Malinowski Jacek, 0000-0002-4413-1868
Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland,
jacek.malinowski{at}ibspan.waw.pl

Comprehensive model of operation process of complex technical system designed for simulation purposes

Keywords

system, operation process, dependent components, maintenance policy, discrete-event simulation, optimization

Abstract

In this chapter the author presents a comprehensive model of a multi-component technical system aimed at simulating and optimizing its operation process. Among other things, the model postulates load-related dependencies between the system's components and delayed repairs or replacements scheduled according to the components' priorities, where delays result from limited maintenance personnel. During the last several decades researchers in the field of reliability theory constructed various maintenance models, more or less applicable to real systems. Many authors follow purely analytical approach which, due to restrictive assumptions adopted for the purpose of analytical tractability, results in limited applicability of the considered models. These assumptions include: mutually independent components, exponentially distributed time-to-failure and time-to-repair, repairs started immediately after failures or carried out in negligible time, etc. The model proposed here does not impose such limitations, because it is designed to use simulation rather than analytical methods for computing purposes. The following assumptions bring the model closer to reality in comparison with its counterparts from the literature: 1) components are mutually dependent, i.e. a component's failure rate can depend of the states of some other components 2) after a repair a component can be "as good as new", "as good as used" or "as good as old" (perfect, imperfect, or minimal repair) 3) if maintenance personnel is busy, newly failed components await repair in a priority queue, 4) the state of a component may be hidden and its failure can only be revealed by inspection. The chapter's main result is a quite elaborate algorithm simulating the modeled system's behavior over time. Examples are given how, based on the proposed model and the adopted maintenance policy, selected reliability-related parameters can be optimized by repeated simulation. Although computationally intensive, the simulation approach allows to find performance and reliability characteristics for systems whose complexity or way of functioning rule out the application of analytical methods.

1. Introduction

During the past decades many reliability researchers designed and analyzed various maintenance models with the aim of finding optimal values of the maintenance parameters. The surveys of earlier and recent publications on maintenance modeling and optimization can be found in (Alabdulkarim et al., 2013; Chouhan et al., 2013; Nakagawa, 2008; Nakamura et al., 2017; Riane et al., 2009; Sarkar et al., 2011; Wang, 2002). Many works in this area endeavored to obtain exact analytical solutions, but the requirement of analytical tractability made it necessary to impose quite restrictive assumptions on the adopted models (components' independence, exponentially distributed time-to-failure

DOI: 10.26408/srsp-2021-15.

and time-to-repair, repairs started immediately after failures). The maintenance model proposed in this chapter is free from the above constraints, because it is mainly designed for the purpose of simulating the system's operating process rather than deriving analytical formulas allowing to compute and optimize its reliability parameters.

Admittedly, there exists a considerable number of publications regarding simulation of operation and maintenance processes, but they are mostly branch of industry-specific (Dietrich et al., 2017; Lyubchenko et al., 2020; Münsterberg, 2017; Panteleev et al., 2014; Soltanali et al., 2019; Vishnu & Regikumar, 2016), while there is rather a lack of literature on modelling and analysing generic processes. Lu et al. (Lu et al., 2019) is one of a few papers filling this gap. The current paper aims to build a possibly universal, widely applicable model allowing to detail the chronology of all events in the considered process and perform its technical and cost analysis. Thus, it takes the discrete-event simulation approach to maintenance modelling.

Let us begin with formulating the rules according to which the considered system operates. It is assumed that its components can be either fully operable or failed, and are interdependent in a certain sense. This means that the failure rate of a component can depend on the states of some other components, and a component's failure can cause some other components' failures. After a failure, a component is either replaced or repaired depending on its accumulated wear defined as the integral of the component's wear rate over time. More accurately, the integration interval starts at the moment when the new component is installed and ends at the moment when it fails. A component's replacement also takes place when the component reaches its age or wear limit, even if still in operable state (preventive replacement). Due to a limited number of maintenance personnel, a failed component can possibly wait in a queue for being serviced. The queue is organized on a priority basis which means that the higher priority components have precedence over the lower priority ones. However, for the sake of computer implementation, it is more convenient to have several queues, each one being a FIFO sequence with components of equal priority.

We will now outline the procedure simulating the behavior of a multi-component system operating in line with the above assumptions. The system starts working at time $t_0 = 0$, with all the components being new and fully operable. The procedure generates a sequence of time points in which components change their states. Hence, if a certain component changes its state at t_k, then t_{k+1} is the next instant at which the same or another component changes its state, $k \geq 1$. Although we distinguish only two "functional" states of a component – working and failed, the model also defines a number of states for a failed component, as set out in Section 3. Thus, a state change of a component occurs when one of the following events takes place:

- it fails,
- its hidden failure is detected,
- it reaches operating-age limit,
- its repair or replacement is completed.

We distinguish two types of failures – self-revealing and hidden ones. A failure of the latter type can be detected only during an inspection or when the component is preventively replaced, and until then it remains unrevealed. The probability that a failure is of one or the other type is known for each component. If this probability is equal to one, the component is subject to failures of one type only. When a failure is revealed, the component undergoes a repair or replacement, provided that the maintenance personnel is available, otherwise it is scheduled for repair or replacement and awaits its turn in a maintenance queue, according to the rules given in Section 4.

The presented maintenance model has several adjustable parameters. One of them is the age limit for a component's repair; if a component is found to be failed before this limit is reached, it will be repaired, otherwise it will be replaced. The second key parameter is the age limit for a component's operation; on reaching this limit the component will be preventively replaced, although still operable. The limit for operation should exceed the limit for repair, because a component too old to be repaired if failed can be too new to be replaced if operable. Other tunable parameters include the number of maintenance personnel and the time between consecutive inspections necessary to reveal hidden failures.

Each failure and maintenance action incurs a cost, and so does remaining of a component in a failed state. The system operator's task is to minimize the total operating cost during a specific time period, or the long-run operating cost per unit time. Either of the above costs is the objective function of the implicit optimization problem whose solution consists in finding the optimal values of the adjustable parameters defined in the previous paragraph, which are the problem's decision variables. According to the adopted notation (see Section 2) these are the parameters *RL*[*i*], *OL*[*i*], *MT* and *T*. Clearly, there can be more decision variables if the present model is extended. However, due to the model's complexity, it is not possible to find the exact solution of the considered optimization problem by means of an analytical method. Instead, the sought values will be obtained numerically with the use of repeated simulation, hence they will be quasi-optimal rather than optimal.

The rest of the chapter is organized as follows. In Section 2 the used notation is presented. Section 3 describes the life cycle of a single component from the start of operation to the completion of repair or replacement. Section 4 sets out the detailed rules for the system's operation. Based on these rules the algorithm simulating the system operation process is presented in Section 5. For illustration, the numerical results of its simplified version are discussed in Section 6, and the optimal values of two decision parameters – *OL*[*i*] and *T* – are found. The chapter is finished with Section 7 containing several conclusions and prospects for future work.

2. Notation and definitions

CDF – cumulative distribution function,
IFR – increasing failure rate,
PDF – probability density function,
TTF – time-to-failure,
TTR – time-to-repair,
n – number of components in the system,
$e_1,\ldots,e_n$ – the individual components,
Q – number of different maintenance priority levels (maintenance queues),
len[*q*] – the current length of the *q*-th maintenance queue, $q = 1,2,\ldots,Q$,
mq[*i*] – the number of maintenance queue to which e_i is assigned,
ndx[*q*, *r*] – the index of the component waiting in place *r* in queue *q*,
MT – the number of maintenance teams employed,
av_mt – the number of currently available maintenance teams,
Z[*i*] – set of components whose states can impact the wear rate of e_i. It should be noted that $i \notin Z[i]$,
rev[*i*] – the binary variable determining whether failures of e_i are self-revealing, if they are, then *rev*[*i*] = 1, else *rev*[*i*] = 0,
T – the time between two consecutive inspections performed in order to reveal hidden failures of the components for which *rev*[*i*] = 0,
$X_i(t)$ – the functional state of e_i at time *t*, $X_i(t) = 1$ if e_i is in operation, and $X_i(t) = 0$ if e_i is failed or under maintenance,
$Y_i(t)$ – the detailed state of e_i at time *t*, $Y_i(t) = -1$ when e_i is awaiting repair, $Y_i(t) = 0$ when e_i is being repaired or replaced, $Y_i(t) = 1$ when e_i is in operation, and $Y_i(t) = 2$ when e_i is in the state of undetected failure,
$A_i(t)$ – the actual age of e_i at time *t*, given by $t - s$, where *s* is the moment when e_i starts to operate as a new or replaced component,
$a_i(t)$ – the aging rate of component e_i at time *t*, $a_i(t)$ depends on $X_i(t)$ and the functional states of the components in *Z*[*i*] at time *t*, thus $a_i(t)$ can vary over time. The aging rate of a component under repair is assumed to be zero,
$E_i(t)$ – the effective age of e_i at time *t*, expressed by the integral $\int_{[s,t]} a_i(u)\,du$, where *s* is the moment when e_i starts to operate as a new or replaced component,
RL[*i*] – age limit for the repair of e_i: if a failure of e_i is revealed at time *t* and $A_i(t) \geq RL[i]$ then e_i is deemed too old to be repaired and must be replaced,
OL[*i*] – age limit for the operation of e_i: the still operable e_i must be replaced at time *t* when $A_i(t)$ reaches *OL*[*i*] (preventive replacement), as pointed out earlier, *OL*[*i*] > *RL*[*i*],
$L(i, E)$ – the function generating the remaining lifetime of the operable e_i, provided that its effective age equals *E* and its aging rate is constant and equal to 1, $L(i, E)$ returns a random value and is called when the new, repaired, or replaced e_i starts operating. For example, if $L(i, E)$ is to return a Weibull distributed random value, where s_i is the shape parameter and the scale parameter expressing the aging rate of e_i is equal to 1, then

$$L(i,E) = [E^{\sigma_i} - \ln(1-U)]^{1/\sigma_i} - E, \tag{1}$$

U being the random value of the uniform distribution. Equation (1) is obtained using the inverse *CDF* of the Weibull distribution,

$R(i, E)$ – the function generating the time length of e_i's repair ($A < RL[i]$) or replacement ($A \geq RL[i]$), where E is the effective age of e_i, $R(i, E)$ returns a random value and is called when a repair or replacement of e_i begins,

$S_i(t)$ – the time elapsing from t to the next state change of e_i, provided that $Y_i(t) \geq 0$, the aging rate of e_i remains unchanged after t if $Y_i(t) = 1$, and no preventive replacements are carried out. Let us note that the actual time to the next state change of e_i can differ from $S_i(t)$, because e_i's aging rate can vary due to possible state changes of the components in $Z[i]$. In view of the preceding definitions we have:

$$S_i(t) = \begin{cases} L\big(i, E_i(t)\big)/\alpha_i(t) & \mid Y_i(\mathrm{t}) = 1 \\ R\big(i, E_i(t)\big) & \mid Y_i(\mathrm{t}) = 0 \end{cases} \tag{2}$$

if e_i enters state 1 or 0 at time t, and

$$S_i(t) = \lceil t/T \rceil \cdot T - t \tag{3}$$

if $Y_i(t) = 2$, where $\lceil \cdot \rceil$ denotes the least integer upper bound. $S_i(t)$ has to be recalculated when e_i is operable and a component in $Z(i)$ changes its functional state (the details are given further in the chapter),

$K_{fail}[i]$, $K_{rpr}[i]$, $K_{rplc}[i]$ – the cost of e_i's failure, repair or replacement,

$k_2[i]$ – the cost per unit time incurred due to remaining of e_i in state 2 (unrevealed failure),

$k_{\leq 0}[i]$ – the cost per unit time incurred due to remaining of e_i in state ≤ 0 (awaiting or under service),

k_{team} – labor cost of one team per unit time,

$K(t)$ – the total operating cost incurred up to time t,

k_{long} – the long run operating cost per unit time, defined as $\lim_{t\to\infty} K(t)/t$.

3. The life cycle of a single component

The state changes of a single component occur according to a random process with the state-space {–1, 0, 1, 2}, i.e. the state of e_i at time t is expressed by the random variable $Y_i(t)$ defined as follows:

$Y_i(t) = -1$: e_i is queued for service,

$Y_i(t) = 0$: e_i is undergoing repair or replacement,

$Y_i(t) = 1$: e_i is in operable condition,

$Y_i(t) = 2$: e_i is in the state of unrevealed failure.

If $Y_i(t) = -1$, then e_i is awaiting maintenance in the queue $mq[i]$, and is moved forward as the components preceding e_i are taken for repair or replacement. Let us note that if r is the place of e_i in the queue $mq[i]$, then $ndx[mq[i], r] = i$. This equality will be used in step 5 of the algorithm presented in Section 5. The inter-state transitions of a component are illustrated in Figure 1.

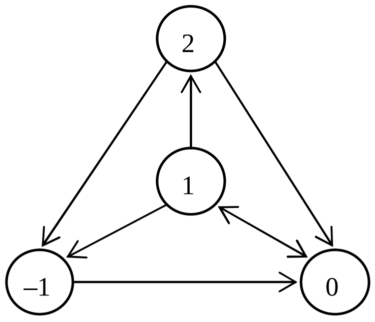

Figure 1. Possible transitions between a component's states.

Direct transition from 1 to 0 or from 2 to 0 takes place if a maintenance team is available when a component's failure is self-revealed or revealed by an inspection. For a component with no hidden failures the diagram in Figure 1 is simpler, because it does not contain the node representing state 2, as well as the arcs connected with it.

The diagram in Figure 2 illustrates the behavior of a component in the maintenance queue, i.e. placing a component in the queue, moving it ahead, and taking it for repair or replacement.

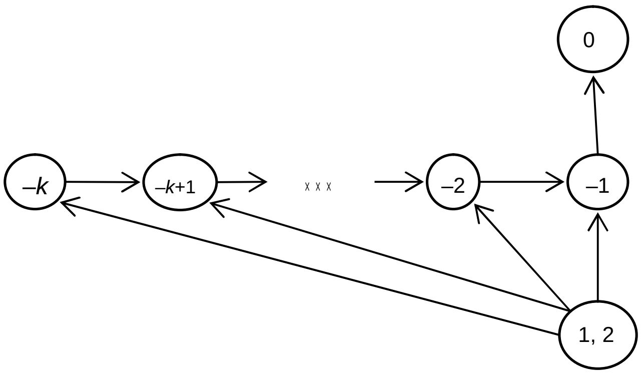

Figure 2. Dynamics of a maintenance queue.

Places in the queue are indicated with negative numbers that must not be confused with a component's states (however, note that if a component is in place –1, its state is also –1). The diagram doesn't illustrate situations when multiple components leave the queue at the same time (such an event has probability 0 if components are serviced independently of each other). If simultaneous transitions to state 0 were taken into account, the diagram would have to include transitions from $-k$ to $min(-k + s, 0)$, where $k \geq 2$ and $2 \leq s \leq av\_mt$, which would increase its complexity.

4. The rules for the system's operation

Since failures of some components are hidden ($rev[i] = 0$), it is necessary to perform regular inspections in order to reveal failed components and repair or replace them. Such inspections are performed every T time units, where T is one of the parameters that can be optimized. Let us assume that t is the time point when a self-revealing failure occurs or a hidden one is detected. If a maintenance team is available at t, the failed component (say e_i) is taken for repair ($A_i(t) < RL[i]$) or replacement ($A_i(t) \geq RL[i]$). If all maintenance teams are busy, e_i is placed in the respective maintenance queue, whose number is given by $mq[i]$. Besides, when a component's actual age reaches $OL[i]$, the component is taken or scheduled for replacement, even if still operable. Let us note that e_i can be in the state of undetected failure, and thus inoperable, when its actual age reaches $OL[i]$.

A component that cannot be serviced immediately when its failure occurs or is detected (due to temporal unavailability of maintenance personnel) is placed in one of Q maintenance queues, according to its maintenance priority. As mentioned earlier in this section, the maintenance priority of e_i (its respective queue number) is given by $mq[i]$ which belongs to the set $\{1,2,\dots,Q\}$. Each queue contains components with equal priorities and is managed on a first-in-first-out basis. The components in queue no. q are serviced before those in queue no. $q + 1$, where $1 \leq q \leq Q - 1$. The detailed state of a component waiting in a maintenance queue is equal to -1, i.e. $Y_i(t) = -1$ if e_i is failed and waiting to be serviced. In turn, the index of the component waiting in place r in queue q is given by $ndx[q, r]$, and $len[q]$ denotes the length of queue q. The arrays $mq[\cdot]$, $ndx[\cdot]$ and $len[\cdot]$ are needed by the simulation algorithm.

The components placed in maintenance queues wait until one of the maintenance teams becomes available, then the first to be serviced (say e_i in queue $mq[i]$) is taken for repair or replacement, and those remaining in the queue are moved one place forward. If $A_i(t) < RL[i]$, where t is the time when e_i leaves the queue, then e_i will be repaired, otherwise ($A_i(t) \geq RL[i]$) it will be replaced. Let us note that the age of e_i can be less than $RL[i]$ when e_i fails, but, due to waiting in the queue or remaining in state 2 (undetected failure), the age of e_i can exceed $RL[i]$ when e_i is taken for servicing. Thus, a component suitable for repair at failure can be no more suitable when its servicing starts. Also note that the actual age of e_i has to be known in order to decide whether it should be either repaired or replaced, or to stop the operation of e_i when its actual age reaches $OL[i]$. Therefore it is necessary for each component to record the time when it is first put in operation.

5. The simulation algorithm

According to the assumptions listed in the introduction, an algorithm for simulation of the system operation process has been constructed. Its main function is to generate the sequence of time points at which the system components change their detailed states, i.e. the sequence (t_k, $k \geq 1$), where t_k is the k–th instant at which any such change occurs. The algorithm runs in a loop whose body is composed of the following steps.

Step 1

Assign initial values to all the variables quantifying the system operation process. This is realized by the following pseudocode.

```
$k \leftarrow 0$;  $t_0 \leftarrow 0$;  $av\_mt \leftarrow MT$;
for $q = 1,\dots,Q$ do {
    $len[q] \leftarrow 0$;
}
for $i = 1,\dots,n$ do {
    $Y_i(t_0) \leftarrow 1$;  $A_i(t_0) \leftarrow 0$;  $E_i(t_0) \leftarrow 0$; $K(t_0) \leftarrow 0$;
    $S_i(t_0) \leftarrow L(i,0)/a_i(t_0)$;
}
```

At $t_0 = 0$ all components are operable ($Y_i(t) = 1$ for $i = 1,2,\dots,n$) and their effective age is equal to 0, therefore their anticipated sojourn times in the operable state are generated using $L(i, 0)$.

Step 2

If $k = k\_max$, stop. Otherwise add 1 to k.

Step 3

Compute $t_k - t_{k-1}$ as the smaller of the following two values:

$$min\big(\, S_i(t_{k-1}) \mid i\colon Y_i(t_{k-1}) \geq 0 \,\big) \quad (4)$$

and

$$min\big(\, OL[i] - A_i(t_{k-1}) \mid i\colon Y_i(t_{k-1}) \geq 1 \,\big). \quad (5)$$

The first minimum is computed over all i such that e_i is in a non-negative state at t_{k-1}, and the second – over all i such that e_i is operable or its failure is unrevealed at t_{k-1}. It is easy to check that t_k is the first moment after t_{k-1} when any state change occurs. Note that the components queued for maintenance are irrelevant to the value of t_k. Further, if $Y_i(t_{k-1}) = 1$ and $S_i(t_{k-1}) \leq OL[i] - A(t_{k-1})$, put

$fail[i] \leftarrow 1$,

otherwise put

$fail[i] \leftarrow 0$.

The variable $fail[i]$ determines whether e_i fails at t_k and is used in step 7 (cost calculation). Finally, for each i such that $Y_i(t_{k-1}) \neq 0$, put

$A_i(t_k) \leftarrow A_i(t_{k-1}) + (t_k - t_{k-1})$

and

$E_i(t_k) \leftarrow E_i(t_{k-1}) + a_i(t_{k-1})\cdot(t_k - t_{k-1})$.

If, in turn, $Y_i(t_{k-1}) = 0$ then put $A_i(t_k) \leftarrow A_i(t_{k-1})$ and $E_i(t_k) \leftarrow E_i(t_{k-1})$, because neither actual nor effective age of a component increases when it is under maintenance.

For clarity, the code realizing step 3 is presented below.

```
$t_k \leftarrow t_{k-1} + min$ [(4), (5)];
for $i = 1,\dots,n$ do {
  if ( $Y_i(t_{k-1}) = 1$ and $S_i(t_{k-1}) \leq OL[i] - A(t_{k-1})$)
  then {
    $fail[i] \leftarrow 1$;
  }
  else {
    $fail[i] \leftarrow 0$;
  }
  if ( $Y_i(t_{k-1}) \neq 0$ ) then {
    $A_i(t_k) \leftarrow A_i(t_{k-1}) + (t_k - t_{k-1})$;
    $E_i(t_k) \leftarrow E_i(t_{k-1}) + a_i(t_{k-1})\cdot(t_k - t_{k-1})$;
  }
  else {
    $A_i(t_k) \leftarrow A_i(t_{k-1})$;
    $E_i(t_k) \leftarrow E_i(t_{k-1})$;
  }
}
```

Step 4

Assign $Y_i(t_{k-1})$ to $Y_i(t_k)$ for each $i \in \{1,2,\dots,n\}$; this operation ensures that the components which do not change their states at t_k will retain them in the interval $[t_k, t_{k+1})$.

Step 5

Compute $Y_i(t_k)$ for each e_i such that $Y_i(t_{k-1}) \geq 0$ and $Y_i(t_k) \neq Y_i(t_{k-1})$, i.e. e_i is not awaiting service in the interval $[t_{k-1}, t_k)$ and its next state change (relative to t_{k-1}) takes place at t_k[*]. Note that every such e_i fulfills either of the following conditions:

$$min(\, S_i(t_{k-1}), OL[i] - A_i(t_{k-1})\,) = t_k - t_{k-1} \quad (6)$$

for i such that $Y_i(t_{k-1}) \geq 1$, or

$$S_i(t_{k-1}) = t_k - t_{k-1} \quad (7)$$

for i such that $Y_i(t_{k-1}) = 0$. If e_i fulfills (6) or (7), its state at t_k *is* determined as follows:

a) if $Y_i(t_{k-1}) = 2$ then e_i is placed in the maintenance queue at t_k, i.e. e_i changes its state to –1 at t_k,
b) if $Y_i(t_{k-1}) = 1$ then e_i changes its state to 2 at t_k, provided that $rev[i] = 0$ and an inspection or preventive replacement is not planned at t_k, otherwise e_i changes its state to –1 at t_k,
c) if $Y_i(t_{k-1}) = 0$ then e_i's repair or replacement is completed at t_k, i.e. e_i changes its state to 1 at t_k.

[*] If e_i is awaiting service in interval $[t_{k-1}, t_k)$, then it can change its state at t_k only if servicing of at least one component is finished at t_k. Thus, a change of e_i's state or place in the queue is secondary relative to state changes of components passing from state 0 to state 1. For this reason, the states or places in the queues of the components awaiting service are updated in the next (sixth) step.

In case (c) the variable *av_mt* is increased by the number of components changing their states to 1 at t_k. Moreover, if e_i's replacement rather than repair ends at t_k, the actual and effective age of e_i is set to 0. Let us note that e_i's replacement begins at $t < t_k$, thus $A_i(t) \geq RL[i]$ and, since $A_i(t)$ does not change in $[t, t_k)$, it holds that $A_i(t_k) = A_i(t)$. The code implementing step 5 is given below.

```
for $i$ = 1,2,…,$n$ do {
    if ( $Y_i(t_{k-1}) = 1$ and (6) holds ) then {
        if ( $rev[i] = 0$  and  $t_k < \lceil t_k/T \rceil \cdot T$  and
        $A_i(t_k) < OL[i]$ ) then {
            $Y_i(t_k) \leftarrow 2$;
        }
        else {
            $q \leftarrow mq[i]$;  $len[q] \leftarrow len[q] + 1$;
            $ndx[q, len[q]] \leftarrow i$; $Y_i(t_k) \leftarrow -1$;
        }
    }
    if ( $Y_i(t_{k-1}) = 2$  and  (6) holds ) then {
        $q \leftarrow mq[i]$;  $len[q] \leftarrow len[q] + 1$;
        $ndx[q, len[q]] \leftarrow i$; $Y_i(t_k) \leftarrow -1$;
    }
    if ( $Y_i(t_{k-1}) = 0$  and  (7) holds ) then {
        $Y_i(t_k) \leftarrow 1$; $avl\_r \leftarrow avl\_r + 1$;
        if ( $A_i(t_k) \geq RL[i]$ ) then {
            $A_i(t_k) \leftarrow 0$; $E_i(t_k) \leftarrow 0$;
        }
    }
}
```

Step 6

If $av\_mt > 0$, the states of at most *av_mt* first-to-be-serviced components are set to 0 at t_k, then *av_mt* and the maintenance queue parameters are updated accordingly. This task is accomplished by the code below:

```
if $av\_mt > 0$ then {
    for $q$ = 1,2,…,$Q$ do {
        $aux \leftarrow min( av\_mt, len[q] )$;
        for $r$ = 1,2,…,$aux$ do {
            $Y_{ndx[q, r]} \leftarrow 0$;
            $ndx[q, r] \leftarrow ndx[q, r + aux]$;
        }
        $len[q] \leftarrow len[q] - aux$;
        $av\_mt \leftarrow av\_mt - aux$;
        if $av\_mt = 0$ then break;
    }
}
```

Step 7

Compute $K(t_k)$ by adding the cost incurred during $(t_{k-1}, t_k]$ to $K(t_{k-1})$. This is realized as follows:

```
$K(t_k) \leftarrow K(t_{k-1}) + (t_k - t_{k-1}) \cdot MT \cdot k_{team}$ ;
for $i$ =1,2,…,$n$ do {
    if ( $Y_i(t_{k-1}) \leq 0$ ) then {
$K(t_k) \leftarrow K(t_k) + (t_k - t_{k-1}) \cdot k_{\leq 0}[i]$;
    }
    if ( $Y_i(t_{k-1}) = 2$ ) then {
        $K(t_k) \leftarrow K(t_k) + (t_k - t_{k-1}) \cdot \mathrm{k}_2[i]$;
    }
        if ( $Y_i(t_{k-1}) \neq 0$ and $Y_i(t_k) = 0$ ) then {
        if  ( $A_i(t_k) < RL[i]$ ) then {
                $K(t_k) \leftarrow K(t_k) + K_{rpr}[i]$;
        }
        else {
                $K(t_k) \leftarrow K(t_k) + K_{rplc}[i]$;
        }
    }
    if ( $Y_i(t_{k-1}) = 1$  and  $Y_i(t_k) \neq 1$  and
    $fail[i] = 1$ ) then {
        $K(t_k) \leftarrow K(t_{k-1}) + K_f[i]$;
    }
}
```

Remark 1. The cost of repair or replacement is added when either of these actions starts at t_k, because it is uncertain, prior to computing t_k, whether the actual age of e_i at t_k has reached $RL[i]$.

Remark 2. The last IF command adds $K_f[i]$ to $K(t_k)$, provided that e_i fails at t_k. This is determined by the variable $fail[i]$ whose value is assigned in step 3.

Step 8

Obtain $S_i(t_k)$ if at t_k the state of e_i changes to nonnegative, or e_i remains in state 1, but at least one e_j in $Z(i)$ changes its state. Otherwise put

$$S_i(t_k) \;\neg\; S_i(t_{k-1}) - (t_k - t_{k-1}).$$

The following piece of code realizes step 8:

```
for  $i$ = 1,2,…,$n$ do {
    if ( $Y_i(t_k)$ $^1$ $Y_i(t_{k-1})$ ) then {
        if ( $Y_i(t_k) = 0$ ) then {
            $S_i(t_k) \leftarrow R(i, E_i(t_k))$;
        }
        if ( $Y_i(t_k) = 1$ ) then {
            $S_i(t_k) \leftarrow L(i, E_i(t_k)) / a_i(t_k)$;
        }
        if ( $Y_i(t_k) = 2$ ) then {
            $S_i(t_k) \leftarrow \lceil t_k / T \rceil \cdot T - t_k$;
        }
    }

    else if ($Y_i(t_k) = Y_i(t_{k-1}) = 1$ and $Y_j(t_k) \neq Y_j(t_{k-1})$
    for at least one $j \in Z[i]$) then {
        $S_i(t_k) \leftarrow [\, S_i(t_{k-1}) - (t_k - t_{k-1}) \,] \cdot a_i(t_{k-1}) / a_i(t_k)$;
    }

    else {
        $S_i(t_k) \leftarrow S_i(t_{k-1}) - (t_k - t_{k-1})$;
    }
}
```

Remark 1. If e_i changes its state to 2 at t_k, then $t_k < \lceil t_k / T \rceil \cdot T$ and $A_i(t_k) < OL[i]$ in view of the rule (b) in step 5, thus $S_i(t_k) > 0$.
Remark 2. If the state of e_i remains unchanged at t_k, but the state of at least one e_j, $j \in Z(i)$, changes at t_k, then the aging rate of e_i also changes and the difference $S_i(t_{k-1}) - (t_k - t_{k-1})$ lengthens or shortens depending on the ratio $a_i(t_{k-1}) / a_i(t_k)$.

Step 9
Go to step 2.

For better understanding, the flow chart of the above algorithm is presented in Figure 3.

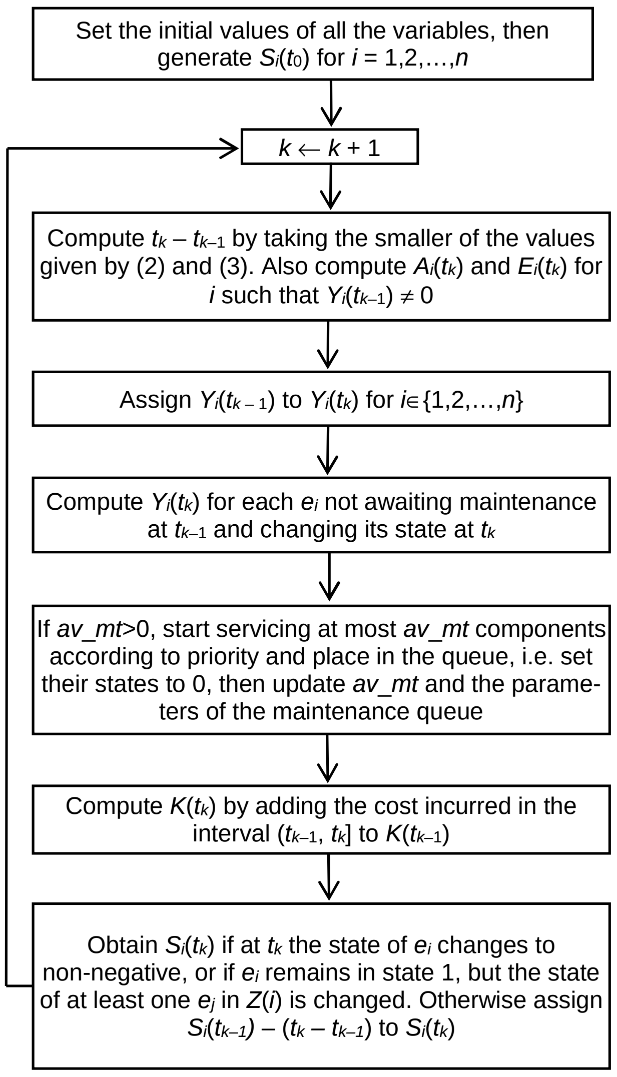
Set the initial values of all the variables, then generate $S_i(t_0)$ for $i = 1,2,\ldots,n$
$k \leftarrow k + 1$
Compute $t_k - t_{k-1}$ by taking the smaller of the values given by (2) and (3). Also compute $A_i(t_k)$ and $E_i(t_k)$ for i such that $Y_i(t_{k-1}) \neq 0$
Assign $Y_i(t_{k-1})$ to $Y_i(t_k)$ for $i \in \{1,2,\ldots,n\}$
Compute $Y_i(t_k)$ for each e_i not awaiting maintenance at t_{k-1} and changing its state at t_k
If av_mt>0, start servicing at most av_mt components according to priority and place in the queue, i.e. set their states to 0, then update av_mt and the parameters of the maintenance queue
Compute $K(t_k)$ by adding the cost incurred in the interval $(t_{k-1}, t_k]$ to $K(t_{k-1})$
Obtain $S_i(t_k)$ if at t_k the state of e_i changes to non-negative, or if e_i remains in state 1, but the state of at least one e_j in $Z(i)$ is changed. Otherwise assign $S_i(t_{k-1}) - (t_k - t_{k-1})$ to $S_i(t_k)$

Figure 3. Flow chart of the simulation algorithm.

6. The illustrative example

In this section we present the results obtained by implementing the algorithm from Section 5. Since the program implementing the algorithm is still under development, its current version corresponds to a simplified system model with the following features.

- The components are identical and mutually independent, i.e. the aging rate of a component is independent of the states of other components. As a consequence, $S_i(t_k)$ does not need to be updated when e_i remains in state 1 and a component in $Z(i)$ fails or is repaired at t_k (see step 8).
- All the components are non-repairable, thus a component is always replaced after failure with no regard to its age, i.e. the parameter $RL[i]$ is not defined for e_i, however, the parameter $OL[i]$ remains valid.
- When e_i enters state 1 or 0, $S_i(t)$ is simulated using the functions $L(i, E)$ and $R(i, E)$, where $E = 0$. This is because failed components are replaced with new ones whose effective age is equal to zero. Further, we assume that the time it takes to replace a component is not related to its effective age at failure.
- Each component's failures are self-revealing, i.e. $rev[i] = 1$ for $i = 1,\ldots,n$. As a consequence, there is no need for inspections, and there are only two non-negative states, i.e. 1 and 0.
- The system parameters have the following values:
  Number of components (n): 5
  Type of TTF's distribution: Weibull
  Scale parameter: 0.05
  Shape parameter: 1.5

Type of *TTR*'s distribution: Weibull
Scale parameter: 0.5
Shape parameter: 1.5
Unit of time: 1 day
Number of service queues (Q): 2
Components assigned to queue 1 (highest priority): e_1, e_2, e_3
Components assigned to queue 2: e_4, e_5
Number of service teams (MT): 1, 2, 3;
$K_{rplc}[i]$: 1000;
$K_{fail}[i]$: 4000;
$k_{\leq 0}[i]$: 900;
k_{team}: 40.

We adopt the two-parameter version of the Weibull distribution, i.e. the random variable W is Weibull-distributed if its *CDF* is expressed as follows:

$$Pr(W \leq t) = 1 - \exp[\,-(l\,t)^a] \qquad (8)$$

where l is the scale and a is the shape parameter. Thus, the location parameter is assumed to be zero. Note that in the provided example $a = 1.5$, i.e. $a > 1$. This is an intended choice, because preventive replacements are only justified if the distribution of the component's *TTF* belongs to the *IFR* class (Barlow & Proschan, 1975). As well known, the distribution defined by (6) has the *IFR* property if and only if $a > 1$. Other distributions of *TTF* and *TTR* are studied in (Barlow & Proschan, 1975) and (O'Connor & Kleyner, 2011).

The presented simulation algorithm was used for the purpose of finding the optimal values of the age limit for a component's operation and the number of service teams, i.e. the values of $OL[i]$ and MT minimizing k_{long} – the long run operating cost per unit time. Let $OL^*[i]$ be the optimal age limit for e_i. Since all the five components are assumed to be stochastically identical, it is reasonable to take $OL[1] = \ldots = OL[5] = OL$ and find $OL^* = OL^*[1] = \ldots = OL^*[5]$ by computing the cost functional for a wide range of the values of OL and choosing the OL for which the functional attains its minimum. The obtained results are shown in Table 1.

The parameter *k_max* is the number of iterations of the main loop (see step 2). The values of k_{long} are computed for *k_max* = 10000 and *k_max* = 100000. In the first case k_{long} reaches its minimum for $OL = 21$ and $MT = 2$. However,

Table 1. The values of k_{long} for various OL and MT (the asterisk indicates the local minimum of k_{long})

OL	*k_max* = 10000 k_{long} for MT = 1, 2, 3	*k_max* = 100000 k_{long} for MT = 1, 2, 3
60	1815, 1732, 1801	1792, 1747, 1786
50	1778, 1721, 1783	1790, 1743, 1777
45	1807, 1715, 1776	1784, 1737, 1787
40	1774, 1725, 1788	1774*, 1739, 1772
35	1776, 1734, 1777	1782, 1725, 1765
34	1799, 1743, 1769	1778, 1726, 1763
33	1763, 1759, 1750	1794, 1722, 1757
32	1771, 1739, 1762	1786, 1724, 1752
31	1769, 1721, 1764	1774*, 1724, 1759
30	1771, 1717, 1753	1781, 1713*, 1753
29	1766, 1704, 1758	1779, 1720, 1748
28	1776, 1714, 1728	1782, 1718, 1746
27	1777, 1702, 1739	1784, 1720, 1759
26	1779, 1727, 1754	1788, 1719, 1742*
25	1776, 1704, 1746	1789, 1709*, 1750
24	1781, 1709, 1758	1787, 1719, 1748
23	1795, 1693, 1792	1800, 1716, 1747
22	1814, 1690, 1763	1807, 1717, 1742*
21	1790, 1681*, 1725	1808, 1714, 1754
20	1796, 1737, 1743	1813, 1717, 1759
19	1835, 1725, 1760	1829, 1729, 1756
18	1846, 1730, 1772	1844, 1732, 1759

k_{long} behaves unsteadily between $OL = 18$ and $OL = 60$ (it has several local minima in that interval for each MT = 1,2,3), therefore we need to increase the optimization accuracy by taking a larger *k_max*. The computations carried out for *k_max* = 100000 show that k_{long} is not unimodal w.r.t. OL for a fixed MT, and has several local minima. It can be seen that k_{long} attains its minimum for $MT = 2$ and $OL = 25$. It is also interesting that there are two optimal values of OL if MT is equal to 1 or 3, namely $OL = 31, 40$ for $MT = 1$, and $OL = 22, 26$ for $MT = 3$.

For better illustration, the numerical results from Table 1 are presented in Figures 4–5. It can be seen that, as expected, increasing *k_max* leads to smoother cost curves. By taking still larger values of *k_max* the noise due to insufficient sampling would be further reduced, and the position of the minimum would be determined with even better accuracy. Also note that the presented results are obtained for a simplified model, thus adding to the model's complexity would adequately increase the satisfactory value of *k_max*.

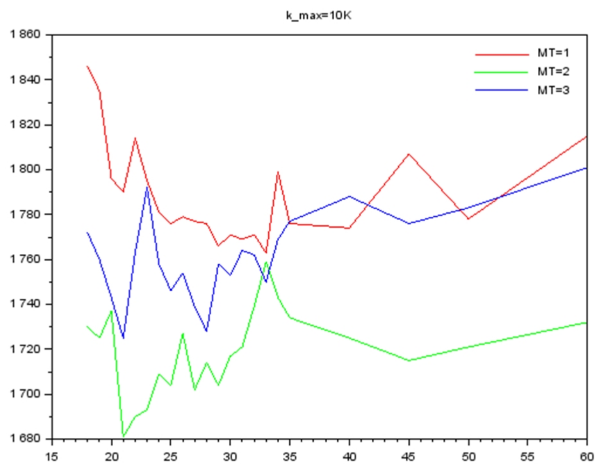

Figure 4. The graphs of k_{long} for *k_max* = 10k and three values of *MT*.

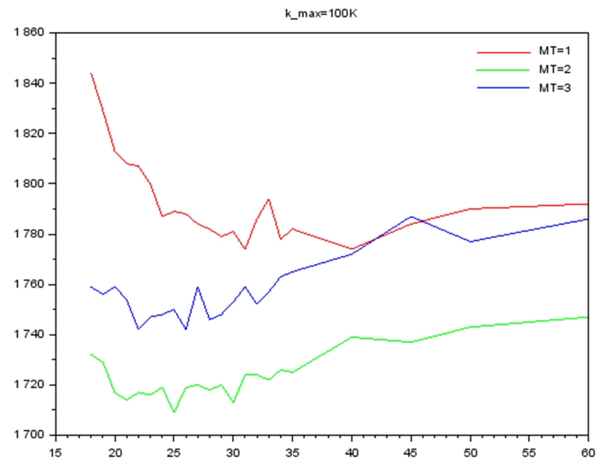

Figure 5. The graphs of k_{long} for *k_max* = 100k and three values of *MT*.

7. Conclusion

We have presented a versatile algorithm simulating the process of operation and maintenance of a complex technical system, designed for the purpose of computing its operation costs and optimizing the operation parameters. It is intended to be used where, due to the system's complexity, analytical methods are inapplicable, but the optimal values of the decision parameters can be estimated by repeated Monte Carlo simulation.

The demonstrated algorithm is similar to that presented in (George–Williams & Patelli, 2015). However, it is different, because our algorithm determines consecutive time points when components change their operational states, along with the respective components' new states. As a consequence, the time axis becomes divided into a sequence of intervals such that no component changes its state within any of them. This allows for a very detailed analysis of the system operation process. Another advantage of our method is the possibility of considering the dependence of a component's aging rate on the states of other components. Also, the possible states of a component are different in the above-cited paper, but our model can be easily modified to comply with the one considered there. Last but not least, the model considered in this paper adopts a realistic assumption that, due to limited service personnel, failed components may wait in a service queue for being repaired or replaced.

Although we define the system operating cost as the sum of costs related to individual components, we can include the system state, expressed as a given function of the components' states (known as the structure function), in the total cost calculation. Other key characteristics, such as the system's availability, can also be easily calculated, similarly as in step 7 of the algorithm.

As is well known, the main disadvantage of stochastic simulation is its high time complexity. The program implementing the simplified version of the algorithm, outlined in Section 6, executes in about 8 seconds for *k_max* = 10000, and in about 80 seconds for *k_max* = 100000 (on a machine with Intel® Core™ i5–7400 CPU). This agrees with the expectation that the algorithm's run time grows linearly with *k_max*. Admittedly, the times given above are rather short for a Monte Carlo simulation, the reason being a small number of components (5). If that number changes to 10, the run time increases to 15 seconds for *k_max* = 10000, which means that the algorithm's time complexity in relation to n is somewhat less than $O(n)$. This is because the steps 3–5 and 7–8 are realized as "for" loops with n iterations, while step 6 takes at most $Q{\cdot}MT$ basic operations (see its code), and usually $Q{\cdot}MT$ is much smaller than n. Clearly, a growing number of components and their diversity result in the corresponding increase of *k_max* necessary to determine the optimum values of the decision variables with the required accuracy. However, components of many complex systems can be grouped into sets of components with identical parameters, which decreases the number of decision variables and lowers the optimization procedure's complexity.

To sum up, an attempt has been made to construct a possibly comprehensive maintenance model of a complex technical system and develop an algorithm simulating the operation process of thus modelled system. The author's intention was to encompass a wide range of maintenance models to be found in the relevant literature, and to provide a relatively simple tool for improving the cost-effectiveness of system operation and maintenance. This aim seems to have been (at least in part) achieved, but there is still significant work to do. The future research should more broadly consider the issue of inter-component dependence, i.e. define in more detail how the aging rate of e_i is influenced by the states of components in $Z(i)$. The above issue has been studied in (Zhang & Horigome, 2001; Dukhovny & Marichal, 2012; Yang et al., 2013; Nakamura et al., 2017; Zhang & Wilson, 2017), from where some ideas can be borrowed. Also, the algorithm should take multiple failure modes into account. Then, for a given component, its failure is self-revealing or not depending on the mode of the failure. Finally, the extended model should take into consideration the influence of ambient conditions (e.g. temperature, humidity, salinity etc.) on the components' aging rates.

References

Alabdulkarim, A.A., Ball, P.D. & Tiwari, A. 2013. Applications of simulation in maintenance research. *World Journal of Modelling and Simulation* 9, 14–37.

Barlow, R.E. & Proschan, F. 1975. *Statistical Theory of Reliability and Life Testing: Probability Models*. Holt, Rinehart and Winston, New York.

Chouhan, R., Gaur, M. & Tripathi, R. 2013. A survey of preventive maintenance planning models, techniques and policies for an ageing and deteriorating production systems. *HCTL Open International Journal of Technology Innovations and Research* 3, 89–107.

Dietrich, T., Krug, S. & Zimmermann, A. 2017. A discrete event simulation and evaluation framework for multi UAV system maintenance processes. *2017 IEEE International Systems Engineering Symposium (ISSE)*, 1–6.

Dukhovny, A. & Marichal, J.–L. 2012. Reliability of systems with dependent components based on lattice polynomial description. *Stochastic Models* 28, 167–184.

George–Williams, H. & Patelli, E. 2015. Monte-Carlo based reliability/availability analysis algorithm for efficient maintenance planning. *Transactions, 23rd Conference on Structural Mechanics in Reactor Technology,* Manchester, UK.

Jardine, A.K.S. & Tsang, A.H.C. 2013. *Maintenance, Replacement, and Reliability*. CRC Press, Taylor & Francis Group.

Lu, Z., Liu J., Dong. L & Liang, X. 2019. Maintenance process simulation based maintainability evaluation by using stochastic colored petri net. *Applied Sciences* 9(16), 3262.

Lyubchenko, A.A., Kopytov E.Y., Bogdanov A.A. & Maystrenko V.A. 2020. Discrete-event simulation of operation and maintenance of telecommunication equipment using AnyLogic-based multi-state models. *Journal of Physics: Conference Series* 1441, 012046.

Münsterberg, T. 2017. *Simulation-based Evaluation of Operation and Maintenance Logistics Concepts for Offshore Wind Power Plants. Innovations for Maritime Logistics Volume 2.* Fraunhofer Verlag.

Nakagawa, T. 2008. *Advanced Reliability Models and Maintenance Policies. Springer Series in Reliability Engineering*, Springer, London.

Nakagawa, T. 2014. *Random Maintenance Policies. Springer Series in Reliability Engineering*. Springer, London.

Nakamura, S. & Qian, C.H. & Nakagawa, T. (Eds.). 2017. *Reliability Modeling with Computer and Maintenance Applications*. World Scientific, Singapore.

O'Connor, P. & Kleyner, A. 2011. *Practical Reliability Engineering*. John Wiley & Sons.

Panteleev, V.V., Kamaev, V.A. & Kizim, A.V. 2014. Developing a model of equipment maintenance and repair process at service repair company using agent-based approach. *Procedia Technology* 16, 1072–1079.

Riane F., Roux O., Basile O. & Dehombreux P. 2009. Simulation based approaches for maintenance strategies optimization. *Handbook of Maintenance Management and Engineering*, 133–153.

Rubinstein, R.Y. & Kroese, D.P. 2008. *Simulation and the Monte Carlo Method, 2nd edition*. John Wiley & Sons.

Soltanali, H., Rohani, A., Tabasizadeh, M., Abbaspour-Fard, M.H. & Parida, A. 2019. Operational reliability evaluation-based maintenance planning for automotive production line. *Quality Technology & Quantitative Management* 17(2), 186–202.

Sarkar, A., Panja, S.C. & Sarkar, B. 2011. Survey of maintenance policies for the last 50 years. *International Journal of Software Engineering & Applications* 2(3), 130–148.

Vishnu, C.R. & Regikumar, V. 2016. Reliability based maintenance strategy selection in process plants: a case study. *Procedia Technology* 25, 1080–1087.

Wang, H. 2002. A survey of maintenance policies of deteriorating systems. *European Journal of Operations Research* 139(3), 469–489.

Yang, Q., Zhang, N. & Hong, Y. 2013. Reliability analysis of repairable systems with dependent component failures under partially perfect repair. *IEEE Transactions on Reliability* 62(2), 490–498.

Zhang, T. & Horigome, M. 2001. Availability and reliability of system with dependent components and time-varying failure and repair rates. *IEEE Transactions on Reliability* 50(2), 151–158.

Zhang, X. & Wilson, A. 2017. System reliability and component importance under dependence: a copula approach. *Technometrics* 59(2), 215–224.