13th International Conference on Probabilistic Safety Assessment and Management (PSAM 13)
2~7 October, 2016 • Sheraton Grande Walkerhill • Seoul, Korea • www.psam13.org

PyCATSHOO:
Toward a new platform dedicated to dynamic reliability assessments of hybrid systems

Hassane CHRAIBI[1], Jean Christophe HOUDEBINE[2], Alain SIBLER[3]

[1]*EDF R&D/MRI, 7 Boulevard Gaspard Monge, PALAISEAU France, 91120, hassane.chraibi@edf.fr*
[2]*Aristè SARL, 2 rue du Marché 92160 Antony France, Jean-christophe.houdebine@ariste.fr*
[3]*EDF R&D/MRI, 7 Boulevard Gaspard Monge, PALAISEAU France, 91120, alain.sibler@edf.fr*

In the energy field as well as in other domains, situations where safety margins should be finely evaluated are becoming more and more frequent due to the evolution of economic and regulatory contexts. Taking into account the dynamics of physical phenomena in the studied systems is one of the levers to face these situations. A few years ago, EDF has developed a prototype of PyCATSHOO, a new tool able to use this lever. This prototype was tested in several test cases. The ability to deal with industrial-sized systems of the approach it is implementing has thus been validated. PyCATSHOO has since been redesigned to meet the requirements of industrial studies. So we are currently using PyCATSHOO in several safety studies and we plan its distribution as an open source tool.
PyCATSHOO provides a set of system modeling and 0D simulation tools. These tools cover both the domain of deterministic and continuous phenomena and the domain of stochastic discrete events. They also give convenient means to manage the interaction between these two kinds of phenomena.
Through the paradigm of the distributed stochastic hybrid automata, PyCATSHOO implements an approach based on the theoretical framework of the Piecewise Deterministic Markov processes.
This paper presents such an approach and the implementation of PyCATSHOO through a test case inspired from the spent fuel cooling system of a nuclear power plant. This test case mainly aims at illustrating the use of PyCATSHOO and at demonstrating its abilities by comparing a classic approach results with the results of a hybrid approach. By contrast to the first approach the latter takes into account the inertia of the physical phenomenon of the spent fuel cooling system. The study shows at which extent this difference affects the safety assessment results.
Finally, the paper also outlines the associated open source project which pulls together the efforts of several industrial and academic organizations in the purpose of developing an efficient and user-friendly industrial open source platform dedicated to the dynamic reliability assessment of hybrid systems.
The recent uses of PyCATSHOO have shown that it is an effective tool able to simulate both purely physical models and hybrid models where deterministic continuous phenomena and discrete stochastic behaviors are embedded. In the latter case, the carried out studies such as the one presented in this paper have shown that taking into account of the physical phenomena behind the occurrence of the assessed feared event can greatly refine the assessment of the safety margins.
Keywords: *Dynamic reliability, stochastic hybrid systems, probabilistic safety assessment, piecewise deterministic Markov processes, PyCATSHOO.*

I. INTRODUCTION

Classic methods of probabilistic safety assessments, widely used in nuclear domain, reveal their limits when fine dependability assessments are needed. Yet, such is the case owing to the evolution of economic and regulatory contexts that impose better evaluations of safety margins and detailed knowledge of the risks associated with the operated systems, especially in some transitory phases where the systems are the most vulnerable.

This trend affects equally several other domains such as hydropower generation, electrical power distribution and transportation, etc.

To meet these requirements, the system modeling must account for the dynamic of the process they implement. This leads to accounting not only for functional and dysfunctional dependencies, but also for mutual dependency with physical phenomena which may be governed by ordinary differential equations.

The areas of applicability of event and fault trees methods are far away from meeting these requirements. Several works propose various approaches aimed to overcome this lack. All These approaches exclude analytical methods and impose simulation as a quantification tool due to the complexity level when addressing the industrial-sized systems. But concerning

the modeling, they are based on various paradigms which ease one or another aspect of the issue addressed. In particular, when the hybrid character of the system is taken into account, much amongst the proposed approaches, as in [1] and [2], are based on discrete stochastic (respectively continuous deterministic) tools in which a continuous deterministic modeling (respectively stochastic discrete) mechanism is laboriously introduced. None of these tools natively implements the two modeling mechanisms nor the means of their cohabitation. This is exactly the goal that PyCATSHOO was designed to reach.

Hence, as a tool dedicated to carry out dependability assessments of hybrid systems, PyCATSHOO is obviously able to deal with both the stochastic discrete event systems and the continuous deterministic systems. But its major strength lies in the means it provides for facilitating the modeling of the interaction of these two worlds.

PyCATSHOO introduced the paradigm of distributed stochastic hybrid automata (DHSA). The principles of this paradigm have been already described [3]. PyCATSHOO was then in prototype stage whereas a nearly industrial version is currently in use at EDF/R&D. In the next section we will remind briefly the principles of this new version. To do so, we will rely on the presentation of the modeling approach of the Spent Fuel Pool Cooling System (SFPCS) we have used to illustrate PyCATSHOO capabilities. Beforehand, we will describe such a system and we will introduce the initiating and feared events retained for the quantitative analysis.

Finally, after the result discussion we will give recent news about our project which aims to release an open source version of PyCATSHOO.

II. SPENT FUEL POOL COOLING SYSTEM (SFPCS)

After its usage in the reactor of a nuclear power plant, the spent fuel is stored in water-filled bays for several years, in order to allow the fuel to cool down as its decay heat decreases. The water storage also provides effective radiation shielding. An active cooling system of the spent fuel pool is therefore required to remove the spent fuel's residual decay heat.

The present study aims to assess the reliability of a simplified system representative of the EPR SFPCS.

II.A. General design of the EPR SFPCS

The SFPCS consists in three trains: two identical main trains and a third fully independent train (or "stand-by train"). In the simplified design adopted for the present study (as shown in Figure 1), each train is composed by the following series-connected components: a valve (VA), a check-valve (CV), a pump (PU) and a heat-exchanger (HE).

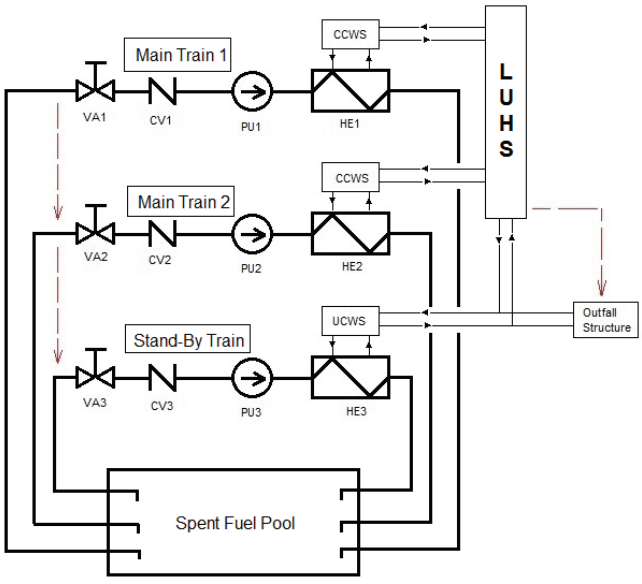

Fig. 1. Simplified general design of the SFPCS.

Depending on the power state of the plant (hence on the quantity of fuel stored in the spent fuel pool), one or two trains are required to cool the spent fuel pool. If only one train is needed, the first main train is initially operating, while the two remaining trains are in stand-by. In case of failure, the second main train starts, assuring the spent fuel pool cooling

continuity. The third train is started as a backup after the failure of one or two of the main trains (depending on how many trains are required in the specific power state of the plant).

The main trains are cooled by the Component Cooling Water System (CCWS). The third train (or "stand-by train") is cooled by the Ultimate Cooling Water System (UCWS). In case of failure of the water pumping station or "Loss of Ultimate Heat Sink" (LUHS), the third train's heat exchanger can be supplied by the Outfall Structure.

Each train's electrical supply is assured by a specific Switch Board Division (SBD). In case of Loss Of Off- side Power (LOOP), each SBD can be supplied by a dedicated Emergency Diesel Generator (EDG).

As shown above, the SFPCS and its electrical supply are highly redundant systems with multiple reconfiguration possibilities. The complexity of the dependencies between each line can only be apprehended by a dynamic model.

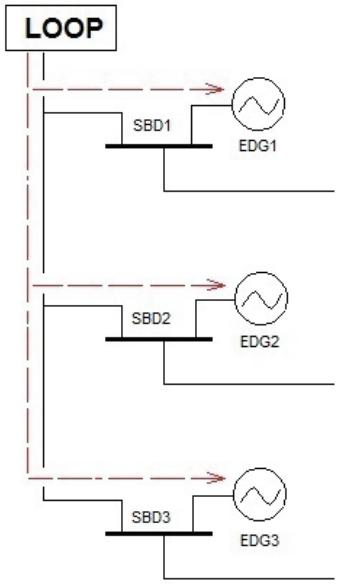

Fig. 2: Simplified design of the SFPCS's electrical supply

II.B. Initiating events

In the present study, we aim to calculate the reliability of the SFPCS under the component reliabilities and initiating event frequencies shown in table 1, for a 7846 hours mission time.

We also consider the effect of the following specific initiating events, potentially triggering a loss of the SFPCS:

- Non-repairable Loss of Ultimate Heat Sink (LUHS) for a duration of 100 hours;
- Non-repairable Loss Of Off-side Power (LOOP) for a duration of 192 hours.

II.C. Unwanted events

The reliability of the SFPCS is assessed in relation to the following unwanted events which may happen after the occurrence of an initiating event:

UE1. Pool water temperature rising above 80°C (safety margin from water boiling);
UE2. Pool water temperature rising up to 100°C (beginning of water boiling and pool level decrease);
UE3. Pool water supply falling below the minimum level (16m) before uncovering the fuel rods.

III PyCATSHOO modelling logic of the SFPCS

Figures 3 presents the layout of the SFPCS's design from the perspective of the PyCATSHOO modelling logic. Figure 4 presents the equivalent for its electrical supply.

For each line (main trains and stand-by train), CCWS and UCWS consist of an additional cross-flow heat-exchanger fed by two series-connected sets of components (a valve, a check-valve and a pump).

Bold arrows represent hydraulic and electrical flows, whereas thin arrows represent logical flows used for system monitoring.

For each component C_{ij}, $i \in \{1, 2, 3\}$ stands for the line number (main train 1, 2 or stand-by train) and $j \in \{1, 2, 3\}$ stands for the heat-exchanger cross-flow level within the line.

As reported in [3], the first step when modelling with PyCATSHOO consists in providing a generic modelling for each kind of the system components. This modelling is given by a Python class which comprises 4 groups of information. The first group gathers the state variables of the component and the incoming information from the outside. The second group declares several message boxes which organise the information exchanges between the component and the rest of the system. The third group gives the behaviour of the component as an ensemble of stochastic automata. And finally the fourth group provides the evolution rules of the component state variables.

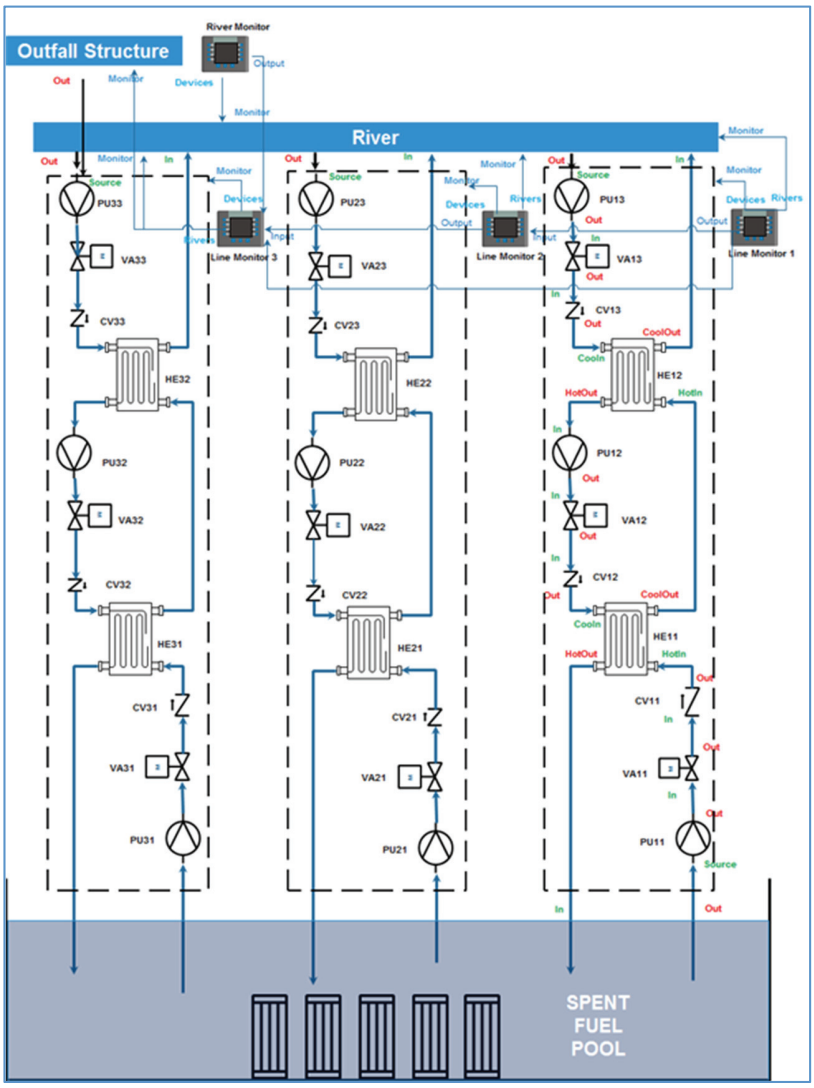

Fig. 3. PyCATSHOO modelling logic of the SFPCS[1].

[1] In the three trains, we assume that there is no pressure drop and that the pumps impose their following nominal flow rates $Q_{PU11} = 550m^3h^{-1}$, $Q_{PU12} = 520m^3h^{-1}$, $Q_{PU13} = 650m^3h^{-1}$,

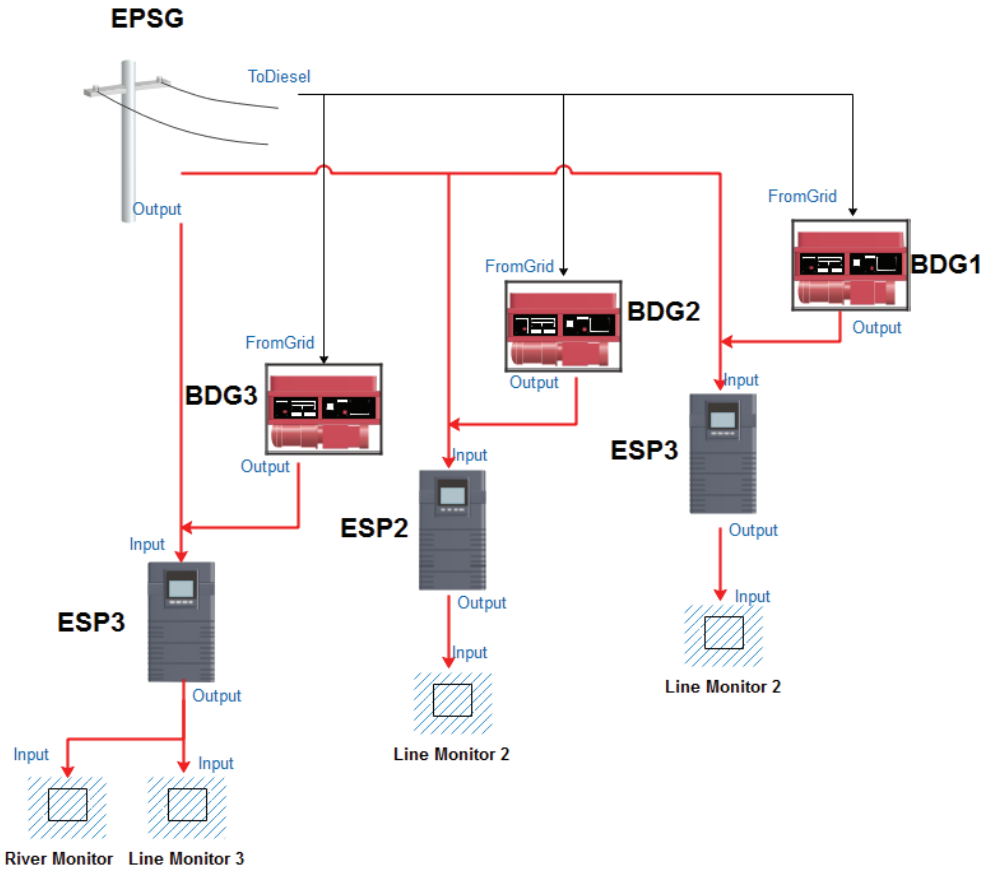

Fig. 4. PyCATSHOO modelling logic of the SFPCS's electrical supply.

To illustrate this let's take the example of the class which stands for an Exchanger.
Here we have an excerpt from the first group contents:

```
self.po_Tho  = self.addPortOut("Tho", Pyc.VarType.double, 0.)
self.po_Qho  = self.addPortOut("Qho", Pyc.VarType.double, 0.)
self.po_Tco  = self.addPortOut("Tco", Pyc.VarType.double, 0.)
self.po_Qco  = self.addPortOut("Qco", Pyc.VarType.double, 0.)

self.pi_Tci  = self.addPortIn ("Tci")
self.pi_Qci  = self.addPortIn ("Qci")
self.pi_Thi  = self.addPortIn ("Thi")
self.pi_Qhi  = self.addPortIn ("Qhi")
```

Where *Tco*, *Qco*, *Tho* and *Qho* are the temperature and the flow rate of hot and cold outflows while *Tci*, *Qci*, *Thi* and *Qhi* are the temperature and the flow rate of hot and cold inflows. The values of the last four variables are provided from the outside of the component while the values of the four first are provided by the exchanger itself.

The second group provides the message boxes which ensure the communication of the Exchanger with its connected components:

```
self.addMessageBox("CoolIn")
self.addMessageBoxImportPort("CoolIn", self.pi_Qci, "flow")
self.addMessageBoxImportPort("CoolIn", self.pi_Tci, "temperature")
```

These instructions create a message box called *"CoolIn"* dedicated to be connect to the component which brings cool water to the Exchanger. This message box receives two informations from the connected component: the value of the flow rate and the value of its temperature.

```
self.addMessageBox("CoolOut")
self.addMessageBoxExportPort("CoolOut", self.po_Qco, "flow")
self.addMessageBoxExportPort("CoolOut", self.po_Tco, "temperature")
```

Here we create another message box to be connected to the component which receives the outflow of cooling water. This outflow is also characterized by the flow rate and the temperature values of the cooling water.

The aim of the third group is to give an automata based model of the Exchanger behaviour. In the studied system, the exchanger has only one automaton with two states; a failure state and an operational state. The transitions between these two states follow exponential probability laws where *lambda* and *mu* are the transition rates. The creation of the automaton is given in the following excerpt.

```
self.automaton = self.addAutomaton("StateAutomaton")
self.st_ok     = self.addState    ("StateAutomaton", "OK", 0)
self.st_ko     = self.addState    ("StateAutomaton", "KO", 1)

trans = self.st_ok.addTransition("OK_to_KO")
trans.setDistLaw(Pyc.IDistLaw.newLaw(self, Pyc.TLawType.expo, self.po_lambda))
trans.addTarget(self.st_ko, Pyc.TransType.fault)

trans = self.st_ko.addTransition("KO_to_OK")
trans.setDistLaw(Pyc.IDistLaw.newLaw(self, Pyc.TLawType.expo, self.po_mu))
trans.addTarget(self.st_ok, Pyc.TransType.rep)
```

The parameters of a transition distribution law may be constant or may vary over time or according to other state variables of the component. In the last case we need to declare these parameters as the resolution results of an ordinary differential equation or of an explicit equation (see below). The transition must also be informed of this situation with the following instruction:

```
trans.setModifiable(Pyc.TransModifType.cont_modifiable)
```

This instruction means that the parameters of the transition are continuously updateable.

The fourth and final group is dedicated to the calculation of the rate and temperature of water outflows depending on the characteristics of hot and cold water inflows. Thanks to a class method, PyCATSHOO provides the means to achieve such a calculation using either explicit expressions or ordinary differential equations.

Our countercurrent heat exchanger only needs explicit expressions as shown thereafter.

$$T_{Ho} = T_{hi} - \frac{\left(1 - e^{\frac{KS}{\rho C}\left(\frac{1}{Q_{hi}} - \frac{1}{Q_{ci}}\right)}\right) * \left(T_{hi} - T_{ci}\right)}{Q_{hi}\left(\frac{1}{Q_{ci}} - \frac{e^{\frac{KS}{\rho C}\left(\frac{1}{Q_{hi}} - \frac{1}{Q_{ci}}\right)}}{Q_{hi}}\right)} \qquad (1)$$

$$T_{Co} = \frac{\left(1 - e^{\frac{KS}{\rho c}\left(\frac{1}{Q_{hi}} - \frac{1}{Q_{ci}}\right)}\right)\left(T_{hi} - T_{ci}\right)}{Q_{ci}\left(\frac{1}{Q_{ci}} - \frac{e^{\frac{KS}{\rho C}\left(\frac{1}{Q_{hi}} - \frac{1}{Q_{ci}}\right)}}{Q_{hi}}\right)} + T_{ci} \qquad (2)$$

$$\begin{aligned} Q_{Co} &= Q_{ci} \\ Q_{ho} &= Q_{hi} \end{aligned} \qquad (3)$$

Where C is the mass heat capacity[2], ρ the density[3], and KS a characteristic coefficient[4] of the countercurrent heat exchangers.

Prior to writing the method which calculates the temperatures and the flow rates of hot and cold outflows, we need to create a manager of the Piecewise Deterministic Makov Process in charge of monitoring the calculation of these continuous state variables and of synchronizing this calculation with stochastic discrete state changes of the system.

```
self.system().addPDMPManager("PDMP")
```

Then we have to specify which state variable are addressed by the PDMP

```
self.addODECalculatedPort("PDMP",self.po_Tco)
self.addODECalculatedPort("PDMP",self.po_Tho)
self.addODECalculatedPort("PDMP",self.po_Qco)
self.addODECalculatedPort("PDMP",self.po_Qho)
```

And finally we give the name of the method which performs the calculation of these state variables according to the equations 1, 2 & 3.

```
self.addODE     ("PDMP","pdmpMetho")
```

In the case of the Pool, the two main continuous state variables concerned by the PDMP Manager are the level of water and its temperature. These two variables are governed by an ordinary differential equation depending on the value of the temperature. The Pool energy balance Eq. (4&5) give an example of such equations.

$$\left.\begin{aligned} \frac{dT}{dt}\rho CSL &= P + \rho C\left(Q_{in}T_{in} - Q_{out}T\right) \\ \frac{dL}{dt} &= 0 \end{aligned}\right\} \; if \;\; P + \rho C\left(Q_{in}T_{in} - Q_{out}T\right) < 0 \;\; or \;\; T < 100 \qquad (4)$$

[2] $C = 4180\ J.Kg^{-1}.°K^{-1}$
[3] $\rho = 990\ \mathrm{Kg.m^{-3}}$
[4] $KS = 4.8e6\ W.°K^{-1}$

$$\left.\begin{aligned} -\frac{dL}{dt}\rho lS &= P+\rho C\left(Q_{in}T_{in}-Q_{out}T\right) \\ \frac{dT}{dt} &= 0 \end{aligned}\right\} if\ \ P+\rho C\left(Q_{in}T_{in}-Q_{out}T\right)>0\ and\ T=100 \tag{5}$$

$$\begin{aligned} T(0) &= 50°C \\ L(0) &= 19m \end{aligned} \tag{6}$$

Where:
S = 77 m^2 and L are the area of the swimming pool and its water level.
P is the residual power of the spent fuel.
l is the mass latent heat of vaporization[5].

To make the PDMP manager able to address such equations we only need to give the expression of the derivative of the temperature and the level as given in the following example which stands for the Eq. (4).

```
self.po_T.setDvdtODE( P + ρC(Q_in T_in − Q_out T_out)/ ρCSL )
self.po_L.setDvdtODE(0)
```

Once the generic modelling achieved, one can give specific information about the studied system. In addition to the dependability data, the architecture of the system has to be modelled. This is mainly done by creating the required instances of the previously created classes and by connecting the messages boxes of such instances according to the architecture given in Fig. 3 & 4.

```
…
self.pu12 = Pump     ("PU12", self)
self.he12 = Exchanger("HE12", self)
…
self.connectByName("HE12", "HotOut", "PU12", "In")
…
```

IV. SFPCS reliability assessment

The reliability of the SFPCS has been assessed in several previous studies as a part of the EPR Flamanville 3 plant's Probabilistic Safety Assessment. Paper [4] uses the BDMP framework [5] to tackle the problem of assessing a system's reliability with a highly dynamic behavior. This framework allows to define such dynamic models using a fault-tree like approach with interesting mathematical properties (in particular, the possibility to reduce combinatorial explosion inherent to Markov models). Nevertheless, this approach strictly relies on stochastic discrete events and does not manage their interaction with deterministic and continuous physical phenomena, as it can be taken into account by a hybrid tool such as PyCATSHOO.

[5] *l=2.257e6 J.°K^{-1}*

Table 1 shows the components' data used to perform the system's reliability calculations. For reasons of confidentiality, the chosen values are fictitious, although their orders of magnitude are representative of the components' comparative performances (active components like EDGs and pumps being less reliable than passive components like valves).

TABLE I. Reliability data

Component	λ (h^{-1})	γ	MTTR(h)
SBD	1,0 10-6	-	24
EDG	2,3 10-3	1,8 10-3	39
VA	8,7 10-8	1,1 10-4	10
CV	7,0 10-9	5,2 10-7	12
PU	1,1 10-5	2,7 10-5	18
HE	2,9 10-6	-	120
CCWS	8,4 10-6	-	72
UCWS	2,6 10-5	-	48
Outfall Structure	5,0 10-6	2,0 10-4	23
LOOP	4,0 10-6	-	24
LUHS	1,2 10-9	-	30

IV.A. Assumptions

The present study is only based on the cooling mission of the SFPCS. Accidental draining or water makeup systems are not included.

The system's reliability is quantified for the power state of the plant with minimal fuel quantity in the spent fuel pool and a residual power of 5,85 MW to evacuate. Hence, the mission requires the functioning of only one train among the three.

The present study takes into account the Common Cause Failures (CCF) of all active components (pumps and EDGs), for all identical components belonging to the same system and performing the same function, namely for components of main train 1 and 2 at the same cross-flow level. The stand-by train's components are of diversified conception, thus not subject to CCF. For a matter of simplification, we assume that a component failing for CCF reasons affects all the other components of its CCF group, which are then lost for sure. With the design shown on Figure 3 and Figure 4, all CCF groups taken into account contain two identical components, except for the EDG CCF group, which contains three identical components. All the CCF reasons of the affected components are considered to happen while the components are operating. The beta factor of the MGL letters model quantifying the CCF fraction of the global failure rates is assumed to be of 5% for the CCF groups with two components (pumps), and of 2% for the CCF group with three components (EDGs).

IV.B. Quantification and results

The purpose of the assessment carried out is to quantify the probability distribution of each of the three unwanted events listed above. But, in order to compare this study to the one based on a classic approach which solely takes into account discrete events, we have also quantified the probability distribution of the simultaneous loss of the three trains. For the rest of the article we will denote UE0 this unwanted event.

We have performed such quantifications in three different contexts:

C1. A 100 hours mission in case of non-repairable loss of ultimate heat sink (LUHS) in addition to the potential failures which frequencies are shown in table 1.

C2. A 192 hours mission in case of non-repairable loss of off-side power (LOOP) in addition to the potential failures which frequencies are shown in table 1.

C3. A 7846 hours mission subject to the potential failures which frequencies are shown in table 1.

The following table II gives, for each of the three contexts, the probabilities that the unwanted events occur before the end of the mission. The evolutions of such probability over time (from 0 to the instant of the end of the mission) are given by the three following figures (Figure 5, 6 & 7).

TABLE II. Probability of unwanted events

Context / Unwanted Event	C1(LUHS)	C2(LOOP)	C3
UE0 (Simultaneous loss of 3 trains)	1,16 10-2	1,41 10-2	5,09 10-3
UE1 (T > 80 °C)	6,27 10-3	7,78 10-3	2,00 10-3
UE2 (T=100°C)	4,31 10-3	5,26 10-3	8,90 10-4
UE3 (L < 16)	1,01 10-3	7,30 10-4	2,00 10-5

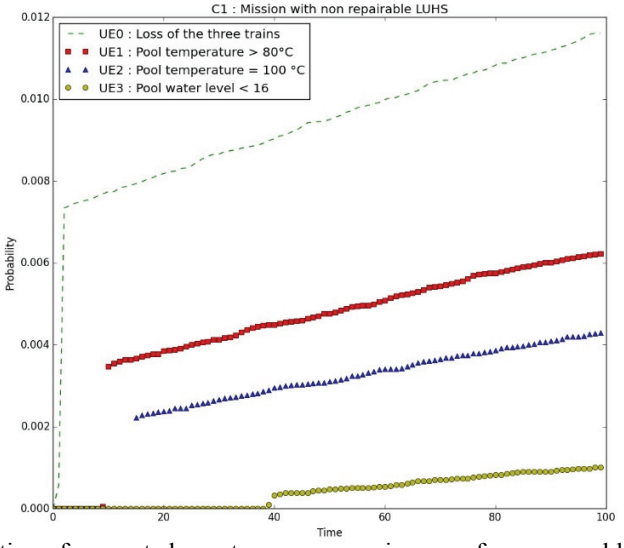

Fig. 5. Probability distribution of unwanted events occurrences in case of non-reparable loss of ultimate heat sink.

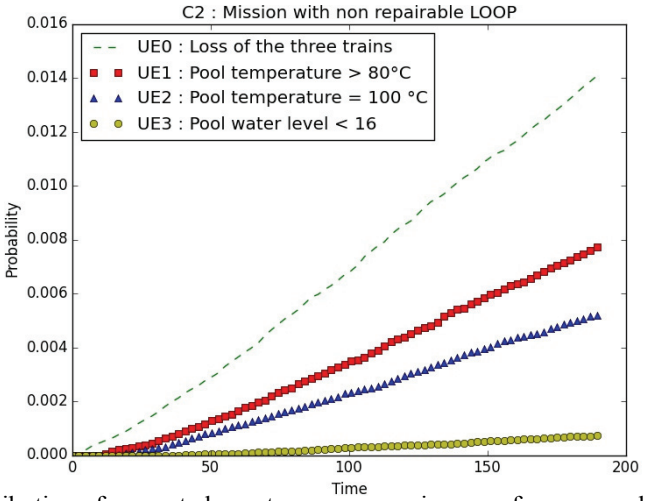

Fig. 6. Probability distribution of unwanted events occurrences in case of non-reparable loss of off-side-power.

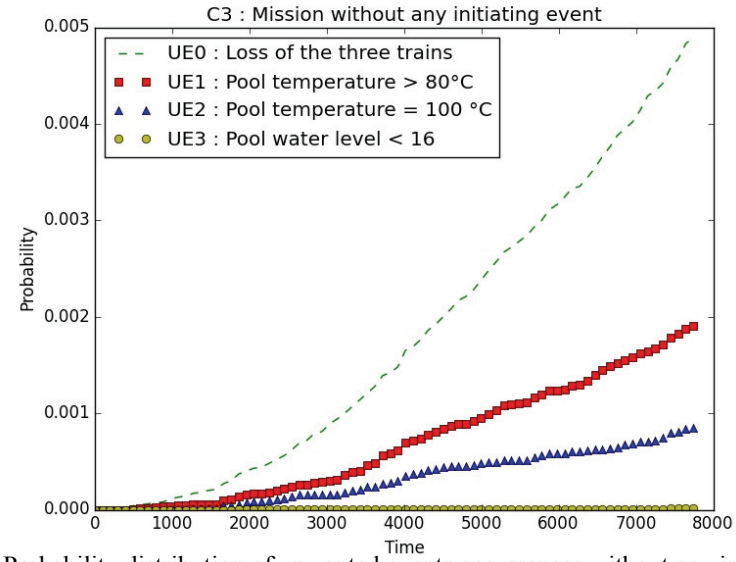

Fig. 7. Probability distribution of unwanted events occurrences without any initiating event.

Although one should not draw any conclusion with regard to the reliability of the studied system, reliability data being fictitious, the hierarchy in terms of reliability between the three contexts is however consistent with reality. Hence the C3 context values are the lowest even if the observation time is the largest, because in this context the LOOP and LUHS initiators are considered as repairable within the entire observation time. Likewise, the order of magnitude of the C1 and C2 are the same while the C2 observation time is larger. This indicates that the system is more reliable in case of a loss of off-side-power than it is in case of a loss of the ultimate heat sink. This is predictable since the loss of ultimate heat sink involves the loss of the two first trains while in case of a loss of off-side-power the three trains remain functional provided that the corresponding emergency diesel generators remain functional also.

But the most important findings is in the difference in terms of occurrence probabilities, between the four expressions of the unwanted events for the three contexts.

The occurrence probability of the three trains' loss, which corresponds to the criteria calculated by classic methods, is by far the highest. In particular this probability is two orders of magnitude higher than the occurrence probability of the fuel rod uncovering. This illustrates that the use of pure discrete methods, which prevent from using the right formulation of the unwanted event representing the failure of the actual system mission, may lead to very pessimistic conclusions, hence the advantage of the hybrid PyCATSHOO approach which avoid this situation.

V. Conclusion

The test case we presented in this article illustrates the ability of PyCATSHOO in modeling and assessing hybrid systems. This ability is required when one wants to provide a realistic modeling of the studied system mission and thus when one wants to use the right unwanted event formulation. The benefit from doing so is, as shown above, the lower level of conservatism of the assessment result comparatively to classic approach.

PyCATSHOO provides an innovative modeling approach, inspired from multi agent systems and based in distributed hybrid stochastic automata (DHSA). Indeed, as shown above, every system component is modeled as an object whose behavior, represented by one or several automata, is encapsulated within an object and thus hidden to rest of the system. Message boxes which preserve this encapsulation ensure the interaction between the multiple actors of the system. This approach makes it possible to easily reuse the component models when modelling and assessing different systems of the same nature.

When it comes to model the hybrid character of a system as the one presented in this article, PyCATSHOO provides various means to greatly ease the task of addressing the complexity induced by the interaction between continuous deterministic phenomena and the discrete behavior. This includes for instance the ability to formulate, in a declarative way, the differential

equations whose solving is taken in charge by PyCATSHOO in coordinated manner with the discrete events which occur in the system.

The modelling presented in this article uses the Python language interface of PyCATSHOO. PyCATSHOO provides a second interface based on C++ language. The use of the latter is optional but can greatly improve the computation efficiency. The use of these two languages makes it possible to benefit from a huge number of computation libraries released in these languages including tools like Matlab which provides a Python binding of its API (application programing interface).

Several works are currently on going in order to improve the modeling and computation capacities of PyCATSHOO.

A thesis work is currently carried out in order to develop new algorithms to speed up the Monte Carlo simulation engine of PyCATSHOO. On the other hand we are currently working on a plan of cooperation with several industrialists, academics and small or medium-sized enterprises (SME) with the objective to continue the co-development of PyCATSHOO. In particular, this co-development should focus on developing a modeling environment based on Model Driven Engendering concepts. The results of this co-development should be a new PSA plate-form dedicated to the modeling and assessment of hybrid complex systems. We are also considering to make this plate-form in open source.

REFERENCES

1 H. CHRAIBI, A. Leger, H. BARTHOMEUF, and C. NEISS. GASPART, A New Method and Tool for Reliability Assessments of Gated Spillway Systems. MMR2015 9th international Conference on Mathematical Methods in Reliability, Tokyo (Japan), 1-4 June 2015,

2. M. BOUISSOU, H. ELMQVIST, M. OTTER, A. BENVENISTE, Efficient Monte Carlo simulation of stochastic hybrid systems. Proceedings of the 10th International Modelica Conference. March 10-12, 2014, Lund, Sweden
3. H. CHRAIBI, *Dynamic reliability modeling and assessment with PyCATSHOO: Application to a test case*. PSAM 2013- International Topical Meeting on Probabilistic Safety Assessment and Management, Tokyo, 15-17 Avril 2013.
4. M. SORDELET and H. HIBTI, *Reliability of the EPR fuel pool cooling system using a dynamic approach*, ANS PSA 2011-International Topical Meeting on Probabilistic Safety Assessment and Analysis, Wilmington NC, 13-17 mars 2011.
5. M. BOUISSOU and J.L. BON, *A new formalism that combines advantages of fault-trees and Markov models: Boolean Logic Driven Markov Processes*, Reliability Engineering & System Safety, **82**, 2 (2003).