INL REPORT

INL/EXT-xxxxxxx
Limited Release
Printed 07/30/2013

RAVEN: Dynamic Event Tree Approach

Andrea Alfonsi, Cristian Rabiti, Diego Mandelli, Joshua Cogliati, Robert Kinoshita

Prepared by
Idhao National Laboratory
Idaho Falls, Idaho 83415

The Idaho National Laboratory is a multiprogram laboratory operated by
Battelle Energy Alliance for the United States Department of Energy
under DOE Idaho Operations Office. Contract DE-AC07-05ID14517.

Issued by the Idaho National Laboratory, operated for the United States Department of Energy by Battelle Energy Alliance.

INL/EXT-xxxxxxx
Limited Release
Printed 07/30/2013

RAVEN: Dynamic Event Tree Approach

Andrea Alfonsi
Idaho National Laboratory
P.O. Box 1625
Idaho Falls, ID 83415-3840
andrea.alfonsi@inl.gov

Cristian Rabiti
Idaho National Laboratory
P.O. Box 1625
Idaho Falls, ID 83415-3840
cristian.rabiti@inl.gov

Diego Mandelli
Idaho National Laboratory
P.O. Box 1625
Idaho Falls, ID 83415-3840
diego.mandelli@inl.gov

Joshua Cogliati
Idaho National Laboratory
P.O. Box 1625
Idaho Falls, ID 83415-3840
joshua.cogliati@inl.gov

Robert Kinoshita
Idaho National Laboratory
P.O. Box 1625
Idaho Falls, ID 83415-3840
robert.kinoshita@inl.gov

Acknowledgment

This work is supported by the U.S. Department of Energy, under DOE Idaho Operations Office Contract DE-AC07-05ID14517. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

Contents

Figures

Tables

Summary

Conventional **E**vent-**T**ree (**ET**) based methodologies are extensively used as tools to perform reliability and safety assessment of complex and critical engineering systems. One of the disadvantages of these methods is that timing/sequencing of events and system dynamics is not explicitly accounted for in the analysis. In order to overcome these limitations several techniques, also know as **D**ynamic **P**robabilistic **R**isk **A**ssessment (**DPRA**), have been developed. **M**onte-**C**arlo (MC) and **D**ynamic **E**vent **T**ree (**DET**) are two of the most widely used D-PRA methodologies to perform safety assessment of **N**uclear **P**ower **P**lants (NPP). In the past two years, the Idaho National Laboratory (INL) has developed its own tool to perform Dynamic PRA: **RAVEN** (**R**eactor **A**nalysis and **V**irtual control **EN**vironment). RAVEN has been designed in a high modular and pluggable way in order to enable easy integration of different programming languages (i.e., C++, Python) and coupling with other application including the ones based on the MOOSE framework, developed by INL as well. RAVEN performs two main tasks: 1) control logic driver for the new Thermo-Hydraulic code RELAP-7 and 2) post-processing tool. In the first task, RAVEN acts as a deterministic controller in which the set of control logic laws (user defined) monitors the RELAP-7 simulation and controls the activation of specific systems. Moreover, RAVEN models also stochastic events, such as components failures, and performs uncertainty quantification. Such stochastic modeling is employed by using both MC and DET algorithms. In the second task, RAVEN processes the large amount of data generated by RELAP-7 using data-mining based algorithms. This report focuses on the first task and shows how it is possible to perform the analysis of dynamic stochastic systems using the newly developed RAVEN DET capability. As an example, the Dynamic PRA analysis, using Dynamic Event Tree, of a simplified pressurized water reactor for a Station Black-Out scenario is presented.

1 Introduction

RAVEN (**R**eactor **A**nalysis and **V**irtual control **EN**viroment) [**?**, **?**] is a software tool that acts as the control logic driver for the newly developed Thermal-Hydraulic code RELAP-7 (**R**eactor **E**xcursion and **L**eak **A**nalysis **P**rogram). The goal of this report is to highlight the newly developed Dynamic Event Tree module embedded in the code and its utilization in conjunction with RELAP-7.

As for all PRA software the capability to control the plant evolution during the simulation itself it is a plus as it is for the analysis of the propagation of the uncertainty. For these reasons, a strict interaction between RELAP-7 and RAVEN is a key of the long-term success of the overall project. In system safety analysis code a similar need is expressed by the implementation of the control logic of the plant. As a consequence the optimization of resources imposes the integration of this task under a common project that is naturally RAVEN. For this reason the control logic of the plant is simulated under RAVEN. This also offers the flexibility to easily implement proprietary control logic without any change in the RELAP-7 code that is also regarded as a facilitator factor of the quick deployment of RELAP-7.

In order to summarize what RAVEN is, it can be said that it is a multi-purpose **P**robabilistic **R**isk **A**ssessment (PRA) software framework that allows dispatching different functionalities. It is designed to derive and actuate the control logic required to simulate the plant control system and operator actions (guided procedures) and to perform both Monte-Carlo sampling of random distributed events and Event Tree based analysis. In order to facilitate the input/output handling, a **G**raphical **U**ser **I**nterface (GUI) and a post-processing data mining module (under development) are available.

This report provides an overview of the DET structure, highlighting the mathematical framework from which its structure is derived and its software layout. In addiction it will be shown a **S**tation **B**lack **O**ut (SBO) DET based analysis of a simplified **P**ressurized **W**ater **R**eactor (PWR) model.

2 RAVEN Overview

2.1 Mathematical Framework

Let be $\bar{\theta}(t)$ a vector describing the plant status in the phase space; the dynamic of the plant, including the control system, can be summarized by the following equation [**?**]:

$$\frac{\partial \bar{\theta}}{\partial t} = \bar{H}(\theta(t), t) \tag{1}$$

In the above equation it is assumed the time differentiability in the phase space. Performing an arbitrary decomposition of the phase space, the following statement is obtained:

$$\bar{\theta} = \begin{pmatrix} \bar{x} \\ \bar{v} \end{pmatrix} \tag{2}$$

The decomposition is made in such a way that $\bar{x}$ represents the unknowns solved by RELAP-7, while $\bar{v}$ are the variables directly controlled by the control system (i.e., RAVEN). Equation **??** can now be rewritten as follows:

$$\begin{cases} \dfrac{\partial \bar{x}}{\partial t} = \bar{F}(\bar{x}, \bar{v}, t) \\ \dfrac{\partial \bar{v}}{\partial t} = \bar{V}(\bar{x}, \bar{v}, t) \end{cases} \tag{3}$$

It is possible to note that the function $\bar{V}(\bar{x}, \bar{v}, t)$ representing the control system, does not depend on the knowledge of the complete status of the system but on a restricted subset (i.e. control variables) $\bar{C}$:

$$\begin{cases} \dfrac{\partial \bar{x}}{\partial t} = \bar{F}(\bar{x}, \bar{v}, t) \\ \bar{C} = \bar{G}(\bar{x}, t) \\ \dfrac{\partial \bar{v}}{\partial t} = \bar{V}(\bar{x}, \bar{v}, t) \end{cases} \tag{4}$$

The system of equations in Eq. **??** is fully coupled and has commonly been solved by an operator splitting approach. The reasons for this choice are several:

- Control system reacts with an intrinsic delay
- The reaction of the control system might move the system between two different discrete states and therefore numerical errors will be always of first order unless the discontinuity is treated explicitly.

RAVEN uses this approach to solve Eq. **??** which now becomes:

$$\begin{cases} \dfrac{\partial \bar{x}}{\partial t} = \bar{F}(\bar{x}, \bar{v}_{t_{i-1}}, t) \\ \bar{C} = \bar{G}(\bar{x}, t) \\ \dfrac{\partial \bar{v}}{\partial t} = \bar{V}(\bar{x}, \bar{v}_{t_{i-1}}, t) \end{cases} \qquad t_{i-1} \leq t \leq t_i = t_{i-1} + \Delta t_i \tag{5}$$

Even if all information needed is contained in $\bar{x}$ and $\bar{v}$, it is not often practical and efficient to implement the control logic for complex system . Consequently, a system of auxiliary variables has been introduced.

The auxiliary variables are those that in statistical analysis are artificially added, when possible, to non-Markovian systems into the space phase to obtain a Markovian behavior back, so that only the information of the previous time step is needed to determine the future status of the system. Thus, the introduction of the auxiliary variables into the mathematical framework leads to the following formulation of Eq. **??**:

$$\begin{cases} \dfrac{\partial \bar{x}}{\partial t} = \bar{F}(\bar{x}, \bar{v}_{t_{i-1}}, t) \\ \bar{C} = \bar{G}(\bar{x}, t) \qquad\qquad t_{i-1} \leq t \leq t_i = t_{i-1} + \Delta t_i \\ \dfrac{\partial \bar{a}}{\partial t} = \bar{A}(\bar{x}, \bar{C}, \bar{a}_{t_{i-1}}, \bar{v}_{t_{i-1}}, t) \\ \dfrac{\partial \bar{v}}{\partial t} = \bar{V}(\bar{x}, \bar{C}, \bar{v}_{t_{i-1}}, \bar{a}, t) \end{cases} \tag{6}$$

2.2 Software Overview

RAVEN [**?**], is plugged with the software environment MOOSE [**?**]. MOOSE is a computer simulation framework, developed at Idaho National Laboratory (INL), that simplifies the process for predicting the behavior of complex systems and developing non-linear, multi-physics simulation tools. Other than providing the algorithms for the solution of the differential equation, MOOSE also provides all the manipulation tools for the `C++` classes containing the solution vector. This framework has been used to construct and develop the Thermal-Hydraulic code RELAP-7, giving an enormous flexibility in the coupling procedure with RAVEN.

RELAP-7 is the next generation nuclear reactor system safety analysis. It will become the main reactor systems simulation toolkit for RISMC (**R**isk **I**nformed **S**afety **M**argin **C**haracterization) [**?**] project and the next generation tool in the RELAP reactor safety/systems analysis application series.

RAVEN has been developed in a high modular and pluggable way in order to enable easy integration of different programming languages (i.e., `C++`, `Python`) and coupling with other applications including the ones based on MOOSE. The code consists of four main modules:

- RAVEN/RELAP-7 interface
- Python Control Logic
- External Python Manager
- Graphical User Interface

The RAVEN/RELAP-7 interface, coded in `C++`, is the container of all the tools needed to interact with RELAP-7/MOOSE. It has been designed in order to be general and pluggable with different solvers simulta- neously in order to allow an easier and faster development of the control logic/PRA capabilities for multi physics applications. The interface provides all the capabilities to generate the

monitored quantities and accordingly modify the controlled parameters in the RELAP-7/MOOSE calculation.

The control logic module is used to drive a RAVEN/RELAP-7 calculation. Up to now it is implemented by the user via `Python` scripting. The reason of this choice is to try to preserve generality of the approach in the initial phases of the project so that further specialization is possible and inexpensive. The implementation of the control logic via `Python` is rather convenient and flexible. The user only needs to know few `Python` syntax rules in order to build an input. Although this extreme simplicity, it will be part of the GUI task to automatize the construction of the control logic scripting in order to minimize user effort.

The core of PRA analysis is contained in an external driver/manager. It consists of a `Python` framework that contains the capabilities and interfaces to drive a PRA analysis. Its basic infrastructure in connection with the DET module will be discussed in section **??**

As previously mentioned, a GUI is not required to run RAVEN, but it represents an added value to the whole code. The GUI is compatible with all the capabilities actually available in RAVEN. Its development is performed using `QtPy`, which is a `Python` interface for a `C++` based library (`C++`) for GUI implementation. The GUI is based on a software named Peacock, which is a GUI interface for MOOSE based application and, in its base implementation, is only able to assist the user in the creation of the input. In order to make it fit all the RAVEN needs, the GUI has been specialized and is in continuous evolution.

3 The Dynamic Event Tree Methodology

Conventional ET based methodologies are extensively used as tools to perform reliability and safety assessment of complex and critical engineering systems. One of the disadvantages of these methods is that timing/sequencing of events and system dynamics is not explicitly accounted for in the analysis.

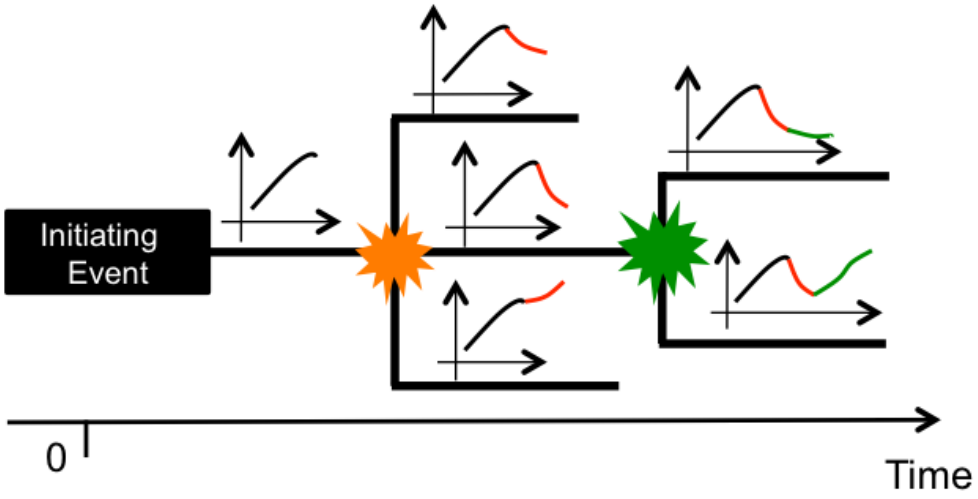


Figure 1: Dynamic Event Tree Conceptual Scheme.

In order to overcome these limitations a "dynamic" approach is needed. The DET technique brings several advantages [**?**], among which the fact that it simulates probabilistic system evolution in a way that is consistent with the severe accident model. In DET, event sequences are run simultaneously starting from a single initiating event. The branchings occur at user specified times and/or when an action is required by the operator and/or the system, creating a deterministic sequence of events based on the time of their occurrence (see Fig. **??**).

This leads to a more realistic and mechanistically consistent analysis of the system taken in consideration. The DPRA, in general, and the DET methodologies, in particular, are designed to take the timing of events explicitly into account which can become very important especially when uncertainties in complex phenomena are considered.

The main idea of this methodology is to let a system code (i.e., RELAP-7) determine the pathway of an accident scenario within a probabilistic "environment".
Figure **??** schematically shows the DET logic. As already mentioned, the accident sequence starts with an initiating event. Based on an user defined branching logic, driven by **P**robabilistic **D**istribution **F**unctions (PDFs), an event occurs at a certain time instant. The simulation spoons n different branches. In each of them, the branching event determines a different consequence (carrying on associated probabilities). Each sequence continues until another event occurs and a new set of branching is spooned. The simulation ends when an exit condition or a maximum mission time is reached.

4 ANalysis of Dynamic REactor Accident evolution Module

RAVEN is now able to perform DET analysis through the newly developed module "**AN**alysis of **D**ynamic **RE**actor **A**ccident evolution" (ANDREA), that has been included in the RAVEN external Python manager.

Following the philosophy of the DET approach, this module lets RELAP-7 find the path of an accident sequence, in a probabilistic point of view. When the system achieves certain conditions that would lead to alternative accident ways (i.e. an event occurs), a sampler generates a new set of branches (new possible scenarios), associating for each a conditional probability.

For the analysis of complex systems, the number of branches may become extremely big. In order to avoid unacceptable growth of problem due by an excessive number of branches, the user needs to specify an exit logic (termination laws), for example maximum mission time, rules based on the simulator physic model (i.e. Maximum temperature of the fuel cladding, etc.), and/or probabilistic thresholds. In other similar codes, one of the most common termination law is a probability cut-off: a branch execution is stopped when its probability falls below a given limit.

This approach may introduce a big impact on the probability of key events, if the user defined limit is not small enough that the influence on the final branch probability is not negligible. In order to avoid these issues and preserve the probability conservation, the user can not directly input, in RAVEN, a branching probability cut-off.

As can be inferred from above, RAVEN provides capabilities to:

- explore possible pathways through which the system can evolve
- quantify the probability of these scenarios

These main tasks are accomplished based on user specified branching and termination laws, model of the system in RELAP-7, probability assignment rules to accident sequences (either by inputed values or/and by distribution functions). The synergy among the different RAVEN modules gives to the package the flexibility to summarize all the state of art DET capabilities. Indeed, as in similar tools [**?**], the RAVEN/ANDREA allows the user use different approaches for defying the branching logic:

- branching/failures on demand (i.e. time and field triggers)
- branching based on failure probability distributions
- multi-branching scenarios

The list above, in conjunction with the brief overview of the mathematical framework analyzed in section **??**, gives an indication on how RAVEN (and the DET module) combines the capabilities present in other similar codes [**?**].

In RAVEN there is no distinction between active (e.g. circulation pumps, valves, controlled systems, etc.) and passive (e.g. steam generators, condensers, pipes, etc.) component behaviors; all are treated as a aleatory uncertainties, since the physical conditions under which a branching would occur for the them are determined by the thermal-hydraulic code RELAP-7 without user "control".

The user can model also the epistemic uncertainties, uncertainties due by lack of knowledge on the phenomena analyzed (e.g. Heat capacity coefficient, etc.).

In RAVEN, from an user point of view, either the aleatory and epistemic uncertainties are undifferentiated, since the branching laws are inputted in the same way. All triggers are identified by their own PDFs and activated when an user defined probability threshold, on the associated Cumulative Distribution Function (CDF), is overpassed. The probability thresholds on the CDFs are automatically handled by the DET module; the user only needs to specify them in the input.

The user implements the DET logic in the `Python` control logic input file, under the function named "*dynamic_event_tree*". In this function, all the features available in RAVEN can be used (online monitoring, controlling, auxiliary system, probability distributions, utilities, etc.).

```
def dynamic_event_tree(monitored, controlled, auxiliary):
  if distcont.checkCdf('CladFailureDist', monitored.avg_temp_clad_CH1):
    auxiliary.CladDamaged = True
    return
  if distcont.checkCdf('CladFailureDist', monitored.avg_temp_clad_CH2):
    auxiliary.CladDamaged = True
    return
  if distcont.checkCdf('CladFailureDist', monitored.avg_temp_clad_CH3):
    auxiliary.CladDamaged = True
    return
  if distcont.checkCdf('auxBackUpTimeDist', monitored.time - auxiliary.scram_start_time):
    auxiliary.AuxSystemUp = True
    return
  return
```

Figure 2: Example of DET logic

As an example, Figure **??** lists set of branching rules. The "checkCdf" function, available in the distribution container, checks if the associated threshold has been passed, and, in case, send a branching signal to RAVEN/RELAP-7 in order to print a restart file, that will be handled by the DET module.

4.1 Software Infrastructure

In this section the software infrastructure for the DET generation is briefly presented. As already mentioned, the DET module is part of RAVEN `Python` external driver, which represents the core of PRA analysis.

The external driver manager has the control of the different modules that support the DET calculation:

- General Calculation Driver and Job Handler
- Visualization and Input supporting GUI
- DET and Probability Engine
- Database manager
- Post-processing and Data Mining module

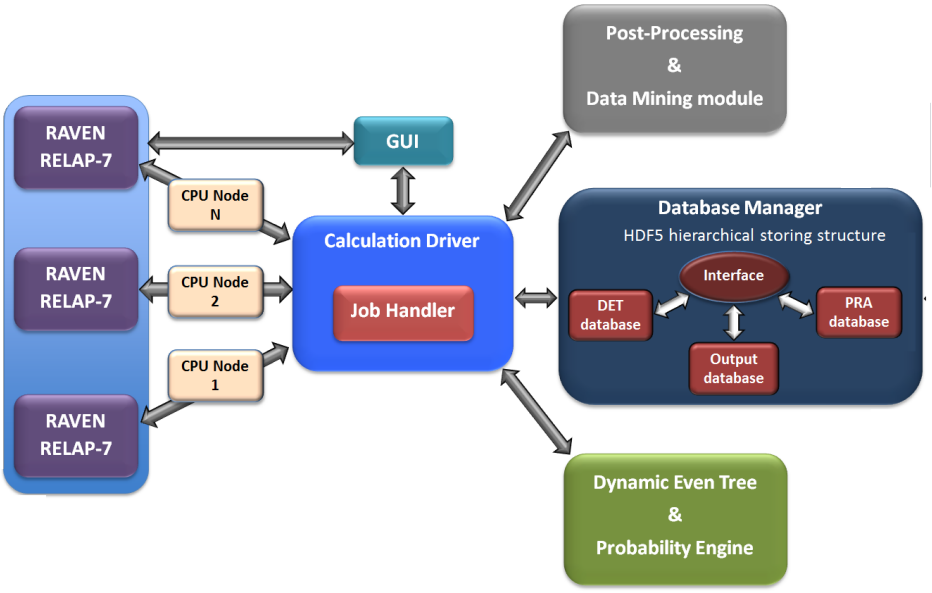


Figure 3: Software calculation infrastructure scheme

- Distribute Computing Environment Interface

The General Calculation Driver is in charge to communicate information among the different modules. It is the only software branch that is aware of which modules are participating to the calculation.

Figure **??** shows a schematic overview of the calculation structure, highlighting the most important modules that are involved in a DET calculation. Following an initiating event, the DET module provides initiating conditions as well as the duration of the simulation (optionally) to the system simulator (i.e. RAVEN/RELAP-7).
The Calculation Driver, through the Job Handler module, runs the simulator until a stopping condition is reached. The DET module decides whether to branch or not depending on the information received, through the general calculation driver, by RAVEN/RELAP-7. The probability engine is in charge of computing the likelihood of branching generated by trigger signals (e.g. probability threshold on Clad Failure distribution overpassed). If the DET module decides that n branches are needed, it communicates to the Calculation Driver/Job Handler, that manages the jobs through a queue system, to execute the n branches in n different subprocesses (in parallel, if the machine is multi-processors, in sequence otherwise). The resulting tree structure, branch probabilities, trigger information, and simulation results are sent to the Database manager, that, based on the kind of information received, distributes the data among the sub-structures (DET, PRA, Output databases). The database manager is able to store data in different formats(e.g. HDF5, CSV, etc.).

The user can decide to perform post-processing operations and/or data mining manipulation on the fly, through the Post-processing and data mining module. The whole calculation may be visualized through the RAVEN GUI, that is able to exploit the communication capabilities present in the external calculation driver for following the simulation evolution (e.g. monitored variables or probability evolution through different branches, etc.).
The whole calculation infrastructure is designed to be agnostic regarding the machine in which it

is running with extremely good performance either in PCs/workstations and HPC clusters.

5 Dynamic PRA analysis on a simplified PWR model

In order to show the capabilities introduced with the new DET module, a simplified PWR Dynamic PRA analysis is here presented. Figure **??** shows the scheme of the PWR model. The reactor vessel

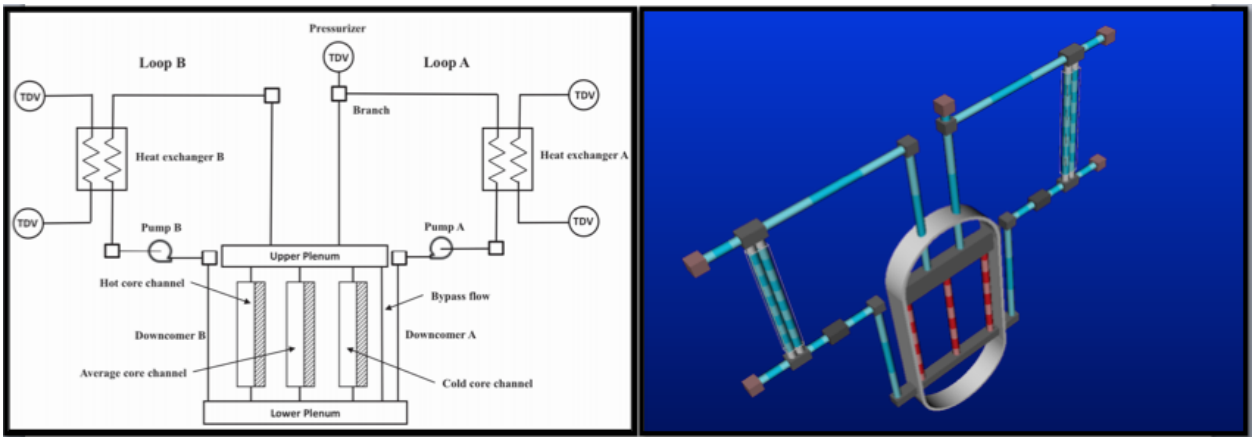


Figure 4: PWR model scheme

model consists of the Down-comers, the Lower Plenum, the Reactor Core Model and the Upper Plenum. Core channels (flow channels with heat structure attached to each of them) were used to describe the reactor core. The core model consists of three parallel core channels and one bypass flow channel. There are two primary loops, i.e., loop A and loop B. Each loop consists of the Hot Leg, a Heat Exchanger and its secondary side pipes, the Cold Leg and a primary Pump. A Pressurizer is attached to the Loop A piping system to control the system pressure. A Time Dependent Volume (pressure boundary conditions) component is used to represent the Pressurizer. Since the RELAP-7 code does not have the two-phase flow capability for all needed components yet, single-phase counter-current heat exchanger models are implemented to mimic the function of steam generators in order to transfer heat from the primary to the secondary.

In order to perform a Dynamic PRA analysis on this simplified model, it has been necessary to control unconventional parameters (i.e. inlet/outlet friction factors), since RELAP-7 still has limitations for the component controllable parameters and models. In the following paragraph, the PRA station black out sequence of events is reported.

5.1 Station Black Out (SBO) analysis

The simulation of a SBO initiating event inferred the introduction, in the control logic, of several components (see Fig. **??**):

- Set of 3 diesel generators (DGs) and associated emergency buses
- Primary power grid line 138 KV (connected to the NSST switchyard)

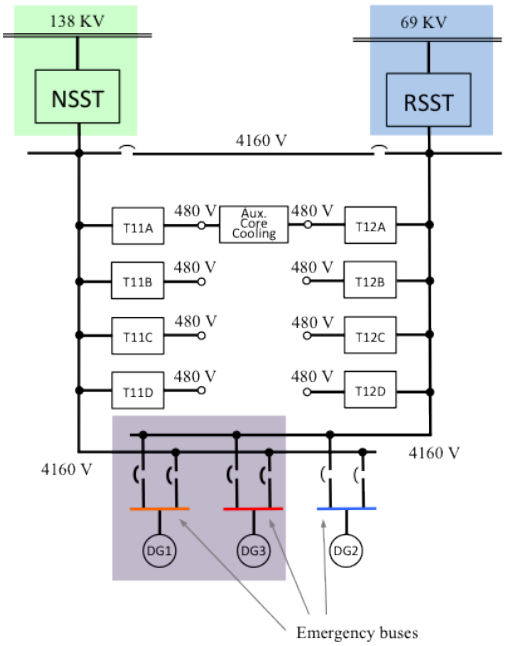
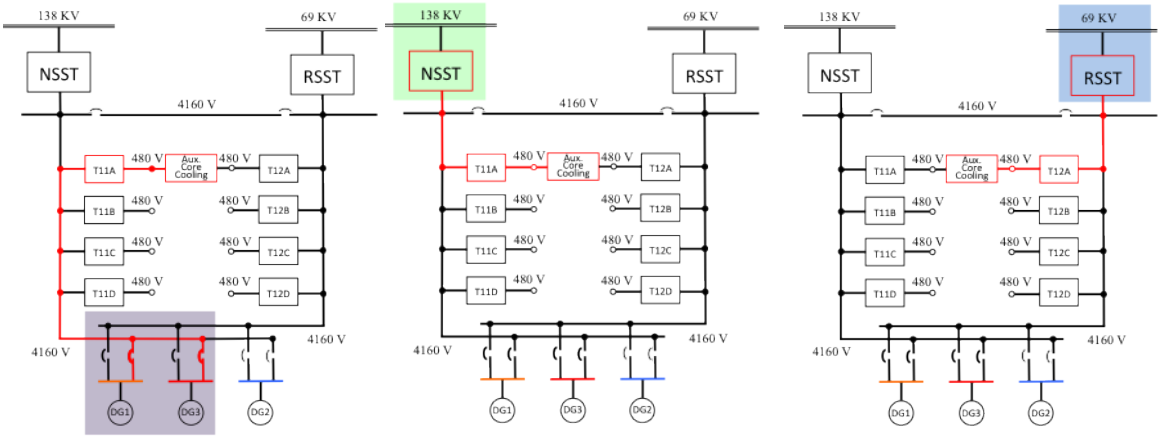


Figure 5: Scheme of the electrical system of the PWR model

- Auxiliary power grid line 69 KV (connected to the RSST switchyard)
- Electrical buses: 4160 V (step down voltage from the power grid and voltage of the electric converter connected to the DGs) and 480 V for actual reactor components (e.g., reactor cooling system)

The scenario is the following:

- An external event causes a loss of off-site power (LOOP) due to damage of the 138 kV line and RSST switchyard; the reactor successfully scrams and, thus, power generated in the core follows the characteristic exponentially decay curve
- The set of DGs fails to start and, hence, conditions of SBO are reached (4160 V and 480 V buses are not energized); all cooling systems are subsequently off-line
- Without the ability to cool the reactor core, its temperature starts to rise
- In order to recover AC electric power on the 4160 V and 480 V buses, two recovery teams are assembled with the following strategy:
  - Recovery Team 1 focuses on the recovery of the DGs: due to internal damage at the DG building, two DGs (i.e., DG1 and DG3) need to be repaired (see Fig. **??** (a))
  - Recovery Team 2 focuses on the recovery of the RSST switchyard; 69KV line is energized but the RSST switchyard needs to be recovered (see Fig. **??** (b))



(a) (b) (c)

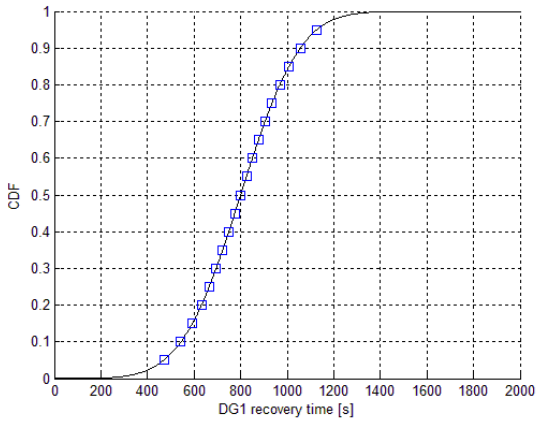
Figure 6: AC power recovery paths through: DGs (a), RSST (b) and 138KV line (c). Red lines indicate electrical path to power Auxiliary cooling system

- Meanwhile the owning company is working on the restoration of the primary 138 KV line (see Fig. **??** (c))
- When the 4160 KV buses are energized (through the recovery of the DGs, RSST or 138KV line), the auxiliary cooling system is able to cool the reactor core and, thus, core temperature decreases.

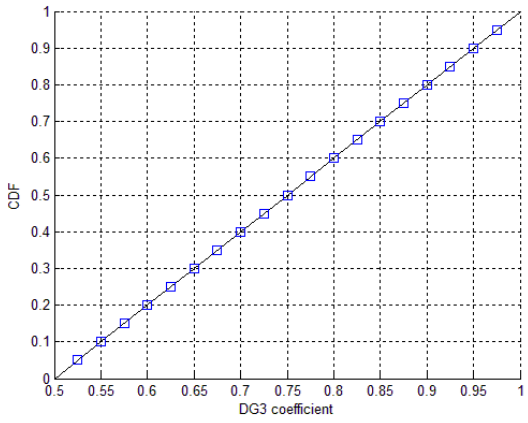
Given the uncertainties associated to the recovery of both DGs, RSST and 138KV line, it has been modeled these three recovery events as stochastic events with probabilistic distributions associated to them. Given the time scale associated to the dynamics of the RELAP-7 PWR model the distribution were as follows:

- DGs: a dead time of 100s is required by Team 1 to gather at the DGs building and DG1 repair time $TDG1$ has a normal distribution having mu = 800 and sigma = 200. This distribution is also truncated such that $0 < TDG1 < 2500$. The recovery time of DG3, TDG3 , is proportional to $TDG1$. Such relation has modeled using a multiplication factor T12, i.e., $TDG3 = TDG1 \times T12$. T12 is uniformly distributed between [0.5 1]
- RSST: a dead time of 400s is needed to assess the damage at the RSST switch-yard and to plan its recovery. Recovery time for RSST, TRSST , is normally distributed with mu = 1400 and sigma= 400
- 138KV line: the recovery of the main AC line T138 is normally distributed with mu = 2000 and sigma = 500

In addition, the clad failure temperature $TC, fail$ is not fixed but it is probabilistic distributed with a triangular distribution with the following parameters:

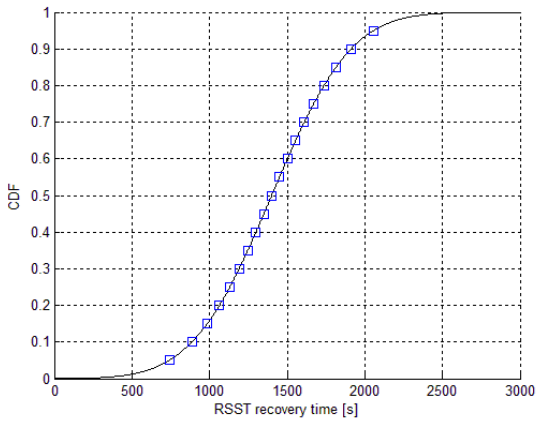


(a)

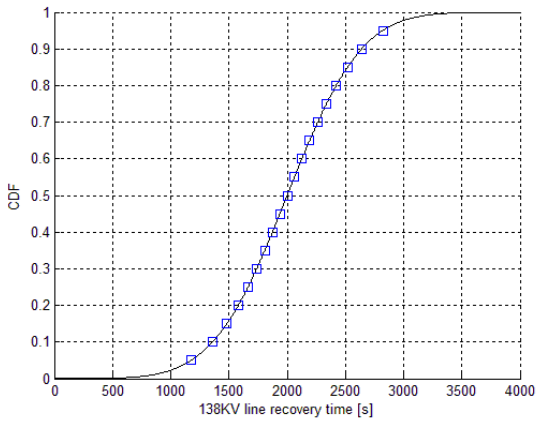


(b)

Figure 7: Cumulative Distribution Functions and DET bins: (a) DG1, (b) DG3 coefficients



(a)



(b)

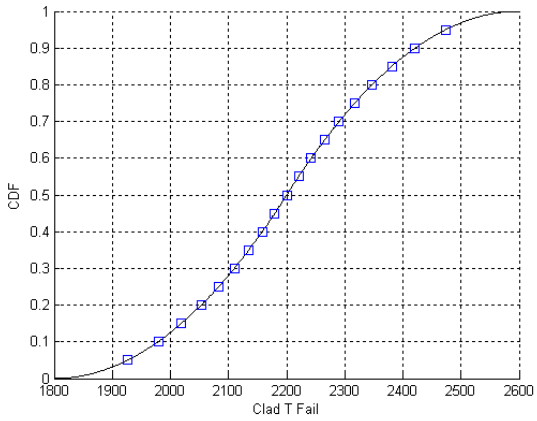
Figure 8: Cumulative Distribution Functions and DET bins: (a) 138kV line, (b) RSST

- mode: xPeak = 1477.59 K, 10CFR regulatory limit
- lower bound: xMin = 1255.37 K, PRA success criterion
- upper bound: xMax = 1699.82 K, Urbanic-Heidrick transition temperature [**?**]

The Dynamic Event Tree calculation has been run considering equally spaced branching probability thresholds (ESBP).

The probability threshold values are reported below:

- DG1 recovery time distribution (Fig. **??** (a) - blue points):
  - Probability Thresholds: 0.05, 0.10, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95



(a)

Figure 9: Cumulative Distribution Function and DET bin for Clad Failure Temperature Distribution

  - Variable Values : 471.03, 543.69, 592.71, 631.68, 665.10, 695.12, 722.94, 749.33, 774.87, 800.00, 825.13, 850.67, 877.06, 904.88, 934.90, 968.32, 1007.29, 1056.31, 1128.97

- DG3 recovery factor distribution (Fig. **??** (b) - blue points):

  - Probability Thresholds: 0.05, 0.10, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95
  - Variable Values : 0.525, 0.550, 0.575, 0.600, 0.625, 0.650, 0.675, 0.700, 0.725, 0.750, 0.775, 0.800, 0.825, 0.850, 0.875, 0.900, 0.925, 0.950, 0.975

- RSST recovery time distribution (Fig. **??** (a) - blue points):

  - Probability Thresholds: 0.05, 0.10, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95
  - Variable Values : 742.06, 887.38, 985.43, 1063.35, 1130.20, 1190.24 1245.87, 1298.66, 1349.74, 1400.00, 1450.26, 1501.34, 1554.13, 1609.76, 1669.80, 1736.65, 1814.57, 1912.62, 2057.94

- 138 kV line recovery time distribution (Fig. **??** (b) - blue points):

  - Probability Thresholds: 0.05, 0.10, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95
  - Variable Values : 1177.57, 1359.22, 1481.78, 1579.19, 1662.76, 1737.80, 1807.34, 1873.33, 1937.17, 2000.00, 2062.83, 2126.67, 2192.66, 2262.20, 2337.24, 2420.81, 2518.22, 2640.78, 2822.43

- Clad Failure Temperature distribution (Fig. **??** - blue points):

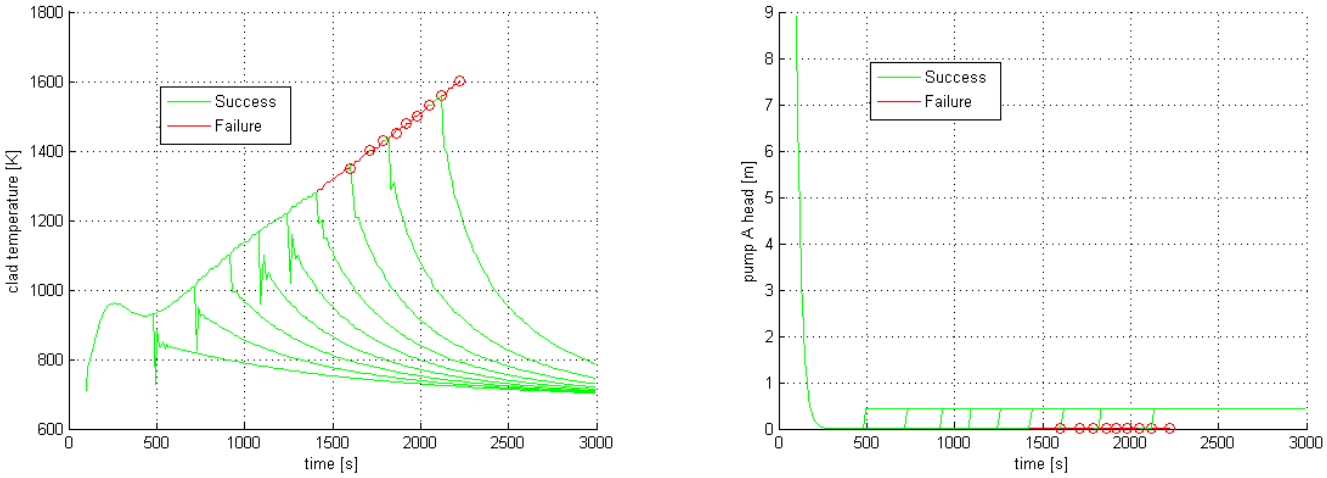


Figure 10: Case ESBP DET I: Clad Temperature and Pump Head temporal profiles (all histories)

- Probability Thresholds: 0.05, 0.10, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95
- Variable Values : 1329.37, 1354.78, 1376.74, 1395.86, 1412.68, 1427.70, 1441.33, 1453.96, 1465.94, 1477.60, 1489.26, 1501.24, 1513.87, 1527.50, 1542.51, 1559.34, 1578.45, 1600.40, 1625.79

Since the scope of this demo is to show the new capabilities contained in RAVEN, and RELAP-7 is not optimized for long simulation times, the transient has been accelerated in order to simulate a maximum of 2500 seconds. In the following section, the simulations results are shown and discussed.

5.2 Results

DET analysis have been performed, obtaining a data-set composed by 37 DET branches resulting in 18 complete histories.

Figures **??** and **??** show the histories obtained for the ESBP DET I and ESVV DET II respectively. Following the station back-out condition, the pump head decreases (pump coast-down) and the temperature of the core rises since reactor core cooling is not available. Such temperature increasing continues until one of the following conditions occurs:

- temperature of the clad reaches the clad failure temperature, indicated with a red circle (red scenarios in Fig.s **??** and **??**), or,
- DG power is restored (green scenarios in Fig.s **??** and **??**).

Note that following the recovery of the DG power (green scenarios) the clad temperature drops and an oscillating behavior can be observed. This is caused by the sudden activation of the auxiliary core cooling system.

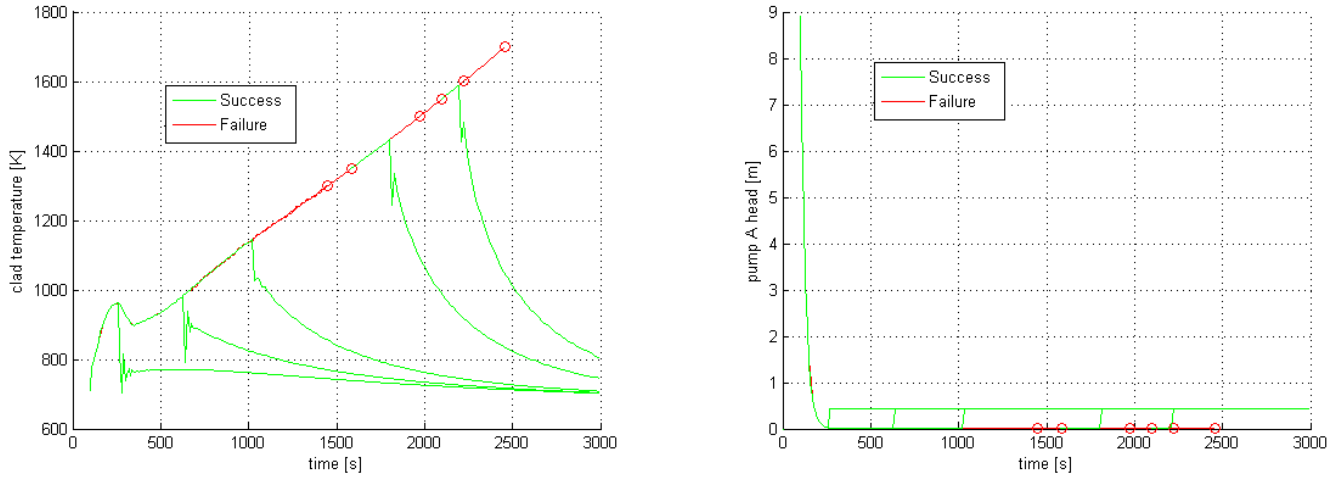


Figure 11: Case ESVV DET II: Clad Temperature and Pump Head temporal profiles (all histories)

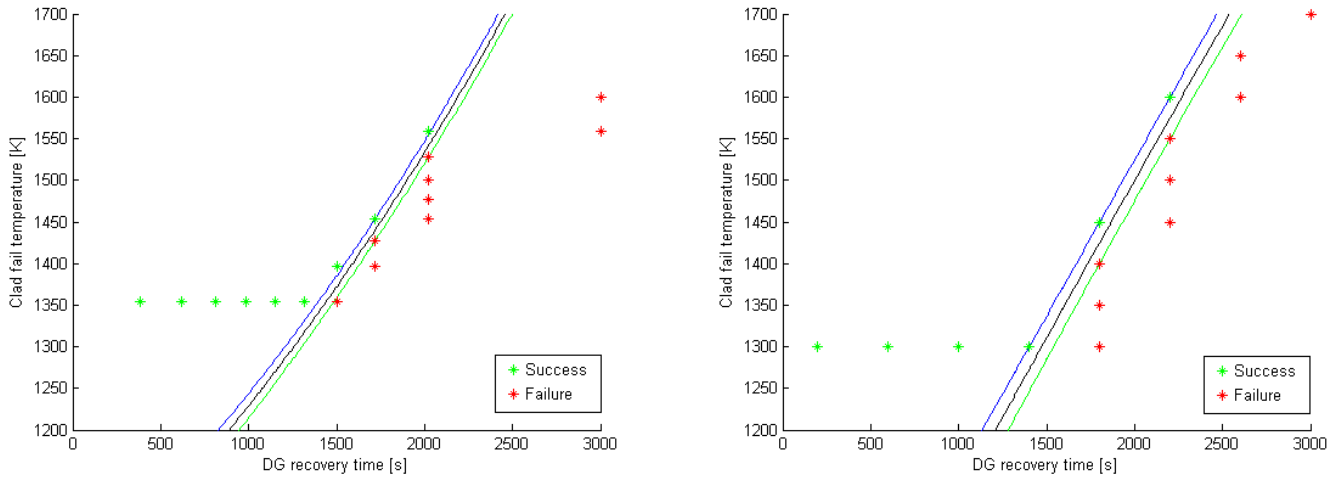


Figure 12: DET Limit Surface: case ESBP DET I(left) and case ESVV DET II (right)

From a safety point of view, the most important information is the frequency of failures in the phase space. This information can be shown plotting the final outcome (green points for system success and red points for system failure) of all the simulations for each data sets in a 2-dimensional space (DG recovery time vs. clad failure temperature) as shown in Fig. **??**. The limit surface, i.e. the boundary between system failure and system success regions (black line), is also indicated in Fig. **??**. Such boundary was determined using a Support Vector Machine (SVM) based algorithm [**?**]. Note that most of the points in this 2-dimensional space (for both data sets) lie in proximity of the limit surface while samples generated by a classical Monte-Carlo based algorithm would be evenly distributed (see Fig. **??**). This example shows one of the advantages of DET algorithm in terms of "choosing" the best set of candidate samples when performing PRA analyses.

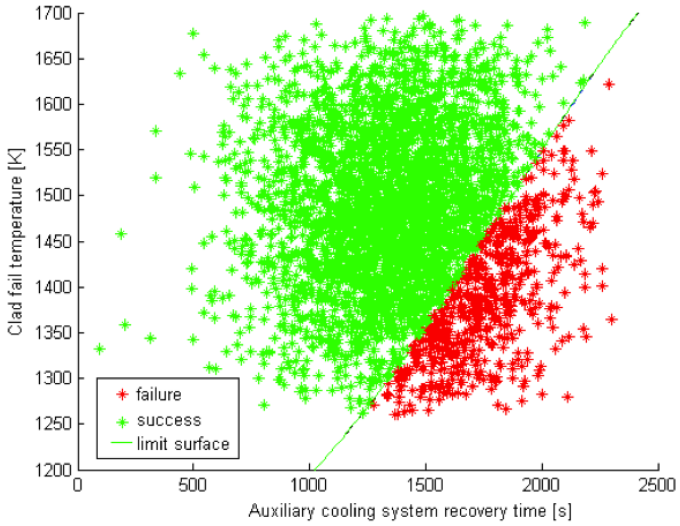


Figure 13: Monte-Carlo Limit Surface.

6 Conclusions

fin

Idaho National Laboratory