Industrial system example modeling for the assessment of maintenance strategies

Batteux Michel, Selma Khebbache, Seo Sin-Seok

Institut pour la Maîtrise des Risques
Sûreté de Fonctionnement - Management - Cindyniques

Modélisation d'un exemple de système industriel pour l'évaluation de stratégies de maintenances

Industrial system example modeling for the assessment of maintenance strategies

BATTEUX Michel
IRT SystemX
2, boulevard Thomas Gobert
91120 Palaiseau
michel.batteux@irt-systemx.fr

KHEBBACHE Selma
IRT SystemX
2, boulevard Thomas Gobert
91120 Palaiseau
selma.khebbache@irt-systemx.fr

SEO Sin-Seok
Safran Tech, DST / IRT System X
Rue des Jeunes Bois, Châteaufort
78114 Magny-Les-Hameaux
sin-seok.seo@safrangroup.com

Résumé — La gestion de la maintenance d'installations industrielles de production est un facteur important de compétitivité. Les travaux présentés dans cette publication consistent à modéliser un exemple de système industriel issu de la littérature : le système AGR ; pour l'évaluation de stratégies de maintenances. Nous utilisons deux outils considérant le système suivant deux niveaux : l'outil mp-sim qui se positionne au niveau des composants, et la plateforme OpenAltaRica qui se positionne au niveau système. Ces deux outils sont basés sur le cadre mathématique des systèmes à événements discrets stochastiques. Les indicateurs évalués sont calculés par simulation stochastique sur ces modèles.

Mots-clefs — Maintenance de systèmes industriels, modélisation, systèmes à événements discrets stochastiques

Abstract— Managing the maintenance of industrial plants is an important factor of competitiveness. Works presented within this publication show the modeling of an industrial system example coming from the literature: the AGR system; to evaluate maintenance strategies. Two tools are used with different level abstraction on the system. The mp-sim tool considers the component level. The OpenAltaRica platform considers the system level. These tools are based on the stochastic discrete event systems mathematical framework. Indicators to assess on these models are calculated by stochastic simulation.

Keywords — Maintenance of industrial systems, Modeling, stochastic discrete event systems

I. INTRODUCTION

The maintenance of industrial production systems is one of the major actual challenges, and the management of their maintenances is an important competitiveness factor. In fact, a suitable maintenance strategy should increase the system availability, and decreases costs due to interventions. There are two different kinds of maintenance policies: corrective maintenances repairing the system after the occurrences of failures, preventive maintenances to maintain the system before the occurrences of failures. Combining different kinds of maintenance policies on components of a system can thus be a good solution. Nevertheless, it has to be finely analyzed, so to search the optimal maintenance strategies on the system, according to specified criteria (e.g. availability, cost, etc.).

In this publication, we show the modeling and assessment of maintenance strategies and time availability of an industrial example coming from the literature: the AGR system [3]. We focus on two levels: the component level and the system level. For both, the components are modeled with their degradation and failure processes. At the component level we study different kinds of maintenance strategies. At the system level, we compose the components so to get the time of availability of the system according to occurrences of failures of components. We only consider the reliability diagram point of view of the system, and does not represent functional reconfigurations. This overall modeling is based on the mathematical framework of stochastic discrete event systems (see [4] and [5] for an overall introduction). Models are designed with the mp-sim tool for the component level, and the OpenAltaRica platform for the system level.

These works are carried out within the research project MPO (for "Maintenance Prévisionnelle et Optimisation" in French), realized at IRT SystemX[1] with different industrial and academic partners. This project deals with the optimization of maintenance strategies for production systems.

The remainder of this publication is organized as follow. Section II makes a brief description of this AGR system example. Section III presents the mathematical framework of stochastic discrete event systems, and how it is implemented within the mp-sim tool and the OpenAltaRica platform. Section IV presents the modeling of the AGR system with mp-sim and OpenAltaRica, and is followed by the experiments in section V. Section VI discusses about works combining the two tools with an optimization algorithm to get (one of) the

[1] www.irt-systemx.fr/projets/mpo

best maintenance strategy for the system. Finally, section VII concludes this publication.

II. The AGR System example

The AGR system example is taken from [3]. It is a lubrication system of a turbo-pump of a 900MW nuclear power plant. It is composed of several heterogeneous components: pumps, valves, filters, a heat exchanger and sensors. These components are organized in several redundant lines. Each component can fail according to several failure modes, which occurrences are led by different degradation mechanisms: for instance, the oxidation of the electrical contacts of a pump, leading to a shutdown or a failure to start. Figure 1 presents the architecture diagram of this AGR system

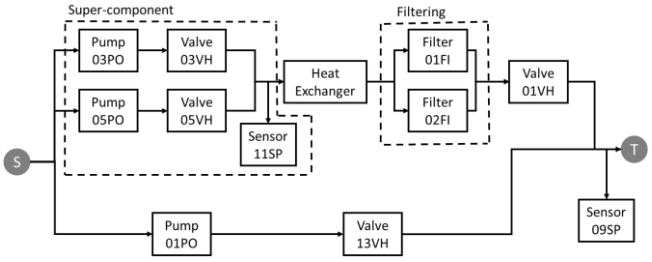

Figure 1: Architecture diagram of the AGR system

Document [3] also presents the degradations and failures of the components. For instance, a filter can have two degradations, leading to two failure types: a clogging degradation leading to a hydraulic lost and a hole degradation leading to a chemical lost. The degradations have different levels, from level 0 meaning no degradation, to level 2 meaning high degraded. The transitions towards higher degradation levels follow a stochastic law (Weibull or exponential). Furthermore, the changes between the working of a component and its failures, according to the degradation and its level, follow a stochastic law.

Different kinds of maintenance strategies are defined: corrective maintenance, planned maintenance and condition-based maintenance according to inspection and tests performed on components.

For the works of this publication, our interest is on the modeling of such degradations and failures processes at the component and system levels, and to get the time of availability of the system according to these degradations and failures.

III. The Stochastic Discrete Event Systems Framework

Discrete event systems (DES) are a mathematical framework to describe the behavior of systems. The state of the system can be in a finite or infinite set and transitions between states are performed according to occurrences of events. These occurrences of events are associated to delays, which can be deterministic or stochastic. In case of stochastic delays, we name a stochastic discrete event systems (SDES)

Figure 2 is a graphical representation of a SDES of a component which can be in four states: STANDBY, WORKING, FAILED and MAINTENANCE. Arrows linking states represent transitions and their labels describe the events. Plain arrows represent stochastic transitions whereas dotted ones represent deterministic transitions.

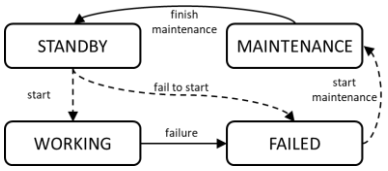

Figure 2: Graphical representation of a SDES

For both tools presented after (mp-sim and OpenAltaRica), we consider stochastic simulation of discrete event systems. Stochastic simulation is a versatile tool to compute performance indicators of discrete event systems ([9]). It consists in drawing at random a sample of executions of the model, to observe a number of quantities during these executions, and to make statistics on these observations.

A. The mp-sim implementation

Mp-sim (Maintenance Policy SIMulation) is a Python based maintenance policy simulations tool. It is intended for component level or black-box system simulations. Mp-sim uses SimPy ([1]-[2]) as its core simulation framework. SimPy is an open source library for discrete event simulations and it is process-based. The processes in SimPy are defined by Python generator functions and can be used to model active components like customers, vehicles or agents. SimPy also provides various types of shared resources to model limited capacity congestion points like servers, checkout counters and tunnels.

We identified modeling components for maintenance policy simulations from existing research efforts (Figure 3). Among them, the two right side components, a degradation model and a maintenance policy, are the most important and core items for maintenance policy simulations. The left side components, which are usage, performance metric, cost, time (delay), maintenance quality, maintenance type, system configuration, data quality and optimization, are also important but their necessity is dependent on specific requirements of actual use cases.

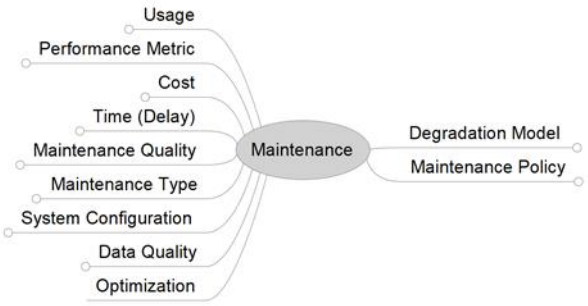

Figure 3: Identified Modeling Components in mp-sim

In the mp-sim degradation model, there are two main features: time to failure (TTF or lifetime) and degradation level at a given time or a given usage. The TTF is the time difference between a failure time and its introduction time of a maintenance object while degradation levels are an evolution of health states during a course of its start and failure. As you can easily infer, the two features affect each other and are closely linked together; the short TTF implies a steeper degradation level curve and vice versa.

After identifying the two essential features for a degradation model, we can think of two different approaches: top-down and bottom-up. The top-down approach starts from

a TTF distribution that reflects statistical characteristics of a population of maintenance objects. Well-known probability distributions for modeling lifetimes include Weibull, Lognormal, Exponential and Bathtub curve distributions. Once a TTF is drawn from a given probability distribution, we can create a degradation curve that is adjusted to fail at the drawn TTF of the instance. The degradation curve can take different shapes such as linear, stepwise, exponential and logarithm.

The bottom-up approach takes the opposite process; it starts from modeling a degradation curve then TTF is decided when a degradation level reaches a threshold. Two aspects to consider for modeling a degradation curve are increment and unexpected failure. Increment models a difference between two consecutive usages or time units of a maintenance object. It is obtained from a given probability distribution each time the maintenance object is used, then it is cumulatively added to the last degradation level. By repeating this process, we can obtain a degradation curve that contains evolution of degradation levels over time. When modeling the increment, we can also consider correlations with previous degradation levels or different usages. Depending on characteristics of an increment distribution, a degradation curve can be monotonically increase or fluctuate. More specifically, if an increment distribution contains only positive values then it is monotone, otherwise it can fluctuate. Another aspect to consider for modeling a degradation curve is an unexpected failure that means a sudden break down at once rather than gradual degradations. It models random failures in real world due to unexpected events such as a bird crash to an airplane engine. We can model an unexpected failure with a random process that has a certain occurrence probability and apply it each time when a maintenance object is used. We can consider correlations with previous degradation levels or usages for the unexpected failure modeling too.

Figure 4 shows class diagram of degradation models implemented in mp-sim. At the top, *BaseDegradModel* provides interfaces for all the degradation models. Then *BaseTDDegradModel* and *BaseBUDegradModel* inherit the *BaseDegradModel* and implement common functionalities for top-down and bottom-up degradation models respectively.

Mp-sim supports three top-down degradation models:

- *TopDownDegradModel*: Used for simple top-down degradation model that has one TTF distribution
- *GeneralTDDegradModel*: Used for a more generalized and advanced top-down degradation model that has one or more degradation mechanisms. Each degradation mechanism has one or more degradation levels, and each degradation level has one or more TTF distributions.
- *MixedTDDegradModel*: Used for mixed usage cases where two or more different usages were assumed in a maintenance target object. Final TTF is obtained by applying weighted average.

In addition to the top-down degradation models, mp-sim supports two bottom-up degradation models:

- *BottomUpDegradModel*: Used for simple bottom-up degradation model that has one increment distribution and one unexpected failure probability
- *MixedBUDegradModel*: Used for mixed usage cases where two or more different usages were assumed in a maintenance target object. Each increment is obtained from multiple increment distributions.

From the perspective of maintenance policy simulations with discrete events, we can think of three fundamental events: failure, repair and inspection. A failure occurs when simulation time reaches a TTF, which is obtained from a degradation model, without proper repair actions. A repair event occurs periodically (scheduled) or after an unexpected failure depending on a maintenance policy. An inspection occurs periodically by a predefined schedule and it checks a degradation level of a maintenance object. According to the inspection result, we decide whether to do a maintenance or not. These discrete events essentially introduce their corresponding expenses (costs) and delays (downtime) to the system.

A maintenance policy in mp-sim is all about how to allocate the repair and inspection events. For that, mp-sim provides a generic and extendible way to allocate the two events for implementing new maintenance policies without much effort. More specifically, mp-sim provides the abstract class *BaseMO* that deals with the three main events behind the scene and subclasses of *BaseMO* only need to specify where to place repair and/or inspection events (Figure 5).

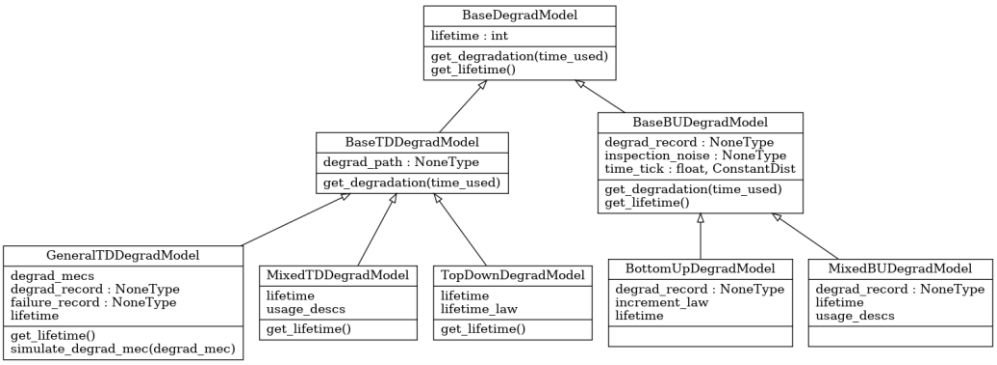

Figure 4: Class Diagram of Degradation Models in mp-sim

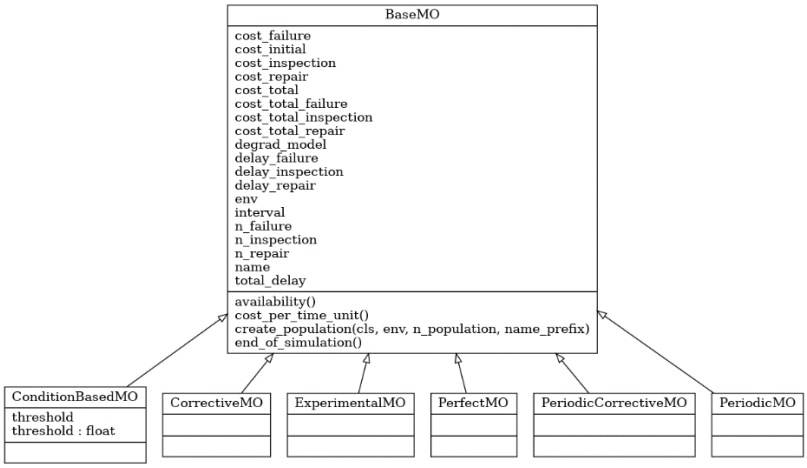

Figure 5: Class Diagram of Maintenance Policies in mp-sim

As of now, mp-sim provides six different maintenance policy implementations:

- *CorrectiveMO*: Let it fail, then repair.
- *PerfectMO*: Repair right before a failure.
- *PeriodicMO*: Repair periodically and wait until the next scheduled repair even if there was a failure.
- *PeriodicCorrectiveMO*: Repair periodically and repair right after a failure like the corrective one.
- *ConditionBasedMO*: Inspect periodically and repair when a degradation level is over a threshold.
- *ExperimentalMO*: Template for implementing new maintenance policies.

B. *The AltaRica 3.0 implementation*

AltaRica 3.0 is a high level and stochastic event-based modeling language, initially dedicated to the assessment of complex critical systems ([6]). The language is based on the mathematical framework GTS (for Guarded Transition Systems [7]) to describe the behavior of the system under study. The execution of an AltaRica 3.0 model is based of stochastic discrete event system ([8]).

A versatile set of assessment tools are developed for AltaRica 3.0, composing the OpenAltaRica platform:

- An interactive simulator making possible to play 'what-if' scenarios and to validate models ([10]).
- A compiler of AltaRica 3.0 models into fault trees ([11]).
- A generator of critical sequences.
- Finally, a stochastic simulator ([12]).

For this publication, we focus on this stochastic simulator assessment tool. For AltaRica 3.0 models, the random dimension comes with delays of events. For stochastic delays a random choice is made for the date to fire the transition, whereas for deterministic delays a random choice is made when several transitions can be fired at the same date. Then statistics are made on indicators based on observers defined in the model. Indicators are values calculated at specific times of the executions from the successive values.

Previous works have considered the modeling and assessment of different maintenance strategies with AltaRica 3.0: for instance [13] or [14] for corrective and planned maintenances, or [15] for combination of different kinds.

IV. Modeling of the AGR System

The AGR system example, presented section II was modeled by both tools: mp-sim and OpenAltaRica. These two modeling complement each other: with mp-sim we focus on the component point of view and with OpenAltaRica we focus on the system point of view.

A. *mp-sim modeling*

The AGR system can be modeled by using *GeneralTDDegradModel* in mp-sim. In fact, this generalized top-down degradation model is inspired by the AGR system. As noted in Chapter III.A, this generalized top-down degradation model can have one or more degradation mechanisms. Each degradation mechanism can have one or more degradation levels, and finally, each degradation level can have one or more TTF distributions. Figure 6 shows a code example that realizes the "Clogging" degradation mechanism of 01FI and 02FI filters in the AGR system. Other degradation mechanisms can also be realized following the similar way.

A complete degradation model for the filters consists two degradation mechanisms, namely "Clogging" and "Hole degradation". A TTF is drawn from the two degradation mechanisms and histogram of the drawn TTFs of the filters is shown in Figure 7.

```
clogging_degradation = DegradMechanism(
    name='Clogging Degradation',
    degrad_levels=[
        DegradLevel(
            name=0,
            transition_law=WeibullDist(shape=4, scale=200),
            failure_modes=[
                FailureMode('Hydraulic Lost',
                            ExponentialDist(rate=10**-5)),
            ],
        ),
        DegradLevel(
            name=1,
            transition_law=WeibullDist(shape=2, scale=100),
            failure_modes=[
                FailureMode('Hydraulic Lost',
                            ExponentialDist(rate=0.004)),
            ],
        ),
        DegradLevel(
            name=2,
            transition_law=None,
            failure_modes=[
                FailureMode('Hydraulic Lost',
                            ExponentialDist(rate=0.04)),
            ],
        )
    ],
)
```

Figure 6: Code Example of a Filter Degradation Mechanism

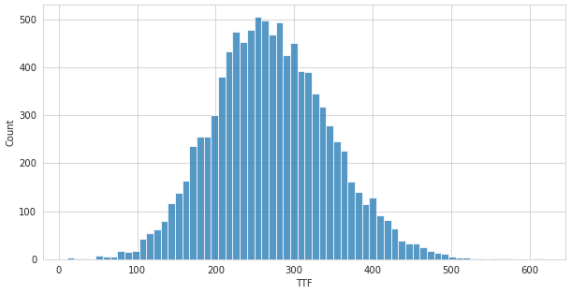

Figure 7: TTF Histogram of the filter

B. AltaRica 3.0 modeling

With the AltaRica 3.0 modeling language, one can design by the top-down or the bottom-up approaches, and even combines both. For the AGR system, the model is decomposed on three main parts: the definition of classes representing components of the system, the definition of the main block representing the system, and other elements to define domains of values.

For all the components, we have defined a modeling pattern to model behaviors of degradations and failures. For this publication, and due to lack of place, we only present it instantiated on the class filter. Other components are defined with the same patterns and according to their respective degradations and failures.

1) AltaRica 3.0 generic domains

Figure 8 presents the AltaRica 3.0 domains of values used within the different parts of the model. The first domain named `agr::OMode` is used to define if a component is on operation or on maintenance. The second domain `agr::FState` is used to define functional state of components. We only consider the AGR system with a dysfunctional point of view, thus functional states of components are limited to one of the two working/failed pattern. Finally, the third domain `agr::CDegradation` is used to define the state level of a degradation.

```
domain agr::OMode {OPERATION, MAINTENANCE}
domain agr::FState {WORKING, FAILED}
domain agr::CDegradation {D0, D1, D2}
```

Figure 8: AltaRica 3.0 generic domains of values

2) AltaRica 3.0 classes for a filter

As previously introduced in section II, a filter can have two degradations, leading to two failures: a clogging degradation leading to a hydraulic lost, and a hole degradation leading to a chemical lost.

Figure 9 introduces the two classes defining a filter. The first class `agr::BehavioralFilter`, the most important, defines a filter from its internal behavior point of view: three state variables `vsMode`, `vsFState` and `vsIsDegraded`, a maintenance part describes after and the two blocks, also describes after to define the degradations and failures. The second class `agr::Filter` inherits the first class and adds its external behavior with Boolean flow variables and the assertion defining how the output flow is updated according to the input flow and the internal state variables. Boolean variables are considered because we only focus on dysfunctional point of view of the system, meaning, when the component is failed, it sends the value false.

```
class agr::BehavioralFilter
  agr::OMode vsMode (init = OPERATION);
  agr::FState vsFState (init = WORKING);
  Boolean vsIsDegraded (init = false);
  // Maintenance part
  // ...
  // Degradations and failures
  block CloggingDegradation
    // ...
  end
  block HoleDegradation
    // ...
  end
end

class agr::Filter
  extends agr::BehavioralFilter;
  Boolean vfIn, vfOut (reset = false);
  assertion
    vfOut := vfIn and vsMode == OPERATION
                  and vsFState == WORKING;
end
```

Figure 9: AltaRica 3.0 classes for a filter

The AltaRica 3.0 part of the clogging degradation of a filter is presented in Figure 10. It first introduces a dedicated state variable vsD representing the level of degradation and transitions to change from level 0 to level 1 and from level 1 to level 2. Weibull delays of events labelling these transitions are taken from [3]. Then it defines the occurrences of the failure '*hydraulic lost*' according to the level of the degradation by the three transitions.

The same degradation pattern, instantiated in Figure 10, is used to define the second degradation of a filter: a hole degradation leading to a chemical lost. Furthermore, it is also used to define all degradation and failures for components.

Maintenance of a filter are abstract in the sense that they are started if the component is degraded or failed. We assume a deterministic delay of 0.5 time units (expressed in day) to start a maintenance, and a deterministic delay parametrized with the value 0.5 time units, done in [3], to realize the maintenance. After the maintenance, the functional state of the component is set to `WORKING`.

```
block CloggingDegradation
  agr::CDegradation vsD (init = D0);
  event evD0D1 (delay = Weibull(4.0,200.0));
  event evD1D2 (delay = Weibull(2.0,100.0));
  transition
    evD0D1: owner.vsOMode == OPERATION and
            owner.vsFState == WORKING and
            vsD == D0
      -> {vsD := D1;
          owner.vsIsDegraded := true;}
    evD1D2: owner.vsOMode == OPERATION and
            owner.vsFState == WORKING and
            vsD == D1
      -> {vsD := D2;
          owner.vsIsDegraded := true;}
  event evHydraulicLostD0
            (delay = exponential(1.0e-4));
  event evHydraulicLostD1
            (delay = exponential(0.004));
  event evHydraulicLostD2
            (delay = exponential(0.04));
  transition
    evHydraulicLostD0: vsD == D0 and
              owner.vsOpMode == OPERATION and
              owner.vsFState == WORKING
      -> owner.vsFState := FAILED;
    evHydraulicLostD1: vsD == D1 and
              owner.vsOpMode == OPERATION and
              owner.vsFState == WORKING
      -> owner.vsFState := FAILED;
    evHydraulicLostD2: vsD == D2 and
              owner.vsOpMode == OPERATION and
              owner.vsFState == WORKING
      -> owner.vsFState := FAILED;
end
```

Figure 10: AltaRica 3.0 part for the Clogging of a filter

Other classes defining other components of the AGR system are implemented with the same pattern as the one of the class filter.

3) AltaRica 3.0 main block of the AGR system

The main block of the AltaRica 3.0 is defined with the same structure as the AGR system depicted in Figure 1. It means that this main block is composed of two sub-parts defined as (sub-)block: the main line with the super component, the heat exchanger, the filters, and the valve; and the spare line with the pump and the valve. Figure 11 shows this main structure.

The Boolean observer `oOut` is used to compute indicators by assessment tools. For our purpose, it is used to compute statistics, by stochastic simulation, on the time availability of the system.

V. Experiments

A. Component-level simulations with mp-sim

Figure 12 shows availability, in term of mean "sojourn-time" in days, of a 01FI or 02FI filter simulated using mp-sim. In the simulations, we consider a mission time of 1 825 days (representing 5 years), 10-day of delay when there is a failure and 0.5-day of delay for maintenance interventions. The corrective maintenance strategy resulted in around 1 760 available days while periodic maintenance strategy resulted in around 1 813 available days with 95-day of maintenance interval. The best maintenance strategy was the periodic + corrective one that resulted in around 1 815 available days with 115-day of maintenance interval.

```
block AGRSystem
  Boolean vfIn, vfOut (reset = false);
  block MainLine
    Boolean vfIn, vfOut (reset = false);
    block SuperComponent
      // ...
    end
    agr::HeatExchanger he;
    block Filtering
      // ...
    end
    agr::Valve V;
    assertion
      SuperComponent.vfIn := vfIn;
      HE.vfIn := SuperComponent.vfOut;
      Filtering.vfIn := HE.vfOut;
      V.vfIn := Filtering.vfOut;
      vfOut := V.vfOut;
  end
  block SpareLine
    Boolean vfIn, vfOut (reset = false);
    agr::Pump P;
    agr::Valve V;
    assertion
      P.vfIn := vfIn;
      V.vfIn := P.vfOut;
      vfOut := V.vfOut;
  end
  agr::Sensor S
  assertion
    vfIn := true;
    MainLine.vfIn := vfIn;
    SpareLine.vfIn := vfIn;
    S.vfIn := MainLine.vfOut or SpareLine.vfOut;
    vfOut := S.vfOut and
             (MainLine.vfOut or SpareLine.vfOut);

  observer Boolean oOut = vfOut;
end
```

Figure 11: AltaRica 3.0 main block of the AGR system

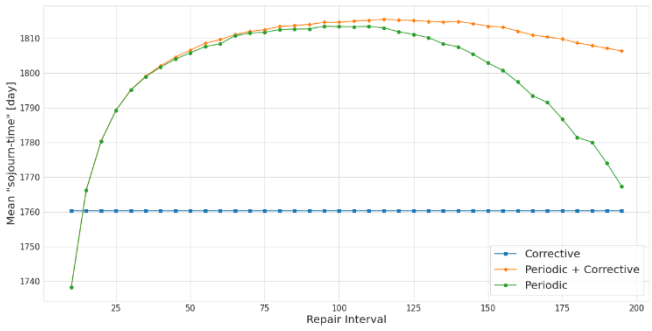

Figure 12: Filter Availability Obtained from mp-sim Simulations

B. System-level simulations with AltaRica 3.0

For stochastic simulation of the AltaRica 3.0 model presented in sub-section IV.B of the AGR system, we consider a time mission of 1 825 days (representing 5 years). We have defined a 'sojourn-time' indicator on the observer `oOut` to the value true. It represents the total duration of the different time intervals with the observer `oOut` takes the value true, during the time interval [0.0, 1 825.0]. Statistics of this indicator provides the mean time of availability of the system.

Table 1 presents results obtained by the stochastic simulator. 100 000 executions of the model have been realized and statistics are calculated on these executions. The time to realize these executions is 3 minutes on a laptop, which is quite small according to the size of the model and the number of fired transitions presented in the first part of the table. The second part of the table indicates statistics of the indicator

'sojourn-time' on the observer 'oOut': the mean, the standard deviation (named SD) and the 95% lower and upper bound (named lb and ub) confidence range. These statistics are provided at different dates of execution: each year.

Table 1: Results of stochastic simulations of the AltaRica 3.0 model

Fired transitions		Mean Min Max	3296.22 2943 4093	
Indicator '`sojourn-time`' on Observer '`oOut`'				
Dates	Mean	SD	95% lb	95% ub
365.0	324.79	10.47	324.73	324.86
730.0	638.93	19.28	638.81	639.05
1 095.0	941.35	27.47	941.18	941.52
1 460.0	1 238.36	36.31	1 238.14	1 238.59
1 825.0	1 531.39	46.32	1 531.1	1 531.67

We do not realize a comparison of our results with results from [3]. On the one hand, the system level model and the calculated indicators are not the same. In fact, the AltaRica 3.0 model does not consider some functional features of the system, for instance the control to switch from one line to the other according to failures. Furthermore, we do not consider the number of required repairers to realized maintenance, and also the cost of the maintenance. On the other hand, our concern is focus on the relation between simulation and optimization tools, see the next section VI. Thus, the use of this example may continue with this next works.

VI. Discussion

In this publication, we consider two approaches to assess maintenance strategies and time availability of the AGR system: the mp-sim tool with a component point of view and the OpenAltaRica platform with a system point of view. Our objective is to get first information about the maintenance strategies at the component level, so to parametrize the model of the system level with first values. In fact, our works are integrated in a more global project to combine simulation tools with optimization algorithms, so to get the best, or one of the best, maintenance strategies of a system. First works considering small (or toy) examples were already presented: see [13] and [14]. The idea is to define planned maintenance of components thanks to a genetic algorithm.

VII. Conclusion

In this publication, we presented how to model and assess maintenance and the time availability of an industrial example coming from the literature: the AGR system. After a brief presentation of this system, we have considered modeling with two point of views, and two different tools. The mp-sim tool considers the component level point view and produces information about the best maintenance strategies for each component of the AGR system. The OpenAltaRica platform considers the system level point of view and produces information about the time availability of the system according to failures of components.

The models, at both component and system levels, consider the stochastic discrete event systems mathematical framework. For each component we defined degradations and failures as states, and the changes between levels of degradation and from the working to failed state as transitions. On the one hand, it provides a modeling pattern to model and assess such behaviors of degradations leading to failures. On the other hand, stochastic discrete event systems are the accurate level of abstraction, not only because behaviors of degradations and failures are clearly defined in terms of occurrences of events, but also because assessment algorithms (e.g. stochastic simulation) are efficient.

Even if some future works were indicated within the discussion part (VI), in can also be interesting to made some sensitive analyses on the different parameters of the model. In fact, such analyses can conclude, on the one hand, on the relations between the variations of some parameters and the results of the model, in terms of the time availability, or other criteria. On the other hand on the relation between some of these parameters.

Acknowledgment

These works are carried out within the research project MPO, realized at the SystemX Technological Research Institute. They have been supported by the French government under the ''France 2030" program.

[1] N. Matloff. "Introduction to discrete-event simulation and the simpy language". Davis, CA. Dept of Computer Science. 2. 2008.

[2] TeamSimPy, "SimPy overview," [Online]. Available: https://simpy.readthedocs.io/en/latest/index.html. [Accessed 08 06 2022].

[3] V. Zille. "Modélisation et évaluation des stratégies de maintenance complexes sur des systèmes multi-composants". PhD thesis, Université de Technologie de Troyes, Institut Charles Delaunay, 2009

[4] Armin Zimmermann. "Stochastic Discrete Event Systems". Springer, Berlin, Heidelberg, Germany, 2008.

[5] Christos G. Cassandras and Stéphane Lafortune. "Introduction to Discrete Event Systems". Springer, New-York, NY, USA, 2008.

[6] M. Batteux, T. Prosvirnova and A. Rauzy. "Altarica 3.0 in 10 modeling patterns". Intenational Journal of Critical Computer-Based Systems 9(1–2), 133–165. 2019

[7] A. Rauzy. "Guarded transition systems: a new states/events formalism for reliability studies". Journal of Risk and Reliability 222(4), 495–505. 2008

[8] M. Batteux, T. Prosvirnova and A. Rauzy. "Abstract executions of stochastic discrete event systems". Intenational Journal of Critical Computer-Based Systems 10-3, 202-226. 2022

[9] E. Zio. "The Monte Carlo Simulation Method for System Reliability and Risk Analysis". Springer Series in Reliability Engineering. London, England: Springer London. 2013

[10] M. Batteux, T. Prosvirnova and A. Rauzy. "Enhancement of the AltaRica 3.0 stepwise simulator by introducing an abstract notion of time", in Proceedings of European Safety and Reliability Conference Safe Societies in a Changing World (ESREL 2018), Trondheim, Norway, June, 2018.

[11] T. Prosvirnova and A. Rauzy. "Automated generation of minimal cutsets from AltaRica 3.0 models", International Journal of Critical Computer-Based Systems, Vol. 6, No. 1, pp.50–79. 2015

[12] B. Aupetit, M. Batteux, A. Rauzy & J.-M. Roussel "Improving performances of the AltaRica 3.0 stochastic simulator". In Proceedings of European Safety and Reliability Conference (ESREL). Zurich, Switzerland. September 2015.

[13] S. Khebbache & M. Batteux. "Simulation based optimization for maintenance strategies using Altarica 3.0". In Proceedings of the 31st European Conference on Operational Research (EURO 2021). Athens, Greece. July 2021.

[14] M. Batteux, S. Khebbache & Y. Souami. "Simulation of complex system based on optimization methods for Maintenance scheduling". In Proceedings of the 31st European Safety and Reliability Conference (ESREL). Angers, France. September 2021.

[15] M. Batteux, T. Prosvirnova and A. Rauzy. "Modélisation de combinaisons de maintenances en AltaRica 3.0". In Actes du congrès Lambda-Mu 22 (actes électroniques), IMdR. Le Havre, France. October, 2020