Int. J. Production Economics 143 (2013) 3–12

Contents lists available at ScienceDirect

Int. J. Production Economics

journal homepage: www.elsevier.com/locate/ijpe

Optimization of preventive maintenance through a combined maintenance-production simulation model

O. Roux [a,*], D. Duvivier [b,c], G. Quesnel [d], E. Ramat [b,c,d]

[a] Louvain School of Management and Catholic University of Mons, Chaussée de Binche, 151, B-7000 Mons, Belgium
[b] Univ Lille Nord de France, F-59000 Lille, France
[c] Université du Littoral Côte d'Opale, LISIC, BP 719, F-62228 Calais Cedex, France
[d] INRA, UR875 Biométrie et Intelligence Artificielle, F-31326 Castanet-Tolosan, France

ARTICLE INFO

Article history:
Received 8 January 2010
Accepted 4 November 2010
Available online 19 November 2010

Keywords:
Metaheuristics
Simulation
Decision-making
Multimodeling
Petri-nets
Maintenance

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 modeled 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.

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 (Percy and Kobbacy, 2000). The optimization consists in determining the best "parameters combination" which provides the best values of the technical and economical criteria (see for instance Rezg et al., 2005; Boschian et al., 2009). However, in most cases, it appears to be very difficult to use analytical approaches without formulating restrictive hypotheses. 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 more and more efficient solutions with an optimization tool and to evaluate them via a simulation model until a halt criterion is satisfied. This approach has already been studied in the literature (see for instance Boschian et al., 2009; Riane et al., 2009). This integration is illustrated in Fig. 1. In the following sections, according to Talbi (2002), this assembly of different units, at different levels of combinations is called a "hybrid model".

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 model. 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 (Wang, 2002; Roux et al., 2008). We consider for this paper one basic replacement policy (Bloc Replacement Policy BRP). Barlow and Proshan (1976) consider also

* Corresponding author.
E-mail addresses: olivier.roux@fucam.ac.be (O. Roux), david.duvivier@lisic.univ-littoral.fr (D. Duvivier), gauthier.quesnel@toulouse.inra.fr (G. Quesnel), ramat@lisic.univ-littoral.fr (E. Ramat).

0925-5273/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2010.11.004

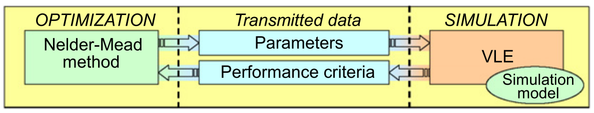


Fig. 1. Optimization and simulation integration.

the BRP where the replacements are undertaken at KT with $K=1,2,3,\ldots$ and T a fixed time, or at failure (see Fig. 2). Only new items are used to perform replacement. C_p and C_c are respectively the preventive and corrective replacement costs. Similarly T_p is defined as the duration for preventive maintenance action, and T_c as the duration for corrective maintenance operation. This maintenance strategy is also used in the simulation model described in the following sections.

In the result section, the availability is considered as the criterion to maximize. Effectively, since we are simultaneously considering production and maintenance in a context where production costs are higher than maintenance costs, the availability is a more adequate criterion than the maintenance costs. However, our model can be used to optimize the maintenance costs if needed.

3. Simulation

This section presents the VLE simulator and the underlying paradigm and formalism. This simulator relies on strong concepts and intrinsically provides multimodeling 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 (Peterson, 1977).

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 1980s, Fishwick and Zeigler (1992) 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), (Fishwick, 1995). 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 called models aggregation. In this context, a major issue is how to deal with the coupling of heterogeneous models. Several works dealing with the coupling of heterogeneous models have already been published. For a review of concepts and techniques, see the book of Zeigler et al. (2000). With DEVS, discrete event system specification, Zeigler (1976) 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 (DEVS) formalisms and their associated extensions, in particular Petri-nets.

3.1.1. Discrete event simulation

Our works take place in the Modeling and Simulation (M&S) theory defined by Zeigler (1976). 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 Zeigler et al.'s (2000) book 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, δ_{int}: $S\rightarrow S$ is the internal transition function δ_{ext}: $Q\times X\rightarrow S$ is the external transition function, λ: $S\rightarrow Y$ is the output function, $ta: S\rightarrow\mathbb{R}_0^+$ is the time advance function, $Q=\{(s,e)|s\in S, 0\le e\le ta(s)\}$ is the set of total states, 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. This leads to the following notation:

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

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 develops the VLE simulator, based on the DEVS formalism.

3.1.2. VLE

VLE (Quesnel et al., 2009, 2007, *Virtual Laboratory Environment*[1]) 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 (DSDE) formalism (Barros, 1997) which provides the abstract simulators for parallel DEVS (PDEVS) (Zeigler et al., 2000) for the parallelization of atomic models and dynamic structure DEVS (DSDEVS) (Barros, 1996) for the M&S of systems where drastic changes of structures and behaviors can occur 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; for instance, cellular automata, ordinary differential equations (ODE), spatialized ODE, difference equations, various finite state automata (Moore, Mealy, UML statecharts, Petri-nets, etc.) and so on.

This framework can be used to model, simulate, analyze and visualize dynamics of complex systems. His main features are: multimodeling abilities (coupling heterogeneous models), a general formal basis for modeling 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

In order to include the Petri-nets formalism in VLE, the DEVS approach is applied to the Petri-nets (Peterson, 1977). Works dealing with the mapping of Petri-nets into DEVS exists (see Jacques and Wainer, 2002 for instance). In these works, places and transitions are specified as atomic models and the network as coupled model. In our approach, a Petri-net simulator is wrapped in a DEVS simulator. In the following paragraphs, 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 and $p=card(P)$;
$T=\{t_1,\ldots,t_t\}$ is a set of transitions and $t=card(T)$;
$F\subseteq(P\times T)\cup(T\times P)$;

[1] http://www.vle-project.org

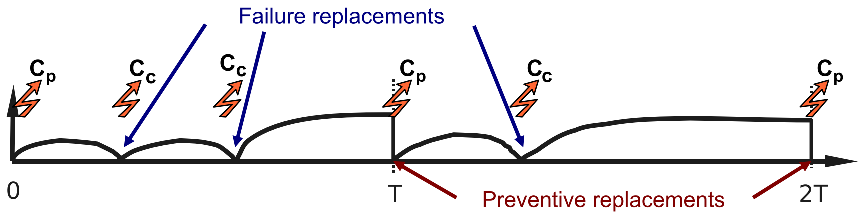
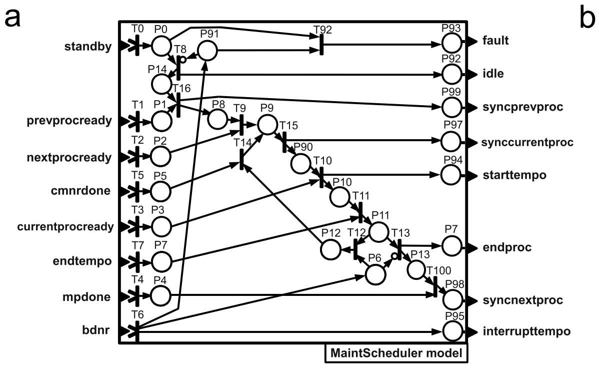
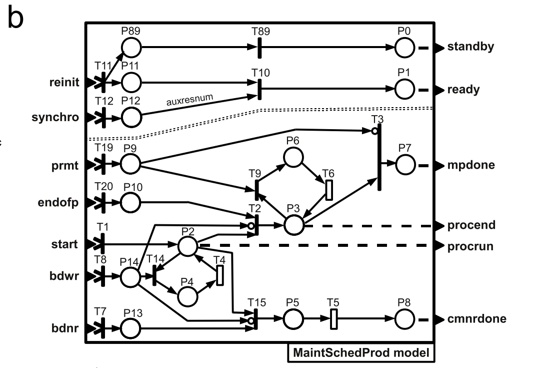


Fig. 2. Availability of the system subject to the Block Replacement Policy.



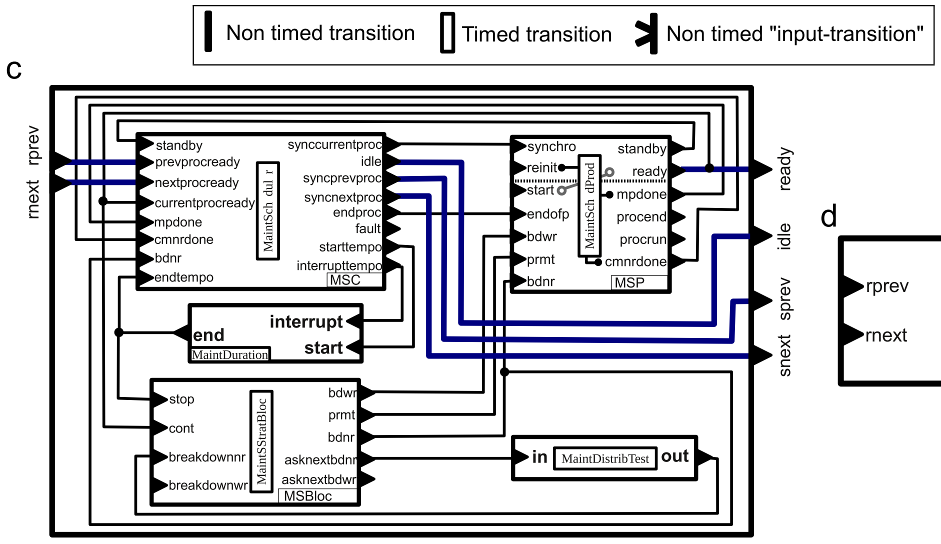
Input/Output-ports:
bdnr: non-recoverable breakdown
cmnrdone: end of corrective maintenance
currentprocready: scheduler is ready
endproc: normal end of current process
endtempo: end of (external) temporization
fault: process rejected due to breakdown
idle: scheduler in idle state
interrupttempo: stop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



Input/Output-ports:
bdnr: non-recoverable breakdown
bdwr: breakdown with resume/recovery
cmnrdone: end of corrective maintenance
endofp: end of (current) process/product
mpdone: product & prev. maintenance done
prmt: preventive maintenance request
procend: process end or prev. maintenance
procrun: performing/processing a product
ready: ready to perform next product
reinit: initialization post-process/bdnr/prmt
standby: component received a reinit signal
start: immediate start of process
synchro: external synchro of component
Timed transitions, related parameters:

T4:	bdwr duration, bdwrdur parameter
T5:	cmnr duration, cmnrdur parameter
T6:	prmt duration, prmtdur parameter
T89:	might be timed to simulate setups

Initial marking (fixed):
1 token at P11 and 1 token at P89 at t=0.



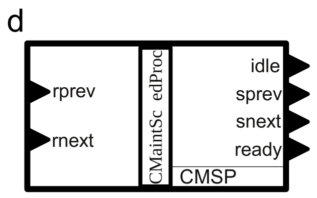


Fig. 3. The MSP component. (a) Maint-scheduler Petri-net (MSC model). (b) Maint-sched-prod Petri-net (MSP model). (c) Details of the MSP component. (d) CMSP component.

W is a weight function: $F \to \mathbb{N}, W = \{\ldots,((p_i,t_j),w_{ij}^-),\ldots,((t_j,p_i),w_{ji}^+)\}$;
m_0 is the initial marking: $P \to \mathbb{N}, m_0 = \{(p_1,m_1^0),\ldots,(p_p,m_p^0)\}$.

In the context of this paper, this definition is not sufficient, therefore timed transition and inhibitor arc are added to specification and simulator. The definition of a petri-net is now completed by a wrapping function denoted χ. This function is divided into two parts:

$$\chi_T = \{(p,t_j)\} \quad \text{where } p \in X \text{ and } t_j \in T$$

$$\chi_P = \{(p,p_i)\} \quad \text{where } p \in Y \text{ and } p_i \in P$$

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 a Petri-net. If an event arrives on a port belonging to χ_T, the transition t_j is fired.

In this approach, information 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 in various ways so as to study the interactions 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. In the presented work, the VLE simulator is used to implement our so-called "MSP component" and the optimization tool is a Nelder–Mead (Simplex) method, but other optimization methods might also be used. The MSP component comprises several parameters (number of jobs to be performed, statistical distributions, durations of maintenances, etc.) 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 Fig. 3 in their basic version. The first one is the scheduler, named MaintScheduler (see Fig. 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 replaced by a more realistic scheduler based on a set of decision rules for instance. The second Petri-net-based model, named MaintSchedProd, is the actual operating part of the component, in charge of coupling production and maintenance aspects (Fig. 3(b)). The whole "Maintenance and Scheduling Production" model (MSP for short) is largely based on classical Petri-nets used in production systems (see for instance Proth and Xie, 1995). 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 version of the MSP component presented in Fig. 3, 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 instance of the same 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. Preventive maintenance (prmt) is also handled by the MSP component but the related event does not stop the job being processed, instead the preventive maintenance occurs at the end of the current job.

As shown in Fig. 3, the CMSP component is a coupled model. This coupled version of the MSP component is composed of several interchangeable and parametrized 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.

More sophisticated versions of the MSP component are also used to generate the results presented in Section 5, when comparing VLE to other simulators. These versions include additional external synchronization input-ports as well as supplementary output-ports providing information about the state of the MSP component to external "schedulers". These added functionalities are not detailed here since they will require lots of space without providing substantial useful information on the functioning of 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 place 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 modeled). 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 phases (i.e. a token is send to place P3). In addition to this default path, when current MSP component is processing a 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 parametrize 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 or setup 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 breakdowns with recovery (bdwr), stocks and auxiliary resources. This leads to a systematic "default wiring" where the output-port ready is connected to input-port start and the 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 are not explained 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: When an event occurs at its start input-port, a random duration (using a uniform distribution for 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 these orders on the basis of various criteria so as to implement classical dispatching rules such as SPT (shortest processing time first) for example. 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, etc.). 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 parametrizable 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 Fig. 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" breakdown occurrences. In the presented

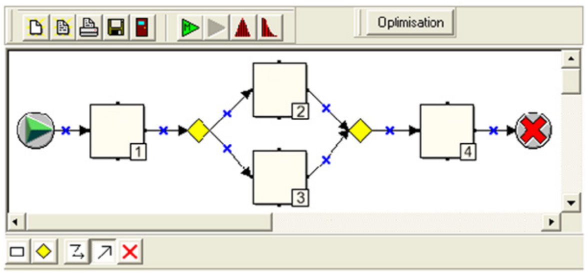


Fig. 5. The OptiMain GUI used to assemble four coupled models in a parallel/serial scheme.

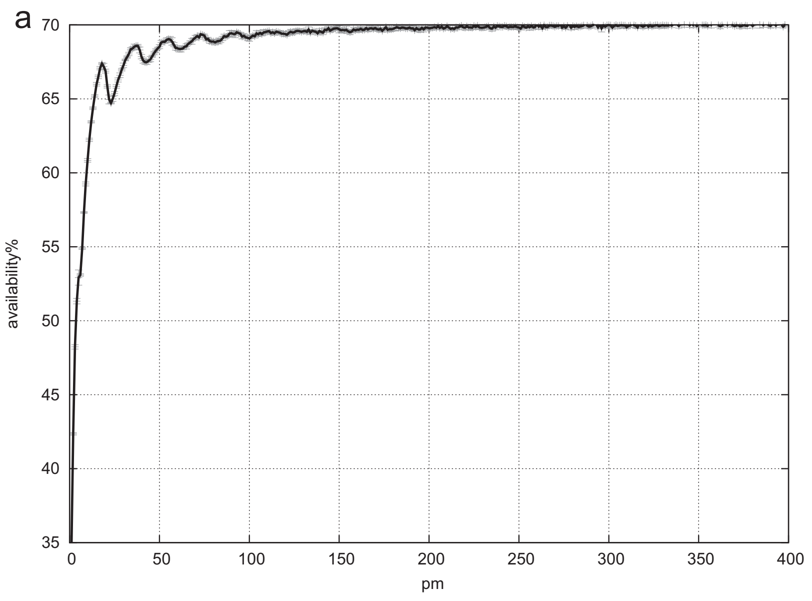
```
tbpm ⟵ pm
WHILE NOT EndOfSimulation DO
  Generate tbf using MaintDistrib distribution
  IF tbf < tbpm THEN  // Next event is a breakdown (bdnr) followed by corrective maintenance
    Wait tbf time-units
    IF OutputAllowed THEN Send an event on bdnr output-port
    tbpm ⟵ tbpm − tbf
  ELSE                // Next event is a preventive maintenance (prmt)
    Wait tbpm time-units
    IF OutputAllowed THEN Send an event on prmt output-port
    tbpm ⟵ pm

tbf          : time before failure
tbpm         : time before preventive maintenance
pm           : preventive maintenance period; this is a parameter of the algorithm
OutputAllowed: boolean variable (flip-flop on cont and stop input-ports)
MaintDistrib : Probability distribution (this is seen as a parameter)
```

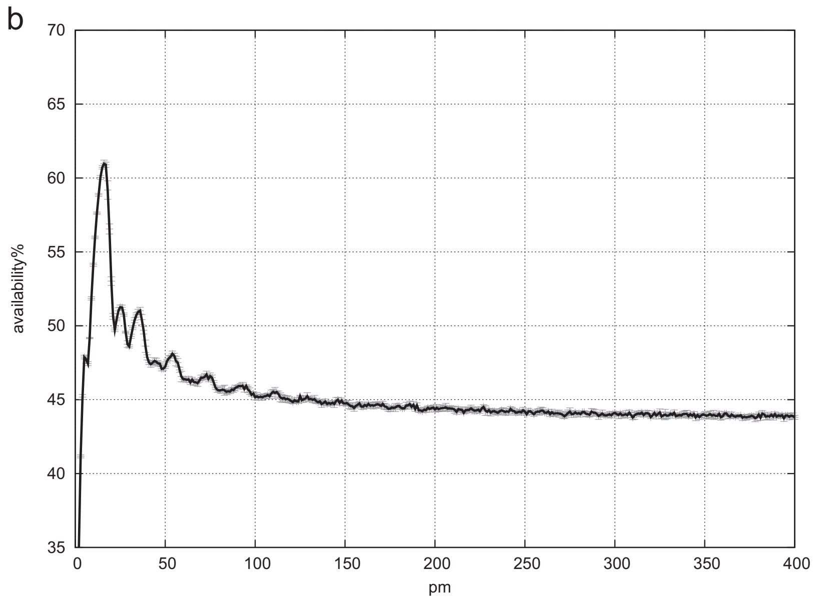
Fig. 4. MaintStratBloc algorithm.

example, the `MainStratBloc` model is directly implemented in C++. However, it might also be implemented via a Petri-net, a finite state automaton or a more sophisticated coupled-model. The `MaintStratBloc` 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 (through `bdwr` and `bdnr` output-ports) as well as preventive maintenance requests (through `prmt` output-port). The algorithm is given in Fig. 4.

The three main atomic-models `MaintScheduler`, `MainSchedProd` and `MaintStratBloc` constitute the heart of the `MSP` component. The following sub-section presents one simple example of additional models that can be used to enhance the component or to allow the assembly of several components: The `MaintSwitch` model.



Parameters:
jobs durations: uniform distribution [2,5]
breakdowns distribution: Weibull(13,20)
bdnr / prmt duration: $T_c=2.5$ / $T_p=2.5$
total duration: 10000



Parameters:
jobs durations: uniform distribution [2,5]
breakdowns distribution: Weibull(13,20)
bdnr / prmt duration: $T_c=9.5$ / $T_p=3.5$
total duration: 10000

Fig. 6. To maintain or not to maintain? This is the question! (a) One CMSP with short breakdowns. (b) One CMSP with long breakdowns.

4.4. MaintSwitch model

The `MaintSwitch` model is a simple Petri-net 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` (see Fig. 7). If one or several couple(s) of events are simultaneously sent to the input-ports `avail1` and `avail2`, one output-port (`out1` or `out2`) is randomly selected.

4.5. Coupled CMaintSchedProd component

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

4.6. Modeling through a graphical user interface

In order to facilitate the composition of simulation models, we are adapting a GUI (see Fig. 5) 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 (Lust et al., 2005; Roux et al., 2008). 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 (1965) (Simplex) method, but other tools might be used. To be more precise, our implementation uses the Nelder Mead method included in the Gnu Scientific Library (Gough, 2009). 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 of 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

In order to assess our coupled model, two "academical" scenarii based on two parametrizations of a `CMSP` component are used to determine the optimal preventive maintenance period (`pm`) on a mono-component production system. In Fig. 6, the abscissa are the values for `pm`. The ordinates are the percentage of availability, this corresponds to the ratio between the total time while the `CMSP` is processing jobs over the total elapsed time (i.e. 10 000 in these examples). In the first case (Fig. 6(a)), the breakdowns have no significant impact on the production excepted when preventive maintenance occurs so often that it reduces the availability of the machine or equipment modeled by the `CMSP`. In the second case

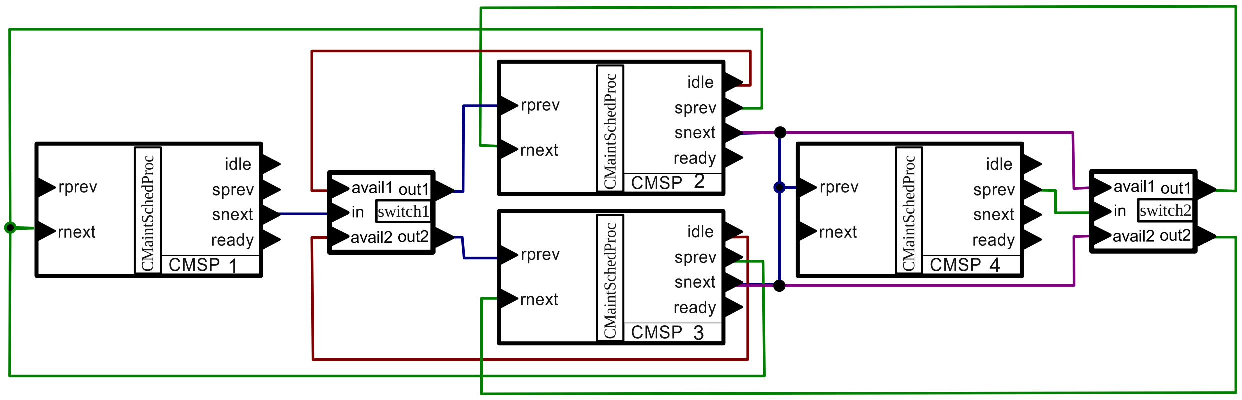


Fig. 7. Four coupled models in a parallel/serial scheme.

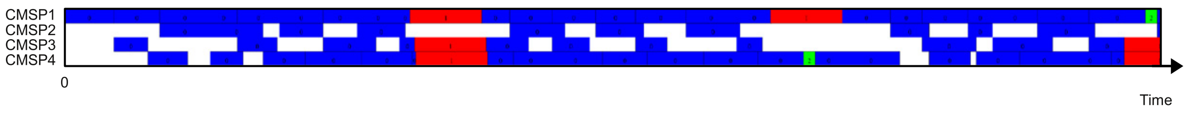


Fig. 8. Gantt chart (first operations at the beginning of the simulation $T \in [0,1000]$).

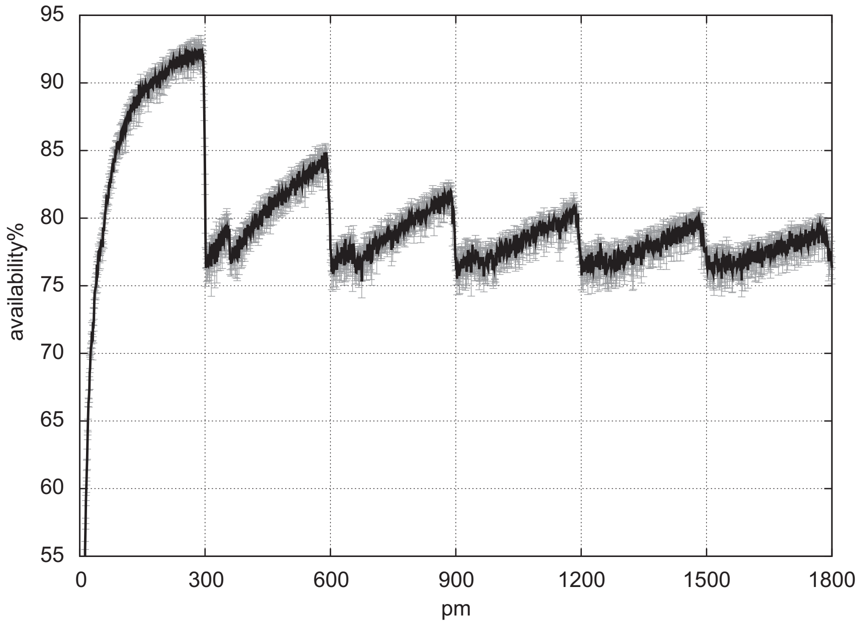


Fig. 9. Impact of the maintenance period (pm) on the availability in the 4-CMSP model.

(Fig. 6(b)), the breakdowns have significant impact on the production and the maintenance period must be carefully adjusted. These are well-known results, but they perfectly illustrate the fact that it is crucial to estimate the impact of the maintenance on the production before trying to optimize the maintenance policy. So, we assume that the studied problem is such that the maintenance has significant impact on the production in the remainder of this paper.

Several preliminary tests have been done, using serial or parallel combinations of CMSPs. By extension, all production structures are accessible to our simulation model through a decomposition in serial and/or parallel pairs of CMSPs. In order to present the minimum serial and parallel pairs of CMSPs, the considered simulated and optimized model is composed of four identical CMSPs as presented in Figs. 5 and 6.

Our aim is to determine optimal values of the preventive maintenance periodicity for the BRP strategy. The components are parametrized as follows: the duration of preventive maintenance action is set to $T_p=10$ and duration of corrective maintenance operation is $T_c=60$, lifetime of each component is modeled by a Weibull distribution ($\lambda=300$, $k=200$). The processing times of the scheduled jobs are randomly generated according to a uniform distribution ranging from $d_{\min}=20$ to $d_{\max}=50$ time-units. The presented average results are obtained from 20 runs of $T=18\,000$ time-units. Each evaluation corresponds to the average value over seven simulations to take stochastic aspects into account. In the presented results, the set of parameters $(\lambda,k,T_p,T_c,d_{\min},d_{\max},T,\mathrm{pm})$ is common to all the CMSP excepted for the pm parameter. The number of runs (20), the number of simulation per evaluation (7) and the number of time-units per simulation (18 000) are chosen to obtain statistically significant results.

The Nelder–Mead optimization method gives the following average results for components 1–4, $(\mathrm{pm}^1,\mathrm{pm}^2,\mathrm{pm}^3,\mathrm{pm}^4)=(281.29,467.55,390.57,292.54)$ with availability equals to 92.9% in 23 iterations (i.e. 525 simulations, corresponding to 75 evaluations). The results are obtained in approximately 8 min on a Intel Core2 Duo CPU running at 2.80 GHz. 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 a Gantt chart given in Fig. 8. These gaps and delays in turn modify the global availability of the system.

In order to illustrate the quality of the solution found by the Nelder Mead algorithm, it is possible to use a combination of pm^i inspired by the best ever found solution to plot the variation of the availability while pm varies (see Fig. 9). To be more precise, $\mathrm{pm}^1=\mathrm{pm}^4=\mathrm{pm}$, $\mathrm{pm}^2=\mathrm{pm}^3=k*\mathrm{pm}$, where k is the ratio between the maintenance period of CMSP_2 and CMSP_3 and the maintenance period of CMSP_1 and CMSP_4 inspired by the best ever found solution. The only modification applied to the best ever found solution is the balance of pm^1 and pm^4 as well as pm^2 and pm^3 to take the symmetrical aspects of the modeled problem into account. In Fig. 9, each evaluation corresponds to the average value over seven simulations, the standard-deviation is also plotted via vertical segments above and below each average value. This graphical result shows that the “optimal” solution found by the Nelder Mead method is on top of a periodic curve that shows an overall decreasing amplitude. The periodical vertical fronts correspond to the values of pm that are multiple of the mean time between failure (MTBF), conditioned by the Weibull parameters in this example. The decrease of the availability illustrates the increasing degradation of the system, subject to more and more breakdowns, due to the lack of preventive maintenance.

In order to compare our results to the OptiMain (Lust et al., 2005; Roux et al., 2008) and Reliabilitix (Basile, 2007; Fleurquin et al., 2009) models, it seems reasonable to configure our model to obtain similar results based on Fig. 9, however it is not possible to integrate the production in the OptiMain and Reliabilitix tools. Moreover, these tools are based on a global synchronization of components, incompatible with the previously presented functioning of the CMSPs. Indeed, the CMSPs are configured in such way that the corrective maintenance automatically starts as soon as possible in each component without global synchronization. This makes sense when considering a production line with several teams of maintainers. However, the OptiMain and Reliabilitix tools are based on several strategies which impose to wait that all parallel components fail before stopping the whole system, start the maintenance of all components that require maintenance and restart the whole system at the end of last maintenance action. Moreover, in order to generalize our model and to extend our comparisons to other tools, the following “repair-strategies” are available in our model: repair all components independently (indep), repair all components in parallel (para), repair all components in sequence (seq), repair all parallel components simultaneously and other components in sequence (seqpara). In addition to these strategies, it is possible to perform maintenance

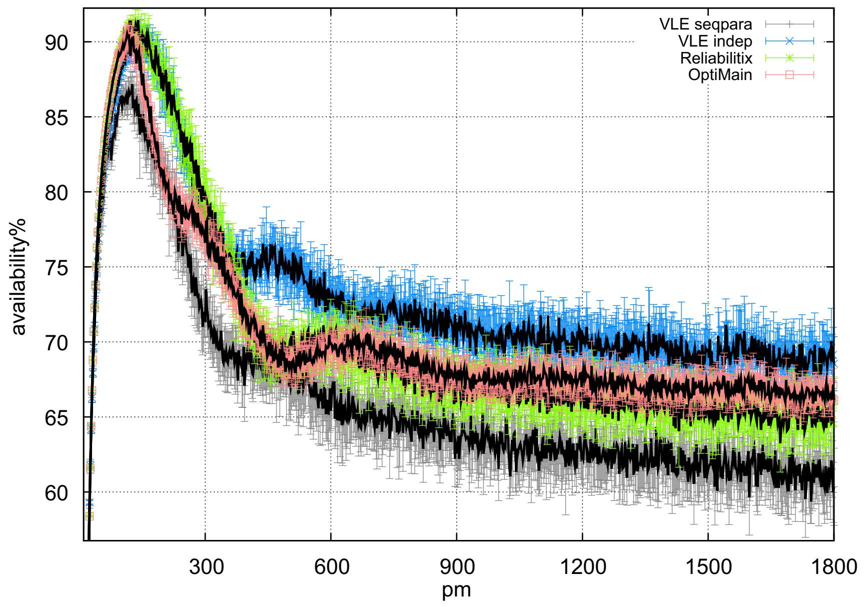


Fig. 10. Comparison of VLE, OptiMain and Reliabilitix on the 4-CMSP model.

Table 1
Comparison of best availabilities given by the models.

Model name	Optimal pm	Average availability (%)	Std. dev. availability
OptiMain	114	90.53	0.6044
Reliabilitix	140	91.45	0.6601
VLE seqpara	126	87.17	0.8084
VLE indep	146	90.40	1.0962

actions while some components of the system are still running or to impose to stop the whole system before starting maintenance actions.

In the previously presented results, lifetime of each component is modeled by a Weibull distribution ($\lambda=300$, $k=200$). These parameters lead to almost constant MTBF. This is useful to amplify the impact of the variation of pm on the system in Fig. 9, but not realistic when considering actual systems. In the following results, the Weibull distribution ($\lambda=300$, $k=5$) and the seqpara and indep repair-strategies are used. The production has also been disabled in our CMSPs to compare our results to OptiMain and Reliabilitix. Fig. 10 shows that the results of the three models are quite similar and even indistinguishable when considering different values of the preventive maintenance period (pm). When pm is greater than 350, Optimain and Reliabilitix tend to give indistinguishable values. It is also possible to parametrize our CMSPs to obtain similar results. The slight differences in the results of the three models might be explained by the completely different approaches and modeling tools that are used to model the described problem. Further investigations also show that the differences between the pseudo-random number generators might partly explain the differences in the results. However these results are sufficient to experimentally validate our model by comparison to two previously published approaches when disabling the production.

Moreover, in order to show the impact of the repair-strategies as well as the global synchronization mechanism implemented in our model, we have chosen to plot the maximum (using indep strategy with no synchronization) and the minimum values (using seqpara strategy with a global synchronization mechanism that implies to stop the whole system whenever a breakdown occur) that are generated by the various configurations available in our model. The obtained minimum and maximum results perfectly bound the results obtained by Optimain and Reliabilix and experimentally validate the obtained results.

The values for the optimal preventive maintenance period and the corresponding availabilities given by the models are presented in Table 1. On the basis of pm values, these results show that, when using the indep strategy, our model (denoted by VLE indep) provides similar results than those obtained by Reliabilitix. When using the seqpara strategy, our model (denoted by VLE seqpara) provides similar results than those obtained by Opti-Main. On the basis of the average availability, the obtained results are similar when taking the standard deviation into account. However, the seqpar strategy gives the worst value for the average availability, this is due to the fact that CMSP1, CMSP2//CMSP3 and CMSP4 are repaired in sequence, but also to the fact that we add a strict condition which imposes that each maintenance action requires to stop the whole system. The validation of our hybrid model is confirmed by additional tests which are not presented in this paper due to lack of space.

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 modeling tools/languages) to guide the implementation of the models.

Our results have been experimentally validated by comparisons to two previously published approaches when disabling the production. However, contrary to these approaches, our model is able to integrate production aspects and to simulate sophisticated scheduling and maintenances strategies thanks to the integration of several extensions in our VLE simulator such as “Decision rules”.

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 of 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

Barlow, R.E., Proshan, F., 1976. Theory of Modeling and Simulation. Wiley Interscience.

Barros, F.J., 1996. Dynamic structure discrete event system specification: formalism, abstract simulators and applications. Winter Simulation 13 (1), 35–46.

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

Basile, O., 2007. Evaluation on the uncertainty affecting reliability models. Journal for Quality in Maintenance Engineering 13 (2), 137–151.

Boschian, V., Rezg, N., Chelbi, A., 2009. Contribution of simulation to the optimization of maintenance strategies for a randomly failing production system. European Journal of Operational Research 197, 1142–1149.

Fishwick, P.A., 1995. Simulation Model Design and Execution. Prentice Hall.

Fishwick, P.A., Zeigler, B.P., 1992. A multi-model methodology for qualitative model engineering. ACM Transaction on Modeling and Simulation 2 (1), 52–81.

Fleurquin, G., Letot, C., Dehombreux, P., Riane, F., 2009. Assessing uncertainty from data collection to maintenance optimization. In: ICRMS 2009. 8th International Conference on Reliability, Maintainability and Safety, Chengdu, China, 2009, pp. 174–179.

Gough, B., 2009. GNU Scientific Library Reference Manual, third ed. Network Theory Ltd.

Jacques, C., Wainer, G.A., 2002. Using the CD++ DEVS toolkit to develop petri nets. In: SCS Conference.

Lust, T., Roux, O., Riane, F., Dehombreux, P., 2005. Simulation-based framework for maintenance optimization. In: ISC'2005, IPK Fraunhofer Institute, Berlin, Germany, pp. 23–27.

Nelder, J.A., Mead, R., 1965. A simplex method for function minimization. Computer Journal 7 (4), 308–313.

Percy, D.F., Kobbacy, K.A.H., 2000. Determining economical maintenance intervals. International Journal of Production Economics 67, 87–94.

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

Proth, J.-M., Xie, X., 1995. Les réseaux de Petri pour la conception et la gestion des systèmes de production. Masson.

Quesnel, G., Duboz, R., Ramat, E., Traore, M., 2007. VLE - a multi-modeling and simulation environment. In: Moving Towards the Unified Simulation Approach, Proceedings of the Summer Simulation 2007 Conference, SCS, ACM, San Diego, USA, pp. 367–374.

Quesnel, G., Duboz, R., Ramat, E., 2009. The virtual laboratory environment—an operational framework for multi-modelling, simulation and analysis of complex dynamical systems. Simulation Modelling Practice and Theory 17, 641–653.

Rezg, N., Chelbi, A., Xie, X., 2005. Modeling and optimizing a joint inventory control and preventive maintenance strategy for a randomly failing production unit: analytical and simulation approaches. International Journal of Computer Integrated Manufacturing 18 (2–3), 225–235.

Riane, F., Roux, O., Basile, O., Dehombreux, P., 2009. Simulation based approaches for maintenance strategies optimization. In: Ben-Daya, M., Duffuaa, S.O., Raouf, A., Knezevic, J., Ait-Kadi, D. (Eds.), Handbook of Maintenance Management and Engineering. Springer London, pp. 133–153 (Chapter 7).

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

Talbi, E.-G., 2002. A taxonomy of hybrid metaheuristics. Journal of Heuristics 8 (2), 541–564.

Wang, H., 2002. A survey of maintenance policies of deteriorating systems. European Journal of Operational Research 139, 469–489.

Zeigler, B.P., 1976. Theory of Modeling and Simulation. Wiley Interscience.

Zeigler, B.P., Kim, D., Praehofer, H., 2000. Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press.