International Conference
on Industrial Engineering and Systems Management
IESM 2009
May 13-15
MONTRÉAL - CANADA

Optimization of preventive maintenance through a combined maintenance-production simulation model [⋆]

Olivier ROUX [a], David DUVIVIER [b], Gauthier QUESNEL [c], Eric RAMAT [b,c]

[a] *LSM and Fucam, Chaussée de Binche, 151 B-7000 Mons - Belgium*

[b] *LIL - ULCO, 50, rue Ferdinand Buisson, F-62228 Calais Cedex - France*

[c] *INRA, UR875 Biométrie et Intelligence Artificielle, F-31326 Castanet-Tolosan - France*

Abstract

Maintenance problems are crucial aspect of nowadays industrial problems. However, the quest of the efficient periodicity of maintenance for all components of a system is far from an easy task to accomplish when considering all the antagonistic criteria of the maintenance and production views of a production system. Thus, the objective is to simultaneously ensure a low frequency of failures by an efficient periodic preventive maintenance and minimize the unavailability of the system due to preventive maintenance. This implies a minimum impact on the production. In this paper, several tools are combined to collaborate in order to optimize multi-component preventive maintenance problems. The structure of the maintenance-production system is modeled thanks to a framework inspired by our previous research projects. The dynamic aspects are modelled by a combination of timed petri-nets and PDEVS models and implemented in our VLE simulator. The parameters of the resulting simulation model are optimized via a Nelder-Mead (Simplex) Method.

Key words: Metaheuristics, simulation, decision-making, multi-modeling, Petri-nets, maintenance

1 Introduction

The present economical context requires from companies that they practice an optimal exploitation of their production tools. In this purpose, every decision maker is asked to assure a maximum availability of these production tools at minimal cost. The optimization consists in determining the best "parameters combination" which provides the best values of the technical and economical criteria. However, in most cases, it appears to be very difficult to use analytical approaches without formulating restrictive hypothesis. In order to evaluate these performance criteria, simulation is the best adapted solution. In this paper, we suggest an approach integrating optimization and simulation. This approach consists in generating solutions more and more efficient with an optimization tool and to evaluate them via a simulation model until a halt criterion is satisfied. This integration is illustrated in figure 1.

[⋆] This paper was not presented at any other revue. Corresponding author M. O. ROUX.
Email addresses: olivier.roux@fucam.ac.be (Olivier ROUX), david.duvivier@lil.univ-littoral.fr (David DUVIVIER), gauthier.quesnel@toulouse.inra.fr (Gauthier QUESNEL), ramat@lil.univ-littoral.fr (Eric RAMAT).

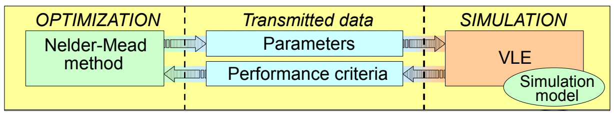


Fig. 1. Optimization and simulation integration

Our work aims to provide a framework to facilitate the optimization of production and maintenance through simulation. This paper focuses on the simulation aspect. We want to develop a generic modeling tool for simulation, easy to understand by decision makers. The objective is to facilitate the creation of simulation models by the use of constructs (elementary components).

The remainder of this paper is organized as follows. The second section presents the maintenance problem; the third section introduces the simulation paradigms, formalisms and tools that constitute the bases of our framework; the fourth section depicts our modeling component; the fifth section describes an application of our optimization-simulation hybrid. Finally several conclusions and perspectives are given.

2 Maintenance strategies

A maintenance strategy is defined as a decision rule which establishes the sequel of maintenance actions. Each maintenance action allows one to maintain or restore the system in a specified state by using the appropriate resources. Cost and duration are incurred to execute each maintenance action. Many papers dealing with preventive maintenance and replacement strategies have been published in the last two decades [13]. We consider for this paper one basic replacement policy (Bloc Replacement Policy BRP).

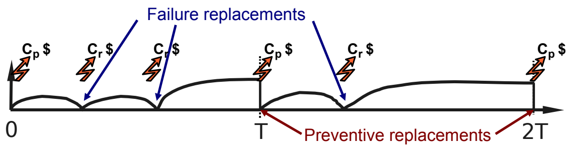


Fig. 2. Availability of the system subject to the Block Replacement Policy

Barlow and Proshan [1] consider also the BRP where the replacements are undertaken at KT with $K = 1, 2, 3, \ldots$ and T a fixed time, or at failure (see figure 2). Only new items are used to perform replacement. C_p and C_r are respectively the preventive and corrective replacement costs.

3 Simulation

In this section, we present the VLE simulator and the underlying paradigm and formalism. This simulator relies on strong concepts and intrinsically provides multi-modeling capabilities. This perfectly matches the objective of the simulation and modeling tool that we are currently implementing. This is also largely facilitated by the available extensions such as Petri-nets [9].

3.1 DEVS, VLE and Petri-nets

Nowadays, it is recognized that multimodeling is a powerful concept for the modeling and simulation of large complex systems. At the end of 80's, P.A. Fishwick and B.P. Zeigler [5] introduced the multimodeling basis concepts. One can define multimodels as large models which are composed of different types of models (i.e. different paradigms or formalisms) [4]. Concepts like refinement and hierarchical composition are basis of multimodeling. The first describes the decomposition of one model into several other ones in order to refine the behavior of the composed model. The last defines the opposite process: it is say models aggregation. In this context, a major issue is how to deal with the coupling of heterogeneous models. Several works deal with the coupling of heterogeneous models. For a review of concepts and techniques, see the book of B.P. Zeigler *et. al.* [15]. With DEVS, Discrete Event System Specification [14], B.P. Zeigler has provided formal basis for the construction of coupled model in a network or graph manner. In this section, we focus on the Discrete Event System Specification formalisms (DEVS) and their associated extensions, in particular Petri-net.

3.1.1 Discrete Event Simulation

Our works take place in the Modeling and Simulation (M&S) theory defined by B. P. Zeigler [14]. M&S theory tends to be as general as possible. It addresses major issues of computer sciences. From artificial intelligence to model design and distributed simulations, M&S theory aims to develop a common framework (formal and operational) for the specification of dynamical systems. Many theoretical basis and formal extensions to DEVS were carried out, therefore, we advise the second edition of B. P. Zeigler's book [15] to have an overall picture of these works. DEVS defines an atomic model as a set of input and output ports and a set of state transition functions: $M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$
where: X is the set of input values and Y is the set of output values
S is the set of sequential states
$\delta_{int} : S \rightarrow S$ is the internal transition function
$\delta_{ext} : Q \times X \rightarrow S$ is the external transition function
$\lambda : S \rightarrow Y$ is the output function
$ta : S \rightarrow \mathbb{R}_0^+$ is the time advance function
$Q = \{(s, e) | s \in S, 0 = e = ta(s)\}$
where Q is the set of total states and e is the time elapsed since last transition

Every atomic model can be coupled with one or several other atomic models to build a coupled model. This operation can be repeated to form a hierarchy of coupled models. The set of atomic and coupled models and their connections are named the structure of the model.

$$DEVS_N = \langle X, Y, D, EIC, EOC, IC \rangle$$

Where X and Y are input and output ports, D the set of models, EIC, EOC and IC, respectively, input, output and internal connections. Moreover, DEVS is an operational formalism, i.e. it provides the algorithms (the abstract simulators) that implement the formal models. So, since the beginning of the DEVS works, several DEVS simulators are implemented. The next section develop the VLE simulator, based on the DEVS formalism.

3.1.2 VLE

VLE [11][12] (*Virtual Laboratory Environment*) is a software and an API (*Application Programming Interface*) which supports multimodeling and simulation by implementing the DEVS abstract simulator. VLE is oriented toward the integration of heterogeneous formalisms. Furthermore, VLE is able to integrate specific models developed in most popular programming languages into one single multimodel.

VLE implements the Dynamic Structure Discrete Event formalism (DSDE) [3] which provides the abstract simulators for Parallel DEVS (PDEVS) [15] for the parallelization of atomic models and Dynamic Structure DEVS (DSDEVS) [2] for the M&S of systems where drastic changes of structures and behaviors can occurred over time. DSDE abstract simulators gives to VLE the ability to simulate distributed models and to load and/or delete atomic and coupled models at runtime. VLE proposes several simulators for particular formalisms. In addition to DSDE, for instance, cellular automata, ordinary differential equations, difference equation, Petri-net and so on.

This framework can be use to model, simulate, analysis and visualize dynamics of complex systems. His main features are: multi-modelling abilities (coupling heterogeneous models), a general formal basis for modelling dynamic systems and an associated operational semantic, a modular and hierarchical representation of the structure of coupled models with associated coupling and coordination algorithms, coupling of pre-existing models, distributed simulations, a component based development for the acceptance of new visualization tools, storage formats and experimental frame design tools, and free and open source software.

3.1.3 Petri-nets

The DEVS approach is applied to the Petri-nets [9]. Works dealing with the mapping of Petri-nets into DEVS exists (see [6] for instance). In these works, places and transitions are specified as atomic models and the network as coupled model. In our approach, we wrap a Petri-net simulator in a DEVS simulator.

In the following, we give the definition of a Petri-net: $PN = (P, T, F, W, m_0)$
Where: $P = \{p_1, \ldots, p_p\}$ is a set of places, where: $p = card(P)$,
$T = \{t_1, \ldots, t_t\}$ is a set of transitions, where: $t = card(T)$,
$F \subseteq (P \times T) \cup (T \times P)$,
W is a weight function: $F \rightarrow \mathbb{N}$, where $W = \{\ldots, ((p_i, t_j), w^-_{ij}), \ldots ((t_j, p_i), w^+_{ji})\}$
m_0 is the initial marking: $P \rightarrow \mathbb{N}$, where $m_0 = \{(p_1, m^0_1) \ldots (p_p, m^0_p)\}$

In the context of this paper, this definition is not sufficient, therefore timed transition and inibitor arc are added to specification and simulator. The definition of a petri-net is now completed by wrapping function denoted χ. This function is divided into two parts: χ_T is the input transition-wrapping function and χ_P is the output place-wrapping function. The n-uple of χ_P defines the effect of the marking of an output place p of the model. When a token arrives to an output place, an output event is build and send to associated output port. The function χ_T is related to input events which have an effect on transition of petri-net. If an event arrives on a port belonging to χ_T, the transition t_j is fired.

$$\chi_T = \{(p, t_j)\} \text{ where: } p \in X \text{ and } t_j \in T$$
$$\chi_P = \{(p, p_i)\} \text{ where: } p \in Y \text{ and } p_i \in P$$

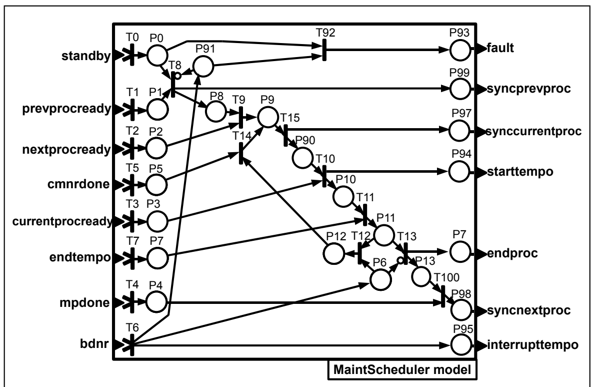
In this approach, informations related to the events are ignored. When a token or a set of tokens arrives in a place then one can send an event. This event is marked. The internal dynamic of a Petri-net is controlled by the marking and the structure of the network. While one or more transitions are enabled then the marking evolves. This evolution is independent of the concept of time, except in the case of timed transition.

4 The MSP component

One of the objectives of this research is to create a versatile "component" that can be assembled at will, in various ways so as to study the interaction of scheduling and maintenance in production systems. Our work aims at presenting the coupling of "as easily understandable as possible" simple-models rather than one monolithic dedicated model. Another objective of our research is to study the integration of decision and optimization tools with simulation models applied to various fields. The optimization tool is a Nelder-Mead (Simplex) method but other methods might also be used. In the presented results, the VLE simulator is used to implement our so-called "`MSP` component". This component comprises several parameters (number of jobs to be performed, statistical distributions, durations of maintenances...) which are used as degrees of freedom to be tuned by the decision markers and/or the optimization method. Several `MSP` components are assembled to simulate production systems. Each component contains two Petri-nets presented in figure 3. The first one is the scheduler, named `MaintScheduler` (see figure 3(a)). It is responsible for local scheduling rules as well as internal and external synchronization aspects. The presented results are based on a basic scheduler but it might be remplaced by a more realistic scheduler. The second Petri-net-based model, named `MaintSchedProd`, is the actual operating part of the component, in charge of coupling production and maintenance aspects (figure 3(b)). The "Maintenance and Scheduling Production" model (`MaintSchedProd`, or `MSP` for short) is largely based on classical Petri-nets used in production systems (see for instance [10]). It has been adapted to be used in conjunction with a scheduler and a maintenance strategy to work properly in our VLE simulator.

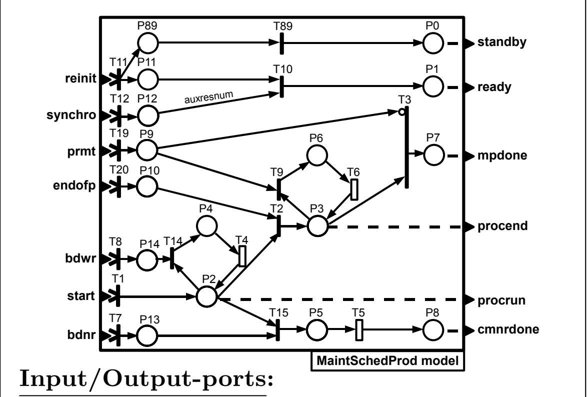
In the `MSP` component, two kinds of events may stop the production. The first one is the "breakdown with recovery" (`bdwr` for short) event. It can only occur when a process is running and (only) leads to an extra-duration in the processing time. The second one is the "breakdown with no recovery" (`bdnr` for short). This event stops the production and the current job is discarded. A new job needs to be rescheduled to replace the discarded one. An additional delay related to the corrective maintenance needed to repair the component is also considered.

As shown in figure 3, the `CMSP` component is a coupled model. This coupled version of the `MSP` component is composed of several interchangeable and parameterized models. These models are presented in the following sub-sections. Due to the implementation of the Petri-nets in the VLE simulator, all input-ports are connected to specific input transitions ">|" that only accept the input port as incoming event and no incoming arc. Several internal links are not represented directly when considering the detailed view of the `MSP` component (see figure 3(c)).



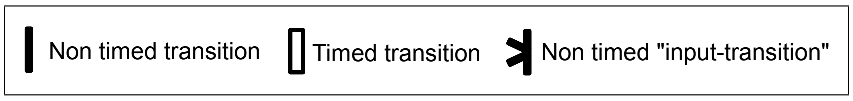
Input/Output-ports:
standby: external synchronization
prevprocready: previous component is ready
nextprocready: next component is ready
currentprocready: current component is ready
cmnrdone: end of corrective maintenance
mpdone: product & preventive maintenance done
bdnr: non-recoverable breakdown
endtempo: end of (external) temporization
fault: current process rejected due to breakdown
syncprevproc: synchronize previous component
syncnextproc: synchronize next component
synccurrentproc: synchronize current component
starttempo: start (external) temporization
endproc: normal end of current process
interrupttempo: stop temporization (breakdown)
Initial marking (model parameters):
P1token: number of tokens at place P1 at t=0
P2token: number of tokens at place P2 at t=0

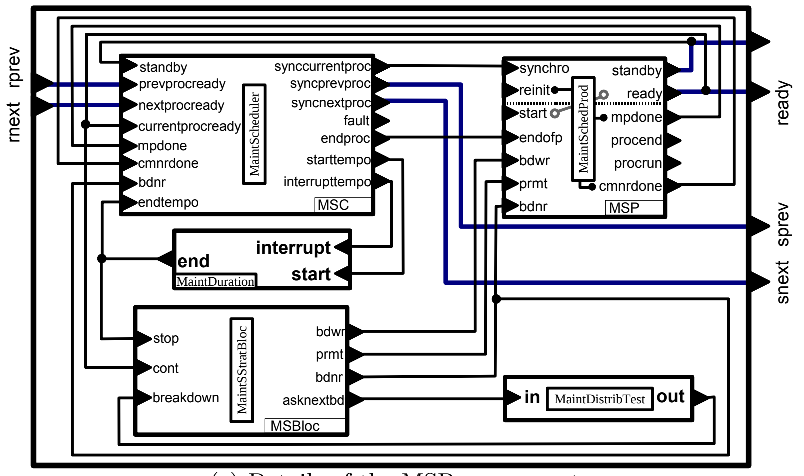
(a) Maint-schedulder Petri-net (MSC model)



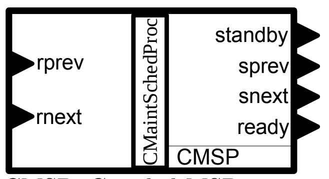
Input/Output-ports:
synchro: external synchronization of component
reinit: reinitialization post-process/maintenance
prmt: preventive maintenance request
endofp: end of (current) process/product signal
bdwr: breakdown with resume/recovery
start: immediate start of process
bdnr: non-recoverable breakdown
standby: component received a reinit signal
ready: component is ready to perform next product
mpdone: product & preventive maintenance done
procend: process end or preventive maintenance
procrun: performing/processing a product
cmnrdone: end of corrective maintenance
Timed transitions (and related parameters):
T4: bdwr duration, equal to bdwrdur parameter
T5: cmnr duration, equal to cmnrdur parameter
T6: prmt duration, equal to prmtdur parameter
Initial marking (fixed):
One token at P11 and one token at P89 at t=0.

(b) Maint-sched-prod Petri-net (MSP model)





(c) Details of the MSP component



(d) CMSP: Coupled MSP component

Fig. 3. The MSP component

4.1 MaintSchedProd model

The `MainSchedProd` model is designed to be compatible with production problems comprising stocks and auxiliary resources. The initial marking depends on the configuration of the overall `MSP` component. In the presented results, there are one token at `P11` (`reinit`) and one token at `P89` (`standby`) when starting the simulation (at t=0). The `MainSchedProd` Petri-net model is composed of two sub-nets. The first-one deals with synchronization aspects. It is composed of two input-ports (`synchro` and `reinit`) and two output-ports (`standby` and `ready`). The other sub-net comprises all remaining input-ports and output-ports and acts as the "operating part" (that is, the machine or the component of a machine to be modelled). Assuming that `reinit` and `standby` are appropriately initialized, here is a summary of the model constraints and internal functioning.

When no maintenance and breakdown event occur, the default path of tokens is the following. An external event on input-port `start` indicates that the current `MSP` component is processing a job (i.e. there is a token in place `P2`). At the end of current job, an external event is received on input-port `endofp` to indicate the end of processing phasis (i.e. a token is send to place `P3`). In addition to this default path, when current `MSP` component is processing as job, three paths are also possible when a recoverable breakdown `bdwr`, a non-recoverable breakdown `bdnr` or a preventive maintenance `prmt` occurs. This respectively send the token from `P2`, to `T14`→`P4`, `T15`→`P5` or `T2`→`P3`→`T9`→`P6`. As illustrated by the latter path, in this version of the `MSP` component preventive maintenance is performed just after the end of the current job. Several (consecutive) preventive maintenance requests `prmt` are allowed. They are memorized via several tokens in `P9` and performed through several cycles in the loop constituted of `P3, T9, P6, T6`. Similarly, several (consecutive) breakdowns with recovery `bdwr` are allowed. They are memorized via several tokens in `P14` and performed through several cycles in the loop constituted of `P2, T14, P4, T4`. Contrary to previous events, there is no loop when considering `P5` (`start` of `bdnr`). In fact, when `bdnr` occurs the objective is to restart the production as soon as possible just after corrective maintenance. All corrective maintenance operations are done at one time. The current product is scrapped and a new product must be processed.

Several parameters are available in this atomic-model. The first one is the number of tokens (parameterized by the `auxresnum` parameter) that are required to fire the `T10` transition. It is used to allow multiple synchronization signals/events before sending the `ready` signal. It might also be used to include auxiliary resources in our models. In the presented results, this parameter is systematically set to one. Three additional parameters are available through the (constant) durations of the `T4`, `T5` and `T6` timed-transitions. They are used to parameterize the duration of breakdowns and maintenances. In this paper, these parameters are set to constant values, but it is also possible to add several additional models to implement variable durations either randomly generated (see for instance the `MaintDuration` model) or read from actual production logs/histories. Setups might be included by using a timed transition at `T10`, transport delays might also be considered by using a timed transition at `T89`.

This paper focuses on maintenance aspects, so the `MaintSchedProd` model is configured in a simplified version, ignoring stocks and/or auxiliary resources. This leads to the following systematic "default wiring":
- output-port `ready` is connected to input-port `start`;
- output-ports `mpdone` and `cmnrdone` are connected to input-port `reinit`.

The `MaintSchedProd` needs a scheduler to obtain start/stop events, as well as a maintenance strategy to deal with breakdowns and preventive maintenance.

4.2 MaintScheduler model

The "Scheduler" model (`MaintScheduler`, or `MSC` for short) is based on a basic scheduling algorithm. In the presented study, its aim is to systematically load the production process at its maximum level of production through the `MaintSchedProd` model. This allows us to concentrate on maintenance aspects when considering heavily loaded periods. The details of its functioning is not explained in detail in this paper due to lack of space. However, it can be summarized in a few words. This scheduler acts as an infinite loop that sends sequences of events to the `MSP` model to process as many as possible jobs while taking into account the events provided by the maintenance strategy (`MaintStrat<strat>`). There is also an internal loop (materialized through the cycle `P10`→`T11`→`P11`→`T12`→`P12`→`T14`→`P9`→`T15`→`P90`→`T10`→`P10`) that is used to restart scrapped jobs resulting from `bdnr` breakdowns. The place `P10` acts as `P2` of the `MSP` model. A token in this place indicates that a job in running. The place `P11` send its token to `T13` when no non-recoverable breakdown occur, it sends its token to `T12` otherwise.

The `MaintScheduler` uses the `MaintDuration` model to generate process durations. The `MaintDuration` model is reduced to its simplest version to generate random durations. Its functioning is the following. When an event occurs at the `start` input-port, a random duration (using uniform distribution with ($\mu = 5$, $\sigma = 2$) is this example) is generated and a countdown is started. When the countdown is over or when an event occurs at the `interrupt` input-port, an event is send to the `end` output-port and this model is set to an "idle state" waiting for next start event. However, it is possible to replace this model by a more sophisticated one that reads a list of orders from a file and sorts the orders according to their duration. The sorting key might be based on one criterion (or several criteria) so as to implement various classical dispatching rules such as SPT (Shortest Processing Time first)... The interrupt input-port is also used to regenerate same duration to model several processing attempts of the same order when breakdowns occur.

The initial marking depends on the interconnections of the overall MSP component, as well as the kind of production (cyclic production...). In order to facilitate the integration of this model in various schemes, the initial marking (i.e. the number of tokens available at t=0) in places P1 and P2 are parameterizable through the configuration file of the models. The duration of the scheduled products are not stored and/or generated in/by this model. It also needs an external MaintDuration model to compute these durations. In this paper, the durations are randomly generated by the MaintDuration model. It is a temporization that can be interrupted when a breakdown occurs (see figure 3(c)).

The tandem MaintScheduler-MainSchedProd requires a maintenance strategy to deal with breakdowns and preventive maintenance.

4.3 MaintStrat<strat> model

The "Maintenance-Strategy" models (MaintStrat<strat>, or MS<strat> for short) are based on various maintenance strategies. Several models MaintStrat<strat> can be used. For instance, the bloc-strategy is available through the MainStratBloc model. It relies on a MaintDistrib<Distrib> model to "compute" breakdowns occurrences. In the presented example, the MainStratBloc model is directly implemented in C++ and included in the collection of classes and objects available in the VLE simulator. However, it might also be implemented via a Petri-net or a more sophisticated coupled-model. The functioning of the MaintStratBloc is the following. It is controlled by two input-ports, namely cont and stop, that are respectively used to (re-)activate or deactivate the outputs. The output-ports are used to send recoverable and non-recoverable breakdowns (respectively through bdwr and bdnr output-ports) as well as preventive maintenance requests (through prmt output-port). The algorithm is given in figure 4.

```
tbpm ⟵ pm
WHILE NOT EndOfSimulation DO
    Generate tbf using MaintDistrib distribution
    IF tbf < tbpm THEN
        // Next event is a breakdown (bdnr) followed by corrective maintenance
        Wait tbf time-units
        IF OutputAllowed THEN Send an event on bdnr output-port
        tbpm ⟵ tbpm − tbf
    ELSE
        // Next event is a preventive maintenance (prmt)
        Wait tbpm time-units
        IF OutputAllowed THEN Send an event on prmt output-port
        tbpm ⟵ pm
    ENDIF
ENDWHILE
tbf, tbpm: respectively, time before failure and time before preventive maintenance
pm: period of maintenance (this is a parameter of the algorithm/model)
OutputAllowed: boolean variable (flip-flop on cont and stop input-ports)
MaintDistrib: Probability distribution (this is seen as a parameter)
```

Fig. 4. MaintStratBloc algorithm.

The three main atomic-models MaintScheduler, MainSchedProd and MaintStratBloc constitute the heart of the MSP component. Additional models might also be used to enhance the component or to allow the assembly of several components. The following sub-section presents one of those models that is used in the presented experimental problem.

4.4 MaintSwitch model

The MaintSwitch model is a simple Petri-net (see figure 5). It is used to interconnect MSP components. The objective of this model is to send an event from the input-port (in) to the output-port out1 or out2 according to (respectively) the events received through input-ports avail1 or avail2. If several events generate simultaneous tokens in places P1 and P3, one output-port (out1 or out2) is randomly selected.

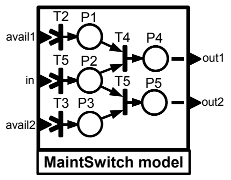


Fig. 5. MaintSwitch Petri-net model

4.5 Coupled CMaintSchedProd component

Thanks to the intrinsic properties of DEVS models, the "Coupled Maintenance and Scheduling Production" model (`CoupledMaintSchedProd`, or `CMSP` for short) is a coupled model composed of the previously described models linked together (see figure 3(d)). It constitutes the "building bloc" (i.e. an elementary component) of our parameterized simulation model to be optimized through an optimization tool.

4.6 Modelling through a Graphical User Interface

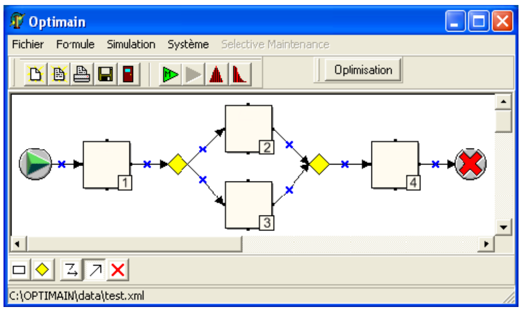


Fig. 6. The OptiMain GUI used to assemble four coupled models in a parallel/serial scheme

In order to facilitate the composition of simulation models, we are adapting a GUI (see figure 6) to the development of maintenance-production optimization and simulation models based on the VLE simulator. Previously dedicated to the rapid development of maintenance simulation models, this GUI was developed in the context of the OptiMain project [7,13]. However, in this paper the production aspects are also considered contrary to the OptiMain project that exclusively focuses on the maintenance point of view.

5 Our results

After a brief introduction to the Nelder-Mead optimization method, this section presents the results obtained by the optimization of a production system via our hybrid model.

5.1 Nelder Mead

In the presented study, the optimization tool is a Nelder-Mead [8] (Simplex) method, but other tools might be used. Nelder-Mead is a local optimization method which is frequently used. This deterministic method is known as "direct": it tries to solve the problem by directly using the value of the objective function, without calling upon its derivative. This method is especially appreciated for its robustness, its simplicity, its low use of memory (few variables) and its short computing time. This algorithm is robust because it is very tolerant with the noises in the values of the objective function. Contrary to the other methods which start from an initial point, the Nelder-Mead method uses a "polytope" departure. A polytope is a geometrical figure of (n+1) points, N being the dimension of the problem. The starting polytope is composed by a randomly selected point in the search space; and n other points selected so as to form a base, generally an orthogonal base. At each iteration of the algorithm, the n+1 points are used to determine a set of points which are obtained by using very simple algebraic operations, which result in elementary geometrical transformations (reflection, contraction, expansion, and polytope contraction). These points are accepted or rejected according to the value of the objective function. The polytope changes, it extends, contracts, with each movement. Thus it adapts to the search space, until it approaches an optimum. To determine the adequate transformation, the method uses only the value of the objective function at the considered points. With each transformation, the worst current point is replaced by the new given point. The stopping condition of the algorithm depends on the difference in value of the objective function between the best and the worst points.

5.2 Experimental results

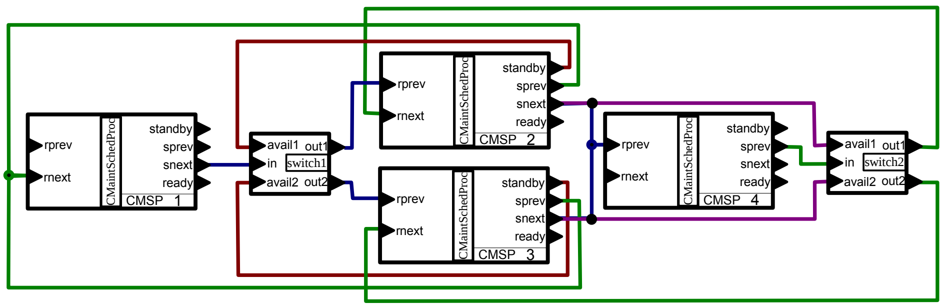


Fig. 7. Four coupled models in a parallel/serial scheme

Several preliminary tests have been done, using serial or parallel combinations of `CMSP` components. By extension, all production structures are accessible to our simulation model through a decomposition in serial and/or parallel pairs of `CMSP` components. In order to present the minimum serial and parallel pairs of `CMSP` components, the considered simulated and optimized model is composed of four identical `CMSP` components as presented in figures 6 and 7. Our aim is to determine optimal values of the preventive maintenance periodicity for the BRP strategy. The components are parameterized as follows: the duration for preventive maintenance action is set to T_p=0.01 and duration for corrective maintenance operation is T_c=0.1, lifetime of components 1 to 4 are associated with a normal distribution ($\mu = 5$, $\sigma = 2$). The processing times of the scheduled jobs are randomly generated according to a uniform distribution ranging from 1 to 5 time-units. The breakdowns with recovery (`bdwr`) are not used in this paper. The presented average results are obtained from twenty runs of 1800 time-units. The Nelder-Mead optimization method gives the following results for components 1 to 4, $(T_p^1, T_p^2, T_p^3, T_p^4) = (2.391, 3.484, 3.284, 2.396)$ with availability equals to 0.874. The values of T_p^i are close to those obtained thanks to the OptiMain tool. However there is a gap between the availability obtained via the OptiMain tool and the presented multimodel due to the integration of the production aspects in the `CMSP` components. The synchronizations resulting from the consideration of the production leads to add gaps in the schedule as well as extra-delays before preventive maintenance periods. This is illustrated by the Gantt chart given in figure 8. These gaps and delays indirectly modify the global availability of the system. The validation of our hybrid is confirmed by additional tests which are not presented in this paper due to lack of space.

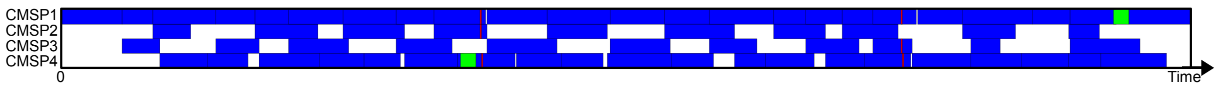


Fig. 8. Gantt chart related to the four CMSP models (first operations at the beginning of the simulation)

6 Conclusions and perspectives

Thanks to our VLE simulator, we have presented in this paper an hybrid method composed of the Nelder-Mead algorithm hybridized with a simulation multimodel. This multimodel is decomposed into several models implemented in the VLE simulator. This implementation is largely simplified by the extensions of the VLE simulator which provides several skeletons (similar to design patterns or constructs in other modelling tools/languages) to guide the implementation of the models.

All possibilities of the simulation model are not used in this paper. Our next objective is to provide a new framework to optimize the combined scheduling of production and maintenance. Short term work will consist in the integration of more efficient maintenance strategies as well as sophisticated schedulers. We are also working on the building blocs (constructs) of our multimodels that will provide a complete GUI, easy to understand by decision makers.

References

[1] R. E. Barlow and F. Proshan. *Theory Of Modeling and Simulation.* Wiley Interscience, 1976.

[2] F. J. Barros. Dynamic structure discret event system specification: Formalism, abstract simulators and applications. *Winter Simulation*, 13(1):35–46, 1996.

[3] F. J. Barros. Modeling formalisms for dynamic structure systems. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 7:501–515, october 1997.

[4] P. A. Fishwick. *Simulation Model Design and Execution.* Prentice Hall, 1995. ISBN 0130986097.

[5] P. A. Fishwick and B. P. Zeigler. A multi-model methodology for qualitative model engineering. *ACM transaction on Modeling and Simulation*, 2(1):52–81, 1992.

[6] C. Jacques and G. A. Wainer. Using the cd++ devs toolkit to develop petri nets. In *SCS Conference*, 2002.

[7] T. Lust, O. Roux, F. Riane, and P. Dehombreux. Simulation-based framework for maintenance optimization. In *ISC'2005*, pages 23–27, Berlin, Germany, June 9-11 2005. IPK Fraunhofer Institute.

[8] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7(4):308–313, 1965.

[9] J. L. Peterson. Petri nets. *ACM Computing Surveys*, 9(3):223–252, 1977.

[10] J.-M. Proth and X. Xie. *Les réseaux de Petri pour la conception et la gestion des systèmes de production.* Masson, 1995. ISBN 2225846499.

[11] G. Quesnel, R. Duboz, and É. Ramat. The virtual laboratory environment - an operational framework for multi-modelling, simulation and analysis of complex dynamical systems. *Simulation Modelling Pratice and Theory*, 2009. To be published.

[12] G. Quesnel, R. Duboz, É. Ramat, and M.K. Traore. VLE - A Multi-Modeling and Simulation Environment. In *Moving Towards the Unified Simulation Approach, Proceedings of the Summer Simulation 2007 conference*, pages 367–374, San Diego, USA, July 2007. SCS - ACM.

[13] O. Roux, M. A. Jamali, D. Aït Kadi, and E. Châtelet. Development of simulation and optimization platform to analyze maintenance policies performances for manufacturing systems. *International Journal of Computer Integrated Manufacturing*, 21(4):407–414, 2008.

[14] B. P. Zeigler. *Theory Of Modeling and Simulation.* Wiley Interscience, 1976.

[15] B. P. Zeigler, D. Kim, and H. Praehofer. *Theory of modeling and simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems.* Academic Press, 2000.