Dynamic Event Tree Analysis Through RAVEN

A. Alfonsi*, C. Rabiti*, D. Mandelli*, J.J. Cogliati*, R.A. Kinoshita*, A. Naviglio[†]

* Idaho National Laboratory
2525 Fremont Avenue, Idaho Falls, ID 83415
{andrea.alfonsi, cristian.rabiti, diego.mandelli, joshua.cogliati, robert.kinoshita}@inl.gov

[†] Università di Roma "La Sapienza", Dipartimento Ingegneria Energetica e Nucleare
Via Eudossiana, 18, 00184, Rome, Italy
antonio.naviglio@uniroma1.it

ABSTRACT

Conventional **Event-Tree** (**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 **Dynamic Probabilistic Risk Assessment** (**D-PRA**), have been developed. **Monte-Carlo** (MC) and **Dynamic Event Tree** (**DET**) are two of the most widely used D-PRA methodologies to perform safety assessment of **Nuclear Power Plants** (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 also models 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 paper 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.

Key Words: Reactor Simulation, Probabilistic Risk Assessment, PRA, Dynamic Event Tree, RELAP-7

1. INTRODUCTION

RAVEN (**R**eactor **A**nalysis and **V**irtual control **EN**viroment) [1,2] 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 paper is to highlight the newly developed Dynamic Event Tree module embedded in the code and its utilization in conjunction with RELAP-7. RAVEN 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, based on dimensionality and cardinality reduction, are available. This paper wants to provide an overview of the DET structure, highlighting the mathematical framework

from which its structure is derived, the main conceptual decisions have been taken and its software layout. In addiction it will be shown a **S**tation **B**lack **O**ut (SBO) Dynamic Event Tree 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 [3]:

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

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:

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

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. 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.

Thus, RAVEN is using this approach to solve Eq. 4 which becomes:

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

Even if all information needed is contained in $\bar{x}$ and $\bar{v}$, it is not a practical and efficient way to implement the control system. Hence, 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 system into the mathematical framework leads to the following formulation of the Eq. 5:

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

2.2. Software Overview

RAVEN [4], is plugged with the software environment MOOSE [5]. 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) [6] 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 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 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 in a Python framework that contains the capabilities and interfaces to drive a PRA analysis. Its basic infrastructure in connection with

the Dynamic Event Tree module will be discussed in section 4.1

As previously mentioned, a Graphical User Interface (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 present 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

In technological complex systems, as nuclear power plants, an accident scenario begins with an initiating event and, through the interaction of dynamics and stochastics, evolves over time. This mutual action leads to the production of infinitely many different scenarios, that define a continuous dynamic event tree with infinite branches. At each time point, the stochastic variability of the accident outcomes is determined by a multi-variate probability distribution. The PRA analysis needs an approximation to this distribution for selected consequence variables. A way to achieve this goal is an Event Tree approach. In dynamic PRA analysis, Conventional Event Tree (ET) 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.
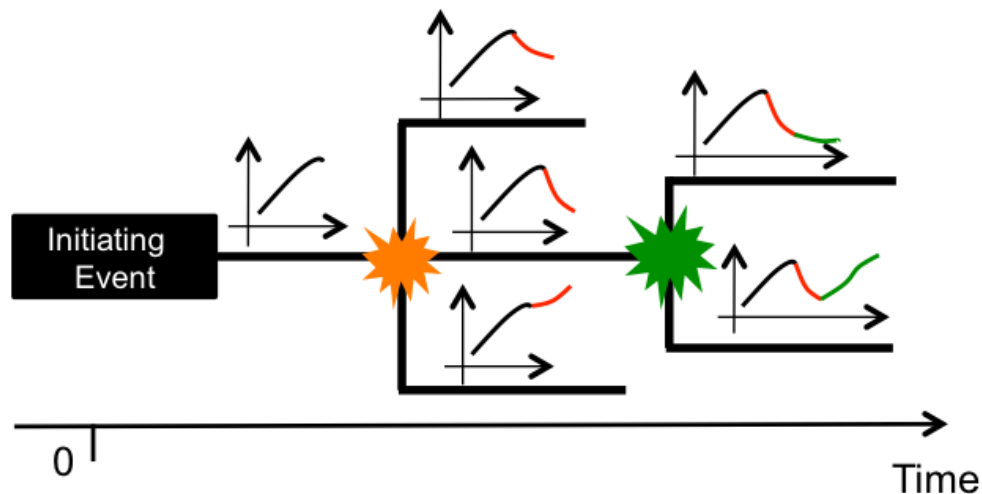

Figure 1: Dynamic Event Tree Conceptual Scheme.

One of the disadvantages of this method is that timing/sequencing of events and system dynamics is not explicitly accounted for in the analysis.In order to overcome these limitations a "dynamic" approach is needed. The Dynamic Event Tree (DET) technique brings several advantages [7], among which the fact that it simulates probabilistic system evolution in a way that is consistent with the severe accident model. This leads to a more realistic and mechanistically consistent analysis of the system taken in consideration. The dynamic PRA, in general, and the Dynamic Event Tree 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. Hence, 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 1 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 $t = T$. 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. RAVEN APPROACHING DYNAMIC METHODOLOGIES: ANDREA MODULE

RAVEN is now able to perform Dynamic Event Tree 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. As easily understandable, in complex cases, 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 truncated when its probability falls below a given limit. No countermeasures are generally taken to conserve the global probability. 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 a branching probability cut-off.
As can be inferred from the previous dissertation, the package is aimed to provide 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 [7], 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 2.1, gives an indication on how RAVEN (and the DET module) combines the capabilities present in other similar, but not equal, 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". On the other side, the user can act on 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 the triggers are identified by Probability Distribution Functions (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 dynamic event tree logic in the previous Python control logic input, 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.).

```python
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 Dynamic Event Tree logic

As an example, in Figure 2, it is reported a set of branching rules. The "checkCdf" function, available in the distribution container, check 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 will be briefly presented. As already mentioned, the DET module is part of RAVEN Python external driver, which represents the core of PRA analysis.
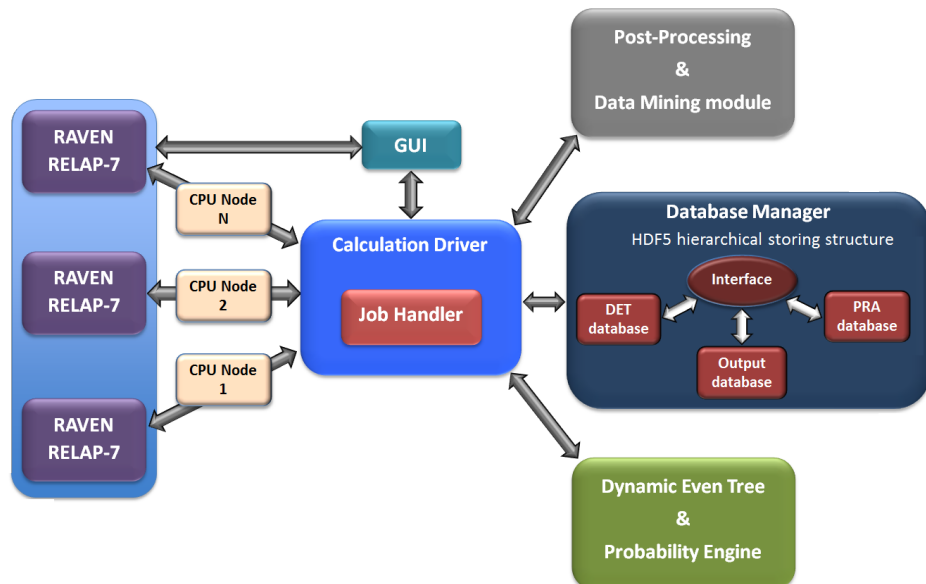

Figure 3: Software calculation infrastructure scheme.

The external driver manager has the control of the different modules that support the Dynamic Event Tree calculation:

- General Calculation Driver and Job Handler
- Visualization and Input supporting GUI
- Dynamic Event Tree and Probability Engine
- Database manager
- Post-processing and Data Mining module
- 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 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 Dynamic Event Tree 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. DEMO FOR A PWR PRA ANALYSIS

In order to show the capabilities introduced with the new Dynamic Event Tree module, a simplified PWR Dynamic PRA analysis has been employed. Figure 4 shows the scheme of the PWR model. The reactor vessel 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
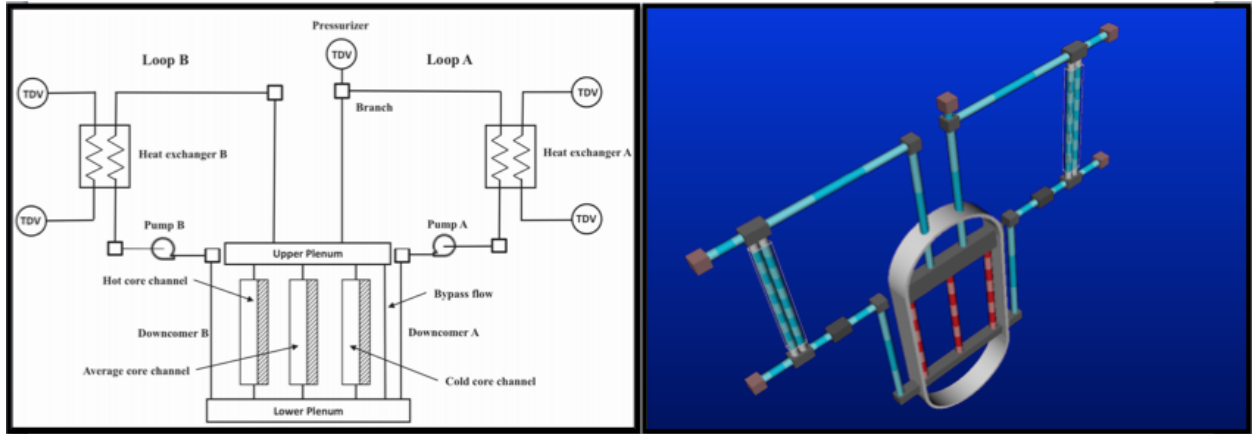
Figure 4: PWR model scheme.

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 Probabilist Risk Assessment analysis has been performed simulating a Station Black Out accident, sampling the probability thresholds (i.e. branching triggers) on the CDFs representing the recovery time of the diesel generators (DGs) t_1 (Normal distribution, mu = 1100 s, sigma = 700 s, xMin = 3 s, xMax = ∞) and the clad failure temperature TCf (Triangular distribution, xPeak = 1477.59[*] K, xMin = 1255.37[†] K, xMax = 1699.82 K [8]). Two sets of Dynamic Event Tree calculations have been run. They differ for the strategy through which the probability thresholds have been chosen:

- ESBP, equally spaced branching probability thresholds

- ESVV, branching probability thresholds corresponding to equally spaced variable values (TCf and t_1)

The probability threshold values for both calculations are reported in the list below:

- **First DET calculation** :

  - Clad Failure Temperature distribution (Fig. 6 - - green points):
    * Probability Thresholds: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
    * Variable Values : 1354.75, 1395.92 , 1427.50, 1454.13, 1477.59, 1501.05, 1527.68, 1559.27, 1600.43
  - DGs recovery time distribution (Fig. 5 - green points):
    * Probability Thresholds: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
    * Variable Values : 382.52, 621.06, 813.45, 985.89, 1151.28, 1319.82, 1502.67, 1718.86, 2021.09

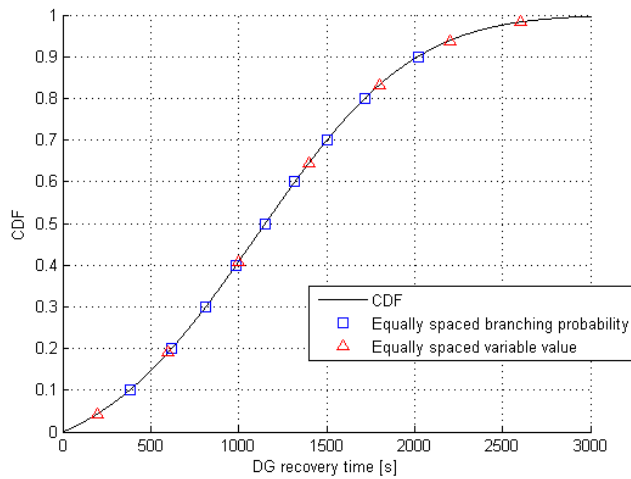[*]Typical PRA success criteria.
[†]10 CFR50.46 limit.

Figure 5: Diesel Generator Recovery Time CDF.

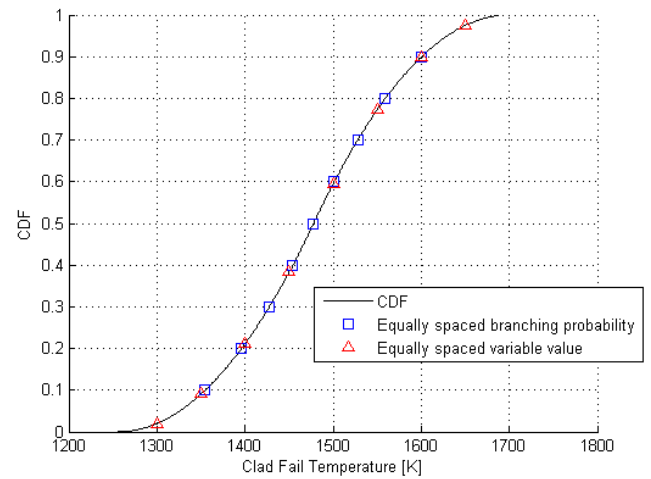Figure 6: Clad Failure Temperature CDF.

- **Second DET calculation**:

    - Clad Failure Temperature distribution (Fig. 6 - red points):
        * Probability Thresholds: 0.0200, 0.0910, 0.2120, 0.3840, 0.5960, 0.7730, 0.8990, 0.9750
        * Variable Values : 1300.00, 1350.00, 1400.00, 1450.00, 1500.00, 1550.00, 1600.00, 1650.00
    - DGs recovery time distribution (Fig. 5 - red points):
        * Probability Thresholds: 0.0433, 0.1901, 0.4086, 0.6451, 0.8315, 0.9383, 0.9829
        * Variable Values : 200.00, 600.00, 1000.00, 1400.00, 1800.00, 2200.00, 2600.00

In addiction, a Monte-Carlo (MC) calculation has been run. It has been decided to perform only 50 samplings (randomizing t_1 and TCf), in order to maintain a similar numerosity (number of "samplings") among MC and DET calculations.

The SBO transient is based on the following sequence of events (starting from a steady-state operational condition of the Nuclear Power Plant [9]):

- 100.0 seconds, transient begins

- 101.0 seconds, loss of power grid and immediate shutdown of the reactor(scram):

    - Pump coast-down;

    - Decay heat power;

    - Diesel Generators and residual heat removal system (RHRS) not available.

- t_1, recovery of the diesel generators

- t_2, end of transient either for clad failure or 3000 seconds of simulation (PRA success)

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 3000 seconds. In the following section, the simulations results are shown and discussed.
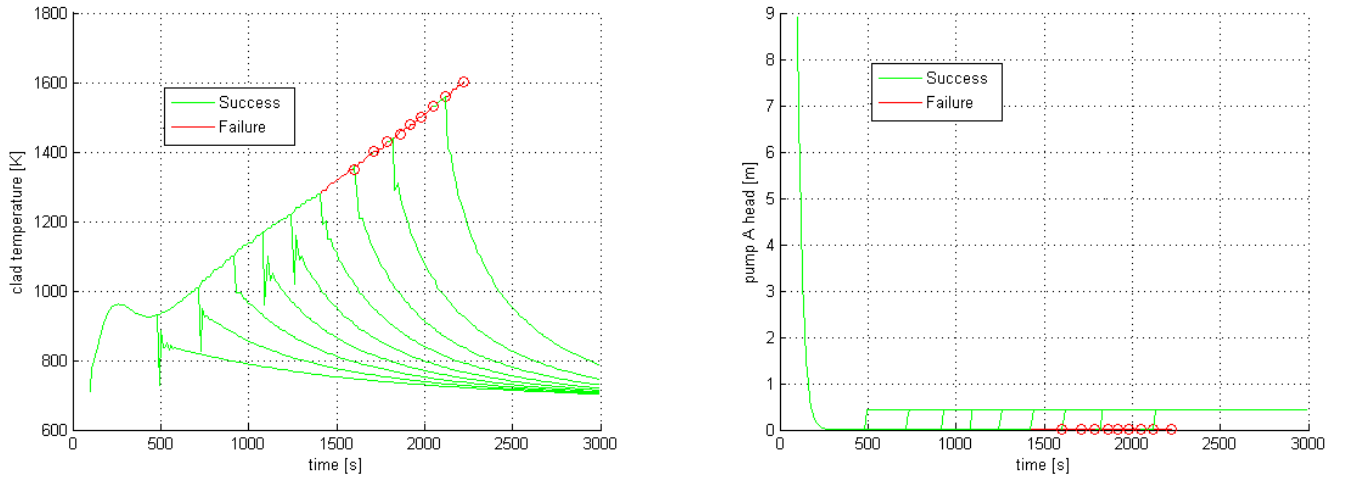
Figure 7: Case ESBP DET I: Clad Temperature and Pump Head temporal profiles (all histories)

5.2. Results

DET analyses have been performed, obtaining the following data sets:

- *ESBP DET I*: 37 DET branches resulting in 18 complete histories

- *ESVV DET II*: 31 DET branches resulting in 15 complete histories

Note that the number of branches (and, hence, the number of histories) are different between the two different calculation performed since a difference number of branching thresholds have been considered (see Section 5.1).
Figures 7 and 8 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 7 and 8)

- DG power is restored (green scenarios in Fig.s 7 and 8)

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 pressure waves created by the sudden activation of the auxiliary core cooling system.

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. 9. The limit surface, i.e. the boundary between system failure and system success regions (black line), is also indicated in Fig. 9. Such boundary was determined using a Support Vector Machine (SVM) based algorithm [10]. 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. 10). This example shows one of the advantages of DET algorithm in terms of "choosing" the best set of candidate samples when performing PRA analyses.
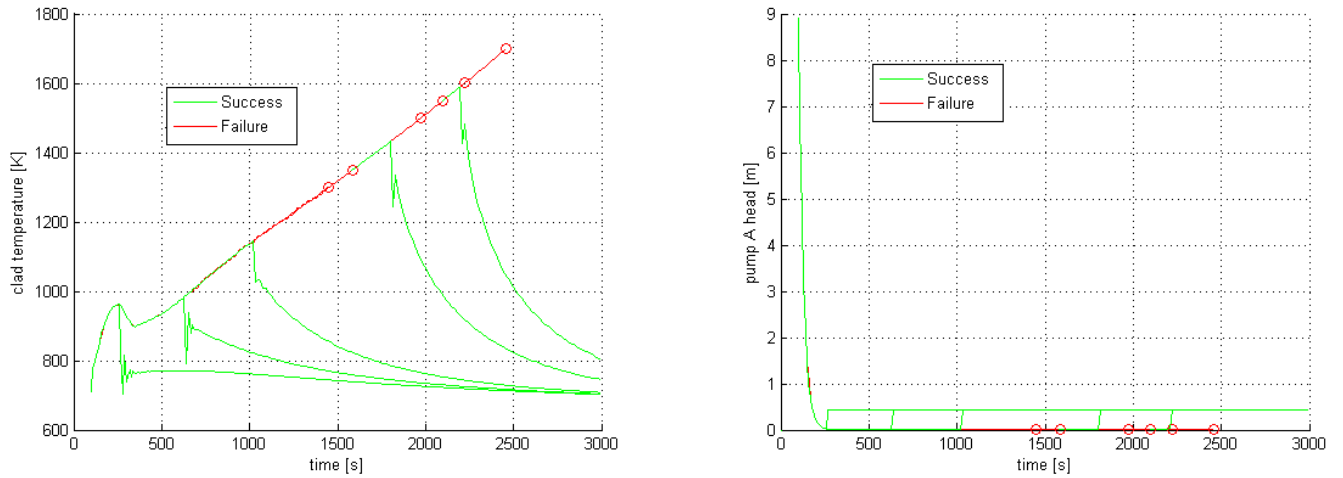
Figure 8: Case ESVV DET II: Clad Temperature and Pump Head temporal profiles (all histories)
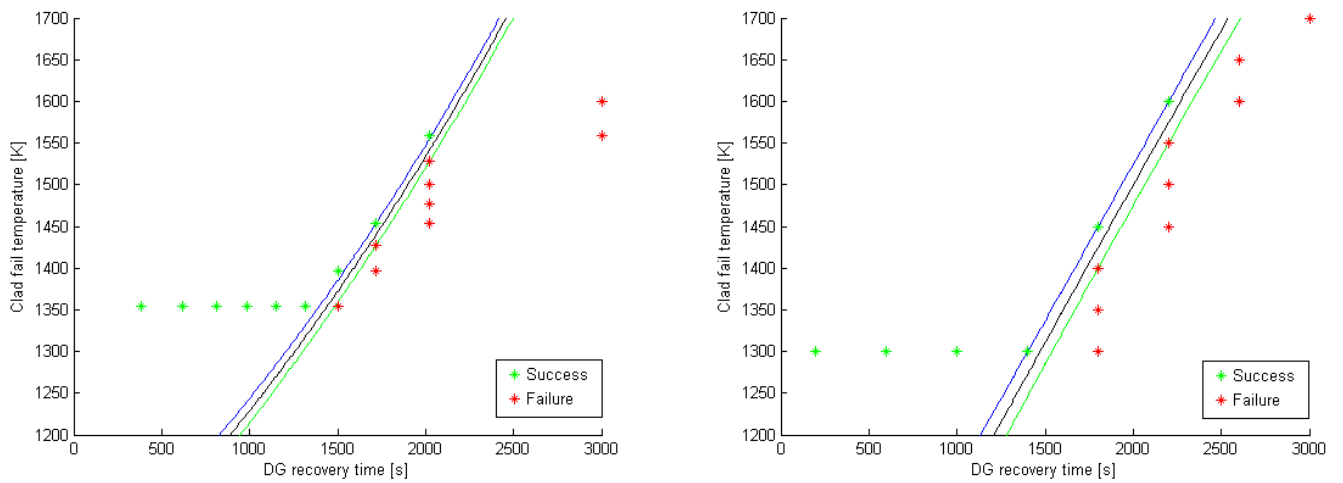

Figure 9: DET Limit Surface: case ESBP DET I(left) and case ESVV DET II (right)

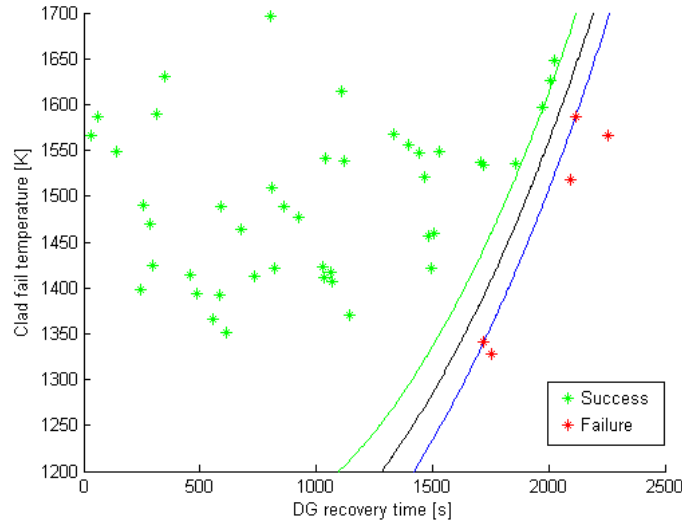A. Alfonsi, C. Rabiti, D. Mandelli, J.J. Cogliati, R.A. Kinoshita

Figure 10: Monte-Carlo Limit Surface.

6 CONCLUSIONS AND FUTURE WORK

This paper presented RAVEN as a tool to perform dynamic PRA through Dynamic Event Tree sampling. In particular, the software structure and all the components that are involved in the computation have been described, including system simulator (i.e., RELAP-7) and the control logic, characterized by monitor system dynamics and on-line control of selected parameters. An example of PRA analysis has been also shown for a SBO scenario for a simplified PWR loop. The description of the implementation for such case demonstrates how the flexibility of the software framework provides state-of-the-art tools to perform Dynamic PRA, uncertainty quantification and plant control. Next capabilities, to be implemented to RAVEN, and currently under development, include adaptive dynamic event tree generation, adaptive sampling [10] and advanced data mining algorithms [11].

REFERENCES

[1] C. Rabiti, A. Alfonsi, J. Cogliati, D. Mandelli, and R. Kinoshita, "Reactor analysis and virtual control environment (raven) fy12 report," Tech. Rep. INL/EXT-12-27351, Idaho National Laboratory (INL), 2012.

[2] C. Rabiti, A. Alfonsi, D. Mandelli, J. Cogliati, and R. Martineau, "Raven as control logic and probabilistic risk assessment driver for relap-7," in *Proceeding of American Nuclear Society (ANS), San Diego (CA)*, vol. 107, pp. 333–335, 2012.

[3] C. Rabiti, D. M. A. Alfonsi, J. Cogliati, and B. Kinoshita, "Mathematical framework for the analysis of dynamic stochastic systems with the raven code," in *Proceedings of International Conference of mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2013), Sun Valley (Idaho)*, pp. 320–332, 2013.

[4] A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, and B. Kinoshita, "Raven as a tool for dynamic probabilistic risk assessment: Software overview," in *Proceedings of International Conference of mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2013), Sun Valley (Idaho)*, pp. 1247–1261, 2013.

[5] D. Gaston, G. Hansen, S. Kadioglu, D. A. Knoll, C. Newman, H. Park, C. Permann, and W. Taitano, "Parallel multiphysics algorithms and software for computational nuclear engineering," *Journal of Physics: Conference Series*, vol. 180, no. 1, p. 012012, 2009.

[6] D. Mandelli and C. Smith, "Integrating safety assessment methods using the risk informed safety margins characterization (rismc) approach," in *Proceeding of American Nuclear Society (ANS), San Diego (CA)*, vol. 107, pp. 883–885, 2012.

[7] A. Hakobyan, T. Aldemir, R. Denning, S. Dunagan, D. Kunsman, B. Rutt, and U. Catalyurek, "Dynamic generation of accident progression event trees," *Nuclear Engineering and Design*, vol. 238, no. 12, pp. 3457 – 3467, 2008.

[8] V. F. Urbanic and T. R. Heidrick, "High-temperature oxidation of zircaloy-2 and zircaloy-4 in steam," *Nuclear Materials*, vol. 251, no. 75, 1978.

[9] D. Anders, R. Berry, D. Gaston, R. Martineau, J. Peterson, H. Zhang, H. Zhao, and L. Zou, "Relap-7 level 2 milestone report: Demonstration of a steady state single phase pwr simulation with relap-7," Tech. Rep. INL/EXT-12-25924, Idaho National Laboratory (INL), 2012.

[10] D. Mandelli and C. Smith, "Adaptive sampling using support vector machines," in *Proceeding of American Nuclear Society (ANS), San Diego (CA)*, vol. 107, pp. 736–738, 2012.

[11] D. Mandelli, A. Yilmaz, and T. Aldemir, "Scenario analysis and pra: Overview and lessons learned," in *Proceedings of European Safety and Reliability Conference (ESREL 2011), Troyes (France)*, 2011.