security modeling does not have to consider random failure of safety systems. However, we do include the random failures for equipment directly used as an alternative prevention strategy.

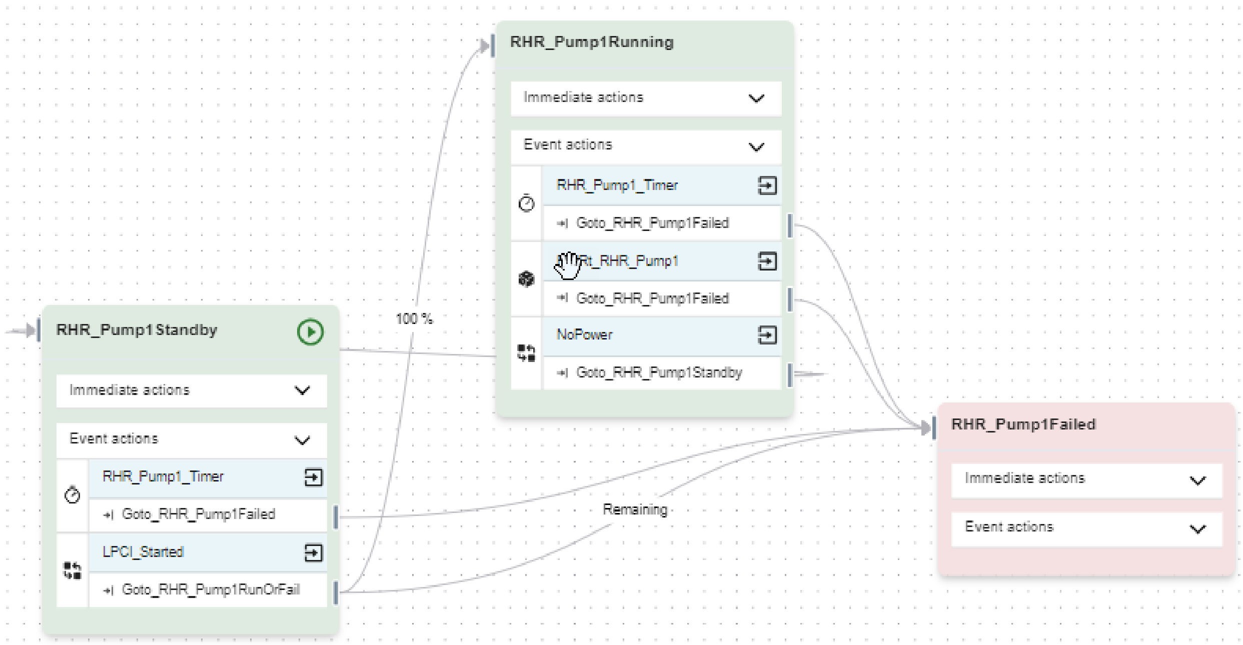


Figure A-2. Diagram for the RHR pump showing the states events and actions.

A-2.1.2 “OpActions” Diagrams

These diagrams are operator action diagrams, modeling the different procedures for the tasks. Most of these diagrams are the tasks used for preventing core damage after an attack; further details can be found in Section A-3.3.

A-2.1.3 “Plant” Diagrams

These are the main diagrams modeling attack scenario progression, mitigation, and determine outcome. “Initiate_Attack” is the starting point for the simulation and is described in Section A-3.2. “Attack_Response” handles events after the attack along with prevention strategies and is described in Section A-3.3. “EvalPumpTiming” determines if there was core damage or not, depending on timing of all the events, and is detailed in Section A-3.4.

A-2.1.3.1 “Other” Diagrams

This category only has one diagram, which is just the loading of the FoF simulation data as described in A-3.1.

A-2.2 Logic Trees

All the logic trees are listed in the left side bar as shown in Figure A-3.

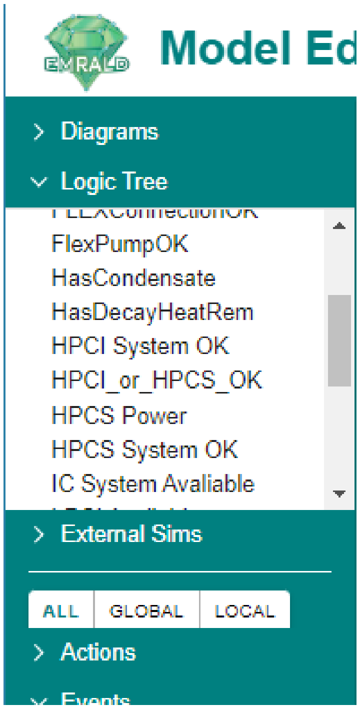


Figure A-3. Logic tree list showing several of the logic trees used in the model.

Logic trees are used to evaluate systems or combinations of different component or single state evaluation diagrams to determine if they are operable. This is done by using Boolean logic similar to fault trees in classical PRA. However, in this case, it is evaluating the components' state. Each time a state changes, all the logic trees that depend on that state are updated with a new top value. For example, the high-pressure coolant injection system (HPCS), as shown in Figure A-4, requires the turbine-driven pump and either offsite power or the HPCS diesel generator to be operable.

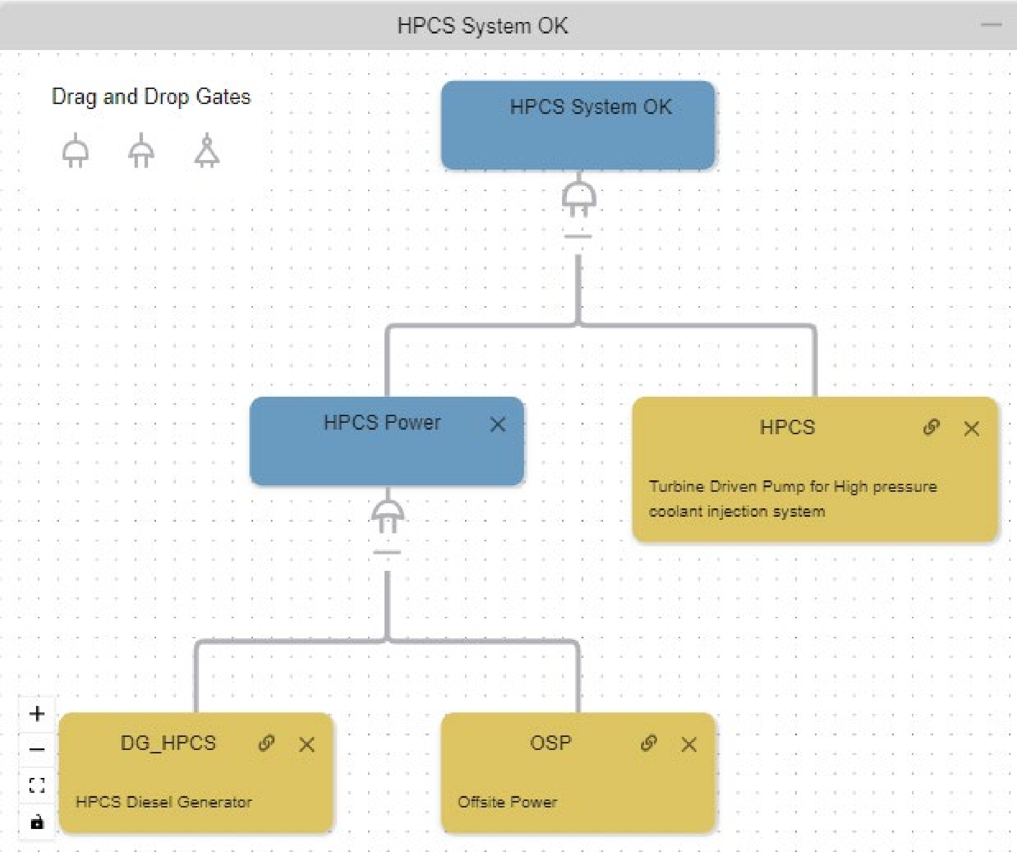
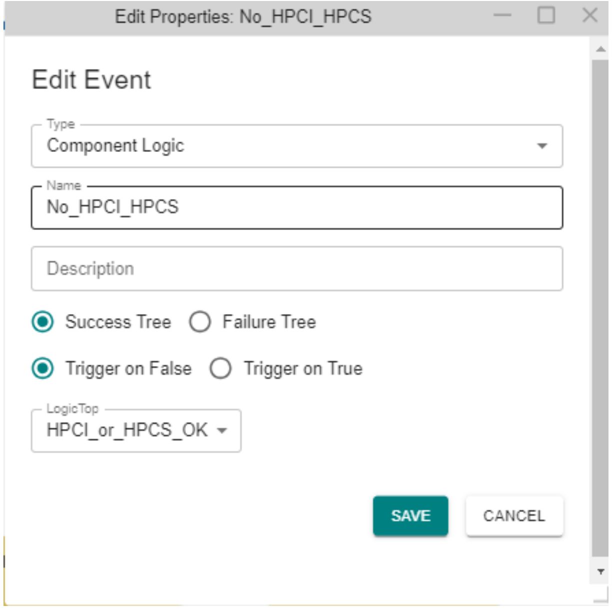


Figure A-4. Example of a logic tree.

Logic trees are used by component logic events. For example, in Figure A-5, the “No_HPCI_HCS” event is triggered when the logic tree “HPCI_or_HPCS_OK” is false. The same logic tree can be used to evaluate if a system is operable or failed by selecting the corresponding option “Trigger on False” or “Trigger on True.”



Edit Event

Type: Component Logic

Name: No_HPCI_HPCS

Description

● Success Tree ○ Failure Tree

● Trigger on False ○ Trigger on True

LogicTop: HPCI_or_HPCS_OK

Figure A-5. Component logic events evaluate a logic tree and are triggered according to the selected properties.

A-2.3 Actions, Events, States

These lists show all the actions, events, and states for all the different diagrams. Items can be dragged and dropped from these lists to be used in different locations of the model.

A-2.4 Variables

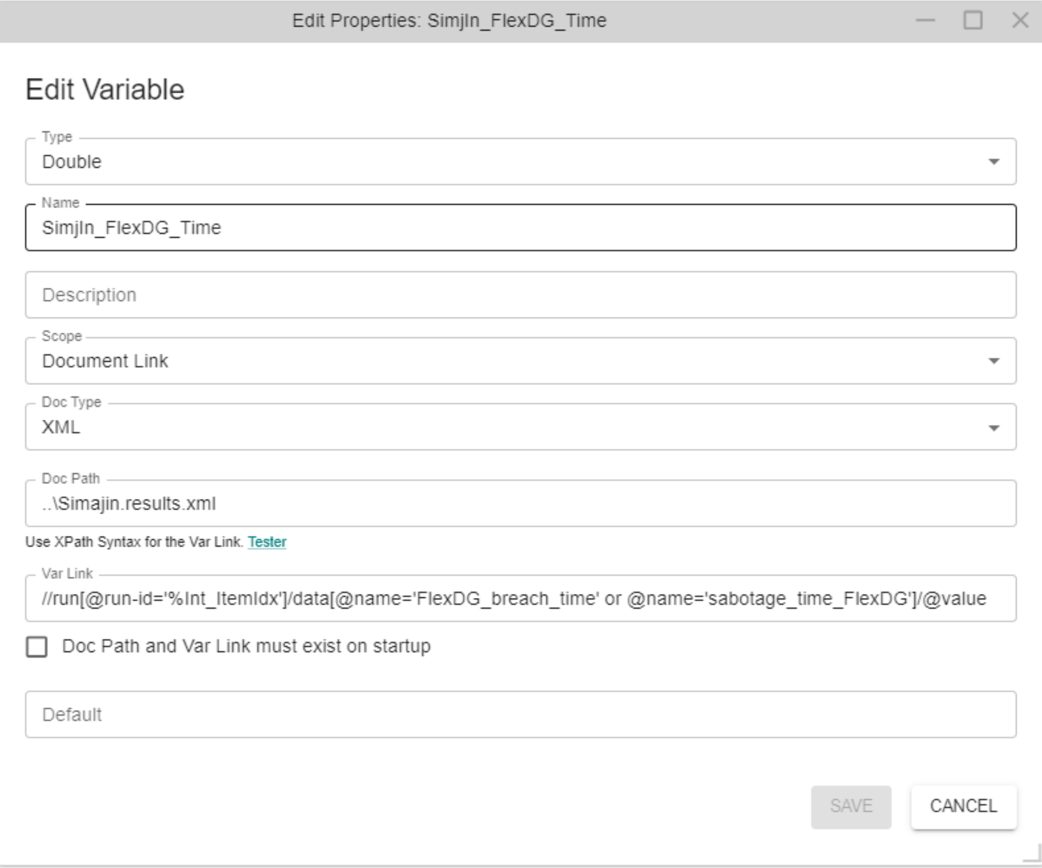
Variables are used for many different purposes in the physical security mode.

A-2.4.1 Component Hit Times

There is a variable tied to each component that can be hit by the adversary. This variable is the time the component is hit by the adversary. By default, it is a very large number so that it will not occur unless changed when loading the FoF simulation data. These variables are named in this format "[component name]_HitTime."

A-2.4.2 FoF Event Times

These variables are document link variables and tie directly to the FoF result data. The example shown in Figure A-6 is for the Simajin simulation results and tie to the XML output. When the variable is used in the EMRALD simulation, it goes to the XML file and pulls that value. See Section A-3.1 for the loading of data from the FoF results. The field in the variable properties indicates a relative path to the result file and the XPath expression for the value to be retrieved. The modeler must make sure the names for the variables in the Simajin results match the variable in EMRALD representing the same event time.


Edit Properties: SimjIn_FlexDG_Time

Edit Variable

Type: Double

Name: SimjIn_FlexDG_Time

Description

Scope: Document Link

Doc Type: XML

Doc Path: ..\Simajin.results.xml

Use XPath Syntax for the Var Link. Tester

Var Link: //run[@run-id='%Int_ItemIdx']/data[@name='FlexDG_breach_time' or @name='sabotage_time_FlexDG']/@value

☐ Doc Path and Var Link must exist on startup

Default

Figure A-6. A document link variable linking to FoF XML results.

A-3. General EMRALD Physical Security Model Structure

The EMRALD model can be viewed as the following steps for each attack simulation run:

1. Load FoF simulation timing data. This includes the attack detection time, when adversaries hit targets, if objectives were met, when the attack is considered over, etc.
2. Initial attack preparation strategy. If there are any operator procedures or events that trigger a plant response, this is where these are handled, such as diesel generators starting up on loss of offsite power.
3. Attack outcome. If the primary attack targets are not successful, then no further evaluation is needed. If they were successful, then how long after the engagement is the security team ready for after attack operator action prevention tasks, such as doing a security sweep and escorting an operator?
4. After attack response. Evaluate the systems after an attack and determine what operator actions/procedures can be used to prevent core damage or radioactive release.
5. Evaluate outcomes. Use the initial operator action times, component hit times, and preventive operator action times to determine if there was core damage or radioactive release. Run the thermal hydraulics analysis if not within known boundary conditions.

A-3.1 Load Force-on-Force Simulation Timing Data

The EMRALD model requires data from a FoF simulation tool .For this example, the RhinoCorp’s Simajin software is being used; however, other tools can be used by loading the timing results from a different tools output data. The “Load_Simanij_Data” diagram has only one state with a long list of immediate actions, as shown in Figure A-7. This is not the starting point of the simulation but is called from the “Initiate_Attack” diagram discussed in Section A-3.2. When the state is entered, the actions, outlined in red on the right-hand side of Figure A-7, assign the time of each event that could impact the model from the FoF simulation results to a variable inside of EMRALD.

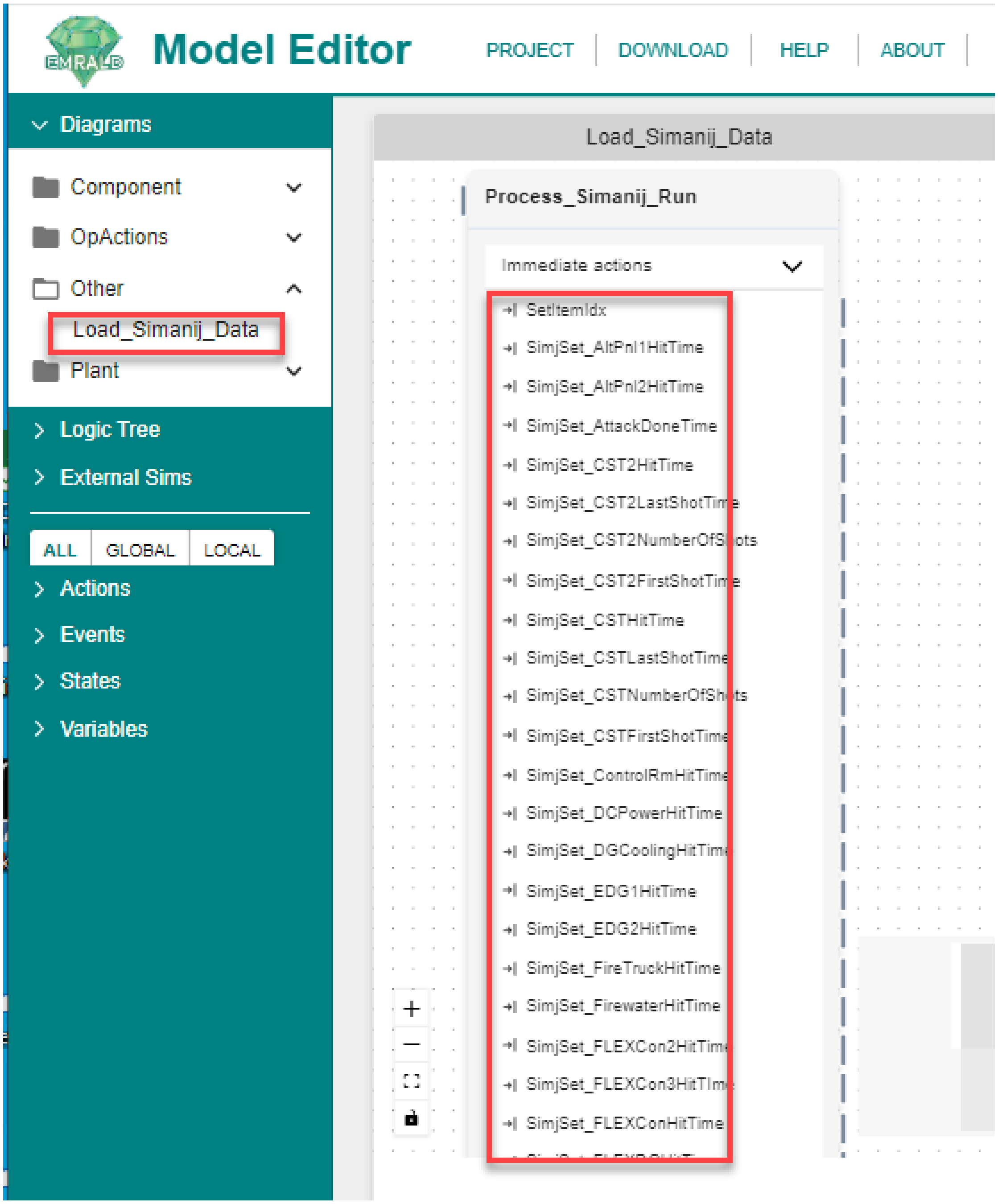
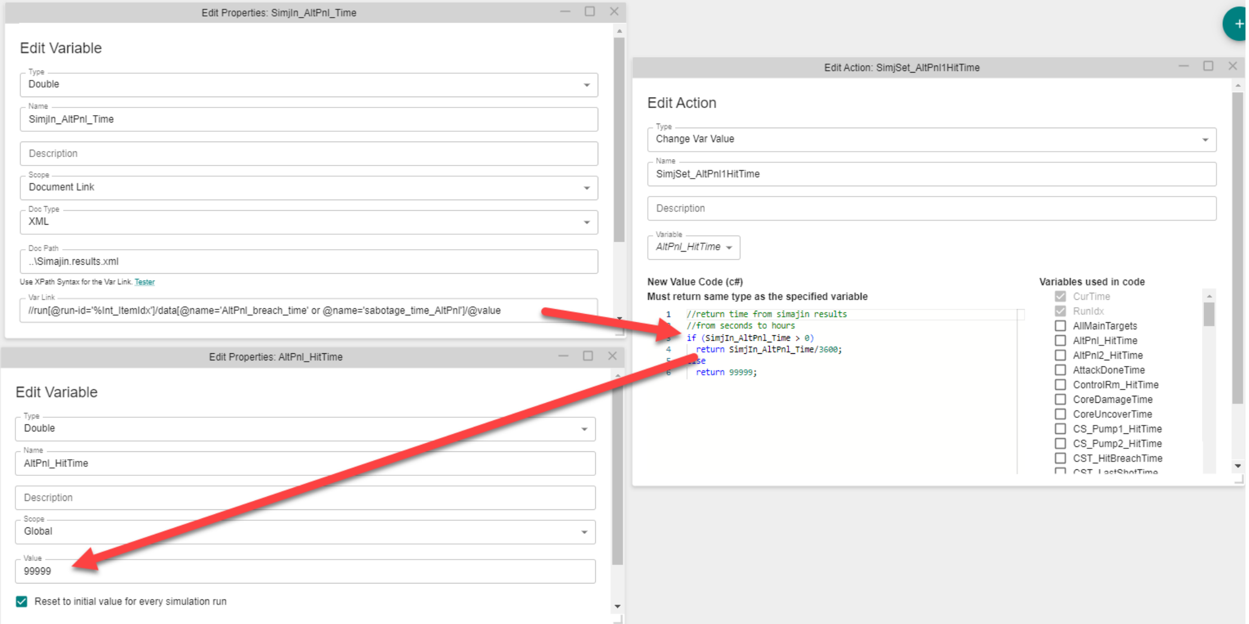


Figure A-7. Loading of FoF simulation data in the diagram Load_Simanij_Data

Refer to Figure A-8 for the following info. For each event from the FoF simulation that needs to be used in the EMRALD model, two variables are created in EMRALD. The first is the one used when doing any calculation in EMRALD, and the second is the variable that links directly to the FoF data. Two variables are used to separate the generic model from the FoF tool used and the format of its data. We always want the time variable used by EMRALD to have the same format, in this case, hours. The variable that reads the data from the FoF data is a document link variable where an XPath expression indicates what value to pull out of the FoF results file. The action that loads the data reads the document link variable and converts it into consistent format, in this example, from seconds to hours. By default, each event time variable in the model is set to a very high value; this means that it will have no effect on the simulation. The default value in the Simajin results is 0 or is not included in the results file if the event did not occur, and if the event was achieved, then the number of seconds from the start of the simulation to the end is shown. If 0 is used in the EMRALD evaluation, then the event would happen as soon as the simulation starts; so the 0 must be converted to a high time value that will not occur during the simulation run.



Edit Properties: SimjIn_AltPnl_Time

Edit Variable

Type: Double

Name: SimjIn_AltPnl_Time

Description

Scope: Document Link

Doc Type: XML

Doc Path: ..\Simajin.results.xml

Use XPath Syntax for the Var Link. Tester

Var Link: //run[@run-id='%Int_ItemIdx']/data[@name='AltPnl_breach_time' or @name='sabotage_time_AltPnl']/@value

Edit Properties: AltPnl_HitTime

Edit Variable

Type: Double

Name: AltPnl_HitTime

Description

Scope: Global

Value: 99999

☑ Reset to initial value for every simulation run

Edit Action: SimjSet_AltPnl1HitTime

Edit Action

Type: Change Var Value

Name: SimjSet_AltPnl1HitTime

Description

Variable: *AltPnl_HitTime*

New Value Code (c#)
Must return same type as the specified variable

```
//return time from simajin results
//from seconds to hours
if (SimjIn_AltPnl_Time > 0)
  return SimjIn_AltPnl_Time/3600;
else
  return 99999;
```

Variables used in code

☑ CurTime
☑ RunIdx
☐ AllMainTargets
☐ AltPnl_HitTime
☐ AltPnl2_HitTime
☐ AttackDoneTime
☐ ControlRm_HitTime
☐ CoreDamageTime
☐ CoreUncoverTime
☐ CS_Pump1_HitTime
☐ CS_Pump2_HitTime
☐ CST_HitBreachTime
☐ CST_LastShotTime

Figure A-8. The variable “Simjln_AltPnl_Time” (top left) links to the FoF simulation data using an XPath expression. When the action “SimjSet_AltPnlHitTime” occurs (right), it reads the data from “SimjIn_AltPnl_Time” and assigns “AltPnl_HitTime” (bottom left) with the correct value for the simulation run.

A-3.2 Initial Attack Preparation Strategy

The main starting point for the attack scenarios is the “Initiate_Attack” as shown in Figure A-7 through Figure A-9. The play button in the upper right corner of the “AttackSetup” state indicates that this is a starting state of simulation. This state immediately starts the loading of the FoF simulation results through the immediate action “Goto_Process_S_Runs.” From this point, all the events under the “AttackSetup” state, as shown in Figure A-10, captures the modeling aspects described in the subsequent subsections. Actions do not cause the exit of this state; so all the events being monitored can be triggered, not just the first one as long as they occur before other events stop the simulation.

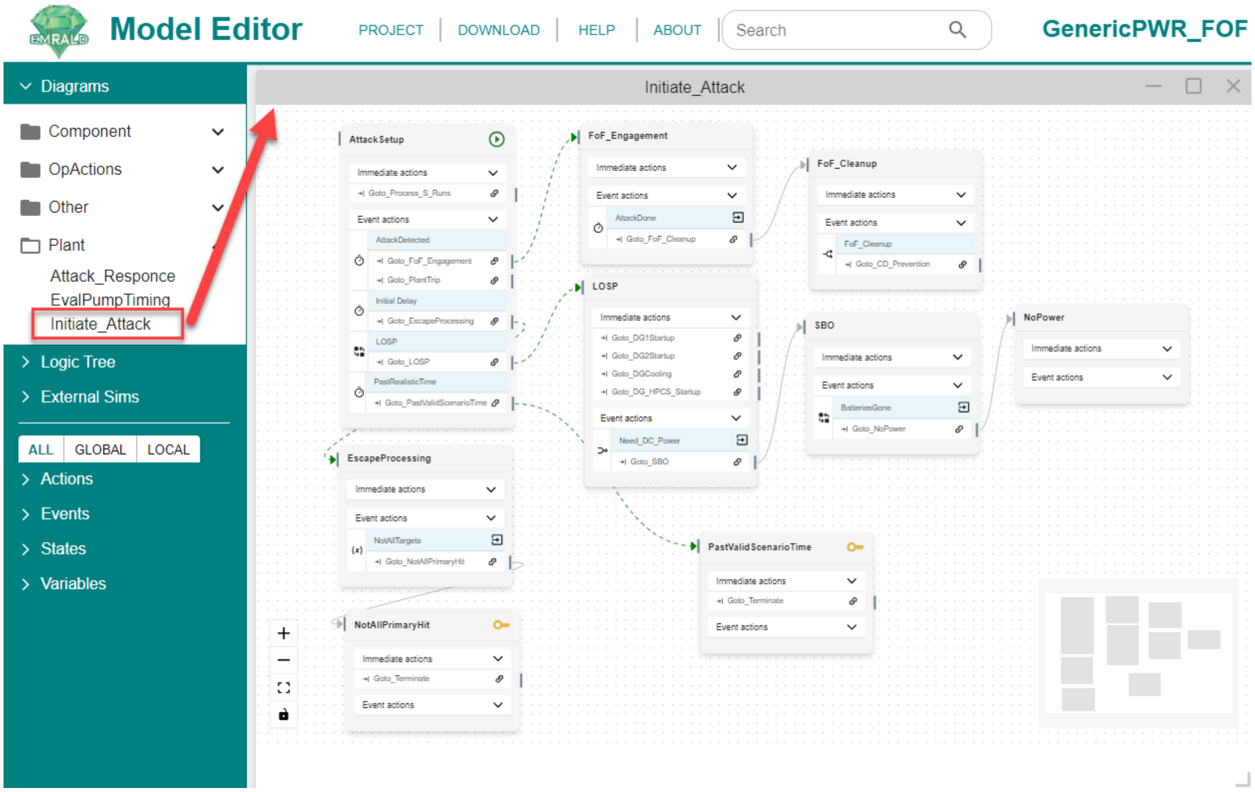


Figure A-9. The “Initiate_Attack” diagram shown here is the starting point for attack scenario evaluation.

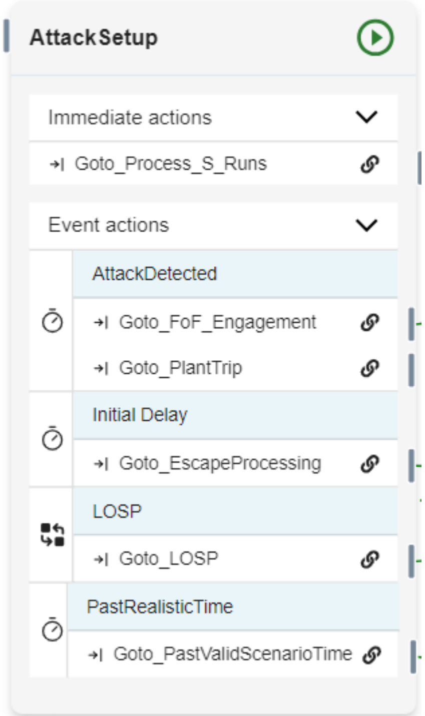


Figure A-10. The “AttackSetup” starting state starts processing the FoF data and then monitors for the listed events.

A-3.2.1 Attack Stages

As shown in Figure A-11, the first event “AttackDetected” is triggered when the assigned time from the FoF data is reached. When this event is triggered, two actions are taken, the safety systems are initiated from a plant trip action (should be removed or modified to match specific plant procedures), and the “FoF_Engagement” state is entered. Once the end of the engagement time is reached (loaded from the FoF data), then the “AttackDone” event is triggered and moved to the state FoF_Cleanup. Note this does not mean the event has reached an all clear; instead, it is the point when engagement has ended. The “FoF_Cleanup” event is a distribution sampling how long the facility will likely take before they can sweep the area needed to safely escort an operator or send a security person after attack tasks to prevent core damage if needed. This is handled by the “Attack_Response” diagram, which is started by the “Goto_CD_Prevention” action once the cleanup is done; see Section A-3.3.

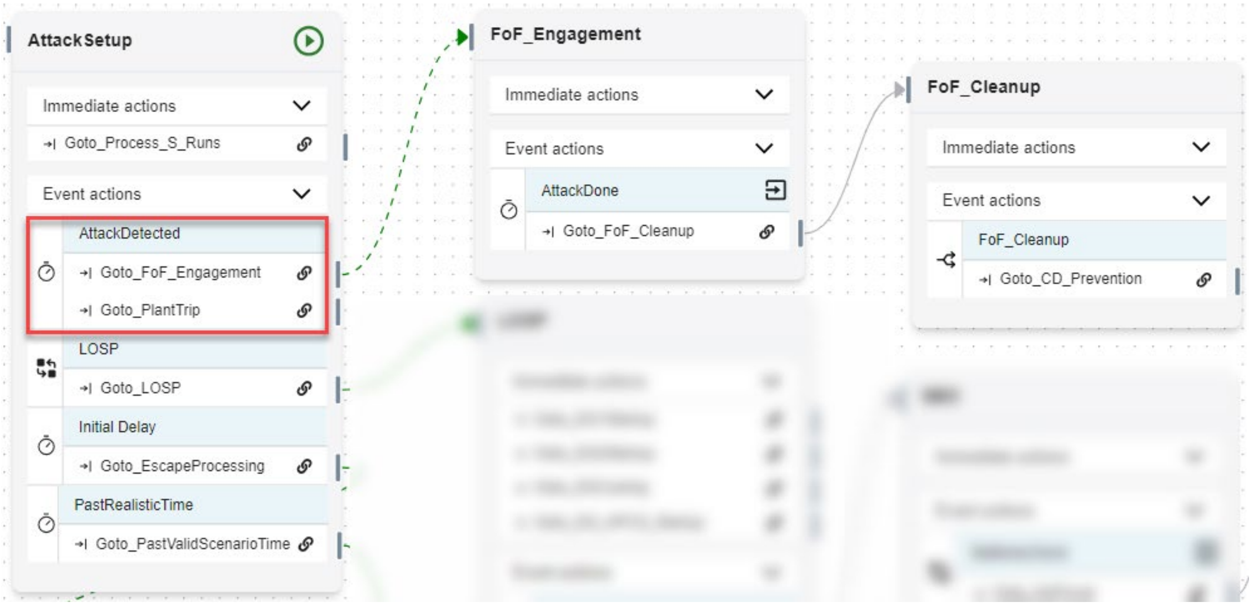


Figure A-11. Attack progression states.

A-3.2.2 Plant Condition

If or when the adversary cuts the offsite power, the “LOSP” event, as shown in Figure A-12, is triggered. Then, the “LOSP” state is entered where it starts backup power systems, as shown in Figure A-13, and the state of the plant power is monitored (the model will need to be modified for the correct number of diesel generators or other AC backup power sources to match an actual facility). Once all AC power systems are gone, then the logic tree “ACPowerOK”, shown in Figure A-14, evaluates as false and triggers the event “Need_DC_Power”. This moves the simulation into the station blackout (SBO) state where an event monitors when the batteries are depleted. Once the batteries are depleted, the “NoPower” state is entered. Other diagrams use this state to trigger events related to no power. (The “DCPower” diagram determines how long the DC power lasts and will need to be adjusted to match the facility)

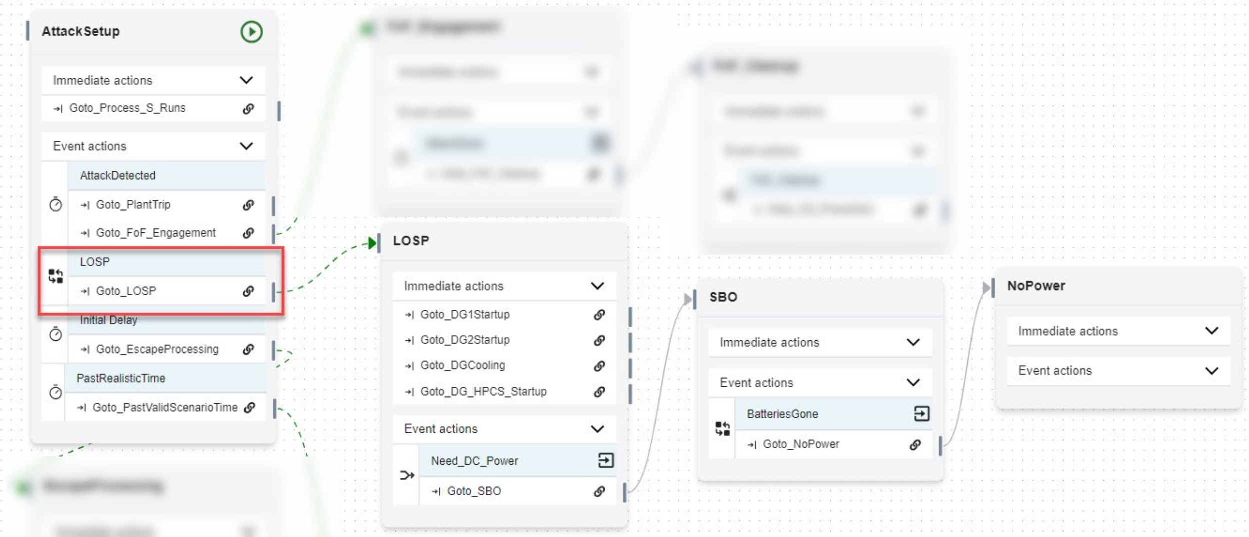


Figure A-12. Loss of offsite power event triggers an evaluation of the plant condition.

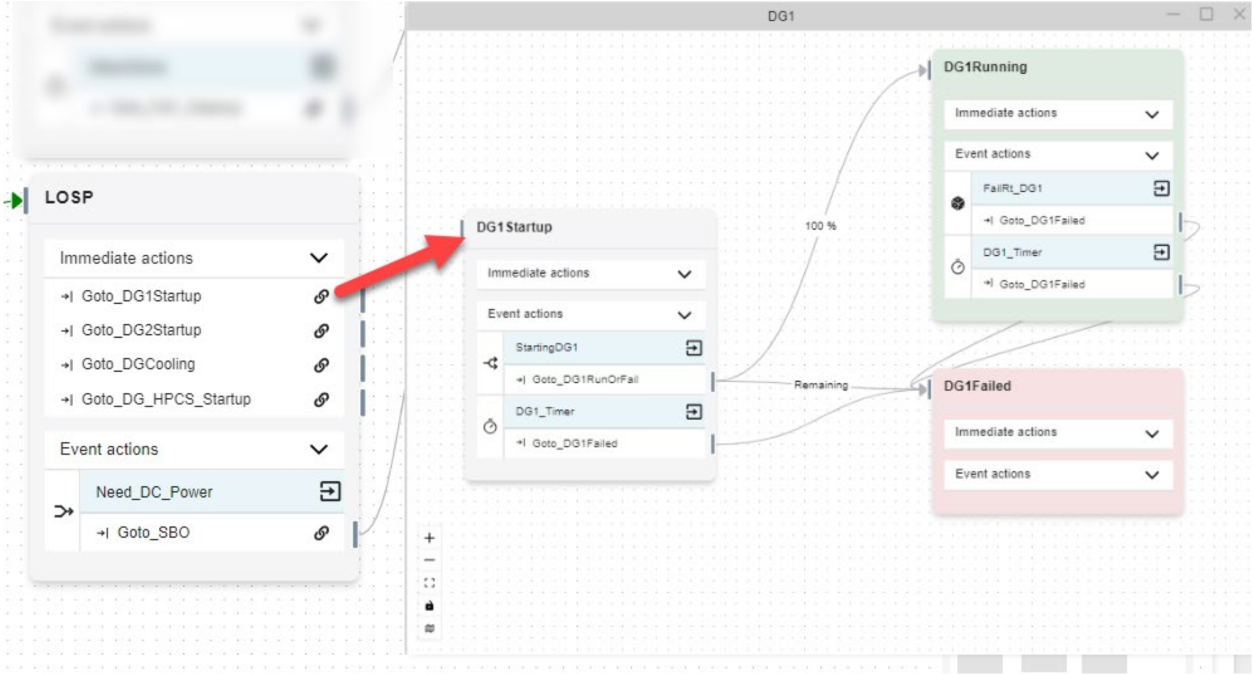
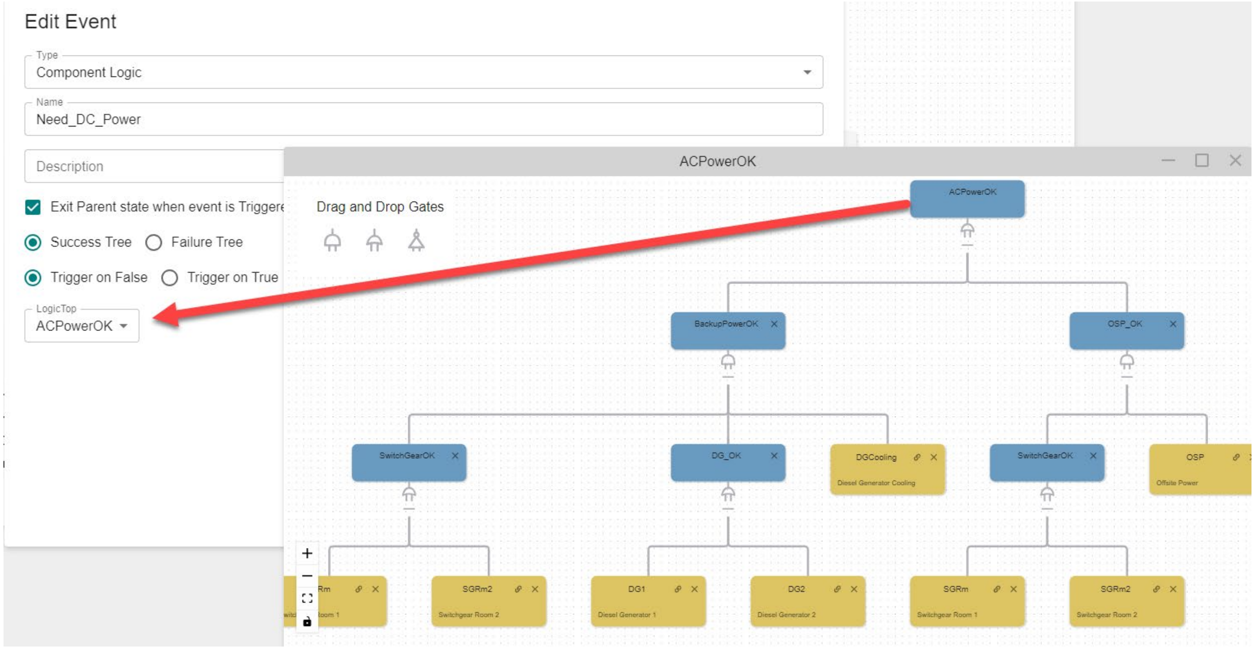


Figure A-13. The entering of LOSP state triggers the starting of several power safety systems including the diesel generator "DG1."



Edit Event

Type: Component Logic

Name: Need_DC_Power

Description

☑ Exit Parent state when event is Triggere

● Success Tree ○ Failure Tree

● Trigger on False ○ Trigger on True

LogicTop: ACPowerOK

Figure A-14. The event "Need_DC_Power" is triggered when the "ACPowerOK" logic tree evaluates as false.

A-3.2.3 Optimizing Scenario Runs

The events “Initial Delay” and “PastRealisticTime” are used to optimize the scenario runs and catch any modeling errors. The “InitialDelay” triggers an evaluation to see if the adversaries hit all the primary targets of the scenario’s target set. If not, then it moves to “NotAllPrimaryHit” and ends that simulation run. There is no need to evaluate further if they were not all hit; then, it was a failed attack, and no other protective measures are needed. “PastRealisticTime” is a timer set for 24 hours. If the simulation runs that long, then the simulation moves to “PastValidScenarioTime,” and that simulation run ends. If there are any results that end in “PastValidScenarioTime,” then these cases need to be debugged and determined what caused it to get to this state.

A-3.2.4 On Attack Operator Actions Or Defense Strategies

To execute any defense strategies on detection of an attack, other events can be added to the “Attack Setup” and linked to additional diagrams, such as operator actions of filling a steam generator for a PWR (included in the PWR generic model) or sending an operator to a secondary location. The 1.0 version of the generic BWR does not have any initial operator actions; refer to Section A-5.2 for how to add them.

A-3.3 After Attack Response

The diagram “Attack Response,” shown in Figure A-15, runs through after attack core damage prevention options. The generic models contain several options that may or may not be applicable to the facility. These can easily be disabled, or new ones added; see Section A-5.2. The simulation enters the “CD_Prevention” state when “FoF_Cleanup” is done; see Section A-3.2.1. The “CoolingPastRAPT” event occurs if 8 hours pass and at least one safety cooling system is operating past the plants RAPT time. If the “CoolingPastRAPT” event occurs before “No_ECCS_Cooling” occurs, then the plant enters the “Safe_Shutdown” key state, and the simulation run ends as a safe criteria has been met. However, if this state is hit, then it is likely that the scenario could be moved to the RAPT categorization and does not need to be evaluated through this process. An example of this would be a scenario that floods a room to take out targets, and the flooding target time is over the RAPT time. Review any scenarios that end in the “Safe_Shutdown” key state. If “No_ECCS_Cooling” occurs before the RAPT time, then additional core damage prevention options would be implemented to prevent core damage, and an analysis of the timing will be needed to determine if core damage occurs before the RAPT time.

The following subsections go over possible prevention options included in the generic models and the order in which they are evaluated. The order should also be adjusted according to plant preferences. Each prevention option uses a logic tree to evaluate if the system is still available for use; if not, an action moves it to the next option. If the system is available, then an action starts the diagram that models the procedures for that prevention option. Note, all the diagrams modeling the prevention procedures described in the following sections can be modified to include more operator action steps or tailored to the site.

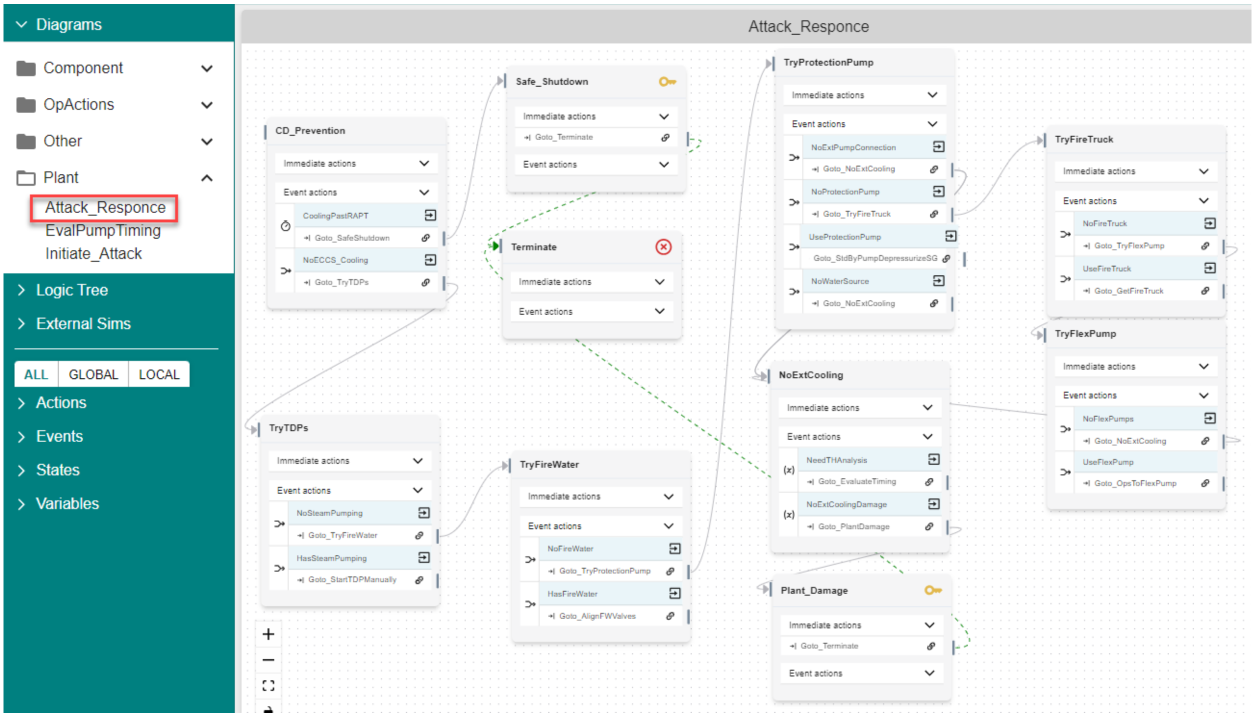
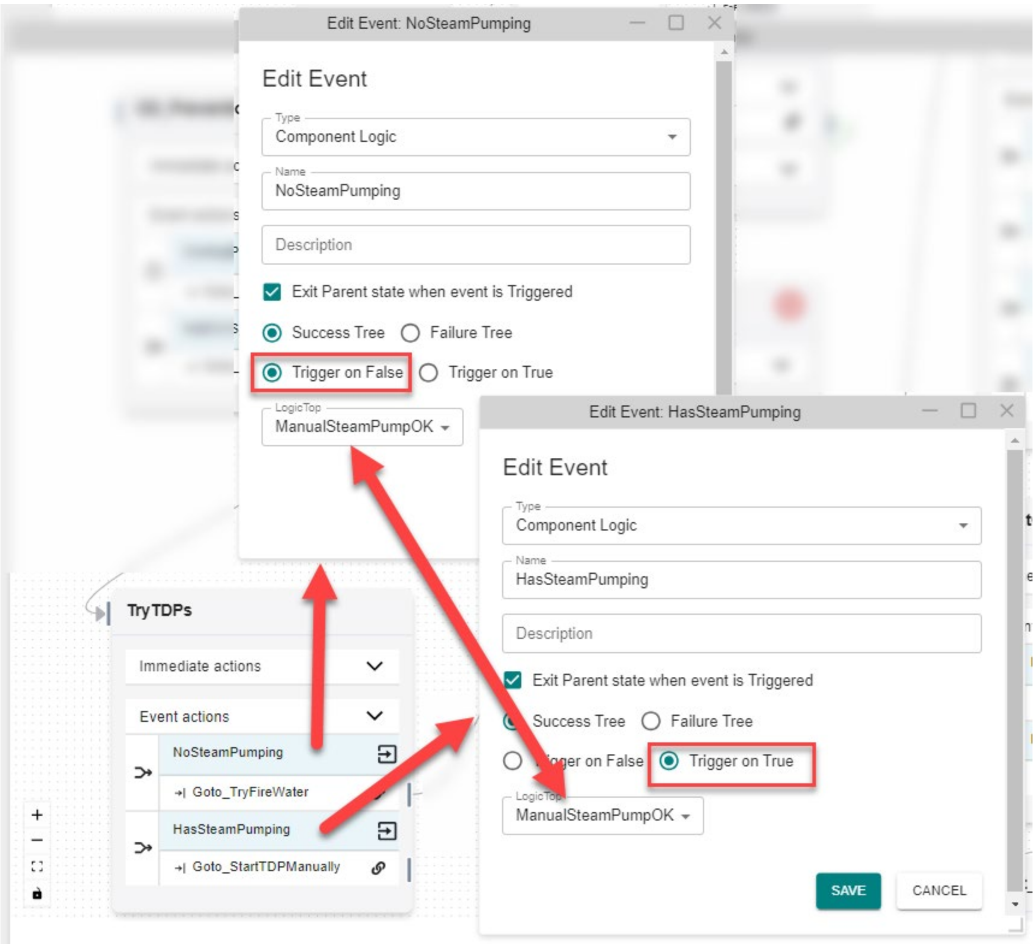


Figure A-15. The "Attack_Response" diagram that handles core damage prevention options after an attack.

A-3.3.1 Manual Turbine-Driven Pump

The first option is manually running the turbine-driven pump. The events in “TryTDPs” determine if the pump is still available. As shown in Figure A-16, this is done using the “NoSteamPumping” event and the “HasSteamPumping” event. They both evaluate the same logic tree top, but one is triggered when the top is false and the other when the tree is true, indicated by the radio button outlined in the red box in Figure A-16. One and only one of these events will occur as soon as the state is entered. If the steam turbine cannot be run manually, then it moves to “TryFireWater” described in the next section. If it can be run manually, then the action “Goto_StartTDPManually” starts up the “ManualTDP” diagram shown in Figure A-17. The “ManualSteamPumpOK” logic tree, shown in Figure A-18, shows the components or pieces used to determine if the system is available. In this case for the BWR model, we need the HPCS intact, either the turbine driven pump for the reactor core isolation cooling (RCIC) system or the turbine driven pump for the HPCI system, and the tools to manually run the system. If the plant does not want to include the prevention option of manually running a turbine-driven pump, an easy way to do so is to make sure that “ManualTDP_Tools” evaluates as a false; see Section A-5.3.


Edit Event: NoSteamPumping

Edit Event

Type: Component Logic

Name: NoSteamPumping

Description

☑ Exit Parent state when event is Triggered

● Success Tree ○ Failure Tree

● Trigger on False ○ Trigger on True

LogicTop: ManualSteamPumpOK

Edit Event: HasSteamPumping

Edit Event

Type: Component Logic

Name: HasSteamPumping

Description

☑ Exit Parent state when event is Triggered

● Success Tree ○ Failure Tree

○ Trigger on False ● Trigger on True

LogicTop: ManualSteamPumpOK

Figure A-16. This figure shows how two events use the same logic tree to determine if the prevention action can be taken—one triggers on false and the other on true.

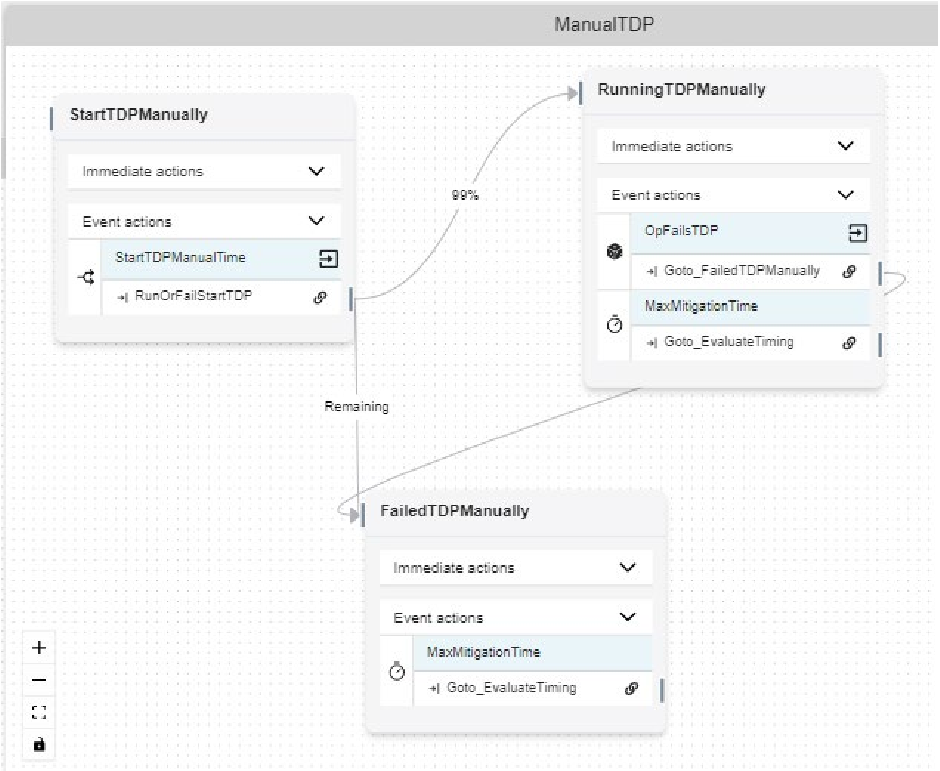


Figure A-17. ManualTDP diagram.

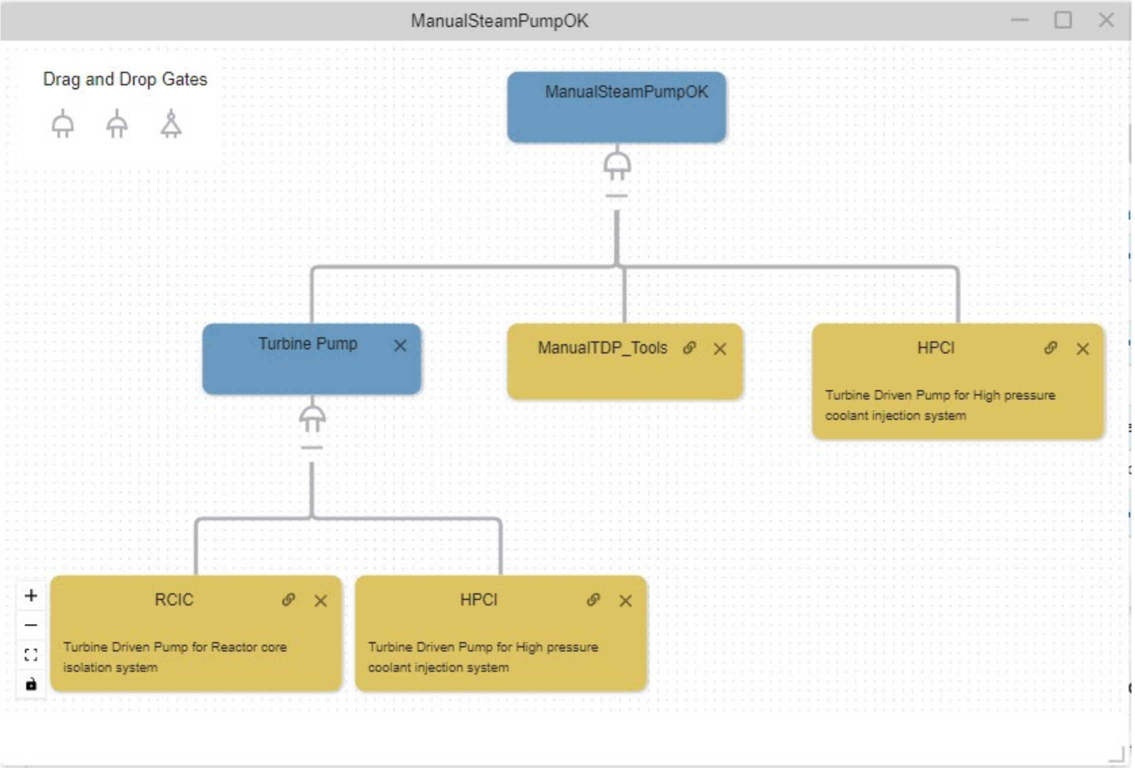

ManualSteamPumpOK
Drag and Drop Gates
ManualSteamPumpOK
Turbine Pump
ManualTDP_Tools
HPCI
Turbine Driven Pump for High pressure coolant injection system
RCIC
Turbine Driven Pump for Reactor core isolation system
HPCI
Turbine Driven Pump for High pressure coolant injection system

Figure A-18. Logic tree evaluating if the manual operation of the turbine-driven pump is available.

Manually running the TDP starts in “StartTDPManually” where a distribution event samples a lognormal distribution for the time it takes the operator to get the pump going. This data needs to come from the plant according to trial runs. There is a probability that the operator will fail in starting the TDP; this should come from the PRA team for operator actions. Once the simulation time reaches the sampled starting TDP time and if it is successful, then it moves to the “RunningTDPManually.” Here the failure rate event “OpFailsTDP” samples when the operator fails during the process of manually running the pump. The failure rate for this should also come from the plant’s PRA team. The other event “MaxMitigationTime” is the RAPT time and used to indicate that this task is not needed for the simulation anymore and the scenario can be evaluated to see if there was core damage.

A-3.3.2 Fire Water For Cooling

Fire water should only be included if the system cannot be disabled from outside of the protected area. The logic tree used determines if the fire water system, shown in Figure A-19, consists of having a fire water supply and the feed valves. One of these would be a secondary target for the attack scenario to prevent the use of firewater. Either could also be set failed/false to disable the fire water prevention option if the facility cannot justify using fire water.

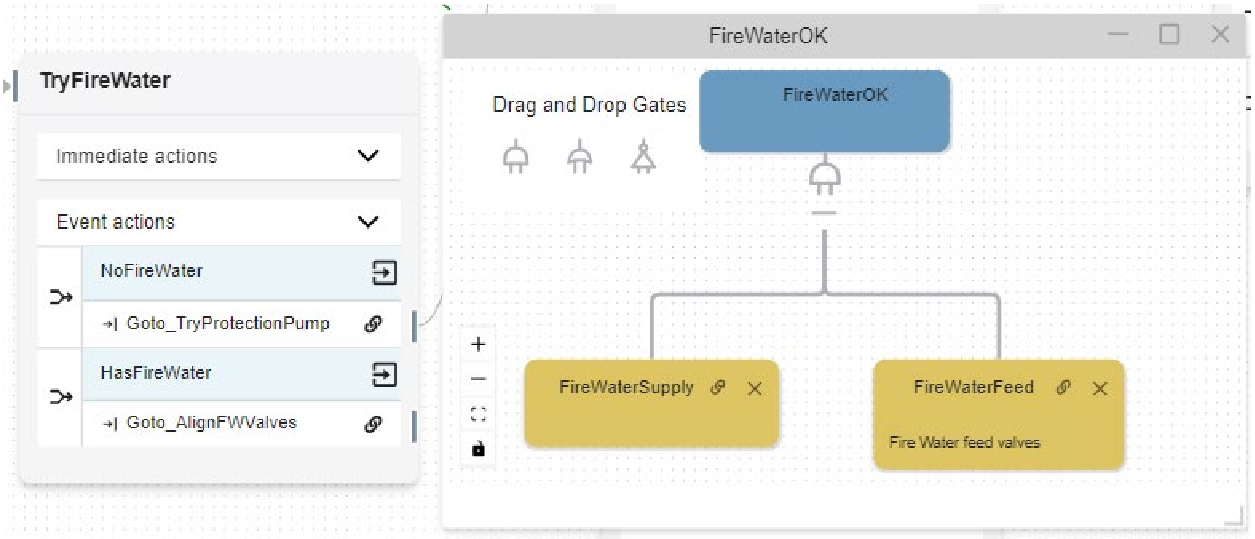


Figure A-19. The prevention option of using fire water evaluates the “FireWaterOK” logic tree.

In the generic model to use fire water, the valves must be aligned and depressurized enough to let fire water in as shown in Figure A-20. For a PWR this would be depressurizing the steam generators, for a BWR it would be the core pressure. Both actions use a distribution for the timing and should be set according to the facility specific times. After the event “Depressurized” occurs so that the fire water is running, three actions take place. The action “Set_AltCoolingStartTime” saves the time that alternate cooling was achieved. Then “Goto_EvaluateTiming” action starts the evaluation to determine if there was core damage. Last, the action “Goto_FireWaterUsed” just moves to the key state “FireWaterUsed” to track how many times the fire water prevention option was used. The “FireWaterUsed” state is optional and can be removed or similar events can be added to other prevention options to track how many times they are used.

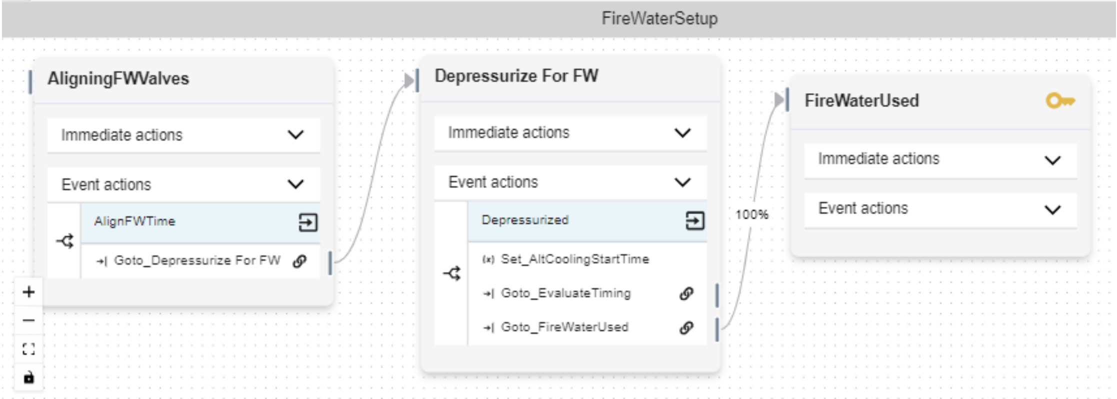


Figure A-20. The "FireWaterSetup" diagram modeling the procedure for using fire water for cooling.

A-3.3.3 Physical Security Cooling Pump

This prevention option uses a cooling pump, the same as the FLEX pump, but it is strategically located inside the protected area. All prevention options after the fire water require a pump connection and water source; so when the “TryProtectionPump” state is entered, the events “NoExtPumpConnection” and “NoWaterSource” evaluate if they are available. If not, then the simulation moves to the “Plant_Damage” key state and terminates as shown in Figure A-21. All the results that end in

“Plant_Damage” mean that none of the preventions options were available. No thermal hydraulics were run to do that evaluation. It is assumed that scenarios that have no emergency cooling and do not execute a prevention option will result in core damage.

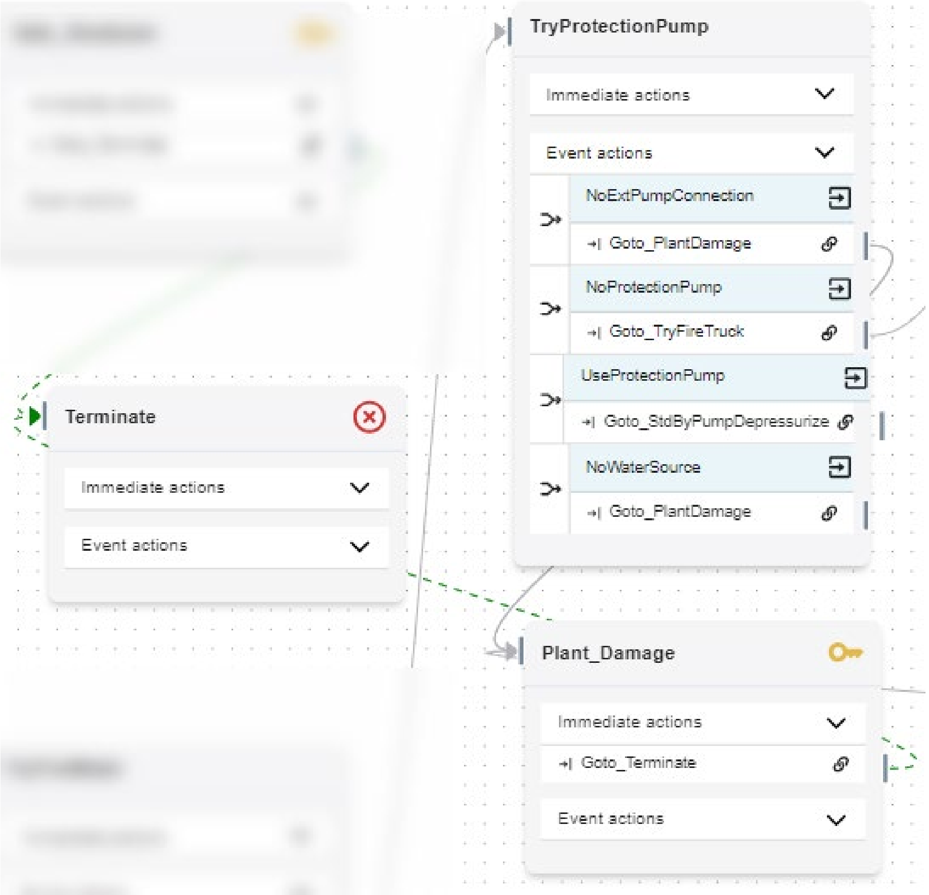


Figure A-21.The protection pump prevention option checks to see if there is a connection point and a water source to terminate early.

The logic tree to evaluate if the protection pump availability is shown in Figure A-22. This included the water source and connection to make sure it is not triggered if they are not available. If the protection pump is available, then the “PumpSetup” diagram is started from the “Goto_StdByPumpDepressurize.”

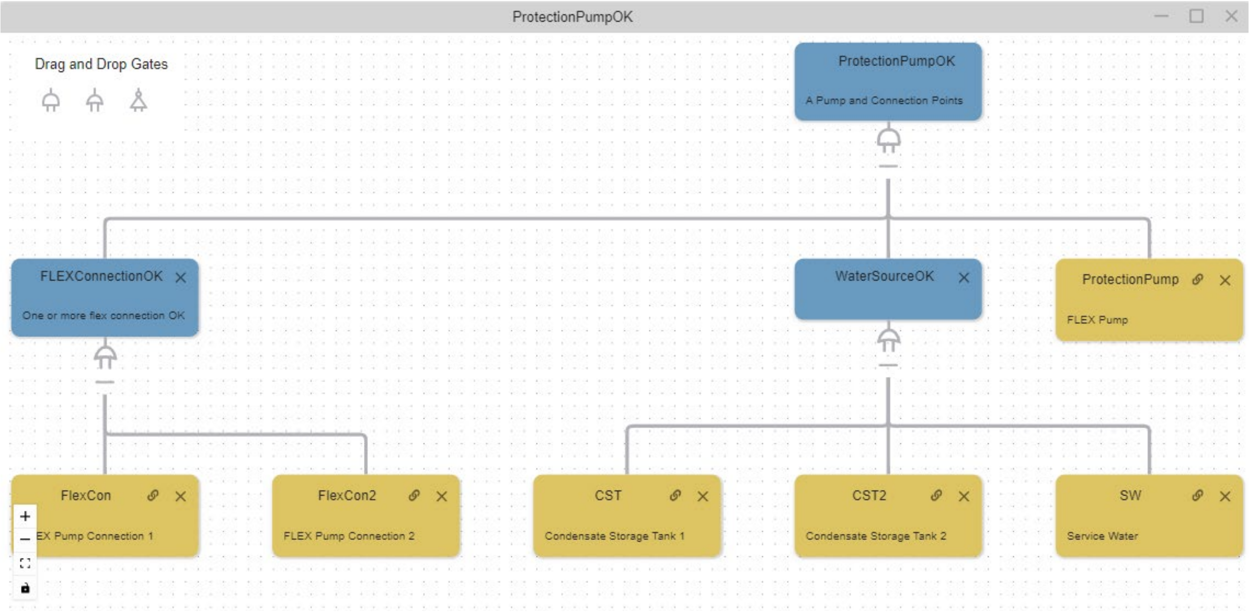


Figure A-22. Logic tree to determine if the protection pump can be used.

For all the pumps whether the physical security protection pump, fire truck pump or FLEX pump, the “PumpSetup” procedures, shown in Figure A-23, are eventually executed. For the physical security pump prevention strategy, there are no other procedures before these. In the other two cases, other procedures for getting a pump to the location are done first; see Section A-3.3.4 and A-3.3.5 for details. There are distributions for each of the tasks that need to be assigned according to facility drill data. For this generic model, the steps are to depressurize, connect pump to suction, connect pump to discharge, and align the valves. The events “NoExtPumpConnection” and “NoFlexSource” are semi-redundant as the procedures will not be started if not available but ensure if the model is modified that not having those items available will cause the pump to not be used. In the final state, “StdByPumpReady” checks to see if any standby pump is running, and if so, it then saves off the start time and calls the “Goto_EvaluateTiming” to determine if there will be core damage.

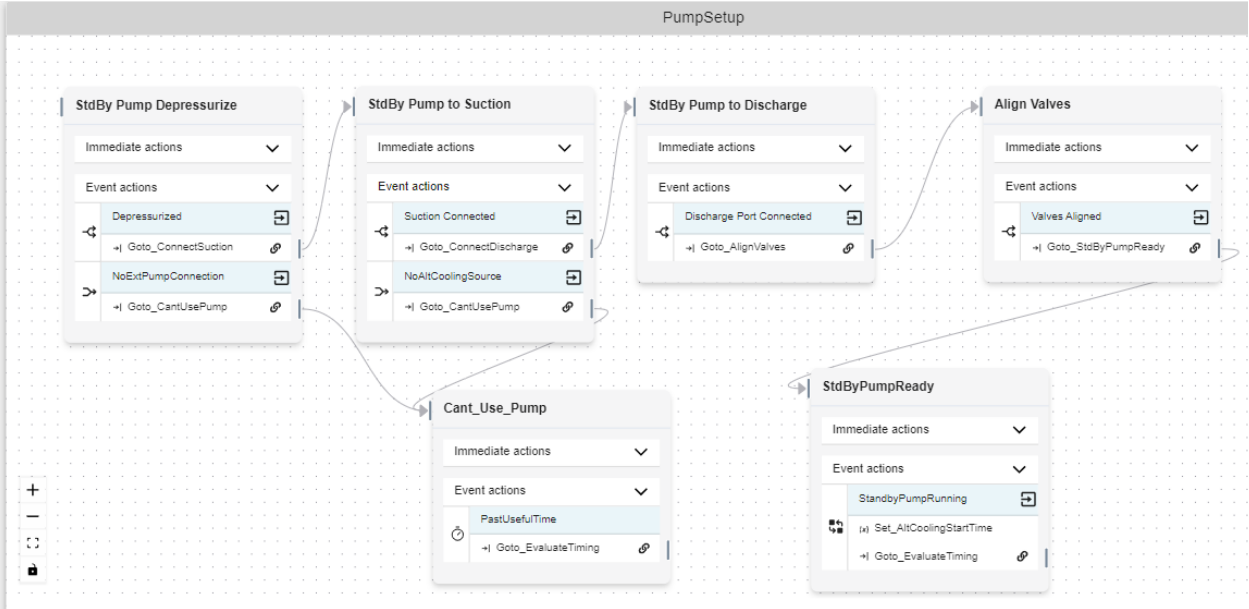


Figure A-23. Procedures for setting up the protection pump for core cooling.

A-3.3.4 Cooling Water And Pump From Fire Truck

The generic model includes using the pump from a fire truck to provide cooling water. This requires that the truck is inside the protected area and it has the proper connections and pumping requirements. If the logic tree "FireTruckOK" shown in Figure A-24 evaluates to true, then the event "UseFireTruck" is triggered and starts the procedures in the diagram "FireTruckSetup."

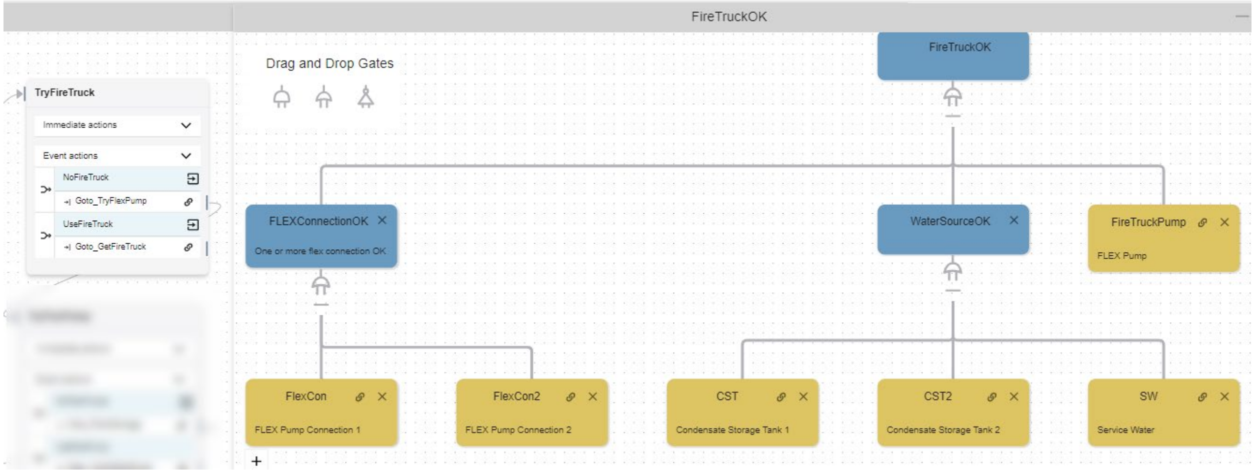


Figure A-24. Logic tree "FireTruckOK" evaluates if the fire truck can be used.

The "FireTruckSetup" diagram is very simple; it only has one state "GetFireTruck," as shown in Figure A-25. The time for getting the fire truck needs to be set according to facility drill data. After the fire truck is placed, the common "PumpSetup" procedures are started.

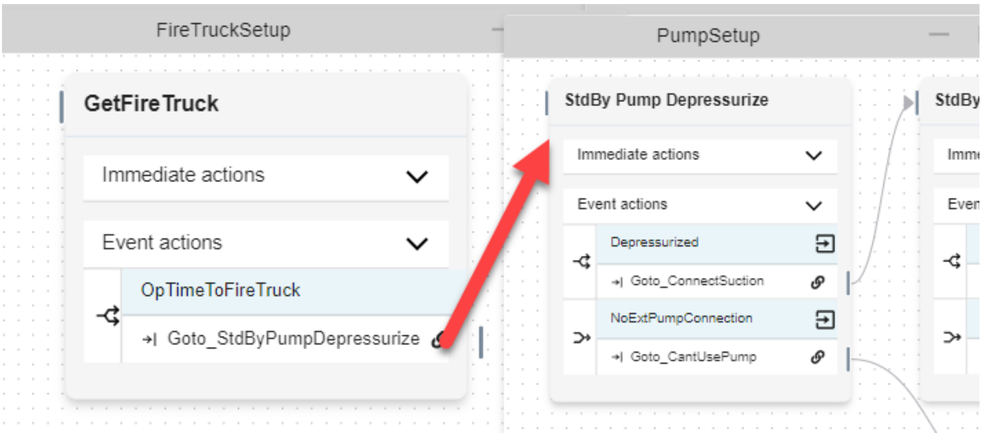


Figure A-25. The procedure of using a fire truck as a prevention option.

A-3.3.5 FLEX Pump

Using the FLEX pump as a prevention option must be shown to be viable for attack scenarios; typically, FLEX equipment is outside the protected area and usually cannot be used as part of the protection strategy. If the FLEX equipment is staged or other justification allow for it, then include it in the prevention options. If the equipment is staged, it may be more beneficial to put it in order before the "ProtectionPump" state. The "FlexPumpOK" logic tree is similar to the ones in the previous sections.

The procedures for using the FLEX pump are in the "FlexPumpSetup" as shown in Figure A-26. The tasks include sending an operator or staff to get the pump, clearing a route if needed, and transporting the pump to the designated location. As with other tasks, use timing from facility data or drills to determine the distributions in each of the events. After event "FLEX Pump at Stating Area," the common standby pump procedures are executed which in turn call the "Goto_EvaluateTiming" to determine if there will be core damage.

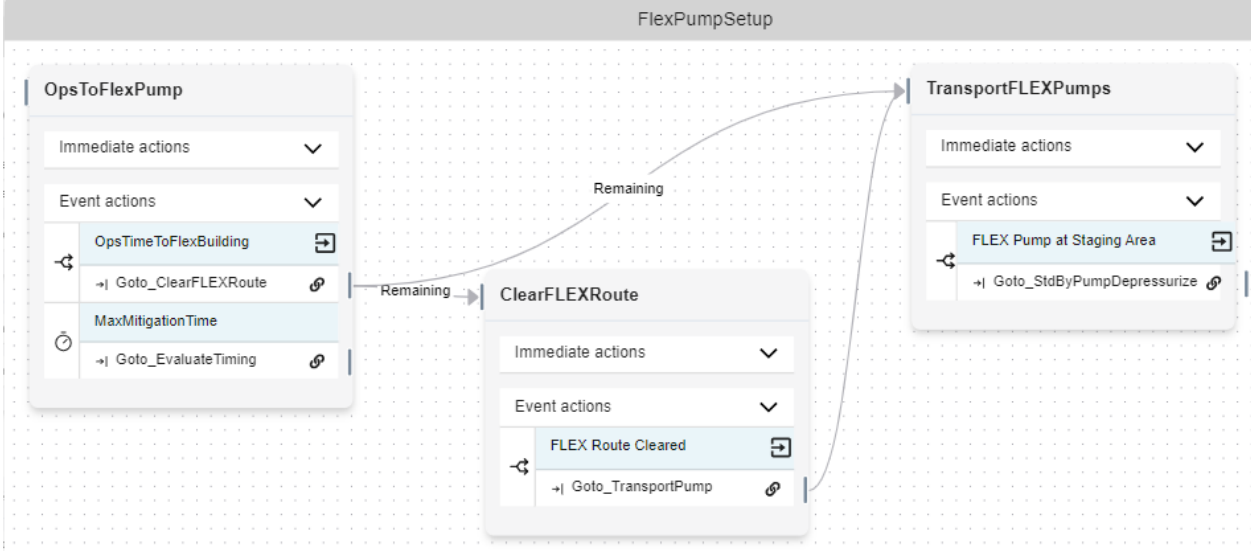


Figure A-26. The procedures for using the FLEX pump as a prevention option.

A-3.4 Evaluate Timing Outcomes

The final piece of the simulation run is determining if there was core damage. As each scenario will have different times for when the adversaries hit their primary targets and there are different times the prevention options were implemented, a custom thermal hydraulics evaluation may be needed to determine if there was core damage. The diagram "EvalPumpTiming," shown in Figure A-27, starts in the state with the same name after the prevention methods have started and we know the times for all the variables used in the thermal hydraulics evaluation. EMRALD is set up to use the MAAP thermal hydraulics software. As this can take 2–20 min per run, the model allows the analyst to optimize and only run it when it is necessary. For example, a previous analysis may inform that if reactor cooling is not established within 3 hours, then it does not matter when components are sabotaged because core damage will certainly happen regardless. The "DamageBounds" event allows you to specify the conditions where you know there will be plant damage. On the other side, the "FlexEasilyInTime" event allows for conditions where it is known that there will not be plant damage. These events greatly reduce the computation time of running the full analysis.

The "NeedThAnalysis" event has the boundaries where we do not easily know if there would be core damage. This event moves the simulation to the RunTH state which executes the thermal hydraulics model through the immediate action "Run_MAAP" and assigns the core damage time after it is done with the "Set_CoreDamageTime" event, which executes right after the MAAP is done running. The event "CD_Time" is assigned to the core damage time, and the "MaxMitigationTime" event is the RAPT time. Whichever occurs first is the outcome of the simulation run.

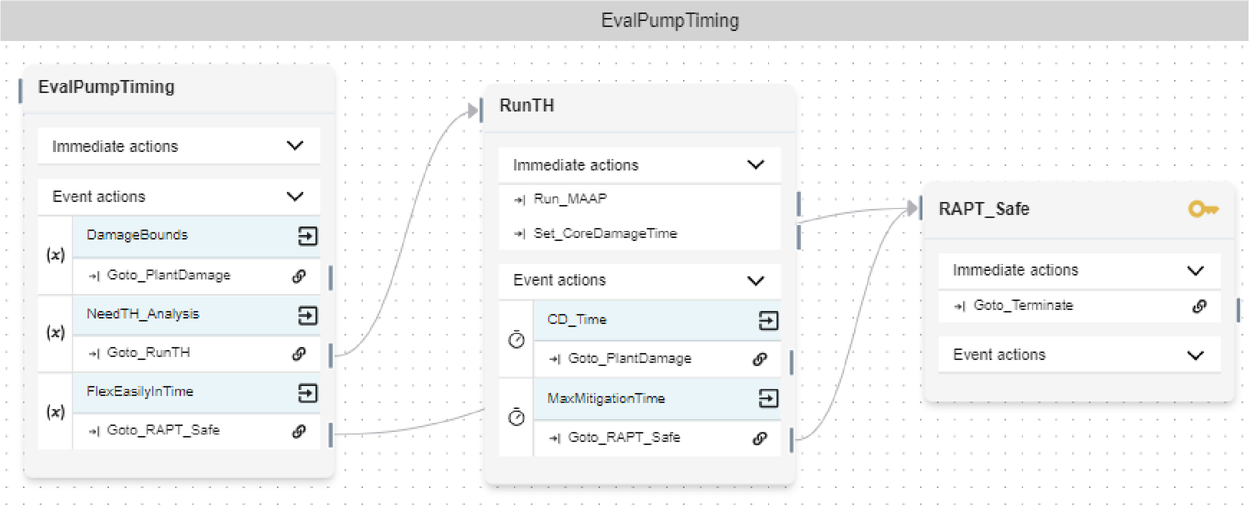


Figure A-27. The "EvalPumpTiming" determines if there is core damage.

A-4. Safety Systems

These sections go over the different safety systems modeled in the generic EMRALD models. If any of the safety systems are OK, then no after-attack alternative-protection methods are needed. Each generic model has a main logic tree evaluating the BWR safety systems. This tree evaluates the different systems and major components or other requirements for the system that could be part of the target sets. If a particular facility does not have one of those systems, then the components can just be set as failed on startup of the simulation. See Section A-5 for how to do this.

A-4.1 Common Items

A-4.1.1 Some Control

Both models evaluate if the facility still can control the safety system. For the generic model, either the control room or one of two alternate panels must be available as shown by the Boolean logic in Figure A-28.

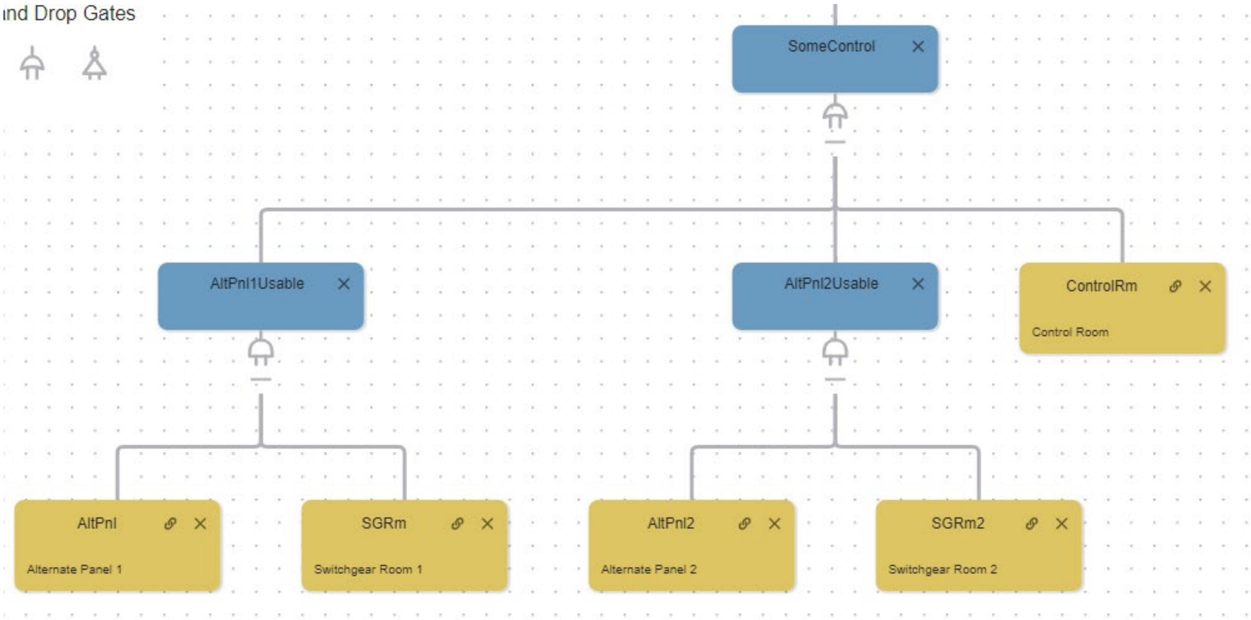


Figure A-28. The “SomeControl” branch of the logic tree.

A-4.1.2 AC Power

The “ACPowerOK” logic tree shown in Figure A-29 evaluates if there is a source of AC power either from offsite power or diesel generators. If the tree evaluates to a false, then any other logic tree using it as a subbranch will get a false or an event evaluating the tree for failure is triggered. The generic model has two diesel generators; if a facility has more or other sources of AC power, they need to be added.

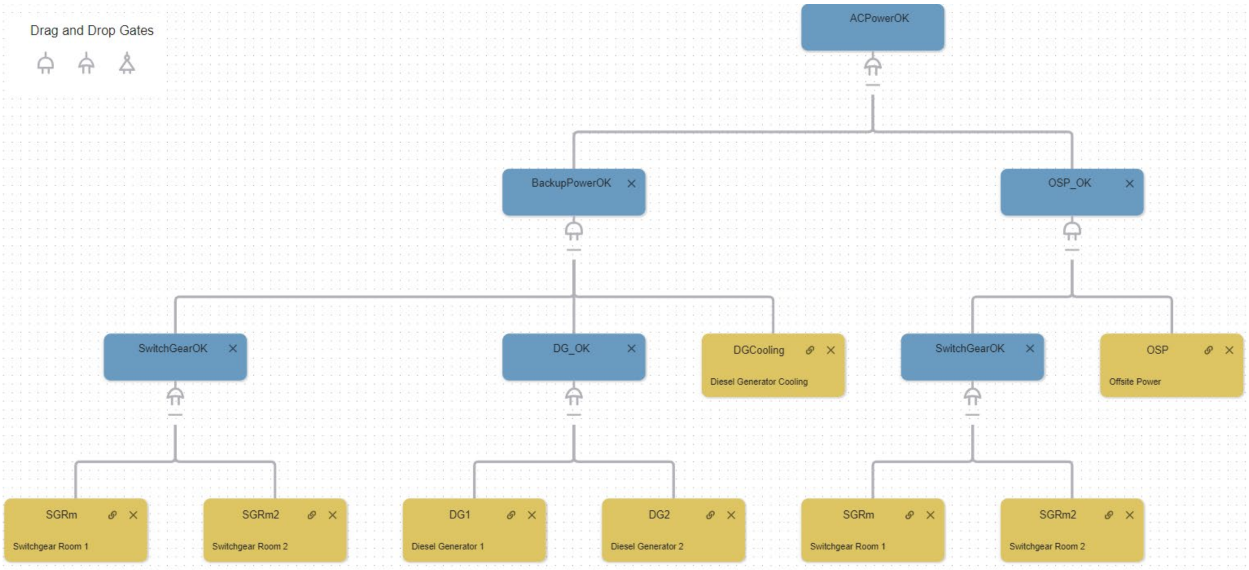


Figure A-29. ACPowerOK logic tree.

A-4.2 Included PWR Systems

The generic PWR currently only looks at safety systems on the secondary cooling side. While there are things that can be done on the primary side, scenarios involving primary cooling do not get adjusted from EMRALD, and evaluation should stop at the FoF simulation results. They must still be included in the post reduction evaluation to make sure their protection level does not go down. The primary tree evaluated for PWRs is “AFWPumpingOK” for auxiliary feedwater (AFW) systems as shown in Figure A-30. All the safety systems evaluated for the PWR are contained in this logic tree. For AFW to be running, it requires a cooling pump, control, and cooling water from condensate.

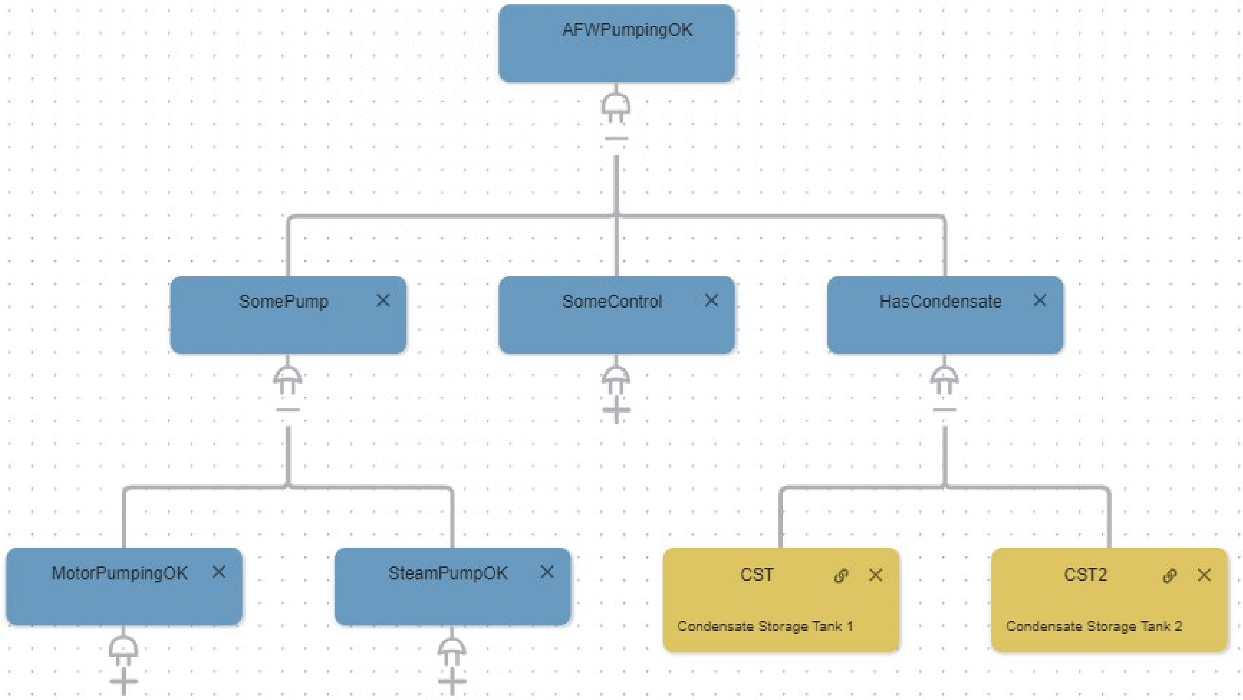


Figure A-30. Logic tree for the auxiliary feedwater system.

A-4.2.1 Condensate

The generic PWR model has two condensate tanks: CST and CST2. If a facility has other options designed as part of their safety system, they can simply create a new component diagram for them along with being a target and then add it to the "HasCondensate" gate shown in Figure A-30.

A-4.2.2 Steam Pumps

Two turbine-driven pumps are part of the generic model, and if one is running, then pumping is available as shown in Figure A-31. If a facility only has one, they can either delete the node from the tree or go to the "TDP2" diagram and start it off in the failed state.

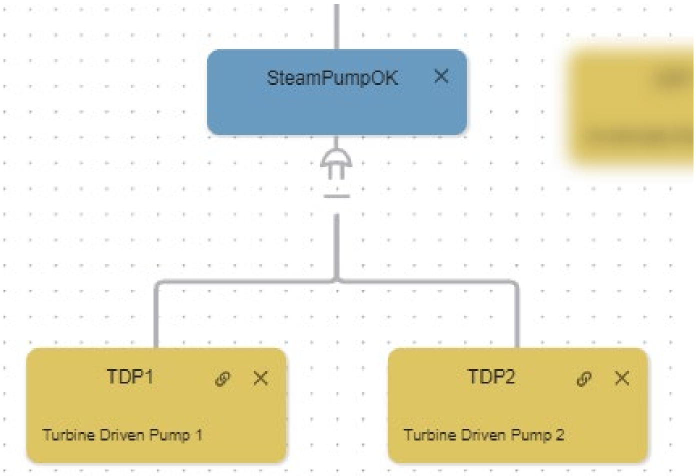


Figure A-31. The steam pumping options branch in the "AFWPumpingOK" tree.

A-4.2.3 Motor Driven Pumps

There are also electric pumping options if steam-driven system is not available, as shown in Figure A-32. Motor-driven pumps requires both power and a pump. The model has two motor-driven pumps (MDP), "MDP1" and "MDP2." AC power or DC power from the battery system can run these pumps. If the pumps are running on DC battery power, then when the batteries are depleted, this branch of the tree will fail. As in other cases, add or remove MDPs to match the facility.

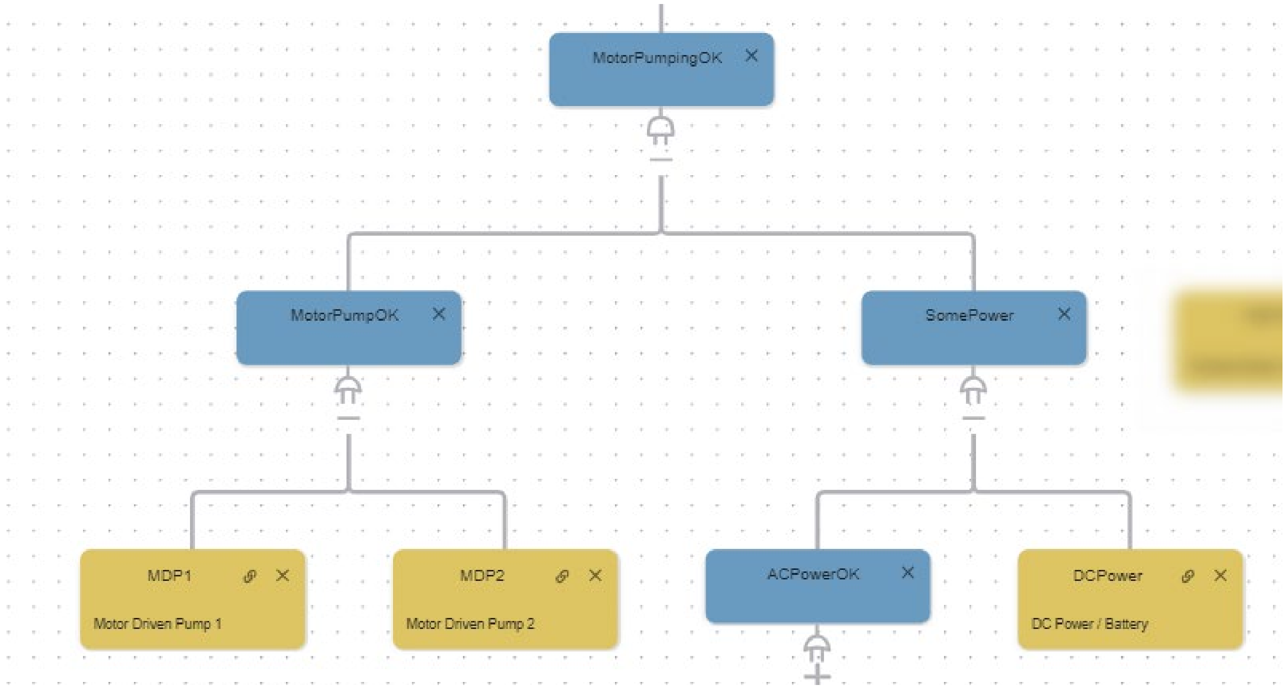


Figure A-32. Motor-driven pumping option for the "AFWPumpingOK" logic tree.

A-4.3 Included BWR Systems

The main logic tree evaluating the BWR safety systems is the "ECCS OK" logic tree. This tree evaluates the different systems and major components or other requirements for the system that could be part of the target sets. If a particular facility does not have one of the systems, then the components can just be set as failed on startup of the simulation. See Section A-5 for how to do this.

A-4.3.1 Isolation Condenser

The isolation condenser (IC) provides cooling if the core is pressurized, and IC components are available as shown in Figure A-33. If there is more than one target for IC components, then those components can be added as diagrams, and this can be broken down into an "OR" gate with the different components under it.

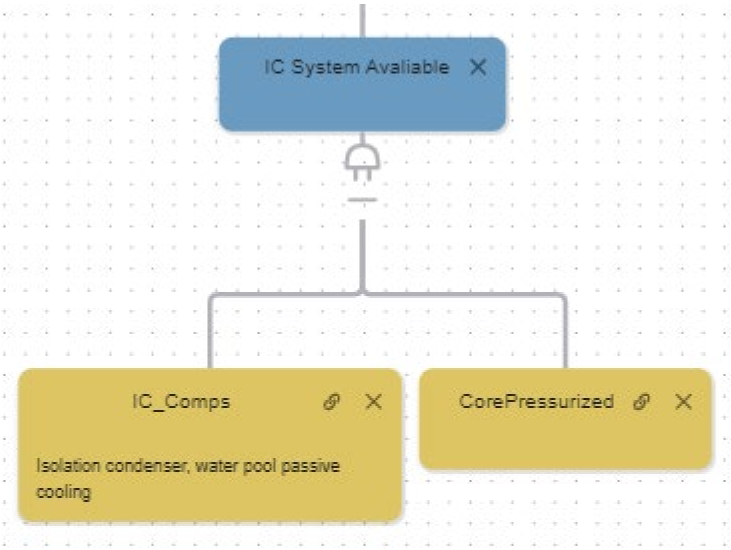


Figure A-33. IC system availability logic tree branch.

A-4.3.2 Water Source

Both high-pressure and low-pressure core injection systems require a water source. This requirement is satisfied by using an "AND" gate between the "PrimaryWaterSource" and the "ActiveCoreInjection" gate as shown in Figure A-34. The generic BWR has a wet well, a two condensate storage tanks.

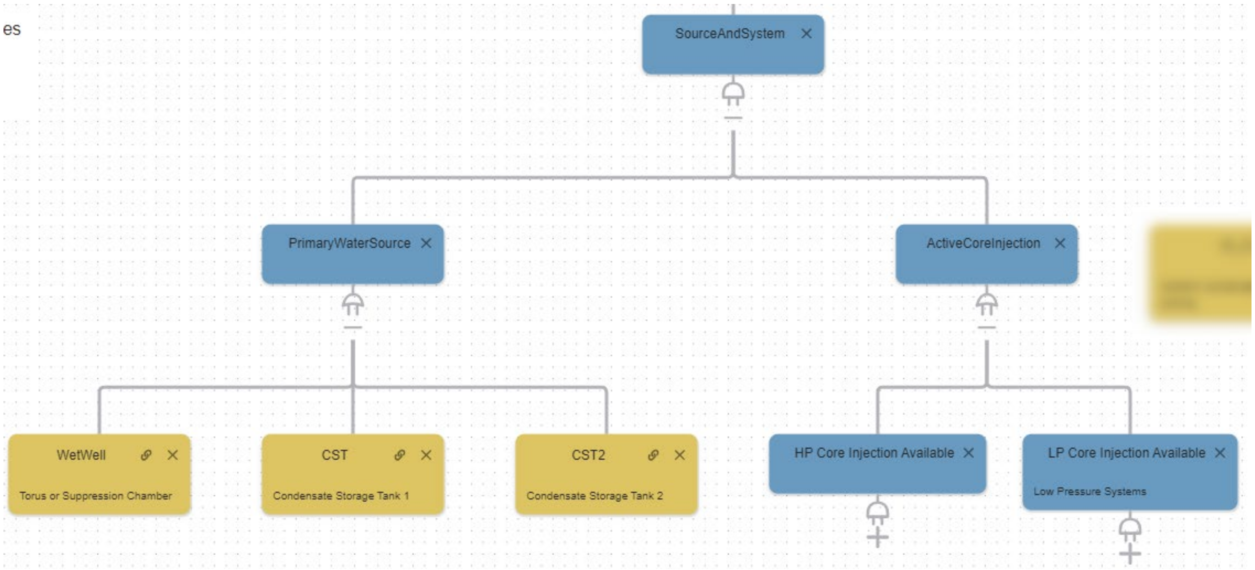

es
SourceAndSystem
PrimaryWaterSource
ActiveCoreInjection
WetWell
Torus or Suppression Chamber
CST
Condensate Storage Tank 1
CST2
Condensate Storage Tank 2
HP Core Injection Available
LP Core Injection Available
Low Pressure Systems

Figure A-34. Water source options for the core injection options.

A-4.3.3 High Pressure Injection Options

To have a high-pressure system working, the core must be pressurized by either the high-pressure core injection (HPCI), HPCS, or the RCIC system as shown in Figure A-35.

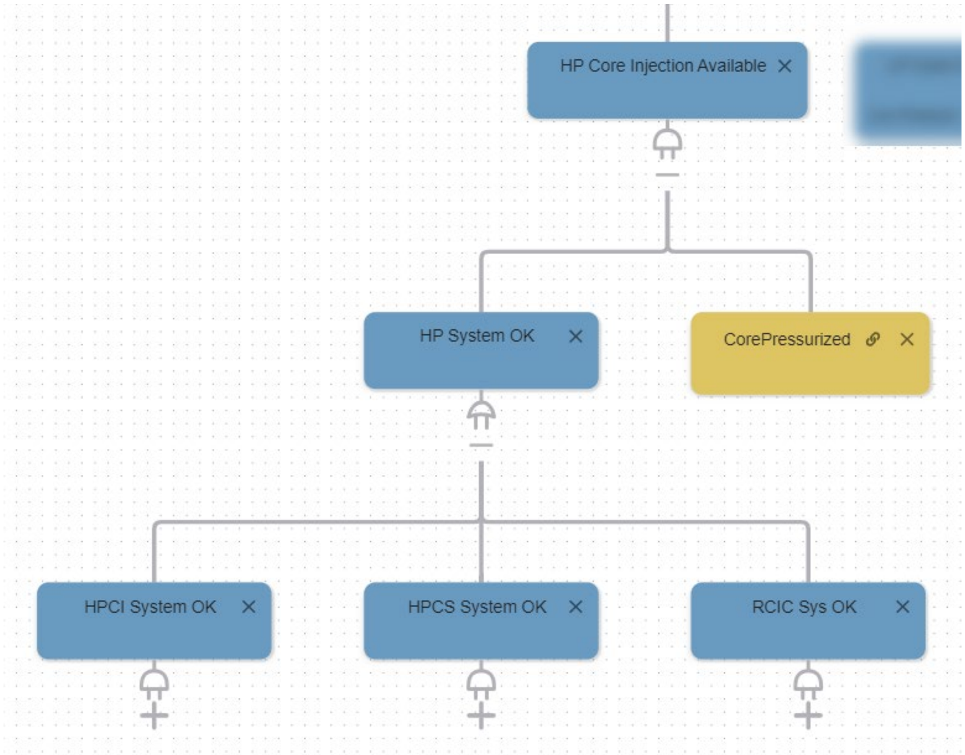


Figure A-35. High-pressure safety system options.

The HPCI system requires either power from batteries or the main AC power and the HPCI turbine-driven pump to be available, as shown in Figure A-36.

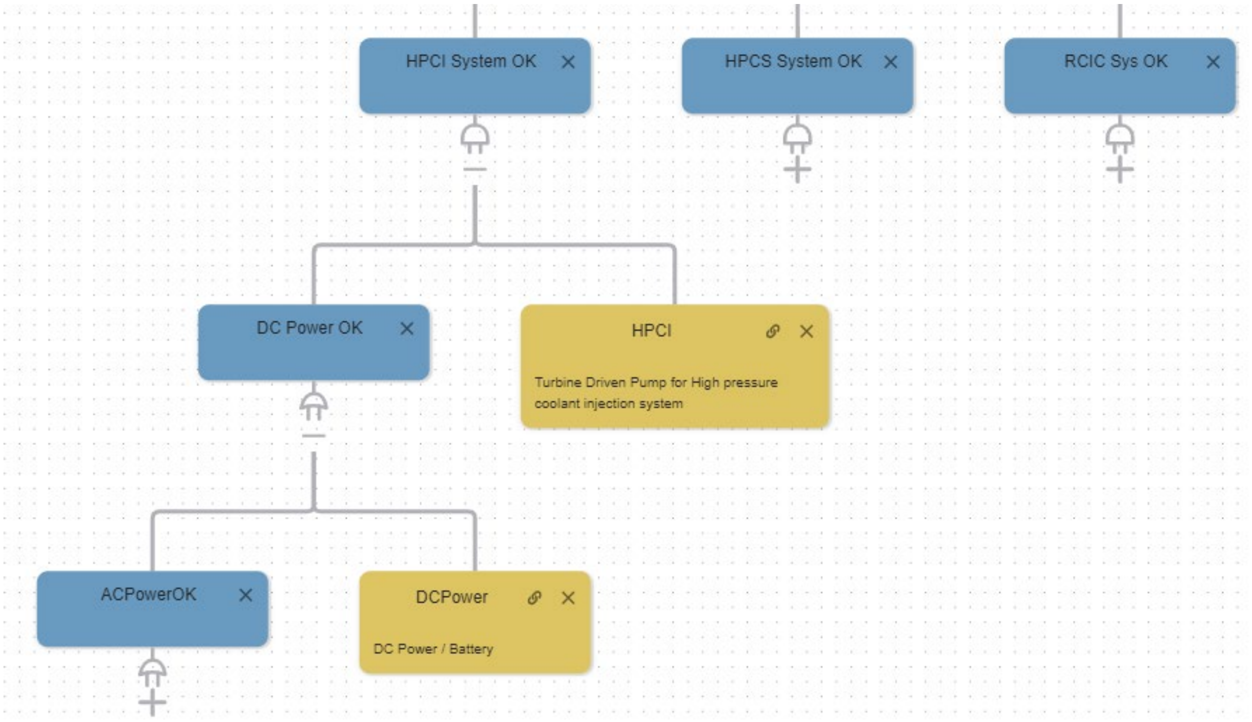


Figure A-36. The HPCI system logic tree branch.

The HPCS system requires a dedicated HPCS diesel generator or offsite power and the HPCS motor-driven pump as shown in Figure A-37.

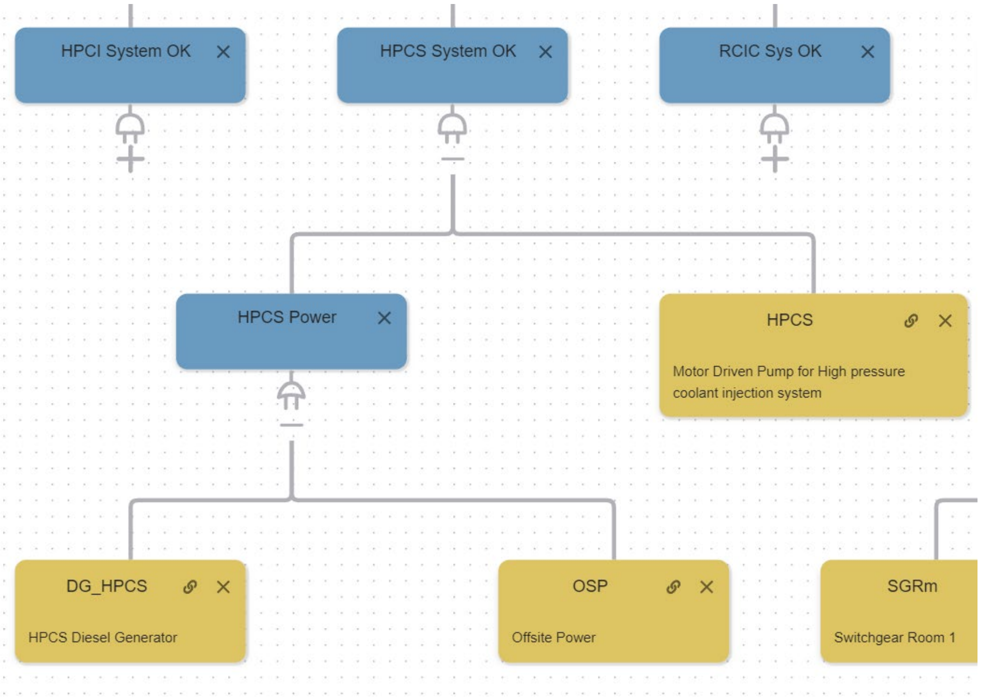


Figure A-37. The HPCS system logic tree branch.

The RCIC system requires the RCIC turbine-driven pump and power from either the batteries or the main AC power as shown in Figure A-38.

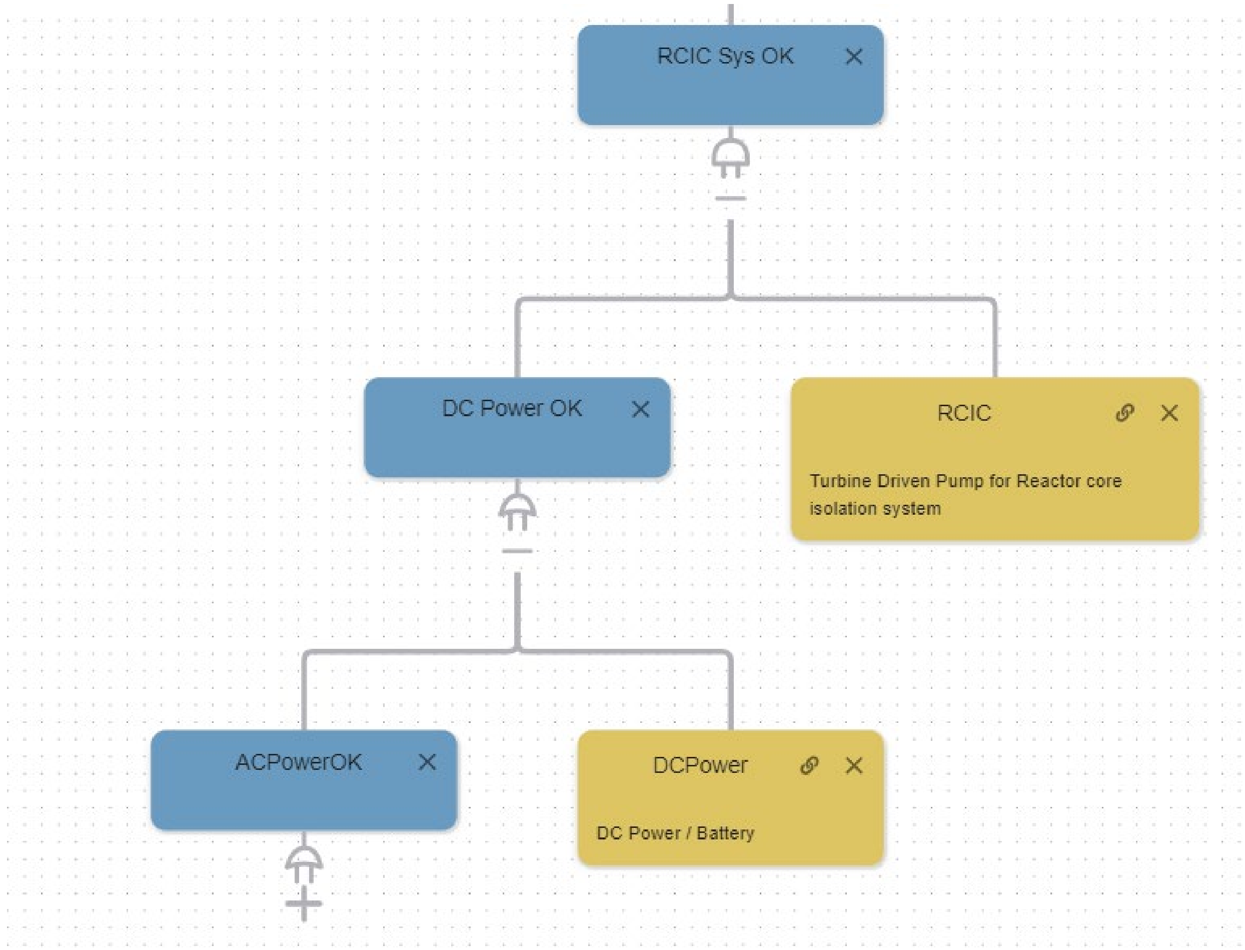


Figure A-38. The RCIC system logic tree branch.

A-4.3.4 Low-Pressure Injection

For low-pressure coolant injection, the core must be depressurized, operators must be able to depressurize and inject with the low-pressure coolant injection, or the low-pressure core spray must be available as evaluated in logic tree branch “LP Core Injection Available” shown in Figure A-39.

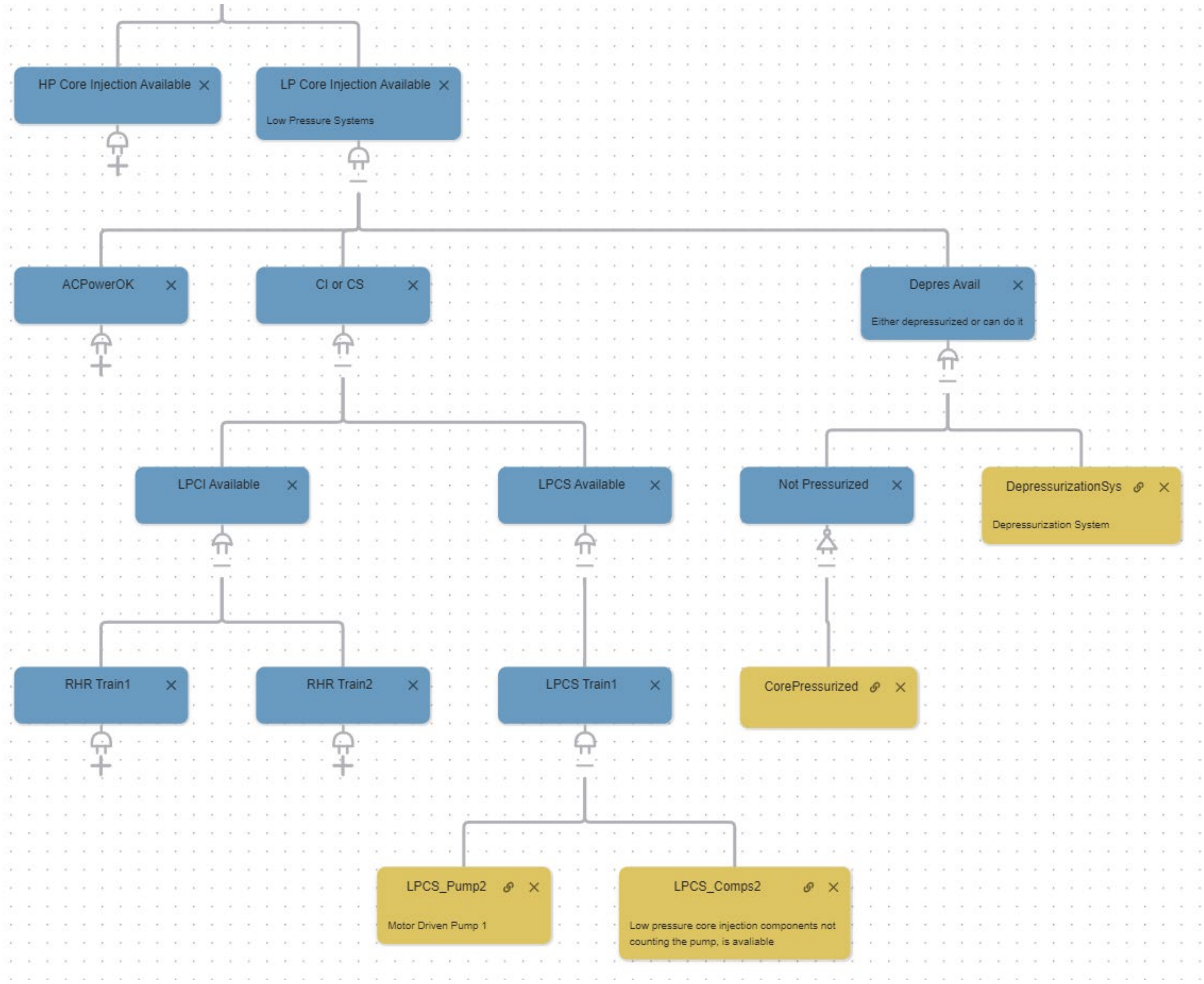


Figure A-39. Low-pressure core injection logic tree branch.

Both the LPCI and LPCS have trains consisting of a pump and components as shown in Figure A-40. Each of these can be a target but would require multiple of them to disable all the LPI options.

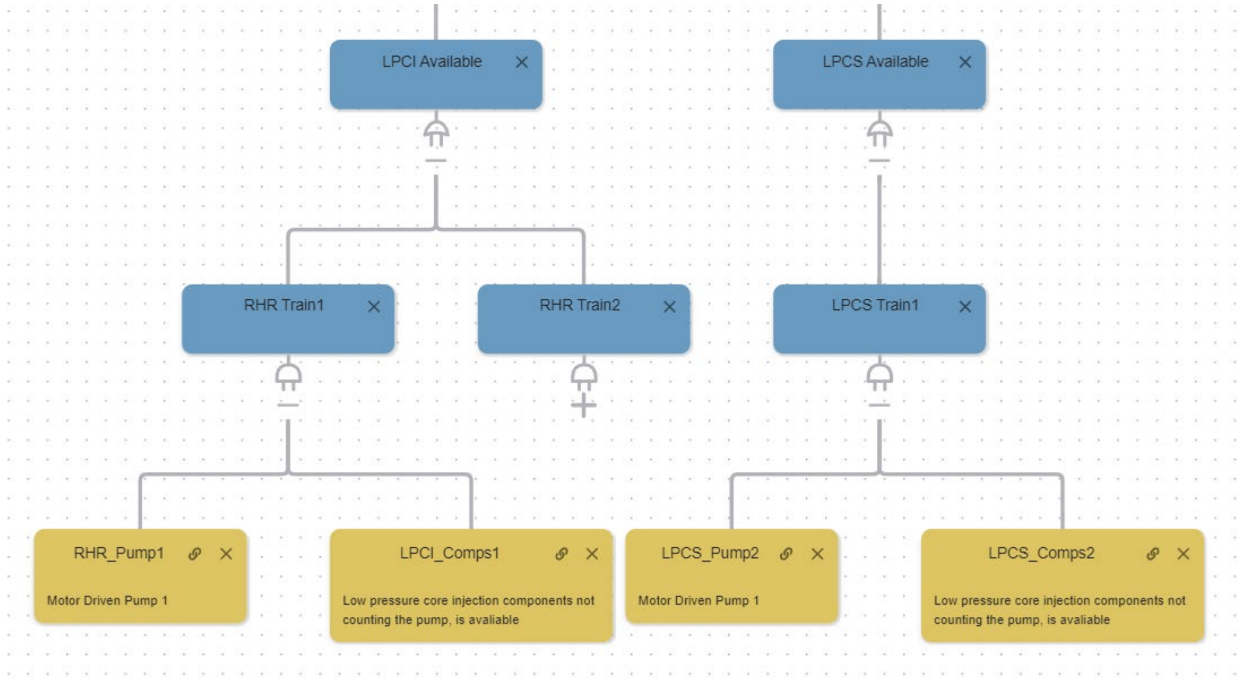

LPCI Available
LPCS Available
RHR Train1
RHR Train2
LPCS Train1
RHR_Pump1
Motor Driven Pump 1
LPCI_Comps1
Low pressure core injection components not counting the pump, is avaliable
LPCS_Pump2
Motor Driven Pump 1
LPCS_Comps2
Low pressure core injection components not counting the pump, is avaliable

Figure A-40. The logic tree branches for the trains under LPCI and LPCS.

A-5. Customizing for a Specific Facility

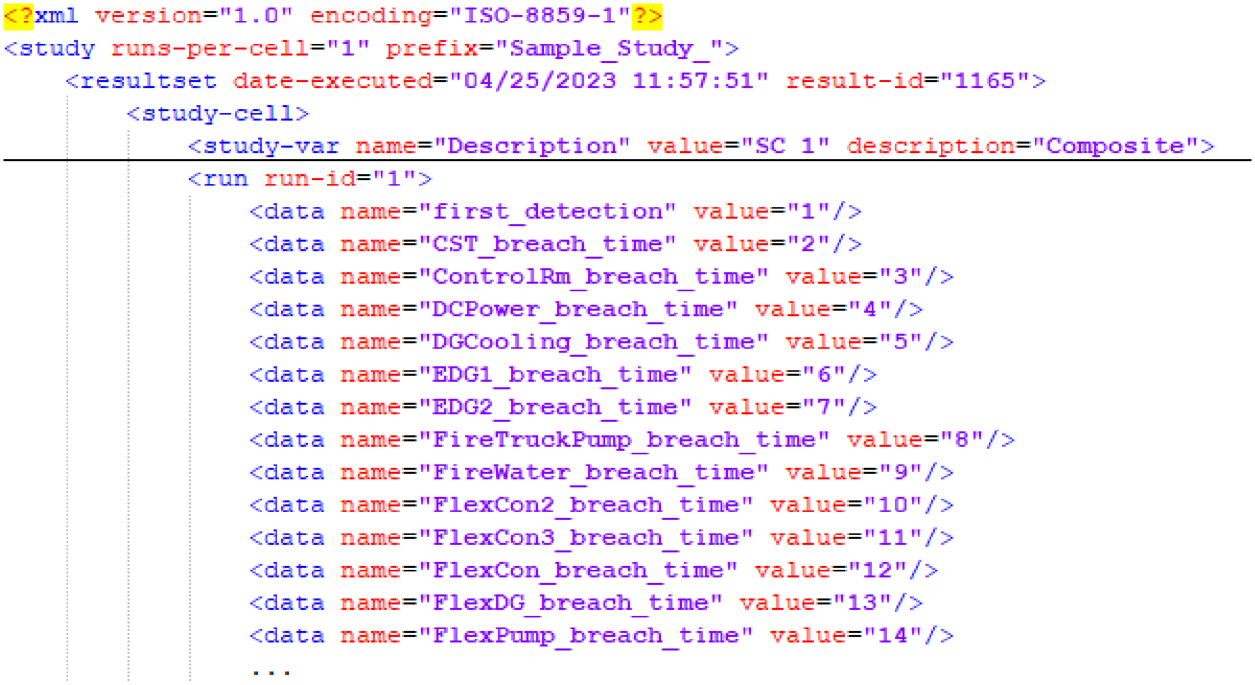
The generic model represents common features for PWRs but must be modified to match the facility and physical security scenarios. This section goes over how to first set up test cases and then show examples on modifying the model. This section is meant as a general guide and cannot cover all the things that may need to be changed to match a facility.

A-5.1 Test Cases

When customizing, it is recommended to develop test case input files for the scenarios and verify that the model reaches the points it should. For example, all FoF result files that hit the primary targets should start the evaluation of alternative prevention options. And all cases that do not hit the primary targets should end in the state “NotAllPrimaryHit.” To develop test cases, create a dummy FoF result file with the features to be tested. The examples shown use RhinoCorps Simajin results file format. The generic model assumes the doc path for the results is in one folder up from the EMRALD model location and is called “Simajin.results.xml.” This can be changed but must be done for all the “SimjIn_” variables.

A-5.1.1 Verify Target Hit Cases

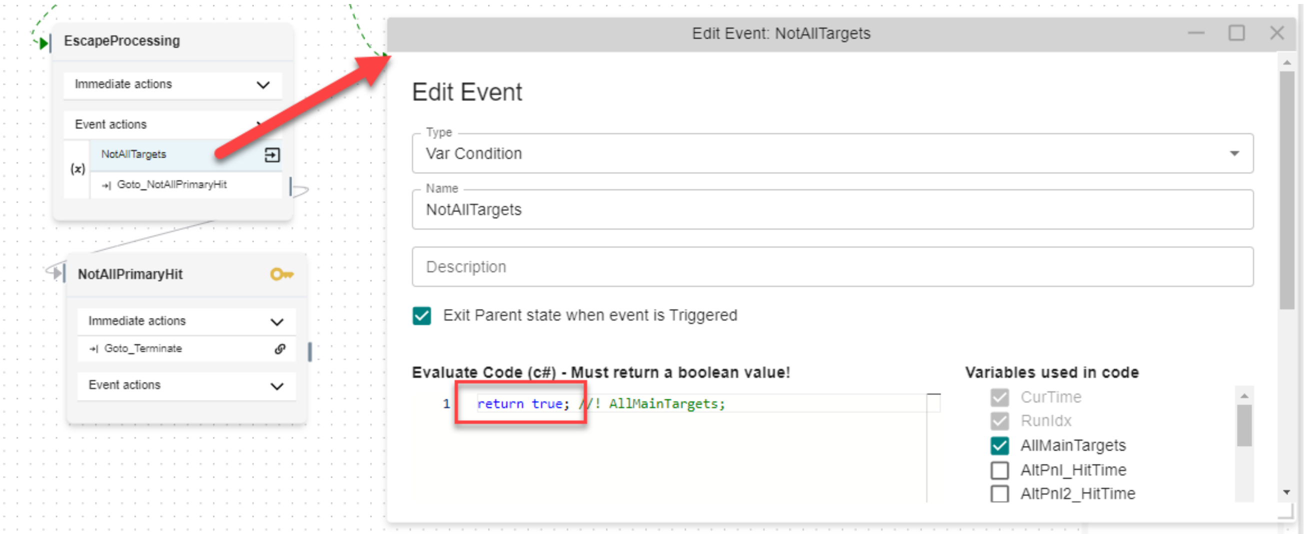
Create an FoF result file with hit times for every target and event time that is available in the model as shown in Figure A-41. Use different hit times for each of the components to make sure that none are duplicated.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<study runs-per-cell="1" prefix="Sample_Study_">
    <resultset date-executed="04/25/2023 11:57:51" result-id="1165">
        <study-cell>
            <study-var name="Description" value="SC 1" description="Composite">
            <run run-id="1">
                <data name="first_detection" value="1"/>
                <data name="CST_breach_time" value="2"/>
                <data name="ControlRm_breach_time" value="3"/>
                <data name="DCPower_breach_time" value="4"/>
                <data name="DGCooling_breach_time" value="5"/>
                <data name="EDG1_breach_time" value="6"/>
                <data name="EDG2_breach_time" value="7"/>
                <data name="FireTruckPump_breach_time" value="8"/>
                <data name="FireWater_breach_time" value="9"/>
                <data name="FlexCon2_breach_time" value="10"/>
                <data name="FlexCon3_breach_time" value="11"/>
                <data name="FlexCon_breach_time" value="12"/>
                <data name="FlexDG_breach_time" value="13"/>
                <data name="FlexPump_breach_time" value="14"/>
                ...
```

Figure A-41. Example FoF result file to test input data.

This is just to test that the variables are all getting assigned correctly, so there is no need to run the full model. There are several ways to make the model stop early; a simple way is to open the "Initiate_Attack" diagram and use the existing "EscapeProcessing" state making sure the "NotAllTargets" hit event returns a "True" value, as shown in Figure A-42. Note be sure to change this back when testing is complete!



Edit Event

Type: Var Condition

Name: NotAllTargets

Description

☑ Exit Parent state when event is Triggered

Evaluate Code (c#) - Must return a boolean value!

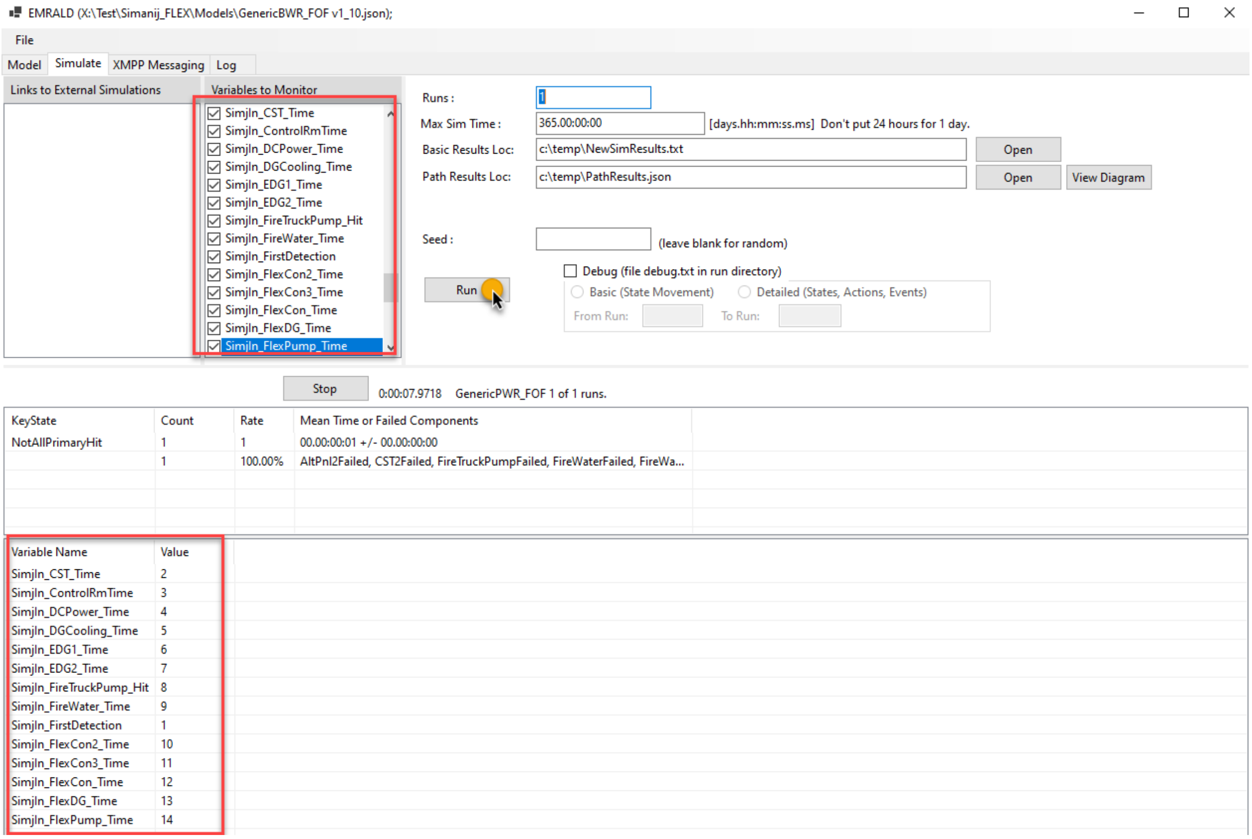
```
return true; //! AllMainTargets;
```

Variables used in code

☑ CurTime
☑ RunIdx
☑ AllMainTargets
□ AltPnl_HitTime
□ AltPnl2_HitTime

Figure A-42. How to set NotAllTargets event to return True.

Note the "run-id" and follow the instructions in Section A-7 for setting up and running the model. When running the model in the "Variables to Monitor" list, check all the variables getting data from the FoF results—in this case everything that starts with "SimjIn_", as shown in Figure A-43. Set runs to 1 and click the run button. In the bottom left, the values for the variables will be loaded. Make sure they are the same as what was in the test file.



EMRALD (X:\Test\Simanij_FLEX\Models\GenericBWR_FOF v1_10.json);

☑ SimjIn_CST_Time
☑ SimjIn_ControlRmTime
☑ SimjIn_DCPower_Time
☑ SimjIn_DGCooling_Time
☑ SimjIn_EDG1_Time
☑ SimjIn_EDG2_Time
☑ SimjIn_FireTruckPump_Hit
☑ SimjIn_FireWater_Time
☑ SimjIn_FirstDetection
☑ SimjIn_FlexCon2_Time
☑ SimjIn_FlexCon3_Time
☑ SimjIn_FlexCon_Time
☑ SimjIn_FlexDG_Time
☑ SimjIn_FlexPump_Time

Runs : 1
Max Sim Time : 365.00:00:00 [days.hh:mm:ss.ms] Don't put 24 hours for 1 day.
Basic Results Loc: c:\temp\NewSimResults.txt
Path Results Loc: c:\temp\PathResults.json
Seed : (leave blank for random)
□ Debug (file debug.txt in run directory)
○ Basic (State Movement) ○ Detailed (States, Actions, Events)
From Run: To Run:

0:00:07.9718 GenericPWR_FOF 1 of 1 runs.

KeyState	Count	Rate	Mean Time or Failed Components
NotAllPrimaryHit	1	1	00.00:00:01 +/- 00.00:00:00
	1	100.00%	AltPnl2Failed, CST2Failed, FireTruckPumpFailed, FireWaterFailed, FireWa...

Variable Name	Value
SimjIn_CST_Time	2
SimjIn_ControlRmTime	3
SimjIn_DCPower_Time	4
SimjIn_DGCooling_Time	5
SimjIn_EDG1_Time	6
SimjIn_EDG2_Time	7
SimjIn_FireTruckPump_Hit	8
SimjIn_FireWater_Time	9
SimjIn_FirstDetection	1
SimjIn_FlexCon2_Time	10
SimjIn_FlexCon3_Time	11
SimjIn_FlexCon_Time	12
SimjIn_FlexDG_Time	13
SimjIn_FlexPump_Time	14

Figure A-43. Example of testing data from FoF results.

A-5.1.2 Verify Primary Scenarios

Another test that should be run is to verify that the scenarios for the main targets will trigger the alternative prevention options. To do, this create FoF input files similar to the one for the previous section, but only include the components for the primary targets. For Simajin results, make sure that “target_goal_count” has the value of 1 meaning that all the main targets were hit. To edit the model for testing, edit the properties of the “TryTDP’s state in the “Attack_Response” diagram and make it a key state. This is done by right-clicking the state, selecting “Edit Properties,” and then changing the “Type” to “Key State” as shown in Figure A-44. Then add the “Goto_Terminate” from the actions list to the immediate actions of the “TryTDPs” state as shown in Figure A-45. Now all the test scenarios with just the primary targets hit should end in the “TryTDPs” state. If they do not, then debug and figure out why. If you get any results that end in “Safe_Shutdown,” then the safety system logic is incorrect or your input file is incorrect.

After these tests pass, then change the state back to a standard state and remove the “Goto_Terminate” immediate action.

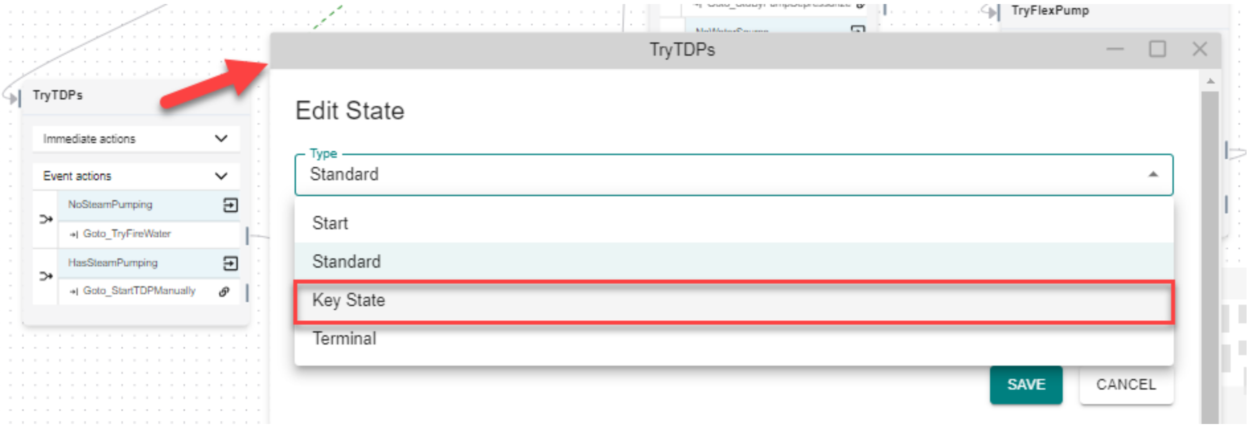


Figure A-44. Edit the properties of "TryTDPs" to change to a key state.

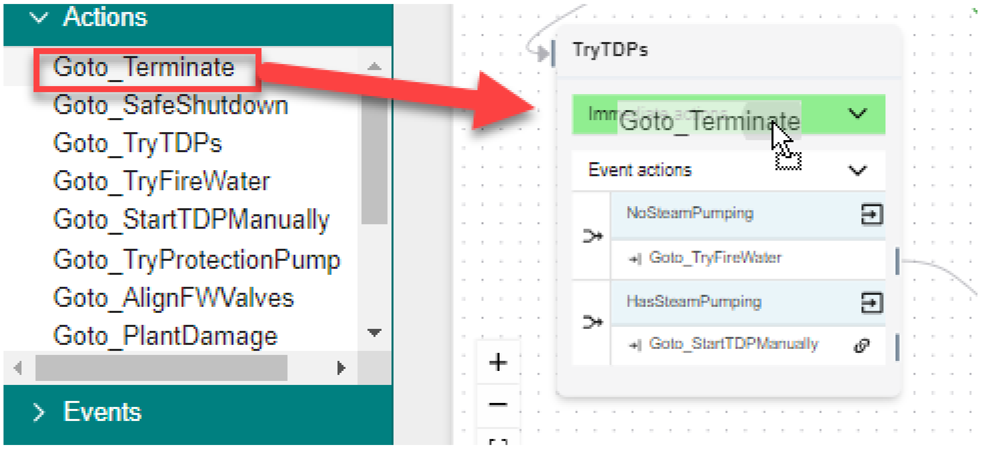


Figure A-45. Adding the "Goto_Terminate" to the immediate actions of "TryTDPs" state.

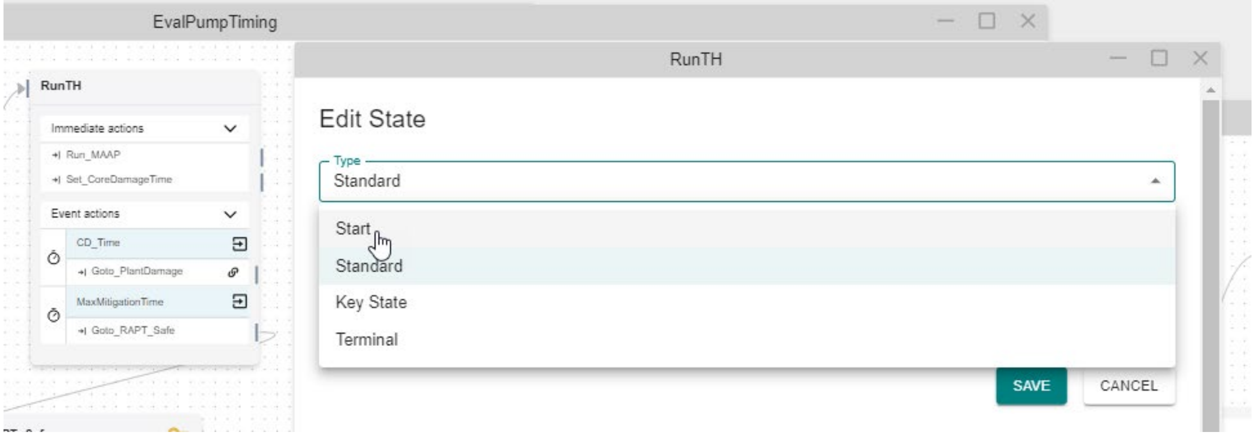
A-5.1.3 Verify Alternative Prevention Strategies

This test will verify that the alternative prevention strategies are being executed in accordance with the targets hit. Here we just want to verify that each alternative option will get to the evaluation section correctly. We do not want to run the thermal hydraulics so change the "EvalPumpTiming" state in the "EvalPumpTiming" diagram to a key state and add the "Goto_Terminate" action like with the previous test. Again, undo these modifications when done testing and issues have been fixed.

Next, copy one of the FoF result input files from the previous primary scenario tests and run. The results should end in "EvalPumpTiming" key state, and if the "View Diagram" is selected, you should see the first alternative prevention option in the path. Then add targets to the FoF result file that prevent the first prevention option. Run the model again, and the results diagram should show the second prevention option used. Repeat this until all alternative prevention options have been tested.

A-5.1.4 Thermal Hydraulics Testing.

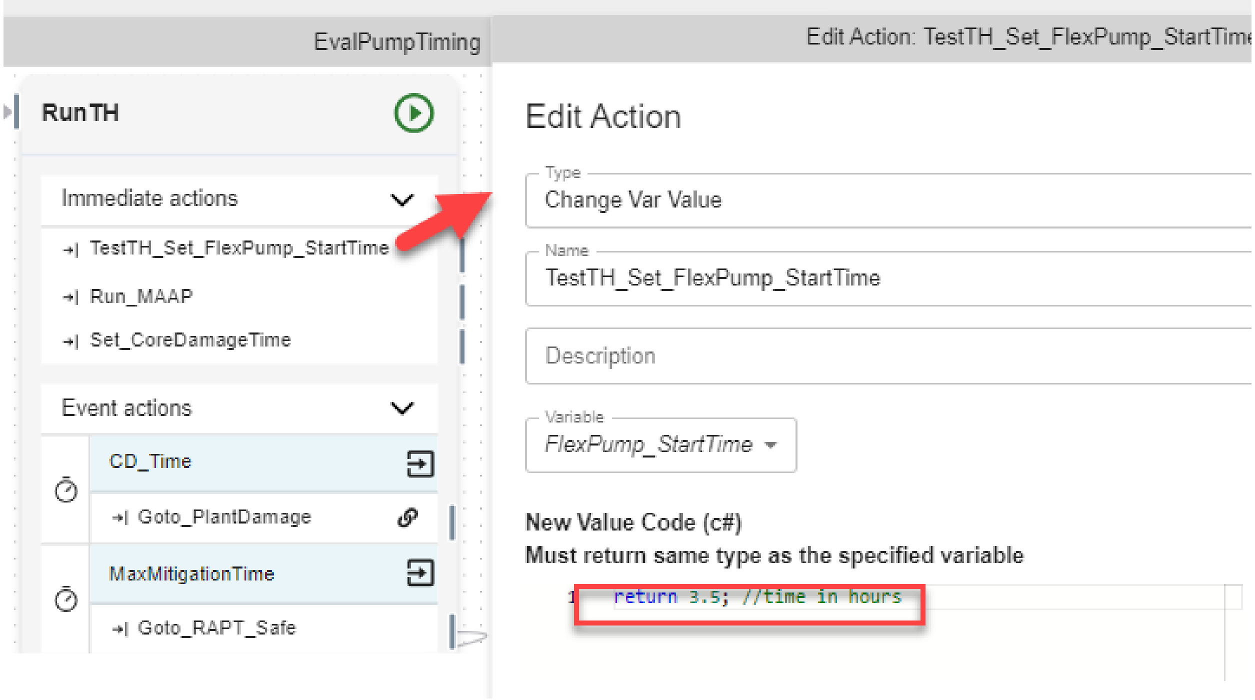
A simple method to test the thermal hydraulic (TH) model will setup and execute the MAAP model immediately when running the model. To do this, make the "RunTH" state in the "EvalPumpTiming" diagram a starting state by editing the state properties and changing its type to "Start" as shown in Figure A-46. Then the parameter values that will be changed in the TH model need to be assigned test values. This can be done by adding "Change Var Value" actions in the immediate actions of the "RunTH" state that assign the desired test values as shown in Figure A-1. This should be done for all the EMRALD variables used in the MAAP model; see Section A-6. Move them before the “Run_MAAP” action. Then run the EMRALD model, and the resulting state will be “PlantDamage” or “RAPT_Safe” depending on the MAAP results.



Edit State

Type: Standard

Figure A-46. For this test change the “RunTH” state type to “Start.”



Edit Action

Type: Change Var Value

Name: TestTH_Set_FlexPump_StartTime

Description

Variable: FlexPump_StartTime

New Value Code (c#)
Must return same type as the specified variable

```
return 3.5; //time in hours
```

Figure A-47. Assessing test values for variables used in the MAAP model by adding immediate actions.

A-5.2 Adding Initial Attack Response Options

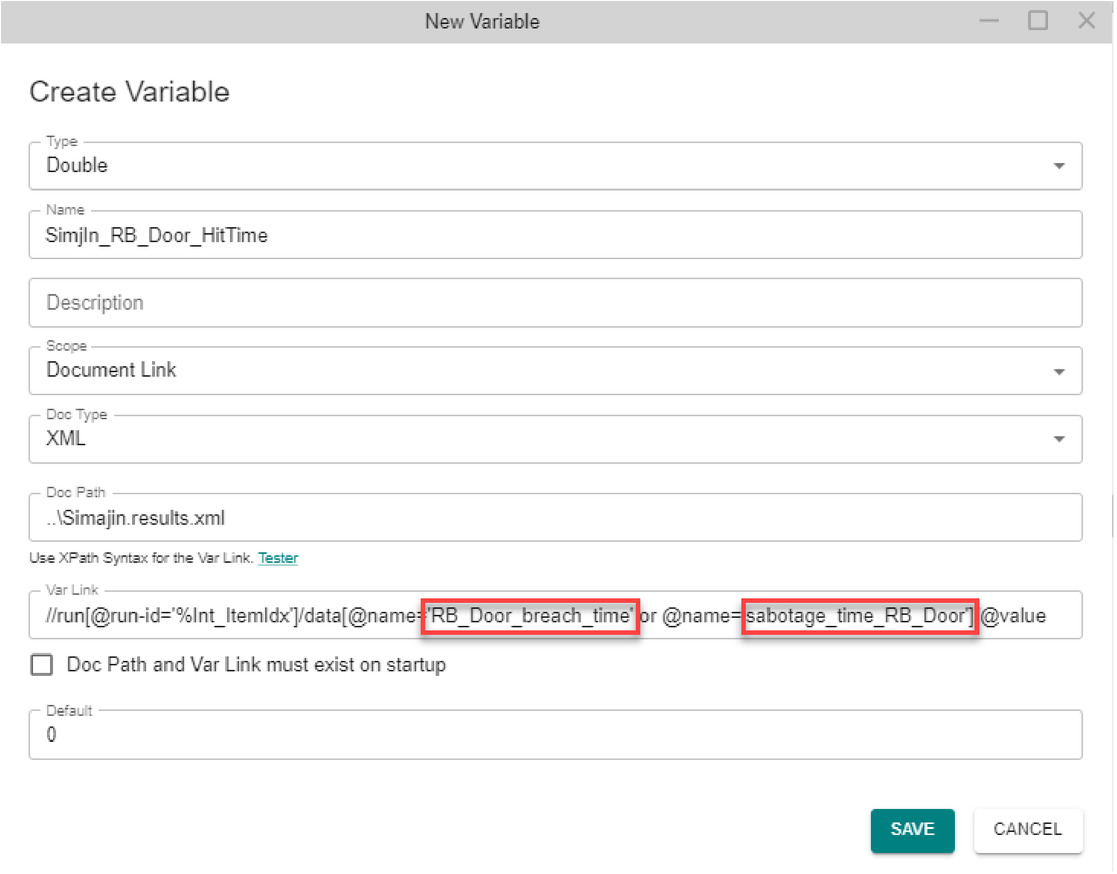
There may be protection strategies that can be implemented when an attack starts. For example, in a PWR, the steam generators normally operate at a fraction of the total volume they can hold. If an attack is detected, then the operators could start filling the steam generators with extra inventory that could greatly extend the time to core damage if key safety systems are lost. This is included in the generic PWR, but there may be other tasks such as sending an operator to an alternate location. For this example of adding an initial attack response, we are going show how to model having an operator exit the control room and get to a location before the adversary breaches the building. Note this could possibly be done in the FoF simulation model instead, but this shows how to do something if you cannot do it in the FoF simulation. For example, if the operator has actions that have multiple/repeatable steps if failed the first time or depend on other data in the EMRALD model, then it cannot be done in the FoF simulation.

There will need to be a new action taken when the attack is detected—the same for all actions that are initiated when there is an attack detected. However, the support pieces need to be added first. The following subsections go over the support pieces, and the last subsection shows how to add the new action.

A-5.2.1 Get FoF Result Data Piece

The time for the adversary entering the building is needed. First make sure the FoF model outputs the time for the adversary entering the building in the results; if not, the FoF modeling team should add it and provide the name. Next add a new document link variable that links to that result output time, as shown in Figure A-48. Make sure the name for the result item, in the "Var Link" field as highlighted in Figure A-48, is the same as what is in the FoF results file.

The model was designed not to use the FoF data directly as it could have different time types or may not exist at all; instead, the data is converted into a corresponding variable, which is used for model evaluations. To do this, another variable is created typically with the same name but without the prefix of "SimjIn_" and with a default value of a very large number so the EMRALD simulation time never reaches the representative time if it not read from the FoF data. Finally, a new immediate action needs to be created and added to the "Process_Simanij_Run" that takes the "SimjIn_RB_Door_HitTime" variable value and converts it into hours. This is explained in more detail in Section A-3.1 and shown in Figure A-8.



Create Variable

Type: Double

Name: SimjIn_RB_Door_HitTime

Description

Scope: Document Link

Doc Type: XML

Doc Path: ..\Simajin.results.xml

Use XPath Syntax for the Var Link. Tester

Var Link: //run[@run-id='%Int_ItemIdx']/data[@name='RB_Door_breach_time' or @name='sabotage_time_RB_Door']/@value

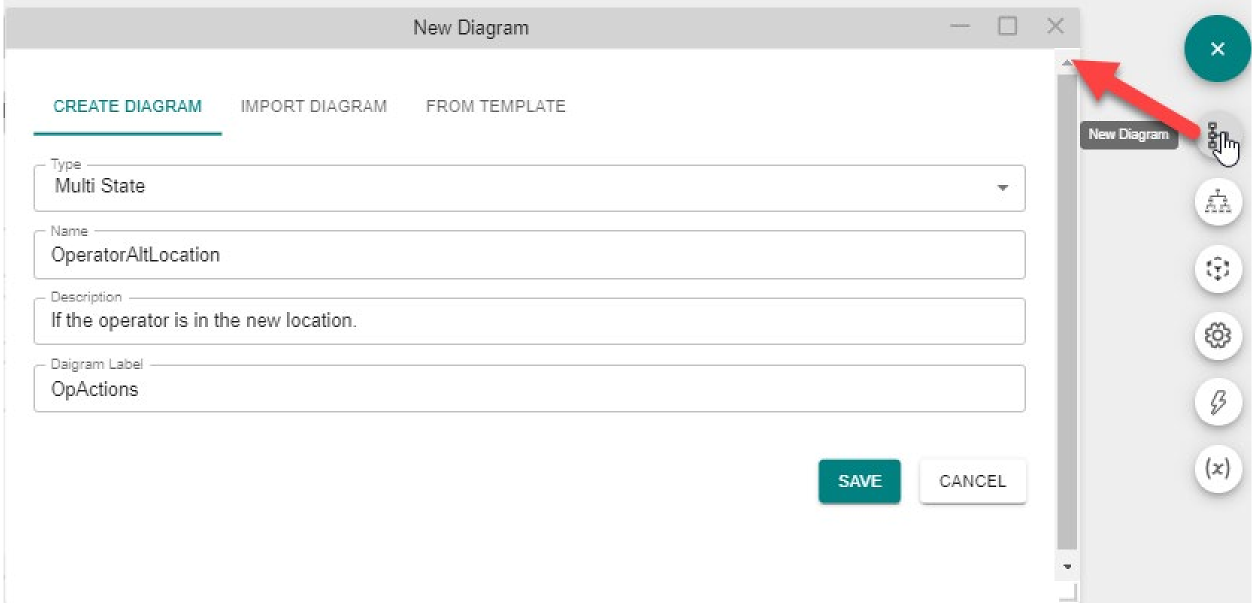
☐ Doc Path and Var Link must exist on startup

Default: 0

Figure A-48. New document link variable to get the FoF time.

A-5.2.2 New Operator Action Diagram

For this example, we are going to assume that to use the alternate control panels this task is complete. These criteria can be modeled in “AltPanel” and “AltPanel2” component in multiple ways such as a variable being assigned and evaluated or a diagram being in a specific state. This example will use the latter. This new diagram is needed to represent if the operator made it to the location in time. After bringing up the form to create a new diagram, the type could be “Multi State” or “Single State (Evaluation)” depending on how it is to be used. If there will be multiple branches of execution, then use the “Multi State;” if it will be used in a logic tree evaluation, then use the “Single State.” Although for this example multiple branches are not needed, “Multi State” is being used because the diagram will not be used in a logic tree, and then, we do not have to assign values for the states.


New Diagram

Type: Multi State

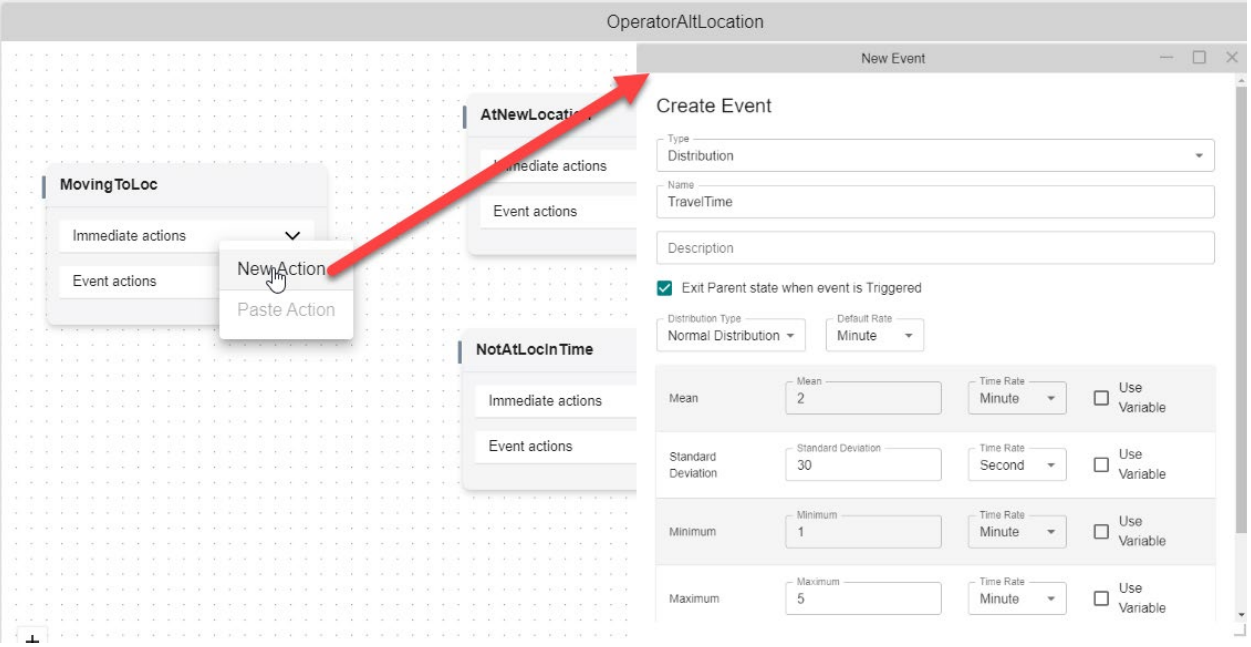
Name: OperatorAltLocation

Description: If the operator is in the new location.

Daigram Label: OpActions

Figure A-49. New diagram to capture operator other location condition.

Next, states can be added to the “OperatorAltLocation” diagram. Add the first state and name it “MovingToLoc” with a type of “Standard.” It will be entered from an action when the attack is detected. Then add two other states called “AtNewLocation” and “NotAtLocInTime” and a new distribution event to the “MovingToLoc” with the statistical parameters for how long it will take the operator to get to the location as shown in Figure A-50. Minimum is the fastest someone could possibly do it, and maximum is the slowest. Add a transition action to this event to go to “AtNewLocation.”



Create Event

Type: Distribution

Name: TravelTime

Description

☑ Exit Parent state when event is Triggered

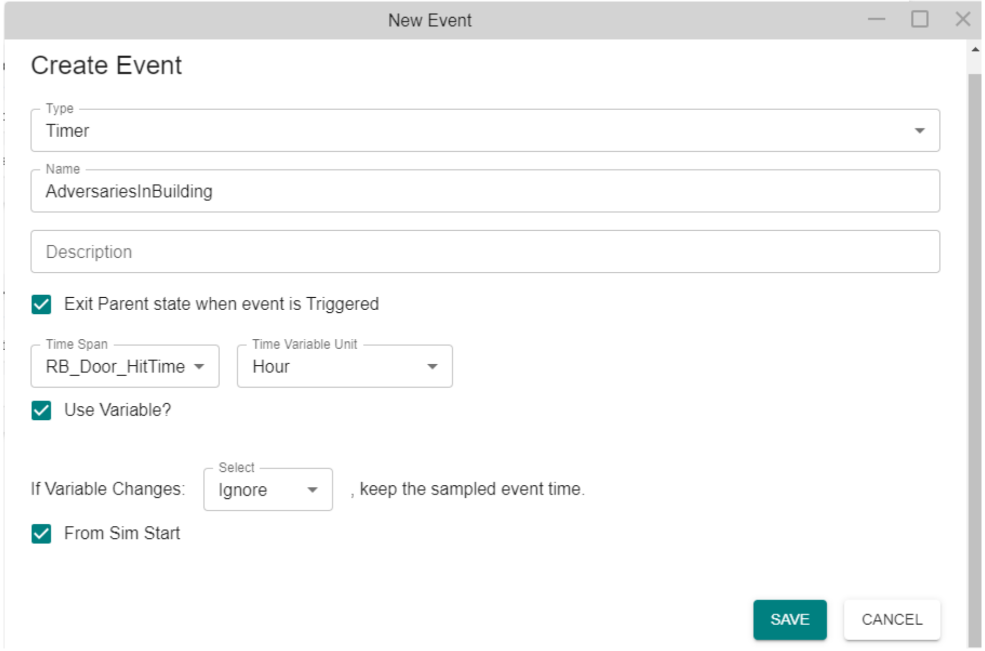
Distribution Type: Normal Distribution

Default Rate: Minute

Mean	Mean: 2	Time Rate: Minute	□ Use Variable
Standard Deviation	Standard Deviation: 30	Time Rate: Second	□ Use Variable
Minimum	Minimum: 1	Time Rate: Minute	□ Use Variable
Maximum	Maximum: 5	Time Rate: Minute	□ Use Variable

Figure A-50. New event for sampling the time of operator to get to the location.

Create another event under “MoveToLoc” called AdversariesInBuilding” to be triggered according to the time from the FoF result data using the “RB_Door_HitTime” variable as shown in Figure A-51. Add a transition action under this event to go to “NotAtLocInTime.” Now when the “MoveToLoc” state is entered, it is a race between the two events and will put this diagram in one of the two states “AtNewLocation” or “NotAtLocInTime” as shown in Figure A-52.



Create Event

Type: Timer

Name: AdversariesInBuilding

Description

☑ Exit Parent state when event is Triggered

Time Span: RB_Door_HitTime

Time Variable Unit: Hour

☑ Use Variable?

If Variable Changes: Select: Ignore , keep the sampled event time.

☑ From Sim Start

Figure A-51. A timer event linked to the variable for when the adversaries get into the building.

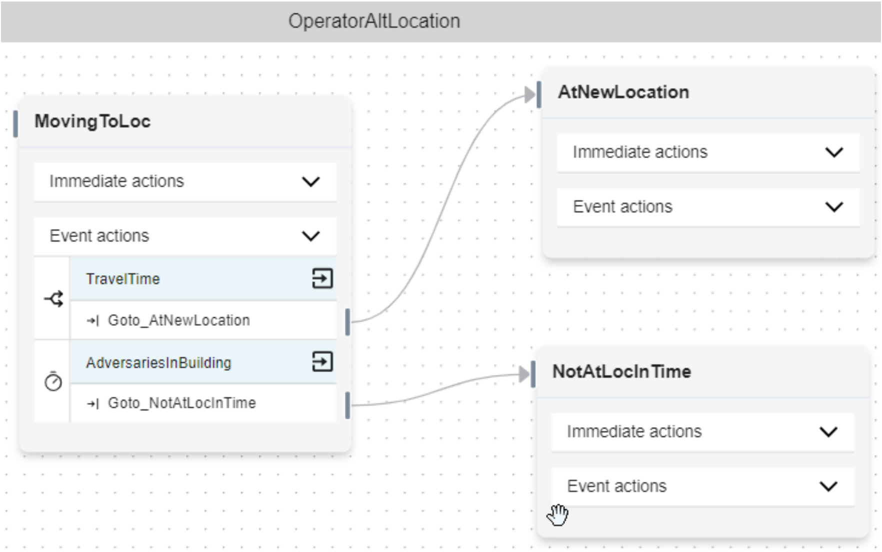
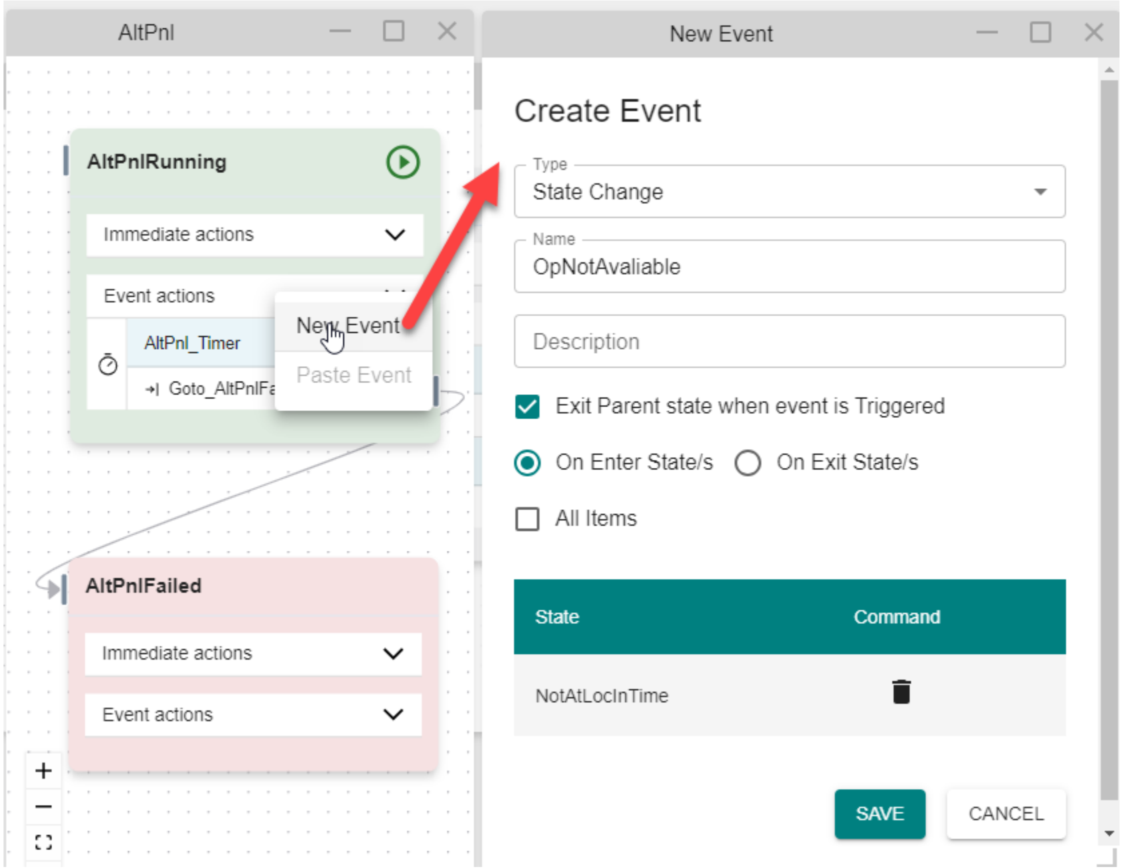


Figure A-52. Finished “OperatorAltLocation” diagram.

A-5.2.3 Evaluate Operator Condition

The state of the operator determined by the “OperatorAtLocation” diagram, can be used by the “AltPanel” and AltPanel2” component diagrams. To do this, add a new “State Change” event to the “AltPnlRunning” state called “OpNotAvaliable.” It should exit the state when triggered and trigger when the “NotAtLocInTime” state is entered. Then add the “Goto_AltPnlFailed” to the event by copying and pasting it from the other event. Now the alternate panel can be disabled by the adversary by either hitting the panel directly or getting into the building before the operator can get to the specified location. Do the same thing for the “AltPnl2” diagram by copying the “OpNotAvaliable” event and pasting it in the “AltPnl2Running” state along with the correct action to move to the failed state.



Create Event

Type: State Change

Name: OpNotAvaliable

Description

☑ Exit Parent state when event is Triggered

● On Enter State/s ○ On Exit State/s

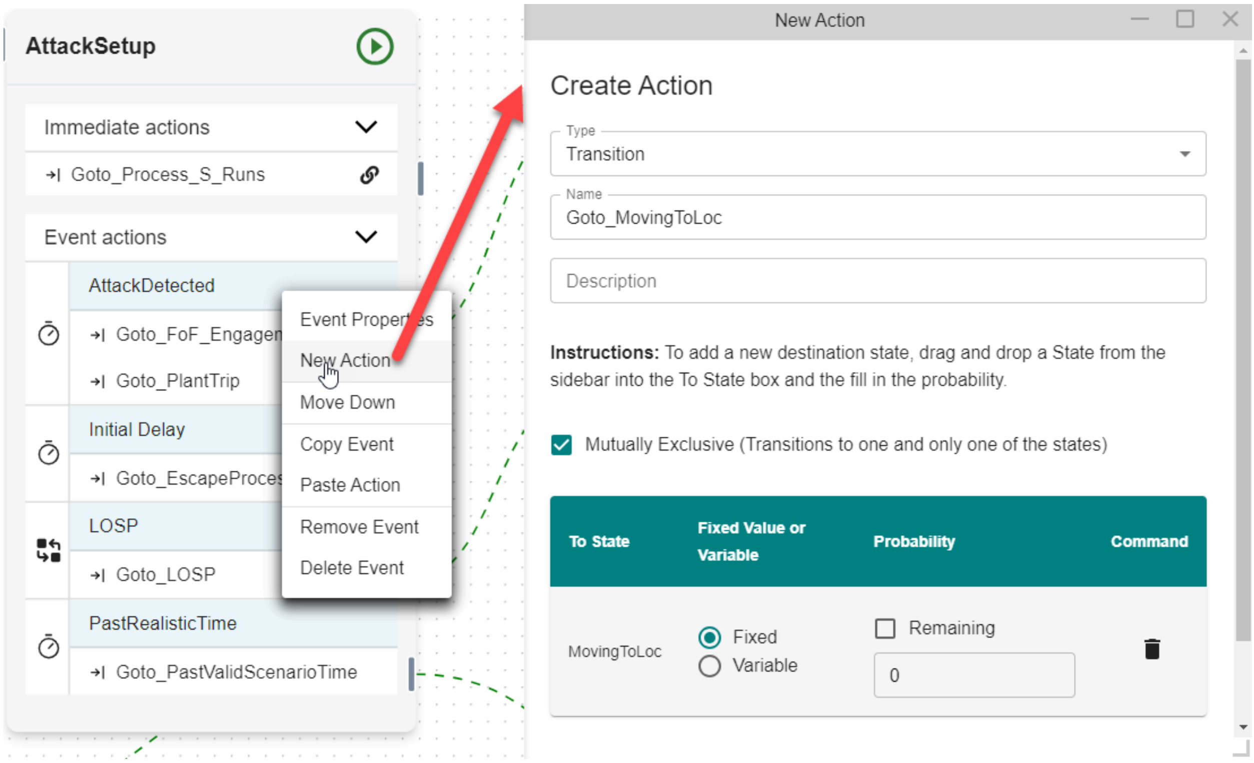
□ All Items

State	Command
NotAtLocInTime	

Figure A-53. An event to trigger if the operator is not available.

A-5.2.4 Initiate Operator Move

With the other pieces of the model done, the final step is to trigger the operator action to send the operator when there is an attack detected. To do this, a new transition action in the “AttackDetected” event under the “AttackSetup” state in the “Initiate_Attack” diagram needs to be created as shown in Figure A-54. All this action does is start the evaluation of the “MovingToLoc” in the “OperatorAltLocation” diagram.



Create Action

Type: Transition

Name: Goto_MovingToLoc

Description

Instructions: To add a new destination state, drag and drop a State from the sidebar into the To State box and the fill in the probability.

☑ Mutually Exclusive (Transitions to one and only one of the states)

To State	Fixed Value or Variable	Probability	Command
MovingToLoc	● Fixed ○ Variable	□ Remaining 0	

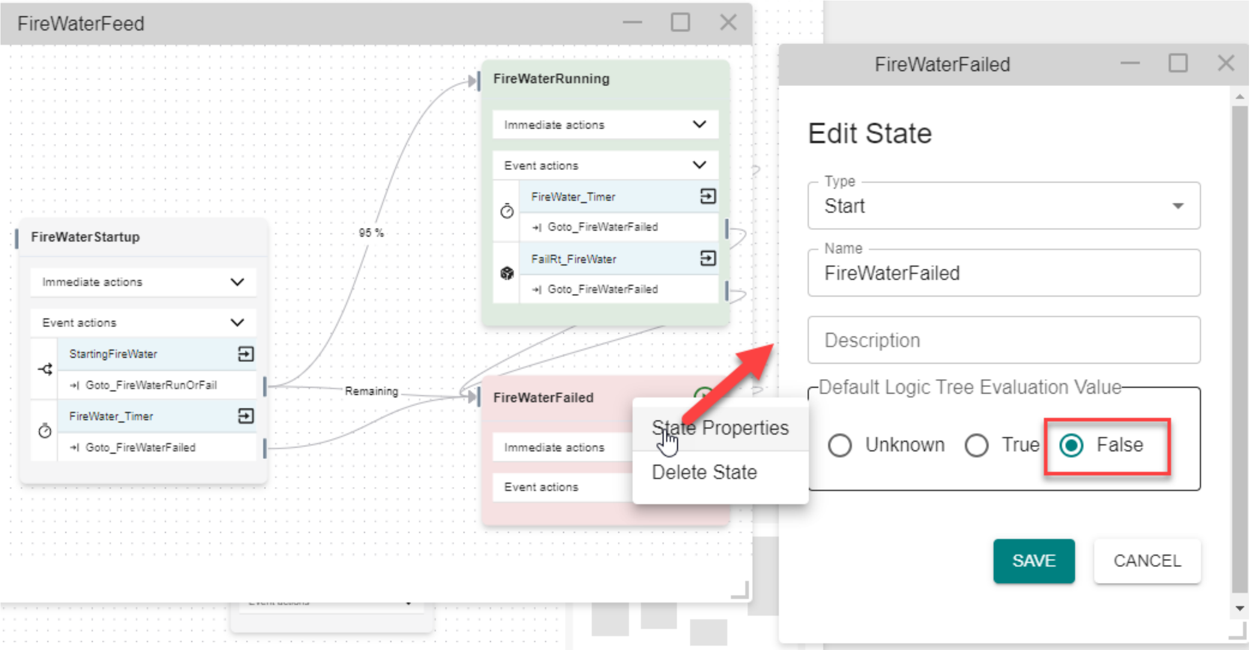
Figure A-54. Action to start the operator going to the new location.

A-5.3 Removing/Enabling after Attack Prevention Options

The generic model has several prevention options that may not be suitable to use for some facilities due to design, location, or unresolved regulatory concerns. Each of the alternative options has a component diagram that can be modified so that it starts in a failed state to disable that option. Also verify that the desired prevention options are enabled by checking the same diagrams. To do this, open the component diagram for the desired prevention option from the list below:

- Manual TDP operation – Diagram “ManualTDP_Tools”
- Fire water cooling – Diagram “FireWaterSupply” and/or “FireWaterFeed”
- Protection pump cooling – Diagram “ProtectionPump”
- Fire Truck pump cooling – Diagram “FireTruckPump”
- FLEX pump cooling – Diagram “FlexPump.”

If you want to disable the option, open the properties for the standby or normal startup state and change the type to “Standard” and then open the properties for the failed state for the component and change the type to “Start” as shown in Figure A-55. This will make the component start in the failed state and not available. To enable, do the opposite.



Edit State

Type: Start

Name: FireWaterFailed

Description

Default Logic Tree Evaluation Value: ○ Unknown ○ True ● False

Figure A-55. Editing the state properties to make the failed state the starting spot.

A-5.3.1 Removing Components

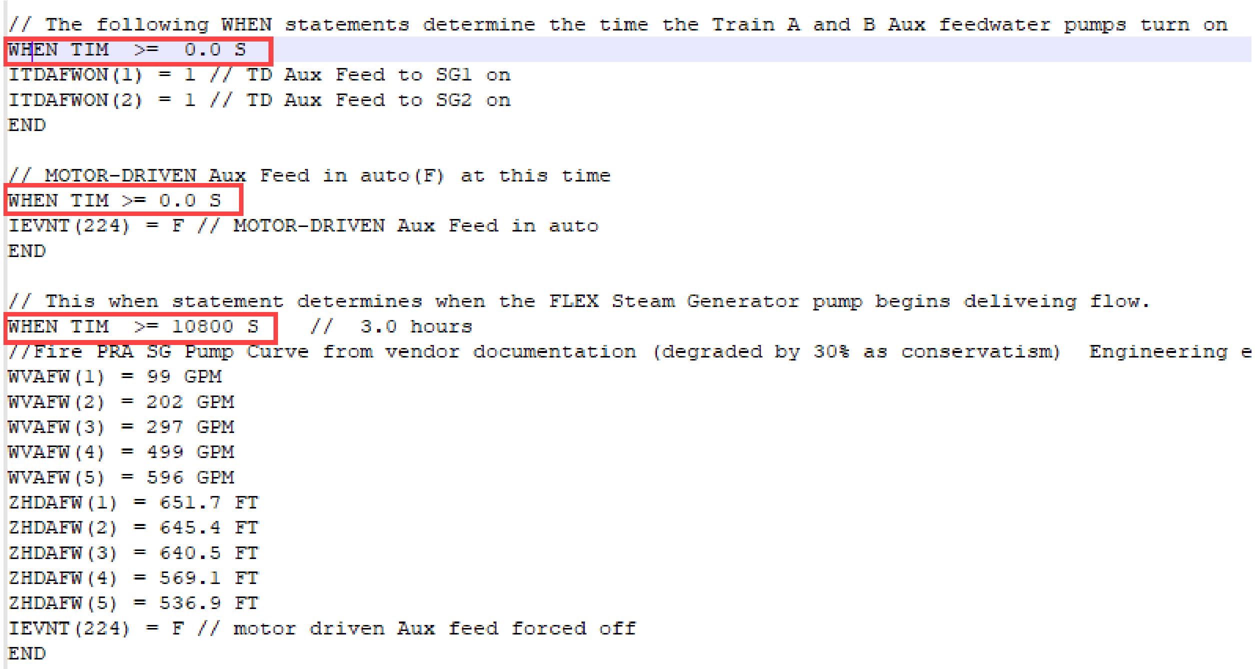
To remove components so that it matches a facility, the user can simply right-click on the component in the left side list and select delete. To keep the model clean, if there is a hit time variable used, then the FoF variable and action loading that data should also be removed

A-6. Setting Up Thermal Hydraulics Analysis

Final determination of core damage comes from the thermal hydraulics analysis. EMRALD has a form that allows for a simplified connection to MAAP. The EMRALD modeler can import MAAP files and link parts of the MAAP input to variables from EMRALD. When a simulation reaches the point to run MAAP, it dynamically adjusts the MAAP file to reflect the timing of all the events from that scenario and gets a result from MAAP to determine if there was possible core damage.

A-6.1 MAAP Files

Linking to MAAP requires the user to have MAAP 5.04 installed on the machine, a .par file that is the plant model and an .inp file created by the facilities MAAP expert. The .inp or input file is the scenario to be ran on the main plant model. The plant MAAP expert needs to create an input file that includes options to turn off the target components and the features to simulate the alternative preventative strategies. For example, the input file will be initialized to trip the reactor from full power. Then there are “WHEN TIM” clauses set up for times when components such as pumps will fail. For a PWR, there needs to be conditions on filling and stop filling times. There also needs to be comments on the blocks that should be adjusted for the dynamic analysis so the EMRALD modeler knows what variables to use at different locations as shown in Figure A-56.

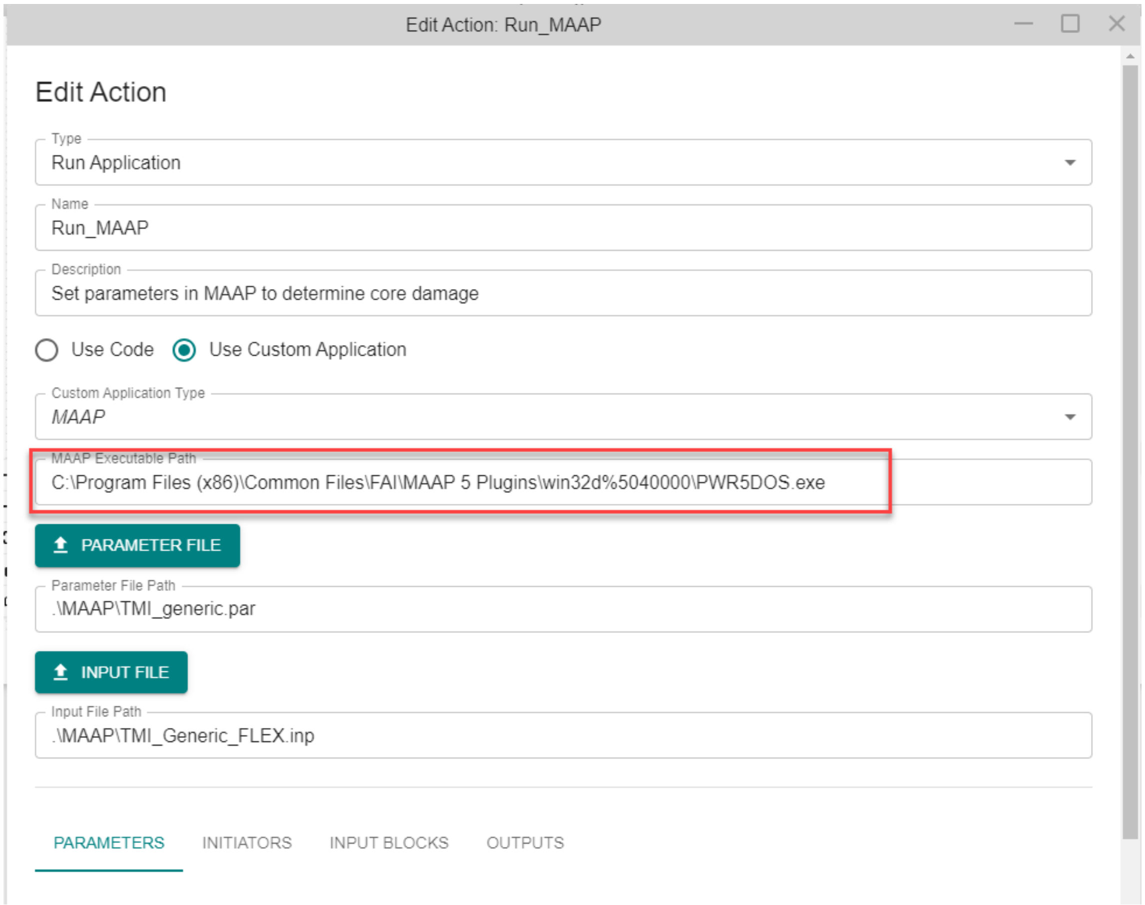


```
// The following WHEN statements determine the time the Train A and B Aux feedwater pumps turn on
WHEN TIM  >=  0.0 S
ITDAFWON(1) = 1 // TD Aux Feed to SG1 on
ITDAFWON(2) = 1 // TD Aux Feed to SG2 on
END

// MOTOR-DRIVEN Aux Feed in auto(F) at this time
WHEN TIM >= 0.0 S
IEVNT(224) = F // MOTOR-DRIVEN Aux Feed in auto
END

// This when statement determines when the FLEX Steam Generator pump begins deliveing flow.
WHEN TIM  >= 10800 S    //  3.0 hours
//Fire PRA SG Pump Curve from vendor documentation (degraded by 30% as conservatism)  Engineering e
WVAFW(1) = 99 GPM
WVAFW(2) = 202 GPM
WVAFW(3) = 297 GPM
WVAFW(4) = 499 GPM
WVAFW(5) = 596 GPM
ZHDAFW(1) = 651.7 FT
ZHDAFW(2) = 645.4 FT
ZHDAFW(3) = 640.5 FT
ZHDAFW(4) = 569.1 FT
ZHDAFW(5) = 536.9 FT
IEVNT(224) = F // motor driven Aux feed forced off
END
```

Figure A-56. Example of MAAP input file with highlighted items that need to be set by EMRALD variables.

A-6.2 Linking to the EMRALD Model

The EMRALD model is linked to EMRALD using a “Run Application” action and the MAAP “Custom Application” selection as shown in Figure A-57. The user must verify that “MAAP Execution Path” is the location of the “PWRSDOS.exe” and then load the parameter or “.pam” file for the plant’s MAAP model. The model’s name is automatically added to the “Parameter File Path” field, but this must be set to either the fixed location of the parameter file or path relative to the EMRALD model being run; refer to Section A-7.2 for recommended file structuring. Next load the input file or MAAP “.inp” file provided by the thermal hydraulics expert and assign the “Input File Path” parameter location. Once these are both loaded, the parameters, initiators, input blocks, and outputs tabs are loaded with the information from the MAAP files ready to be linked to the EMRALD model.



Edit Action: Run_MAAP

Edit Action

Type: Run Application

Name: Run_MAAP

Description: Set parameters in MAAP to determine core damage

○ Use Code ● Use Custom Application

Custom Application Type: *MAAP*

MAAP Executable Path: C:\Program Files (x86)\Common Files\FAI\MAAP 5 Plugins\win32d%5040000\PWR5DOS.exe

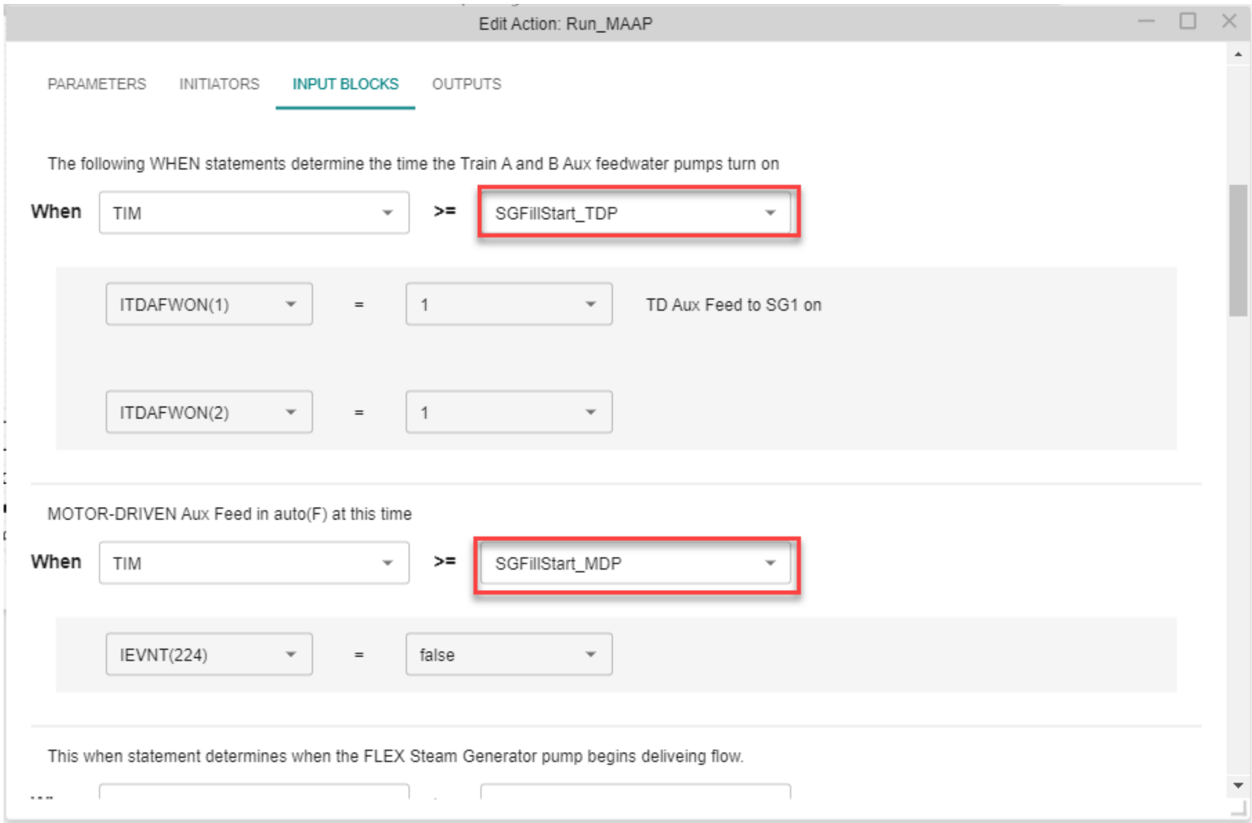
Parameter File Path: .\MAAP\TMI_generic.par

Input File Path: .\MAAP\TMI_Generic_FLEX.inp

Figure A-57. MAAP custom form for loading and linking MAAP model to EMRALD.

The generic EMRALD model assumes that the attack scenarios will call for a manual TRIP of the reactor on attack detection. This assumption means that the analysis parameters and initiators do not need to be adjusted from EMRALD. Consult with the facility MAAP expert to determine if any parameters or initiators need to be dynamically adjusted for the MAAP execution.

The input blocks tab is where the “When” conditions are tied to the EMRALD blocks. Typically, the EMRALD variables are just used in place of the time conditions so that the time is adjusted to when it occurs in the EMRALD simulation. This is shown in Figure A-58, where the red boxes highlight the “SGFillStart_TDP” and “SGFillStart_MDP” variables used to set when the steam generators start being filled by the operator actions after an attack is detected for the generic PWR model.



Edit Action: Run_MAAP

PARAMETERS INITIATORS INPUT BLOCKS OUTPUTS

The following WHEN statements determine the time the Train A and B Aux feedwater pumps turn on

When TIM >= SGFillStart_TDP

ITDAFWON(1) = 1 TD Aux Feed to SG1 on

ITDAFWON(2) = 1

MOTOR-DRIVEN Aux Feed in auto(F) at this time

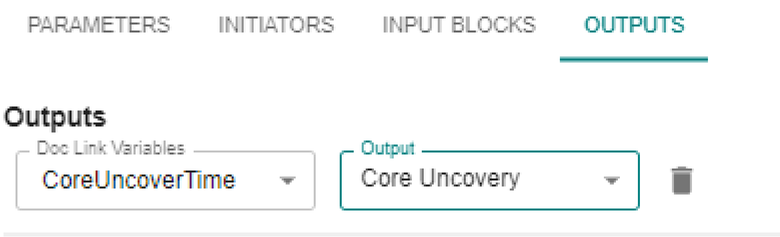
When TIM >= SGFillStart_MDP

IEVNT(224) = false

This when statement determines when the FLEX Steam Generator pump begins deliveing flow.

Figure A-58. Input blocks section for linking MAAP to EMRALD and assigning EMRALD variables to the MAAP conditions.

Getting the results of the MAAP file is set up in the outputs tab as shown in Figure A-59. The EMRALD variable “CoreUncoverTime” is set to the “Core Uncovery” time from the MAAP results. While core uncover does not necessarily mean there is core damage, this a conservative assumption and is easily obtained from the MAAP results. The value of “CoreUncoverTime” is compared to the RAPT time to determine if they can keep the plant safe for the required time limit.



PARAMETERS INITIATORS INPUT BLOCKS OUTPUTS

Outputs

Doc Link Variables: CoreUncoverTime

Output: Core Uncovery

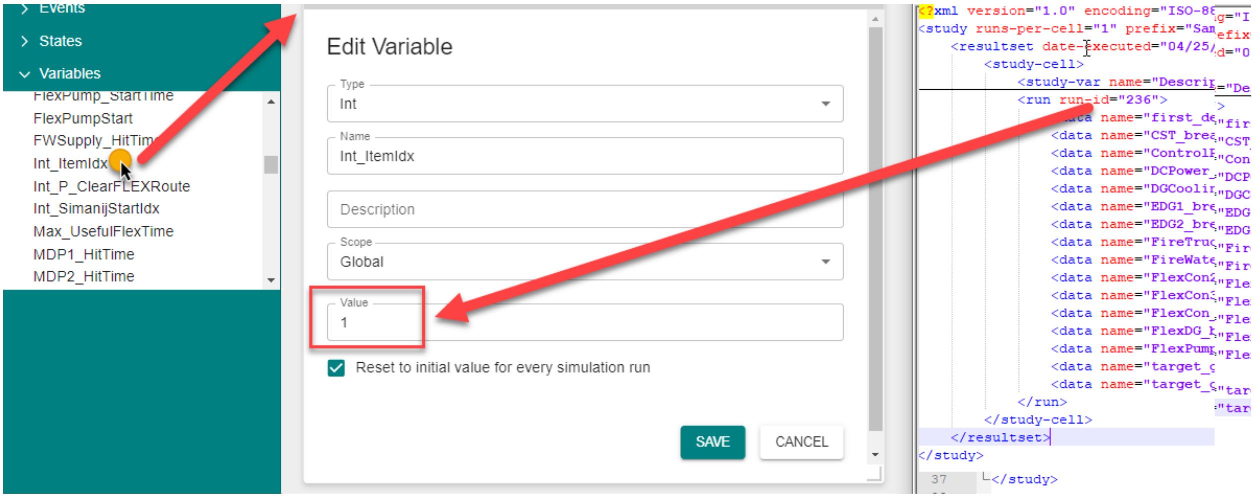
Figure A-59. The EMRALD variable “CoreUncoverTime” is set to the “Core Uncovery” time from the MAAP results.

A-7. Running the Model

To run the EMRALD model, some parameters must be adjusted, and the model files need to be placed in the correct location or changed in the model. This section goes over how to set up the model to run and how to debug when there are issues with the model.

A-7.1 FoF Start Index

Each EMRALD run uses one result set from the FoF simulation. The FoF data from Simajin typically does not start with a “1,” and so the EMRALD item index must be set to the starting index for the Simajin results. To do this, edit the variable “Int_ItemIdx” in the UI and change the value of first “run-id” item in the results XML file as shown in Figure A-60. Alternatively, the saved model can be altered by opening the EMRALD model in a text editor and searching for the “Int_ItemIdx” variable and changing the value then saving.



Edit Variable

Type: Int

Name: Int_ItemIdx

Description

Scope: Global

Value: 1

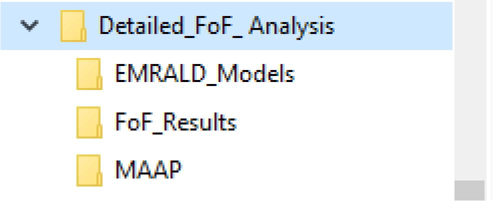
☑ Reset to initial value for every simulation run

```
<?xml version="1.0" encoding="ISO-8
<study runs-per-cell="1" prefix="Sam
    <resultset date-executed="04/25/
        <study-cell>
            <study-var name="Descrip
            <run run-id="236">
                     name="first_de
                <data name="CST_brea
                <data name="ControlR
                <data name="DCPower_
                <data name="DGCoolin
                <data name="EDG1_bre
                <data name="EDG2_bre
                <data name="FireTruc
                <data name="FireWate
                <data name="FlexCon2
                <data name="FlexCon3
                <data name="FlexCon_
                <data name="FlexDG_b
                <data name="FlexPump
                <data name="target_c
                <data name="target_c
            </run>
        </study-cell>
    </resultset>
</study>
37 └</study>
```

Figure A-60. Setting the Int_ItemIdx to the Simajin result file starting number.

A-7.2 File Locations

It is recommended to create a folder on the computer for all the EMRALD FoF analysis. In this example, it is called “Detailed_FoF_Analysis.” Then create a folder under that for all the variations of the EMRALD models to be used (e.g., “EMRALD_Models”). Add another folder named **“MAAP”** for the thermal hydraulics models. **Important:** if you do not call it the “MAAP” folder, then you will need to adjust the relative path for the “RunMAAP” action shown in Section A-6. Create another folder called something like “FoF_Results” for the FoF simulation result files. An example is shown in Figure A-61



Detailed_FoF_ Analysis
- EMRALD_Models
- FoF_Results
- MAAP

Figure A-61. File structure setup.

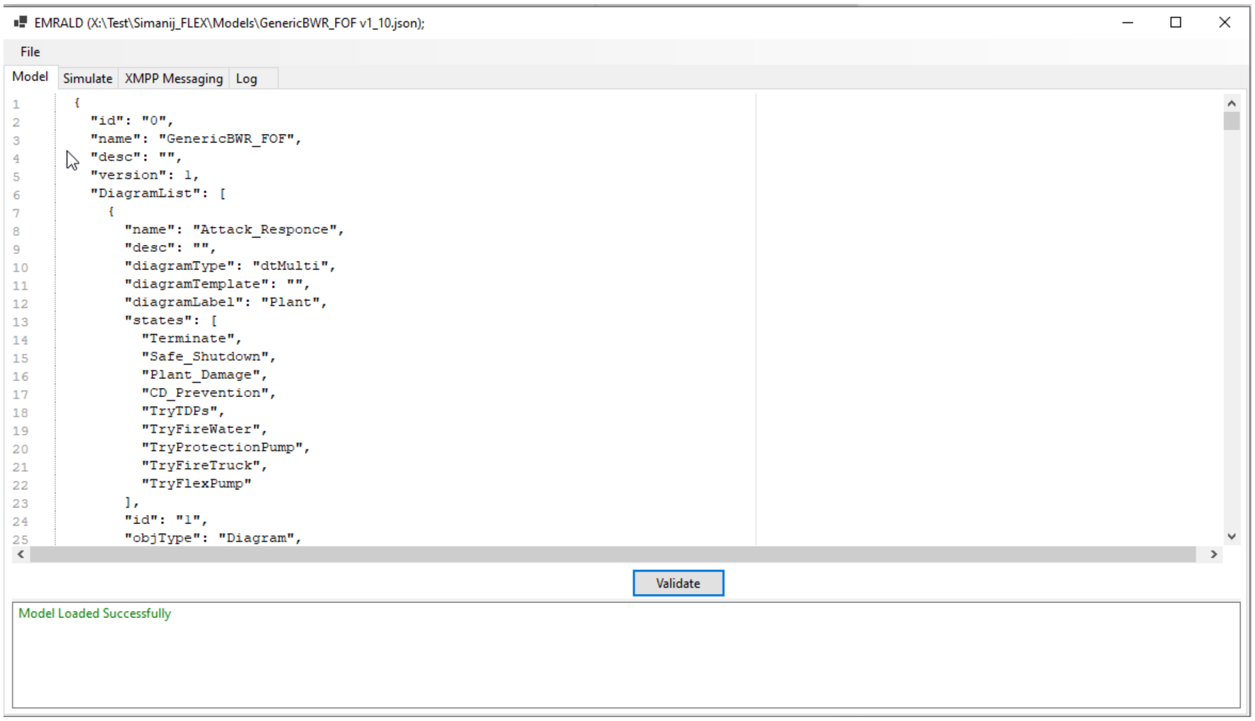
For Simajin results, create a file called “Simajin.results.xml” and copy the results that you want to run into that file if you want it in a different location then update all the “Doc Path” parameters for all the “SijnIn_” variables.

A-7.3 MAAP Install

MAAP 5.04 must be installed. If MAAP is not installed to C:\Program Files (x86)\FAI\MAAP 5.04, then the executable location property must be updated for the “Run_MAAP” immediate action in the “RunTH” state of “EvalPumpTiming” diagram as highlighted in Figure A-57.

A-7.4 Running the EMRALD Simulation

To run the model, first download the EMRALD solve engine from the downloads menu on the EMRALD website. Unzip to a desired location. Locate the “EMRALD_Sim.exe” file and execute it. You may get a notification asking if you want to trust the application. After clearing with your IT department, click yes. After the application opens, select menu File-Open and load the desired EMRALD model in the “EMRALD_Models” folder. This will open the model and check for any syntax errors. If there are errors, close the model, fix them with the web UI, save, and reopen. The model should be loaded as a text file with text at the bottom saying “Model Loaded successfully” as shown in Figure A-62.

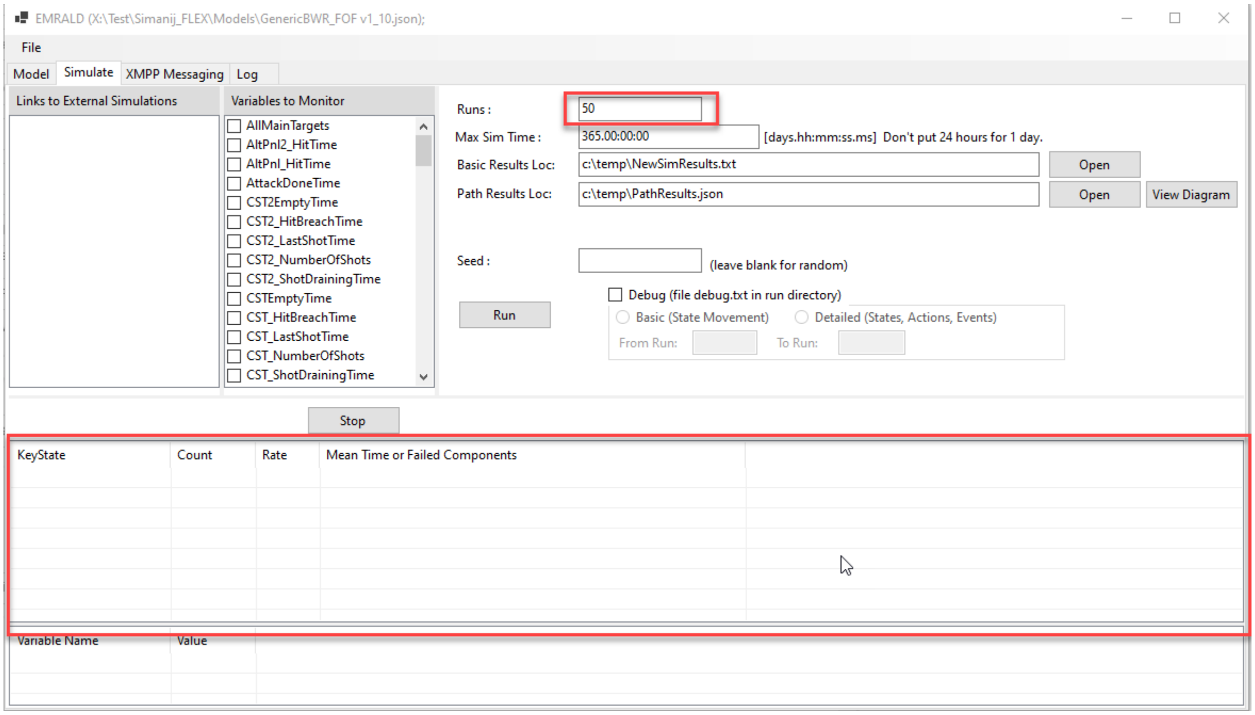

EMRALD (X:\Test\Simanij_FLEX\Models\GenericBWR_FOF v1_10.json);
File
Model
Simulate
XMPP Messaging
Log
Validate

```
{
  "id": "0",
  "name": "GenericBWR_FOF",
  "desc": "",
  "version": 1,
  "DiagramList": [
    {
      "name": "Attack_Responce",
      "desc": "",
      "diagramType": "dtMulti",
      "diagramTemplate": "",
      "diagramLabel": "Plant",
      "states": [
        "Terminate",
        "Safe_Shutdown",
        "Plant_Damage",
        "CD_Prevention",
        "TryTDPs",
        "TryFireWater",
        "TryProtectionPump",
        "TryFireTruck",
        "TryFlexPump"
      ],
      "id": "1",
      "objType": "Diagram",
```

Model Loaded Successfully

Figure A-62. EMRALD user interface after successfully opening the model.

Once it is loaded, the simulations can be run from the simulate tab. In this tab, the user can select variables to watch and output to the results, assign the number of runs, and set up where to save the results. When doing an analysis, the number of runs needs to match the number of results in the FoF results file. While the simulation is running, a tally of the results will be displayed in the bottom section of the form, outlined in red on Figure A-63.


EMRALD (X:\Test\Simanij_FLEX\Models\GenericBWR_FOF v1_10.json);
File
Model
Simulate
XMPP Messaging
Log
Links to External Simulations
Variables to Monitor
Open
Open
View Diagram
Run
Stop

□ AllMainTargets
□ AltPnl2_HitTime
□ AltPnl_HitTime
□ AttackDoneTime
□ CST2EmptyTime
□ CST2_HitBreachTime
□ CST2_LastShotTime
□ CST2_NumberOfShots
□ CST2_ShotDrainingTime
□ CSTEmptyTime
□ CST_HitBreachTime
□ CST_LastShotTime
□ CST_NumberOfShots
□ CST_ShotDrainingTime

Runs : 50

Max Sim Time : 365.00:00:00 [days.hh:mm:ss.ms] Don't put 24 hours for 1 day.

Basic Results Loc: c:\temp\NewSimResults.txt

Path Results Loc: c:\temp\PathResults.json

Seed : (leave blank for random)

□ Debug (file debug.txt in run directory)

○ Basic (State Movement) ○ Detailed (States, Actions, Events)

From Run: To Run:

KeyState	Count	Rate	Mean Time or Failed Components

Variable Name	Value

Figure A-63. EMRALD solve engine settings for simulating the model.

A-7.5 Result Files and Visuals

A text-based basic result file is saved to the path specified in the “Basic Results Loc” when running the simulations. These results give a percentage of how many times the simulation ended in a key state vs. the total number of runs, along with the values of any monitoring variables for each key state run.

The “Path Results” file is a JSON results file that can easily be read programmatically if desired. It provides the paths, timing, and statistics for each state leading to the key end state. The web user interface can load this file and display the paths and statistical data as shown in Figure A-64. To open the file, click on the “View Diagram” button after the simulations are done, or from the EMRALD website, select “Project ->Load Results.” This can be useful in debugging or showing main how the simulation ends up in the key state. This can be helpful in determining where better data could help improve results.

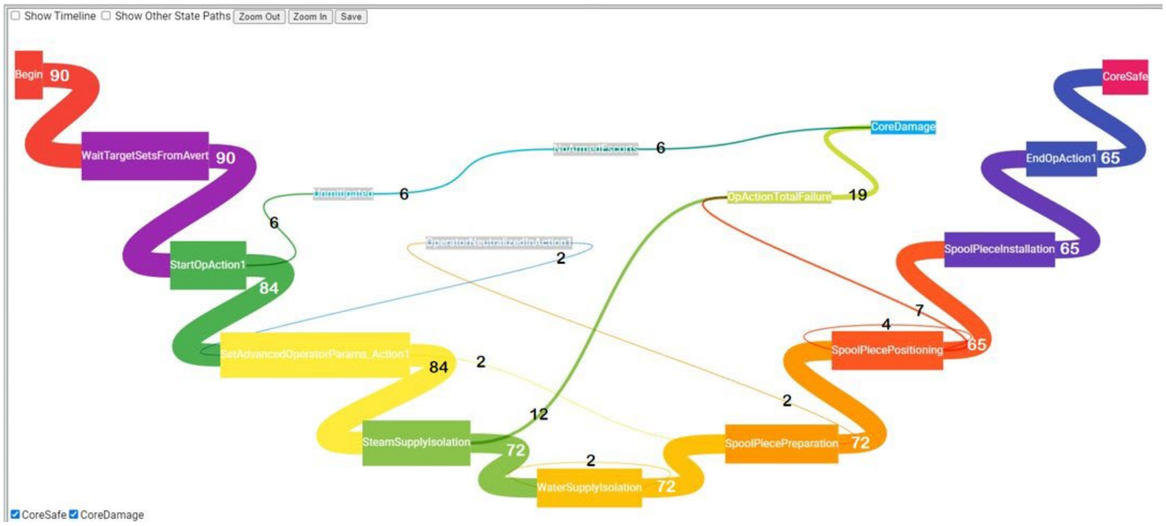
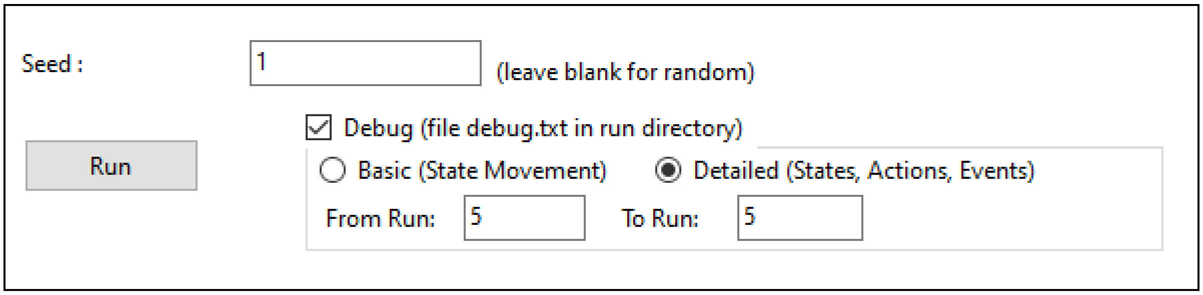


Figure A-64. Example of a “Path Results” file showing the states and events lead to the key states.

A-7.6 Debugging the Model

When the model runs but the results are not coming out as expected for one or more of the simulation runs, there are tools to help debug the model. It is recommended to always use a seed first when running the model and can always be used if desired; however, if multiple batches are run on different machines, do not set the seed, so that it is different for each batch. If an issue is noticed for a simulation run, a debug file can be generated for that run. This is done by selecting the debug check box and assigning the “From Run” and “To Run” to the run that needs to be debugged, as shown in Figure A-65.



Seed : 1 (leave blank for random)

Run

☑ Debug (file debug.txt in run directory)

○ Basic (State Movement) ● Detailed (States, Actions, Events)

From Run: 5 To Run: 5

Figure A-65. Debug settings for EMRALD simulating runs.

The debug file is saved in executable directory for the EMRALD simulation engine exe called “debug.txt.” This file shows the state transitioning, events, actions, and variable assignments in the time order in which thy occur. This is a great to see if the model behaved as expected and does not find any deviations.