DEVELOPMENT AND BENCHMARKING OF RAVEN WITH TRACE FOR USE IN DYNAMIC PROBABILISTIC RISK ASSESSMENTS

DEVELOPMENT AND BENCHMARKING OF RAVEN WITH TRACE FOR USE IN DYNAMIC PROBABILISTIC RISK ASSESSMENTS

By KENDALL BONIFACE, B.ENG.

A Thesis Submitted to the School of Graduate Studies in Partial Fulfillment of the Requirements for the Degree Master of Applied Science

DESCRIPTIVE NOTE

Degree: Master of Applied Science

Year: 2021

University: McMaster University

Department: Engineering Physics

Title: Development and Benchmarking of RAVEN with TRACE for use in Dynamic Probabilistic Risk Assessments

Author: Kendall Boniface, B. Eng.

Supervisor: Dr. David Novog

Number of Pages: xi (preliminary), 107 (text)

LAY ABSTRACT

In order to provide a high degree of confidence in the safety of nuclear reactors, it is imperative to understand how each reactor design behaves under both normal and abnormal conditions. Understanding the outcomes of these conditions (in particular, the abnormal conditions) allows for the identification of potential vulnerabilities in the design, and to subsequently ensure that measures are put in place to prevent consequences that could be harmful to the public and environment.

Dynamic probabilistic risk assessment (DPRA) is a field of study that aids in the identification of these abnormal conditions, and quantifies the probability that they could occur. This thesis aims to demonstrate the capability of a DPRA methodology called dynamic event tree (DET) analysis, whereby all possible sequences of events that can occur after an initial event are explored. A major difference between DET and traditional PRA methodologies is that DETs examine the effect of the timing of events whereas the traditional methods simply employ a pass/fail scenario and therefore cannot capture the effect of timing, the role of partially functioning equipment, equipment that has a delayed response and still may alter the consequences of the event, or human intervention at various points in the transient. The DET capabilities of a driver software called RAVEN are first demonstrated for a simple system, and for more complex systems through the development of a specific interface that allows RAVEN to drive TRACE (a simulation tool used to predict the flow and heat transfer that occurs in reactor systems).

ABSTRACT

The identification of potential accident conditions for a nuclear power plant requires a systematic evaluation of postulated hazards, and accurate methods for predicting the behaviour of the system if these hazards were to occur. It is particularly important to identify scenarios which carry severe consequences (e.g., large radioactive releases to the environment), even if the conditions have a low probability of occurrence, so that preventative measures can be implemented.

Dynamic probabilistic risk assessment (DPRA) is a field of analysis that aims to determine the failure pathways of complex systems while simultaneously analyzing the time-evolution of the proposed accident. By studying the dynamics of the system, DPRA methods are capable of analyzing the impact of impaired or late equipment response, human actions during the transient, and the inter relationship between different systems and failures. This approach promotes realistic predictions of the complex response of the system under accident conditions, and for the dynamics of the accident progression to unfold with timing that is not pre-determined by an analyst, thereby removing potential user bias from the results.

The work that is outlined in this thesis was undertaken in order to demonstrate the DPRA software platform called RAVEN, and to leverage its application in the near-future probabilistic assessment of accident conditions applied to CANDU reactor simulation models. Features of the work include:

- Demonstration of the capability of RAVEN to produce predictable results using the dynamic event tree (DET) approach;
- The development of a code interface to allow RAVEN to drive DET simulations of TRACE simulation models; and

- Demonstration of the capability of the developed RAVEN-TRACE interface to produce predictable results for systems that are well-understood.

ACKNOWLEDGMENTS

First and foremost I would like to thank my supervisor, Dr. David Novog, for his constant technical guidance and overall support throughout this project. His insights have been invaluable, both to this project and to my career thus far, and I am very grateful to have had the opportunity to work with and learn from him.

Secondly I would like to thank my family (my husband Andrew, my parents Lisa and Steve, and my sister Hayley), each of whom showed me patience and encouragement, and did not judge me in all the times I needed to stay locked away in my office.

Finally, but not least of all, I would like to thank my co-workers and manager at Canadian Nuclear Laboratories. During the final months of completing this thesis, they carried a heavier workload without any complaint in order to provide me with the time and energy I required to finish my thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

BE	Basic Event
CANDU	CANada Deuterium Uranium
CCDP	Conditional Core Damage Probability
CCMT	Cell-to-Cell Mapping Technique
CDF	Cumulative Probability Function
CET	Continuous Event Tree
DET	Dynamic Event Tree
DPRA	Dynamic Probabilistic Risk Assessment
ET	Event Tree
FT	Fault Tree
GA	Genetic Algorithm
IE	Initiating Event
INL	Idaho National Laboratory
LOCA	Loss of Coolant Accident
MC	Monte Carlo
NPP	Nuclear Power Plant
PDF	Probability Density Function
PDMP	Piecewise Deterministic Markov Process
PRA	Probabilistic Risk Assessment
RAVEN	Risk Analysis and Virtual control ENvironment
RIDM	Risk-Informed Decision Making
RISMC	Risk-Informed Safety Margin Characterization
ROM	Reduced-Order Model
SBO	Station Blackout
SFP	Spent Fuel Pool
TE	Top Event
TRACE	TRAC/RELAP Advanced Computational Engine
UE	Undesired Event
USNRC	U.S. Nuclear Regulatory Commission

1. INTRODUCTION

In order to license and operate a nuclear power plant (NPP), the safety of the facility must be comprehensively demonstrated for both normal operating conditions, and potential abnormal conditions (referred to as accident conditions) [1]. The identification of potential accident conditions requires a systematic evaluation of postulated hazards that might initiate an accident, and accurate methods for predicting the behaviour of the system under these conditions. It is particularly important to identify accident scenarios which carry severe consequences (e.g., large radioactive releases to the environment), even if the conditions have a low probability of occurrence, so that preventative measures can be implemented, either through improved system design, or required procedural actions.

Probabilistic risk assessment (PRA) is a field of analysis that provides quantitative estimates of the risk associated with the operation of complex engineering systems (including but not limited to NPPs). It provides a methodological framework for identifying potential accident conditions, and for quantifying the probability of occurrence of these conditions. PRA methods have been under development for decades and have been applied in the safety studies for numerous reactor designs, providing valuable information for risk-informed decision-making (RIDM) [2]. Specifically, the results of PRA studies can provide both a qualitative understanding of risk-sensitive equipment/vulnerabilities and processes as well as the quantitative acceptance criteria used in the regulatory licensing process [3]. PRA has also been extensively used for improving the design and operation of a plant in order to optimize its economic benefits [4]. Since the Three Mile Island nuclear accident, risk assessment studies have been an integral part of nuclear safety assessments, providing key insights on safe plant operation [5].

The traditional approach to PRA includes the use of so-called static analysis methods, whereby event trees (ETs) and fault trees (FTs) are employed to model the possible combinations of events (failures) that may lead to damaging the reactor core. For ET methods, the different events that can occur on an accident timeline are laid out in sequential fashion and are assigned a probability of action. The probability of a given consequence is then determined by integrating the failure probabilities of each system involved in event sequences leading to the consequence. Probabilities are assigned for complete component failure and no provisions are made to account for partial failures or delays in the timing of equipment actions. In the FT approach, the probability of an undesired system state is determined by analyzing the possible combinations of individual component failures that may lead to the failure of the system. Often the output of FT results are assigned to the probabilities used in some ET action probabilities.

While the traditional methods of PRA have provided significant insight into plant safety over the last 40 years, these approaches are limited as the methods are pass/fail in nature, and hence do not directly account for physical system behaviour or the timing of the events. The order of events is determined by an analyst a priori, and the effects of variation in the timing of these events cannot be explicitly captured. For example, a late action by one system may provide a longer windows for an unrelated system or action, making it more effective in limiting the consequences. In Risk-Informed Decision Making (RIDM) it is important to have the most realistic assessment of failure probabilities in order to establish the optimal path forward, and hence there is increasing demand to provide more sophisticated methods for assessing risk [3]. The most significant methodological shortcomings found within the traditional PRA framework can be summarized as follows:

- In the ET/FT approach, both the order and timing of events (e.g. component failure or operator action) are pre-determined [6]. Realistically, the predicted accident dynamics (and subsequent risk) are sensitive to both the aleatory (representing parameter variation) and epistemic (due to lack of knowledge) uncertainties present within the system model [7] as well as the timing of any actions that may occur. These uncertainties can significantly affect the predicted scenario progression (including the order and timing of events) and the mode and frequency of failure events. As a direct consequence, the quality of the safety assessment performed using traditional methodologies is highly dependent on the ability of the analyst to accurately order and time safety-significant events and to appropriately perturb uncertain input parameters [3].
- Traditional PRA assigns an outcome to a relatively small number of consequence bins, when in reality may be a spectrum of possible consequences. While the binning of consequences greatly simplifies the analysis burden, it does not provide a full representation of the continuous probability of the spectrum of consequences.
- The traditional methods do not provide a measure of the sensitivity of plant risk to safety margins. Instead, they pose a probability of failure without crediting the safety margins inherent in the plant. For example, a given system may provide a large amount of margin to fuel failure during a given event or may be very tolerant of late response, however the system is either assumed to function or fail without consideration of its over-capacity to perform its function. An emerging field of study is on Risk-Informed Safety Margin Characterization (RISMC) which endeavors to demonstrate the relationship between plant risk and safety margins in

> order to support improvements in economics, reliability and sustained safety in long-term plant operation [4]. Thus, operators and regulators would be able to quantify the benefits of each safety margin and assess the safety and economics associated with improving these margins.

Given the advancement of computational models and computing equipment, it has become possible to extend traditional PRA to include more physical representations of the systems and their time-dependency, and to include the effects of human interactions on the system response. To this end, the field of dynamic PRA (DPRA) integrates the time-dependence and physical modeling of each process into the traditional PRA framework, allowing for more realistic predictions of the complex response of the system under accident conditions, and for the dynamics of the accident progression to unfold in an un-biased manner. The ability to explicitly capture the influence of time and process dynamics on scenario progression should, in theory, increase the accuracy of the system model and shift the responsibility away from the safety analyst and toward the method itself [3].

DPRA methodologies aim to provide a consistent, and computationally efficient framework for analyzing the interactions within the system dynamically, while considering both the aleatory and epistemic uncertainties and integrating human factors into the system model [6]. This framework allows for calculations that include the possible spectrum of both partial equipment failure and delayed response for equipment functionality and an analysis of the full range of overall system consequences without predetermined consequences or consequence binning. It also provides a mechanism for assessment of common-mode failures that is more realistic (i.e., the coincidental probability of simultaneous common-mode failures can be assigned a lower probability than the gradual failure of all

like systems). Finally, such methods can accommodate the probability of human intervention at various points in the transient.

Several approaches to DPRA are now under development, however the dynamic event tree (DET) approach is the most widely applied in literature. The principle is very similar to the traditional ET methodology, which begins with an initiating event and models the plant response along all possible pathways the system can take from this event. In the DET approach however, the timing and sequencing of changes to the system are determined by a time-dependent system model, where the branching conditions are controlled via multivariate probability distributions (i.e. they are not pre-set by the analyst). The range of possible input parameter combinations can be sampled using Monte Carlo (MC) or grid-based sampling techniques, and the event branching occurs dynamically based on the operational rules and failure models defined within the simulation environment. This allows for both the epistemic and aleatory uncertainties to be addressed within the same phenomenological and probabilistic framework and leads to a more comprehensive and systematic coverage of the possible event sequences and their probabilities [6].

While the results of DPRA methods have been assessed in the scholarly literature in relation to nuclear power plants (NPPs), to date, all reactor safety studies that are included in the licensing process of reactor designs use some form of a static methodology for risk assessment. In addition to this, there are no applications in the scholarly literature of DPRA methods to Canadian NPPs, and there are currently no RISMC studies available for the CANada Deuterium Uranium (CANDU) reactor design.

The work that is outlined in this thesis lays the groundwork for the near-future application of DPRA (specifically using the DET methodology) to CANDU reactor

accident scenarios. Idaho National Laboratory (INL) has been developing[1] a generic DPRA software framework called RAVEN (Risk Analysis and Virtual control ENvironment): a highly modular platform that provides an avenue for parametric and probabilistic safety analysis based on the response of complex system codes [8]. RAVEN has been selected as an appropriate platform for the application of DPRA within the CANDU reactor context, and this thesis aims to achieve the following goals:

- Demonstrate that RAVEN can generate predictable results using the DET approach. Karanki *et al.* [9] assessed the failure probability of a simple depleting tank system using the DET approach. This problem includes the consideration of both aleatory and epistemic uncertainties, and has an analytical solution for the failure probability of the system. As such, it provides a suitable benchmark to compare RAVEN-generated results against. Note that in this portion of the work, RAVEN is driving simulations of a simple physical system generated with a system model written in Python (i.e., not a complex, validated system code used for predictions of reactor behaviour).
- Develop a code interface that will allow RAVEN to drive simulations using the TRAC/RELAP Advanced Computational Engine (TRACE) code. TRACE is a best-estimate reactor system code developed by the U.S. Nuclear Regulatory Commission (USNRC) for modeling thermalhydraulic phenomena in reactors, and may be used for predictions of CANDU reactor behaviour. Coupling any code to RAVEN requires an in-depth understanding of the mechanics of the code being driven, and careful implementation of methods to perturb, stop and restart the code.

[1] Development on RAVEN began in 2012 [6]

- Demonstrate that the developed RAVEN-TRACE interface can generate results consistent with other studies in the scholarly literature. Chraibi *et al.* [10] assessed the reliability of a simplified model of a spent fuel pool (SFP) using a DPRA toolkit called PyCATSHOO. Unlike the depleting tank problem, the SFP system includes a thermal component due to the decay heat of the fuel, and includes a relatively large number of component and system failures that may occur and lead to undesired system end-states.

2. LITERATURE REVIEW

2.1 The Mathematical Framework for PRA Methodologies

In order to effectively compare the different analysis methods available in the field of PRA, it is important to understand the formulation of the mathematics from which the methods originate. The theory of probabilistic dynamics is built on the fact that the deterministic evolution of a system can change trajectory as a result of events that can be described by probabilistic laws [11]. With regard to reactor dynamics, these changes can be caused by events initiated by the control and safety systems, operator actions or the failure or malfunction of components. Specifically, probabilistic reactor dynamics aims to compute the probability that the state of the reactor will be in a given domain at a given time [11].

The mathematical formulation begins with the definition of the state vector $\bar{x}$, which stores the system process variables. $\bar{x} \in \mathbb{R}^N$ where N is the number of variables. For simplicity, the software models used for system control (including the human operators) are considered to be multi-state components [3]. These components can then be considered in the same manner as the physical system hardware components, yielding α combinations of possible configurations of the

global system (each with an index i where $i = 1,2,\cdots,\alpha$) [3]. Within each system configuration i, $\bar{x}$ obeys a set of differential equations describing its dynamics [3]:

$$\frac{\partial \bar{x}}{\partial t} = \bar{f}_i(\bar{x}, t) \tag{1}$$

The probability that the system can be found in state $\bar{x}$ and in configuration i at time t is described by the probability density function $\pi(\bar{x}, i, t)$. The distribution is normalized as [3]:

$$\sum_{i=1}^{\alpha} \int_{\mathbb{R}^N} \pi(\bar{x}, i, t)\, d\bar{x} = 1 \tag{2}$$

The total transition rate out of configuration i given that the process variables are described by $\bar{x}$ is $\lambda_i(\bar{x})$ and the total transition rate from configuration j to i given $\bar{x}$ is $p(j \to i|\bar{x})$ such that [3]:

$$\lambda_i(\bar{x}) = \sum_{j \neq i} p(j \to i|\bar{x}) \tag{3}$$

In the basic formulation, it is assumed that the systems under investigation are Markovian in nature, however in reality, the physical systems are semi-Markovian [3]. In some cases it is possible to use supplementary variables to convert a problem to a Markovian form if necessary [11], but for now the Markovian assumption will be followed. One of the main properties of a Markovian system is described by the Chapman-Kolmogorov equation [11]:

$$p(\bar{x}_3, t_3|\bar{x}_1, t_1) = \sum_{\bar{x}} p(\bar{x}_3, t_3|\bar{x}, t_2) p(\bar{x}, t_2|\bar{x}_1, t_1)\,, \quad t_1 < t_2 < t_3 \tag{4}$$

This states that the transition probability from a state vector $\bar{x}_1$ at time t_1, to $\bar{x}_3$ at time t_3 is given by the sum of the probabilities of all possible paths through the state space $\bar{x}$ that connect $\bar{x}_1$ to $\bar{x}_3$. This property allows for a generic mathematical description of the probability of moving from any system configuration to another. Within the framework of Eq. (4), the probability distribution of the entire system can be described by [3]:

$$\frac{\partial \pi(\bar{x}, i, t)}{\partial t} + div\left(\bar{f}_i(\bar{x}, t)\pi(\bar{x}, i, t)\right) + \lambda_i(\bar{x})\pi(\bar{x}, i, t) = \sum_{j \neq i} p(j \to i|\bar{x})\pi(\bar{x}, j, t) \quad (5)$$

An integral formulation of the problem can also be written which illustrates that the probability density at point $(\bar{x}, i, t)$ is the sum of two contributions [3]:

- The scenario in which the system has been in configuration i for the entire transient (up to time t) without any transition (its entire trajectory is deterministically governed by the dynamics of Eq. (1) for system configuration i); and
- The summation of all the transitions from other system configurations to i taking place before time t.

The goal of the probabilistic approach to risk assessment is to use the information in Eq. (5) to determine the probability that the system process variables will exit a bounded safety region. Consequently, in addition to calculating the probability that the system will be in a given state at a given time, a systematic way to define and characterize the safety region (denoted by D) is required. The borders of D are determined by the union of a set of hyperplanes in the system state space and can, in some cases, require specific analysis methods to discover their coordinates [3].

2.2 Available PRA Methodologies

All PRA methodologies attempt to provide estimates to the solution to Eq. (5) and account for both the aleatory and epistemic uncertainties in the prediction of the distribution of risk associated with the system response. Attempts have been made to provide an analytical solution, however these can be extremely complex even for systems with a small number of possible component states [3]. It is easy to see that for complex systems such as NPPs (i.e. with a large values of α and N), a complete determination of $\pi(\bar{x}, i, t)$ can be impractical. Moreover, the safety characteristics of particular interest in these large systems are usually contained in a small portion of $\pi(\bar{x}, i, t)$ [3]. All existing solution schemes provide practical methods for estimating the distribution of $\pi(\bar{x}, i, t)$ by discretizing either the time or process variables[2] (or both) [3]. The following sections provide descriptions of the most prominent PRA methodologies. It is not an exhaustive list of the existing techniques, but addresses the most commonly cited (and by extension, most commonly used) methods in recent literature.

2.2.1 Traditional PRA – Event and Fault Trees

The ET/FT approach has been used in the past for nearly all PRA studies in the nuclear industry. Both techniques provide a logical avenue for identifying undesired failure events and modeling the possible combinations of failures that may lead to core damage states. They allow for a qualitative representation of the system and its possible scenarios, and a quantitative characterization of the risk associated with these scenarios [3].

[2] It is of note that this discretization refers to the numerical solution method in the PRA framework and not the necessary discretization of variables that occurs when using computer codes to solve for process dynamics [2].

The FT analysis method is an approach used to describe the paths that can possibly lead to specific outcomes. The basic components of the system are arranged using Boolean logic gates which encompass the probability, priority and criteria for the occurrence of basic events (BEs) with the intent of determining all possible combinations that could lead to some final event (or top event – TE) [5]. The probability distributions associated with each BE are propagated upward through the tree logic to obtain probability distributions of the TEs. Figure 1 shows a basic FT to illustrate the concept. Some of the important results obtained from FT analysis include cutset and minimal cutset (the smallest combination of component failures which, if they all occur, will cause the TE to occur) probabilities and quantitative rankings of component failure sequences to that lead to system failure.

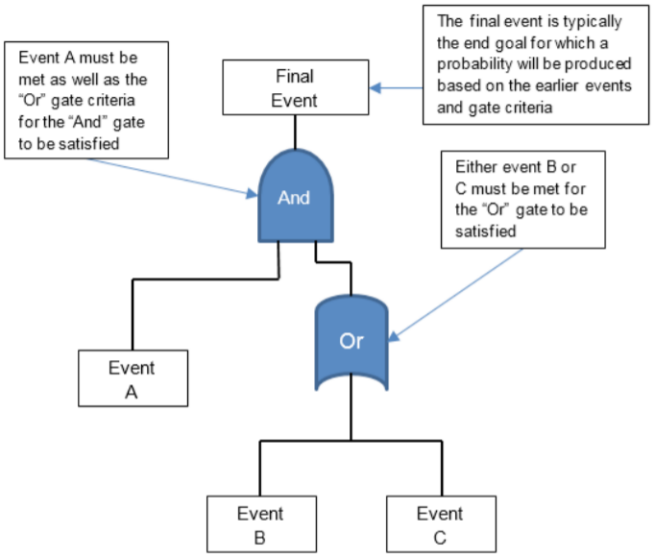

Figure 1: Illustration of the Basic Boolean Logic Concepts used in the Fault Tree Analysis Method [5]

The ET approach begins with initiating events (IEs) and models the possible pathways the system can take from this event. It requires the temporal ordering of potential events within the system and the system response corresponding to each possible sequence can then be simulated. Figure 2 shows the typical structure of an ET describing a system with three safety systems that can potentially fail and affect the event progression [12]. The applied timing of events is often based on timing obtained from typical transients [11]. The overall probability of occurrence for each system path can then be calculated and the final state of the system (either safe or failed) is used to characterize the risk.

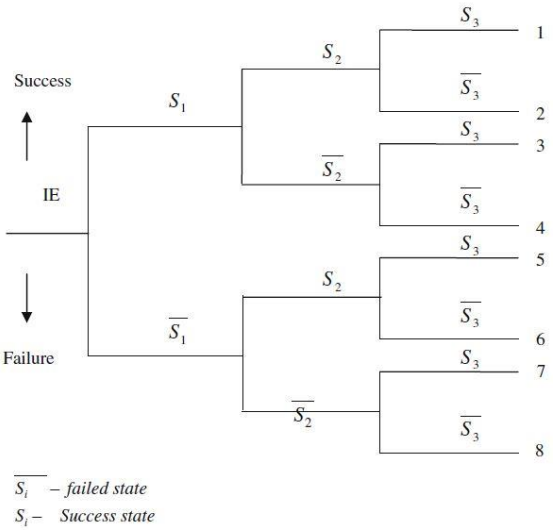

Figure 2: A Typical Event Tree with Three Safety Systems [12]

When using ET analysis, the quantification of risk requires grouping sequences together based on the definition of bounding success criteria [13]. A direct consequence of this is the possibility for introducing false conservatism in the PRA analysis or the potential for underestimating risk [14].

The uncertainty in ET/FT analysis is typically determined using either MC or grid-based sampling techniques applied to the input parameters. Sensitivity analysis is performed by either changing the probabilities associated with components and BEs, or by changing the qualitative configurations (changing event ordering, removing components, etc…) [3].

It is important to note that the ET/FT analysis framework is static – there is no explicit modeling of the time-dependence of significant events except that which is pre-determined and set by the analyst. It has also been shown that the traditional PRA methodology has difficulty accounting for the sensitivity of the system response to uncertainties in the process physics (e.g., lack of knowledge in how well we can predict the response of the system or action) [6]. The burden of proof of quality of the analysis then rests heavily on the ability of the analyst to accurately understand the time-dependence of the system and structure the PRA in an appropriate manner, and to appropriately perturb uncertain parameters [3] to understand the uncertainty in the PRA outcome.

2.2.2 Dynamic PRA

The main objective of risk assessment within the DPRA framework is to account for the effects of both aleatory and epistemic uncertainties simultaneously by including time-dependence in the model of the system [6]. Depending on the choices made by the analyst in setting up the static PRA, it may produce the same results as the dynamic ones, or they may differ (i.e., there may be some user effect). However, if a high-fidelity transient plant computational model is available, then the DPRA results should be analyst independent. This means that there is potential for improvement in scenario modeling (and the subsequent risk estimate) with the incorporation of dynamics in the solution method [3].

Several approaches to solving the probabilistic dynamics equations have been developed and are described briefly below. However the following requirements are common to all of the dynamic techniques [6]:

- A time-dependent system model;
- A definition of the possible system configurations (both normal and abnormal);
- The probabilities associated with transitions between system configurations; and
- Criteria for both system success and system failure.

2.2.2.1 Continuous Time Methods

The solution methods described in this section are continuous in time. They can be useful and accurate for the analysis of simple systems, however they become extremely computationally intensive as the systems become more complex [6]. Both methods are also limited due to their assumption that the transitions between states are instantaneous and that the corresponding system dynamics react immediately to the transitions. In reality, there may be delays that are significant to the evolution of the physical behaviour of the system and cannot be ignored. Mitigating techniques exist in which an additional index is included in the system to represent stimuli, allowing each configuration of component states to be represented by both the physical configuration and the stimulus index. This allows for the time delays to be explicitly accounted for in the model, however the complexity (and corresponding computational effort) is only increased [6]. In addition to these limitations, the continuous-time methods require system-

specific algorithms and so their application to large-scale practical problems is difficult [12].

Continuous Event Trees (CET)

In the CET method, all possible dependencies of transition events are modeled by a set of integral equations under the Markovian assumption [12]. This method is the realization of the solution to Eq. (5) which is generally found using MC techniques. Once the probability density function has been solved for, the statistical properties (PDFs, CDFs, and expected transition times) of all branches of the system are known and can be used for risk assessment purposes [6].

An illustration of the difference between an ET and CET is shown in Figure 3. A typical, time-ordered ET is shown on the left with transitions appearing at predetermined times. A CET is shown on the right for a simple system with one process variable ($N = 1$) and three possible system configurations ($\alpha = 3$). The horizontal trajectories along x are deterministic and the vertical jumps between configurations are stochastic. The transitions can occur at any time, and the initial condition of each trajectory is the final condition of the one corresponding to the previous state [11]. It is also of note that both the static and dynamic ET methodologies can be shown to be special cases of the CET model [6].

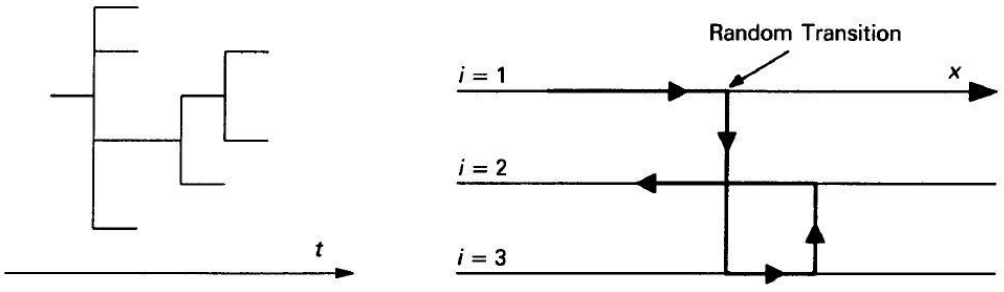

Figure 3: Illustration of the Difference Between an Event Tree (left) and a Continuous Event Tree (right) [11]

A significant incapability of this method is its inability to account for system configuration changes due to control and safety system setpoint crossings (e.g. the opening of a valve due to an overpressure setpoint) [6].

Continuous Cell-to-Cell Mapping Technique (CCMT)

Continuous CCMT is a version of the CET methodology that allows for the transitions that should occur during setpoint crossings [6]. Similar to the CET approach, a continuous Markov process provides the fundamental mathematics, however the physical process variables are partitioned by the user into "cells" [3]. Within a cell, the evolution of the system is modeled just as in CET. The transition probabilities between states are then obtained from a user-defined system model, allowing more variability in the possible system states [6].

2.2.2.2 Discrete Time Methods

The solution methods described in this section are discrete in time.

Monte Carlo (MC) Simulation

The MC simulation methodology is conceptually the most simple of the DPRA techniques. The time at which each of the components or safety systems of interest fails is randomly chosen, and a deterministic model is then used to simulate the system behaviour for that sequence, resulting in an end state that can be compared against success/failure criteria [6]. This process is repeated until sufficient statistics have been achieved, and the probability of consequences of interest can be calculated.

One of the main advantages to this technique is that it is insensitive to the size, complexity and modeling assumptions [12]. However, this method requires a large number of simulations in order to identify low probability, high

consequence events, and can consequently be impractical for certain applications [14]. Hence, computational improvements or reduced-order modelling approaches are often required.

Markov/Cell-to-Cell Mapping Technique

The Markov/CCMT method is similar to the continuous CCMT approach except for the discretization of time [6]. It regards the dynamic evolution of the system in discrete time as the probability of transitions between state space intervals or cells. The technique yields a Markov model that can be used for both inductive tracing of fault propagation (IE to TE) and deductive tracing of fault evolution (TE to IE) and for calculating top event probabilities [6].

Genetic Algorithms

Genetic algorithms are a much more recent addition to the DPRA techniques. The goal is to facilitate intelligent and adaptive resolution in exploring the system scenario space [6] and identifying the safety vulnerabilities of the system [14]. As with other DPRA techniques, the genetic algorithm engine is used in conjunction with a system simulation code. Pre-determined safety critical parameters (safety limits) are then used by the genetic algorithm to guide the selection of input parameters to be used by the system code [14]. The algorithm is designed to take advantage of clues within the results of initial simulations to narrow down the search for potentially vulnerable states. It is not intended to provide more comprehensive or accurate values in the risk assessment process, but rather to provide a computationally efficient method for identifying potential system vulnerabilities that require more detailed exploration [14].

Dynamic Event Trees (DET)

DETs are by far the most widely used dynamic approach to the risk assessment of NPPs in literature [12]. The principle is very similar to the classical ET methodology, however the timing and sequence of changes to the system are determined by a time-dependent system model. The analyst now controls the branching conditions via multivariate probability distributions, and the timing at which the branching occurs is not static (unlike in the ET approach). Branching is then able to occur dynamically and both epistemic and aleatory uncertainties can be addressed within the same phenomenological and probabilistic framework. This leads to a more comprehensive and systematic coverage of the possible event sequences that can arise due to the perturbation of uncertain parameters in the input space [6]. While there is now additional potential for user-effects in the analysis (for example by selecting branch times or conditions), there is an opportunity for each user to explore the results with respect to these inputs and demonstrate a converged solution.

2.3 Dynamic PRA – Application to NPPs in Literature

Several approaches to DPRA are now in a mature stage of development, however relatively few studies have been performed that comprehensively assess what these techniques can offer in comparison to the results of classical PSA. It is stated in Labeau *et al.* [3] (a highly cited paper providing a comprehensive review of dynamic reliability methods and their benefits) and then again in both Aldemir [6] (a survey of DPRA methodologies) and Karanki *et al.* [13] that literature studies on the quantification of risk using dynamic analysis and its comparison to traditional PRA methodologies are limited in both number and scope.

To date, several studies have been published that demonstrate the potential of DPRA concepts by identifying events in an accident sequence that are of high significance to the final quantification of risk. Several of the most frequently cited works presenting DET analysis of NPP transients leading to core damage are:

- Acosta *et al.* [15]: an early study that addresses the treatment of stochastic variations in the operating staff capabilities and concludes that the DET framework can reasonably represent important operator behaviour patterns.
- Izquierdo-Rocha *et al.* [16]: uses the DET approach on a steam generator tube rupture (SGTR) scenario of a Pressurized Water Reactor (PWR) to identify emergency operating procedure (EOP) improvements and demonstrate the feasibility of the DPRA methodologies.
- Kloos *et al.* [17]: uses DET analysis to simulate the progression of a PWR station blackout (SBO), demonstrating the possible spectrum of event sequencing that is otherwise overlooked by the traditional methods.
- Hakobyan *et al.* [18]: presents the results of a PWR SBO simulation using the DET approach and shows that the D-PRA results in a wider variety of accident scenarios by providing a more phenomenologically consistent framework.

These applications of the DET methodology provide useful information pertaining to the dynamics of possible accident scenarios, however they do not report on the final quantification of risk and still lack a thorough and systematic comparison of the accuracy of the dynamic approaches to the traditional ones. It is also significant to note that all of these studies use a binary approach to represent the possible event sequencing in an accident scenario: while the

events are allowed to occur dynamically, they are represented as either the success or failure of a system or operator action, and the potential for partial failure is not considered. This can be significant in the scenario progression where the partial operation of some systems may significantly impact the event trajectory.

More recently, Lee *et al.* [19] published the results of a case study examining the feasibility of DPRA using the DET approach on an SGTR accident in an OPR-1000 NPP. Conditional core damage probabilities (CCDPs) are calculated and a comparison is made between the risk quantified using both the dynamic and conventional static PRA strategies (the results of which are cited from a previous study). However, meaningful discussion comparing the two results is difficult due to the fact that there were a significant number of differences employed in both the system simulation model and the assumptions regarding the progression of the accident.

Karanki *et al.* [13] presented a case study on a DET examination of the effects of a medium break loss of coolant accident (LOCA) on the Zion NPP (now decommissioned). A comparison was made between the results obtained using the DET method and those estimated with a traditional ET model for the scenario progression. This comparison provides some illustration of the potential improvements in the estimate of risk that are possible with dynamic methodologies. The paper particularly highlights the benefits to bypassing the consequence binning rationale used in traditional PRA methods and reports a significant difference in the importance of operator action on the progression of the accident relative to traditional methods (the DET results show in some cases a decrease from traditional PRA in the importance of operator action).

Figure 4 shows a comparison of both the CCDP and the core damage frequency for the LOCA scenario for three ranges of break size. The CCDP results for the

smallest and largest break sizes are significantly higher using traditional methods. This has been attributed to conservatism introduced by the imposition of bounding pass-fail success criteria. For example, using traditional methods it was found that the bounding success criteria for the time window for switching from high-pressure injection to recirculation when the coolant reserve is low under the smallest break size condition is 1050s and is associated with the case that 1 of the total 4 available high pressure trains are available. When this criterion is determined, it is no longer deemed necessary to address the cases where more than one of the high-pressure trains are available. Since it is known that the number of high-pressure trains (and other system variables) can greatly affect the dynamics of the system and therefore affect the progression of events, the DET method includes simulations that consider all cases. It is then shown that when all 4 high-pressure trains are active, the time window for switching to recirculation is 1900s. The traditional PRA results for the CCDP under this break size range do not allow for the possible avoidance of core damage due to additional trains being available, and result in a conservative estimate of the risk [13].

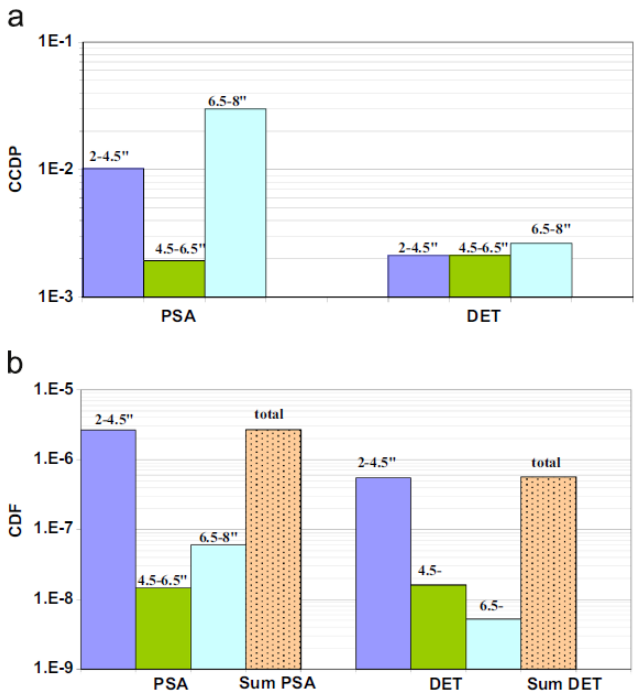

Figure 4: Quantification of Traditional and DET Methods for Several LOCA Scenarios [13]

The current literature suggests that dynamic reliability methods have the potential to yield more accurate descriptions of complex system responses in accidence scenarios, however their use in industrial applications heavily depends on a more comprehensive examination and validation of the methods.

A DPRA study was performed on an SBO scenario using RAVEN applied to a simplified PWR system as shown in Figure 5 [20]. It is of note that RAVEN is a key tool and feature of this thesis, and its architecture is described in detail in Section 3.1.1.

The sequence of events starting from a steady-state operational condition are:

1. Loss of grid power and immediate reactor shutdown (decay heat power, and unavailability of diesel generators and residual heat removal system).

2. Subsequent pump coast-down.
3. Possible recovery of diesel generators.

The transient ends when either the clad temperature reaches its failure threshold or the simulation reaches a prescribed end time (system success).

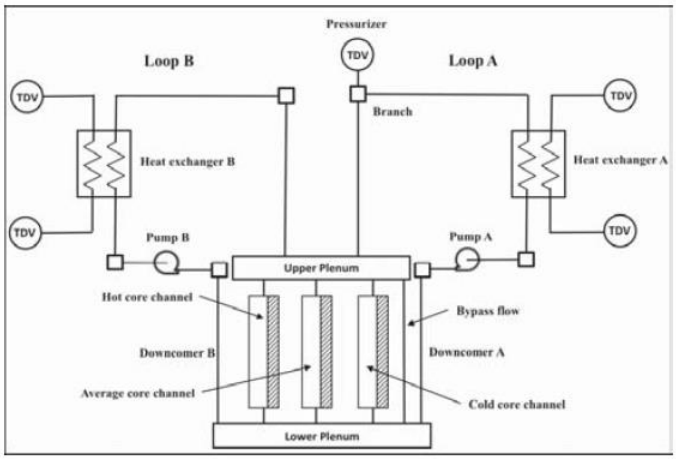

Figure 5: Simplified PWR Schematic Examined using the DET Approach in RAVEN [20]

Two branching triggers were examined in this study:

- The recovery time of the diesel generators (normal distribution, μ = 1100 s, σ = 700 s, min = 3 s, max = ∞); and
- The clad failure temperature (triangular distribution, peak = 1477.59 K, min = 1255.37 K, max = 1699.82 K).

Two different sampling strategies (each with its own corresponding DET) were used. These strategies are shown in Figure 6 and described below:

- ESBP: Branching probability thresholds that are equally spaced on the CDF of each variable; and

- ESVV: Branching probability thresholds corresponding to equally spaced variable values.

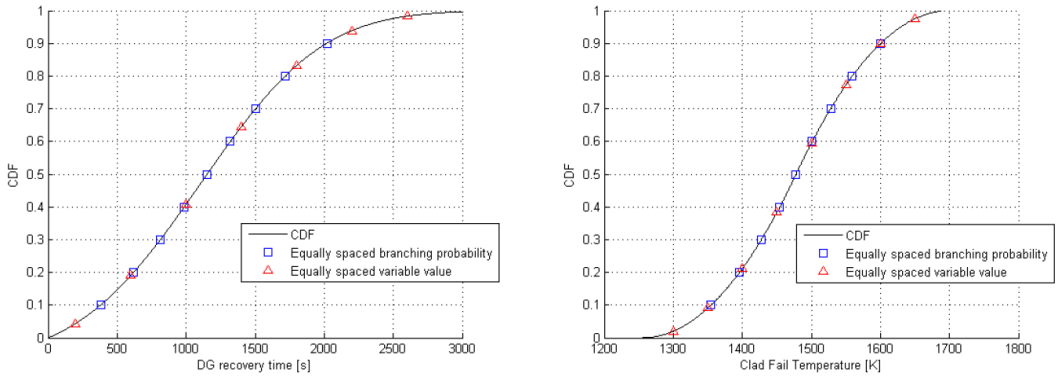

Figure 6: Branching Probability Thresholds for the SBO DET Analysis using RAVEN [20]

Figure 7 shows the temporal evolution of the clad temperature for DETs corresponding to the two different sampling approaches. It is significant to note that a different number of branching thresholds were used in each case, which is the reason that a different number of scenario progressions (or branches) are observed. The evolution of the clad temperature behaves as expected: the temperature rises after the transient begins, until either the diesel generator power is restored, or the clad failure temperature is reached. Both the generator restoration time and the clad failure temperature are sampled parameters, and as such, the success or failure of the system overall depends on the current branch of the DET (or the combined effect of the two parameters).

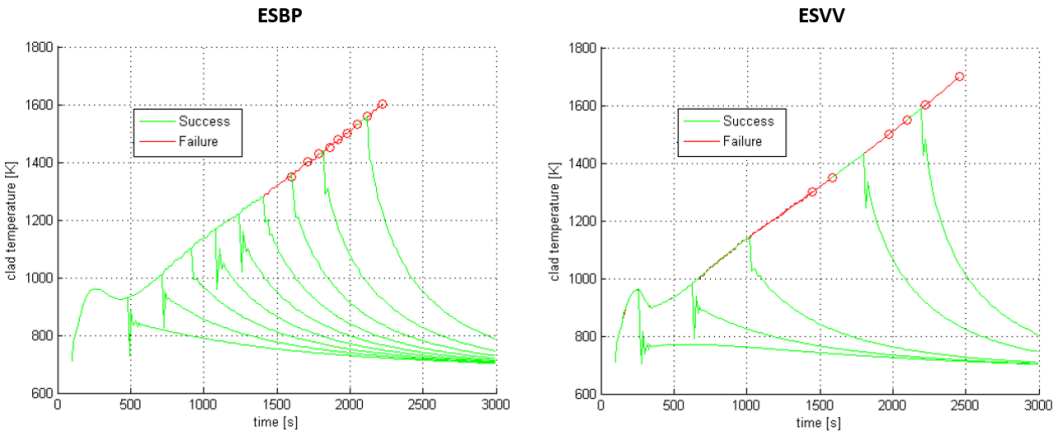

Figure 7: Temporal Evolution of Clad Temperature using the DET Method in RAVEN and Two Different Sampling Strategies [20]

The results reported in [20] demonstrate the capability of RAVEN to perform the mechanics required to drive DET simulations for perturbing the input space of a system described by a complex thermalhydraulics system code.

In 2013, Sandia National Laboratories (SNL) and INL published a report outlining the potential for using both RAVEN and DAKOTA within the same DPRA framework [21]. DAKOTA is a generic software platform that provides a systematic and rapid means of improving or optimizing designs and for understanding the sensitivity and uncertainty associated with simulation-based models. The code is not specific to the nuclear industry and features a high degree of ease and flexibility for interfacing to simulation codes for the purpose of performing the following types of analyses [22]:

1. Parameter studies – exploring the effects of parametric changes within a simulation model.
2. Design of experiments – similar to parameter studies, but with the primary goal of generating thorough input space coverage to ensure good experimental techniques.

3. Uncertainty quantification – uses specified input parameter probability distributions to compute the respective system output response functions.
4. Optimization – uses constraints on input variables or secondary simulation responses to either minimize cost or maximize system performance.
5. Calibration – finds the best agreement between simulated data and a desired output (experimental data).

Each of the methods and algorithms employed by DAKOTA requires an interface to a user-supplied simulation code. Figure 8 shows a simplified, black-box schematic of the code interface and processing scheme used in DAKOTA analyses. Similar to RAVEN, DAKOTA acts as the control logic driver for the simulation code, with user-defined inputs, simulation architecture and monitoring of outputs. One of the most prominent features of this code is the simplicity of its interface structure and its native ability for managing concurrent executions of a computational model in parallel [22].

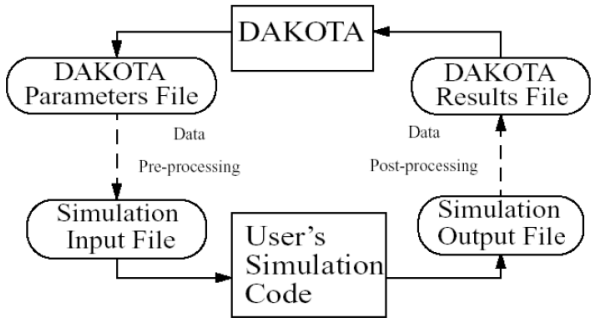

Figure 8: DAKOTA Control Logic Schematic [22]

The SNL/INL study was motivated by the desire to use some of the reliability methods native to DAKOTA in conjunction with the highly developed DPRA simulation architecture native to RAVEN. In order to show the capabilities of this framework, a DPRA study was performed on the same SBO scenario described previously (see Figure 5).

The feature of this study is the software interface developed between DAKOTA, RAVEN and the system code (in this case, RELAP-7), and the inclusion of calculated probabilities of the event sequences. Figure 9 shows a schematic of the framework in which RAVEN was configured to receive data from DAKOTA, spawn RELAP-7 simulations, then feed the results back to DAKOTA for subsequent parameter perturbation in the next iteration [21].

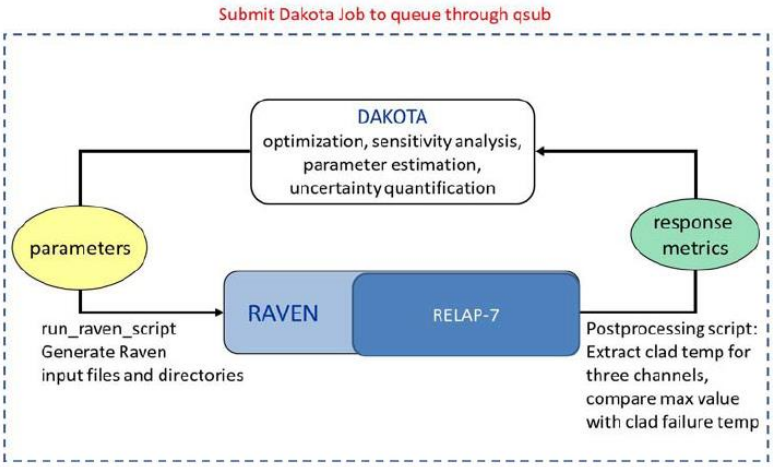

Figure 9: DAKOTA Interface to RAVEN/RELAP-7 [21]

The results of this study include a comparison of the system failure probability using four approaches:

1. (RAVEN) 500 Latin Hypercube samples used to determine the reference failure probability
2. (DAKOTA) Local Mean Value Method in which a function evaluation is taken at the mean of the input variables, then two more function evaluations (one in each direction of the uncertain variables) to calculate the derivatives.
3. (DAKOTA) Local Advanced Mean Value Method in which the same function evaluations are performed, followed by a Taylor series expansion that is used in an optimization scheme to generate an additional point for presumed higher accuracy.
4. (DAKOTA) Global Reliability in which six function evaluations are performed, followed by the addition of consecutive points to refine the prediction until a convergence criterion is met.

Table 1 summarizes the results of the four applied methodologies. It can be seen that the methods perform well with a small number of function evaluations. It is significant to note the linearity of the limit state function under analysis and that divergence from this linearity will affect the accuracy of the failure probability estimate [21]. However, this study demonstrates the potential for computational savings if DAKOTA reliability methods are employed within the DPRA framework.

Table 1: Estimated Failure Probability Using Different Reliability Methods

Method	**Number of Function Evaluations**	**Failure Probability Estimate**
Latin Hypercube Sampling	500	0.13
Local Mean Value Method	3	0.13006
Local Advanced Mean Value Method	4	0.14023
Global Reliability	27	0.13211

In 2016, Électricité de France (EDF) presented work at the 13th International Conference on Probabilistic Safety Assessment and Management conference on DPRA applied to a SFP scenario using a PRA toolkit called PyCATSHOO (PythoniC Object Oriented Hybrid Stochastic AuTomata) [10]. The goal of this toolkit is to provide a generic platform for modeling and assessing the performance of dynamic systems that include both deterministic and stochastic behaviour and features the integration of these two behaviours at a more central level than both DAKOTA and RAVEN [10]. Its framework is based on a representation of the Piecewise Deterministic Markov Process (PDMP) which is a well-known mathematical formulation capable of modeling the evolution of deterministic systems that experience discrete, stochastic events [23]. PDMP modeling of a dynamic system is defined by 1) the ODEs that describe the deterministic system behaviour, 2) the event rates that govern the frequency of occurrence of state transitions, 3) the probability distributions that govern the likelihood of changes of state, and 4) the initial and boundary conditions of the system [23]. It follows that the PDMP modeling of large and complex systems currently faces difficulty due to the potentially vast and multifaceted nature of the governing system of equations, however preliminary studies have shown the capabilities of PyCATSHOO in assessing the reliability of several thermalhydraulic systems [24].

PyCATSHOO can be used in either a Python or C++ environment and is, at its most fundamental level, a dynamic library with three categories of tools [24]:

1. Generic modeling tools for building libraries of classes for the components of a system.
2. Instantiation tools for building specific systems from the generic classes and for the customization and initiation of parallel simulations.
3. Post-processing tools for the analysis of simulation results.

The individual components of a system are linked in a user-defined manner based on the architecture of the physical system. MC simulations are then performed, and for each simulation, the equations of the physical phenomena are solved and the undesired states of the system can then be analyzed [24]. Figure 10 shows a schematic of the generic PyCATSHOO system implementation process. The first step involves the creation of a model for every individual system component. This model defines the purpose of the component (i.e. its governing equations), the explicit rules that tell the code which variables should be inputs and outputs to the rest of the system, and should include the behaviour of the component under both functional and dysfunctional conditions. The second step uses the generic classes built in Step 1 and instantiates them based on the specific composition and architecture of the physical system. The third step creates a specific simulation model based on scenario-specific input data and a user-defined set of indicators that are important to the final safety assessment.

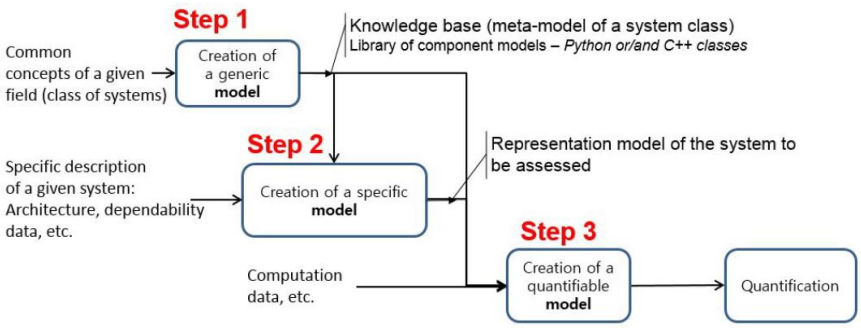

Figure 10: Schematic of General Implementation Process for PyCATSHOO [24]

The 2016 EDF study aims to demonstrate the capabilities of PyCATSHOO by comparing its results to a traditional PRA approach and highlighting the

importance of taking into account the inertia of physical phenomena within the same computational framework as the PRA analysis. A high-level description of the method and results are presented in this section; however it is significant to note that further details are presented in Section 3.2.2 for the purpose of benchmarking against this work.

The physical system is comprised of a SFP and three cooling trains with a large number of components and safety systems that are subject to failure (valves, pumps, heat exchangers, components of the overall cooling water system etc.). Each of these components or safety systems are given a specified failure rate, and in some cases, a specified repair rate.

Three initiating event scenarios were examined: a 100-hour non-repairable loss of ultimate heat sink (LUHS), a 192-hour loss of off-site power (LOOP), and a 7846-hour mission where the system is subject to only the component and system failures, without a specific initiating event.

The transients are monitored for the following unwanted events (UEs):

- UE0: Simultaneous loss of all three trains (classical approach)
- UE1: Water temperature rising above 80°C (safety margin for boiling)
- UE2: Water temperature rising above 100°C
- UE3: Water level dropping below 16 m (fuel uncovery)

Table 2 shows the probability of occurrence of each of the unwanted events calculated within the PyCATSHOO framework for each of the three initiating scenarios. The most notable findings of this study are:

- The hierarchy of the magnitude of failure probability in the three scenarios is consistent with what is expected. C3 has the lowest failure

probability (even though its observation time is the longest) because in this context, LUHS and LOOP events are repairable.

- The general order of magnitude difference between the criteria calculated by classical methods (UE0, or the loss of all three cooling trains) and the other three unwanted system events is also notable. In particular, the probability of occurrence of the loss of three trains is, in two cases, two orders of magnitude higher than the occurrence probability of fuel uncovery. This finding is consistent in current literature and highlights the potential for the introduction of unnecessary conservatism when using classical PRA methodologies.

Table 2: Estimated Failure Probabilities [10]

Scenario:	C1	C2	C3
Unwanted Event			
UE0	1.16e-2	1.41e-2	5.09e-3
UE1	6.27e-3	7.78e-3	2.00e-3
UE2	4.31e-3	5.26e-3	8.90e-4
UE3	1.01e-3	7.30e-4	2.00e-5

3. METHODOLOGY, CODES & MODEL DESCRIPTIONS

This section contains all of the relevant information pertaining to the methodology employed in this research.

The high-level goal in this research has been to demonstrate the capability of the DPRA software called RAVEN for near-future application to DPRA studies of CANDU systems, by using it to drive simulations using an industry-standard thermalhydraulics code called TRACE. In this section, RAVEN and TRACE are first

described, followed by a description of a code interface that was developed as part of this research for the purpose of passing RAVEN and TRACE inputs and outputs between the two codes. Finally, each of the systems that are assessed using RAVEN and the DET approach in this research are described, along with the corresponding implemented simulation models and overall methodological approach taken for each.

3.1 Codes

3.1.1 RAVEN

The RAVEN software platform is a probabilistic-based scenario simulation code that has been developed within the Nuclear Energy Advanced Modeling and Simulation (NEAMS) program [8]. The initial development was focused on allowing for DPRA capabilities for RELAP-7 within the MOOSE framework. Currently however, RAVEN stands as a multipurpose DPRA software platform that can act as the control logic driver for any system code and can be used within the MOOSE framework or as a stand-alone toolkit. It is designed to examine uncertain input parameters and the corresponding system code response using either MC-type sampling, dynamic branching strategies, or a combination of both.

Figure 11 shows a schematic representation of the RAVEN computational framework and the communication among the different, user-definable modules. The main software components are the database manager, the graphical user interface, and the probabilistic and parametric framework (the core of the RAVEN analysis capabilities). The code is characterized by a high degree of modularity and flexibility, allowing custom analysis flow at run-time

through a set of basic components and algorithms available to the user [8]. The user can control:

- The nature and complexity of the RAVEN/system-code interface;
- The probability distributions applied to uncertain system parameters;
- The sampling technique used to perturb the input space;
- The generation of reduced order models (ROMs), allowing for lower computational cost by prioritizing areas of the input space that need to be explored; and
- The post-processing and analysis of the results (including uncertainty quantification and propagation capabilities, sensitivity analyses, and customizable plotting tools).

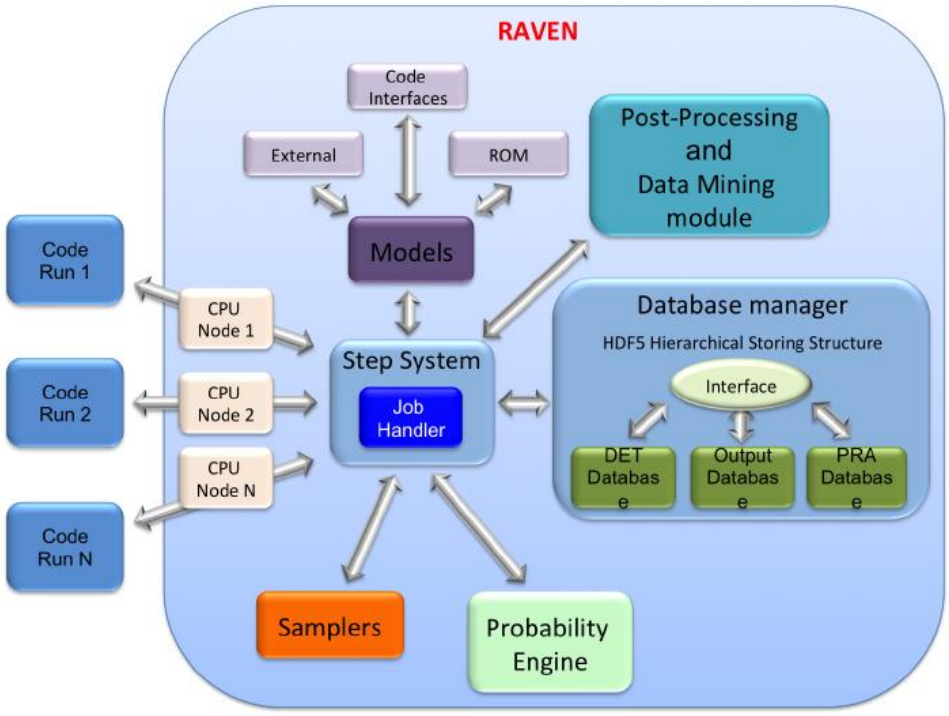

Figure 11: Schematic of the RAVEN Code Architecture [25]

RAVEN has two features embedded in its architecture that are worthy of highlighting. The first is its sophisticated logic for DET execution. In order to avoid unnecessary simulation time, it is highly prudent for a DPRA software platform to take advantage of the restart capabilities of a system code and save the history of the system evolution up to the point where branching conditions occur during a DET simulation. In line with this, the RAVEN framework also includes a logical and systematic methodology for returning to branching points in order to simulate the system evolution along a different trajectory [25]. This, in turn, yields system trajectories and final system states that are logically stored based on the branching strategy and results in a higher ease of subsequent system analysis. The second notable feature of RAVEN is its highly developed ROM options. These ROMs (also known as surrogate models) facilitate the creation of mathematical models that predict the figures of merit of a system in a more efficient manner than by direct simulation. These surrogate models are created via a "training" process by which the internal parameters of the ROM are generated and refined based on carefully chosen system code simulation scenarios. Some obvious implications of this training process include an increase in model accuracy with an increase in the number of training sets and a reduction in the size of the uncertain input domain [25]. However the use of ROMs may greatly reduce computation time, particularly in the analysis of full-scale NPP accident scenarios, and are subsequently worthy of consideration.

RAVEN has also been developed in direct support of the RISMC Pathway Technical Program Plan, which has the overall goal of developing a risk assessment method that is coupled to calculations that quantify plant safety margins in order to allow decision-makers to develop informed margin management strategies [26]. A potential example within the CANDU framework could be the determination of changes in risk that are associated with plant

ageing (a physical process that directly affects safety margins). After CANDU refurbishment there will be a large margin in many safety parameters and/or their timing window. As aging continues after refurbishment some of these may decrease as a result of physical aging of equipment and hence the changes in risk can help operators to better identify future maintenance or actions. A static PRA approach would not provide information on the evolution of risk with age of the station but DPRA can show how the risk changes with life of the reactors.

In practice, both the load and capacity values used to determine safety margins are uncertain in nature. Therefore RAVEN requires a realistic (best estimate) computer simulation tool which can predict the time evolution of the event, the impacts of system redundancies, late or impaired reaction of systems, and human intervention effectiveness. The physics-based results generated by the systems simulation code will allow for more accurate characterization of the time-dependent load that the system will experience during both normal and abnormal transient scenarios, and allow for the creation of distributions that more accurately capture the nature of the system response. RAVEN generates stochastic inputs based on parental uncertainties for important plant system performance parameters, as well as model uncertainties, and provides the transient outcome from these results. Probabilistic safety margins can then be generated once many potential scenarios are generated, giving the probability that load exceeds capacity. The margins resulting from this process can then be used to quantify the impacts on economics, reliability and safety, thereby allowing decision-makers to address margin evaluation, management and recovery strategies on a holistic level.

RAVEN has been selected as the most appropriate PRA driving software to develop and pursue the use of for the eventual goal of performing DPRA studies within the CANDU reactor context. RAVEN's code architecture is conveniently in

line with the DET analyses that will be required, it has the ability to employ flexible surrogate models, its development has been in line with the RISMC technical pathway since its inception, and the open-source nature of the project will allow for customizable changes to be made to the code if new problems arise.

In addition to this, its modular design allows for coupling with any system code (i.e. it is not a nuclear industry specific framework), and a high degree of flexibility in the probabilistic and parametric analysis methods that may be required for its intended application. This high degree of flexibility makes RAVEN an ideal platform for the research outlined in this thesis.

3.1.2 TRACE

TRACE is a best-estimate reactor system code developed by the USNRC for modeling thermalhydraulic phenomena in reactors. It uses a component-based approach to modeling the physical system of interest, whereby each portion of the system is represented by one of a selection of component types (e.g., a pipe, pump, valve, or heat structure). Each component is further subdivided into volumes called cells (the number of subdivisions is usually chosen to balance simulation accuracy and computational effort). Every cell is assigned information such as flow area, length, hydraulic diameter and change in elevation [27].

These components are given connections that instruct TRACE on either the path through which the working fluid can flow, or additional paths through which heat can be transferred (e.g., from a heat structure to the fluid, or through a pipe wall). Where appropriate, boundary conditions such as flow rates/temperatures, or specifying the heat transfer conditions between a component generating heat

and the working fluid are also specified. In this way, the user has flexibility to model a wide range of thermalhydraulic systems related to reactors.

Partial differential equations (PDEs) describing fluid dynamics and heat transfer behaviour for each cell in each component are solved using finite volume numerical methods over a user-specified time range [27]. In order to achieve this, the time range of interest is divided into smaller increments called time steps. At each time step, the PDEs for the entire system are numerically solved, and advancement to the next time step is only achieved if a user-specified convergence criterion is met [27].

TRACE employs a largely one-dimensional (1D) approach to the realization of physical systems: the majority of the hydraulic components (e.g., pipes, valves, pumps) are approximated as 1D. In this way, computational effort is improved significantly with minimal cost in the way of solution accuracy (if any, depending on the application).

In recent years, TRACE has been on-boarded as part of the Canadian nuclear Industry Standard Toolset (IST) [28]. It is being modified for the purpose of analysis of CANDU systems, and it is for this reason that it has been chosen as an appropriate simulation platform for this research.

3.1.3 RAVEN-TRACE Interface

A significant portion of this work has been the development of a specific code interface that allows RAVEN to be coupled with and drive TRACE simulations. This interface is written in Python, and is designed to facilitate DET simulations in as generic a way as possible in order to be useful for future projects.

It is of note that while several code interfaces are available in the RAVEN framework (e.g., RELAP-7, MAAP5), the design of each interface is almost

entirely unique since it is heavily dependent on the architecture of the code it is driving. The developed RAVEN-TRACE interface outlined below was designed based on a thorough understanding of the mechanics of TRACE simulations, in particular methods by which their input parameters can be perturbed, and simulations can be terminated based on specific criteria and restarted again from the termination point with either perturbed, or unperturbed parameters.

3.1.3.1 General Requirements and Considerations for Coupling Codes with RAVEN

In general, an interface between RAVEN and any external application serves the overarching purpose of allowing the application to be run with perturbed inputs. There are several ways in which this can be achieved in the RAVEN framework, and Figure 12 shows a flow chart that can be used to determine the steps required to couple an application to RAVEN. The path highlighted in orange and red shows the TRACE-specific constraints, and indicates that there are two necessary steps for generating a RAVEN-TRACE interface:

1. Create a new code interface with the primary goal of interpreting and perturbing a TRACE input file; and
2. Create a parser that extracts the desired output variables from a TRACE output file and converts them to a CSV which RAVEN is capable of interpreting.

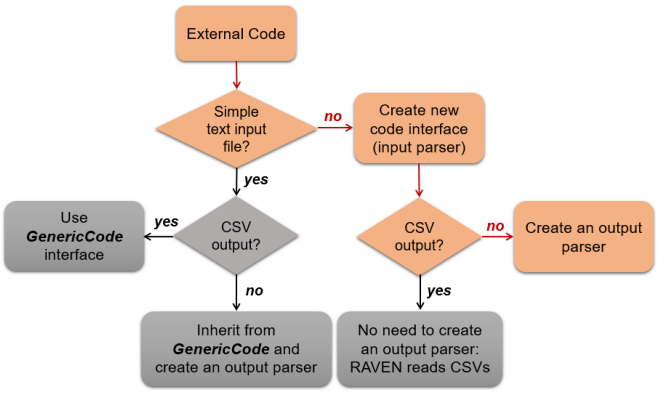

Figure 12: Decision Logic for Coupling a New Application to RAVEN

RAVEN has a built-in capability for perturbing input files of all types through the use of a specific wild-card syntax. The strategy has two requirements for implementation:

1. In the system code input file, the numerical parameters to be perturbed must be replaced with the following syntax: '$RAVEN-*variableName*$'; and
2. In the RAVEN input file, the variable names defined in the <*Samplers*> block must identically match the '*variableName*'s in the system code input file.

Additionally, sampling of parameters in a system code simulation requires a code interface class that:

- Generates a new input file for each simulation, replacing the wild-cards in the original input file with sampled numerical values;
- Executes the system code using each unique input file; and
- Converts the relevant outputs of each simulation to a CSV file so that the data can be interpreted by RAVEN.

3.1.3.2 TRACE-Specific Considerations for Development of DET Interface

In addition to the general code interface requirements mentioned above, several specific methods needed to be written and implemented to enable DET capabilities. The code coupling and sampling strategy is highly dependent on the architecture and requirements of the external code that is being driven. The following is a list of the TRACE-specific considerations that drove the design of the RAVEN-TRACE code interface. It is of note that some of the considerations listed below are generally applicable to all simulations using TRACE, and that others are specific bugs that were discovered during the development of the interface. Future users should be cautious that the list of bugs may not be exhaustive, and that some additional functionality may need to be added to the interface to address newly discovered issues.

General

- TRACE input files do not employ a numerical labeling scheme for the data cards in the input file. Instead, data cards must be placed in a mandatory order in the input file, with required cards being triggered by flags in the preceding cards. Specific labeling of data cards may be done by the user in the form of a comment, but is not mandatory for execution of the code. This is of particular importance to the design of a method by which RAVEN can make changes to the input file that dictate the branches in a DET. Specific text markers have been defined for each type of change that will be made to the input file, and the user is required to insert these text markers as comments in the appropriate places in the input file. During execution of the methods within the RAVEN-TRACE interface, these text markers are used as flags to tell the interface where to make specific changes.

- TRACE generates a binary graphics file that stores the time-dependent parameters of the simulation. The file is in a special format called XTV (X-TRAC-View) that organizes and compresses the data for compatibility with visualisation tools such as AptPlot [27]. RAVEN requires that the data of interest generated during the simulation of a branch be in the format of a CSV file, so a method for converting the binary data to a directly readable format is required.

Trips

- In TRACE, trips provide a powerful framework for initiating changes in conditions or hardware states when specific criteria have been met. In this way, they can be used to drive the transient scenarios that are of interest in DPRA. The entire basis of the RAVEN-TRACE interface for DET simulations is built around the use of trips.
- When a trip criterion has been met, the trip can be specified to dump the time-dependent data up to the point where the trip has been set to a file, and to either 1) continue the calculations with the new, trip-induced conditions, or 2) terminate the calculation. This distinction is made through the sign of the user-defined trip ID number in the input file (a parameter called *IDMP*) [27]. A positive trip ID number allows continuation of the calculation and a negative trip ID number forces the calculation to stop. In this way, trips can be used to drive the logic required to generate the branches of a DET.

Restarting TRACE Simulations

- A calculation can be instructed to begin from a specific point in time through the *DSTEP* parameter in the input file. If *DSTEP* has a value of 0, this indicates an initial calculation (or the first branch of the DET), where

all the input data is defined by the input file. If *DSTEP* has a value less than 0, this indicates that the input data should be retrieved from the last data dump found in the restart file [27]. For this work, these are the only two values for *DSTEP* currently being utilized (0 for the initial branch, and -1 for all other branches), however it is of note that *DSTEP* can also take on an integer value of a specific time step from a restart file from which to begin the simulation.

- A restart calculation requires two input files [27]:
    1. A restart file. In order for the current calculation to retrieve data from a restart file, that file has to be present in the current simulation working directory. In other words, you cannot point to the location of a restart file; it must be copied from its original directory to the current one[3].
    2. A normal, but modified input file. This is a stripped-down version of the original input file, generally containing only information that is new or that has changed from the original.

Other

- TRACE cannot process input parameters that have more than 13 characters. The default in RAVEN is to insert sampled numbers with a high degree of precision, and so these numbers must be truncated to allow TRACE to proceed. The wild cards that are inserted into the TRACE input file can be given format preferences, including a specification of the number of characters to use.

[3] This has been verified by the TRACE development team.

- TRACE cannot process the RAVEN value for infinity (1.8e+308). If this value is inserted into a TRACE input file, it needs to be replaced with a value that TRACE can process (e.g., 1.0e+100).
- The final datapoint in a TRACE calculation is not always found in the XTV file. As such, while the XTV file contains essential information about the overall behavior of the system over the time of interest, information about why the code stopped (i.e., which trip caused the termination) may be missed, and the ASCII output file must be relied on and parsed to retrieve this information.
- In rare cases, TRACE will write duplicate time steps to the XTV file. RAVEN will fail to complete its simulation if this scenario occurs, and a method for searching for and removing duplicate time steps within the interface is required.
- In rare cases, TRACE will insert a numerical value with too many characters in the ASCII output file, resulting in a line of text in the output file that should have spaces between certain words and values, and does not. As mentioned previously, certain parameters need to be retrieved from the ASCII output file. A logical way to do this is to search for a line containing specific words, then split that line into individual elements (based on the 'space' character), and retrieve the element at the appropriate index. If spaces are not present in the expected places, this method fails, and a different approach needs to be taken (i.e., splitting the line based on different criteria).

Several additional implementation lessons that have been learned are summarized in Appendix A.

3.1.3.3 RAVEN-TRACE Interface: High-Level Description

The developed RAVEN-TRACE interface makes use of the following features of TRACE: its ability to initiate transients through the use of trips, and its ability to restart terminated calculations. In this way the branches of a DET can be generated.

Figure 13 illustrates the high-level flow of information as DET simulations are executed. Each time a branch is requested, the sequence shown in Figure 13 is initiated. Each of the methods seen in this figure will be described in detail in subsequent sections. It is of particular importance to note that when the sequence is terminated with a call to *writeXmlForDET*, this instructs RAVEN to save the information at this point to be used for initiating more branches. Careful implementation of the call to this method is central to the success of the DET approach in RAVEN.

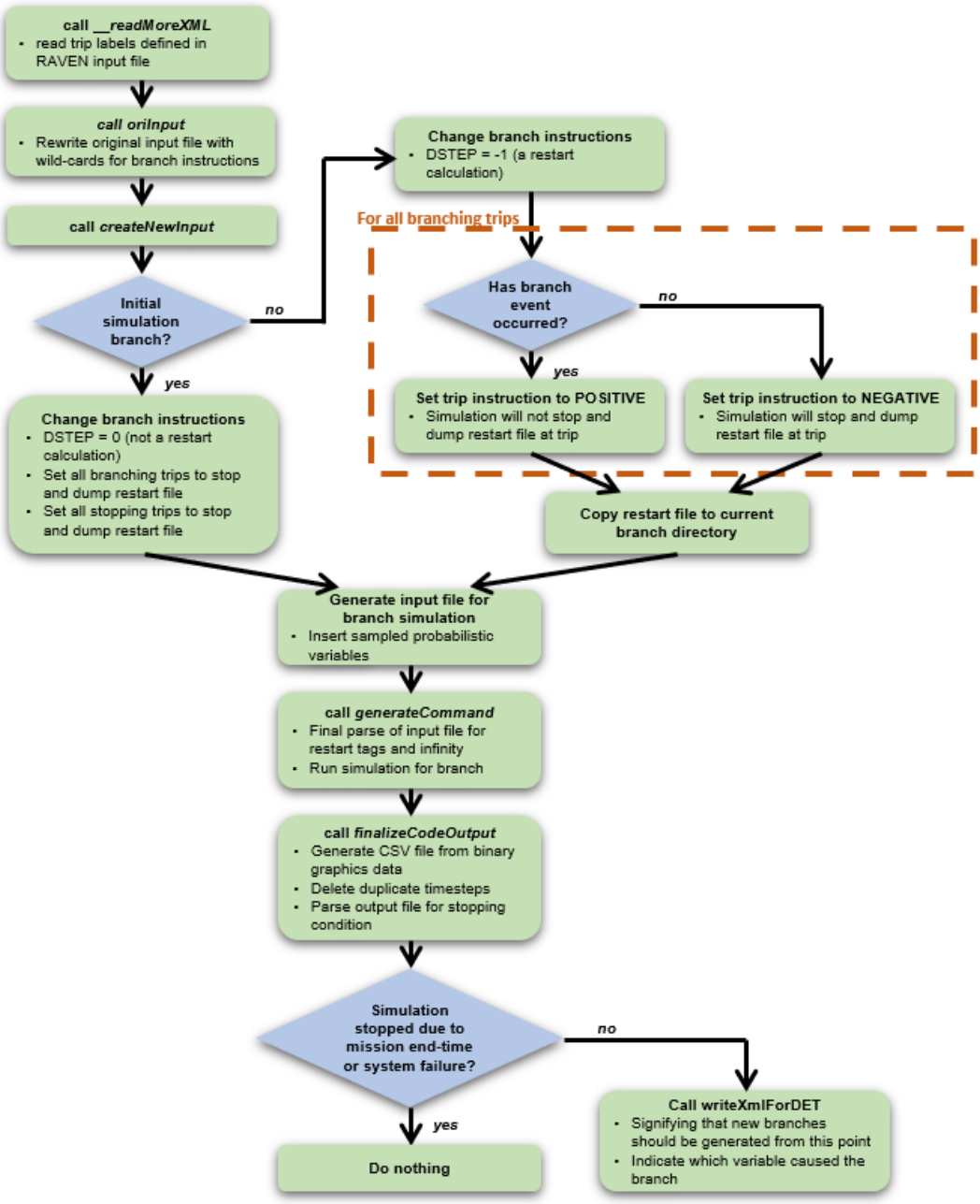

Figure 13: Flow Diagram for RAVEN-TRACE Interface for DET Sampling

3.1.3.4 RAVEN-TRACE Interface Methods

readMoreXML

This method is designed for the purpose of allowing the user to specify custom XML nodes (and associated information or numerical data) that are not part of the default architecture of RAVEN in order for them to be used in the RAVEN-TRACE interface. Since the DET simulations rely almost entirely on the effective use of trips within the TRACE environment, the following two custom XML nodes have been defined, and are required to identify and categorize the trip IDs used in the simulation:

- *branchTripIDs*: a list of the numerical value assigned to each trip in the TRACE input file that is intended to cause the simulation to stop, and for new branches to be spawned from that point.
- *extraSystFailTripIDs*: a list of the numerical value assigned to any additional trips that are intended to cause the simulation to stop due to a failure criterion being met.

Two other custom XML nodes have been defined:

- *branchVarNames*: a list of the names associated with each of the branching variables. The order of this list must match the order of the previously mentioned *branchTripIDs* list. In this way, the interface has a mapping of each trip ID to its associated variable name (e.g., trip ID 9001 is associated with the variable named 'pumpFailTime', and trip ID 9002 is associated with the variable name 'operatorActionTime').
- *traceVars*: a list of the alphanumeric codes associated with all TRACE output variable that the user wants to track (e.g., the status of trip ID 9001 has a code of 'tp9001', and the liquid temperature in cell 1 of pipe ID 10 has a code of 'tln-10A01').

oriInput

The high-level goal of this method is to search the original external code input file for the placement of the variables that dictate specific instructions required for driving the generation of branches and replace them with additional wild cards. When this method is called, RAVEN has already looped through the original input file once and has stored the names of the variables to be sampled from probability distributions. As such, these new wild cards have to be added after the RAVEN simulation has started (i.e., they cannot be added manually to the input file by way of a text editor).

In order to achieve this goal, specific text markers have been defined and placed as comments in the TRACE input file. The method searches the input file for these markers and replaces them with the appropriate wild card format.

The following text markers are currently utilized:

- *%%%*: This marker is placed in the same line as the DSTEP variable (there is only one). The interface then looks for this marker, and replaces the value for DSTEP with $RAVEN-restartCond$. In this way, the variable called 'restartCond' can be changed to either -1 or 0, depending on whether the branch is a restarted calculation or the initial branch (respectively).
- *$$$*: This marker is placed in the same line as the IDMP variable. Lines in the TRACE input file that contain the variable IDMP indicated that a particular trip ID is classified as a 'dump trip'. The interface looks for this text marker and replaces the value for IDMP with $RAVEN-tripIDx$ (where x is replaced with the appropriate trip ID). In this way, the variable called 'tripIDx' can be changed to either a negative or positive value, indicated that the code should either terminate or not (respectively) when the trip condition is met.

createNewInput

This method searches the text in the input file for the current branch for RAVEN wild cards and replaces the wild cards with the corresponding values from the sampling strategy defined in the RAVEN input file. In the case of a DET simulation, this means that it replaces the sampled branching variables (e.g., pumpFailTime and operatorActionTime) with the numerical values appropriate to the current branch.

Several additional parameters (the restart condition and the trip instructions) need to be altered in order to drive TRACE to correctly model the behavior on a branch. These parameters are not sampled from probability distributions, nor can they be given a constant value and as such, it is not possible to change them from the RAVEN input file. Instructions must be written for this purpose in the code interface. Inside the *createNewInput* method, several conditional statements are implemented to this effect.

For the restart condition, the conditional statement is simple:

- If the current branch is the initial branch, set the condition to 0, indicating to start the new calculation from time = 0 s.
- Otherwise, set the condition to -1, indicating to start the new calculation from the last data dump in the restart file.

In order to properly set the trip instruction, a keyworded argument (Kwarg) native to RAVEN called 'happenedEventVarHistory' is used. This Kwarg stores a list of the 'events' that have already occurred on the current branch, and can therefore be used to determine which trips need to be set to terminate the code. If an event has not yet occurred, the trip ID for that event should be set to negative in order to be sure that when the event does occur, the code will stop

and new branches will be generated. If an event has already occurred, the trip ID for that event should be set to positive.

generateCommand

Once the parameters have been properly inserted into the input file, there is an additional step that needs to occur prior to execution of TRACE: values of infinity need to be replaced with a value low enough for TRACE to handle, and the lines of the file that are not needed for the restarted calculation need to be deleted. The generateCommand method first loops through the file looking for 'e+308' and replaces it with '1.0e+100'. It then loops through the file looking for the following two text markers: *&&&start* and *&&&end*. These two markers are placed strategically in the input file as indicators of lines within the file that can be deleted when the branch is a restarted calculation. As mentioned previously, only new or changed information should be present in the input file for a restarted calculation, and these markers allow the user to control where information can be deleted to this effect.

After this is complete, the input file is ready for execution. The restart file from the parent branch is copied into the current folder, and the TRACE syntax for execution of the code is called upon.

finalizeCodeOutput

As mentioned previously, RAVEN requires output data to be in the form of a CSV file in order to be useful. The first purpose of this method is to convert the data of interest from the TRACE output XTV binary file to readable form, and output it to CSV.

After this conversion is performed, the interface needs to determine the following:

- What caused the simulation to stop? There are three options: a branch trigger criterion was met, a system failure criterion was met, or the simulation reached its mission end time.
- If the simulation stopped because a branching criterion was met, what was the variable that triggered this response?

To determine the answers to these questions, the ASCII output file is searched for TRACE-specific groupings of words: 'end of problem' indicates that the mission end time was reached and 'is reset to' indicates that a trip threshold was crossed. If the simulation stopped due to mission end time or system failure, no further action is taken. If the simulation stopped due to a branch criterion, a specific XML file is generated in the current directory. This file is written by calling the method *writeXmlForDET*, and its presence in a directory automatically tells RAVEN that new branches should be generated based on the end conditions of that branch.

One final step is taken: the majority of the files that are automatically output from a TRACE simulation are deleted. All TRACE files (including the XTV and ASCII output) except the input file and restart dump file are deleted. This can save significant space, especially when simulating scenarios with very large numbers of branches.

writeXmlForDET

A call to this method stores the stopping conditions for a branch, only if new branches should be spawned from its end-point. RAVEN has been developed such that the existence of the resulting XML file in a directory is the way in which the code knows to return to this point and generate more simulations. The XML file stores the following stopping conditions: end time, variable that triggered the event, and the current branch probability.

3.1.3.5 Test Case

A simple TRACE model was built for the purpose of developing and testing the interface described in previously. This model is illustrated in Figure 14, and consists of a single pipe component with a constant pressure condition at its outlet, and three individual inlet paths, each with a constant mass flow rate initially, and the capability to change these flow rates when trips are initiated.

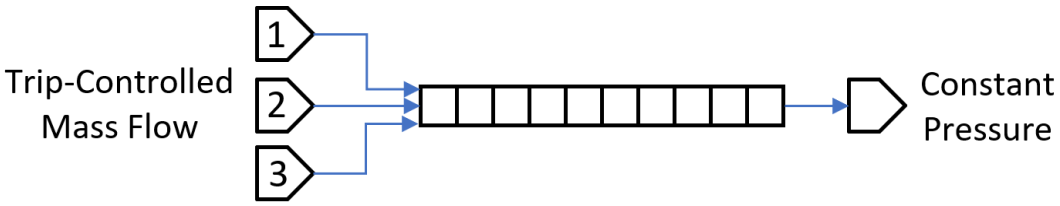

Figure 14: Simple TRACE Model for Developing and Testing RAVEN-TRACE Interface

Each of the three mass flow conditions are described by:

1. Constant rate of 1.5 kg/s before trip. After trip, rate is ramped up to 3.5 kg/s over 5 s.
2. Constant rate of 1.5 kg/s before trip. After trip, rate is ramped down to 0.5 kg/s over 5 s.
3. Constant rate of 0 kg/s before trip. After trip, rate is ramped up to 0.55 kg/s over 5 s.

The failure criteria for the system is a flow rate that exceeds 5.5 kg/s.

Figure 15 shows the results of one example of a DET generated using this model and the developed RAVEN-TRACE interface. At 10 s, the first flow condition trip is initiated, representing the first branching condition. The second and third flow

condition trips are initiated at 30 s and 60 s, respectively, representing additional branching conditions. The probability distribution for each branching variable is uniform over the range of 0 to 100 s. The conditional probabilities for each final branch of the system are calculated in Figure 15. Note that the first conditional probability shown in the figure corresponds to a sequence in which the system failed before the mission end time.

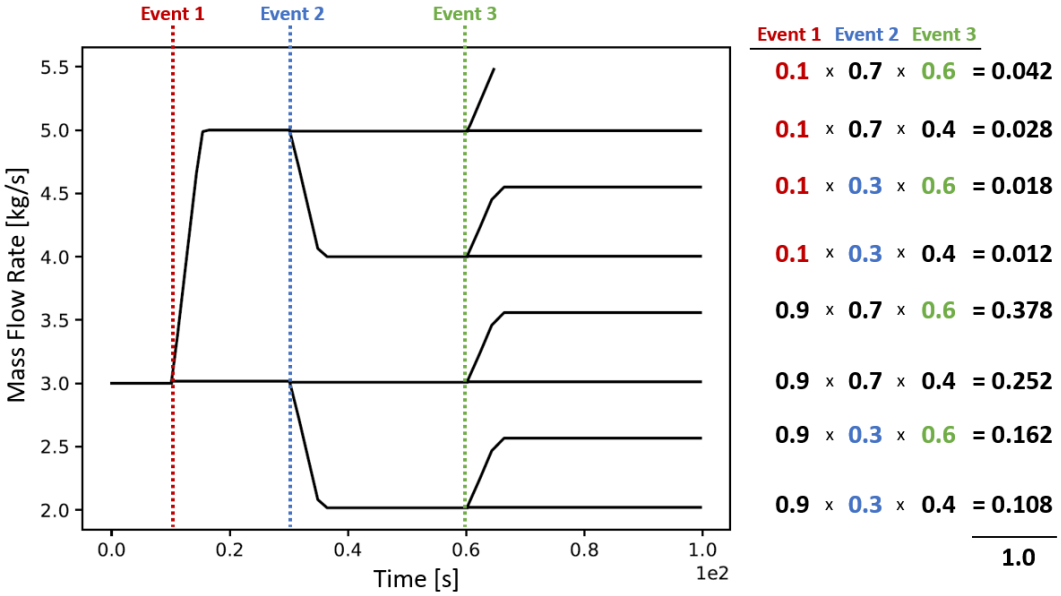

Figure 15: Example of a Simple DET Generated Using the RAVEN-TRACE Interface

Various combinations of branching events were studied in a similar way, demonstrating that the RAVEN-TRACE interface perturbs TRACE files in a predictable manner, resulting in expected branch conditional probabilities.

3.2 Models & Methodology

3.2.1 The Depleting Tank Scenario

The depleting tank scenario is a problem used in literature to investigate control system failures and analysis methods for determining system reliability [29]. The traditional system consists of a tank containing water, where the level is controlled via three active components: two inlet pumps and an outlet valve. Each component has associated failure and repair rates, and the system reliability can be investigated over given mission times. Karanki *et al.* modified this problem for investigation by way of the DET methodology by removing the automated control system aspects, including instead an operator action where the time of action is determined by a defined probability distribution (described in more detail below) [9]. The failure probability of this modified system has an analytical solution, making it an appropriate first step to model using RAVEN in order to assess the capability of RAVEN to generate predictable results.

3.2.1.1 Summary of Karanki et al. [9]

The system consists of a cylindrical tank with a valve located at the bottom that opens spuriously, permitting the water to drain from it. An operator can take action to close the valve and stop the depletion of water; however, the valve must function properly on demand in order for this to occur. The problem considers the following uncertainties, each of which are described by specific probability distributions (summarized in Table 3):

- Aleatory uncertainties (initial water level, time of operator action, and valve response – success or failure – on demand); and
- Epistemic uncertainties (valve discharge coefficient, error factor defining the operator action time PDF, probability of valve failure).

The purpose of the analysis in [9] is twofold: to determine the probability that the water level reaches a critical low level (the system failure criteria) given the uncertainties listed above using the DET approach, and to assess the tradeoff between the accuracy of the calculated failure probability and the required computational resources (which is dependent on the employed discretization scheme, discussed below). The high-level strategy for the simulations is illustrated in Figure 16 and described below:

- Epistemic variables are sampled based on the given probability distributions using a Monte Carlo technique in an outer loop of the simulation.
- For each iteration of the outer loop, the sampled epistemic variables are held constant, and the system response is explored using a DET approach.
- Within each DET, continuous aleatory variables are discretized across the variable range of their given probability distributions using a grid sampling approach.
- The system failure probability of each DET is calculated, resulting in a distribution of the failure probability over the full range of epistemic uncertainties.

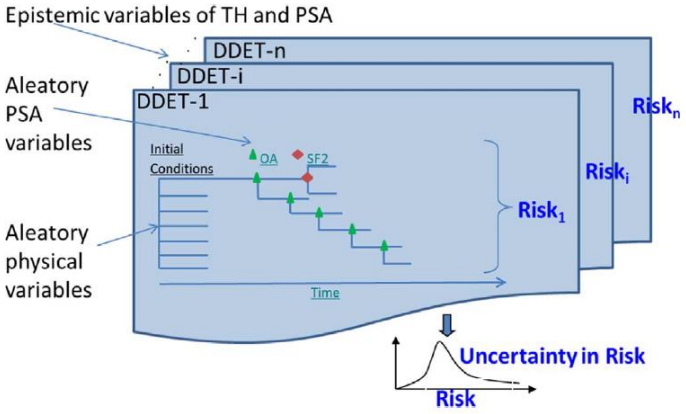

Figure 16: Schematic of DET Methodology for Treating Epistemic and Aleatory Uncertainties [9]

Table 3 summarizes the input data reported in [9], including fixed parameters describing the physical system, parameters that are varied in the reliability assessment, and the probability distribution for each varied parameter.

Table 3: Input Parameters used in Karanki *et al.* [9]

Parameter	Fixed/Varied	Value or Distribution
Radius of Tank, R^2	Fixed	1.0 m
Radius of valve, r^2	Fixed	0.025 m
Critical Level, H_c	Fixed	2.0 m
Initial Level, H_i	Varied (Aleatory)	Normal (μ = 10.0 m, σ = 0.3 m)
Valve Discharge Coeff, C	Varied (Epistemic)	Uniform (min = 0.72, max = 0.98)
Error Factor for Operator Action, EF	Varied (Epistemic)	Uniform (min = 1.8, max = 2.2)
Operator Action Time, t_{OA}	Varied (Aleatory)	Lognormal (mean = 394 s, sigma determined from sampled EF)

Parameter	Fixed/Varied	Value or Distribution
Probability of Valve Failure on Demand, PF_{valve}	Varied (Epistemic)	Lognormal (mean = 2e-4, error factor = 5)

The initial water level in the tank and the operator response time are continuous aleatory variables, and therefore must be discretized in order to generate branches in a DET simulation. The initial tank level is discretized linearly over the entire distribution[4]. Branches representing the response of the valve on demand are binary (i.e., the valve either functions, or fails).

Several discretization schemes are assessed for generating branches for the operator action timing. The range of possible values is described by a lognormal distribution, and this distribution is discretized into either 4, 5, 7, 10 or 20 grid samples (each of which includes a branch that considers the system response if the operator does not act). Table 4 summarizes three of the five schemes, and Figure 17 shows the specific operator action times using the 7-branch scheme (with the error factor of the lognormal distribution held at its mean value of 2.0) [9].

[4] The number of discrete initial water levels and their specific values are not reported.

Table 4: Description of the 4, 5, and 7 Branch Schemes for Depleting Tank Study

4-Branch		5-Branch		7-Branch	
CDF_{OA} Value	**Branch Probability**	**CDF_{OA} Value**	**Branch Probability**	**CDF_{OA} Value**	**Branch Probability**
0.05	5%	0.05	5%	0.05	5%
0.5	45%	0.5	45%	0.5	45%
0.95	50%	0.95	45%	0.9	40%
No Action	5%	0.999	4.9e-2%	0.99	9e-2%
		No Action	1e-3%	0.999	9e-3%
				0.9999	9e-4%
				No Action	1e-4%

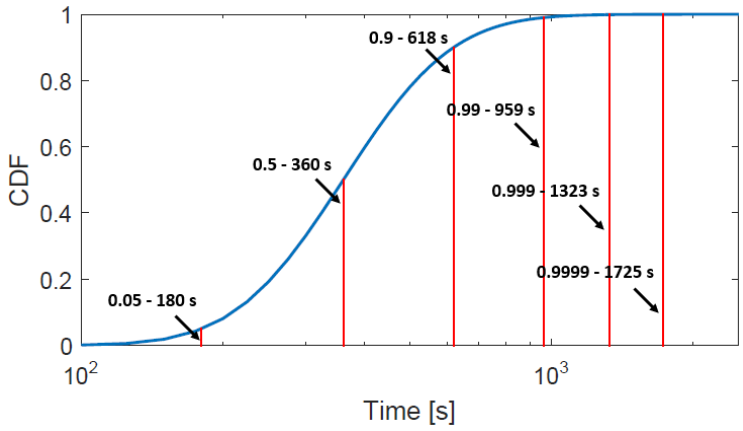

Figure 17: 7-Branch Discretization Scheme for Operator Action Timing for Depleting Tank Study

Two other discretization schemes – a 10 and 20-branch scheme – are assessed in [9]; however, the information present on these schemes is less detailed in the paper. It is known that system failure probability is sensitive to late operator action times, in particular in the region between 0.9 and 1.0 on the operator action CDF [9]. The 10 and 20-branch schemes mimic the 7-branch scheme, and divide this region more finely; however the specific CDF values are not reported.

The time required for the water level in the tank to reach the critical level is based on Bernoulli's equation and is given by:

$$TW = \frac{R^2}{Cr^2}\sqrt{\frac{2}{g}}\left(\sqrt{H_i} - \sqrt{H_c}\right) \quad (6)$$

The system failure probability is a function of the valve failure and human error probabilities (i.e., the probability that the operator action time exceeds the window of time available to take action before the system fails). The probability of operator response times is defined in this problem by a lognormal distribution and so its PDF is known. The PDF of the time window must be derived from the given PDF of the initial water level through a transformation of random variables. The human error probability can then be calculated as the probability of the time window being less than the response time. Figure 18 shows the PDFs for both the time window and the operator action timing, with all uncertain epistemic parameters held at their average values. Figure 19 shows the same distributions with the epistemic parameters held at the minimum, average, and maximum values, highlighting that there would be resultant variation in the human error probability, and consequently the system failure probability.

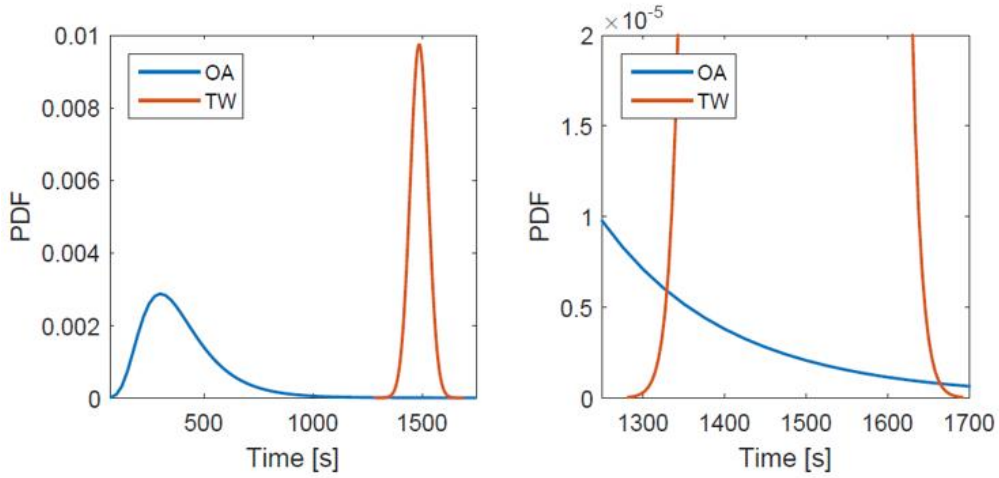

Figure 18: PDF for the Operator Response Time and Time Window

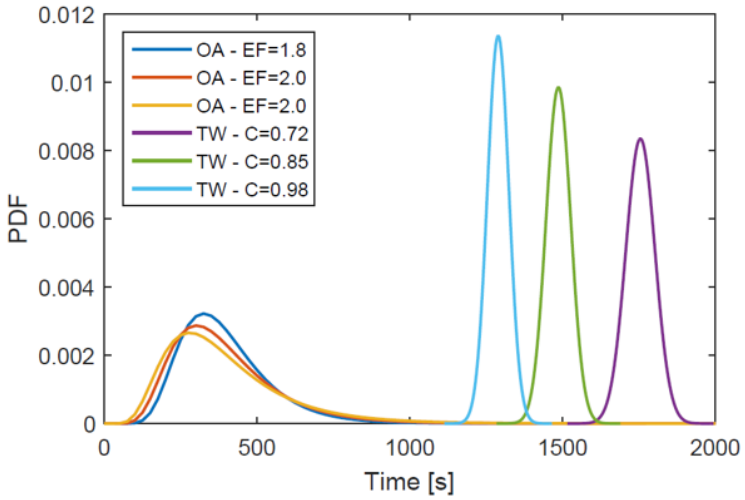

Figure 19: Epistemic Variation in PDFs

With all epistemic parameters held at their mean values, the analytical solution for the system failure probability reported in Karanki *et al.* is 5.98e-4 [9]. Table 5 summarizes the system failure probabilities reported in Karanki *et al.* for each of the operator action discretization schemes, and the ratio of the failure probability for that scheme to the analytical result. It is of note that three

different approaches were assessed for the 5-branch scheme, whereby the first three operator action time branches are associated with CDF values of 0.05, 0.5 and 0.95, and the fourth branch is associated with a CDF value of either 0.99, 0.999 or 0.9999 (recall that the fifth and final branch corresponds to the scenario where the operator does not act). This illustrates the sensitivity of the system failure probability to the discretization of the tail of the operator action CDF.

It can be seen that a larger number of discretization levels results in better accuracy; however it is also noted that satisfactory results may be achieved with a modest level of discretization (e.g., with either 7 or 10 branches).

Table 5: System Failure Probability without Epistemic Uncertainty

# OA Branches	Failure Probability	Failure Probability Overestimation
4	5.02e-2	83.9
5 – 0.99	1.02e-2	17
5 – 0.999	1.19e-3	1.98
5 – 0.9999	5.02e-2	83.9
7	1.19e-3	1.98
10	7.83e-4	1.31
20	6.43e-4	1.07

Figure 20 shows the digitized final results presented in Karanki et al. [9] for each of the operator action discretization schemes, illustrating that the resulting CDF approaches the analytical solution as the number of discretization levels is increased.

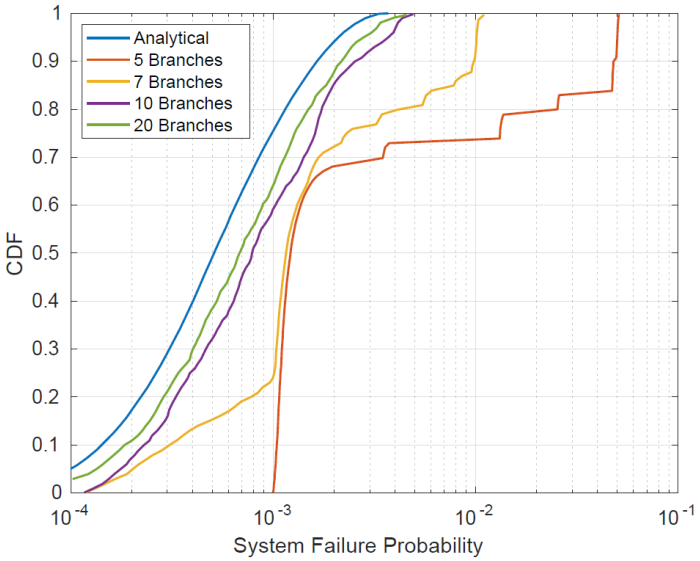

Figure 20: Digitized DET Results Reported in Karanki *et al.* [9] for Different Operator Action Discretization Schemes

3.2.1.2 RAVEN Methodology

A fundamental requirement for DET simulations is that the system code being driven by the PRA framework needs to be capable of stopping when a specific criterion is met, and restarting again from the same place, with altered inputs that explore different potential paths (or branches) for the system response. To this end, a simple python system code was developed to simulate the time-dependent change in water level in the depleting tank scenario. This code was then driven by RAVEN using the DET methodology. The python code was designed to be analogous to TRACE (i.e., it is a stand-alone executable code that takes in an input file, and writes the time-dependent simulated data to an output file).

Figure 21 illustrates the potential DET branches for the depleting tank system, and the python code either allows the tank to deplete, or holds the water level

constant, depending on the parameters defined in the input file. The required inputs include the following:

- Grid sampled value of the initial water level in the tank at the start of the scenario;
- Initial time and initial water level for the branch;
- Epistemically sampled values of the discharge coefficient and valve failure probability; and
- Grid sampled value of the operator action time.

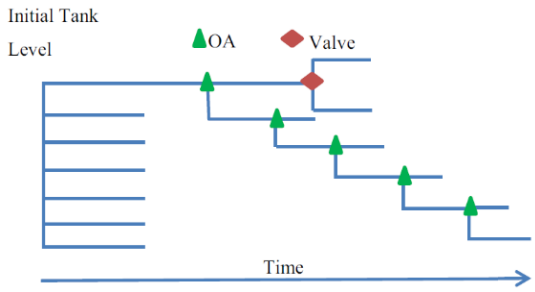

Figure 21: Illustration of DET Branches for the Depleting Tank Scenario [9]

RAVEN drives a single depleting tank DET simulation (without consideration for epistemic uncertainties – this will be described later) in the following way:

- There are three aleatory variables which are sampled in the DET (refer to Figure 21): initial water level, operator action time, and valve response.
- Within RAVEN, these three variables are defined using built-in probability distributions, and sampled using a grid sampling technique whereby thresholds on the corresponding probability distributions are specified. These thresholds represent the locations on the distribution where the code should stop and generate new branches.

- An initial input file for the python code is written with a specifically formatted wild card for each of the sampled variables mentioned above. In this way, RAVEN can replace these wild cards with the appropriate sampled value depending on the branch of the DET.
- RAVEN generates the initial branch of the DET, and the system code begins simulating the initial depletion of the tank. When the time in the system code reaches the first threshold on the operator action time distribution, the following occurs:
    - The simulation is terminated and the temporal evolution of the water level up to that point is written to a CSV file;
    - A flag is passed to RAVEN to indicate that new branches need to be generated; and
    - Two new branches are generated: one where the operator acted at the first threshold, and another where the operator did not act, and the threshold is advanced to the next grid value. Note that on a branch where the operator does act, there are two possible sub-branches: the valve responds successfully, or the valve fails.
- This strategy is iterated for all thresholds defined for each variable, promoting the simulation of all possible system pathways.
- In addition to terminating the branch simulation due to meeting a new branching criterion, the system code is also instructed to terminate if the water level reaches the critical level (i.e., system failure), or if the simulation time reaches a user defined mission end time (i.e., system success).

There are several potential options for introducing epistemic uncertainties into DET simulations within RAVEN. The simplest of these options is to utilize the

'Hybrid DET' functionality, whereby additional variables can be sampled via MC technique within the same RAVEN input file generating the DET. This option is suitable for scenarios where the epistemic variable is passed to the system code for direct use in the prediction of the system response. In the case of the depleting tank scenario, this strategy is used for sampling the valve discharge coefficient.

The two other parameters with epistemic uncertainty in the depleting tank scenario are parameters which define a probability distribution: the error factor defining the lognormal distribution for the operator action time, and the value of the valve failure probability. In order to simulate these uncertainties, an approach called 'RAVEN-running-RAVEN' is used. Two RAVEN input files are required: an inner one which runs the python code in a DET framework (and samples the discharge coefficient using the Hybrid DET method described above), and an outer one which samples the error factor and valve failure probability and passes these values to the section of the inner file which defines the probability distributions used in the inner DET simulation.

3.2.2 Spent Fuel Pool

3.2.2.1 Summary of Chraibi *et al.* [10]

A reliability study performed by Chraibi *et al.* [10] on a simplified SFP scenario was selected as an appropriate case against which to benchmark DET results using the RAVEN-TRACE interface described in Section 3.1.3. The high-level method and results reported in this paper were previously outlined in Section 2.3 in order to highlight the meaningful findings that it provides in the context of advancing DPRA methodologies. Further details from this study are provided

below, as these details have driven specific modeling choices for the TRACE nodalization of the SFP, and the branching strategy employed using RAVEN.

The physical system includes the SFP and three cooling trains which are activated depending on the failure status of the components of the system. These trains have been simplified and shown in Figure 22. The scenario under consideration in the study requires only a single train to be operational for the pool to be sufficiently cooled.

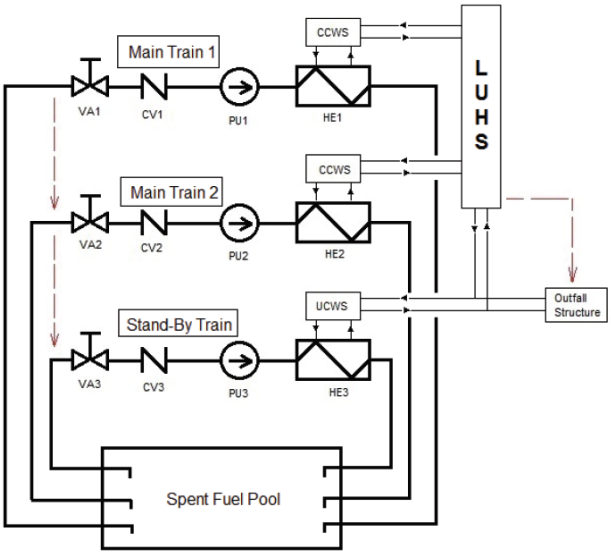

Figure 22: Simplified Schematic of the Spent Fuel Pool [10]

The overall system is comprised of a large number of components and safety systems that are subject to failure (valve, pumps, heat exchangers, components of the overall cooling water system etc.). Each of these components or safety systems are given a specified failure rate, and in some cases, a specified repair rate.

The following scenarios are considered:

- C1: Non-repairable Loss of Ultimate Heat Sink (LUHS) over a mission time of 100 hours
- C2: Non-repairable Loss of Offsite Power (LOOP) over a mission time of 192 hours
- C3: Extended simulations subject only to potential and repairable component system failures (including LUHS and LOOP) over a mission time of 7846 hours

The transient is monitored for the following unwanted events (UEs):

- UE0: Simultaneous loss of all three trains (classical approach)
- UE1: Water temperature rising above 80°C (safety margin for boiling)
- UE2: Water temperature rising above 100°C
- UE3: Water level dropping below 16 m (fuel uncovery)

Explicit formulas are used for the dynamics of the heat exchangers and ODEs are used to describe both the water level and temperature within the SFP [10].

The following is a list of useful information about the system found in [10]:

- The surface area of the pool is 77 m^2.
- The initial water level in the pool is 19 m.
- The residual power from the spent fuel present in the pool is 5.85 MW and it is stated explicitly that only one coolant train is required to sufficiently cool the fuel.
- The initial temperature of the pool water is 50℃, or 323 K (this is assumed to be the average steady state temperature within the pool).

- The PyCATSHOO model assumes that there is no pressure loss in the coolant trains, and that the properties of the coolant remain constant (e.g., the coolant density is held constant at 990 kg/m^3).
- There are two identical 'main trains', and a third, fully independent 'standby train' which is activated as a backup after certain failures have occurred.
- Each train rejects heat from the pool to the river in two stages through countercurrent flow heat exchangers (HEs): first, from a primary coolant loop to a secondary loop, then from the secondary loop to a tertiary system connected to the river.
- Each train contains valves (VA), check valves (CV), and pumps (PU)
- The coolant flow rate through each of the pumps in a single train is:
    - Primary: 550 m^3/h (153 kg/s)
    - Secondary: 520 m^3/h (144 kg/s)
    - Tertiary: 650 m^3/h (181 kg/s)
- The main trains are cooled by the Component Cooling Water System (CCWS) and the standby train is cooled by the Ultimate Cooling Water System (UCWS).
- In the case of a LUHS, the main trains are lost, and the UCWS (which cools the standby train) can be supplied with water from the outfall structure.
- The electrical supply for each train is provided by a train-specific Switch Board Division (SBD).
- In the case of a LOOP, each SBD can be supplied by a train-specific Emergency Diesel Generator (EDG).

- Each component can be in only one of two states: operational or failed.
- The failure and repair rates for each component or system follow exponential probability laws described by the transition rates shown in Table 6.
- Transitions between states occur immediately, with no time delay.
- There is no loss of heat to the environment.
- Water makeup strategies are not considered.

Table 6: Components and Systems Subject to Failure in the SPF Scenario

Component or System	Failure Rate (h^{-1})	Repair Rate (h^{-1})	MTTR
Switchboard Division (SBD)	1.0e-6	-	24
Emergency Diesel Generator (EDG)	2.3e-3	1.8e-3	39
Valve (VA)	8.7e-8	1.1e-4	10
Control Valve (CV)	7.0e-9	5.2e-7	12
Pump (PU)	1.1e-5	2.7e-5	18
Heat Exchanger (HE)	2.9e-6	-	120
Component Cooling Water System (CCWS)	8.4e-6	-	72
Ultimate Cooling Water System (UCWS)	2.6e-5	-	48
Outfall Structure	5.0e-6	2.0e-4	23
Loss of Off-site Power (LOOP)	4.0e-6	-	24
Loss of Ultimate Heat Sink (LUHS)	1.2e-9	-	30

Given the information provided above, it was decided that the LUHS scenario in which the water temperature reaches 80°C was the best candidate to begin examining using RAVEN and TRACE. In the event of a LUHS, both of the main

trains are immediately lost and the standby train is fully relied on for heat removal. Due to this, the TRACE model requires only a single train (rather than all three), simplifying both the thermalhydraulic aspects of the model, and the control system aspects (since implementation of logic to switch between trains is not required). Figure 23 shows the evolution of the failure probability for each of the unwanted events in the LUHS scenario as reported in Chraibi *et al.* [10].

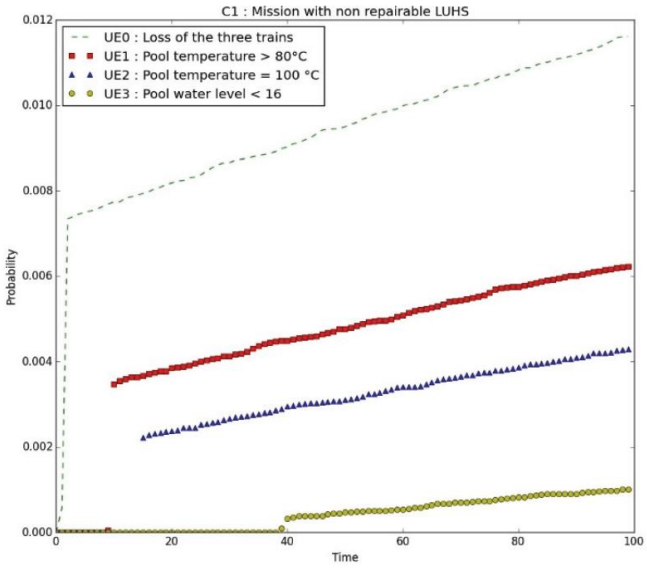

Figure 23: Evolution of Failure Probability in the LUHS Scenario [10]

3.2.2.2 TRACE Model

Figure 24 shows the thermalhydraulic components of the SFP model in TRACE. At a high level, the model consists of:

- A large vertical pipe representing the volume of the pool (connected to an atmospheric pressure boundary condition at the top);
- An active heat structure within this pipe generating the specified 5.85 MW of decay heat from the spent fuel;

- A primary heat transfer loop connected directly to the pool;
- A secondary heat transfer loop connected passively to the primary loop via a heat structure (with no active heat generation); and
- A tertiary heat transfer pathway that represents the connection to the ultimate heat sink (i.e., the river), connected passively to the secondary loop via a heat structure.

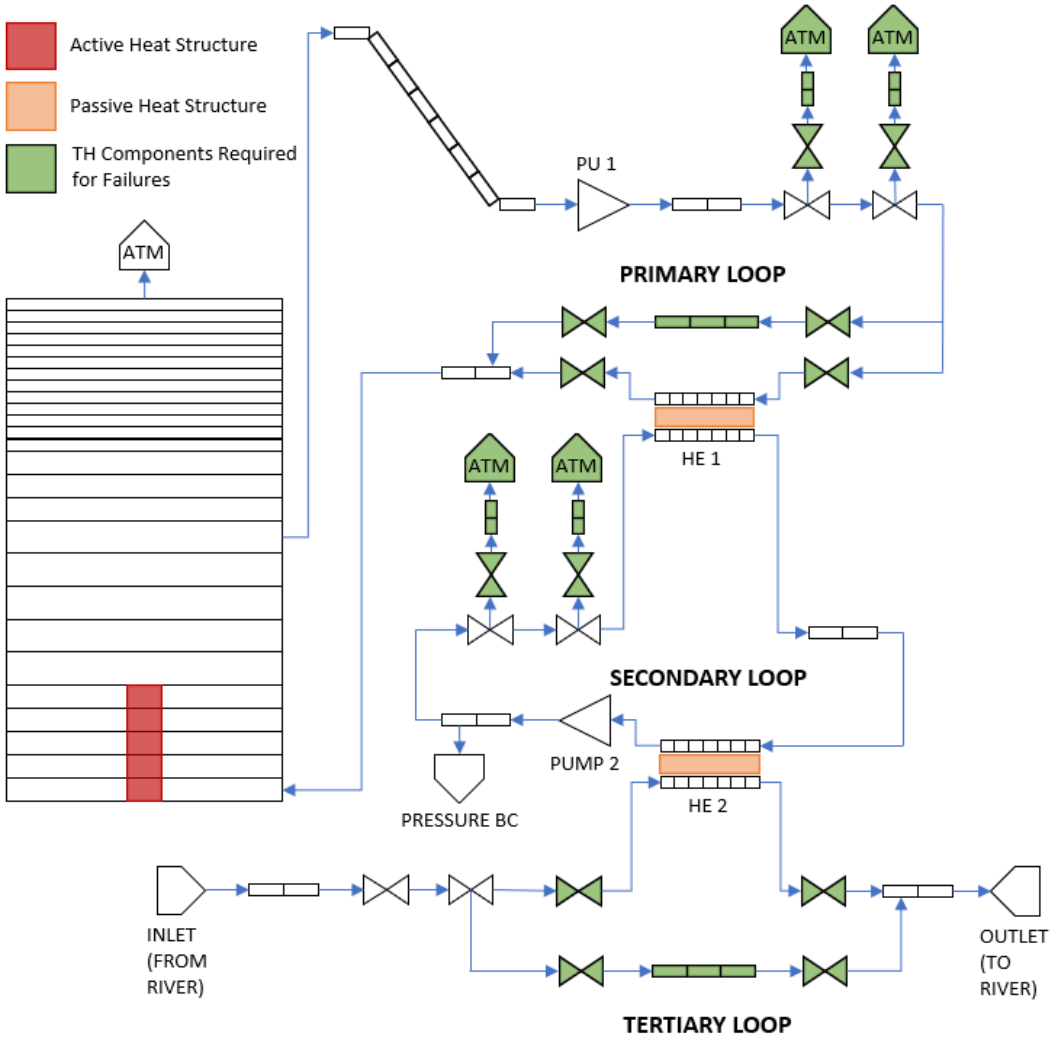

Figure 24: TRACE Model of Spent Fuel Pool

The following modeling choices and strategies have been made in order to model the steady-state and transient behaviour of the SFP:

- As mentioned previously, pressure losses are not modeled in the coolant trains in the PyCATSHOO simulation, leading to the decision to not include any major or minor loss coefficients in the TRACE model (i.e., no wall roughness, and no form loss factors).
- The primary and secondary pumps (labeled PU1 and PU2 in Figure 24) are not modeled as realistic pumps, but rather as something called a 'single junction component', simply representing a flow boundary condition in each loop. The mass flow rate of these components is controlled directly by the control system logic in the TRACE model, and the pumps can be toggled between an operational and failed state, depending on the failures that are active in the current branch of the DET.
- While the primary and secondary loops have closed flow paths, the tertiary loop is realized using an open system, where time-dependent control volumes (called 'fills' in TRACE) represent the boundary conditions (i.e., the inlet and outlet connections to the river). The tertiary pump is not explicitly modeled, because the mass flow rate in the fills can be controlled in the same way as the single junction components representing PU1 and PU2.
- The heat exchangers are modeled in the following way (and illustrated in Figure 25):
    - The tube side consists of 197 parallel pipes;
    - The shell side consists of a single pipe; and
    - A passive heat structure is defined with its left and right boundary conditions tied to the tube and shell components, respectively.

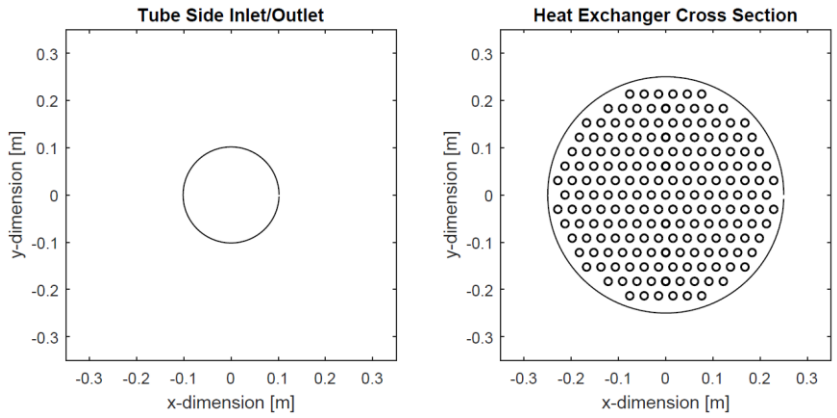

Figure 25: Heat Exchanger Design

- In order to model a failed heat exchanger, the following strategy was implemented (illustrated in Figure 26):
    - A separate flow path was modeled that is not connected to the passive heat structure required for the heat exchanger to reject heat from the shell side to the tube side.
    - Flow through either path is controlled by valves that are either open or closed, depending on the state of the heat exchanger (controlled directly by the control system logic in the model).

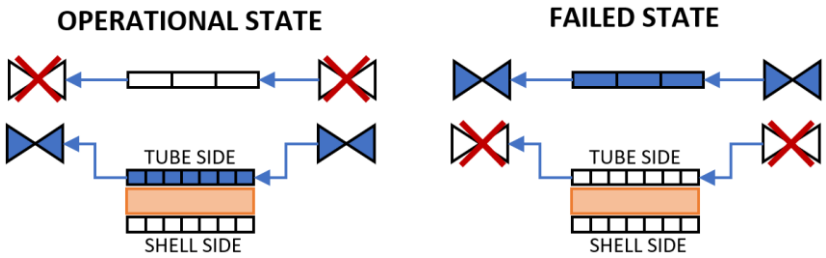

Figure 26: Strategy for Modeling Heat Exchanger Failure in TRACE SFP

- Chraibi *et al.* [10] lacks clarity on the physical representation of failures of valves in the PyCATSHOO model. It was decided that a failure of a valve in this system would most likely manifest as a leak, and thermalhydraulic and control system components were modeled to achieve this. At each valve that may fail, an alternate flow path leading to an atmospheric boundary condition is present. This alternate path is either made available or unavailable through the use of an additional control valve.

Table 7 provides a summary of the way in which each of the component and system failures have been modeled in TRACE. Each of the failures is initiated by way of a trip in the control logic of the TRACE model. When activated, these trips terminate the TRACE calculation and initiate the sequence of logical checks illustrated in Figure 13 through the developed RAVEN-TRACE interface.

Table 7: Summary of Failure Model for Each SFP Component and System

Component or System	Brief Description of Failure Model
PU	Reduction in flow to 1% nominal flow rate *
SBD	Reduction in all pumps to 1% nominal flow rate
EDG	Reduction in all pumps to 1% nominal flow rate
VA	Break/leak in pipe
CV	Break/leak in pipe
HE	Bypass flow path connecting tube and shell side
CCWS	Not modeled **
UCWS	Reduction in flow to 1% nominal flow rate in tertiary loop
Outfall Structure	Reduction in flow to 1% nominal flow rate in tertiary loop
LOOP	Reduction in all pumps to 1% nominal flow rate ***
LUHS	Loss of availability of the two main trains ****

* A full reduction in flow to 0 kg/s manifests scenarios in which TRACE fails to converge during a time step. Appendix C shows an assessment of the impact that several failure condition flow rate values could have on the final solution, and it was determined that using 1% of the nominal rate would have a negligible effect.

** The CCWS cools only the two main trains and is not relied on in a LUHS scenario.

*** This reduction in flow occurs only if the EDG is also in a failed state.

**** LUHS is the initiating event in this scenario.

It is important that the steady state behavior of the system be examined before the model is driven by RAVEN to assess the system outcomes as a result of initiating events and component and system failure sequences. The steady state behaviour of the system over the 100 hour mission time is outlined in Appendix B. The system pressures and temperatures behave as expected, and in line with the information presented in Chraibi *et al.* [10].

3.2.2.3 DET Implementation Using RAVEN

Each of the system and component failures modeled in Chraibi *et al.* [10] are characterized by a defined failure rate (as per Table 6). It is stated explicitly that the transition from operational to failed states for these components follow an exponential probability distribution, and the defined failure rates correspond to the rate parameters for the exponential distribution for each component. It is for this reason that the following branching strategy has been implemented for the SFP scenario:

- Exponential distributions are defined within the RAVEN input file for each of the failure rates provided. Figure 27 shows the distribution for the failure of individual pumps in the system.
- The sampling domain for each of the components or systems is [0 100] hours. In Figure 27, the left hand side shows the exponential distribution over a large range, and the right hand side shows the same distribution over the range of interest in this initiating event scenario. It is significant to note that in this range, the probability distribution is essentially uniform.
- The range of interest is then discretized in order to generate points at which branches are spawned. These points represent the possible times at which each component or system may fail.
- These sampled points are then provided to the TRACE model through the RAVEN-TRACE interface, and the overall system response is determined by exploring all possible combinations of the component and system failures.

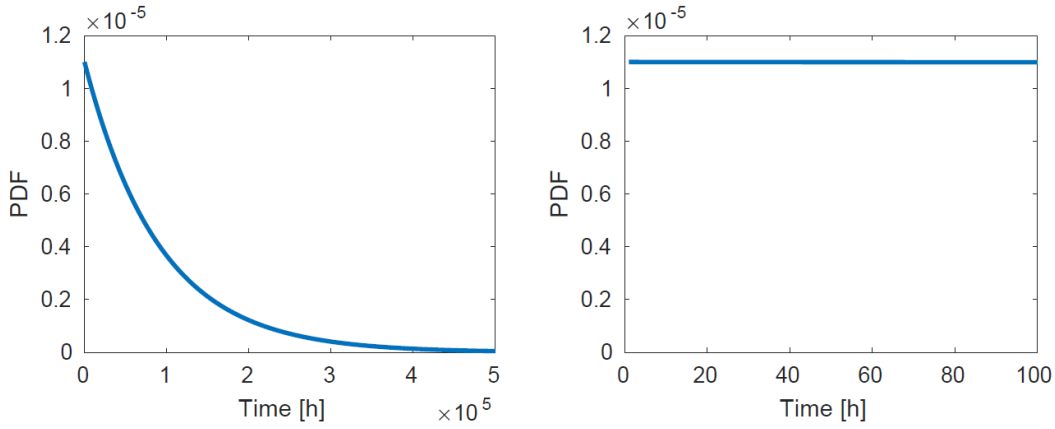

Figure 27: Failure Probability Distribution for the Pumps in the SFP

Figure 28 shows a preliminary test that was performed in order to test the mechanics of the TRACE model, and RAVEN's ability to perturb it and instigate the failures of interest. In this example, only a single component failure is initiated: PU1. The simulation mission time is shortened significantly to 3000 s, and the system failure criteria is set to an extremely low value of 324.6 K. The branching variable dictating the time of failure of PU1 is provided two thresholds (i.e., points on the exponential probability distribution defining the time at which the component fails). When a threshold is reached, the simulation for the current branch is terminated, and new branches are spawned. Each of the branches shown in Figure 28 have an associated label, and are described in Table 8.

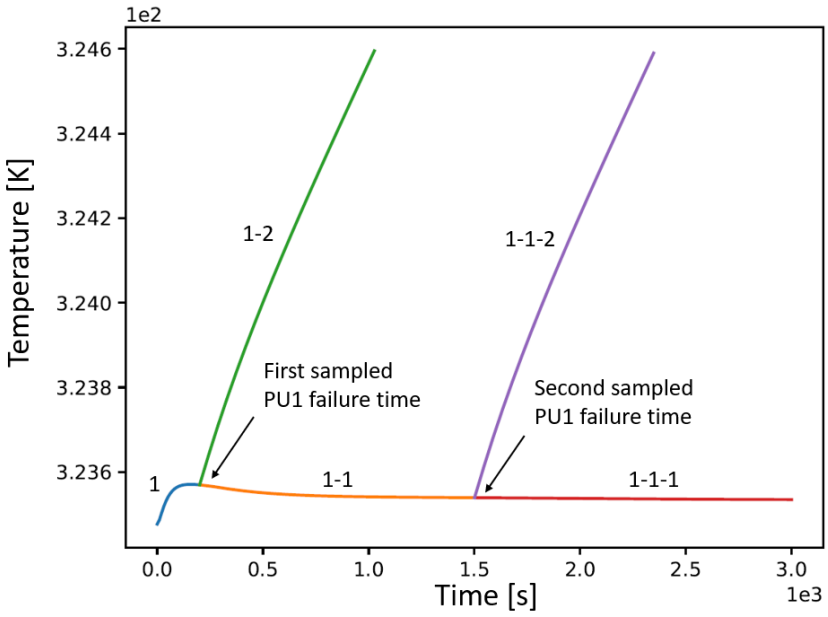

Figure 28: Temporal Evolution of Temperature – Testing a Single Failure

Table 8: Description of DET Branches in Figure 28

Branch	Description	Reason for Termination
1	Initial branch. No failures have occurred	First sampled failure threshold reached
1-2	PU1 failed at the first sampled time	Failure criterion met
1-1	No failures have occurred	Second sampled failure threshold reached
1-1-2	PU1 failed at the second sampled time	Failure criterion met
1-1-1	No failures have occurred	Mission end time reached

Similarly, Figure 29 shows an example where failures of all three of the pumps are initiated. Each of the three failures are provided only a single sampled branching threshold, and the DET explores all possible combinations of these component failures:

- PU1, PU2 or PU3 only
- PU1 and PU2
- PU1 and PU3
- PU2 and PU3
- PU1, PU2 and PU3

A key outcome that can be observed from Figure 29 is that for an early PU1 failure, the dynamics of the system are such that some cases will have a lower temperature than cases with a later failure of the other pumps. This provides a simple illustration of the benefits of a DPRA approach wherein the timing of the events relative to one-another can significantly impact the accident trajectory.

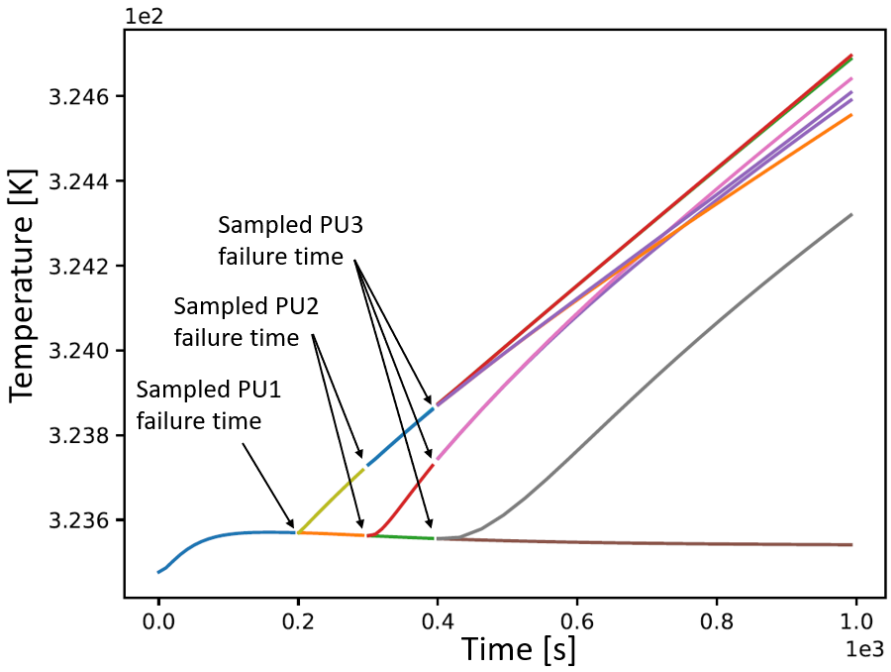

Figure 29: Temporal Evolution of Temperature - Testing Three Failures

A third example is shown in Figure 30 in order to illustrate the DET behaviour if a LOOP were to be initiated, followed by a failure of the EDG. It can be seen that when the LOOP is first triggered, the system temperature does not rise. This is due to the fact that the EDG is not in a failed state, and is supplying power to the pumps. If the EDG then fails while a LOOP is in progress, the pumps lose power and are unable to cool the fuel.

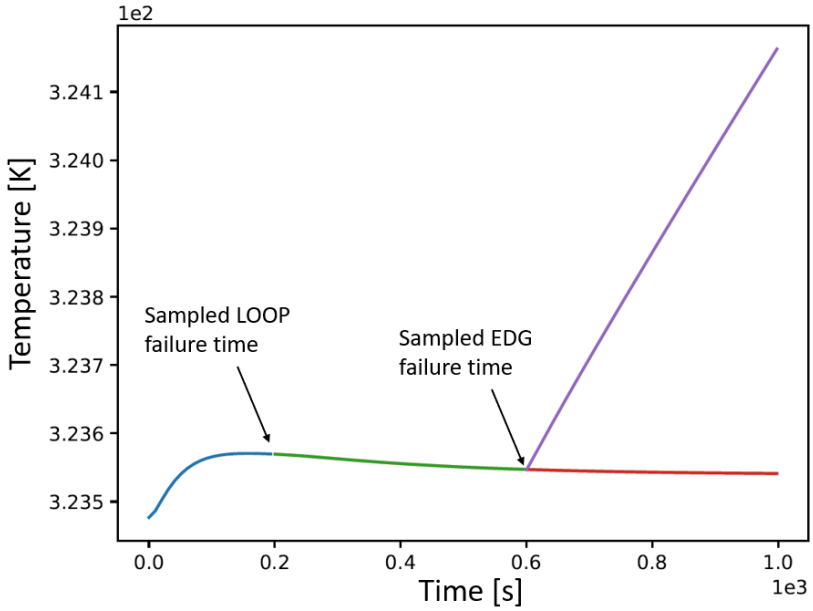

Figure 30: Temporal Evolution of Temperature - Testing a LOOP and EDG Failure

4. RESULTS

4.1 Depleting Tank Scenario

4.1.1 Without Epistemic Uncertainties

Figure 31 shows the temporal evolution of the water level for each branch in the tank RAVEN DET simulations employing the 5-branch and 10-branch schemes for discretizing the operator action time. Each of the horizontal lines represents a branch where the valve responded successfully when the operator took action to close it, and results in system success over the mission time. It is of note that while the right hand plot shows the 10-branch scheme, only eight probability thresholds can be seen. The ninth threshold results in the operator acting too late, regardless of the initial water level in the tank, and is consequently not simulated in this case. When epistemic uncertainties are considered, both the window of time available for the operator to act and the probability distribution that specifies the operator action times are affected, resulting in scenarios where this ninth threshold is significant.

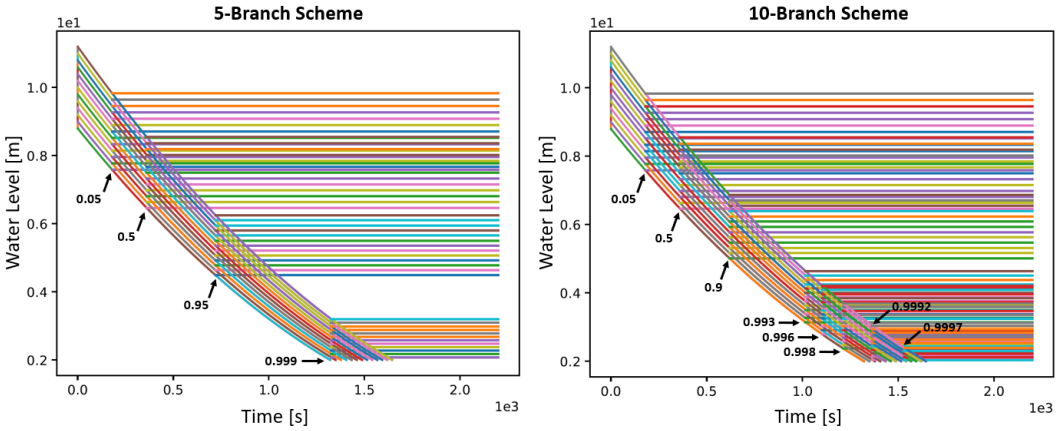

Figure 31: Temporal Evolution of Water Level for All Branches of the DET using Different Operator Action Time Discretization Schemes

After the completion of a DET simulation, the overall system failure probability can be calculated by examining all branches which either reached the mission end time, or resulted in system failure. The summation of the conditional probabilities of these branches is 1.0 (representing all possible final states of the system). The system failure probability is the summation of the conditional probabilities of all final branches which resulted in a water level at or below the critical level for the tank. Table 9 summarizes the system failure probabilities calculated in this way for RAVEN simulations employing each of the operator action discretization schemes (with no epistemic uncertainties considered).

From the trend in the overestimation shown in the table, it can be seen that a low level of discretization in the operator action can produce highly conservative results for the system failure probability, and that a finer discretization scheme can eliminate some of this conservatism. Each of the failure probabilities as calculated using RAVEN show good agreement with the results reported in Karanki *et al.* [9].

Table 9: Comparison of RAVEN and Karanki *et al.* [9] Results for System Failure Probability without Epistemic Uncertainties

	Karanki *et al.* [9]		RAVEN	
# OA Branches	**Failure Probability**	**Failure Probability Overestimation**	**Failure Probability**	**Failure Probability Overestimation**
4	5.02e-2	83.9	5.02e-2	83.9
5 – 0.99	1.02e-2	17	1.02e-2	17
5 – 0.999	1.19e-3	1.98	1.20e-3	2.01
5 – 0.9999	5.02e-2	83.9	5.02e-2	83.9
7	1.19e-3	1.98	1.20e-3	2.01
10	7.83e-4	1.31	9.22e-4	1.54
20	6.43e-4	1.07	6.58e-4	1.10

4.1.2 With Epistemic Uncertainties

Figure 32 shows the CDF for the system failure probability when using the 5-branch scheme for discretizing the operator action timing. The plot shows the digitized results reported in Karanki *et al.* [9] along with three sets of results generated using RAVEN: one in which the initial water level distribution is discretized into 7 levels, and ones in which it is discretized into 13 or 100 levels. As mentioned previously, the specific discretization scheme is not reported for the initial water level, and the results shown in Figure 32 illustrate that the lower and upper bounds and the overall shape of the system failure probability CDF are not sensitive to the discretization scheme chosen. Each scheme results in good agreement with the results reported in Karanki *et al.* [9].

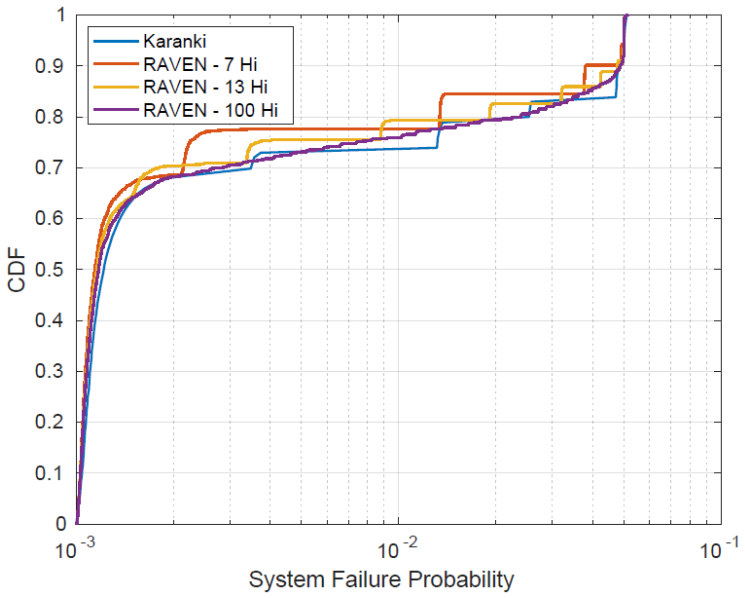

Figure 32: CDF of the System Failure Probability Comparing Several Discretization Schemes for the Initial Water Level (5-Branch Scheme for Operator Action Timing)

The upper and lower limits on the plot for the system failure probability can be verified by analyzing the results of a single DET simulation with the epistemic parameters held at either their lowest possible values (yielding a system with a lower failure probability) or at their highest values. For the case of the 5-branch strategy, the best-case scenario (or the safest possible system) occurs when the discharge coefficient is lowest (0.72), the error factor describing the operator action timing is lowest (1.8) and the probability of failure of the valve is 0, yielding a failure probability of 1.0e-3. The worst case scenario occurs when the discharge coefficient is 0.98, the error factor is 2.2 and the probability of valve failure is reasonably high at 2e-3, yielding a failure probability of 0.0518.

Figure 33 shows the RAVEN-generated results for each operator action discretization scheme which show good agreement with the results reported in Karanki *et al*. [9]. This good agreement (in conjunction with the results reported

in Table 9) demonstrate that RAVEN is capable of generating predicable results using the DET approach, and provides confidence in moving forward with the development of a code interface to allow for DET simulations using TRACE.

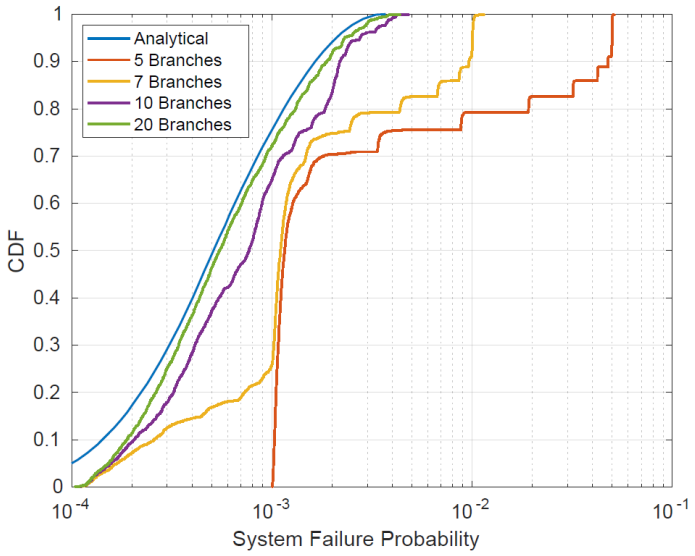

Figure 33: RAVEN-Generated Results for System Failure Probability Considering both Aleatory and Epistemic Uncertainties

4.2 Spent Fuel Pool

A study was first performed to assess an appropriate level of discretization of the probability distributions describing the failures of components in the SFP system. Four scenarios were investigated and are summarized in Table 10.

Table 10: Scenarios to Determine Discretization Scheme

Component Failures	Sampling Frequency
PU1	Every 10 hours
PU1	Every 20 hours
PU1, PU2 and PU3	Every 10 hours (100 s between components)
PU1, PU2 and PU3	Every 20 hours (100 s between components)

Figure 34 shows the temporal evolution of the pool temperature for the two DET simulations examining failures that occur every 10 hours. It can be seen that when PU1 is the only failure considered, there is a single branch every 10 hours that results in a failure of the system (i.e., the average temperature in the pool reaching 353 K). In contrast, when failures of all three pumps are considered, there are several branches (corresponding to different possible combinations of component failures) that result in a failure of the system at each interval.

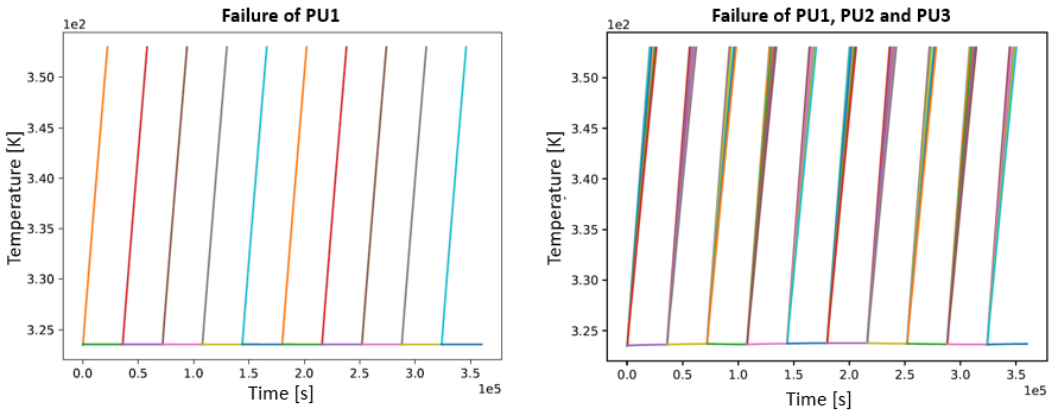

Figure 34: Temporal Evolution of Full-Scale SFP DET with Only Pump Failures Included

Figure 35 shows the evolution of the resulting failure probabilities over the mission time for all four scenarios. Each point on the plots represents a single

branch that resulted in system failure. The corresponding conditional probability of each branch is calculated and integrated across the 100 hour mission time. It can be seen that in the DET simulations where only PU1 fails, the two sampling strategies result in identical failure probability distributions over the mission time. In the case of the DET simulations where all three pumps fail, while the probability distributions are not identical, they both exhibit the same linear trend toward a common value at the mission end time. This behaviour is expected due to the fact that the probability distributions describing the failure events are linear in the sampling range of interest, and that the resulting calculated system failure probability is an integration of the probability of failed branches as a function of the mission time. As discussed in Section 3.2.1, there is a desired balance between accuracy of the results of a DET simulation, and the computational time required to achieve the results. The results shown in Figure 35 provide justification to move forward with the 20-hour discretization scheme in the subsequent simulations due to the fact that this scheme provides a good estimate of the failure probability, and utilizes significantly less computational resources than a finer discretization scheme.

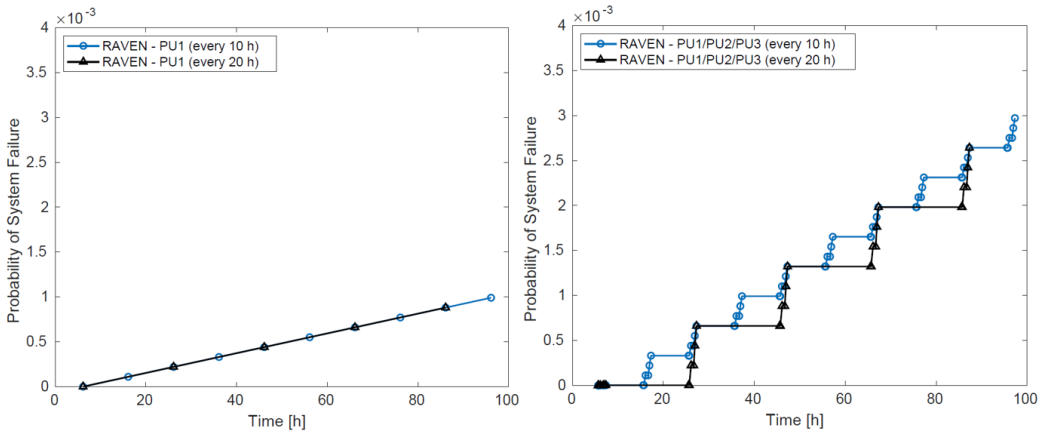

Figure 35: Calculated System Failure Probability for DETs of the SFP with Different Component Failures and Different Discretization Schemes

Using the 20-hour discretization scheme, a first attempt was made at simulating the full-scale SFP DET including all system and component failures (with 100 s between individual component failure times at each of the 20-hour intervals). It was observed that while the possible combinations of system and component failures were generated at the first sampling interval (i.e., failures initiated near time = 0 s), the branching logic did not advance to the next 20-hour interval.

Figure 36 shows the temporal evolution of the temperature for the branches that were generated in this DET. A crucial part of the DET logic is to spawn two new branches at each branching point: one to simulate the system response given that the event happened, and one to continue simulating as if the event did not happen. It is in this way that the DET is able to explore all possible pathways that they system may take. In this particular case, the DET is not generating at least one of the branches that represents the system response if no failures have occurred, and is consequently not able to explore the entire system response[5].

[5] The cause of this was determined to be rooted in the way in which restart files are generated for failures of the valves and heat exchangers.

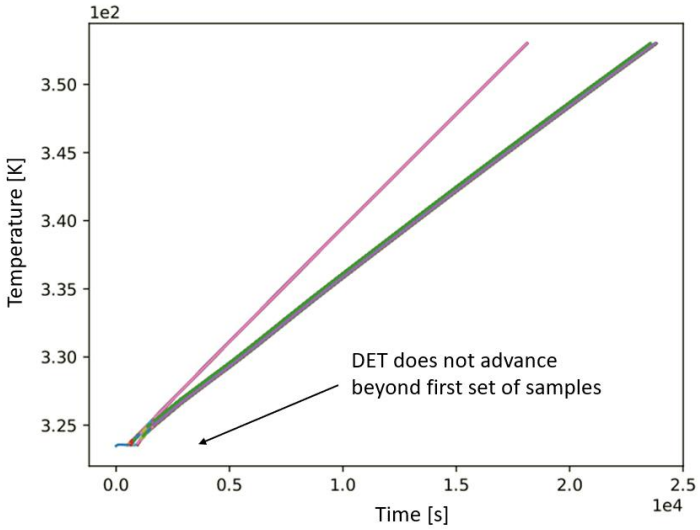

Figure 36: Temporal Evolution of Temperature – Unsuccessful Full-Scale SFP DET Including All System and Component Failures

The issue of how to restart properly was not solved, and it was decided to move forward and simulate the SFP scenario without inclusion of the failures of valves and the heat exchangers. As seen in Table 6, the failure rates for these components are orders of magnitude less than other components (e.g., pumps) and systems (e.g., the UCWS). As such, it is not expected that the exclusion of these components from the DET simulations will significantly affect the calculated overall system failure probability.

With the exclusion of the valve and heat exchanger failures, two new DET simulations were run on the SFP model: one where the failures were initiated every 10 hours, and one where they were initiated every 20 hours. The overall system failure probability was calculated for both, and is shown in Figure 37 along with the failure probability as reported in Chraibi *et al.* [10].

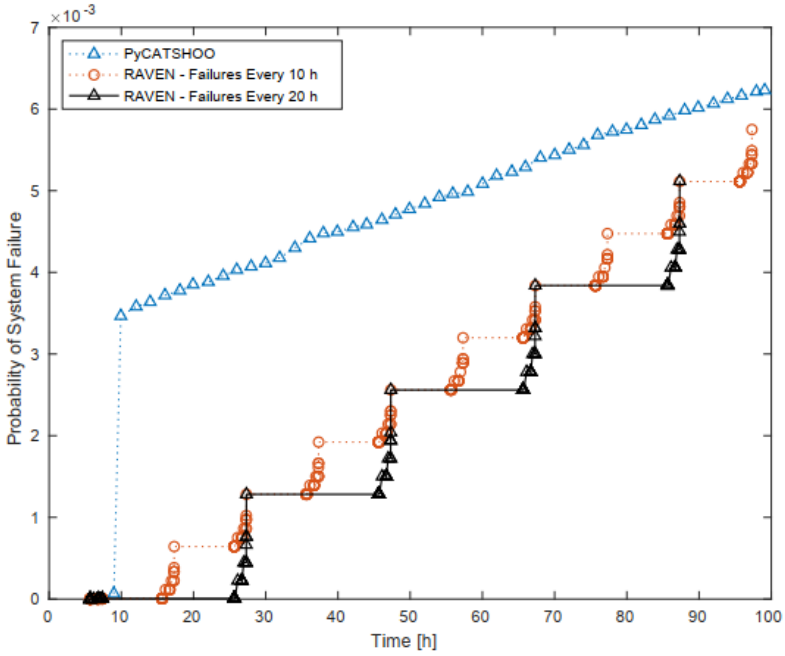

Figure 37: RAVEN-Generated Failure Probability for the SFP System

The following observations can be made on the results shown in Figure 37:

- The calculated failure probability using RAVEN and TRACE is on the same order of magnitude as the PyCATSHOO results.
- The failure probability is zero in both RAVEN and PyCATSHOO calculations between the initial time of the transient and the earliest time at which the system may fail. The time it takes to heat the system from 50℃ to the failure temperature of 80℃ is observed to be approximately 9 hours in the PyCATSHOO model, and 6 hours in the TRACE model. During this time, it is not possible for the system to fail, resulting in a failure probability of zero.
- At each sampling interval, all combinations of the modeled failures result in branches that reach the failure criteria for the system within a relatively small window of time (approximately 4 hours). Given the

modeling strategies implemented in the TRACE model, this observation is not unexpected: each of the modes of failure results in an inability of the cooling system to transfer heat from the pool to the river by way of a loss of flow to the heat exchangers. The heat up rate for the system is slightly different depending on which heat exchanger (and which side: tube or shell) is affected.

- The PyCATSHOO failure probability has an initial step increase to ~0.004 that is not observed in the RAVEN failure probability calculation. The reason for this step increase is not currently understood. Failures that are initiated at the beginning of the mission (i.e., near time = 0 s) result in branches carrying conditional probabilities that are close to zero. In this way, the cluster of RAVEN-generated branches resulting in system failure before 10 hours are observed to contribute very little to the calculated failure probability. The next sampling interval for failures results in branches carrying larger conditional probabilities (due to the location of the samples on the PDFs describing their behaviour), resulting in more significant contribution to the calculated failure probability. Over the mission time, this calculated failure probability is an integration of the conditional probabilities of branches that result in system failure, and the gradual increase observed with the RAVEN-generated results is reasonable.

It is suspected that the discrepancy between the RAVEN and PyCATSHOO-generated results is caused by differences in modeling choices for the SPF and its system failure mechanisms between the two studies. In an effort to verify this, several additional studies were performed (described in detail in Appendix D). These studies utilize the RAVEN-TRACE interface to perturb systems modeled in TRACE that are more simplistic in nature than the SFP system, and that have

failure mechanisms that are well-understood. The calculated system failure probabilities show very good agreement with results generated using other codes and strategies. Of particular note is the second study which examines a tank level control system and compares the system failure results calculated using RAVEN-TRACE and PyCATSHOO. In addition to the good agreement between the results generated using the two PRA software platforms, the overall strategy employed using RAVEN to generate the DET is identical to the strategy employed for the SFP study (i.e., the use of the defined exponential distribution rate parameters and the discretization of the branching thresholds over the mission time). It is for this reason that the following conclusion has been made: lack of clarity in the modeling choices implemented in Chraibi *et al.* [10] (in particular, lack of clarity surrounding the failure modes of systems and components) has likely led to differences in the TRACE realization of the SPF that affect the final system failure calculation, and is the likely cause of the discrepancy between the RAVEN-TRACE and PyCATSHOO-generated system failure results for the SFP system.

5. CONCLUSIONS & FUTURE WORK

The overall purpose of the work outlined in this thesis has been to demonstrate and further develop the capabilities of the RAVEN DPRA software framework in order to leverage its near-future application to the risk assessment of CANDU reactor designs. Throughout the various studies performed, RAVEN has been demonstrated to be capable of generating predictable results using the DET methodology on systems which are well-understood. The calculated system failure probabilities for the following scenarios were shown to have good agreement with either an analytical solution to the problem, or the failure probabilities as calculated by a different DPRA software:

- A depleting tank scenario which considers both epistemic and aleatory uncertainties including the time of operator intervention and the failure on demand of the valve to terminate the conditions leading to system failure.
- A level control system scenario in which the active components are subject to random failures over the specified mission time.

A significant portion of this work has been the development of a code interface that allows RAVEN to drive DET simulations using TRACE. RAVEN was designed with the purpose of being capable of driving any system code; however the DET capability requires careful coordination between the input and output requirements in both RAVEN and the system code. The RAVEN-TRACE interface described in this thesis has been demonstrated to produce predictable results for the two scenarios (depleting tank, and level control) mentioned above.

A model of a simplified SFP was developed and DET simulations of this model were driven using the RAVEN-TRACE interface. Successes in this portion of the research include further demonstration of the mechanics of the interaction between RAVEN and TRACE (i.e., terminating the code when branching thresholds have been met, and restarting with modified inputs), and a demonstration that the resulting overall system failure probability is on the same order of magnitude as results produced using PyCATSHOO. A discrepancy in the shape of the calculated system failure probability as a function of time was observed between results generated by PyCATSHOO and RAVEN-TRACE. This discrepancy was investigated and was concluded to be caused by differences in modeling choices between the two studies.

In order to build on the work outlined in this thesis, the following studies are suggested:

- The RAVEN-TRACE interface that has been developed as a part of this research should be used to drive a DPRA study of accident conditions applied to a high-fidelity CANDU simulation model. A first step in this study would be to drive thermalhydraulic simulations using TRACE. It may then be beneficial to drive simulations in which a reactor physics code such as the Purdue Advanced Reactor Core Simulator (PARCS), has been integrated. Depending on the mechanics required to run coupled TRACE-PARCS simulations, modifications to the RAVEN-TRACE interface may be required, though it is not expected that the general framework of the interface would need to be changed.
- In relation to this, it will be beneficial to apply a RISMC approach to the quantification of the relationship between safety margins and risk estimates for the CANDU reactor design, as this would be a first-of-its-kind study, and could eventually be used as input in licensing and regulatory decision-making processes. It will also be beneficial to investigate the use of ROM's in order to reduce the computational time required for these studies.
- Finally, the RAVEN-TRACE interface could be further developed to include the capability to model partial system failures. The current framework employs a binary approach to failures, where systems and components are considered to be either in an operational or failed state. The inclusion of partial failures may improve the accuracy of the risk estimates calculated using the DET approach. This would require modifications to both the interface described in this thesis, and the fundamental RAVEN framework for DET sampling strategies.

6. REFERENCES

[1] "Reactor Facilities, Licence Application Guide: Licence to Operate a Nuclear Power Plant," Canadian Nuclear Safety Commission, Ottawa, 2017.

[2] H. Taesung, "Probabilistic Risk Assessment for a Loss of Coolant Accident in McMaster Nuclear Reactor and Application of Reliability Physics Model for Modeling Human Reliability," McMaster University, Hamilton, 2006.

[3] P. Labeau, C. Smidts and Swaminathan, "Dynamic Reliability: Towards an Integrated Platform for Probabilistic Risk Assessment," *Reliability Engineering and System Safety,* vol. 68, pp. 219-254, 2000.

[4] C. Smith, C. Rabiti, R. Martineau and R. Szilard, "Light Water Reactor Sustainability Program: Risk-Informed Safety Margins Characterization (RISMC) Pathway Technical Program Plan," Idaho National Laboratory, Idaho Falls, 2016.

[5] "WASH-1400, The Reactor Safety Study," 1975.

[6] T. Aldemir, "A Survey of Dynamic Methodologies for Probabilistic Safety Assessment of Nuclear Power Plants," *Annals of Nuclear Energy,* vol. 52, pp. 113-124, 2013.

[7] T. Aven, "Risk Assessment and Risk Management: Review of Recent Advances on Their Foundation," *European Journal of Operational Research,* vol. 253, pp. 1-13, 2016.

[8] A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, R. Kinoshita and A. Naviglio, "RAVEN and Dynamic Probabilistic Risk Assessment: Software Review," Idaho National Laboratory, Idaho Falls, 2014.

[9] D. Karanki, S. Rahman, D. V.N. and O. Zerkak, "Epistemic and aleatory uncertainties in integrated deterministic and probabilistic safety assessment: Tradeoff between accuracy and accident simulations," *Reliability Engineering and System Safety,* vol. 162, pp. 91-102, 2017.

[10] H. Chraibi, J. Houdebine and A. Sibler, "PyCATSHOO: Toward a New Platform Dedicated to Dynamic Reliability Assessments of Hybrid Systems," in *13th International Conference on Probabilistic Safety Assessment and Management*, Seoul, Korea, 2016.

[11] J. Devooght and C. Smidts, "Probabilistic Reactor Dynamics-I: The Theory of Continuous Event Trees," *Nuclear Science and Engineering,* vol. 111, no. 3, pp. 229-240, 1992.

[12] A. Verma, S. Ajit and D. Karanki, Reliability and Safety Engineering, London: Springer, 2016.

[13] D. Karanki and V. Dang, "Quantification of Dynamic Event Trees - A Comparison with Event Trees for MLOCA Scenario," *Reliability Engineering and System Safety,* vol. 147, pp. 19-31, 2016.

[14] Y. Vorobyev and P. Kudinov, "Development and Application of a Genetic Algorithm Based Dynamic PRA Methodology to Plant Vulnerability Search," in *ANS PSA International Topical Meeting on Probabilistic Safety Assessment and Analysis*, Wilmington, 2011.

[15] C. Acosta and N. Siu, "Dynamic Event Trees in Accident Sequence Analysis: Application to Steam Generator Tube Rupture," *Reliability Engineering and Systems Safety,* vol. 41, pp. 135-154, 1993.

[16] J. Izquierdo-Rocha and M. Sanchez-Perea, "Application of the Integrated Safety Assessment Methodology to the Emergency Procedures of a SGTR of a PWR," *Reliability Engineering and System Safety,* vol. 45, pp. 159-173, 1994.

[17] M. Kloos and J. Peschke, "MCDET: A Probabilistic Dynamics Method Combining Monte Carlo Simulation with the Discrete Dynamic Event Tree Approach," *Nuclear Science and Engineering,* vol. 153, pp. 137-156, 2006.

[18] 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, pp. 3457-3467, 2008.

[19] H. Lee, T. Kim and H. Gyunyoung, "Application of Dynamic Probabilistic Safety Assessment Approach for Accident Sequence Precursor Analysis: Case Study for Steam Generator Tube Rupture," *Nuclear Engineering and Technology,* vol. 49, pp. 306-312, 2017.

[20] A. Alfonsi, D. Mandelli, J. Cogliati, A. Naviglio, C. Rabiti and R. Kinoshita, "Dynamic Event Tree Analysis through RAVEN," in *ANS PSA 2013, International Topical Meeting on Probabilistic Safety Assessment and Analysis*, Columbia, SC, 2013.

[21] Sandia National Laboratories, "DAKOTA Reliability Methods Applied to RAVEN/RELAP-7," Albuquerque, 2013.

[22] Sandia National Laboratories, "Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: User's Manual," Albuquerque, 2018.

[23] P. Fearnhearn, J. Bierkens, M. Pollock and G. Roberts, "Piecewise Deterministic Markov Processes for Continuous-Time Monte Carlo," 2016.

[24] H. Chraibi, "Getting Started with PyCATSHOO," Energy Research and Development, France, 2017.

[25] Idaho National Laboratory, "RAVEN Theory Manual and User Guide," Idaho Falls, 2017.

[26] C. Smith, C. Rabiti, R. Martineau and R. Szilard, "Light Water Reactor Sustainability Program: Risk-Informed Safety Margins Characterization (RISMC) Pathway Technical Program Plan," Idaho National Laboratory, Idaho Falls, 2016.

[27] "TRACE V5.0 USER'S MANUAL - Volumes 1 and 2, United States Nuclear Regulatory Commission," 2011.

[28] K. Groves and D. Novog, "A TRACE/PARCS Coupling, Uncertainty Propagation and Sensitivity Analysis Methodology for the IAEA ICSP on Numerical Benchmarks for Multi-Physics Simulation of Pressurized Heavy Water Reactor Transients," McMaster University, 2020.

[29] T. Aldemir, "Computer-Assisted Markov Failure Modeling of Process Control Systems," *IEEE Transactions on Reliability,* pp. 133-144, 1987.

[30] F. Zhou and D. Novog, "RELAP5 Simulation of CANDU Station Blackout Accidents with/without Water Make-up to the Steam Generators," *Nuclear Engineering and Design,* vol. 318, pp. 35-53, 2017.

[31] P. Sabouri, "Large LOCA BEAU Analysis of a Generic 900 MW CANDU Plant," McMaster University, Hamilton, 2010.

APPENDIX A ADDITIONAL LESSONS LEARNED USING RAVEN TO PERTURB TRACE SIMULATIONS

During the development of the RAVEN-TRACE interface, the following additional lessons were learned:

- RAVEN cannot handle a situation where two failures occur at exactly the same time. The number of generated branches is seen to increase without end if this situation exists.
- At times, running branches in parallel can cause RAVEN to crash. When this occurs, debugging is difficult due to a lack of useful information about the error. Through trial and error, the issue has been observed to disappear if the simulations are run without this parallel feature.
- 'Simple trips' cannot be used for either branching or system failure trips. There is no direct indication that this could cause a problem, however they are not capable of terminating the code.
- Within the original TRACE input file, the user can specify the frequency with which the code dumps data to the graphics, restart and output files. The user should set low frequencies for the restart and output files, as only the information at the end of the branch is useful.

APPENDIX B STEADY-STATE BEHAVIOUR OF TRACE SPENT FUEL POOL MODEL

Figure 38 and Figure 39 show the average temperature within the SFP and pressures at several depths over the 100 hour mission time with no initiated component or system failures. It can be seen that the model rejects heat from the system at a rate that allows for conditions to be held at an average pool temperature of 50°C (323 K), which is the initial water temperature (assumed to be the average steady state temperature in the pool) reported in Chraibi *et al.* [10].

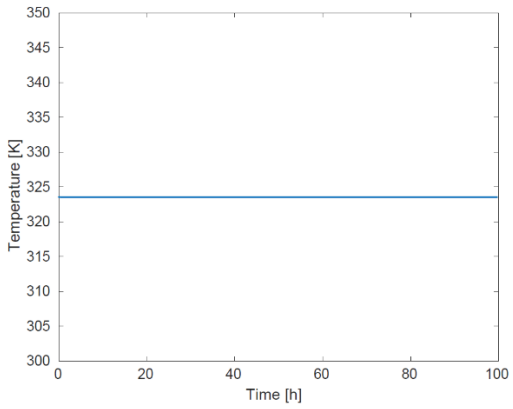

Figure 38: Steady State Temperature in SFP Over 100 Hour Mission Time

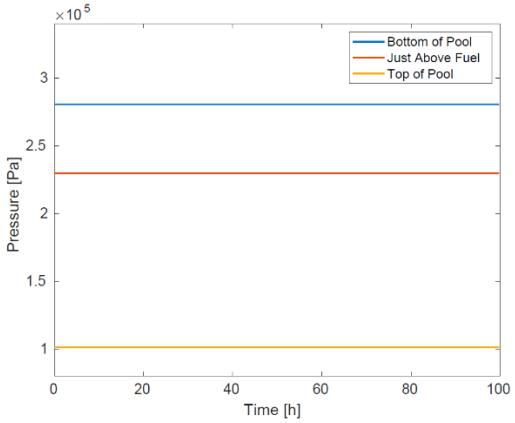

Figure 39: Steady State Pressures in SFP Over 100 Hour Mission Time

APPENDIX C COMPARISON OF PUMP FAILURE FLOW RATES FOR THE SFP DET SIMULATIONS

In some cases for the SFP DET simulations, if the rate of flow in a pump is reduced to 0 kg/s when a failure is initiated, TRACE will fail to converge on a solution and eventually crash. As such, an investigation was performed to determine the lowest possible value that the flow rate can be reduced to while still allowing the TRACE calculations to run to completion.

Since the flow rate in each of the loops within a coolant train is different, this study looks at a percentage of the nominal flow rate and applies these reduced rates to the appropriate loops. Four percentages were examined: 0.5%, 1%, 2% and 5%. In each case, the DET simulations were completed successfully; however, in the 0.5% case, longer simulation times were observed, with multiple warning messages of convergence issues. It is for this reason that 1% of the nominal flow rate has been applied in the DET simulations for the SFP.

Figure 40 shows the range in times at which the system reaches the failure temperature when applying each of the different failure flow rates. It can be seen that the difference in time of failure between the 0.5% and 1% cases is less than 0.5 h.

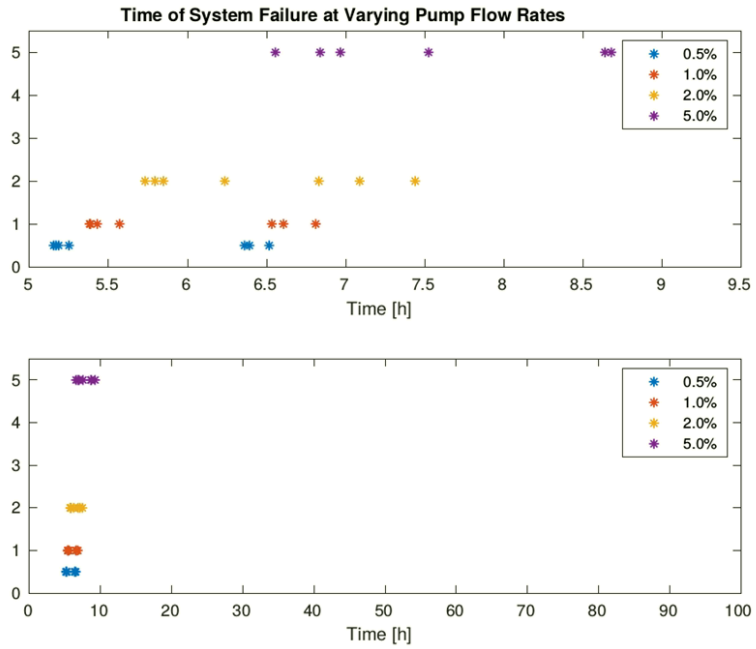

Figure 40: Variation on System Failure Time Using Different Pump Flow Rates during Failure Conditions

APPENDIX D EFFORTS TO DEMONSTRATE THE VALIDITY OF THE APPROACH USED IN THE SFP ANALYSIS

The following two studies were performed in order to aid in demonstrating that the RAVEN-TRACE interface can produce results that are consistent with either results in literature, or results generated using a different PRA driving software.

D.1 Depleting Tank Problem: RAVEN Driving TRACE Simulations

The depleting tank problem that was examined in Section 3.2.1 utilized a simple code written using Python to simulate the thermalhydraulic behavior within the tank. In this study, the tank was modeled instead using TRACE and perturbed using the RAVEN-TRACE interface.

Figure 41 shows the TRACE nodalization of the system. The tank is modeled as a vertical pipe with an atmospheric pressure boundary condition at the top. The discharge path is a pipe at the bottom of the tank that is connected to another atmospheric pressure boundary condition, and this flow path is controlled through the use of a valve. All of the epistemic and aleatory parameters examined in the original study have been considered except for uncertainty in the discharge coefficient (C) of the valve.

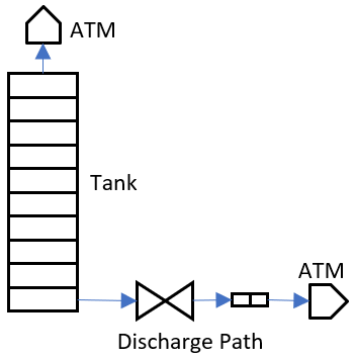

Figure 41: TRACE Model of Depleting Tank System

In order to compare the DET simulations generated for this system with results using the RAVEN-Python approach, the RAVEN-Python approach was modified to not include variation in C, and was re-run.

Figure 42 shows the results of DET simulations using the original RAVEN-Python approach and the 5-branch operator action discretization scheme, without variation in C. Several sets of simulations were performed, each with C held at a different constant value within its range of uncertainty. A low value for C represents a situation where there is more time for the operator to act, and therefore the potential for a lower overall system failure probability. A high value for C represents the opposite. Note that while C is held constant here, there are two other parameters being varied. One is the probability that the valve fails on demand, and the other is the parameters used to define the operator action CDF (which is then sampled at known/constant values on the CDF, but since the shape of the CDF varies, the operator action times vary). It is also significant to note that when C is below ~0.75-0.8, the time window is large enough that the sampled values for the operator action time do not play much of a role: they almost always result in mission success. This is why there is minimal variation in the first three curves.

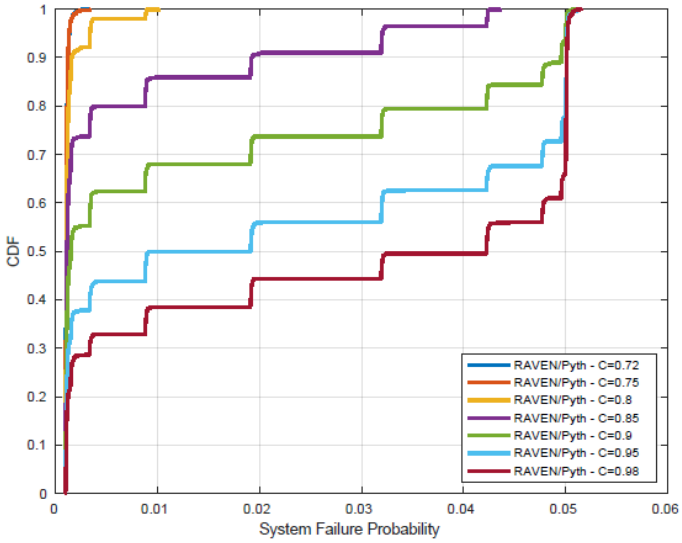

Figure 42: RAVEN-Python Results at Different Discharge Coefficient Values

The TRACE model of the depleting tank has a depletion time that is greater than the time generated using the explicit time window formula provided in Karanki *et al.* [9] (a formula which includes C). As such, the time window from the TRACE model was used to determine the value of C that would result in the same window. This value was found to be 0.765, and was subsequently used in the RAVEN-Python simulations.

It is of note that in order to perturb the initial water level in the tank, a new method needed to be written into the RAVEN-TRACE interface. Unlike all other failures and varied parameters discussed in this thesis so far, the initial height of the tank is described through a series of values specifying the void fraction in each cell of the thermalhydraulic component. As such, this new method takes the initial height as input, calculates the required void fraction for each cell, and modifies the input file appropriately.

Figure 43 shows comparisons of the RAVEN-Python and RAVEN-TRACE results for the 5 and 7 branch operator action schemes. In both cases, the discharge

coefficient was held constant at 0.765 in the Python code in order to reproduce the same window of time available to the operator as observed in the TRACE model. It can be seen that the results generated using the RAVEN-TRACE interface show good agreement with the results generated using RAVEN-Python.

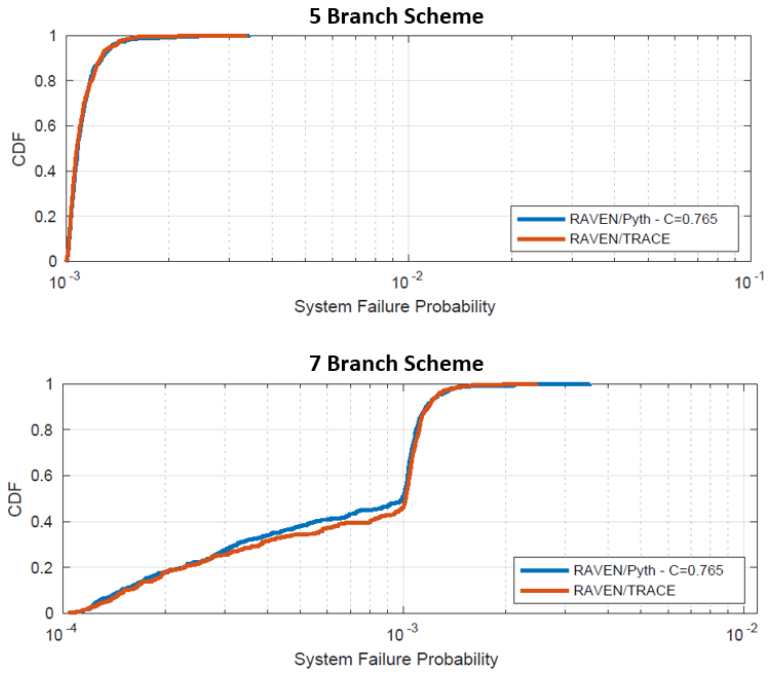

Figure 43: Comparison of RAVEN-Python and RAVEN-TRACE Results

D.2 Level-Control System Problem

The system shown in Figure 44 was modeled in both PyCATSHOO and in TRACE. The system consists of a tank of water, where the level of the water is controlled by the activation of an inlet pump and an outlet valve. The control system is designed to keep the water level between 6 and 8 m under normal operating conditions; however, the pump and the valve are subject to random failure. Each of the components can fail in one of two ways: stuck open, or stuck closed. Once

failed, the component cannot be repaired, and the transition rate from operational to failed state is described by an exponential probability distribution with the rate parameters (per hour) shown in Figure 44. There are two possible overall system failure criteria that may be met: overflow (where the water level rises above 10 m) or dryout (where the water level depletes below 4 m).

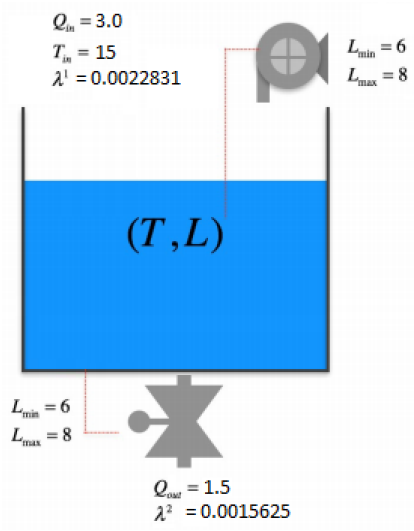

Figure 44: Level Control System Schematic

This system was modeled in TRACE, and perturbed using the same strategy employed for the SFP study. The exponential probability distributions describing the failures of the two components were discretized across the applicable range (a 60 hour mission time), and these discrete values were used as the branching triggers for the DET simulation. Figure 45 shows the evolution of the water level in 1 hour of the DET simulation and Figure 46 shows both the RAVEN and PyCATSHOO-generated results for the failure probability of the two outcomes (overflow and dryout). The RAVEN results show good agreements with the PyCATSHOO results.

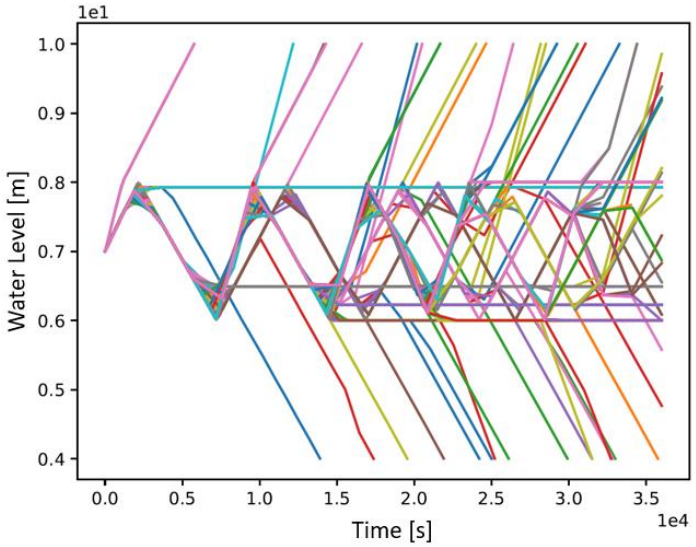

Figure 45: Temporal Evolution of Water Level in DET Simulations of Level Control System

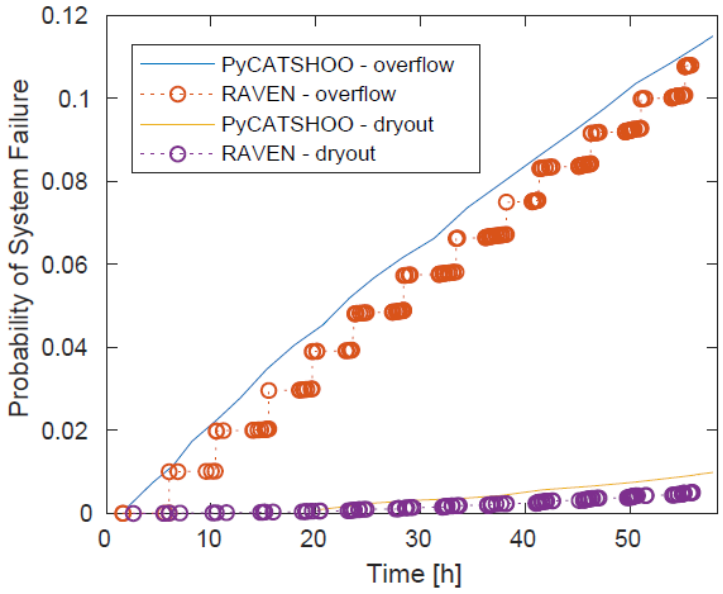

Figure 46: Comparison of System Failure Probability of Level Control System – PyCATSHOO vs. RAVEN