Reliability Engineering and System Safety 168 (2017) 171–188

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

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

Supporting group maintenance through prognostics-enhanced dynamic dependability prediction

J.I. Aizpurua [a,*], V.M. Catterson [a], Y. Papadopoulos [b], F. Chiacchio [c], D. D'Urso [c]

[a] University of Strathclyde, Department of Electronic and Electrical Engineering, 99 George Street, G1 1RD Glasgow, UK
[b] University of Hull, Department of Computer Science, Cottingham Road, Hull, UK
[c] University of Catania, Department of Industrial Engineering, Catania, Italy

ARTICLE INFO

Keywords:
Prognostics
Predictive maintenance
Diagnostics
Dynamic dependability
Maintenance grouping

ABSTRACT

Condition-based maintenance strategies adapt maintenance planning through the integration of online condition monitoring of assets. The accuracy and cost-effectiveness of these strategies can be improved by integrating prognostics predictions and grouping maintenance actions respectively. In complex industrial systems, however, effective condition-based maintenance is intricate. Such systems are comprised of repairable assets which can fail in different ways, with various effects, and typically governed by dynamics which include time-dependent and conditional events. In this context, system reliability prediction is complex and effective maintenance planning is virtually impossible prior to system deployment and hard even in the case of condition-based maintenance. Addressing these issues, this paper presents an online system maintenance method that takes into account the system dynamics. The method employs an online predictive diagnosis algorithm to distinguish between critical and non-critical assets. A prognostics-updated method for predicting the system health is then employed to yield well-informed, more accurate, condition-based suggestions for the maintenance of critical assets and for the group-based reactive repair of non-critical assets. The cost-effectiveness of the approach is discussed in a case study from the power industry.

1. Introduction

The main goal of maintenance is to achieve desirable system dependability whilst minimising cost [1]. Dependability is a term that encompasses a range of attributes which include safety, reliability, availability, and maintainability [2]. Some industries are moving away from traditional time-based or reactive maintenance regimes towards condition-based maintenance (CBM), where intervention is scheduled when monitoring data indicates asset deterioration [1].

CBM applications have explored different areas for cost-effective maintenance planning such as grouping maintenance strategies or updating maintenance models with prognostics information. Grouping maintenance actions together can reduce downtime and personnel costs through considering functionally or spatially related assets within the system [3–12]. Prognostics and health management (PHM) is an integral aspect of CBM which focuses on system degradation management with the following main groups of activities [13]:

- Anomaly detection: monitoring and detection of abnormal conditions in the system operation.
- Diagnostics: if an anomaly is detected, diagnose the cause of the fault.
- Prognostics: predict the likely future degradation of the asset and estimate its remaining useful life.
- Operation and maintenance planning: mitigate the effects of failure and reduce unnecessary planned maintenance.

PHM techniques have emerged as promising solutions for cost-effective asset management and maintenance planning [14–16]. Namely, the connection between prognostics and maintenance enables updating maintenance plans with up-to-date remaining useful life (RUL) estimations [16–18].

The RUL denotes the time distance from the current prediction time, t_p, to the end of the useful life (or failure time) of the system denoted *EOL*:

$$RUL = EOL - t_p \quad | \quad EOL > t_p \tag{1}$$

Given that remaining time after t_p is random, uncertainty representation mechanisms are needed to model RUL [19,20]. Fig. 1 shows the RUL prediction concept, where $Y = \{y_1, \ldots, y_n\}$ denotes gathered data

* Corresponding author.
E-mail address: jose.aizpurua@strath.ac.uk (J.I. Aizpurua).

http://dx.doi.org/10.1016/j.ress.2017.04.005
Available online 21 April 2017

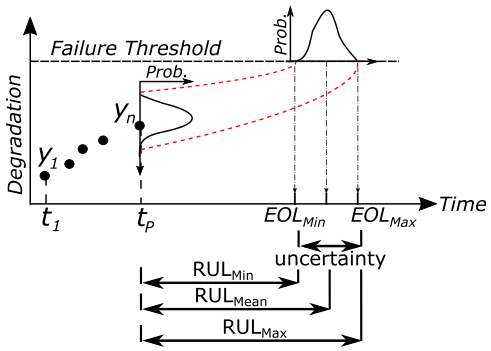

Fig. 1. Remaining useful life prediction.

samples up to the prediction point t_p. The probabilistic estimation of the health state at t_p can be performed through diagnosis methods (e.g. [21]).

Depending on the specific prognostics prediction method, the format of the RUL prediction results will be different [19,16]:

- deterministic RUL values[1] (e.g. calculated employing neural networks [22]);
- RUL values with confidence intervals $RUL \pm CI$ (e.g. estimated with hidden Markov models [23]);
- probability density function (PDF) of the RUL (e.g. derived using particle filters [14,24]).

So as to use prognostics results within CBM planning, one possibility is to parametrize prognostics prediction results [25]. For deterministic prediction results, the RUL value can be used directly assuming a constant degradation rate and confidence bounds can be used to estimate maximum and minimum boundary values [26]. As for the PDF of the RUL, the PDF can be parametrized through regression methods (e.g., Weibull regression [27,28]), or alternatively mean, maximum and minimum RUL values can be calculated [25].

Despite these advances, cost-effective CBM planning is far from trivial in complex industrial systems. Such systems are comprised of many potentially repairable assets, which can fail in different ways and with various effects. The operation of assets and the system is typically governed by dynamics which include time-dependent and conditional events and they cause complexities in the system reliability prediction and maintenance planning [29]. The use of combinatorial failure models (fault trees, reliability block diagrams) to model the failure logic of complex systems has disadvantages for maintenance planning. For instance, in a fault-tolerant system, the criticality of assets can change substantially over time [30]: in a system with two parallel redundant channels, when one fails the criticality of assets within the single remaining channel increases. Combinatorial failure models have limited ability to represent these situations. Therefore, system maintenance strategies based combinatorial failure models may also miscalculate dependability and maintenance costs.

Several dynamic dependability techniques have emerged to enable a more accurate analysis of dynamic scenarios that include state changes and sequencing of failures [31]. The application of these techniques for CBM planning would enable a more accurate health assessment estimation compared with maintenance planning methods based on combinatorial failure models. In this paper, we argue that the increasing capabilities for prognostics and maintenance strategies along with dynamic dependability models create opportunities for improved dependability estimations and system maintenance planning.

[1] Deterministic RUL values are sometimes denoted as average RUL or mean residual life [16]. We will use the term RUL to denote deterministic RUL values and we will explicitly refer to the PDF of the RUL when needed.

In that context, optimization of system maintenance remains an open research problem. Making progress in this area, in our view, requires incorporating accurate prognostic-enhanced dynamic dependability models into maintenance planning. Preliminary work on incorporating prognostics in a dynamic dependability model and informing asset-level maintenance planning has been done in [32,25,33], but this solves only part of the problem. Moving from asset to system level maintenance requires incorporating grouping criteria suited for dynamic failure logic systems. Dynamic dependability models can be used for clustering tasks based on the criticality analysis of assets. However, the connection between dynamic dependability models and potential maintenance strategies is complex because unforeseen events have effects on dependability which are hard to foresee *a priori* and cause effects on the dependability profile of the assets and system making further maintenance decisions harder. In particular, the specification of different groups of assets for maintenance at different intervals becomes hard because grouping criteria and clusters should change dynamically to optimise dependability and cost.

In this paper, our aim is to address some of the above challenges in the dynamic planning of maintenance. The main contribution of this paper is the proposal of an advanced system-level dynamic maintenance planning method building on our earlier work on prognostics-enhanced dynamic dependability models for maintenance [25,33]. The core of the proposed approach is a system-level maintenance planning algorithm which coordinates predictive diagnostics activities and asset-level prognostics information, and interacts with the dynamic dependability model.

The online predictive diagnosis algorithm classifies assets as critical or non-critical according to their importance at the system level. The system maintenance planning algorithm takes this information and interacts with the dynamic dependability model to implement grouping maintenance strategies and predict the consequence on the system health. The dynamic dependability model is updated with prognostics information, so as to yield well-informed, more accurate, condition-based suggestions for the maintenance of assets that have been identified as critical and for the group-based reactive repair of assets that have been identified as non-critical.

This paper is organized as follows. Section 2 presents related work. Section 3 introduces the prognostics-updated system maintenance approach for dynamic systems. Section 4 presents the implementation of the proposed approach for asset and system level maintenance strategies. Section 5 applies the proposed approach to a power transmission substation case study. Section 6 discusses the applicability and limitations of the method and Section 7 presents conclusions and future prospects.

2. Related work

Condition-based maintenance planning has recently gained interest as a possible method of cost-effective maintenance planning for stochastically deteriorating assets and systems [34]. Different cost-effective maintenance strategies have been proposed for asset-level condition-based maintenance focusing on specific failure modes such as fatigue crack growth [14] or pitting corrosion [35].

When designing maintenance strategies for complex systems with multiple assets, engineers need to consider stochastic, structural, and economic dependencies between assets [36]. Stochastic dependency implies that asset degradation impacts the performance of other assets, structural dependency means that maintaining an asset implies the maintenance of other assets, and economic dependency addresses the difference between group and independent maintenance actions.

For efficiency, maintenance strategies with economic dependencies must allow for the grouping of assets in different periods of maintenance. This grouping can be static or dynamic to reflect changes in operational circumstances [37]. Dynamic maintenance grouping methods can be divided into finite horizon planning (no online re-planning) and

rolling horizon planning (long term plan revised as new information becomes available). We will review techniques that deal with a rolling horizon because our work is focused on these systems. These techniques have been addressing various modelling challenges. Table 1 cites papers addressing in different ways the asset degradation model, the system failure logic structure, dependencies, and the use of prognostics.

Degradation model. The majority of maintenance models assume either Gamma [3,4,7,11,9] or Weibull [5,6,8,10] distributions for the degradation modelling of system assets. The Gamma distribution models a monotonic degradation process [38]. The Weibull distribution is a well-studied generic distribution, which integrates other distributions too (exponential, Rayleigh). Embedding predefined, well-studied, degradation processes enables the posterior analytical treatment and implementation of optimization strategies. However, as demonstrated in [16], the use of explicit prognostics information may lead to more robust maintenance strategies. Accordingly, we decouple the degradation modelling process from the maintenance planning process as in [12].

System structure. Early work has looked into series systems [3–5,8,9,12], while gradually series-parallel systems have been studied [6,10]. Recently, the influence of redundancy structures have been considered studying K-out-of-N configurations [11]. This earlier work assumes that a system failure logic can be mapped to a combinatorial Boolean model for calculation of probability. This, however, is a simplistic assumption. In practice, assets interact in a manner that leads to a more complicated failure logic. They can have several inter-dependent failure modes forming a complex network of stochastically dependent failure modes often with dynamic temporal relations among them.

Dependencies. We can see in Table 1 that stochastic dependencies have not been addressed for grouping maintenance strategies. Only Horenbeek and Pintelon [4] consider asset stochastic dependencies by including the influence of corrective repair actions on other subsystems. However, none of the reviewed works take into account the effect of operational changes in the system such as the activation of mechanisms which can trigger further reactions in the system. For multi-component systems without grouping maintenance strategies, degradation interactions have been taken into account, e.g. see [39]. Stochastic (and temporal) dependencies have been studied also through dynamic dependability models [31], but for group-based maintenance planning these dependencies have not been considered because they cause complexities in the analytic formulation.

Prognostics. Some approaches integrate RUL prediction information in the maintenance modelling approach [4,7,10,12]. Aligned with this goal, we tailor maintenance strategies with up-to-date prognostics prediction results. The integration of asset prognostics and system evaluation is done taking into account dependencies between assets [40,41].

Table 1
Maintenance grouping approaches with rolling horizon.

Ref.	Degradation model	System structure	Dependencies	Prognostics
[3]	Gamma	Series	Economic	No
[4]	Gamma	Series	Economic, stochastic, structural	Yes
[5]	Weibull	Series	Economic	No
[6]	Weibull	Series, parallel	Economic	No
[7]	Gamma	Series, parallel	Economic	Yes
[8]	Weibull	Series	Economic	No
[9]	Poisson, Gamma	Series	Economic	No
[10]	Weibull	Series, parallel	Economic	Yes
[11]	Poisson, Gamma	K-out-of-N	Economic	No
[12]	Independent	Series	Structural, economic	Yes

Grouping criteria. Following work in [37,3] all techniques displayed in Table 1 focus on minimising maintenance costs by balancing waste of useful life with increase in failure probability. For each specific approach, this optimization problem has been constrained by including specific operation conditions such as imperfect maintenance actions [4], maintenance opportunities [5], dynamically changing contexts [6], effect of maintenance actions on system reliability [7,10], limited personnel [8], age-based grouping [9] or loss of functionality [12]. From the system reliability perspective, one interesting alternative is to use importance measures to evaluate the degree to which an asset contributes to the system failure and accordingly group assets with similar measures for joint maintenance. For instance, Vu *et al.* [10] use Birnbaum importance measures for defining maintenance strategies. We focus on importance measures too, but for dynamic failure logic instead of assuming a combinatorial failure model.

All the system maintenance models in Table 1 assume combinatorial failure logic. We move beyond this assumption to address dynamic and repairable systems, taking into account economic and stochastic dependencies. In addition, we update the dynamic dependability evaluation model with prognostics prediction results. The evaluation of these systems is achieved through the integration of dynamic dependability evaluation models with alternative maintenance strategies.

Dynamic dependability models enable the modelling and probabilistic analysis of dynamic failure logic systems with stochastic and temporal dependencies (e.g., reconfigurable and fault-tolerant systems [42]). There is a range of dynamic dependability models that address these dependencies: Boolean Driven Markov Processes [43], Dynamic Fault Trees (DFT) [44], Dynamic Bayesian networks [45], Dynamic Reliability Block Diagrams [46], State-Event Fault Trees [47], Temporal Fault Trees [48], or hybrid DFT models [49] (see [31] for a comprehensive overview).

Apart from the analytic approaches displayed in Table 1, there has been work focused on the use of stochastic graphical models [45] and simulation methods for maintenance planning. For instance, Petri nets have been used to evaluate the effect of maintenance parameters on the system performance for different maintenance strategies. Zille *et al.* [50] proposed a reliability centred maintenance strategy, Andrews *et al.* [51] introduced a railway track asset management model, and Andrews and Fecarotti [52] presented an integrated Petri net and Bayesian network modelling approach to evaluate the effect of design and maintenance options on system performance. Similarly, Alrabghi and Tiwari have used discrete event simulations to analyse maintenance strategies for multi-component systems [53], and Koutras *et al.* have employed semi-Markov processes with Monte Carlo simulations to analyse maintenance strategies for complex multi-state systems [54].

In this work we employ stochastic graphical models. Specifically we focus on Stochastic Activity Networks (SAN), which is a variant of Stochastic Petri Nets [55]. We demonstrate that SAN enables the integration of dynamic dependability models [56], prognostics prediction information [25], asset-level maintenance strategies [33], and system-level dynamic grouping strategies (see Section 3).

3. Prognostics-updated system maintenance modelling for dynamic failure logic systems

Fig. 2 shows the proposed maintenance planning approach divided into offline and online activities and models. The offline assessment is divided into two parallel sequences of activities: system dependability and prognostics modelling. The *system design* is the starting point and it defines the functional operation of the system specifying how assets are arranged to perform the system function. Prognostics and dependability modelling activities follow system design and they join in the online evaluation part to update the dynamic dependability model with prognostics results.

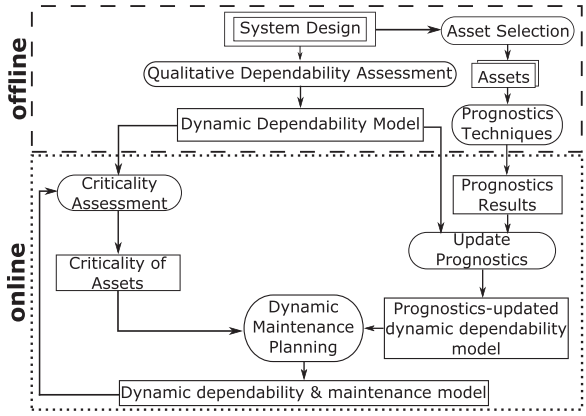

Fig. 2. Prognostics-updated system maintenance approach for dynamic failure logic systems.

The first step in the dependability modelling sequence is the *qualitative dependability assessment*. The result of this activity is a *dynamic dependability model* which defines the various dynamic sequences of asset failures that potentially cause system failure (see Section 3.2). Different types of dynamic dependability formalisms are applicable at this stage [31]. In this paper, without loss of generality, we focus on Dynamic Fault Trees (DFTs) [57] (see Section 3.4).

In parallel, the prognostics sequence starts with the *asset selection* step. Each asset may have a different degradation specification. To specify a prognostics model, degradation equations or run-to-failure data are compulsory [19]. Therefore, the *asset selection* activity for prognostics evaluation is driven by the availability of data or equations. Accordingly, different *prognostics techniques* can be considered to design a prognostics model for each asset [19].

Until this point, the process is performed at design time or offline. We then move to an online analysis process, that enables evolutionary, dynamic planning of maintenance during operation.

The online analysis process is comprised of two sequences of activities focused on regularly updating the dynamic dependability model with prognostics results and evaluating the criticality of assets.

Prognostics results depend on the nature of the prognostics model [19]. One possibility to update the failure rate and maintenance actions is to use the RUL estimation assuming a constant failure rate ($\lambda \approx 1/RUL$) [26]. For prediction models which estimate a deterministic value (including confidence intervals) the RUL values can be mapped directly. As for the probabilistic RUL predictions with density values, maximum, minimum and mean RUL values can be extracted (Fig. 1). We use this approximation to obtain the *prognostics-updated dynamic dependability model*.

In order to determine the dynamically changing degree in which an asset contributes to system failure, we implement a *criticality assessment*. There are several importance measures that can be used to weigh the criticality of an asset [58,59]. The evaluation of importance measures for dynamic repairable systems is an open issue [60,61]. This paper uses the failure criticality index ('criticality' for short) [62,30]. We use this index in an online context, and therefore the index is adapted as new information becomes available. The aim is to classify assets as critical and non-critical into two dynamic clusters with an evolving membership.

Based on the criticality and prognostics information, *dynamic maintenance* decisions are planned and represented on a prognostics-enhanced *dynamic dependability and maintenance model*. This model is connected back with the *criticality assessment* to continually monitor the health of assets.

A number of techniques have been proposed in the literature for the offline activities showed in Fig. 2 (e.g., see [19,31,63]). Therefore, in this paper we will focus on the online activities. For the sake of comparison, we will also examine asset-level maintenance strategies [33] which use the same modelling process as in Fig. 2, without considering the criticality assessment. The *dynamic dependability and maintenance model* could be quantified using analytic formulations or simulation. In the next subsection we will justify our choice.

3.1. Analytic formulations versus simulation

The failure logic of complex dynamic systems can be expressed in minimal cut-sequence sets (MCSQ), i.e. the dynamic counterpart of minimal cut-sets [64].

Merle *et al.* introduced an analytic formulation for non-repairable dynamic systems with stochastic and temporal dependencies [65]. The canonical form of the system failure expression, denoted top-event (TE), is the disjoint sum of all its n independent MCSQs [66]:

$$TE = \sum_{i=1}^{n} MCSQ_i \tag{2}$$

Each $MCSQ_i$ defines the temporal combination of basic events (BE) that cause the TE occurrence. The length of the MCSQ depends on the minimal number of BEs that cause TE occurrence. Assuming that k events cause the TE:

$$MCSQ_i = BE_1 \ \mathbf{OP_1} \ BE_2 \ \mathbf{OP_2} \ \dots \ BE_k \tag{3}$$

where $\mathbf{OP_i}$ is the operator that defines temporal logic, e.g. $\mathbf{OP_i} = \{SAND, PAND, POR\}$ [64]:

- $MCSQ_i = BE_1$ **SAND** BE_2 : BE_1 and BE_2 occur simultaneously;
- $MCSQ_i = BE_1$ **PAND** BE_2 : BE_1 occurs before BE_2, and BE_2 has to occur;
- $MCSQ_i = BE_1$ **POR** BE_2: BE_1 occurs before BE_2, but BE_2 does not need to occur;

These operators can also be combined with classical combinatorial logic operators such as **AND**, **OR**, etc.

In order to analyse the failure expression, first it is necessary to remove redundant MCSQ terms taking into account the priority of dynamic operators [64]. The probabilistic failure expression can then be obtained by applying the inclusion-exclusion principle [67]. Finally, the corresponding probabilistic formula should be applied to each MCSQ term and operator separately [65].

The above algebraic framework can be used for dependability evaluation of systems with non-repairable assets. However, the complexity of the mathematical formulation grows with the system size, and for dynamic systems with repairable assets, the logic and probabilistic formulas cannot be captured by the above semantics.

A feasible alternative is to use simulation-based dynamic dependability techniques. The focus of these approaches has been centred on the quantification of the system failure probability through Monte Carlo simulations using reactive repair strategies, e.g. [44]. In this work we will use simulation-based techniques for the implementation of *dynamic dependability and maintenance model*.

In the next subsection we define the *dynamic dependability and maintenance model*. In Section 4 we explain the representation of asset and system maintenance strategies.

3.2. Dynamic dependability and maintenance model

Fig. 3 shows the proposed online maintenance modelling approach expanding the online activities in Fig. 2.

An informed system maintenance strategy must be based on an overall picture of the systems health which is established through a set of activities and models:

- *Dynamic dependability and maintenance model*: a probabilistic model that integrates prognostics and maintenance decisions and quan-

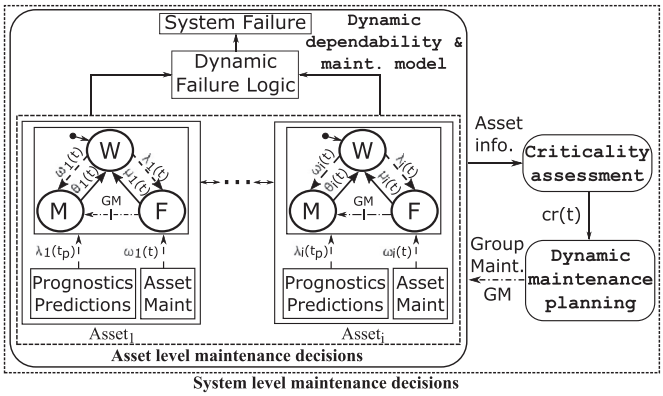

Fig. 3. Proposed online maintenance approach.

tifies failure probabilities first at asset level and then at system level.

- *Criticality assessment*: a system model, which identifies the degree of contribution of each asset to the failure of the system.
- *Dynamic maintenance planning*: maintenance decisions derived from criticality assessment. These define group maintenance and their effects redefine the future health of assets and the next iteration of the process.

Each asset in the dynamic dependability and maintenance model is modelled with three states. The transition from working (W) to failed (F) state is governed by the failure rate $\lambda(t)$. This failure rate can be updated with up-to-date degradation prediction information coming from prognostics models [26,25,33]:

$$\lambda(t) = \begin{cases} \lambda_0 & 0 < t < T_{p_k} \\ \lambda_k & t = T_{p_k} \end{cases} \quad (4)$$

where λ_0 is the initial failure rate estimate typically taken from a reliability database, T_{p_k} is the k-th prediction time instant, and λ_k is the failure rate at k-th prediction time instant. The number of prediction time instants depends on the specific prognostics application (and available data) varying from 1 up to P prediction instants $k = \{1, \ldots, P\}$.

Prognostics prediction times depend on the asset and the application context. In an online monitoring context, RUL predictions can be performed continuously as new data is gathered from the asset under study. However, early RUL predictions may not be very accurate, whereas as the system approaches its end of life, RUL predictions will become more accurate (see [68] for an analysis of the effect of prognostics prediction errors on maintenance planning). In this case, a trade-off between accuracy and timeliness needs to be made. In contrast, in an offline application context, the data is not sampled continuously but at regular time intervals, e.g. transformer dissolved gas data may be sampled quarterly or six times a year [69]. Therefore the prediction time instants in this case will be determined by the availability of monitoring data. As the health of the asset deteriorates (and RUL decreases) the sampling regime may be increased to closely monitor the aging of the asset and perform more frequent prognostics predictions.

Once the asset fails, there are two options depending on the maintenance strategy. If the maintenance is implemented at the asset level, it is repaired with a repair rate $\mu(t)$, which models the mean time to repair from failed to working state (see Section 4.1). However, for system maintenance strategies, if the asset is not critical, it is possible to leave the asset in the failed state until the maintenance planning diagnoses a critical asset triggering a group maintenance signal *GM* (see Section 4.2).

It is possible to avoid the transition to the failed state by performing maintenance prior to the asset failure. The transition to the maintenance state (M) depends on the maintenance strategy and it is determined by the transition rate $\omega(t)$. From the maintenance state, the asset needs a time interval defined by $\theta(t)$ to return back to the working state. Asset maintenance decisions are applied independently to each asset i ($\forall asset_i \in system : 1 \leq asset_i \leq N$) with an individual maintenance rate ω_i. Maintenance decisions at the system-level apply only to critical assets because the rest of non-critical assets are allowed to fail and they are repaired through group maintenance strategies.

The main assumptions adopted in this paper are twofold: assets are in as bad as old state after repair (as in other stochastic models, e.g. [51]), and there is availability of unlimited maintenance resources. We plan to analyse imperfect and resource constrained maintenance strategies in future work (see Section 7).

The online maintenance approach is implemented through the SAN formalism introduced in the following subsection. After that, in Section 3.4, we will introduce the *Dynamic Failure Logic* block in Fig. 3 (or *Dynamic Dependability Model* in Fig. 2) using SAN modelling mechanisms.

3.3. Preliminaries on stochastic activity networks

The Stochastic Activity Networks (SAN) formalism was first introduced in the mid-1980s [70] and it has been used for many different applications. For the sake of readability and simplicity we will introduce the main SAN modelling constructs semi-formally in this subsection. For a full formal definition of SAN refer to [55].

SAN extends stochastic Petri Nets generalizing the stochastic relationships and adding mechanisms to construct hierarchical models. Fig. 4 shows SAN modelling primitives [55].

Places represent the state of the modelled system. Each place contains a certain number of tokens defining the marking of the place. A *standard* place contains an integer number of tokens, whereas *extended* places contain data types other than integers (e.g., floats, array). We will denote the marking function of the place x as m(x), e.g., m(x)=1 means that the place x has a marking equal to 1.

There are two types of *activities*: *instantaneous* which complete in negligible amount of time, and *timed* whose duration has an effect on the system performance and their completion time can be a constant or a random value. When it is a random value, it is ruled by a probability distribution function defining the time to fire the activity.

Activities fire based on the conditions defined over the marking of the net and their effect is to modify the marking of the places. The completion of an activity of any kind is enabled by a particular marking of a set of places. The presence of at least one token in each input place enables the firing of the activity removing the token from its input place(s) and placing them in the output place(s).

Each activity has a *reactivation function* that defines when the activity is aborted and a new activity time is immediately obtained from the activity time distribution. The reactivation function provides a mechanism for restarting activities that have been activated, either with the same or a different distribution. To this end it is necessary that both the reactivation predicate holds for the new marking and for the marking in which the activity was originally activated; and the activity remains enabled.

Another way to enable a certain activity consists of *input gates* and *output gates*. Input and output gates make the SAN formalism general and powerful enough to model complex real situations. They determine the marking of the net based on user-defined C++ rules.

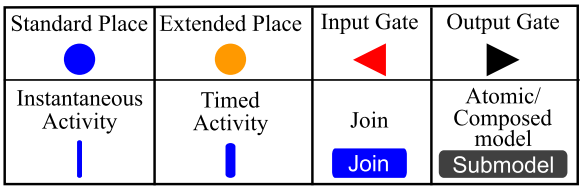

Fig. 4. Notation of SAN elements.

Input Gates (IG) control the enabling of activities and define the marking changes that will occur when an activity completes. A set of places is connected to the input gate and the input gate is connected to an activity. A Boolean condition (or guard) enables the activity connected to the gate and a function determines the effect of the activity completion on the marking of the places connected to the gate. *Output Gates (OG)* specify the effect of activity completion on the marking of the places connected to the output gate. An output function defines the marking changes that occur when the activity completes.

The performance measurements are carried out through *reward functions* defined over the designed model. Reward functions are evaluated as the expected value of the reward function and they are defined based on:

- the marking of the net (*state reward function*), e.g. quantification of the probability for being in a specific place;
- completion of activities (*impulse reward function*), e.g. count the number of times an activity triggers within a time interval.

In order to alleviate substantially the state explosion problem SAN makes use of reduced base models [71]. This concept enables the implementation of join operators and hierarchical modelling of complex systems.

Fig. 5 shows a simple repairable asset example. In this case the SAN places are initialized to working state $< \mathrm{m(W)}, \mathrm{m(F)} >=< 1, 0 >$. The token will move from `W` to the `F` place according to the distribution determined by `fault` timed activity. The time to failure will be calculated with the parameters of the `fault` activity and after the time to failure has elapsed the system will move to the failed state $< 0, 1 >$. After moving to the failure state the time to repair will be calculated from the `repair` timed distribution and the token will move from `F` to `W` place after the calculated time to repair has elapsed.

In this paper we focus on Monte Carlo simulations for the quantification of different probabilities. If we want to evaluate the failure probability or availability we can use the reward functions indicated in Fig. 5 with `F_Rew` and `W_Rew` reward variables respectively. These statements are evaluated for a large number of Monte Carlo trials and the expected value of these random variables evaluated at different time instants will give the failure probability and availability indicators.

Formally, if we want to evaluate the probability of a generic place `x`, at time instant t, first we define the reward function, $r_x(t)$, as follows:

$$r_x(t) = \begin{cases} r_x(t) + 1 & if \quad m(\mathrm{x}) = 1 \\ r_x(t) & if \quad m(\mathrm{x}) = 0 \end{cases} \tag{5}$$

Note that the marking of the place `x` will change according to the SAN atomic logic throughout the lifetime of the system. Besides, for different Monte Carlo trials, the transition times and marking values will be different. If we perform N Monte Carlo trials, the expected value of the reward function of the place, r_x, (probability for being in place `x` at time t), is calculated as follows:

$$\hat{p}_x(t) = \frac{1}{N}\Sigma_{i=1}^{N} r_x^i(t), x \in E \tag{6}$$

where E denotes the set of places in the system e.g., $E = \{\mathrm{W}, \mathrm{F}\}$ in Fig. 5.

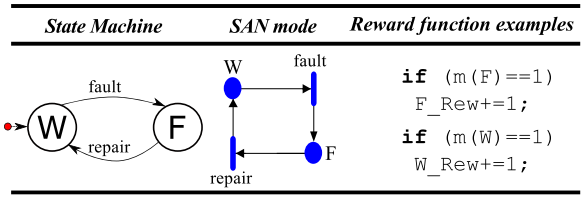

Fig. 5. Repairable asset example in SAN.

The inverse transform sampling method [72] extracts the stochastic occurrence times of timed activities using Monte Carlo simulations. Let *CDF* be a cumulative distribution function, r be a random variable drawn from the uniform distribution $r \sim U([0, 1])$, and *TTF* the time to fire the activity. Then, the inverse sampling method applies the relation $F^{-1}(r) = TTF$ to draw the time to fire according to the CDF.

The same concept as in Eq. (6) applies to the calculation of the number of actions. In this case impulse reward functions are defined over the activity of interest. Let us assume that we define for the activity `a` the impulse reward, r_a, which is dependent on the activity completion time or time to fire, TTF_a:

$$r_a = \begin{cases} 1 & if \quad t = TTF_a \\ 0 & otherwise \end{cases} \tag{7}$$

For each Monte Carlo trial the TTF_a will vary according to the random number r. We can evaluate the number of times an activity `a` fires during a time interval $\Delta t = T_m - t_0$ after performing N simulations as follows:

$$\hat{n}_a(\Delta t) = \frac{1}{N}\Sigma_{i=1}^{N}\Sigma_{\Delta t=t_0}^{T_m} r_a^i(\Delta t), \mathrm{a} \in A \tag{8}$$

where A denotes the set of activities in the model, e.g. in Fig. 5 $A = \{\mathtt{fault}, \mathtt{repair}\}$.

Note that the required number of iterations N in Eqs. (6) and (8) depend on the required confidence level for the reward variables [71]. In this work all the experiments have been performed with a confidence level of 0.99 and a minimum confidence interval of 1e-6. Accordingly, the simulations are halted when all the failure probability estimations satisfy the confidence interval constraints (worst case $N \approx$3e6) .

The SAN models which include the specified SAN elements are modelled in a SAN *atomic model* (see Fig. 7 *Reusable Block* column). The *join operator* links through a compositional tree structure different SAN models in a unique *composed model*. As shown in Fig. 8 places of different atomic models are joined to represent interactions between different SAN models. It is possible to link atomic models, composed models, or combinations thereof. In the tree structure, the composed and atomic SAN models are linked through join operators using shared places between the composed and atomic SAN models. Thus, the analyst can focus on specific characteristics of the asset behaviour through fit-for-purpose atomic and composed models and later join independently validated models to obtain a more complex composed system model.

The SAN model in Fig. 5 can be reused to link the places with other assets. For instance, if we want to initiate another (timed or immediate) activity in another asset when the failure of the model in Fig. 5 occurs, we can join the `F` place with the destination SAN model through SAN join mechanism (see Fig. 8 for an example). For the composed system the quantification of system probabilities is exactly the same as for the simple system in Fig. 5. In this case the reward function in Eq. (5) must be defined in terms of the system failure place (`TE` in Fig. 8).

3.4. Dynamic failure logic modelling in SAN

The dynamic failure logic modelling process is the same for asset and system maintenance strategies. The main difference between these strategies is on the asset-level modelling (see Fig. 3 and Section 4).

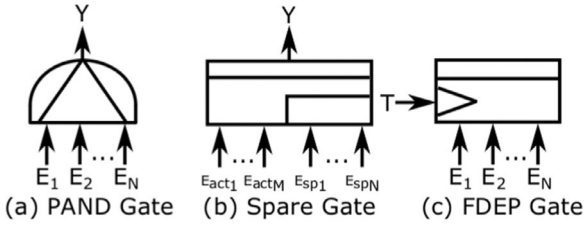

Fig. 6. Subset of Dynamic Fault Tree gates.

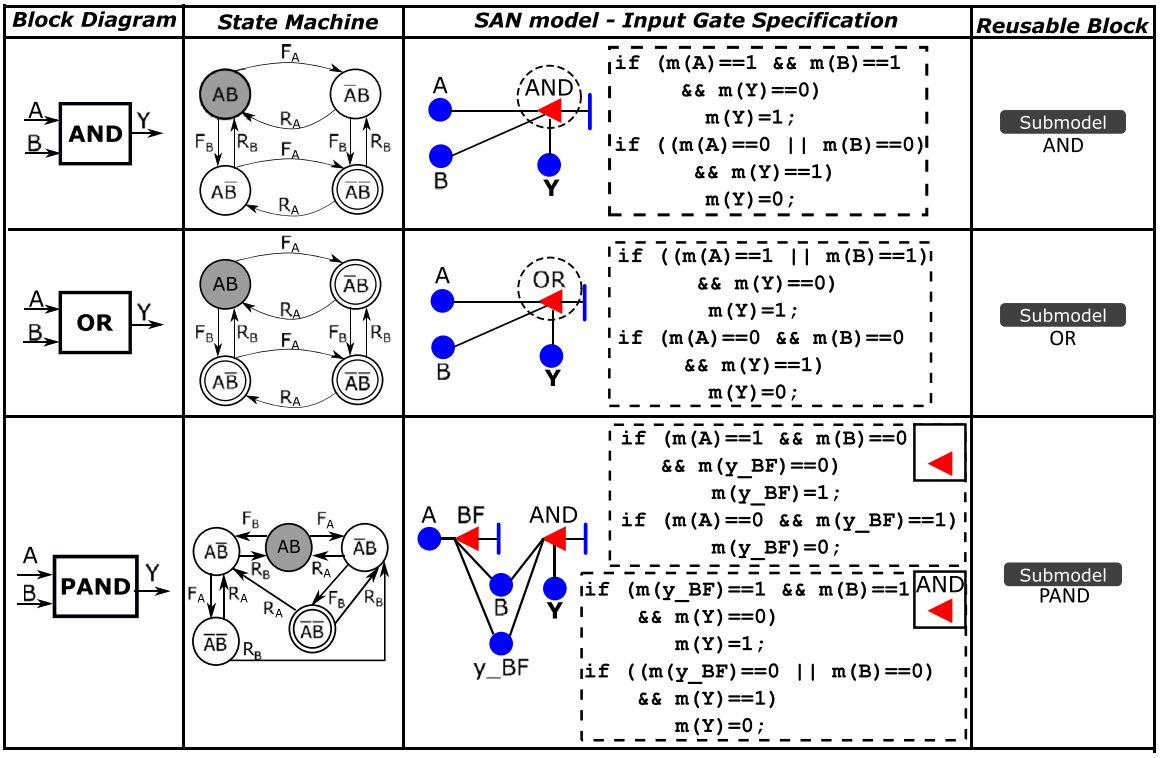

Fig. 7. Specification of repairable DFT gates in SAN.

The dynamic failure logic is formalized using a dynamic dependability model and in this case we use repairable DFTs [57]. The repairable DFT model links low-level repairable basic events (i.e. assets in Fig. 3) with the system-level failure events (top-event) through dynamic gates. Repairable DFT gates are defined as follows [44]:

(a) PAND: $Y = PAND(E_1, \dots, E_N)$; Y is *true* iff all events $\{E_1, \dots, E_N\}$ are true and they occur in order: $E_1 \lhd \cdots \lhd E_N$; otherwise is *false* (Fig. 6a).
(b) Spare: $Y = SP(E_{ac_1}, \dots, E_{ac_M}, E_{sp_1}, \dots, E_{sp_N})$; Y is *true* iff all active events $\{E_{ac_1}, \dots, E_{ac_M}\}$ and all spare events $\{E_{sp_1}, \dots, E_{sp_N}\}$ have failed, otherwise is *false*. Its inputs can be in standby, working or failed state (cf. Fig. 6b).
(c) FDEP: $[E_1, \dots, E_N] = FDEP(T)$; $\{E_1, \dots, E_N\}$ is *true* if the trigger event T occurs or they fail by themselves; otherwise is *false* (cf. Fig. 6c).

A fourth gate called sequence enforcing gate has also been defined in [57]. However note that this gate can be implemented using a spare gate [73] and also note that the FDEP gate can be modelled using OR gates [27].

In this paper we will focus on PAND and SPARE gates. Fig. 7 shows the specification of repairable DFT gates in SAN using state machines and their corresponding SAN model [56]. Note that we have not included the SPARE gate in Fig. 7 because this is modelled as an AND gate, including basic events which can be in standby states (e.g. see Fig. 10) and implementing the activation logic between basic events. In the state machine the initial state is highlighted in grey, failure states are identified with doubled circles, and F_x and R_x indicate failure and repair events of x. The resultant reusable blocks are used to link with basic events and other gates. Note also that the DFT gates in Fig. 7 are directly extendible to gates with N input events.

So as to evaluate the output of a gate it is necessary to link the gates in Fig. 7 with basic events (i.e., assets). To this end we synthesize all the expressions in the DFT model using the join operator in SAN [56]. Fig. 8 shows a synthesis example assuming assets with independent reactive repair strategies as also shown in Fig. 5.

At the lowest level first we model reactive repair strategies for each asset in the system (`Asset1`, `Asset2`, `Asset3`) with independent repair rates. Then we synthesize the PAND operation by linking the `A` and `B` places of the PAND gate in Fig. 7 with `KO_Asset1` and `KO_Asset2` places respectively. To this end, we use the join operator of SAN. This will result in the creation of the `PAND_A1A2` reusable block. Then we link the output place of this submodel (`PAND_Out`) and the `KO` place of the Asset3 with the inputs of the OR gate in Fig. 7.

The system-level failure probability is then quantified by monitoring the `Y` place of the OR gate (renamed to `TE` in Fig. 8) which indicates the occurrence logic of this gate (see Fig. 7) and the failure of the system in Fig. 8. The marking of a place is monitored through reward variables as explained in Section 3.3. In this case we replace the place `x` in Eq. (5) with the place `TE` of Fig. 8. The marking of this place will be dependent on the underlying failure logic (`PAND` gate) and asset (`Asset1`, `Asset2`, `Asset3`) models. The asset-level failure probability is quantified by monitoring the failure place of the asset under study, e.g. for `Asset3` we need to define a reward variable by replacing `x` with `KO_Asset3` in Eq. (5) (see also Fig. 5).

Fig. 9 shows the system level failure probability quantified with RAATSS [44] and SAN models with constant hypothetical failure and repair parameters (λ_1=0.1 years^{-1}, μ_1=2 years^{-1}, λ_2=0.05 years^{-1}, μ_2=1 years^{-1}, λ_3=0.5 years^{-1}, μ_3=3 years^{-1}). RAATSS implements repairable DFT models for assets with reactive repair strategies based on Adaptive Transition Systems [44].

Results shown in Fig. 9 confirm the validity of the DFT gates proposed in Fig. 7 and the synthesis process shown in Fig. 8. These concepts can be generalized for more complex systems and it is possible to include reconfiguration activities so as to activate standby elements by setting their standby place when required [42], as in the `Activate` place in Fig. 10 . We use the same modelling process throughout the paper to evaluate the system failure probabilities at asset and system levels.

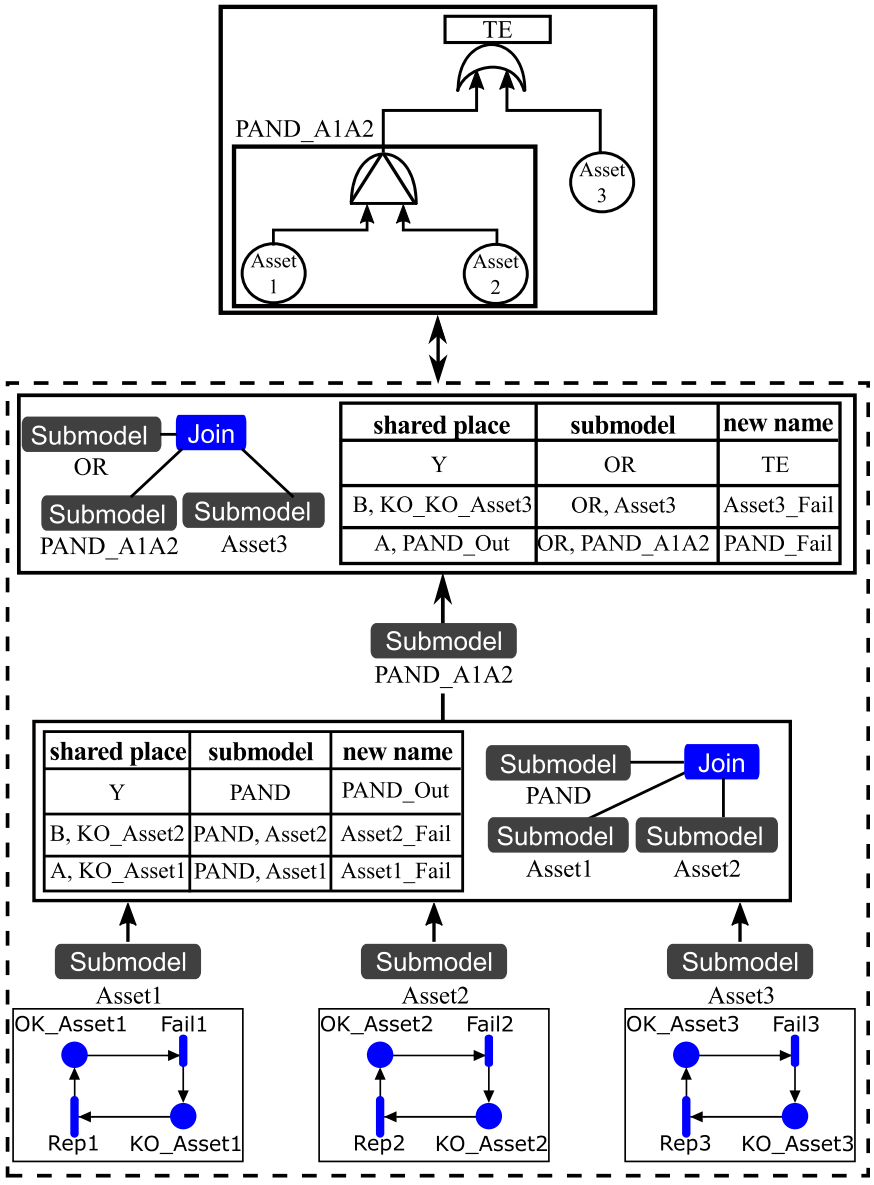

Fig. 8. DFT synthesis example.

4. Multi-level maintenance modelling for dynamic failure logic systems

In this section we will present detailed maintenance modelling concepts to represent condition-based asset and system level maintenance strategies based on the approach shown in Fig. 3.

4.1. Asset-level maintenance strategies

Condition-based maintenance strategies can use prognostics predictions to adapt the maintenance schedule and extend the useful life of the asset. As shown in Fig. 3, the maintenance instant $\omega(t)$ is defined as [74]:

$$\omega(t) = \frac{1}{RUL(T_p) - SF} \tag{9}$$

where $RUL(T_p)$ is the remaining useful life at prediction time T_p and SF is a safety factor which integrates the time required to trigger maintenance and uncertainties associated with the RUL prediction.

The selection of the SF for timely maintenance decision making depends on two factors. The first is the time needed to trigger maintenance, and this is asset and failure specific. For instance, offshore wind turbines require travel time which is dependent on the weather and sea conditions [75]. The second factor is the uncertainty associated with RUL prediction. Assuming deterministic RUL predictions with the confidence interval CI, the final predictions will be $RUL \pm CI$. In order to avoid failure occurrences, the most conservative SF may be selected assuming the worst case scenario, i.e. the sum of the confidence interval and maximum time needed to trigger the asset maintenance.

Accordingly, when a new prognostics prediction is performed at any prediction time instant T_p, the condition-based maintenance schedule can be adapted with up-to-date condition information of the asset:

$$\omega(t) = \begin{cases} \omega_0 & 0 < t < T_{p_k} \\ \omega_k & t = T_{p_k} \end{cases} \tag{10}$$

where ω_0 is the initial maintenance rate, T_{p_k} is the k-th prediction time instant, and ω_k is the maintenance rate at k-th prediction time instant as defined in Eq. (9).

Fig. 10 shows the asset-level condition-based maintenance model implemented in SAN divided into `Asset`, `Control`, and `Maintenance` blocks.

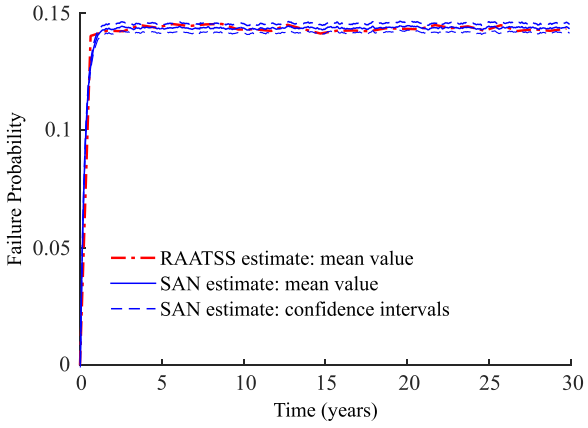

Fig. 9. Failure probability assessment of the system in Fig. 8.

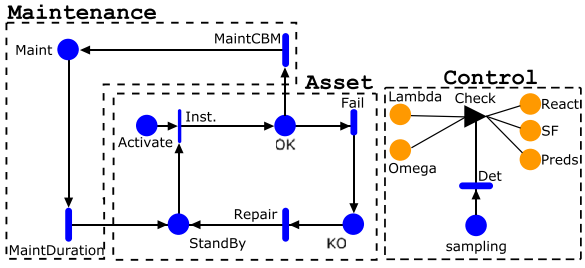

Fig. 10. Asset CBM model for asset-level maintenance.

The `Asset` block integrates `OK`, `KO`, `StandBy` and `Activate` places with `Fail` and `Repair` timed activities and an instantaneous activity `Inst`, which links `StandBy` and `Activate` places. The `Fail` activity is updated with prognostics prediction results stored in the `Preds` extended place (in the `Control` block). When the asset is repaired, it remains in the `StandBy` state until it receives an `Activate` signal from the reconfiguration mechanism.

The `Control` block implements the system update actions through the `Check` OG. This block is executed deterministically, every `Det` time period (Δt simulation timestep) controlling the marking of the `sampling` place. The `Fail` and `MaintCBM` activities in the `Asset` block are reactivated and updated with new transition rates via the `Lambda` and `Omega` extended places respectively ($\lambda(t)$ and $\omega(t)$ in Fig. 3). The `SF` place stores the safety factor to calculate $\omega(t)$ (cf. Eq. (9)). The update logic embedded in the `Check` OG monitors the prediction time instants stored in `Preds`, updates `Lambda` and `Omega` places, and enables their reactivation through the `React` extended place. Algorithm 1 defines the prognostics update process implemented in the `Check` OG.

Algorithm 1. Update of failure and maintenance rates in SAN using prognostics prediction information.

for $t = 0 : \Delta t : T_m$ **do** ▷ throughout the mission time
  if t=$t_{p_{k,i}}$, $\forall asset_i \in system : 1 \le i \le N$, k={1,…,P} **then** ▷ prognostics prediction instant t_{p_k} for the i-th asset
    $\lambda_i(t) = 1/RUL_{k,i}$ ▷ update failure rate with prognostics at k-th prediction instant for the i-th asset
    $\omega_i(t) = 1/(RUL_{k,i} - SF_i)$ ▷ update CBM rate with Eq. (9), assuming exponential distribution
    $resample(asset_i, t_{p_k}, \lambda_i(t))$ ▷ resample the failure distribution with the new $\lambda_i(t)$ in SAN
    $resample(asset_i, t_{p_k}, \omega_i(t))$ ▷ resample the CBM distribution with the new $\omega_i(t)$ in SAN
  end if
end for

In Algorithm 1, if the time equals the prediction time t_{p_k} (where $k = \{1, \dots, P\}$ and P denotes all the prediction instants), of any of the prognostics models of any asset ($\forall asset_i \in system : 1 \le i \le N$), we update the corresponding asset's failure rate and condition-based maintenance rate in Eq. (9) with the newly obtained RUL prediction. The resampling property is already implemented in SAN through reactivation functions (see Section 3.3). `Line 5` updates the `Fail` activity and `line 6` updates the `MaintCBM` activity obtaining new activity times drawn from the exponential distribution with the prognostics-updated parameters.

Fig. 11. Online maintenance planning sequence.

The `Maintenance` block implements planned shutdown events. The `MaintCBM` timed activity models the $\omega(t)$ event in Fig. 3 and Eq. (9). `MaintCBM` has a reactivation logic to update the transition rate to the `Maint` place according to prognostics prediction results and prediction instants stored in the `Preds` place. The `MaintDuration` activity models the $\theta(t)$ event in Fig. 3.

Accordingly, when modelling asset maintenance strategies, each asset in the DFT model (e.g., see Fig. 8) is modelled using an instance of the model shown in Fig. 10 with the corresponding asset-specific parameters.

4.2. System-level maintenance strategies

The system group-based maintenance approach uses all the activities and models in Fig. 3. Fig. 11 shows the high-level operation sequence comprised of four sequential steps:

(a) *Dynamic dependability and maintenance model* informs *maintenance planning* about the health status of assets and failure time instants throughout the lifetime (see Section 4.2.1).
(b) *Dynamic maintenance planning* calls *criticality assessment* to assess operative assets' criticality.
(c) *Criticality assessment* diagnoses if the asset under study is critical (see Section 4.2.2).
(d) The *dynamic maintenance planning* implements maintenance actions depending on the criticality of assets (see Section 4.2.3).

The criticality assessment is implemented through C++ libraries which are connected with the SAN model. In this way the *criticality assessment* can be called as a conventional function with its input parameters from any SAN model (see Section 4.2.2).

4.2.1. Dynamic dependability and maintenance model

Assets with system-level maintenance are modelled in SAN as shown in Fig. 12 divided into `Asset`, `Control`, `Maintenance`, and `System` blocks. The `Asset`, `Control`, and `Maintenance` blocks perform in the same manner as for asset-level maintenance models (see Section 4.1).

The `System` block implements waiting options according to the system maintenance strategy. That is, `GroupRep` can trigger group repair actions when `WaitCritical` is activated and `MaintCBM`

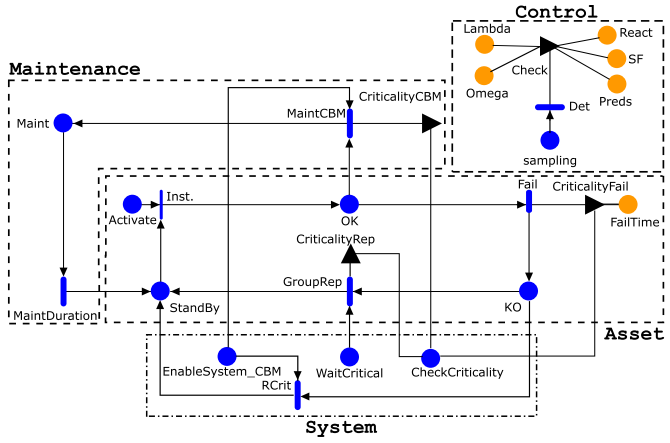

Fig. 12. Asset CBM model for system-level maintenance.

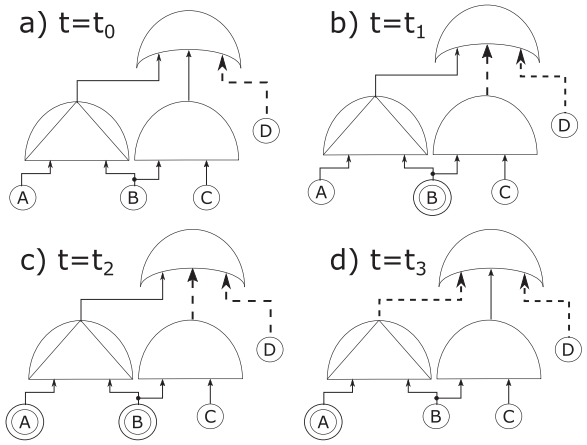

Fig. 13. Criticality change example.

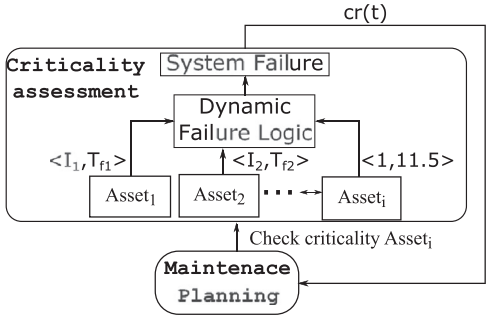

Fig. 14. Criticality assessment.

can trigger CBM actions when `EnableSystem_CBM` is activated. Besides, the `System` block triggers the criticality analysis by setting the `CheckCriticality` place and calling the *criticality assessment* when there is a failure occurrence (through the `CriticalityFail` OG), repair activity (through the `CriticalityRep` OG), or maintenance action (through the `CriticalityCBM` OG). The `RCrit` activity triggers repair actions of critical assets in case the CBM action is missed, with the same parameters as `GroupRep` activity.

The *Dynamic Failure Logic* block in the *Dynamic Dependability and Maintenance model* (Fig. 3) implements in SAN the failure logic of the DFT gates and links failed states of the assets with the failure logic (see Section 3.4). In parallel, after every failure, repair or maintenance occurrence of any asset, the marking of `CheckCriticality` is set, and the health state and failure time information of each asset (`KO`, `FailTime` places in Fig. 12) are sent to the *criticality assessment* (see Section 4.2.2).

After diagnosing if there is a critical asset, maintenance decisions are adopted by the *dynamic maintenance planning*. This activity changes the marking of the `WaitCritical` and `EnableSystem_CBM` places depending on the specific situation, and accordingly it triggers group-based and critical assets maintenance actions, respectively.

The `MaintCBM` activity is a stochastic event and the transition time is extracted according to the inverse transform method (Section 3.3). There may be cases that the `Fail` activity fires faster than `MaintCBM` activity, and the asset transits to the `KO` state. The `RCrit` activity repairs individual critical assets that miss the transition to the `Maint` place. If we change the `MaintCBM` to a instantaneous activity, then the system failure probability will be always zero because preventively we move the asset to a safe state as soon as it is diagnosed as critical. We have used this strategy to validate the correctness of the diagnostics function.

4.2.2. Criticality assessment

The goal of this module is the identification of assets that cause the system failure. The criticality of assets changes throughout the lifetime of the system due to the simultaneous occurrence of failure, repair and maintenance events. For instance, in the example shown in Fig. 13 , the criticality, $cr(t)$, changes at different time instants as follows (critical paths are indicated with a dashed line and failed events are indicated with a double circle):

a) $cr(t_0) = \{D\}$
b) Asset B fails with $t_0 < t < t_1$; $cr(t_1) = \{C, D\}$
c) Asset A fails with $t_1 < t < t_2$; $cr(t_2) = \{C, D\}$
d) Asset B is repaired with $t_2 < t < t_3$; $cr(t_3) = \{B, D\}$

The criticality of dynamic repairable systems can be evaluated with indicator functions and failure timestamps of all the basic events of the system. The indicator function I_i takes values in the set $S = \{0, 1\}$ and indicates the actual health state of the basic event i. That is, I_i=1 if asset i is in the failed state, I_i=0 if it is working. The failure timestamp t_{f_i} indicates the failure occurrence time of the basic event i: $t_{f_i} > 0$ if failure occurred, t_{f_i}=0 if working or repaired.

Accordingly, for the implementation of the criticality assessment we use a parallel health monitoring DFT model which implements the same dynamic failure logic as the *dynamic dependability and maintenance model*, but it is only used for diagnosis purposes and not probabilistic quantification. It takes health state indicators and failure time instants of all the assets and evaluates their criticality.

If we want to foresee which are the critical assets that can cause the system failure occurrence, one possibility is to change the indicator function of each operative asset to the failed state and evaluate the system failure expression, diagnosing the system-level effect for each potential asset failure. Every time there is a change of health state in any asset (failure, maintenance or repair), it is necessary to check if the failure of an operative asset would cause the system failure, given the specific operation conditions. Fig. 14 shows the criticality analysis assuming that we are interested in evaluating the criticality of the Asset_i at 11.5 time units. Note that the asset models in the criticality assessment are comprised of health state indicators and the failure timestamps.

This concept can be integrated within the SAN modelling formalism as a standalone C++ library which can be used at runtime during the SAN model simulation. Accordingly, we have created the corresponding libraries to implement and evaluate the qualitative outcome (i.e., criticality) of DFT repairable gates as shown in Table 2. The behaviours of repairable DFT gates are defined according to their input events (in_1, in_2), which can be extended to an arbitrary number of input events.

Instantiating and linking the inputs and outputs of the gates in Table 2 according to the system failure logic enables the qualitative criticality assessment of the effect of a low-level basic event failure on the system failure occurrence. For instance, the criticality assessment

Table 2
C++ libraries for the qualitative DFT assessment.

Function	Description
$\{I_{out}, t_{f_{out}}\}$=**AND**($[I_{in_1}, t_{f_1}], [I_{in_2}, t_{f_2}]$): $S_{in_1} \times \mathrm{IR}^+_{in_1} \times S_{in_2} \times \mathrm{IR}^+_{in_2} \rightarrow S_{out} \times \mathrm{IR}^+_{out}$	I_{out} = 1 iff I_{in_1}=1 and I_{in_2}=1; $t_{f_{out}}$=max(t_{f_1}, t_{f_2})
$\{I_{out}, t_{f_{out}}\}$=**OR**($[I_{in_1}, t_{f_1}], [I_{in_2}, t_{f_2}]$): $S_{in_1} \times \mathrm{IR}^+_{in_1} \times S_{in_2} \times \mathrm{IR}^+_{in_2} \rightarrow S_{out} \times \mathrm{IR}^+_{out}$	I_{out} = 1 iff I_{in_1}=1 or I_{in_2}=1; $t_{f_{out}}$=min(t_{f_1}, t_{f_2})
$\{I_{out}, t_{f_{out}}\}$=**PAND**($[I_{in_1}, t_{f_1}], [I_{in_2}, t_{f_2}]$): $S_{in_1} \times \mathrm{IR}^+_{in_1} \times S_{in_2} \times \mathrm{IR}^+_{in_2} \rightarrow S_{out} \times \mathrm{IR}^+_{\mathrm{out}}$	I_{out} = 1 iff I_{in_1}=1 and I_{in_2}=1 and $t_{f_1} < t_{f_2}$; $t_{f_{out}}$=max(t_{f_1}, t_{f_2})

model in Fig. 13 is instantiated as follows:

$$\{I_{TE}, t_{f_{TE}}\} = \mathbf{OR}(\mathbf{PAND}([I_A, t_{f_A}], [I_B, t_{f_B})], \mathbf{AND}([I_B, t_{f_B}], [I_C, t_{f_C}]), [I_D, t_{f_D}]) \quad (11)$$

Assuming that the health state of the system in Fig. 13 is $< I_A, I_B, I_C, I_D > = < 1, 1, 0, 0 >$ and failure times are $< t_{f_A}, t_{f_B}, t_{f_C}, t_{f_D} > = < 12, 8.5, 0, 0 >$; the criticality result would be (cf. Fig. 13c): I_{TE}=1 only iff I_C=1 or I_D=1, because $t_{f_A} > t_{f_B}$.

To generate the system-level failure expression the designer has to know beforehand the DFT failure logic of the system (*Dynamic Dependability Model* in Fig. 2). Based on this logic the designer needs to synthesize the failure logic in a C++ library and link the library with the SAN model. The generated library will instantiate the gates in Table 2 according to the system-specific failure logic (see Eq. (11) for a library example). The link between the SAN model and C++ libraries enables the use of the generated logic function from the SAN model at runtime, and the foresight of which are the critical assets at each simulation time instant.

Algorithm 2. System level maintenance algorithm.

```
1:  for t = 0 : Δt : T_m do                                              ▷ for all the mission time
2:    if(CheckCriticality_i), ∀asset_i ∈ system : 1 ≤ i ≤ N then         ▷ check if the failure, CBM, or repair of any asset occurs
3:      for j = 1 : N
4:        If (m(OK_asset_j) || m(StandBy_asset_j)) then                  ▷ identify which are the operative assets
5:          let m(KO_asset_j) = 1                                        ▷ change asset state to failed to check if it is critical
6:          cr_j ← criticality([m_1, t_f1], [..., ...], [m_N, t_fN])     ▷ asses the criticality if asset j would fail
7:          if cr_j = true then                                          ▷ If it is a critical asset
8:            m(EnableSystem_CBM_asset_j) = 1                            ▷ Enable the CBM action
9:        end if
10:     end if
11:   end for
12:   if m(EnableSystem_CBM_asset_i) = 1, ∀i ∈N then ▷ If we have identified a critical asset and enabled CBM
13:     for j = 1 : N do                                                 ▷ Among all the system assets
14:       If m(KO_asset_j) then                                          ▷ Check which are the non-critical failed assets
15:         m(WaitCritical_asset_j) = 1                                  ▷ And activate group-repair of non-critical assets
16:       end if
17:     end for
18:   end if
19:  end if                          ▷ at this point, the effect of the fault of the asset i in the criticality has been checked
20: end for
```

The *criticality assessment* evaluates the system failure occurrence given input conditions, but it does not affect directly the *dynamic dependability and maintenance model*. This model is connected with the *dynamic maintenance planning* through the implementation of waiting-to-repair and waiting-to-maintain signal options as shown in Fig. 12. Therefore, the *criticality assessment* informs the *dynamic maintenance planning*, and this impacts on the performance of the *dynamic dependability and maintenance model* through repair and maintenance decisions.

4.2.3. Dynamic maintenance planning

The *dynamic maintenance planning* module monitors actively the criticality of all the operative assets through the *criticality assessment* module so as to adapt to the failure, repair, and maintenance events. We keep track of failure, repair and maintenance events and failure time instants throughout the simulation by monitoring the marking of the places in the SAN model (Fig. 12). Every time the *dynamic maintenance planning* calls the *criticality assessment*, a criticality signal, *cr*, will be generated for each operative asset.

In order to trigger a group maintenance action the *dynamic maintenance planning* has to identify a critical asset. Two actions are adopted, depending on whether the asset is critical or non-critical (see Fig. 12, `System` block).

1. Critical asset: the maintenance planning activates the `EnableSystem_CBM` place immediately after diagnosing a critical asset. Thereby, if the asset is critical and it is in the `OK` state, it will be maintained according to the condition-based maintenance interval implemented through the `MaintCBM` activity. If the `Fail` activity triggers before `MaintCBM`, the critical asset will move first to the `KO` state, and then it will be repaired through `RCrit` activity.
2. Non-critical asset: if the asset is in the `KO` state, it will be repaired immediately after receiving the `WaitCritical` signal. The maintenance planning will broadcast the `WaitCritical` signal to all the failed non-critical assets as soon as a critical asset is diagnosed.

Algorithm 2 shows the grouping process for system-level condition-based maintenance strategies evaluated throughout the mission time T_m:

- `Lines 2-6`: if there is any fault among all the system assets, or if any asset is repaired or maintained, `CheckCriticality i` is activated $\forall\ asset_i \in system : 1 \leq i \leq N$ (Fig. 12). Then we check the criticality of all the remaining operative assets in the system through the criticality analysis.
- `Lines 7-9`: if any of the operative assets are critical, cr_j, then we enable the condition-based maintenance for the asset *j*.
- `Lines 12-18`: if there is any critical asset in the system, we repair all the failed assets together by enabling the group-based repair.

The outcome of the criticality assessment is the system-level indicator function $I_{TE}(t)$ which indicates if the given input conditions at time *t* cause the system failure (see Fig. 11). According to Algorithm 2 the *dynamic maintenance planning* can check the criticality of all the operative basic events in the system to foresee a critical failure occurrence. The *dynamic maintenance planning* takes this information and decides to trigger a group repair or wait until a critical failure occurs.

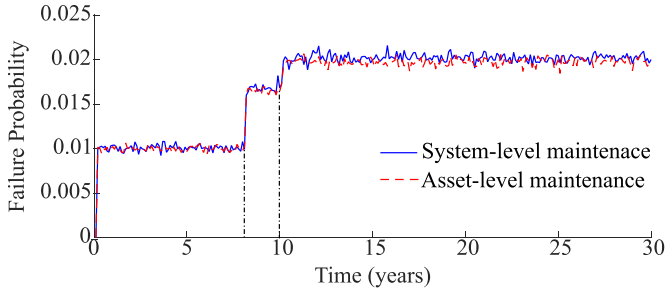

Fig. 15. System failure probability with different maintenance strategies for the DFT in Fig. 13.

Table 3
Repair and maintenance actions in Fig. 15.

Asset	*Asset-level*		*System-level*	
	#CBM	**#Repair**	**#CBM**	**#GroupRepair**
A	11.2	4.8	2.3	$\sum$ = 17.45
B	11.15	4.8	4.09	
C	11.22	4.8	3.59	
D	11.2	4.8	5.3	
$\sum$	44.77	19.2	15.28	

After detecting a critical asset, instead of triggering the group repair instantaneously as in Algorithm 2 `lines 12--18`, it is possible to postpone the group repair up to the CBM time of the critical asset. However, this decision increases the failure probability of the system maintenance strategy. There is a race condition between `MaintCBM` and `Fail` activities to move the asset from `OK` to `Maint` or `KO` state, respectively. The later we trigger the maintenance action in a critical state, the higher the likelihood to miss the maintenance time, and accordingly, asset and system failure probabilities increase.

4.2.4. System maintenance planning sensitivity

The effectiveness of the system maintenance planning depends on the criticality of assets. In turn, the criticality of assets depends on the DFT model, their position in the DFT model, and their failure parameters. So as to examine the sensitivity of the proposed system maintenance algorithm we will analyse different DFT models with different redundancy and criticality levels. Let us assume that all the basic events of all the DFT models analysed in this subsection have the same specifications: $\lambda_{t_0} = 1/10$ years^{-1}, $\lambda_{t_1} = 1/6$ years^{-1}, $\lambda_{t_2} = 1/5$ years^{-1}, t_0=0; t_1=8 years, t_2=10 years, *SF*=3 years. We use the same prediction instants and values for all the assets to analyse the sensitivity of Algorithm 2 for different systems under the same conditions. We will relax this assumption in the case study analysed in Section 5.

Fig. 15 shows the system failure probability of the DFT in Fig. 13 using asset and system maintenance strategies. Vertical dashed lines indicate prognostics prediction and failure probability update instants. We can see that the system failure probability is the same for both maintenance strategies.

Table 3 displays the number of asset-level repair actions (#Repair), the total number of system-level group repair actions (#GroupRepair), and maintenance (#CBM) actions for asset and system maintenance strategies over 30 years. With system-level maintenance, the independent sum of asset repair actions is not the same as the total number of repair actions, because assets are repaired in clusters and these clusters change throughout the lifetime of the system. Therefore #GroupRepair denotes the total non-overlapping group repair actions performed in the system.

The number of repair and maintenance actions are quantified using Monte Carlo simulations counting the number of times the maintenance activity `MaintCBM` and repair activities `Repair` and `RCrit` trigger in the models shown in Figs. 10 and Figs. 12 for asset and system maintenance strategies respectively. Each Monte Carlo trial calculates an integer number of actions, but after averaging with the total number of iterations, the final results become real values (see Eq. (8)).

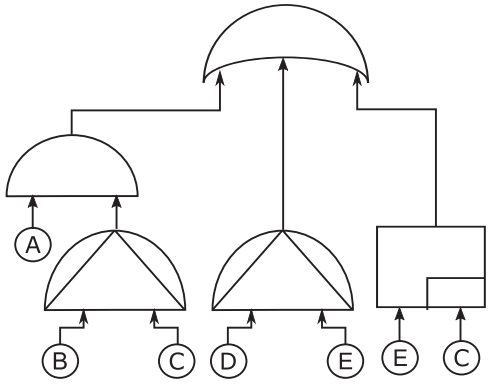

Fig. 16. Example Dynamic Fault Tree model.

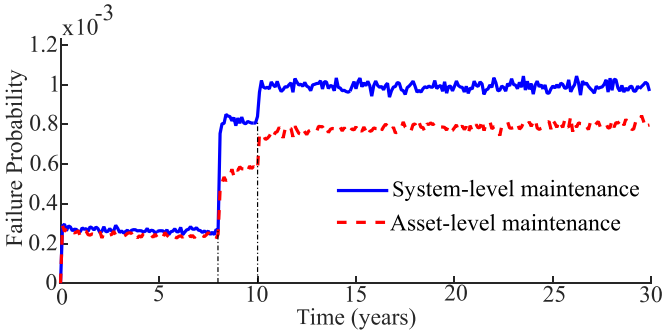

Fig. 17. System failure probability with different maintenance strategies for the DFT in Fig. 16.

Table 4
Repair and maintenance actions in Fig. 17.

Asset	*Asset-level*		*System-level*	
	#CBM	**#Repair**	**#CBM**	**#GroupRepair**
A	11.14	4.76	0.4	$\sum$ = 17.27
B	11.32	4.88	0.37	
C	3.88	2.15	2.7	
D	11.28	4.69	1.54	
E	4.44	2.18	3.63	
$\sum$	42.06	18.66	8.64	

We can see in Table 3 that the system maintenance strategy reduces preventive maintenance and reactive repair actions. For the asset maintenance strategy the number of CBM and repair actions are the same for all the assets because we use the same failure, maintenance and repair parameters. For system maintenance strategies, the more critical the asset, the more maintenance actions and vice-versa. In this case we can see in Table 3 that the most critical asset is *D* and the least is *A*. According to the DFT in Fig. 13 we can see that *D* is a single point of failure and *A* is the least critical because it requires time-dependent events, which are more restrictive than AND logic operations.

If we add more redundancies to the system, we will reduce the criticality of assets and the number of CBM actions in the system-level maintenance strategy. Let us consider the DFT model in Fig. 16, with the same failure, maintenance, and repair rates as before.

Fig. 17 confirms that the system failure probability decreases with respect to Fig. 15. In addition, we can see that the failure probability of the system maintenance strategy is slightly higher than the asset-level configuration.

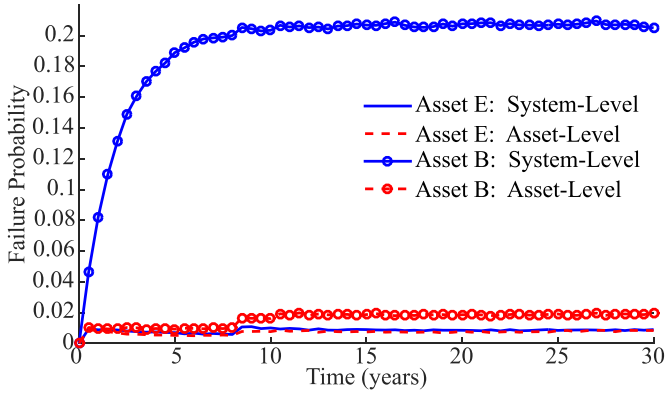

Fig. 18. Asset-level failure probability with different maintenance strategies for the DFT in Fig. 16.

The CBM or group repair is enabled after diagnosing a critical asset. The criticality changes after failure, repair or maintenance actions. This means that, the more reliable the architecture, the less frequently there will be a criticality change in the system. Therefore, the repair and maintenance actions will be enabled less frequently and accordingly, the failure probability for the system maintenance strategy will be increased with respect to the asset maintenance strategy.

Table 4 displays the number of triggered maintenance and repair actions in 30 years. We can see that maintenance and repair activities are reduced with respect to Table 3 for the system maintenance strategy. Since we use the same failure, repair and maintenance parameters as in Fig. 13, the asset maintenance strategies show very similar values to those in Table 3, with the exception of those for assets *C* and *E*. These assets are inputs of a spare gate (with activation priority of E over C) and therefore they can be in standby state, which reduces maintenance and repair actions (see Fig. 10).

We can also observe in Table 4 that the most critical asset is *E* and the least critical asset is *B*. Fig. 18 shows the asset failure probabilities for assets *B* and *E*.

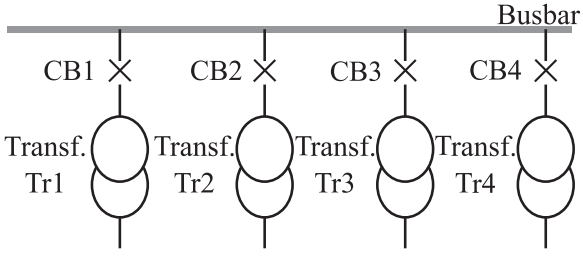

Fig. 19. Transmission substation configuration.

Table 5
RUL values (in years) at prediction times T_p.

Assets	Tr1	Tr2	Tr3	Tr4	CB1	CB2	CB3	CB4
T_{p_0}	10	10	10	10	2	2	2	2
T_{p_1}	4	5	7	7	1	1	1.5	1.5
T_{p_2}	3	4	5	5	0.5	0.5	0.8	0.8

We can see in Fig. 18 that the more critical the asset, the less difference there is between asset failure probabilities under each maintenance regime. Asset *B* is a non-critical asset and accordingly, the failure probability difference for different maintenance strategies is bigger. In the previous case shown in Fig. 13, the asset level failure probabilities are similar for all the assets because their criticalities are similar too. The implemented system maintenance strategy reduces the asset-level differences at the system-level.

5. Case study

The correct operation of a transmission substation is critical for power grid performance. Fig. 19 shows a configuration example of a transmission substation comprised of circuit breakers (CB) and transformers (Transf.).

The repair of the transformer is a very expensive and time consuming process [69]. Accordingly, the transmission substation is designed to be a fault tolerant system. In the configuration shown in Fig. 19, there are always two active transformers and the other two are in standby mode. Any time an active transformer fails, a standby transformer is activated.

We define the failure condition of the transmission substation in Fig. 19 with DFT gates (see Section 3.4). Fig. 20 defines the DFT model of the transmission substation shown in Fig. 19.

The DFT model in Fig. 20 can be interpreted as follows. The system failure will occur either because:

- Two transformers fail and two complementary circuit breakers have already failed (IE1-IE6).
- One transformer fails and three complementary circuit breakers have already failed (IE7-IE10).
- All transformers fail (spare gate).

The spare gate determines the activation priority of the inputs from left to right order. That is, any time Tr_1 is available, its activation is preferred over the rest of transformers which are in standby state.

Accordingly, we have used the Dynamic Fault Tree model in Fig. 20 to evaluate the system and asset failure probabilities with prognostics predictions in Table 5 and μ=0.1 years; θ=0.1 years (Fig. 3).

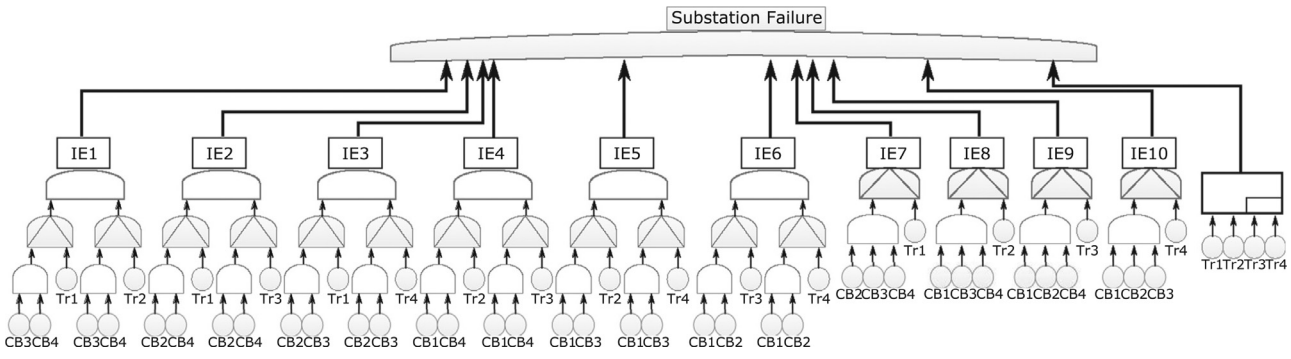

Fig. 20. Dynamic Fault Tree model of the transmission substation in Fig. 19.

Table 5 displays the RUL values assumed for transformers and circuit breakers. If available, it is possible to replace these values with realistic figures for circuit breakers [76,80] and transformers [24]. For simplicity we have adopted hypothetical reasonable values. We assume that at the initial prediction time instant (t=T_{p_0}) the estimated RUL for all transformers is 10 years and 2 years for circuit breakers. Since Tr1 and Tr2 are the preferred transformers, they will age more rapidly than Tr3 and Tr4. The prediction times for transformers are $T_{p_1} = 4$ and $T_{p_2} = 6$ years. As for circuit breakers, the prediction times are $T_{p_1} = 0.5$ and $T_{p_2} = 1$ years. We have assumed that the safety factor for transformers is 1 year and 0.1 year for circuit breakers.

5.1. Implementing the SAN model

Fig. 21 shows the SAN implementation of the transformer substation DFT model (Fig. 20) with system-level maintenance. For transformers, the asset models in Figs. 10 and 12 are implemented for asset and system maintenance strategies, respectively. For circuit breakers the same models are used without `StandBy` and `Activate` places, connecting the `Repair` and `MaintDuration` activities directly with the `OK` place.

Firstly we initialize (i) the parameters of the atomic SAN models of each asset of the system with: failure rate, repair rate, maintenance rate, prediction times and corresponding RUL values (λ_i, μ_i, ω_i, θ_i, T_{p_i}, RUL_i; $\forall asset_i \in system : 1 \leq asset_i \leq N$); and (ii) the marking of its places to the initial state:

- CB1, CB2, CB3, CB4: m(`OK`)=1, m(`KO`)=0, m(`Maint`)=0, m(`WaitCritical`)=0, m(`CheckCritical`)=0, m(`EnableSystemCBM`)=0.
- Tr1, Tr2: m(`OK`)=1, m(`KO`)=0, m(`Standby`)=0, m(`Maint`)=0, m(`Activate`)=0, m(`WaitCritical`)=0, m(`CheckCritical`)=0, m(`EnableSystemCBM`)=0.
- Tr3, Tr4: m(`OK`)=0, m(`KO`)=0, m(`Standby`)=1, m(`Maint`)=0, m(`Activate`)=0, m(`WaitCritical`)=0, m(`CheckCritical`)=0, m(`EnableSystemCBM`)=0.

For each asset, the `Check` output gate implements the update process to resample the failure and maintenance rates using prognostics prediction results through `Lambda`, `Omega`, `React`, `SF`, and `Preds` places as defined in Algorithm 1.

After defining the atomic SAN models for the assets, the system failure logic is defined. The system failure modelling process follows the same process as defined in the synthesis example in Fig. 8. The failure places of assets (denoted as `KO` places) are joined with the failure places of failure logic gates (denoted as `A`, `B` places in Fig. 7) to model the failure logic of the system defined in Fig. 20. Note that Fig. 21 only shows `IE1`, `IE10` and `Spare_Tr` submodels in detail because `IE2`-`IE9` submodels have the same failure logic. At the top-level model, all the places of all the assets are joined so as to create unique states for each asset and manage repeated events.

In parallel, so as to implement reconfiguration events through the `Activate` places of the transformers, a `Reconfigure_Transformer` model is created. The reconfiguration logic in `Reconfigure_Transformer` implements the priority of the transformers. Any time a transformer fails, the transformer with the highest priority is activated. If a transformer with a higher priority is repaired, it remains in the standby state until a lower priority transformer fails. Accordingly, the reconfiguration logic has been implemented in the SAN according to the priority of the transformers and possible failure and repair events [42]. Namely, when a standby transformer needs to be activated, the marking of the corresponding `Activate` place is set. Fig. 22 shows the `Reconfigure_Transformer` model with reconfiguration sequence examples.

At the top-level model, the operation (`OK`), failure (`KO`), `Standby` and `Activate` places of all transformers are joined with the `Reconfigure_Transformer` logic to implement the reconfiguration logic and change the marking of the transformers' places to reflect reconfigurations and state changes.

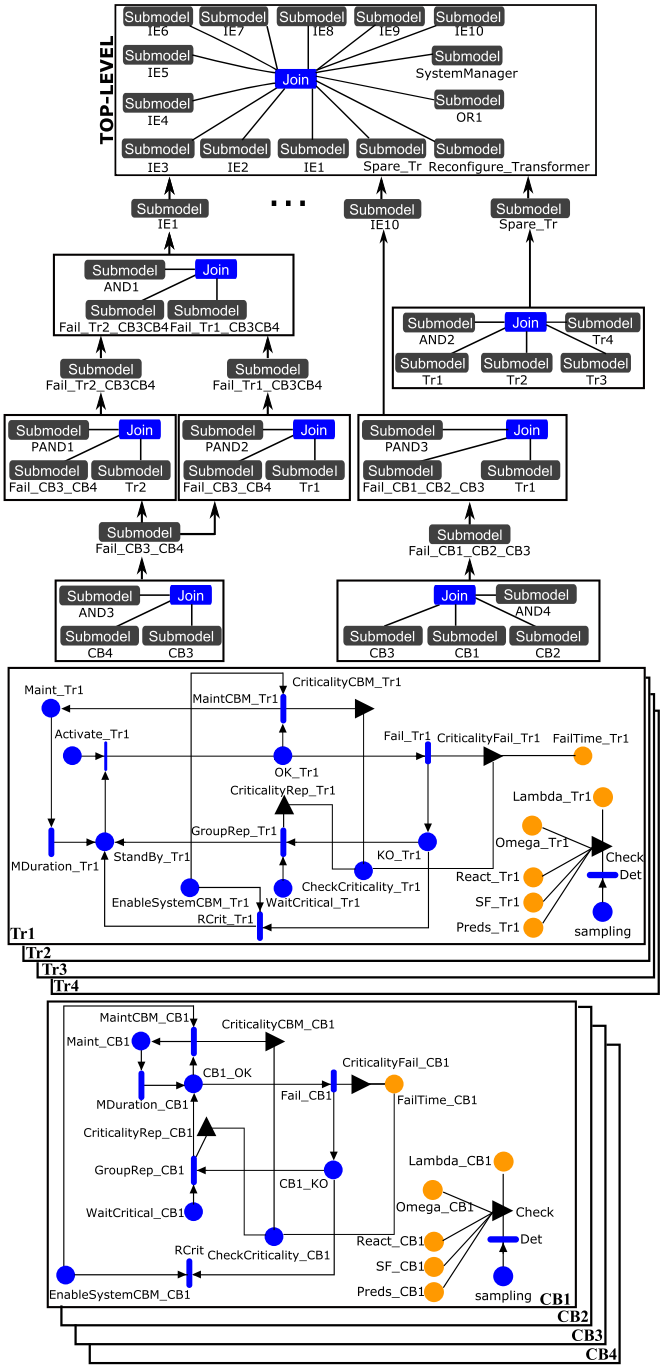

Fig. 21. SAN model of the transmission substation.

Fig. 23 shows the `SystemManager` SAN atomic model, where the `SysManager` input gate implements Algorithm 2.

Any time there is a change in the health state of any asset, the `CheckCriticality` place is set at the asset-level SAN atomic model so as to communicate with the `SystemManager` model. The `SysManager` input gate will take as input `CheckCriticality`, `FailureTime` and `KO` places of all the assets, and first it will evaluate if the change of health state of the asset affects the criticality. Then depending on the situation defined in Algorithm 2, it will activate group repairs and condition-based maintenance actions through `WaitCritical` and `EnableSystemCBM` places, respectively.

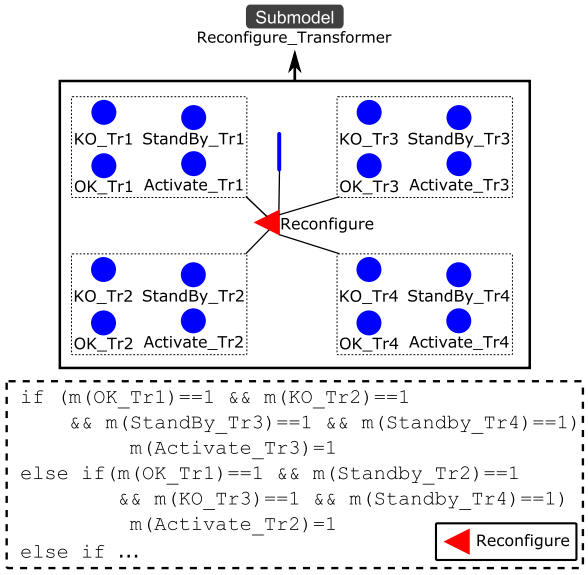

Fig. 22. Reconfiguration process for the transformer.

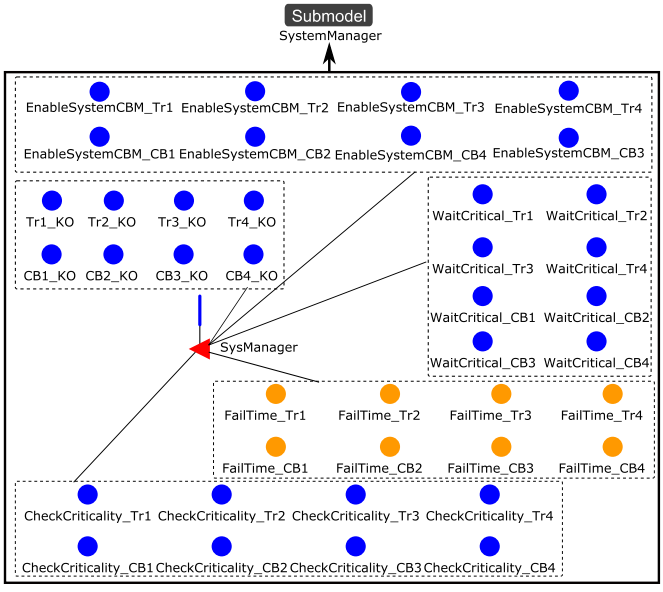

Fig. 23. System manager model.

The SAN modelling process for the asset-level maintenance strategy is the same as in Fig. 21 without system manager dependencies. The asset models are replaced with manager-independent models as shown in Fig. 10 and the top-level model will not have the `SystemManager` model [33].

In both cases, failure, reconfiguration, repair, and maintenance events will impact directly on asset models and top-level system failure logic. According to these events the assets will change their marking (health status) and this will instantaneously impact on the system failure logic changing the marking of the logic gates. Finally, asset and system level failure probabilities are evaluated through reward variables as defined in Eq. (6) by monitoring the places of interest. Namely, asset-level failure probability is monitored through `KO` places and transmission substation system failure is monitored through the `Y` place of the OR gate.

Table 6
Repair and maintenance actions for asset and system level maintenance strategies of the transmission substation in Fig. 20.

Asset	*Asset-level*		*System-level*	
	#CBM	**#Repair**	**#CBM**	**#GroupRepair**
Tr1	0.85	0.72	0.36	Σ = 14.38
Tr2	0.85	0.72	0.36	
Tr3	0.85	0.72	0.041	
Tr4	0.85	0.72	0.007	
CB1	16.38	12.98	0.0197	
CB2	16.33	12.98	0.0208	
CB3	10.58	9.26	0.024	
CB4	10.63	9.41	0.033	
Σ	57.32	47.51	0.5415	

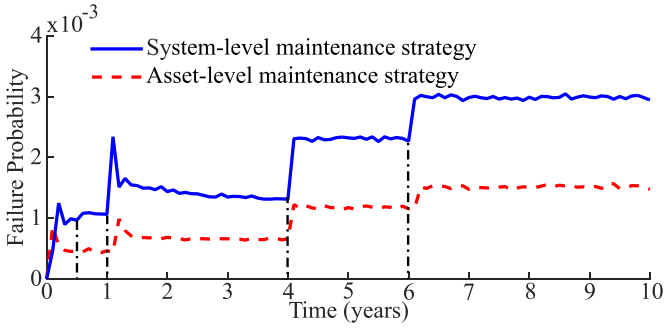

Fig. 24. System-level failure probability of the transmission substation in Fig. 19.

Similarly, the number of condition-based maintenance actions, #CBM, are quantified using Eq. (8) by monitoring the asset-level `MaintCBM` activity. For asset-level maintenance strategies #Repair actions are calculated through the `Repair` activity (Fig. 10), whereas #GroupRepair actions are quantified by the system manager model (Fig. 23) counting the number of independent group repair actions.

5.2. Results

Table 6 shows the number of preventive maintenance and reactive repair actions for asset and system maintenance strategies. Focusing on asset-level results, we can see that the preventive maintenance actions are in proportion with the values in Table 5, e.g. for CB1 with a RUL of 0.5 years at T_{p_2}, it is expected that the maintenance will trigger around 10 times (or more due to the safety factor). When the asset is not repaired preventively, the asset will be repaired reactively.

If we focus on system-level actions in Table 6 we can see that there is a significant reduction in the total number of triggered reactive repair and preventive maintenance actions. The total number of group repair actions are three times less than independent asset repair actions. The CBM actions are also reduced because this is only triggered when an asset is diagnosed as critical.

Fig. 24 shows the effect of preventive maintenance and repair actions on the system failure probability for both strategies. We can see that the asset maintenance strategy has a lower failure probability, at the cost of triggering more repair and maintenance events (Table 6).

Fig. 25 shows the asset failure probabilities for CB1 and Tr1 using system and asset maintenance strategies. We can see that the failure probability difference is greater for non-critical assets (CB1) than for critical assets (Tr1). This reflects the effectiveness of Algorithm 2, implementing preventive maintenance for critical assets and reactive repair for non-critical assets.

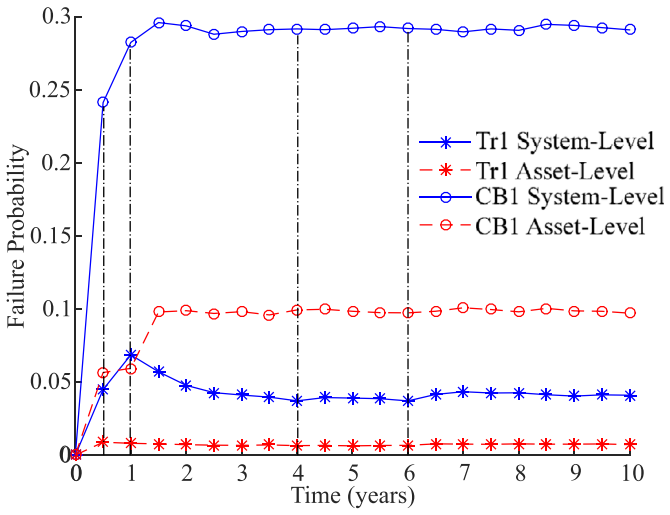

Fig. 25. Asset-level failure probability of the transmission substation in Fig. 19.

6. Discussion

Depending on the application specific requirements, the proposed algorithm permits the evaluation of whether it is cost-effective to trigger more maintenance and repair actions, or whether it is feasible to postpone them at the cost of slightly increased failure probability at the system level. In this paper the cost-effectiveness has been evaluated by considering the number of triggered maintenance and repair events for the different maintenance strategies. However, as part of our future work we will address a more complete cost-effectiveness analysis including asset costs, planned shutdown costs, downtime costs, maintenance costs and prediction costs (see Section 7).

The SAN models presented in this paper impact directly on different dependability attributes. The failure probability estimations for different maintenance strategies shown in Figs. 24 and 25 impact not only on reliability and maintenance planning, but also on safety. The dynamic dependability model quantifies the probability of occurrence of hazardous events, i.e. transmission substation failure in Fig. 20. With the consideration of prognostics information, from the point of view of safety, a better picture of the system health is obtained which can help to improve the definition of safety margin values and avoid hazardous consequences through up-to-date operational information. Note also that the dynamic dependability model includes repair actions, and accordingly, assessing the probability of being in the working state would lead directly to the availability assessment.

Note also that the failure, maintenance and repair rates can be modelled with any parametrized distribution functions. We have used the constant degradation as an example, but it is perfectly possible to use, e.g. Weibull or Gamma distributions, because the time to failure is drawn from the inverse sampling method (Section 3.3). One possible extension may be to use generic distributions for those cases in which the outcome of the prognostics method is the PDF of the RUL (Section 7).

7. Conclusions

In this paper, we proposed a method for prognostics-enhanced maintenance of complex dynamic systems which includes a new algorithm for cost effective grouping of assets. If the failure of an asset is not critical, i.e. it does not cause system failure, we leave it to fail and we repair all non-critical assets together after diagnosing a critical asset. For critical assets, we have implemented condition-based maintenance strategies based on RUL estimations. We have used Stochastic Activity Networks (SAN) for the assessment of the proposed maintenance approach. In SAN we have integrated Dynamic Fault Tree models with prognostics prediction results, criticality assessment, and alternative maintenance strategies in a rich model that can improve the accuracy of planning.

The proposed grouping maintenance algorithm reduces preventive maintenance and reactive repair actions. The experiments presented verify the effectiveness of the algorithm. We have also confirmed that the structure of the dynamic dependability evaluation model plays a vital role in the proposed system-level maintenance strategy. That is, the more reliable the system architecture, the less critical each asset will be, and accordingly preventive maintenance actions will be reduced and the system failure probability is increased.

Future work can address the extension of the proposed framework for the:

- cost-effectiveness analysis [33];
- integration of generic distributions models, e.g. [54];
- integration of other importance measures [59,77];
- analysis of opportunistic maintenance with multi-criteria decision making [78];
- evaluation of resource constrained and imperfect maintenance actions [79].

Acknowledgements

This work was supported by the EPSRC through grant number EP/M008320/1. The authors would like to thank the reviewers for their valuable comments that helped to improve the clarity and completeness of the paper.

References

[1] Kobbacy K, Murthy D. Complex System Maintenance Handbook. Springer Series in Reliability Eng., Springer; 2008.
[2] Avizienis A, Laprie J-C, Randell B, Landwehr C. Basic concepts and taxonomy of dependable and secure computing. IEEE Trans Dependable Secur Comput 2004;1:11–33. doi:10.1109/TDSC.2004.2.
[3] Bouvard K, Artus S, Bérenguer C, Cocquempot V. Condition-based dynamic maintenance operations planning & grouping. application to commercial heavy vehicles. Reliab Eng Syst Saf 2011;96(6):601–10. doi:10.1016/j.ress.2010.11.009.
[4] Horenbeek AV, Pintelon L. A dynamic predictive maintenance policy for complex multi-component systems. Reliab Eng Syst Saf 2013;120:39–50. doi:10.1016/j.ress.2013.02.029.
[5] Van PD, Barros A, Bérenguer C, Bouvard K, Brissaud F. Dynamic grouping maintenance with time limited opportunities. Reliab Eng Syst Saf 2013;120:51–9. doi:10.1016/j.ress.2013.03.016.
[6] Vu HC, Do P, Barros A, Bérenguer C. Maintenance grouping strategy for multi-component systems with dynamic contexts. Reliab Eng Syst Saf 2014;132:233–49. doi:10.1016/j.ress.2014.08.002.
[7] Nguyen K-A, Do P, Grall A. Multi-level predictive maintenance for multi-component systems. Reliab Eng Syst Saf 2015;144:83–94. doi:10.1016/j.ress.2015.07.017.
[8] Do P, Vu HC, Barros A, Bérenguer C. Maintenance grouping for multi-component systems with availability constraints and limited maintenance teams. Reliab Eng Syst Saf 2015;142:56–67. doi:10.1016/j.ress.2015.04.022.
[9] Shafiee M, Finkelstein M. An optimal age-based group maintenance policy for multi-unit degrading systems. Reliab Eng Syst Saf 2015;134:230–8. doi:10.1016/j.ress.2014.09.016.
[10] Vu HC, Do P, Barros A. A stationary grouping maintenance strategy using mean residual life and the birnbaum importance measure for complex structures. IEEE Trans Reliab 2016;65(1):217–34. doi:10.1109/TR.2015.2455498.
[11] Keizer MCO, Teunter RH, Veldman J. Clustering condition-based maintenance for systems with redundancy and economic dependencies. Eur J Oper Res 2016;251(2):531–40. doi:10.1016/j.ejor.2015.11.008.
[12] Verbert K, Schutter BD, Babuška R. Timely condition-based maintenance planning for multi-component systems. Reliab Eng Syst Saf 2017;159:310–21. doi:10.1016/j.ress.2016.10.032.
[13] Vachtsevanos G, Lewis F, Roemer M, Hess A, Wu B. Intelligent fault diagnosis and prognosis for engineering systems. John Wiley & Sons, Inc.; 2007. doi:10.1002/9780470117842.
[14] Cadini F, Zio E, Avram D. Model-based monte carlo state estimation for condition-based component replacement. Reliab Eng Syst Saf 2009;94(3):752–8. doi:10.1016/j.ress.2008.08.003.
[15] Katsouros V, Papavassiliou V, Emmanouilidis C. A bayesian approach for maintenance action recommendation. Int J Progn Health Manag 2013;4(2):1–6.

[16] Huynh K, Grall A, Bérenguer C. Assessment of diagnostic and prognostic condition indices for efficient and robust maintenance decision-making of systems subject to stress corrosion cracking. Reliab Eng Syst Saf 2017;159:237–54. doi:10.1016/j.ress.2016.11.022.

[17] Haddad G, Sandborn PA, Pecht MG. An options approach for decision support of systems with prognostic capabilities. IEEE Trans Reliab 2012;61(4):872–83. doi:10.1109/TR.2012.2220699.

[18] Zio E. M. Compare, Evaluating maintenance policies by quantitative modeling and analysis. Reliab Eng Syst Saf 2013;109:53–65. doi:10.1016/j.ress.2012.08.002.

[19] Aizpurua JI, Catterson VM. Towards a methodology for design of prognostics systems, In: Proceedings of the Annual Conference of the Prognostics and Health Management Society, Vol. 6; 2015.

[20] Sankararaman S. Significance, interpretation, and quantification of uncertainty in prognostics and remaining useful life prediction. Mech Syst Signal Process 2015;52–53:228–47. doi:10.1016/j.ymssp.2014.05.029.

[21] Camci F, Chinnam RB. Health-state estimation and prognostics in machining processes. IEEE Trans Autom Sci Eng 2010;7(3):581–97. doi:10.1109/TASE.2009.2038170.

[22] Fink O, Zio E, Weidmann U. Predicting component reliability and level of degradation with complex-valued neural networks. Reliab Eng Syst Saf 2014;121:198–206. doi:10.1016/j.ress.2013.08.004.

[23] Tobon-Mejia DA, Medjaher K, Zerhouni N, Tripot G. A data-driven failure prognostics method based on mixture of gaussians hidden markov models. IEEE Trans Reliab 2012;61(2):491–503. doi:10.1109/TR.2012.2194177.

[24] Catterson VM, Melone J, Garcia MS. Prognostics of transformer paper insulation using statistical particle filtering of on-line data. IEEE Electr Insul Mag 2016;32(1):28–33. doi:10.1109/MEI.2016.7361101.

[25] Aizpurua J, Catterson V. ADEPS: a methodology for designing prognostic applications. In: Proceedings of the Third European Conference of the Prognostics and Health Management Society 2016; 2016. p. 86–100.

[26] Banjevic D, Jardine AKS. Calculation of reliability function and remaining useful life for a markov failure time process. IMA J Manag Math 2006;17(2):115–30. doi:10.1093/imaman/dpi029.

[27] Chiacchio F, Cacioppo M, D'Urso D, Manno G, Trapani N, Compagno L, Weibull-based A. compositional approach for hierarchical dynamic fault trees. Reliab Eng Syst Saf 2013;109:45–52. doi:10.1016/j.ress.2012.07.005.

[28] Trappey CV, Trappey AJC, Ma L, Tsao W-T. Data driven modeling for power transformer lifespan evaluation. J Syst Sci Syst Eng 2014;23(1):80–93. doi:10.1007/s11518-014-5227-z.

[29] Varde P, Pecht MG. Role of prognostics in support of integrated risk-based engineering in nuclear power plant safety. Int J Progn Health Manag 2012;3(8):1–23.

[30] Aizpurua JI, Muxika E, Papadopoulos Y, Chiacchio F, Manno G. Application of the D3H2 methodology for the cost-effective design of dependable systems. Safety 2016;2(2):9. doi:10.3390/safety2020009.

[31] Aizpurua JI, Muxika E. Model-based design of dependable systems: limitations and evolution of analysis and verification approaches. Int J Adv Secur 2013;6(1, 2):12–31.

[32] Asadzadeh S, Azadeh A. An integrated systemic model for optimization of condition-based maintenance with human error. Reliab Eng Syst Saf 2014;124:117–31. doi:10.1016/j.ress.2013.11.008.

[33] Aizpurua, JI, Catterson VM, Chiacchio F, D'Urso D. A cost-benefit approach for the evaluation of prognostics-updated maintenance strategies in complex dynamic systems. in: Proceedings of ESREL'16, Glasgow, UK; 2016.

[34] Alaswad S, Xiang Y. A review on condition-based maintenance optimization models for stochastically deteriorating system. Reliab Eng Syst Saf 2017;157:54–63. doi:10.1016/j.ress.2016.08.009.

[35] Caballé N, Castro I, Pérez C, Lanza-Gutiérrez J. A condition-based maintenance of a dependent degradation-threshold-shock model in a system with multiple degradation processes. Reliab Eng Syst Saf 2015;134:98–109. doi:10.1016/j.ress.2014.09.024.

[36] Thomas L. A survey of maintenance and replacement models for maintainability and reliability of multi-item systems. Reliab Eng 1986;16(4):297–309. doi:10.1016/0143-8174(86)90099-5.

[37] Wildeman R, Dekker R, Smit A. A dynamic policy for grouping maintenance activities. Eur J Oper Res 1997;99(3):530–51. doi:10.1016/S0377-2217(97)00319-6.

[38] van Noortwijk J. A survey of the application of gamma processes in maintenance. Reliab Eng Syst Saf 2009;94(1):2–21. doi:10.1016/j.ress.2007.03.019.

[39] Rasmekomen N, Parlikad AK. Condition-based maintenance of multi-component systems with degradation state-rate interactions. Reliab Eng Syst Saf 2016;148:1–10. doi:10.1016/j.ress.2015.11.010.

[40] Manno G, Knutsen K, Vartdal B. A risk based framework for system level condition monitoring. In: Condition Monitoring and Machinery Fault Prevention Technologies; 2014.

[41] Khorasgani H, Biswas G, Sankararaman S. Methodologies for system-level remaining useful life prediction. Reliab Eng Syst Saf 2016;154:8–18. doi:10.1016/j.ress.2016.05.006.

[42] Aizpurua JI, Papadopoulos Y, Muxika E, Chiacchio F, Manno G. On cost-effective reuse of components in the design of complex reconfigurable systems, Quality and Reliability Engineering International. doi: http://dx.doi.org/10.1002/qre.2112.

[43] Bouissou M, Bon J-L. A new formalism that combines advantages of fault-trees and markov models: boolean logic driven markov processes. Reliab Eng Syst Saf 2003;82(2):149–63. doi:10.1016/S0951-8320(03)00143-1.

[44] Manno G, Chiacchio F, Compagno L, D'Urso D, Trapani N. Conception of Repairable Dynamic Fault Trees and resolution by the use of RAATSS, a Matlab® toolbox based on the ATS formalism. Reliab Eng Syst Saf 2014;121:250–62. doi:10.1016/j.ress.2013.09.002.

[45] Portinale L, Raiteri DC. Modeling and analysis of dependable systems: a probabilistic graphical model perspective. River Edge, NJ, USA: World Scientific Publishing Co., Inc; 2015.

[46] Distefano S, Puliafito A. Dependability evaluation with dynamic reliability block diagrams and dynamic fault trees. IEEE Transanctions Dependable Secur Comput 2009;6(1):4–17.

[47] Kaiser B, Gramlich C, Forster M. State-event fault trees - a safety analysis model for software-controlled systems. Reliab Eng Syst Saf 2007;92(11):1521–37.

[48] Papadopoulos Y, Walker M, Parker D, Sharvia S, Bottaci L, Kabir S, Azevedo L, Sorokos I. A synthesis of logic and bio-inspired techniques in the design of dependable systems. Annu Rev Control 2016;41:170–82 10.1016/j.arcontrol.2016.04.008.

[49] Chiacchio F, D'Urso D, Manno G, Compagno L. Stochastic hybrid automaton model of a multi-state system with aging: reliability assessment and design consequences. Reliab Eng Syst Saf 2016;149:1–13. doi:10.1016/j.ress.2015.12.007.

[50] Zille V, Bérenguer C, Grall A, Despujols A. Modelling multicomponent systems to quantify reliability centred maintenance strategies. Proc Inst Mech Eng, Part O: J Risk Reliab 2011;225(2):141–60. doi:10.1177/1748006X11402269.

[51] Andrews J, Prescott D, Rozieres FD. A stochastic model for railway track asset management. Reliab Eng Syst Saf 2014;130:76–84. doi:10.1016/j.ress.2014.04.021.

[52] Andrews J, Fecarotti C. System design and maintenance modelling for safety in extended life operation. Reliab Eng Syst Saf 2017. doi:10.1016/j.ress.2017.01.024.

[53] Alrabghi A, Tiwari A. A novel approach for modelling complex maintenance systems using discrete event simulation. Reliab Eng Syst Saf 2016;154:160–70. doi:10.1016/j.ress.2016.06.003.

[54] Koutras V, Malefaki S, Platis A. Optimization of the dependability and performance measures of a generic model for multi-state deteriorating systems under maintenance. Reliab Eng Syst Saf 2017. doi:10.1016/j.ress.2017.01.002.

[55] Sanders WH, Meyer JF. Stochastic activity networks: Formal definitions and concepts. In: Lectures on Formal Methods and Performance Analysis, Vol. 2090 of LNCS, Springer Berlin Heidelberg; 2001, pp. 315–343. doi: http://dx.doi.org/10.1007/3-540-44667-2_9.

[56] Aizpurua JI. Functionality and dependability assurance in massively networked scenarios, [Ph.D. thesis], Electronics and Computing Department, Mondragon University (January 2015).

[57] Dugan J, Bavuso S, Boyd M. Dynamic fault-tree models for fault-tolerant computer systems. IEEE Trans Reliab 1992;41(3):363–77. doi:10.1109/24.159800.

[58] Borgonovo E. Differential, criticality and birnbaum importance measures: an application to basic event, groups and SSCs in event trees and binary decision diagrams. Reliab Eng Syst Saf 2007;92(10):1458–67. doi:10.1016/j.ress.2006.09.023.

[59] Kuo W, Zhu X. Importance measures in reliability, risk, and optimization. John Wiley & Sons, Ltd; 2012. doi:10.1002/9781118314593.

[60] Van PD, Barros A, Bérenguer C. Reliability importance analysis of markovian systems at steady state using perturbation analysis. Reliab Eng Syst Saf 2008;93(11):1605–15. doi:10.1016/j.ress.2008.02.020.

[61] Van PD, Barros A, Bérenguer C. From differential to difference importance measures for markov reliability models. Eur J Oper Res 2010;204(3):513–21. doi:10.1016/j.ejor.2009.11.025.

[62] Wang W, Loman J, Vassiliou P. Reliability importance of components in a complex system. in: Reliability and Maintainability, 2004 Annual Symposium - RAMS; 2004, p. 6–11.

[63] An D, Kim NH, Choi J-H. Practical options for selecting data-driven or physics-based prognostics algorithms with reviews. Reliab Eng Syst Saf 2015;133:223–36. doi:10.1016/j.ress.2014.09.014.

[64] Walker M, Papadopoulos Y. Qualitative temporal analysis: towards a full implementation of the fault tree handbook. Control Eng Pract 2009;17(10):1115–25.

[65] Merle G, Roussel J-M, Lesage J-J. Quantitative analysis of dynamic fault trees based on the structure function. Qual Reliab Eng Int 2014;30(1):143–56.

[66] Tang Z, Bechta Dugan J. Minimal cut set/sequence generation for dynamic fault trees. in: Reliability and Maintainability, 2004 Annual Symposium - RAMS; 2004, p. 207–13. doi: http://dx.doi.org/10.1109/RAMS.2004.1285449.

[67] Trivedi KS. Probability and statistics with reliability, queuing and computer science applications. 2nd edition. Chichester, UK: John Wiley and Sons Ltd; 2002.

[68] Azadeh A, Asadzadeh S, Salehi N, Firoozi M. Condition-based maintenance effectiveness for series-parallel power generation system-a combined markovian simulation model. Reliab Eng Syst Saf 2015;142:357–68. doi:10.1016/j.ress.2015.04.009.

[69] CIGRÉ, Transformer Reliability Survey, no. 642; 2015.

[70] Meyer JF, Movaghar A, Sanders WH. Stochastic activity networks: Structure, behavior, and application. in: International Workshop on Timed Petri Nets. IEEE Computer Society, Washington, DC, USA; 1985, p. 106–15.

[71] Sanders WH, Meyer JF. Reduced base model construction methods for stochastic activity networks. IEEE J Sel Areas Commun 1991;9(1):25–36. doi:10.1109/49.64901.

[72] Zio E. The monte carlo simulation method for system reliability and risk analysis, Springer Series in Reliability Engineering; 2013.

[73] Boudali H, Crouzen P, Stoelinga M, Compositional A. Semantics for dynamic fault trees in terms of interactive markov chains. Springer Berlin Heidelberg; 2007. p. 441–56. doi:10.1007/978-3-540-75596-8_31.

[74] Aizpurua JI, Catterson VM. On the use of probabilistic model-checking for the verification of prognostics applications, in: IEEE International Conference on Intelligent Computing and Information Systems; 2015, pp. 7–13. http://dx.doi.org/10.1109/IntelCIS.2015.7397225.

[75] Shafiee M, Finkelstein M, Berenguer C. An opportunistic condition-based maintenance policy for offshore wind turbine blades subjected to degradation and environmental shocks. Reliab Eng Syst Saf 2015;142:463–71. doi:10.1016/j.ress.2015.05.001.

[76] Rudd S, Catterson V, McArthur S, Johnstone C. Circuit Breaker Prognostics Using SF_6 Data. in: Power and Energy Society General Meeting, 2011 IEEE; 2011. p. 1–6. doi: http://dx.doi.org/10.1109/PES.2011.6039599.

[77] Wu S, Chen Y, Wu Q, Wang Z. Linking component importance to optimisation of preventive maintenance policy. Reliab Eng Syst Saf 2016;146:26–32. doi:10.1016/j.ress.2015.10.008.

[78] Cavalcante CA, Lopes RS. Multi-criteria model to support the definition of opportunistic maintenance policy: a study in a cogeneration system. Energy 2015;80:32–40. doi:10.1016/j.energy.2014.11.039.

[79] Do P, Voisin A, Levrat E, Iung B. A proactive condition-based maintenance strategy with both perfect and imperfect maintenance actions. Reliab Eng Syst Saf 2015;133:22–32. doi:10.1016/j.ress.2014.08.011.

[80] Aizpurua JI, Catterson VM, Abdulhadi IF, Garcia MS. A Model-Based Hybrid Approach for Circuit Breaker Prognostics Encompassing Dynamic Reliability and Uncertainty, in: IEEE Transactions on Systems, Man, and Cybernetics: Systems vol. 99, pp.1-12, doi:10.1109/TSMC.2017.2685346.