INL/EXT-13-30332

RAVEN: Dynamic Event Tree Approach Level III Milestone

Andrea Alfonsi
Cristian Rabiti
Diego Mandelli
Joshua Cogliati
Robert Kinoshita

July 2013

INL/EXT-13-30332

RAVEN: Dynamic Event Tree Approach
Level III Milestone

Andrea Alfonsi
Cristian Rabiti
Diego Mandelli
Joshua Cogliati
Robert Kinoshita

July 2013

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

INL/EXT-13-30332
Unlimited Release
Printed 07/30/2013

RAVEN: Dynamic Event Tree Approach Level III Milestone

Andrea Alfonsi
Idaho National Laboratory
P.O. Box 1625
Idaho Falls, ID 83415-3870
andrea.alfonsi@inl.gov

Cristian Rabiti
Idaho National Laboratory
P.O. Box 1625
Idaho Falls, ID 83415-3870
cristian.rabiti@inl.gov

Diego Mandelli
Idaho National Laboratory
P.O. Box 1625
Idaho Falls, ID 83415-3850
diego.mandelli@inl.gov

Joshua Cogliati
Idaho National Laboratory
P.O. Box 1625
Idaho Falls, ID 83415-3870
joshua.cogliati@inl.gov

Robert Kinoshita
Idaho National Laboratory
P.O. Box 1625
Idaho Falls, ID 83415-2210
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

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 are 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 to perform 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, the control logic infrastructure is used to model stochastic events, such as components failures, and perform uncertainty propagation. Such stochastic modeling is deployed 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 analysis of dynamic stochastic systems using the newly developed RAVEN DET capability. As an example, a DPRA analysis, using DET, of a simplified pressurized water reactor for a Station Black-Out (SBO) 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). RAVEN has been designed in a highly modular and pluggable way in order to enable easy integration of different programming languages (i.e., C++, Python) and coupling with other MOOSE based and external applications.
The goal of this report is to highlight the newly developed Dynamic Event Tree (DET) module embedded in the code and its utilization in conjunction with RELAP-7.

As for all **P**robabilistic **R**isk **A**ssessment (PRA) software the capability to fully control the plant evolution during the simulation represents a highly desirable feature 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 codes, 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.The final outcome is a very general and flexible implementation of the plant control logic that will easily allow the integration of proprietary information without any change in the RELAP-7 code. This is also regarded as a facilitating factor for the quick deployment of RELAP-7.

In summary, RAVEN is a multi-purpose 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 and Event Tree based analysis of the probabilistic behavior of the NPP. 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 implementation. In addition a **S**tation **B**lack **O**ut (SBO) DET based analysis of a simplified **P**ressurized **W**ater **R**eactor (PWR) model will be shown.

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 1 can now be rewritten as follows:

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

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. 4 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. 4 which now becomes:

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

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. 5:

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

2.2 Software Overview

RAVEN [**?**], is plugged into the software environment MOOSE [**?**]. MOOSE is a computer simulation framework, developed at Idaho National Laboratory (INL),which simplifies the process for predicting the behavior of complex systems and developing non-linear, multi-physics simulation tools. MOOSE provides the algorithms for the solution of generic partial differential equations, and all the manipulation tools needed to extract information from the solution field. 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 the 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 highly modular and pluggable way in order to enable easy integration of different programming languages (i.e., `C++`, `Python`) and coupling with other MOOSE based applications and not only. 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 simultaneously 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 extract 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 simulation, as the real plant control system would do. 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 (pre-generated control logic blocks) 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. Though simple by itself, the GUI will provide tools to automate 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 4.1

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 `PyQt4`, which is a `Python` interface for `Qt` (https://qt-project.org). `Qt` is a popular library used for cross-platform GUI development. RAVENs GUI is developed starting from Peacock, which is a GUI interface for generic MOOSE based applications. Because RELAP-7 is rather different from the other MOOSE based applications, much effort has been required to specialize Peacock for RAVEN and further addition will continue to follow RELAP-7 development.

3 The Dynamic Event Tree Methodology

In the ET approach, starting from an initiating event (i.e. accident event), several final outcomes are forecasted by assembling "all" combinations of events that might follow. This leads to the classical tree structure. The probability of a specific outcome (branch) is given by the product of the events along that branch. Two of the disadvantages of these methods are that, first timing/sequencing of events and system dynamics are not explicitly accounted for in the analysis and second the status and behavior of the system has no influence on the likelihood of the events. In

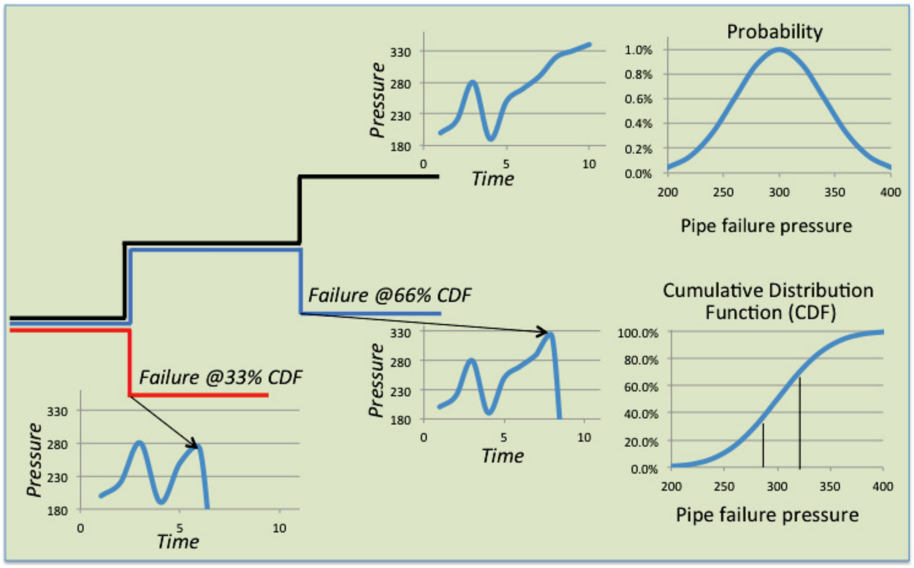


Figure 1: Dynamic Event Tree Conceptual Scheme.

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 time evolution of the accident scenario. In DET, event sequences are run simultaneously starting from a single initiating event. The branchings occur at user specified times/conditions 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. 1).

This leads to a more realistic and mechanistically consistent analysis of the considered system. DPRA, including DET methodologies, are designed to take the timing of events explicitly into account, which can become very important especially when uncertainties associated to 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".

From a simulation point of view, a DET analysis starts with a single simulation that, most likely, represents the NPP immediately after an initiating event (an event that may lead to a risky state of the NPP). Every time the simulation faces an event affected by a probabilistic behavior (e.g. failure of a relief valve, etc.), several simulation branches are spawned. Each simulation accounts for a fraction of the possible outcomes. Figure 1 exemplifies this approach. After an initiating event, the simulation follows the accident sequence and a pipe is affected by a probability of failure described by a **P**robabilistic **D**istribution **F**unction (i.e. pipe failure probability as function of the internal pressure) and a corresponding **C**umulative **D**istribution **F**unction (**CDF**). In addition, the user provides a grid of probability thresholds for the CDF. Every time the simulation detects a pressure corresponding to a threshold in the probability grid, a new set of simulation branches is generated (ie., a branch where the pipe is damaged and another where nothing happened).

In general each sequence continues until another event occurs and a new set of branches is spawned. The simulation ends when an exit condition or a maximum mission time is reached.

4 Analysis of Dynamic Reactor Accident Evolution Module

RAVEN is able to perform DET analysis through the newly developed module "**A**nalysis of **D**ynamic **R**eactor **A**ccident **E**volution", that has been included in the RAVEN external Python manager.

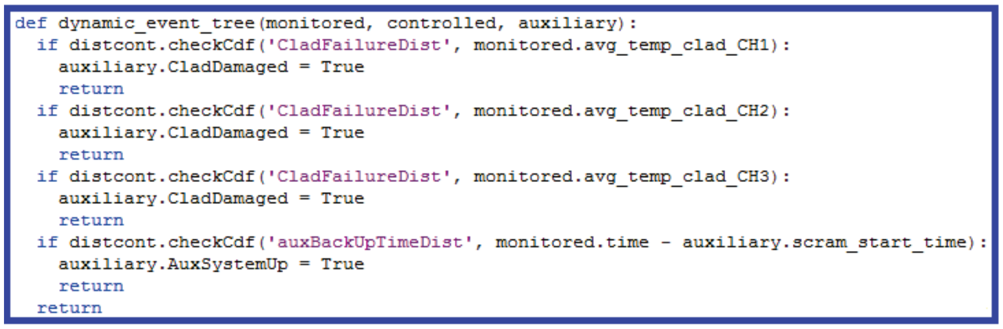
Following the philosophy of the DET approach, this module lets RELAP-7 find the several possible outcomes of an accident sequence. When the system reaches certain conditions, for which more than one outcome may be possible (e.g. opening system of a valve damaged), a sampler generates the corresponding new set of branches (alternative sequences), associating a conditional probability to each.
The user defined branching laws can be based on the following logics:

- Thresholds, either in terms of cumulative probability distribution or variable magnitude. An example is the failure of a pipe as a function of pressure probability distribution
- On demand. An example is failure on demand of an actionable component (ie. relief valve)
- Time driven. This is the simplest case where the user defines a branching logic, consequentially changing the plant status, at a certain point in time

For the analysis of complex systems, the number of branches may become extremely large. In order to avoid unacceptable growth of problem due by an excessive number of branches, the user needs to specify exit conditions for the simulation(termination laws). For example, maximum mission time, rules based on the simulator physical model (i.e. Maximum temperature of the fuel cladding, etc.).
In other similar codes, one of the most common termination laws is a probability cut-off: a branch's execution is stopped when its probability falls below a given limit. This approach should be used with caution since it may have a large impact on the probability estimation of the possible final outcomes. This might happen when the number of branches is very large, and, therefore, the magnitude of their conditional probability is very low. In such a case, a large number of branches might be terminated before reaching the final outcome, biasing the analysis. In order to avoid these issues and preserve the probability conservation, the user can not directly input, in RAVEN, a branching probability cut-off. In RAVEN there is no distinction between the treatment of aleatory and epistemic variables. For example, the pressure at which a pipe fails to operate can be considered an epistemic variable, but it is treated the same as the recovery time of the auxiliary cooling system. Both variables can be handled with a branching logic driven by the associated CDFs or variables magnitude. The only case in which they might be treated differently is when an epistemic variable sets the initial value of a parameter. A classical example of this situation is the uncertainty associated to the parameters used in the RELAP-7 equations. Even in this case, it does not seem necessary to have a special treatment for these variables. In the DET module, this special case can be handled with a multiple branch logic (two or more branches) at the begin of the initial simulation (ie. the simulation that represents the initiating event).

The above consideration, in conjunction with the brief overview of the mathematical framework analyzed in section 2.1, gives an indication on how RAVEN (and the DET module) combines all the capabilities present in other similar codes [**?**].

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 2 lists set of branching rules. The “checkCdf” function, available in the distribution container, checks if the associated cumulative probability threshold has been passed, and, if it has, sends 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’s `Python` external driver, which represents the core of PRA analysis.

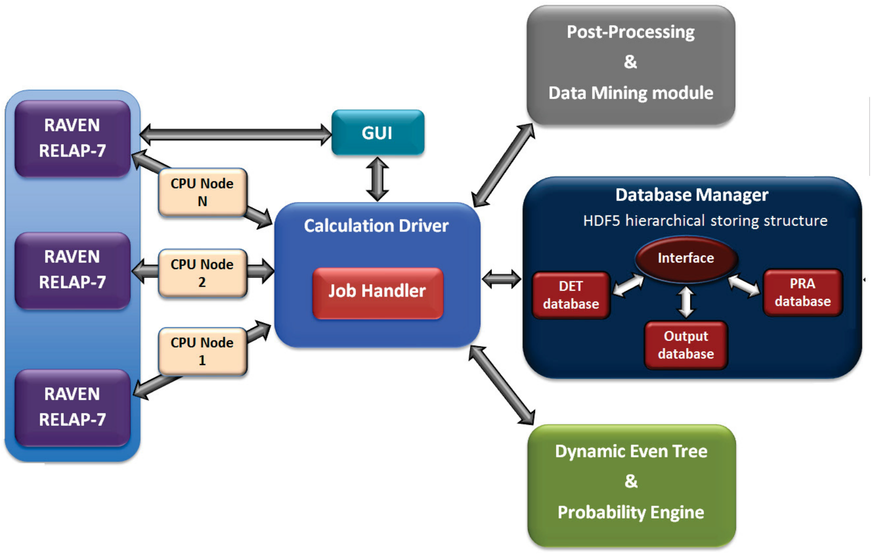


Figure 3: Software calculation infrastructure scheme

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
- Distributed Computing Environment Interface

The General Calculation Driver is responsible 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 3 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 initial 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 the control logic of RAVEN/RELAP-7 detects that a stopping condition is reached. The DET module decides whether to branch or not depending on the information received, through the general calculation driver, from 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 exceeded). 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-processor, 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 4 shows the scheme of the PWR model. The reactor vessel

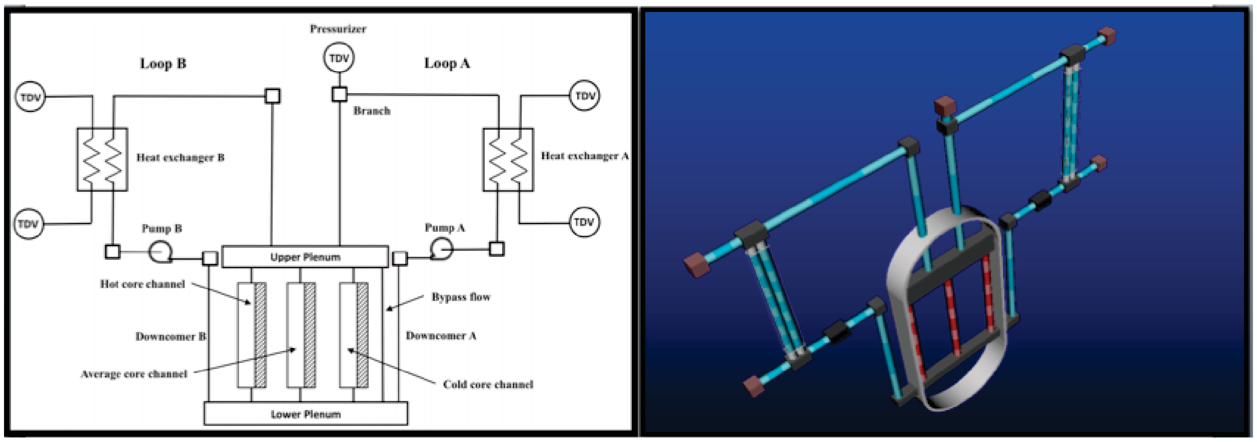


Figure 4: PWR model scheme

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) are 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 two-phase flow capability has not being used for this test, 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 DPRA analysis of 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 required the introduction, in the control logic, of several components (see Fig. 5):

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

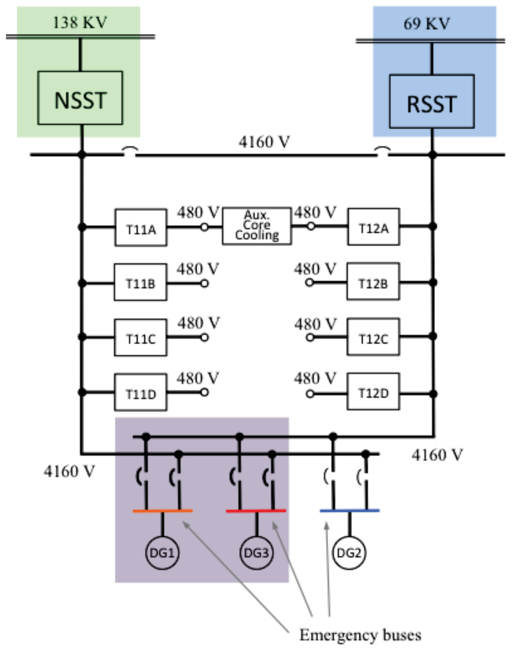
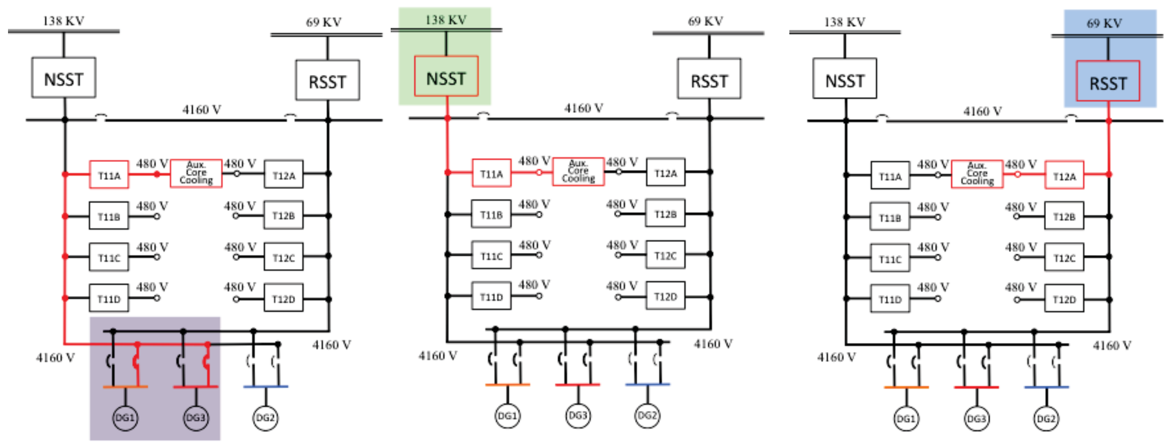


Figure 5: Scheme of the electrical system of the PWR model

- 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. 6 (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. 6 (b))



(a) (b) (c)

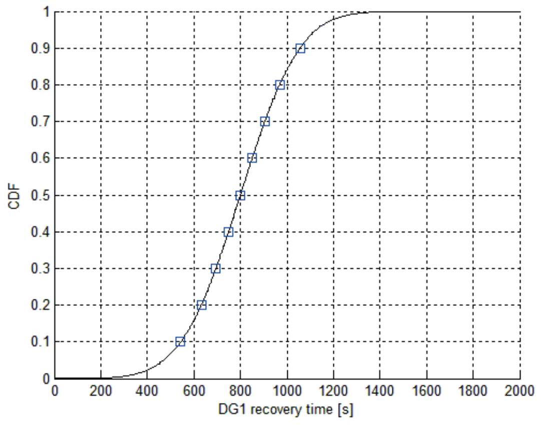
Figure 6: AC power recovery paths through: DGs (a), RSST (b) and 138KV line (c). Red lines indicate electrical path to power Auxiliary cooling system

- Meanwhile the owning company is working on the restoration of the primary 138 KV line (see Fig. 6 (c))
- When the 4160 V 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.

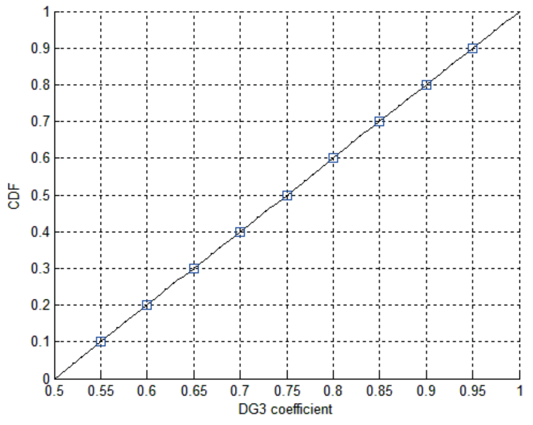
Given the uncertainties associated to the recovery of both DGs, RSST and 138KV line, a stochastic model has been used to represent these events . Given the time scale associated to the dynamics of the RELAP-7 PWR model the corresponding probability distribution functions were as follows:

- 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 $T_{C,fail}$ is not fixed but it is probabilistic distributed with a triangular distribution with the following parameters:

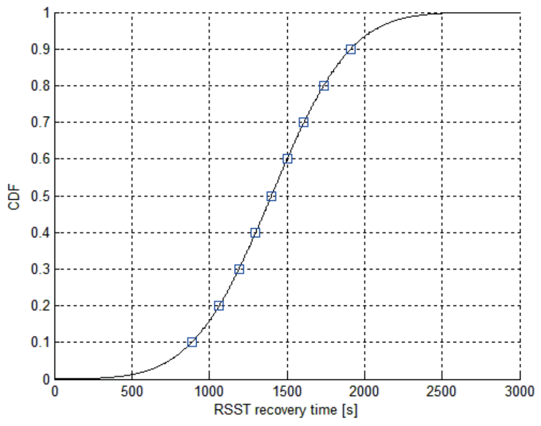


(a)

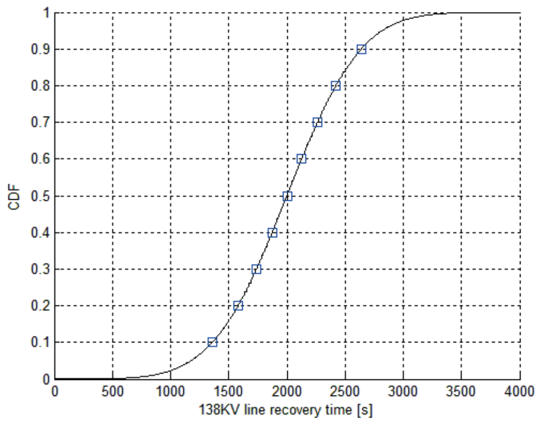


(b)

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



(a)



(b)

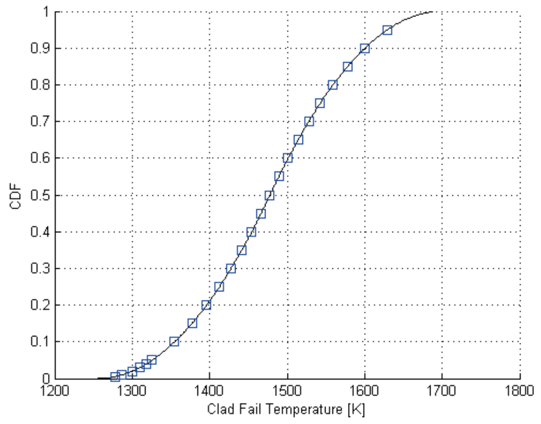
Figure 8: Cumulative Distribution Functions and DET bins: (a) 138kV line, (b) RSST

- 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 DET calculation has been run considering equally spaced branching probability thresholds (ESBP).

The probability threshold values are reported below:

- DG1 recovery time distribution (Fig. 7 (a) - blue points):
  - Probability Thresholds: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9



(a)

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

  - Variable Values (s) : 543.69 , 631.68, 695.12, 749.33, 800.00, 850.67, 904.88, 968.32, 1056.31

- DG3 recovery factor distribution (Fig. 7 (b) - blue points):
  - Probability Thresholds: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
  - Variable Values (-) : 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95

- RSST recovery time distribution (Fig. 8 (a) - blue points):
  - Probability Thresholds: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
  - Variable Values (s) : 887.38, 1063.35, 1190.24, 1298.66, 1400.00, 1501.34, 1609.76, 1736.65, 1912.62

- 138 kV line recovery time distribution (Fig. 8 (b) - blue points):
  - Probability Thresholds: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
  - Variable Values (s) : 1359.22, 1579.19, 1737.80, 1873.33, 2000.00, 2126.67, 2262.20, 2420.81, 2640.78

- Clad Failure Temperature distribution (Fig. 9 - blue points):
  - Probability Thresholds: 0.005, 0.01, 0.02, 0.03, 0.04, 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 (K) : 1304.57, 1307.51, 1313.27, 1318.89, 1324.38, 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

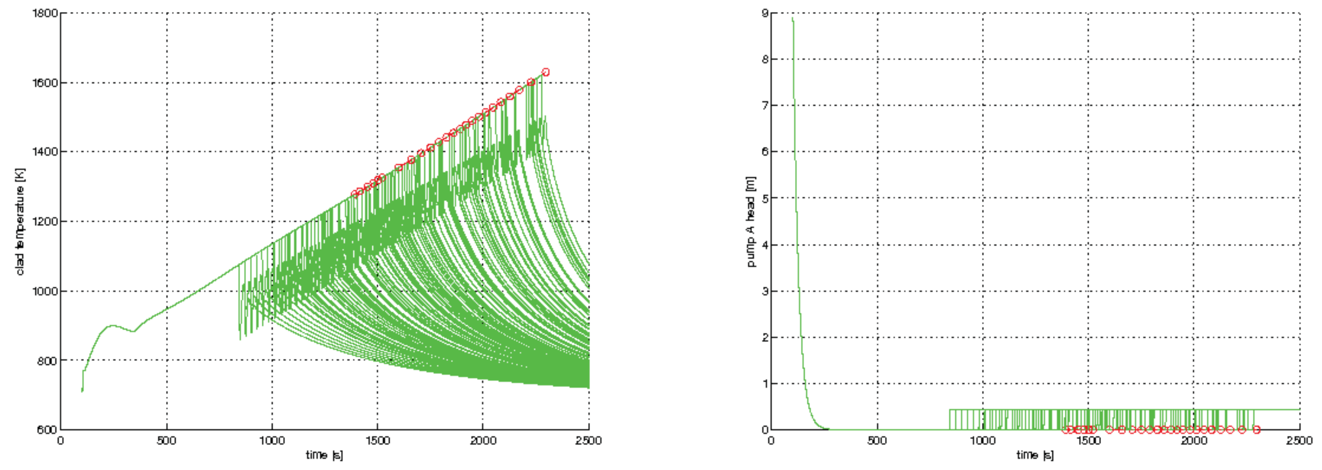


Figure 10: Clad Temperature (Left) and Pump Head (Right) temporal profiles (all histories)

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 reach a reasonable number of failures at of 2500 seconds. In the following section, the simulations results are shown and discussed.

5.2 Results

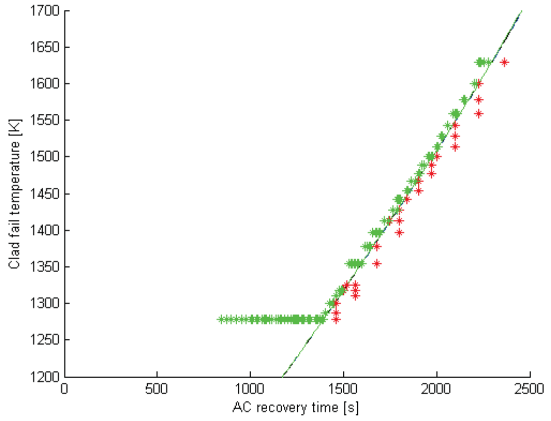
DET analysis has been performed, obtaining a data-set composed by 593 DET branches resulting in 290 complete histories.

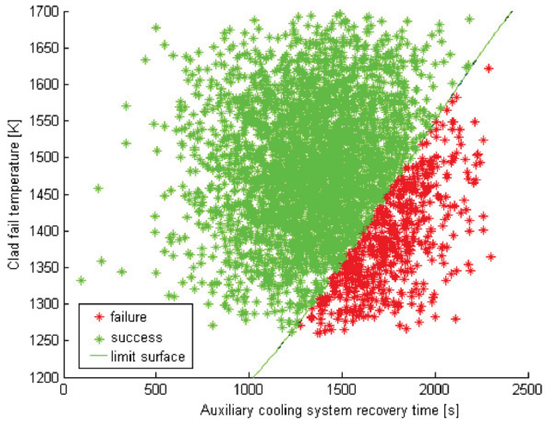
Figure 10 shows the histories obtained for the calculation performed. Following the station back-out condition, the pump head decreases (pump coast-down) and the temperature of the core rises since reactor core cooling is not available. Such temperature increasing continues until one of the following conditions occurs:

- Temperature of the clad reaches the clad failure temperature, indicated with a red circle (red scenarios in Fig. 10, or,
- Auxiliary cooling system is restored (green scenarios in Fig. 10).

Note that the back up of the cooling system causes oscillations on the clad temperature. These oscillations are determined by the sudden insertion of the auxiliary system that imposes a mass flow rate 5% of the operational one. The initial local maximum is due to the sudden loss of heat sink while the fuel still behaves as a heat reserve; after a while the remaining cooling capacity (pumps inertia) succeeds in reducing the temperature. The situation inverses again when the primary and secondary pumps completely stop.

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





(a) (b)

Figure 11: Limit Surfaces: (a) Dynamic Event Tree, (b) Monte-Carlo

space. Since the DET sampling has been performed on 5 distributions, among which 4 distributions sampled the same parameter *ACS* (Auxiliary Cooling System) Recovery Time, it is possible to collapse all the outcomes in a 2D space (ACS recovery time vs. clad failure temperature) as shown in Fig. 11 (a). The limit surface, i.e. the boundary between system failure and system success regions (black line), is also indicated in Fig. 11a. 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. 11 (b)). This example shows one of the advantages of DET algorithm in terms of “choosing” the best set of candidate samples when performing PRA analyses [**?**].

6 Conclusions

The milestone of developing the DET methodology has been fully achieved. The DET module implemented in RAVEN is fully functional and makes RAVEN a state-of-art PRA simulation tool. In addition a framework (part of a larger general framework [?]) that makes this results replicable and stable has been generated. The framework is capable of deploying the DET and handling the data post processing. Data outputs are currently stored in HDF5 compressed format, which is a general portable format supported in almost all computing architectures [?]. The DET framework still needs to be integrated within the RAVEN/Peacock GUI, in order to show in real time some important information about the DET calculation (Branches' status, associated conditional probabilities, etc.).
The methodology has shown interesting advantages with respect a standard Monte-Carlo approach:
- Strong reduction in computational time while preserving the physical and probabilistic behavior of the plant;
- As shown in Fig. 11a, the methodology is "failure oriented", since as soon as the a failure region is met, subsequent samplings are performed in its proximity.
These and other minor advantages make the DET a promising methodology to perform PRA analysis.

In the near future, an Adaptive Dynamic Event Tree methodology is going to be developed, in order to investigate low probability, high risk zones of the input space.

Idaho National Laboratory