DYNAMIC PROBABILISTIC RISK ASSESSMENT WITH PyCATSHOO: THE CASE OF THE EMERGENCY POWER SUPPLY OF A NUCLEAR POWER PLANT

Keoni Sanny, Claudia Picoco, Tunc Aldemir
The Ohio State University, Columbus, OH, 43210, United States of America
sanny.2@osu.edu, picoco.1@osu.edu, aldemir.1@osu.edu

The application of the PyCATSHOO software modeling library to a nuclear power plant's emergency power supply system (EPSS) is introduced. The EPSS was previously established as a benchmark for dependability assessment techniques applied to dynamic stochastic systems.

I. INTRODUCTION

Traditional methodologies used in Probabilistic Risk Assessment (PRA) include the static event-tree (ET) / fault-tree (FT) methods (Ref. 1). ETs predict possible ways a system – e.g., nuclear power plants (NPPs) – can evolve starting from an initiating event and considering a set of events assumed to either occur or not occur (Top Events). FTs evaluate the probability of occurrence of a Top Event (TE) in terms of a set of contributing basic events.

The ET/FT is the standard approach currently used for quantifying the risk of NPP operation and is required by regulators. Although static in principle, several techniques have been developed to embed dynamic aspects, such as recoveries, within the analysis, as those proposed for example in (Ref. 2) and in (Ref. 3).

As time is not explicitly modeled with the traditional ET/FT approach, accounting for the interaction among hardware/software/process/human behavior is difficult and may require subjective judgement. Furthermore, although the order of events is preset by the analyst in the ET/FT approach, the order of events as well as the mode and frequency of system failure can be sensitive to uncertainties in the process physics (Refs. 4 - 6) which may be difficult to account for with the ET/FT approach in a comprehensive and verifiable fashion. Such a sensitivity to process physics is particularly relevant for future reactors which are expected to rely on passive components and natural phenomena for safety.

Approaches that are often referred to as Dynamic PRA (DPRA) methodologies have been proposed in literature to meet such challenges of the traditional ET/FT approach. An overview of DPRA methods proposed to date can be found in (Ref. 7). PyCATSHOO (Ref. 8) is a DPRA tool that has been developed by the Électricité de France (EDF) research and development team that is particularly effective in modeling complex maintenance activities, and complements capabilities of the system level applicable DPRA tools (e.g. Refs. 9 - 12). PyCATSHOO keeps track of the time of occurrence of events, allows recoveries and reconfigurations, and models both deterministic continuous and stochastic discrete phenomena by implementing the concept of the Piecewise Deterministic Markov Process (PDMP) (Ref. 13). It is well known that phenomena that are non-Markovian in nature can be included in a Markovian framework using a sufficient number of auxiliary variables. (Ref. 14) gives an example of such an inclusion process for modeling aging. PyCATSHOO also allows the user to integrate a set of differential equations within the analysis to describe the physical dynamic evolution of the system as stochastic events (e.g., failures and recoveries) occur.

This paper aims at presenting the application of PyCATSHOO for the dependability assessment of the emergency AC power system of a NPP. Dependability analysis in this context indicates that the frequency and duration of emergency power loss events are captured (Ref. 15). Dependability analysis of the NPP AC power system is crucial for the safety of the plant because AC power provides electricity to the active safety systems in case of an accident (Ref. 16). The case of the AC power system has also been widely assessed in literature using static methods, as for example in (Ref. 17).

Two cases have been analyzed: (i) a simple AC power system used to introduce the tool and (ii) the AC power system of a NPP to show a real application. Both systems have been previously analyzed in (Ref. 18) and (Ref. 15), respectively, using the Boolean Driven Markov Process (BDMP) approach. The results obtained in (Ref. 15) and (Ref. 18) have been used to validate our model.

This paper is organized as follows. Section II provides details on the PyCATSHOO tool, and Section III showcases PyCATSHOO's application to a simple electrical system. In Section IV, PyCATSHOO is applied to the emergency power supply system (EPSS) of a NPP. Finally, some conclusions are drawn in Section V.

II. PyCATSHOO Tool

Risk assessment of systems is concerned with two basic behaviors: continuous (e.g. the evolution of temperature in a heated room) and discrete (such as the flipping of a switch) (Ref. 13). PyCATSHOO (PythoniC Object Oriented Hybrid Stochastic AuTomata) is a tool developed by EDF that is capable of modeling both of these behaviors by using stochastic hybrid automata for

continuous, deterministic evolution and discrete stochastic automata for purely discrete events. Modeling with automata allows the system to be defined in terms of states and transitions among states (Ref. 20). PyCATSHOO then uses Monte Carlo sampling for simulation.

PyCATSHOO is a freeware C++ written library with APIs (Application Programming Interfaces) both in Python and C++ (Ref. 19).

PyCATSHOO tools come in three categories (Ref. 20):

1. Modeling of individual components,
2. Modeling of systems composed of various component arrangements, and
3. Analysis of results.

All three categories are used in the two cases considered in this paper.

PyCATSHOO can be used on Windows, Linux, or Mac (Ref. 13). More detailed description of the characteristics of the PyCATSHOO library and how to use it can be found in PyCATSHOO's user's guide (Ref. 20).

III. CASE STUDY 1: A SIMPLE SYSTEM

In this application, PyCATSHOO is used to model a relatively simple electrical system. The model is validated using previously established dependability assessment results (Ref. 18).

In Section III.A the AC power system is described. In Section III.B, parameters used in the analysis are provided. Finally, the PyCATSHOO model and results are presented in Section III.C.

III.A. System Description

A representation of the physical system is shown in Fig. 1. The goal of the simple system is to supply electricity to a *BUSBAR*, a component which distributes electricity to critical equipment (e.g., instrumentation or pumps in a nuclear power plant) (Ref. 15). The system has redundancy in that electricity can be supplied to the *BUSBAR* either from the *GRID* (i.e., transmission/distribution network) or from a backup *diesel generator*.

As seen in Fig. 1, the *GRID* supplies power to the *BUSBAR* through Line 1 (components *CB_up_1*, *transfo1*, and *CB_dw_1*) in the system's normal operating state. Should one of the components in Line 1 fail in operation (i.e., experience a short circuit), circuit breaker *CB_dw_2* will attempt to close in order to feed the *BUSBAR* through Line 2 (*CB_up_2*, *transfo2*, and *CB_dw_2*). Should powering the *BUSBAR* from the *GRID* become impossible, *CB_dies* attempts to close and the *diesel generator* attempts to start.

A short circuit occurs on a component when the electrical current is allowed to travel on an undesired path

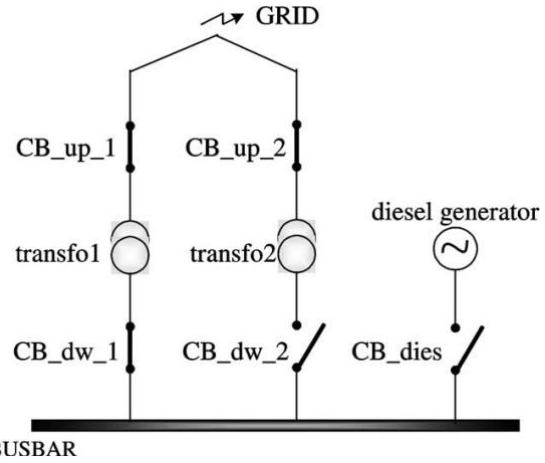

Fig. 1. The initial state of the simple model. *CB* indicates circuit breaker and *transfo* indicates transformer. (Ref. 18)

of low or zero impedance, such as a connection to the ground (not shown in Fig. 1). If the short circuit is not isolated, an extremely high current arises on the path between the power source and the short circuit, possibly damaging all electrical components on the path and the power source itself. Circuit breakers are normally able to isolate the short circuit by automatically opening when the abnormally high current is detected (Ref. 15). Short circuit propagation can thus disable the *GRID* in this system if, for example, *transfo2* experiences a short circuit and *CB_up_2* fails to open (Ref. 18). All components are repairable. For feeding the *BUSBAR*, the system prioritizes Line 1 over Line 2 and Line 2 over the *diesel generator* after every repair (Ref. 18).

III.B. Failure and Recovery Rates

In this system, no deterministic events occur. The failure rate for each component is defined by the parameter λ which represents average number of failures per hour. Components can also experience on-demand failure (e.g., the *diesel generator* failing to start or a circuit breaker failing to open or close). On-demand failure is represented by γ, the average number of failures per attempt. The recovery rate of a component from on-demand or in-operation failure is defined by the probability parameter μ, the average number of repairs per hour. These parameters for this system have been taken from (Ref. 18) as follows for verification of the results against (Ref. 18) results: $\lambda = 10^{-4}$ failures/hour, $\gamma = 10^{-3}$ failures/attempt, and $\mu = 0.1$ repairs/hour for each component. For in-operation failure and recovery, an exponential distribution has been assumed with parameters λ or μ for failure and repair, respectively.

III.C. PyCATSHOO Implementation and Results

The behavior of the system is studied over the course of 10^4 hours or about 1 year. During this mission time, components can fail and be repaired multiple times. The system can therefore fail and recover multiple times, where failure is defined as the *BUSBAR* not being fed by either the *GRID* or the *diesel generator* (Ref. 18).

In Section III.C.1, the assumptions made in the model are described. Section III.C.2 presents analysis and results from the dependability assessment.

III.C.1. Model Assumptions

The assumptions of this model attempt to reproduce the assumptions of (Ref. 18). Particularly:

1. *All components can experience failure except the BUSBAR itself.*
2. *The GRID is modeled as a single component and is also allowed to fail*, for example as a consequence of a climatic event (e.g., a hurricane).
3. *Not every circuit breaker opening or closing is subject to failure on demand in the model.* This assumption is mostly applied to recovery sequences. An example is when *CB_dw_1* has experienced a short circuit and the *BUSBAR* is being fed by Line 2 (meaning *CB_dw_2* has successfully closed). When *CB_dw_1* is repaired, *CB_dw_2* is not subject to on-demand failure: *CB_dw_2* automatically opens and allows Line 1 to power the *BUSBAR*. The goal of this assumption for the illustration in (Ref. 18) was to reduce the number of states to be considered and optimize calculation time since dependability assessment was performed by sequence exploration. It is worth noting that PyCATSHOO does not use sequence exploration. In this respect, on-demand failure for all components could have been implemented here without significant impact on computation time for quantitative results.
4. *A line on which a failure has occurred is powered down so that other components on that line generally do not fail.* If *transfo1* has experienced a short circuit, for example, *CB_dw_1* is not allowed to have a short circuit until Line 1 is back in operation. The fail to open/fail to close states of circuit breakers sometimes present an exception, however. For example, if *transfo1* has a short circuit and *CB_up_1* fails to open and *transfo1* is repaired before *CB_up_1*, Line 1 could be asked to operate since *CB_up_1* is still closed and capable of transmitting electricity. *CB_up_1* could thus experience a short circuit while in its "fail to open" state as a result.

A more detailed description of the system's behavior is provided in (Ref. 18).

III.C.2. Dependability Assessment Analysis and Results

The resulting PyCATSHOO model consists of 5 Python classes: Line 1, Line 2, Line Diesel, Grid, and Busbar. Each component within a class is defined as a different automaton that defines the component's behavior (Ref. 20). A total of 10 automata are therefore defined, corresponding to *CB_up_1*, *transfo1*, *CB_dw_1*, *CB_up_2*, *transfo2*, *CB_dw_2*, *BUSBAR*, *GRID*, *diesel generator*, and *CB_dies*. The number of possible states (e.g., working, available, failed, etc.) in each automaton depends on the component considered.

"Message boxes" are used in PyCATSHOO to allow communication between Python classes, ensuring each component can communicate with the rest of the system (Ref. 20). The model presented here uses 24 message boxes.

Once the system model is created in PyCATSHOO in terms of components (e.g., transformers, circuit breakers, diesel generator) each with an associated λ, γ and μ, the analysis can be performed. The mission time is set to 10^4 hours as indicated earlier and the number of simulations is arbitrarily set to one million.

A PC with a second generation Intel Core i7 processor takes about 13 minutes to simulate 1 million missions, producing values of Mean Time To Failure (MTTF), unreliability, and unavailability within 2% of those listed in (Ref. 18). One million missions also reliably produce the three most common sequences leading to failure according to (Ref. 18). The probability of the fourth most commonly occurring sequence is on the order of 10^{-4}.

PyCATSHOO directly produces probability distribution figures and an HTML containing the most common sequences. Table I shows the three most common sequences as produced by PyCATSHOO.

TABLE I. Three most common sequences

Number of Occurrences	**Average failure length (hours)**	**Fractional probability contribution to failure**	**Sequence description**
1016	4.9	0.33	[[Grid.Failure, 1], [Line_D.DG_RS,1]]
1006	4.97	0.33	[[Grid.Failure, 1],[Line_D.CB_RC,1]]
998	4.67	0.32	[[Grid.Failure, 1],[Line_D.DG_FailureInFunction,1]]

In Table I, *Line_D.DG_RS* indicates that the *diesel generator* failed to start, *Line_D.RC_CB* indicates that the circuit breaker next to the *diesel generator* failed to close,

and *Line_D.DG_FailureInFunction* indicates that the *Diesel Generator* failed while operating. The "1" next to each failure state indicates the state is active; a "0" indicates that the failure state is passive, as in a repair.

Table II shows other results produced by the model from the same run of 1 million missions and the corresponding results in (Ref. 18). To calculate the times shown in Table II, which are clearly larger than the length of a single mission (10,000 hours), each simulated mission was treated as a continuation of the previously simulated mission. To determine the MTTF, the time between each system recovery and the next failure was computed. The summation of these times was then divided by the total number of failures to obtain MTTF.

Unavailability was computed as the time between system failure and the next recovery. If the system did not recover before the end of a mission, the beginning of the next mission was counted as a recovery since the system is automatically reset to its initial state for every simulation.

Figure 2 displays part of the graphical output from PyCATSHOO with the cumulative probability distributions (CDFs) for time to failure *F(t)* since the most recent recovery shown in Fig. 2b and time to recovery *R(t)* since the most recent failure in Fig. 2d. As in Table II, the times in Fig. 2 were produced by stringing adjacent missions together.

PyCATSHOO also makes possible the observation of multiple failures in a single mission. In Figure 3b, the CDF for time to first failure within a mission is displayed. Instead of calculating time to failure since the most recent recovery as in Figs. 2a and 2b, Figs. 3a and 3b use the beginning of each mission as the reference point, so the maximum time to failure is 10,000 hours. The CDF in Fig. 3b does not equal 1 at 10,000 hours since the system does not experience failure in every mission. Instead, the value of *F(t)* in Fig. 3b at the end mission time coincides with the unreliability reported in Table II, as expected.

TABLE II. Simple model results

	PyCATSHOO	**(Ref. 18)**
Total Number of Failures:	3085	-
Total time between recovery and next failure (hours):	1.00e+10	-
MTTF (time/number of failures) (hours):	3.24e+06	3.29e+06
Number of missions experiencing failure:	3080	-
Unreliability (number of failed missions/total number of missions):	0.0031	0.0030
Total time system was unavailable (hours):	14863	-
Unavailability (time unavailable/total simulation time):	1.49e-06	1.51e-06

After the system recovers from its first failure in a mission, it is possible that the system can fail a second time in that mission. Figures 3c and 3d display the occurrences of second failures in a single mission with reference to both the start of the mission and the first recovery.

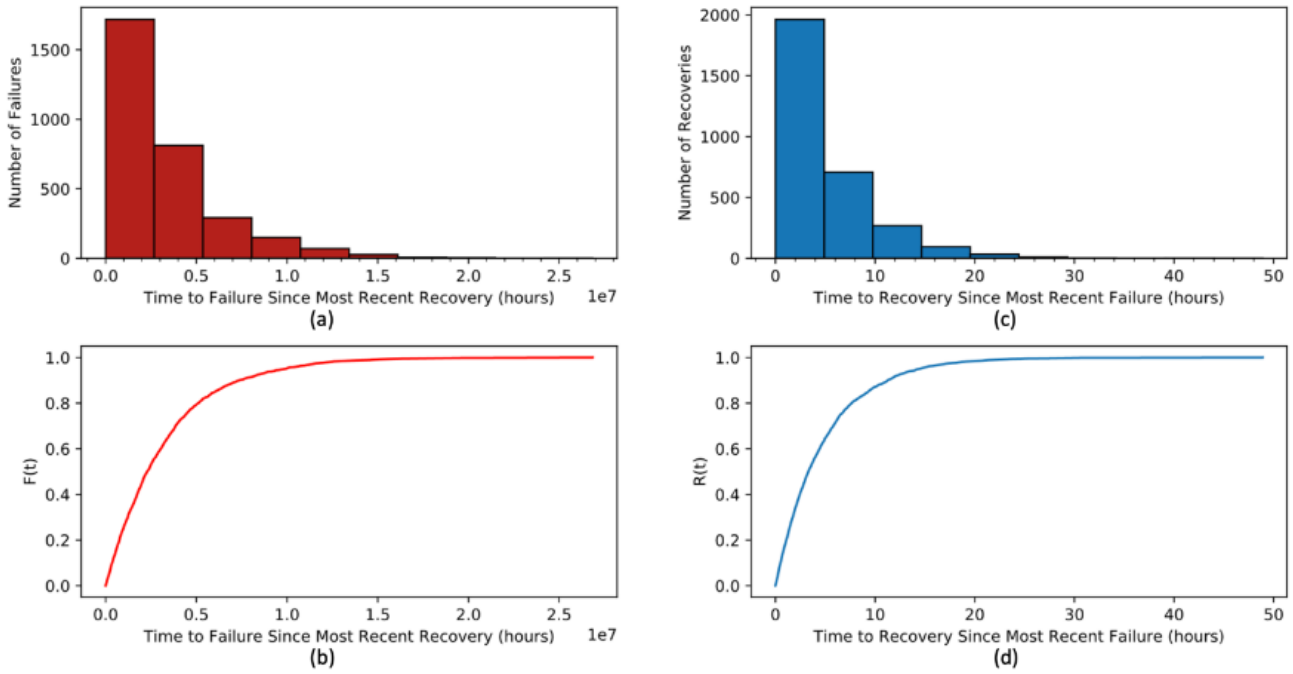

Fig. 2. Case Study 1 failure *F*(t) and recovery *R*(*t*) time distributions.

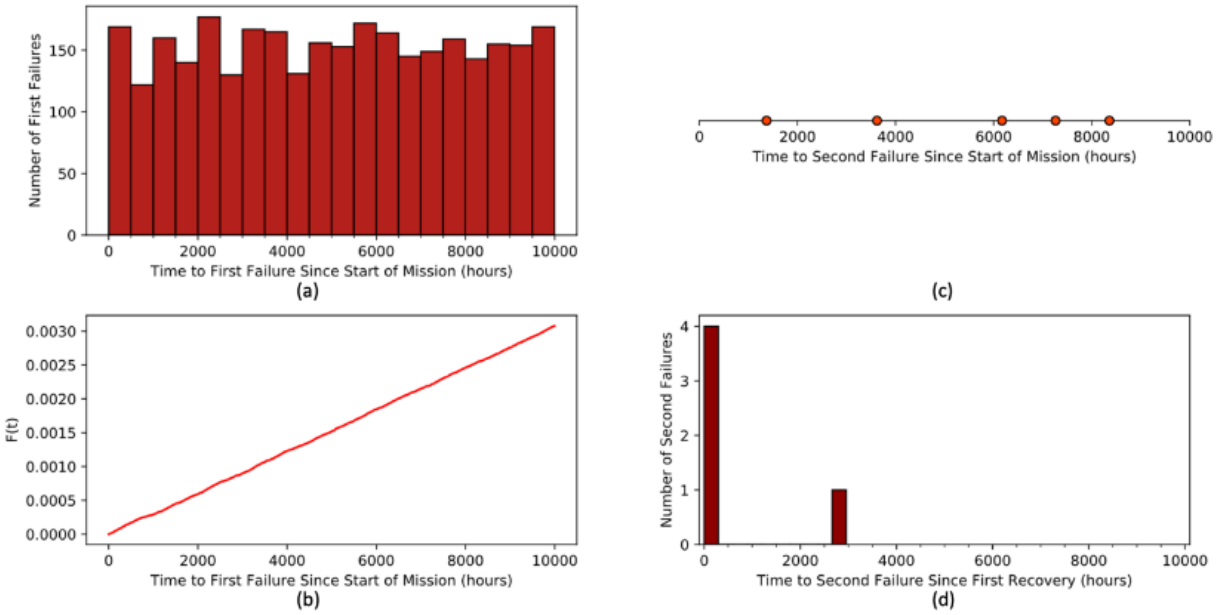

Fig. 3. Case Study 1 first and second failure distributions

IV. CASE STUDY 2: EPSS OF A NUCLEAR POWER PLANT

In this application, PyCATSHOO is used to retrieve the dependability characteristics of the EPSS of a NPP. The model comes from a previously established dependability assessment proposed as a benchmark in (Ref. 15).

In Section IV.A, the AC power supply system is described. The PyCATSHOO model and results are presented in Section IV.B.

IV.A. System Description

This section describes most of the system's characteristics, but more details (and graphical representations) can be found in (Ref. 15). Figure 4 displays part of the system.

The goal of the benchmark system is to supply electricity to a critical system such as the primary circuit pumps of the NPP. These critical components, not explicitly shown in Fig. 4, are fed by two busbars, *LHA* and *LHB*. System failure takes place when neither of these busbars is able to transmit electricity from a high voltage power source. In order of priority, there are five high voltage power sources: the reactor itself (*UNIT*), a transmission network (*GRID*), two diesel generators (*DGA* and *DGB*), and a gas turbine (*TAC*).

A *SUBSTATION* and two distribution lines (*GEV* and *LGR*) help carry power from the *GRID*. When the *GRID* or the connection between the *GRID* and the *UNIT* is lost, *UNIT* attempts to switch to "house load" mode where the reactor feeds just the systems within the plant.

In addition to the high voltage part of the system shown in Fig. 4 a low voltage part (illustrated in Ref. 15) supplies auxiliary power for high voltage circuit breaker operation. When the low voltage part of the system fails, certain high voltage circuit breakers (such as *CB_LHA1* and *CB_LGD2*) are no longer able to open or close. The low voltage power sources include backup batteries, each with a fixed, deterministic lifespan of one hour. AC/DC converters are also present in the low voltage lines.

All components can fail and be repaired, and short circuit propagation is again a factor in this system. Common cause failures (CCFs) are also present; extreme events (e.g., hurricanes or earthquakes) can take out both distribution lines and/or both backup diesel generators.

IV.B. PyCATSHOO Implementation and Results

System behavior is again examined within a mission time of 10^4 hours, and the system can again fail and be repaired multiple times. Failure rates in the form of λ and γ are also implemented as in Case Study 1.

In Section IV.B.1, the assumptions made in the model are described. Section IV.B.2 presents analysis and results from the dependability assessment.

IV.B.1. Model Assumptions

The PyCATSHOO model presented here uses similar assumptions as the model presented in (Ref. 15) and the

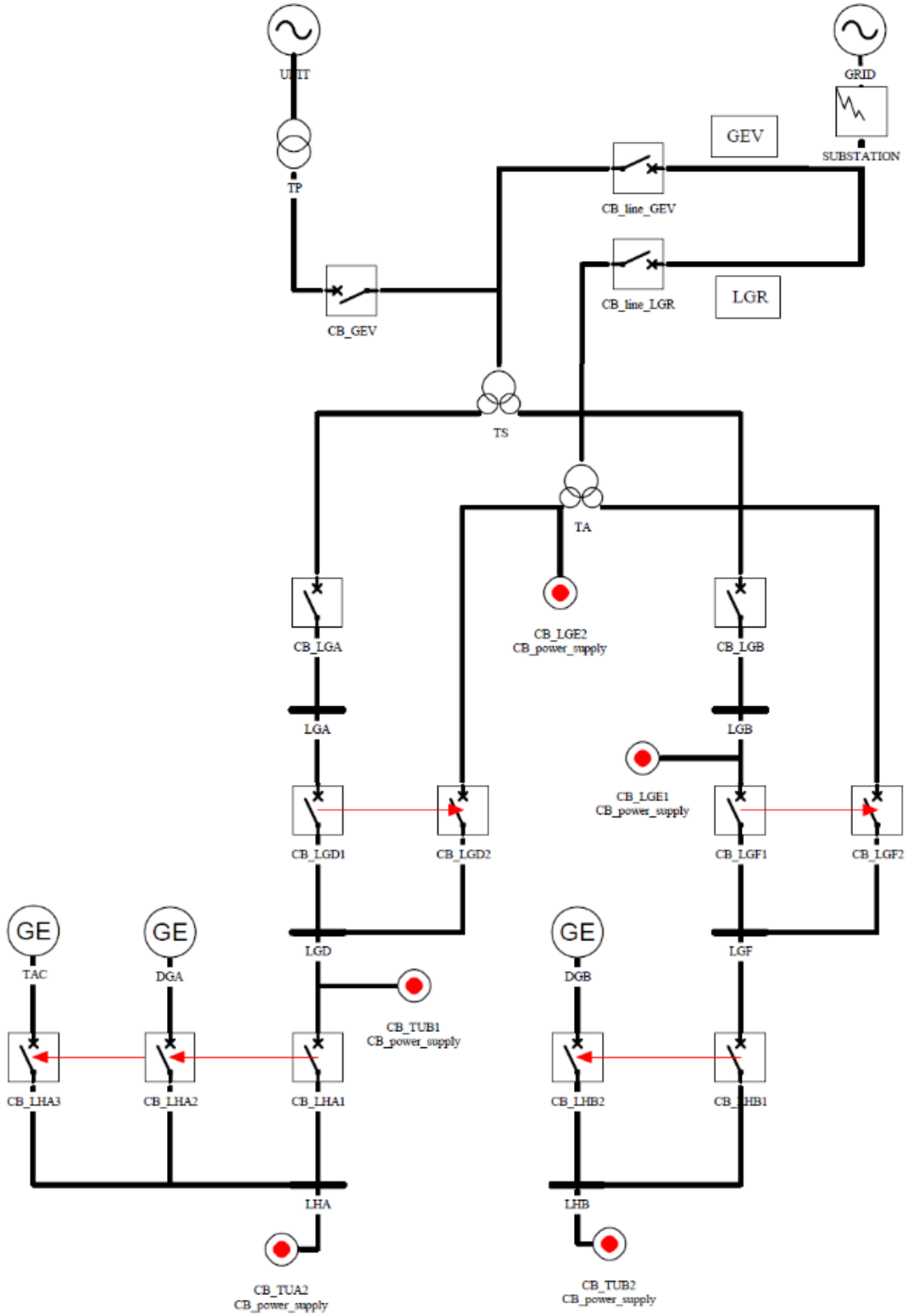

Fig. 4. The high voltage part of the system in Case Study 2. *UNIT* is the nuclear reactor. *TP*, *TS*, and *TA* are transformers. Component names starting with *CB_* indicate circuit breakers. *GEV* and *LGR* are distribution lines. *TAC* is a gas turbine, *DGA* and *DGB* are diesel generators, and *LHA* and *LHB* are the critical busbars. A red circle indicates a connection to the low voltage part of the system, with the name of the connecting low-voltage component listed underneath. Red arrows indicate priority/circuit breaker reconfiguration strategy for feeding *LHA* and *LHB*. (Ref. 15)

simpler model presented in Section III of this paper. Particularly:

1. *Each battery can supply power for exactly 1 hour before losing charge*. It is assumed that the batteries are replaced/return to full capacity after every use. PyCATSHOO's discrete stochastic automata are used to model all other system behavior.
2. *Possible failure of the busbars is modeled* for this case. For both LHA and LHB a failure rate of $2 \cdot 10^{-7}$ is assumed (Ref. 15).
3. *House load functioning cannot be repaired until connection to the GRID is restored.* This assumption reflects reality (Ref. 15).
4. *Not all circuit breaker openings/closings are subject to failure in the model* (though all circuit breakers are subject to experiencing a short circuit at any time while conveying electricity in the model). This assumption was not made in order to save computational time for MTTF, unreliability, and unavailability estimation. As stated previously in Section III.C.1, PyCATSHOO does not implement sequence exploration and thus the number of possible states is not a problem for computation. The issue instead is the modeling complexity that comes with the size and interconnectedness of the system: specifically, defining when each circuit breaker should open or close and the consequences of the failure to do so. If a short circuit occurs on a component, for example, and an adjacent circuit breaker fails to open, other circuit breakers may be prompted to open. These circuit breakers could fail to open, and the short circuit could continue to propagate in both the high and low voltage parts of the system until circuit breakers successfully cut it off. The number of consequences for circuit breakers failing to open/close becomes quite large when one considers that a short circuit could occur on almost any component in the system. Although this complexity can be handled by PyCATSHOO, the only on-demand circuit breaker failures allowed here are those represented in the BDMP diagrams in (Ref. 15) for illustration purposes. This approach is also justified as the BDMP diagrams depict all of the allowed transitions for the model presented in (Ref. 15), and the diagrams are the product of a trimming process whereby transitions with only minor contributions to system failure are omitted (Ref. 18). As stated in (Ref. 18), a slight increase in μ can account for the fact that not all on-demand circuit breaker failures are allowed in recovery.
5. *CCF are integrated as well within the model*: (i) diesel generators DGA and DGB (see Fig. 4) can experience CCF when one or both of them attempt to start (with failure rate $\gamma = 2 \cdot 10^{-4}$) or they can experience CCF when one or both of them are operating (with failure rate $\lambda = 5 \cdot 10^{-5}$ h^{-1}), and (ii) components GEV and LGR (each representing a distribution line) can also experience a CCF in operation with failure rate $\lambda = 10^{-6}$ h^{-1}.
6. *Every component is modelled individually.* As another way to reduce combinatorial explosion for sequence exploration, (Ref. 15) combines some components in the low voltage part of the system and approximates them using a single summation of their failure rates and a weighed sum of their repair rates.

IV.B.2. Dependability Assessment Analysis and Results

The resulting PyCATSHOO model consists of 24 Python classes. Each component in the system is defined as a different automaton that describes the component's behavior (Ref. 20). A total of 69 automata are modeled. The number of possible states (e.g., working, available, failed, etc.) in each automaton depends on the component considered. "Message boxes" are used in PyCATSHOO to allow communication between Python classes, ensuring each component can communicate with the rest of the system (Ref. 20). The classes in this model communicate through 310 message boxes.

Since the system has so much redundancy, 10 million simulations are required to achieve meaningful results – about 7 hours on a PC with a second generation Intel Core i7 processor.

Table III displays some of the most common failure sequences obtained from a run of 10 million missions. Table IV contains the definitions of the states listed in Table III in order of appearance. The results in Table III are similar to those obtained in (Ref. 15), but the relative number of occurrences among these most common sequences was not consistent between runs of 10 million missions. Due to the high number of possible failure sequences (Ref. 15) and the low failure rate of the system as a whole, more simulations would be required for Monte Carlo sampling to produce consistent results for the three most common sequences.

More results are displayed in Table V for the same run of 10 million missions. Table V also compares the results obtained with PyCATSHOO with (Ref. 21). For runs of 10 million missions, the unreliability produced by our model was observed to come within 8% of (Ref. 21).

Figure 5 displays failure and recovery time distributions for the same run of 10 million missions. Long times to failure in Figs. 5a and 5b (e.g., $1.6 \cdot 10^9$ hours) are a result of the system's high redundancy, while long times to recovery in Figs. 5c and 5d (e.g., 500 hours) are a result of the long repair times associated with some CCF s (such as *CCF.DGA_DGB_KOinFunction*, which has a mean repair time of 400 hours – Ref. 15). For the same run of 10 million missions, Figure 6 displays the distributions for time to first failure within a single mission.

TABLE III. Most common failure sequences (1 in sequence description indicates that the component has failed)

Number of Occurrences	**Average failure length (hours)**	**Fractional probability contribution to failure**	**Sequence description**
40	105.16	0.094	[[CCF.GEV_LGR_KO, 1], [Line_U.Uhouse_KOinfunction,1], [CCF.DGA_DGB_KOinFunction, 1], [Line_TAC.TAC_KOinFunction, 1]]
31	9.96	0.073	[[CCF.GEV_LGR_KO, 1], [Line_U.Uhouse_KOinfunction, 1], [Line_DGA.DG_KOlonginFunction, 1], [Line_TAC.TAC_KOinFunction, 1], [Line_DGB.DG_KOshortinFunction, 1]]
15	60.40	0.035	[[CCF.GEV_LGR_KO, 1], [Line_U.Uhouse_KOinfunction, 1], [Line_DGB.DG_KOlonginFunction, 1], [Line_DGA.DG_KOlonginFunction, 1], [Line_TAC.TAC_KOinFunction, 1]]

TABLE IV. Definitions of PyCATSHOO states

PyCATSHOO State	**Definition**
CCF.GEV_LGR_KO	Distribution line CCF
Line_U.Uhouse_KOinfunction	House load mode failure in operation
CCF.DGA_DGB_KOinFunction	Diesel generator CCF in operation
Line_TAC.TAC_KOinFunction	Gas turbine failure in operation
Line_DGA.DG_KOlonginFunction	Diesel Generator A failure in operation with long repair time
Line_DGB.DG_KOshortinFunction	Diesel Generator B failure in operation with short repair time
Line_U.Uhouse_RS	House load mode failure on demand

TABLE V. Results

	PyCATSHOO	**(Ref. 21)**
Total Number of Failures:	424	-
Total time between recovery and next failure (hours):	9.97e+10	-
MTTF (time/number of failures) (hours):	2.35e+08	-
Number of missions experiencing failure:	386	-
Unreliability (number of failed missions/total number of missions):	3.86e-05	3.84e-05
Total time system was unavailable (hours):	15658	-
Unavailability (time unavailable/total simulation time):	1.57e-07	-

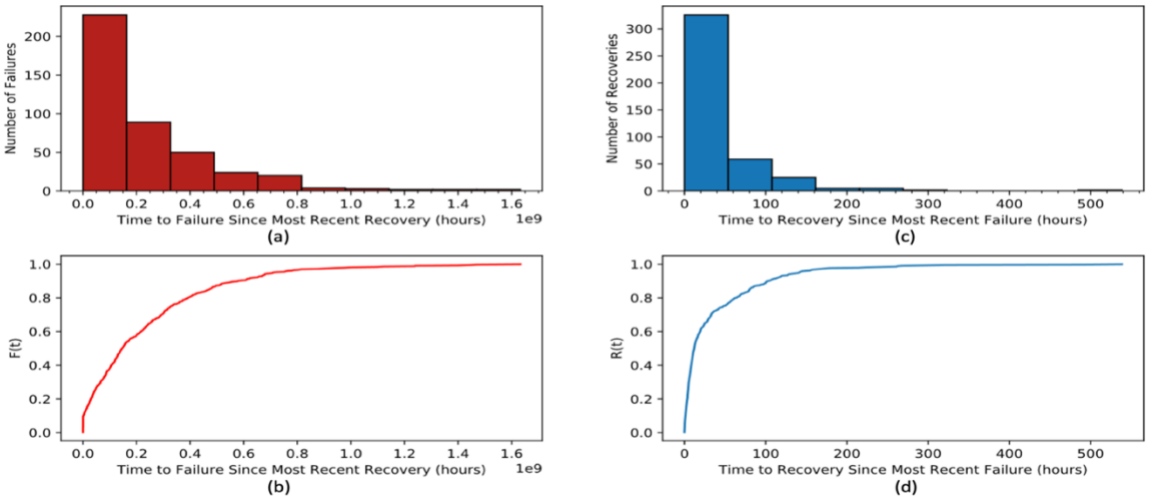

Fig. 5. Distributions of benchmark model failure and recovery times after a run of 10 million missions.

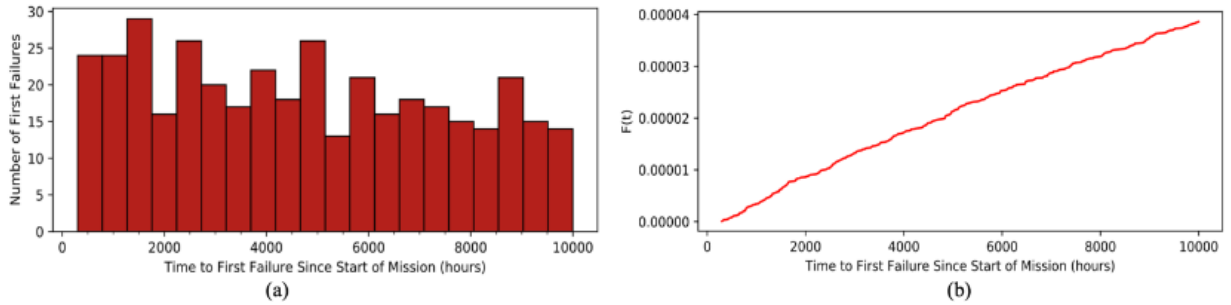

Fig. 6. Sample benchmark distribution for time to first failure in a single mission

TABLE VI. Most common sequences causing multiple failures in a single mission, starting at most recent failure (1 in sequence description indicates initiation of state, 0 indicates departure from state). See Table IV for the definitions of PyCATSHOO states

Number of Occurrences	**Average failure length (hours)**	**Sequence resulting in more than one failure in a mission**
12	59.74	[[Line_TAC.TAC_KOinFunction, 0], [counter.recovery], [Line_TAC.TAC_KOinFunction, 1]]
9	5.65	[[Line_DGB.DG_KOshortinFunction, 0], [counter.recovery], [Line_DGB.DG_KOshortinFunction, 1]]
6	10.02	[[Line_DGA.DG_KOshortinFunction, 0], [counter.recovery], [Line_DGA.DG_KOshortinFunction, 1]]

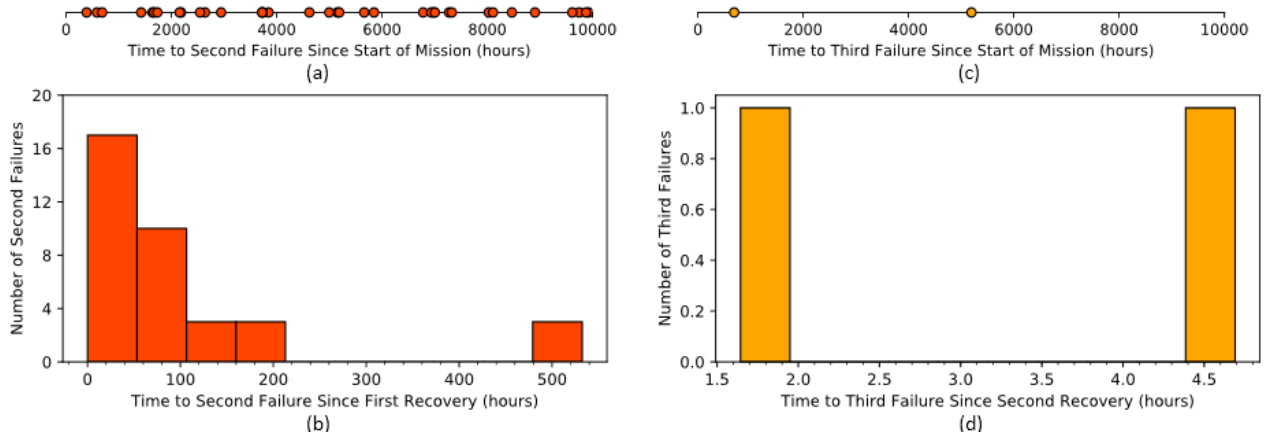

Fig. 7. Benchmark model single-mission second and third failures

In Fig. 6a, system failure appears to occur at a higher rate toward the start of the mission. This observation is due to the stochastic nature of the Monte Carlo scheme PyCATSHOO uses. Other runs of 10 million missions appeared to have failures more concentrated at the end or middle of the mission.

As seen in Table VI, the most common sequences leading to multiple failures in a single mission can be retrieved using PyCATSHOO. Sequences are displayed starting with the transition that leads to recovery from the most recent failure and ending with the transition that leads to the second or third mission failure. The first entry in Table VI shows that the most common sequence leading to a second or third failure was recovery of the system due to repair of the gas turbine followed by another failure of the gas turbine (12 total occurrences). This is expected because the gas turbine is the last high voltage power source used after all other high voltage power sources have failed.

Figure 7 shows that all second and third failures occurred within a month of recovery (under 700 hours), and most occurred within the first few days which indicates that multiple failures occurred when the system was only able to reach a partially recovered state.

V. CONCLUSION

PyCATSHOO is a useful tool under development for dependability assessment of systems containing both discrete stochastic and continuous deterministic events and complements capabilities of system level applicable DPRA

tools. This paper aims at illustrating an application of PyCATSHOO for an actual case study. A benchmark problem for the case of the EPSS of a NPP, as proposed in (Ref. 15), has been used for the illustration. The analysis of a simple electrical system has been performed as a first example to present the tool and the results. Then, PyCATSHOO has been applied to the case of an actual EPSS. PyCATSHOO results agree well with the results reported in (Ref. 18) and (Ref. 15) for the simple electrical system and the EPSS, respectively.

Reproducing the results of (Ref. 15) and (Ref. 18) has served as a proof of concept and validation process for the model. Future activities could include model exploration and improvement through the integration of more dynamic aspects, such as non-detection of certain failures and the system's response to the physical effects of failures (such as fire).

ACKNOWLEDGEMENT

The authors would like to thank Dr. Valentin Rychkov for his assistance in the use of PyCATSHOO and his support during the development of this case study.

REFERENCES

1. US NUCLEAR REGULATORY COMMISSION, "Probabilistic Risk Assessment," retrieved January 2019 from https://www.nrc.gov/about-nrc/regulatory/risk-informed/pra.html.
2. M. HIBTI and A. DUTFOY, "Dealing with system recoveries in event trees," *PSA 2011*, Wilmington.
3. EPRI, "Treatment of Time Interdependencies in Fault Tree Generated Cutset Results," 1009187, Palo Alto (2003).
4. T. ALDEMIR, "Quantifying Setpoint Drift Effects in the Failure Analysis of Process Control Systems," *Reliab. Engng & System Safety*, **24**, 33-50 (1989).
5. M. HASSAN and T. ALDEMIR, "A Data Base Oriented Dynamic Methodology for the Failure Analysis of Closed Loop Control Systems in Process Plants," *Reliab. Engng & System Safety*, **27**, 275-322 (1990).
6. J. KIRSCHENBAUM, P. BUCCI, M. STOVSKY, D. MANDELLI, T. ALDEMIR, M. YAU, S. GUARRO, E. EKICI, and S. A. ARNDT, "A Benchmark System for Comparing Reliability Modeling Approaches for Digital Instrumentation and Control Systems", *Nucl. Technol*., **165**, 53-95 (2009).
7. T. ALDEMIR, "A survey of dynamic methodologies for probabilistic safety assessment of nuclear power plants," *Annals of Nucl. Energy*, **52**, 113-124 (2013).
8. H. CHRAIBI, J. C. HOUBEDINE, and A. SIBLER, "PyCATSHOO: Toward a new platform dedicated to dynamic reliability assessments of hybrid systems," *PSAM 13*, Seoul (2016).
9. Y. CHANG, A. MOSLEH and R. BARI, "Dynamic PRA using ADS with RELAP5 code as its thermal hydraulic module," *PSAM 4*, New York, pp. 2468-2473, Springer-Verlag (1998).
10. E. HOFER, M. KLOOS, B. KRZYKACZ-HAUSMANN, J. PESCHKE, and M. SONNENKALB, "Dynamic Event Trees for Probabilistic Safety Analysis," *GRS*, Garsching (2004).
11. J. IZQUERDO, J. HORTAL, J. SANCHES-PEREA, and E. MELENDEZ, "Automatic Generation of Dynamic Event Trees: A Tool for Integrated Safety Assessment." Aldemir, T., Siu, N., Mosleh, A., Cacciabue, P.C., Goktepe, B.G. (Eds.), *Reliability and Safety Assessment of Dynamic Process Systems*, NATO ASI Series F, **120**, Springer-Verlag, Heidelberg, pp. 135-150 (1994).
12. C. RABITI, A. ALFONSI, J. COGLIATI, D. MANDELLI, and R. KINOSHITA, "Reactor Analysis and Virtual Control Environment (RAVEN)," *FY12 Report INL/EXT-12-27351*, Idaho National Laboratory, Idaho Falls (2012).
13. V. RYCHKOV and H. CHRAIBI, "Dynamic probabilistic risk assessment at a design stage for a sodium fast reactor," *FR17*, Yekaterinburg (2017).
14. A. GULER, T. ALDEMIR, and R. DENNING, "The Sojourn Time Approach for Modeling Aging in Passive Components," *Trans. Am. Nucl. Soc.*, **108**, 552-554 (2013)
15. M. BOUISSOU, "A Benchmark on Reliability of Complex Discrete Systems: Emergency Power Supply of a Nuclear Power Plant," *MARS 2017*, Uppsala, 2017, 244, 200-216, Electronic Proceedings in Theoretical Computer Science (2017).
16. A. VOLKANOVSKI, A. AVILA, and M. VEIRA, "Statistical Analysis of Loss of Offsite Power Events," *Science and Technol. of Nucl. Installations* (2016).
17. MCINTYRE, T.J. and N. SIU, "Electric power recovery at TMI-1, a simulation model," *Proceedings of the International ANS/ENS Topical Meeting on Thermal Reactor Safety*, San Diego (1986).
18. M. BOUISSOU and J. BON, "A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes," *Reliab. Engng & System Safety*, **82**, 149-163 (2003).
19. PyCATSHOO, Performance assessment of complex systems, pycatshoo.org
20. H. CHRAIBI, "Getting started with PyCATSHOO," retrieved January 2019 from http://pycatshoo.org.
21. M. BOUISSOU, O. BÄCKSTRÖM, RORY GAMBLE, PAVEL KRCAL, and W. WANG, "The I&AB Quantification Method for Large Dynamic Systems in Practice: Two Use Cases."