to standard functioning [9,25]. In this work we propose a method based on Bayesian networks to detect anomalous behaviours in this type of systems, encouraged by the significant examples of successful applications of Bayesian networks to condition monitoring from [27,22], predictive maintenance from [6,3,18] and fault detection/diagnosis from [28,26].

In order to design that DSS, we will start from the methodology for detecting faults and abnormal behaviours described in [22]. Afterwards, we will design a new metric to improve the behaviour of the previous methodology in some general cases. In order to demonstrate the usefulness of our new proposal, we will generate some artificial datasets and we will compare the performance between the original methodology and our proposal.

This study is structured as follows. In Section 2 we briefly introduce Bayesian networks. Section 3 contains the description of our DSS, describing as well the methodology proposed in [22]. Later, in Section 4 we will explain the kind of industrial system our application will work for, and we will discuss the benefits of our proposal through an artificial dataset. Also some tests are carried out in order to evaluate its performance. Section 5 contains our concluding remarks. Finally, in Appendix 1 we describe the designed decision support web-based application, while in Appendix B we show the parameters of the Bayesian networks (probability tables) used to generate the synthetic datasets.

2. Bayesian Networks

Bayesian networks (BNs) are mathematical objects which inherently deal with uncertainty [11,13]. When used for probabilistic reasoning, a BN represents the knowledge base of a probabilistic expert system. From a descriptive point of view, we can distinguish two different parts in a BN, which respectively accounts for the qualitative and quantitative parts of the model. Figure 1 shows an example of a simple BN.

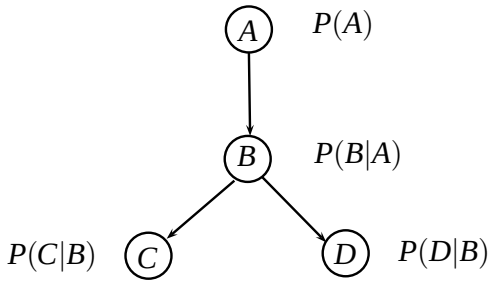

Fig. 1. An example of BN with four variables.

The qualitative part of the network is represented by a directed acyclic graph (DAG), $\mathscr{G}$, whose nodes represent the random variables in the problem domain and whose edges codify relevance relations between the variables they connect. When the network is built by hand with the help of domain experts, these relevance relations are usually of *causal* nature, while when the network is learnt from data, we can only talk about probabilistic dependence, but not causality. The whole graphical model codifies the (in)dependence relations among all variables and can be interpreted by using the D-separation criterion [23] in order to carry out a qualitative or relevance analysis.

On the other hand, the quantitative part of the model consists of a set of conditional probability distributions, one for each node (variable): $P(X_i|pa_{\mathscr{G}}(X_i))$, where $pa_{\mathscr{G}}(X_i)$[a] are the parent nodes of X_i in the DAG $\mathscr{G}$. From the independences codified by the DAG, the joint probability distribution can be recovered from the BN factorization as shows the Eq. (1).

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|pa(X_i)). \qquad (1)$$

Once a BN has been built for a given problem domain, it becomes a powerful tool for probabilistic reasoning, with a great range of exact and approximate convenient algorithms to form the inference engine [4]. Depending on the target domain and the availability of domain experts and/or data, the network can be manually constructed by using knowledge engineering techniques [14,12], automatically learnt from data [7,21], or combining both techniques.

[a] From now on we will simply write $pa(X_i)$ instead of $pa_{\mathscr{G}}(X_i)$ when no confusion about the graph/network is possible

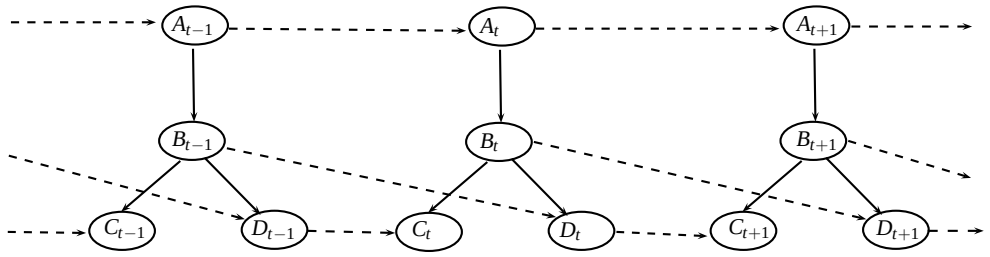

Figure 2: An example of DBN structure unfolded over time. Dashed arcs are the temporal relations.

An important and attractive issue of BNs is their ability to incorporate the *temporal* dimension, allowing in this way reasoning over time. Thus, while a *static* BN represents a fixed capture of the domain environment in a given instant, *dynamic BNs* (DBNs) allow to explicitly represent different instances of the same variables over time, as well as temporal relations between them. Figure 2 shows the most used way of dealing with DBNs in the literature [20], which consists of a basic structure (*slice*), which represents a static BN, together with a set of temporal relations representing the dependences from time $t-1$ to time t. This structure is unfolded as many times as needed in order to forecast the values of variables at time $t+k$. As can be noticed, the Markovian condition is assumed in DBNs.

3. Fault Diagnosis Methodology

One of the main applications of DSS is helping to carry out a predictive maintenance. It builds a model from a set of examples which is used to predict the health status of the monitored system [2]. Usually, data is labelled and we know the class for each instance (if it is a normal behaviour, or on the contrary if some component is failing), so supervised data mining techniques can be used to build the prediction model. However, sometimes this information cannot be acquired, and other approaches have to be used. For example, nearest neighbour based anomaly detection techniques uses the definition of a distance or similarity measure between data instances to determine if an example is an anomaly or not [47,46]. Clustering based anomaly detection techniques tries to group similar data instances into clusters. Afterwards, if an incoming instance does not belong to any cluster it would be an outlier [33,34]. Sometimes, this assumption is relaxed and an incoming instance is marked as an outlier if it is far away from the centroid of the cluster that it belongs to [35,36], or if it belongs to a sparse cluster [37,38]. Statistical anomaly detection techniques say that an observation which is not generated by the assumed stochastic model is an anomaly. These models are built from data using parametric [45,44,43] or non-parametric techniques [42,41]. Also, classification based anomaly detection techniques has been widely used [40,39,22]. These techniques learn a model which distinguishes between normal and anomalous classes.

In this paper we aim to propose a failure detection tool whose goal is to detect generic failures from the information collected in a sensored system. This tool is based on a technique included in the last mentioned group (classification based anomaly detection techniques), and in more detail, using Bayesian networks as models to represent the behaviour of the system. Therefore, our BN-based system is not targeted for the detection of a concrete type of failure in a particular machine or set of machines, but on the contrary, what we aim is to be able of detecting *any anomalous* behaviour of the system (that can be inferred from sensors readings). Obviously, dealing with such a so general problem represents a disadvantage with respect to tailored systems, due to the absence of problem domain knowledge. On the other hand, it is more general and realistic because we can detect previously unknown failures. In our case, the system being monitored can be viewed as a black box (Figure 3).

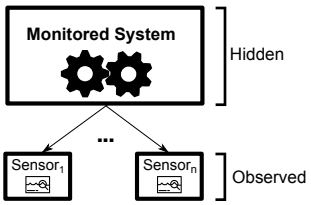

Fig. 3. General system structure

We cannot consider to deal with this problem by using supervised classification methodologies because of the nature of the input data. In particular, we do not have data about each possible wrong behaviour of the system, in fact, we only dispose of data corresponding to one or more status of the machinery working in absence of failures. Our goal is to predict whether the current behavioural state of the physical system is *correct* or on the contrary it stands for some kind of *failure*. Therefore, instead of a fully supervised classification methodology, we opt for the use of an *anomaly detection* [2] based one.

In this type of methodologies, a model is constructed to represent the failure-free behaviour of the physical system, and it is used to check if an incoming sensor reading does not match it, producing a *failure warning* (or *alarm*) in such a case. We will start from the methodology detailed in [22], adapted to our case of such general problems. However, as we will discuss in Section 4, the previous methodology has problems in some specific circumstances. In order to improve the performance of our DSS in such cases, we have introduced a new model definition (in addition to the failure-free behaviour one) to represent the anomalous behaviour. As an overview, the workflow of our DSS is shown in Figure 4.

Every cycle (reading-detection) has the following behaviour: (1) readings are taken from the sensors and stored in a data base; (2) these readings, possibly manipulated, give rise to the observations entered into the expert system for classification; (3) the probabilistic expert system (composed of two Bayesian networks) computes the prediction (fault or non-fault); (4) if a *possible* failure has been identified, an alarm is generated and a human operator supervises it; finally (5) the knowledge-base is updated according to this information (false or true positive alarm).

Therefore, the proposed DSS has been designed as a generic tool, where no specific problem domain has been considered. Nevertheless, it is noteworthy that if some previous knowledge is available, it can be incorporated to our system, as this is one of the main advantages of using a Bayesian network to represent the knowledge base. In order to deal with a so generic failure detection system, we impose the two following assumptions:

- The tool needs a first stage in which it collects data from the monitored system (sensors readings) working properly, i.e. without failures. From these readings, the system will construct/learn a *correct* behaviour model. This is not a hard assumption, since long periods in the absence of failures are common in industrial machinery.
- Some degree of *supervision* is needed. As we do not have prior information about how a failure looks like, once a suspicious behaviour is detected from the sensors readings because it deviates from the (learnt) correct behaviour model, we need that a human operator confirms if the detected *anomalous* behaviour actually corresponds to a failure in the system or if it is just a false positive. This information will be used as feedback to improve the prediction models.

This section is structured in the following subsections. First, in Subsection 3.1 we are going to describe both models for failure-free and anomalous behaviour. Then, in Subsection 3.2 we detail the metrics used in our DSS and how we combine them to detect anomalous behaviours. Finally, in Subsection 3.3 we detail how these models are updated when a human operator tells the system about false or true positive alarms.

3.1. Bayesian networks to detect anomalous behaviours

As we said before, we are going to use two Bayesian networks: $\mathscr{M}_c$, the model to represent failure-free behaviours and $\mathscr{M}_f$ for anomalous behaviour.

Bayesian networks assume discrete values as inputs. However, this is not usual in the case of sensor readings, e.g. temperature. One option would be to use Hybrid Bayesian networks [48] instead, which

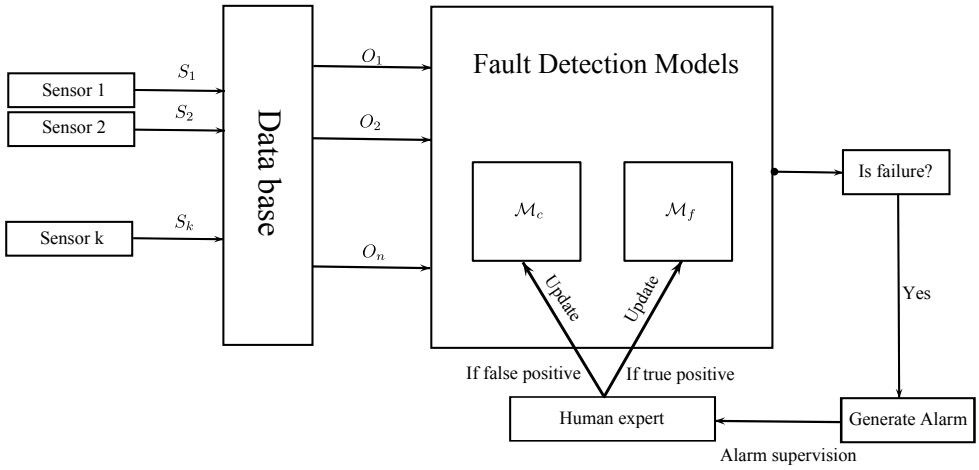

Fig. 4. System behaviour

can deal with real input values using a conditional gaussian model. However, this approach has some drawbacks, as imposing certain restrictions about the topology of the network. Another option, used in this work, consists on applying a pre-processing discretization step, where the discretization intervals can be provided by domain experts, obtained by applying some unsupervised discretization technique [16] or just using equal width binning.

In the following subsubsections we detail how $\mathscr{M}_c$ and $\mathscr{M}_f$ are built (Subsubsection 3.1.1 and 3.1.2 respectively).

3.1.1. *Probabilistic model for* failure-free *behaviour:* $\mathscr{M}_c$

The physical system for which we intend to carry out the predictive maintenance can be in a serie of (*hidden*) states. We have no direct access to these inner states, but to some observable measures provided by a set of sensors (e.g. light, vibration, temperature, etc.) attached to the machinery. Let us assume we have n sensors $\{S_1,\ldots,S_n\}$ each one taking values in $dom(S_i) = \{v_1^i,\ldots,v_r^i\}$. That is, we assume each sensor can take value in a *finite* set of discrete/nominal values.

The main goal is to obtain a probabilistic model which represents the behaviour of the system working properly, that is, without any defective component and so assuming correct sensors readings. This probabilistic model, which we will call $\mathscr{M}_c$ needs to deal with the n sensors but also with the temporal relations among them, as the evolution on the value of a given sensor, will provide clues on its correct or incorrect functioning. Because of these requirements, we have chosen the formalism of DBNs [20] to represent our knowledge base.

For the sake of simplicity and because of the lack of prior domain knowledge we have resorted to a simple DBN model with fixed graphical structure. In the model, each sensor is independent of the rest[b] but depends on itself at time $t-1$, that is, there is no arcs of type $S_i \rightarrow S_j$, but we include *temporal* arcs of type $S_i^{t-1} \rightarrow S_i^t$. In this way, as there are no relations between different sensors, it can be seen as independent Bayesian networks (one per sensor). However, the definition that we propose in this work contemplate a more general situation, where sensors can be dependant of others at the same time t.

In practice, for better readability reasons, we actually deal with a static BN, in which temporal relations are taking into account by unfolding the DBN for a given number of temporal slices (in this work equals to 24). In fact, the way to procede with DBN is to unfold 1 layer at each time, forgetting (or not)

[b] Notice that this is only a modelling assumption, not an strong constraint. In fact, if problem domain knowledge is available indicating direct dependence between two sensors, then this dependence can be simply added to the model, as we will see in Section 4.

the past layers. However, we determined to use this strategy (unfolding the 24 layers at once) because of the monotonous behaviour of these types of environments: even if the machinery carries out different operations depending on the hour of the day, this pattern is usually the same every day (24 hours). The resulting model is shown in Figure 5, where we represent 24 consecutive hourly measures for each sensor, that is, we model a full day of the physical system.

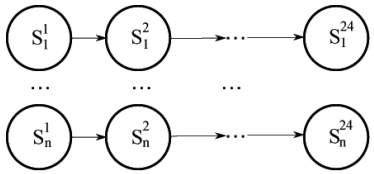

Fig. 5. Graphical structure of the unfolded DBN

Once we have described the structure of our probabilistic graphical model $\mathscr{M}_c$, we need to provide the numerical parameters, that is, the conditional probability tables. To do this we use the dataset containing sensors readings captured during a period of correct functioning of the monitored system. Our first task is to transform the captured dataset, containing hourly readings for the n variables (sensors) to a new one containing $24 \times n$ variables. The process is described in Figure 6.

From the transformed dataset, where each s_i^k belongs to $dom(S_i)$, we estimate the marginal $\{P(S_i^1)\}_{i=1}^{n}$ and conditional $\{P(S_i^t|S_i^{t-1})\}_{i=1,t=2}^{i=n,t=24}$ probability distributions (tables) by using Laplace smoothing [17]. In general, if more dependence relations are included in the graphical model, then the set of conditional probability tables would be estimated as $\{P(S_i^t|pa(S_i^t)\}_{i=1,t=1}^{i=n,t=24}$.

3.1.2. *Probabilistic model for* anomalous *behaviour:* $\mathscr{M}_f$

Apart from $\mathscr{M}_c$ that models the normal behaviour of the physical system, we propose the use of a *failure* model $\mathscr{M}_f$ which models the anomalous behaviour of the physical system, that is, how the readings look-like when some failure is happening or is close to happen. In our case, as we have previously described, due to the generality of the approach we have no data to learn this model. Nonetheless, we have decided to use a random failure model, $\mathscr{M}_f$, which has the same graphical structure as $\mathscr{M}_c$, but whose parameters (local probability distributions) have been randomly generated using an uniform distribution. The rationale behind using this model is that normal sensors readings will have a high likelihood of being generated by $\mathscr{M}_c$ and a very low likelihood of being generated by $\mathscr{M}_f$. However, in the case of *abnormal* sensors readings, the likelihood with respect to $\mathscr{M}_c$ should decrease, while the one with respect to $\mathscr{M}_f$ shoud increase or at least do not change substantially.

3.2. *Anomaly detection procedure*

After learning $\mathscr{M}_c$ and generating $\mathscr{M}_f$, they are used to process new sensor readings. The goal is to predict whether a new reading represents a failure-free behaviour for the machinery, or on the contrary it represents some anomalous behaviour, which can be associated with a failure or with a *warning* that indicates a forthcoming failure.

In the following subsubsections we present two metrics and the way they are combined in order to classify input readings into failure-free and anomalous behaviours. In Subsubsection 3.2.1 we show a metric based exclusively in $\mathscr{M}_c$, used in the methodology proposed in [22]. Then, in Subsubsection 3.2.2, we explain our proposal to improve the performance of the previous methodology. Finally, in Subsubsection 3.2.3 we detail how we detect anomalous behaviours combining the previous metrics.

3.2.1. *Metric* $conf(\mathbf{e})$

The methodology proposed in [22] is based on measuring the conflict between the model and the sensor readings. A fault (an anomaly) will be detected when that measurement reaches a threshold. To detect if a particular case or reading is coherent with the model $\mathscr{M}_c$ or not, it uses a procedure which is based on the well known *conflict* (*conf*) measure proposed in [10] (see also [11]). The measurement is described in Eq. (2).

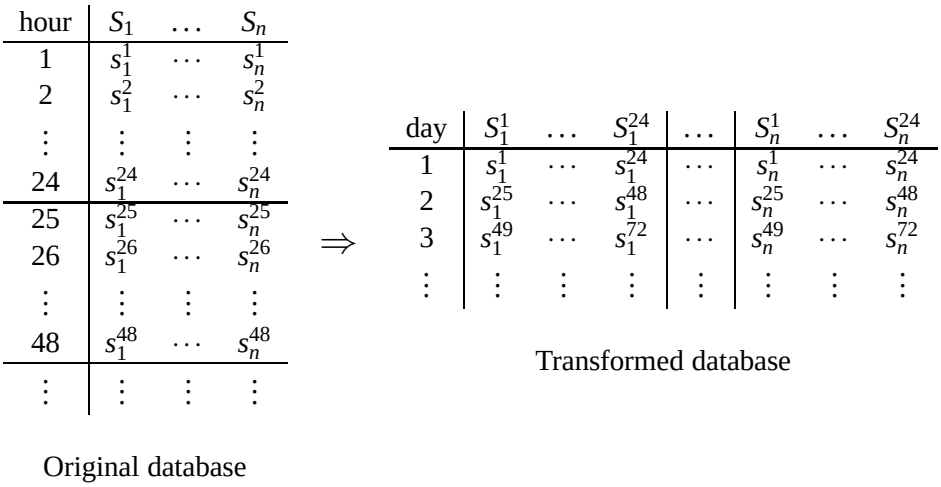

Fig. 6. Database transformation

$$conf(\mathbf{e}) = \ln \frac{P_{\mathscr{M}}(e_1) \times \cdots \times P_{\mathscr{M}}(e_n)}{P_{\mathscr{M}}(\mathbf{e})}, \quad (2)$$

where **e** stands for the joint configuration of n findings, i.e., $\mathbf{e} = (e_1, e_2, \ldots, e_n)$.

The idea behind *conf* measure is that findings coherent with the model should be positively correlated, thus, $P_{\mathscr{M}}(\mathbf{e})$ should be greater than the product of independent (marginal) probabilities, $P_{\mathscr{M}}(e_1), \ldots, P_{\mathscr{M}}(e_n)$. Therefore, a negative number would indicate that **e** is coherent with the model $\mathscr{M}$, while a positive one is an indicator of a conflict. The bigger the number, the more probable the conflict is.

From a computational point of view, computing $conf(\mathbf{e})$ requires two propagations (inferences) [11] over the probabilistic graphical model: (1) in the first one, no evidence is entered into the network, so, marginal probabilities $P(e_1), \ldots, P(e_n)$ are obtained; and (2) in the second one, all the findings $e_1, \ldots, e_n$ are entered as evidence, and $P(\mathbf{e}) = P(e_1, \ldots, e_n)$ is obtained after the propagation by computing in any node the normalization constant.

In our case, as we are interested in the predictive maintenance of the machinery, we use as findings the sequence of readings from time t_i to time t_j, where $1 \leqslant i, j \leqslant 24$ and $w = j - i$ is a positive integer. This is because a failure usually does not happen abruptly, but gradually, so we consider the sequence of readings in order to evaluate possible trends. However, we consider a maximum time window size w (temporary difference between t_j and t_i), because if we take into account the whole set of readings, the information provided by the last measurements would have a small impact as they are diluted by the rest of the readings.

3.2.2. *Metric* $rcf(\mathbf{e})$

The previous metric detects anomalies paying attention to the dependencies between variables. As it will be discussed in Section 4, an uncommon sequence of readings might be considered a failure-free behaviour. In order to improve the performance of our DSS, we will introduce a second measurement (which uses both models, $\mathscr{M}_c$ and $\mathscr{M}_f$). This new measure, called ratio correct vs fault (rcf), is shown in Eq. (3).

$$rcf(\mathbf{e}) = \ln \frac{P_{\mathscr{M}_f}(\mathbf{e})}{P_{\mathscr{M}_c}(\mathbf{e})} \quad (3)$$

At the begining, when the parameters of $\mathscr{M}_f$ has been initialized randomly, a change in the behaviour of the system should not affect substantially to $P_{\mathscr{M}_f}(\mathbf{e})$. On the contrary, if that situation corresponds to an abnormal behaviour, $P_{\mathscr{M}_c}(\mathbf{e})$ should decrease and therefore the ratio $rfc(\mathbf{e})$ should increase. On the other hand, if readings correspond to a normal situation, $P_{\mathscr{M}_c}(\mathbf{e})$ should be higher than $P_{\mathscr{M}_f}(\mathbf{e})$ as the parameters of $\mathscr{M}_c$ has been learnt from data

in abscence of failures. Moreover, when the model $\mathscr{M}_f$ has been updated with data from anomalous behaviours, the measure $rcf(\mathbf{e})$ should improve its performance as the parameters of the model $\mathscr{M}_f$ has been updated using data of anomalous situations, and therefore $P_{\mathscr{M}_c}(\mathbf{e})$ and $P_{\mathscr{M}_f}(\mathbf{e})$ should behave oppositely.

As in *conf(**e**)*, we will use as evidence the sequence of readings from time t_i to t_j, where **e** has the same meaning as in Eq. (2) and the same window size w is used.

3.2.3. *Use of $conf(\mathbf{e})$ and $rcf(\mathbf{e})$ for anomaly detection*

In this section we are going to explain how we use *conf(**e**)* and *rcf(**e**)* to detect anomalous behaviours. The basic idea behind this is to use two thresholds, one for each metric. Each time one measure is greater than its associated threshold, then an anomalous behaviour will be detected. However, we realised that using the whole evidence **e** to compute each measure has some drawbacks. As aforementioned, it is an unusual situation that a component fails, and usually when it happens only a few sensors would be involved. Because of that, if the system is composed of a large number of sensors, the information of an anomaly can be diluted by the rest of sensors and the failure detection can be noteless.

In order to deal with that problem, the previously described measures ($conf$ and rcf) are separately computed for each different sensor in our system, using as evidence the readings from that sensor and from all its ascendants in the network (for time window w). This gives us information about the probability that a concrete sensor reading comes from an anomalous behaviour or not. Moreover, we are able to detect if there is a failure on the machinery and if so, the defective component.

Finally, we use the information of all these individual measures to decide if the input represents a (forthcoming) failure, and so an *alarm* must be fired. We have set two thresholds (t_{conf} and t_{rcf}), one for each measure respectively. If any of the computed measures is greater than its associated threshold, then an *alarm* (related to the evaluated sensor) is triggered. For the sake of clarity, in our experiments (see Section 4) we do not pay attention to what component is failing, but only if any component in the whole system is failing or not.

3.3. *Models updating*

Even if no domain knowledge is used to build the model, the information about its performance can be used as feedback. Therefore, if at some point the system triggers an alarm, a person who checks the status of the monitored system can tell the DSS if the behaviour has been correctly classified or not. This information can be used to update the models and improve their predictions.

Given a certain alarm, if it is marked as a false positive it means that data come from a correct behaviour, so the model $\mathscr{M}_c$ will be updated using this information (as described afterwards). On the contrary, if it is marked as an actual failure, it means that data come from an anomalous behaviour, so the model $\mathscr{M}_f$ will be updated. The data used to make the model updates correspond to those contiguous readings from the first detection of the alarm to the last one (the reading before data is considered again as normal behaviour).

The update process keeps the model structure invariable but changes the parameters (probability tables). Let $\{P(S_i^1)\}_{i=1}^n$ be the marginal probability table for the sensor i in the first layer, and $\{P(S_i^t|S_i^{t-1})\}_{i=1,t=2}^{i=n,t=24}$ the conditional probability table for the sensor i in layer t. First we calculate the probabilities using only the data from the alarm detection, following the same procedure detailed in Subsubsection 3.1.1. That is, first we apply the database transformation procedure (see Figure 6) and then we estimate the marginal $\{P(S_i^1)'\}_{i=1}^n$ and conditional $\{P(S_i^t|S_i^{t-1})'\}_{i=1,t=2}^{i=n,t=24}$ probability tables by using Laplace smoothing [17]. Finally, we combine both parameters to get the updated probability tables ($\{P_u(S_i^1)\}_{i=1}^n$ and $\{P_u(S_i^t|S_i^{t-1})\}_{i=1,t=2}^{i=n,t=24}$) using a weighted average as it is shown in Eq. (4) and Eq. (5):

$$\{P_u(S_i^1)\}_{i=1}^n = \alpha \cdot \{P(S_i^1)'\}_{i=1}^n + (1-\alpha) \cdot \{P(S_i^1)\}_{i=1}^n \tag{4}$$

$$\{P_u(S_i^t|S_i^{t-1})\}_{i=1,t=2}^{i=n,t=24} = \alpha \cdot \{P(S_i^t|S_i^{t-1})'\}_{i=1,t=2}^{i=n,t=24} + (1-\alpha) \cdot \{P(S_i^t|S_i^{t-1})\}_{i=1,t=2}^{i=n,t=24} \quad (5)$$

The parameter α determines the "memory" of the system about past readings. Its range is $[0-1]$. As it closes to 0, it gives more importance to the information about the past. Therefore, the model requires more time to be adapted to a new behaviour. On the contrary, as it closes to 1, the model tends to represent exclusively the recent behaviour of the monitored system, forgeting the past behaviour in a small window of time.

4. Simulated Case of Study

In this section we are going to test the predictive capability of the original methodology explained in [22] and compare it with our proposed DSS. In order to do that, we are going to generate synthetic data representing different scenarios. Our aim is to test the following situations. The system has been trained using data in abscence of failures. Then:

- The behaviour does not change, so no alarms should be triggered.
- The behaviour suddenly changes. Alarms should be triggered, but two scenarios can be tested in this case depending on the given feedback: It comes from an anomaly, or from a change in the behaviour of the monitored system.

For that, we have designed a simulated environment with four different sensors usually presented in industrial machinery: *Temperature (T)*, *Humidity (H)*, *Vibration (V)* and *Active Power (AP)*. To generate the synthetic data from that environment, we have modeled its behaviour using a Bayesian network $\mathscr{B}_b$ and then we have generated samples from it. In order to represent a change in the behaviour of the monitored system, we have used two alternatives:

- Keep the Bayesian network structure of $\mathscr{B}_b$ but change its parameters.
- Change the structure of the Bayesian network $\mathscr{B}_b$.

Next we are going to explain in detail the models and the process to generate the synthetic data from them.

The first model $\mathscr{B}_b$, used to generate the initial behaviour and the first alternative, is shown in Figure 7. We have set a direct dependence between Active Power and Temperature, and also between Humidity and Vibration. These dependences have been replicated in the 24 hourly layers, that is from [0:00-1:00) to [23:00-0:00). Regarding the domain for each variable, we have considered that the four variables take values in the set $\{Low, Medium, High\}$. Therefore, as we deal directly with discrete values instead of real numbers, there is no need to apply the discretization stage.

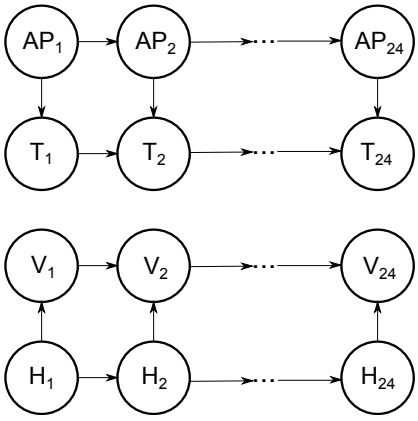

Fig. 7. Graphical structure ($\mathscr{B}_b$) for the simulated case of study

After setting the structure of the model, we have parameterized it into two different ways:

- **Basic behaviour**, which represents the usual machinery functioning ($\mathscr{W}_b$). In this case, $\mathscr{B}_b$ has been parameterized in the following way: all sensors tend to generate the *Low* value with more probability than the *Medium* one, and this *Medium* with more probability than the *High* one. Furthermore, and due to the dependences in the graphical model, Vibration and Temperature tend to follow the measures of Humidity and Active Power respectively, and each sensor tends to follow its own measure in the previous layer (time). To see the specific values, see Table B.1 in the Appendix B.
- **Alternative behaviour 1**, which represents a situation in which we detect more vibration than usu-

ally ($\mathscr{W}_{a1}$). In this case, the network $\mathscr{B}_b$ is parameterized in a different way: Temperature, Humidity and Active Power have a similar behaviour as in basic behaviour ($\mathscr{W}_b$), but Vibration will tend to get higher values. Specific values for Vibration variables are shown in the Appendix B, Table B.2.

The second model structure, $\mathscr{B}_a$ is similar to $\mathscr{B}_b$ but any dependence of the variable *Vibration* has been removed (see Figure 8). The parametrization of this network is as follows:

- **Alternative behaviour 2**, which represents a situation in which vibration readings follow the uniform distribution ($\mathscr{W}_{a2}$). Therefore, we take ($\mathscr{B}_a$) as basic structure and parameterized it according to the following description: All the states of Vibration have the same probability while the remaining probability distributions are set as in the basic behaviour ($\mathscr{W}_b$).

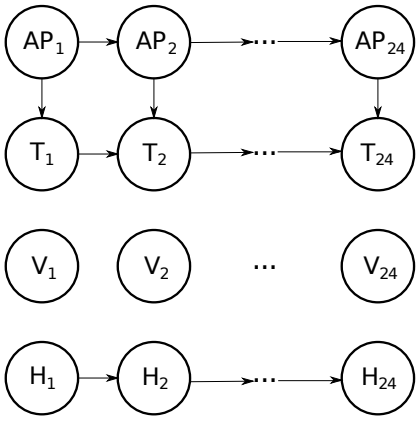

Fig. 8. Alternative graphical structure ($\mathscr{B}_a$) for the simulated case of study

In order to obtain the datasets for the simulations, the models are sampled by layers (first $t = 1$, then $t = 2$, etc.) and inside each layer, a probabilistic logic sampling [8] is guided by a topological ordering (e.g. AP, H, T, V). We consider two cases:

- Initial time slice ($t = 1$). First, variables without parents in the network are (independently) sampled from their marginal distribution. That is, $P(AP_1)$ and $P(H_1)$ for $\mathscr{W}_b$ and $\mathscr{W}_{a1}$, and $P(AP_1)$, $P(H_1)$ and $P(V_1)$ for $\mathscr{W}_{a2}$. Once the values for these variables are known (call them ap, h and v), the rest are sampled from the marginal distributions: $P(T_1|AP_1 = ap)$ and $P(V_1|H_1 = h)$ for $\mathscr{W}_b$ and $\mathscr{W}_{a1}$, and $P(T_1|AP_1 = ap)$ for $\mathscr{W}_{a2}$.
- Rest of time slices ($t \geqslant 2$). Now the values for all the variables in the time slice $t-1$ are known, therefore the marginal distribution to be sampled by order are: (1) $P(AP_t|AP_{t-1} = ap_{t-1})$, $P(H_t|H_{t-1} = h_{t-1})$, $P(T_t|AP_t = ap, T_{t-1} = t_{t-1})$ and $P(V_t|H_t = h, V_{t-1} = v_{t-1})$ for $\mathscr{W}_b$ and $\mathscr{W}_{a1}$; (2) $P(V_t)$, $P(AP_t|AP_{t-1} = ap_{t-1})$, $P(H_t|H_{t-1} = h_{t-1})$ and $P(T_t|AP_t = ap, T_{t-1} = t_{t-1})$ for $\mathscr{W}_{a2}$.

From these models we have sampled four datasets. Each one contains 4320 readings ($\langle ap, t, v, h\rangle$), corresponding to 6 months, 30 days per month and one reading every hour.

- BasicT. This dataset is sampled from the Basic behaviour model $\mathscr{W}_b$ and will be used to Train the correct behaviour model ($\mathscr{M}_c$).
- BasicV. This dataset is sampled from the Basic behaviour model $\mathscr{W}_b$ and will be used to validate the correct behaviour model ($\mathscr{M}_c$).
- AlternativeU. This dataset is sampled from the Alternative behaviour model 1 $\mathscr{W}_{a1}$ and will be used in two different ways, to update the correct and failure behaviour models ($\mathscr{M}_c$ and $\mathscr{M}_f$). In other words, telling the DSS that alarms correspond to a change in the behaviour of the system or to an anomaly.
- AlternativeR. This dataset is sampled from the Alternative behaviour model 2 $\mathscr{W}_{a2}$ and will be used as well in two different ways, to update the correct (change in the behavuour of the system) and failure (anomaly) behaviour models ($\mathscr{M}_c$ and $\mathscr{M}_f$).

4.1. Simulation data

In this section we will discuse how the two models $\mathscr{M}_c$ and $\mathscr{M}_f$ and their behaviours evolve during the simulation. The parameters used are $w = 24$ for the time window, $t_{conf} = t_{rcf} = 1.0$ for the alarm thresholds and $\alpha = 0.5$ for updating the models. Note that even if $w = 24$, in this experimentation the nodes in the last layer of the Bayesian networks are not connected to the nodes in the first layer. That means that when $t = 1$ it only uses data from the first hour

of the day, while when $t = 24$ it uses the information of the whole day (it can be seen as a variable window size, from $w = 1$ to $w = 24$). The experiment is as follows.

Firstly, we create the structure of both models $\mathscr{M}_c$ and $\mathscr{M}_f$ as described in Section 3, that is, dependence relations between sensors in the same layer are not included, because we suppose that we do not have such domain information. Of course, if these relations (or others) are available as problem domain knowledge, they can be added to the graphical structure. Then, we use BasicT dataset to learn the parameters for $\mathscr{M}_c$, that is, $P(AP_1), P(T_1), P(H_1), P(V_1)$ and $P(AP_t|AP_{t-1}), P(T_t|T_{t-1})$, $P(H_t|H_{t-1})$, $P(V_t|V_{t-1})$. In the case of $\mathscr{M}_f$ these distributions are initialized at random (uniform distribution).

After $\mathscr{M}_c$ and $\mathscr{M}_f$ have been built, five different situations are tested: BasicV is used to validate the correct behaviour model ($\mathscr{M}_c$); AlternativeU is interpreted first as normal behaviour and afterwards as anomalous behaviour, in order to check the adaptability of the system; Finally, the same experimentation is done with the data set AlternativeR.

For all the experiments we show three graphics. The first two correspond to the measures (per hour) for $conf$ and rcf respectively, while the last one is the number of detected anomalies per day by our proposal, and so the number of alarms sent. For the first and second graphics we show the first 1440 (two months) measures instead all the 4320 (the whole six months) because graphics get clear and the change in the data trend is inappreciable thenceforth.

4.1.1. BasicV *as correct behaviour*

The dataset BasicV is used to test the proposal. As BasicV comes from the same distribution as BasicT, the process should detect few anomalies, and so the number of *alarms* sent should also be small. In this case, as we know the inputs (sensors readings) correspond to *correct* machinery functioning, the operator will identify the alarm as *false* and model $\mathscr{M}_c$ will be accordingly updated/refined.

As we can observe in Figure 9, the proposed process works properly and the number of alarms stays low or even decreases as the days go on and the model is refined. In this case, both measures, $conf$ and rcf, give the maximum value in the early hours of the day and the minimum at the last one. This is because at the beginning of the day is when these measures takes the lowest amount of information, as we only use readings from the same day. Because of that, as the day progresses and more information can be used, both $conf$ and rcf go to their lowest values. As both measures have a similar behaviour, the performance of the initial methodology and our proposal would be very similar.

4.1.2. AlternativeU *as correct behaviour*

The dataset AlternativeU is now used to test the proposal, but interpreting it as a change in the operation mode of the machinery. That is, something in the functioning, environmental condition, etc. has changed, which produces the differences in the sensors readings regarding the data used for training, however, each time an *alarm* is sent, the operator marks it as correct behaviour (i.e. a false positive).

As we can observe in Figure 10, the number of anomalies detected is very low even at the first days. This is because the model $\mathscr{M}_c$ is quickly updated/refined according to the *false* anomalies detected, and the new data is understood as normal behaviour. It is worthpointing that only vibration readings has changed with respect to the basic behaviour (BasicV), and this change consists on higher values for these readings over the time. Therefore, the most relevant change in the probability distributions will lies in $P(V_1)$, because in the following layers (as well as for the basic behaviour BasicV) sensors tend to follow their own measures in the previous layer.

Finally, again as both measures have a similar behaviour, the performance of the DSS would be very similar, whether we use the proposed improvement or not.

4.1.3. AlternativeU *as anomalous behaviour*

Now we use the same dataset AlternativeU but understanding its readings as failures. Thus, we start with the model $\mathscr{M}_c$ trained with BasicT data. As in this case all the alarms sent by the algorithms are

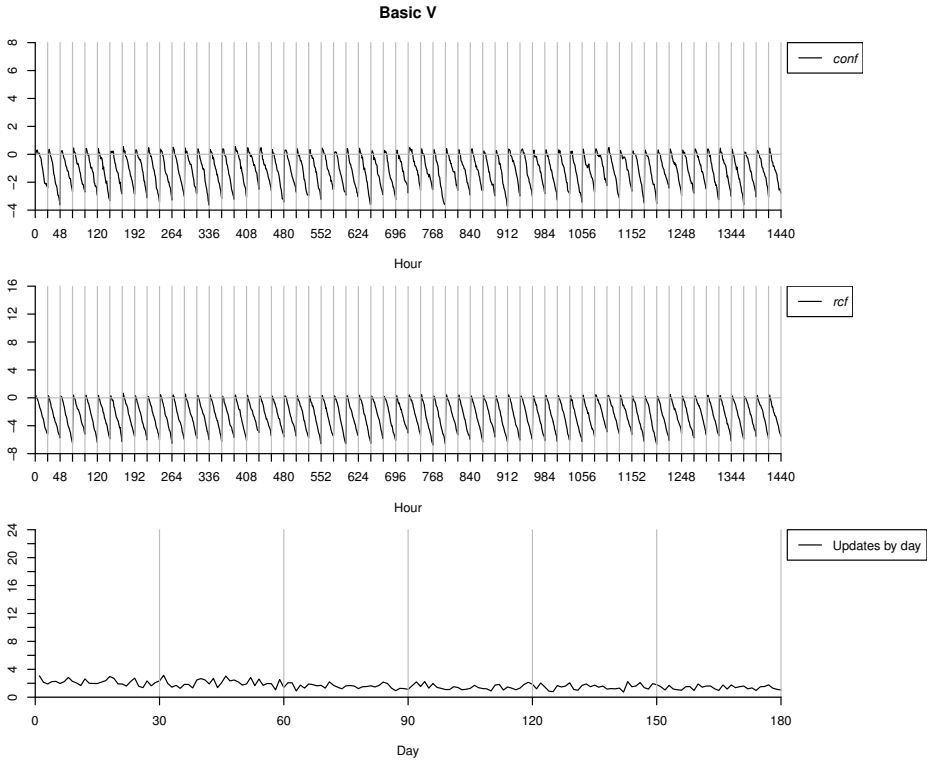

Fig. 9. Testing process over BasicV. Same operation mode.

confirmed as anomalies by the operator, the updated model is $\mathscr{M}_f$ and not $\mathscr{M}_c$.

As we can observe in Figure 11, in the first days the number of anomalies detected is small. However, after a few days, when $\mathscr{M}_f$ is refined, almost all the (24) readings are classified as anomalies. We can see that the values of $conf$ are similar to those in the previous case, where AlternativeU is understood as normal behaviour (just in the early hours of the day values tend to be higher). This is due to $conf$ detects anomalies paying attention to the dependencies between variables, and as we said above, the model $\mathscr{M}_c$ learnt that sensors tend to follow their own measure in the previous layer, so if vibration readings are high it will consider the most probable next reading will be also a high value. On the contrary, once $\mathscr{M}_f$ is refined, rcf gets higher values as $P_{\mathscr{M}_f}(\mathbf{e}) \geqslant P_{\mathscr{M}_c}(\mathbf{e})$.

In this case, the measure $conf$ does not detect any anomalous behaviour. However, once the first triggered alarms are marked as failures, rcf starts to identify correctly the new ones, and is directly responsible of the increase in the number of alarms sent. In this case, our proposal is able to adapt the new situation and classify the new instances as anomalous behaviour while the methodology which only uses the measure $conf$ is not.

4.1.4. AlternativeR *as correct behaviour*

The dataset AlternativeR is now used to test the proposal, but interpreting it as a change in the operation mode of the machinery. That is, something in the functioning, environmental condition, etc. has changed, which produces the differences in the sensors readings regarding the data used for training, however, each time an *alarm* is sent, the operator mark is as correct behaviour.

As we can observe in Figure 12 the number of anomalies detected is high at the first days, while this number decreases as the model $\mathscr{M}_c$ is updated, and the new data is understood as normal behaviour. This is because now we are in a more complex situation than when using AlternativeU as correct

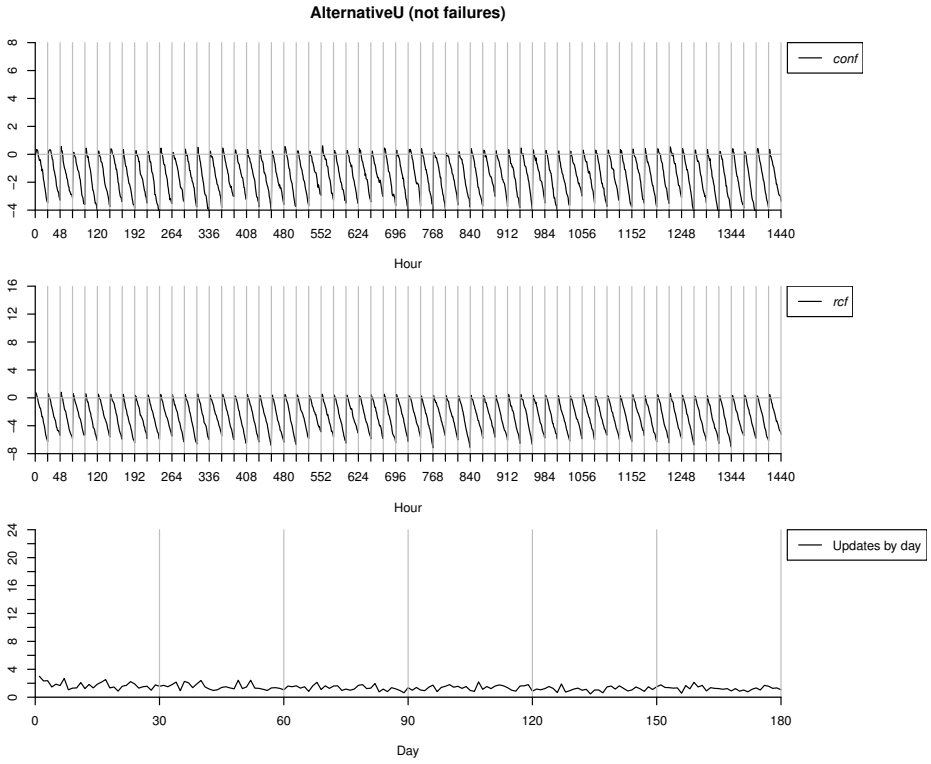

Fig. 10. Testing process over AlternatuveU interpreted as no failures.

behaviour. Now, apart from updating $P(V_1)$, also $P(V_t|V_{t-1})$ needs to be *re-trained* in order to incorporate the behavioural change in $\mathscr{M}_c$.

4.1.5. AlternativeR *as anomalous behaviour*

Finally we use the same dataset AlternativeR but understanding it as failures. Thus, we start with the model $\mathscr{M}_c$ trained with BasicT data. As in this case all the alarms sent by the algorithms are confirmed as anomalies by the operator, the updated model is $\mathscr{M}_f$ and not $\mathscr{M}_c$.

As we can observe in Figure 13, in the first days almost all the (24) readings are classified as anomalies. It is worthpointing that, even if it looks like both measures $conf$ and rcf have the same importance in this case, the first measure has more importance. If we pay attention to the first day, we can see that rcf follow the tend of $conf$. What is really happening is that first $conf$ detects the anomaly (but not rcf), and after a few updates of $\mathscr{M}_f$ then rcf will be able to detect as well as $conf$ the anomalies (but no before these first updates). However, our proposal uses a combination of both measures. Therefore all the cases are detected correctly as failures, so the performance of both methodologies would be quite similar.

5. Conclusions

We have designed a general and robust decision support system tool for health management in industrial environments. The core of the system is a probabilistic expert system based on dynamic Bayesian networks. Fault detection is based on both conflict analysis and likelihood-ratio test.

Different types of failures has been tested, and due to the use of two measures to trigger alarms, they have been correctly detected. It is worth pointing that the second measure based on likelihood-ratio test only affects directly in one of the tested cases. However, it is an important case because it could represent a change in the usual operation

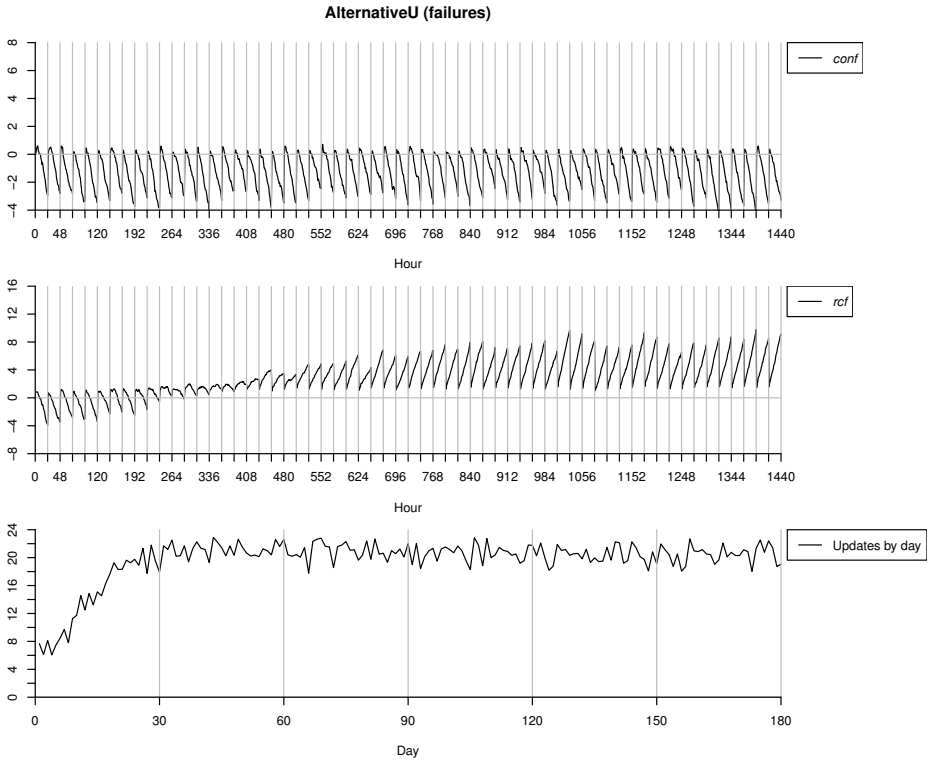

Fig. 11. Testing process over AlternatuveU interpreted as failures.

mode of the monitored system. Additionally, although dependences between sensors are not considered by default, if some knowledge about the problem is available it can be included as a consequence of using Bayesian networks for modelling.

The expert system-based application has been implemented using multi-platform technology, so it can be deployed on any operative system. In order to avoid problems derived from editing the system configuration in parallel, we only allow one person to be editing the the system description at the same time. Because of that, it is recommendable that only one person would be the manager of the system.

Finally, even if the tool can be used on any kind of system, the time window w and the thresholds used to trigger alarms have to be fixed by an expert in order to obtain a good performance.

Acknowledgments

This work has been partially funded by FEDER funds, the Spanish Government and JCCM through projects TIN2013-46638-C3-3-P, TSI-020100-2011-140 and PEII-2014-049-P. Javier Cózar is also funded by the MICINN grant FPU12/05102.

Appendix A Application

The designed software is used as a DSS, so it can be used for both monitoring data and check if the system might be failing or not through the predictions. It follows the web-like client/server model: the information is managed and stored in a centralized system (the server) and clients can access to this information on demand.

On the server side we have two components, which can be deployed in the same machine or not. These are the database server and the web page server. The first one stores the data provided by the sensors, while the web page server provides a

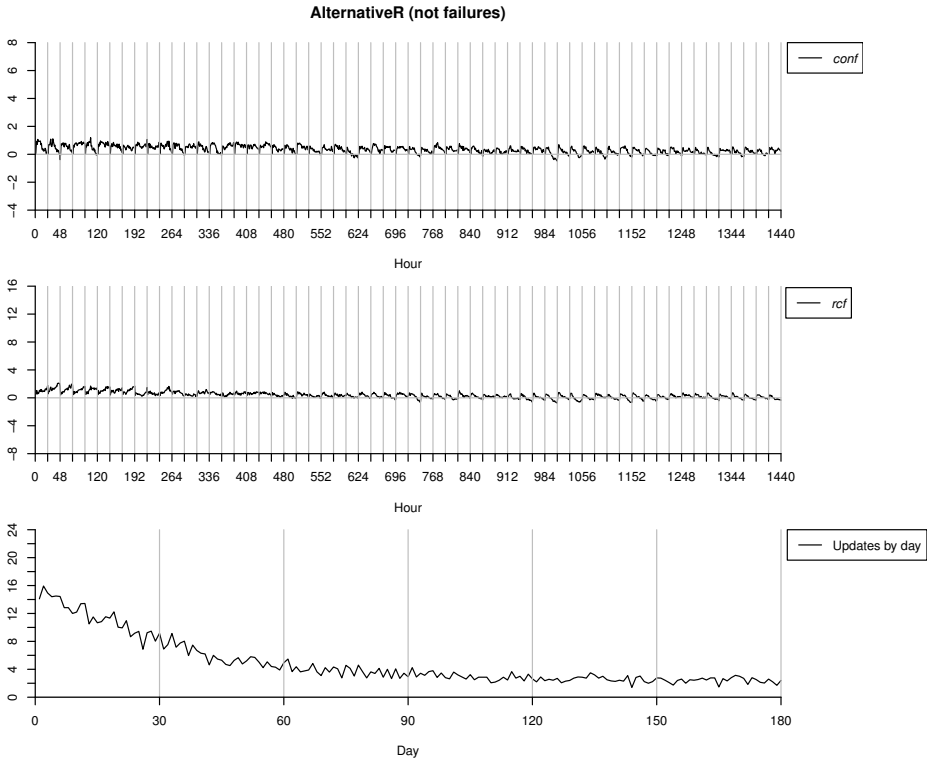

Fig. 12. Testing process over AlternatuveU interpreted as no failures.

web interface for clients to use the application. The technology used to construct the models and make predictions is JAVA, and PHP to generate the web page and to implement web services (used by clients throughout AJAX). There are also some XML files to store information about the monitored system and preferences that clients can configure.

On the client side we use HTML5 plus Javascript to generate the webpage. It also will use AJAX to dynamically load the requested data.

The monitored system is logically divided into a hierarchical structure (see Figure A.1). Motes are the basic components which represent the physical sensors defined by the way we can access to their measures. To give a flexible abstraction layer, the way we can access to those measures is through a database. Hence, physical sensors send their measures to a server in charge of storing the data in a database. Because of that, there is a small delay introduced between the sensor readings and its processing in our DSS. However, for the scope of this application, this delay is assumable.

Machines represent a whole working unit, formed by a set of motes. It is not required that the sets of motes are disjoint. This allows the user to specify physical working units (physical machines with their associated sensors) as well as logical working units (a set of components in charge of some specific tasks, which might be shared between different physical machines).

Operations are associated with machines. They are represented by a subet of sensors (motes) from the associated machine. This is useful because sometimes is not desirable to monitor all the sensors of a particular machine, i.e: if we know the activity of a machine under supervision (cutting, polishing, etc.) we probably would prefer to monitor only the sensors allocated in the module in charge of doing that operation.

In Figure A.2 we can see the interface to manage the motes configuration. Throughout this interface we can specify the whole set of sensors used in our

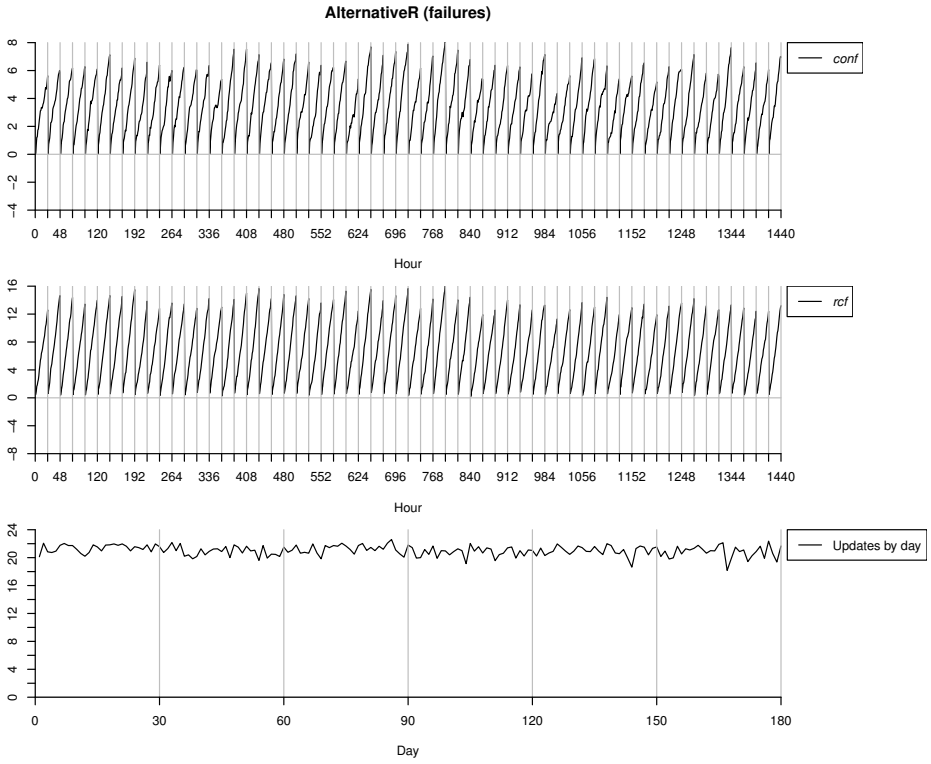

Fig. 13. Testing process over AlternativeU interpreted as failures.

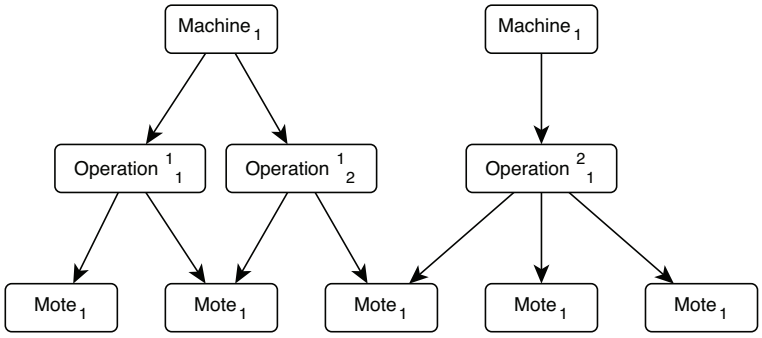

Figure A.1: Logical hierarchical structure of the monitored system.

environment.

Respect to the visualization, the interface is divided into two parts. The first one, designed to manage machine and operation definitions, as well as to monitor the sensor readings. The second part of this interface corresponds to the health status prediction.

We can see the first part in Figure A.3. For the management, we can select the desired machine or operation from their respective drop list and use the edit or remove buttons. To add new machines or operations, the proper option appear in the drop lists.

For the monitorization, the requested data is plotted in two different widgets: a speedometer and a timeline. The first one is used to show the last measure while the last is used to see the trend. In timeline widgets we can define intervals and associate

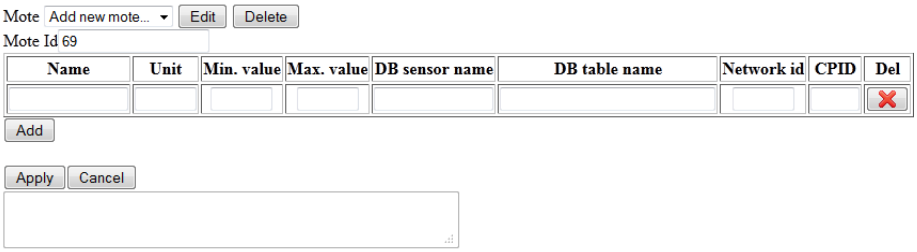

Fig. A.2. Example of use: Motes management.

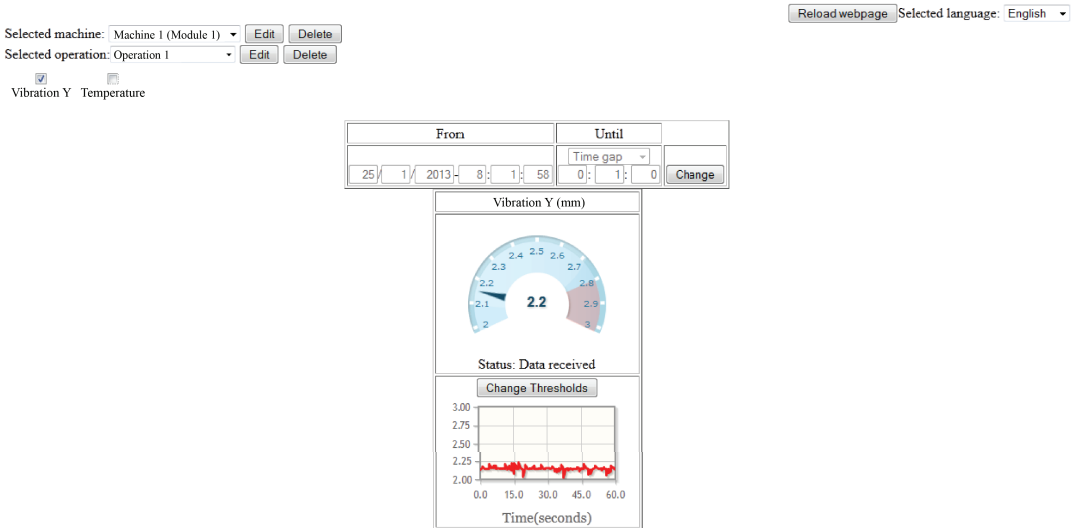

Fig. A.3. Example of use: monitoring data.

colors to each one. Finally, we can set a time window to select the data to be monitored. Every fixed amount of time (specified in a configuration XML file) this window time will go forward the same period of time.

The second part of this interface corresponds to the health status prediction (see Figure A.4). We can learn the models specifying a period of time (data in that interval will be used to build such models), or delete them in order to re-learn later. Once the model has been learnt we can see the measures given by the functions $conf(\mathbf{e})$ and $rcf(\mathbf{e})$ described in section 3 throughout two timeline plots. When the interpretation of these formulas means a failure alert, this information is shown in the table below.

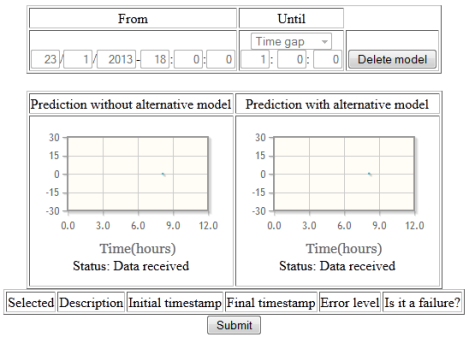

Fig. A.4. Example of use: health status prediction.

Appendix B Bayesian network parameters

In this appendix we are going to detail the parameters for the BN $\mathscr{B}_b$. Note that the parameters for the BN $\mathscr{B}_a$ are the same but those for the vari-

Table B.1: Parameters for nodes in the Bayesian network $\mathscr{B}_b$.

(a) Parameters for nodes without ascendants ($X \in \{AP_1, H_1\}$).

	Low	*Medium*	*High*
$P(X)$	0.600	0.300	0.100

(b) Parameters for nodes with only one ascendant ($\forall t > 1, X \in \{T_1, V_1, AP_t, H_t\}$).

Parent(X)	*Low*	*Medium*	*High*
$P(X = Low)$	0.750	0.200	0.050
$P(X = Medium)$	0.500	0.400	0.100
$P(X = High)$	0.350	0.450	0.200

(c) Parameters for nodes with two ascendants ($\forall t > 1, X \in \{T_t, V_t\}$).

X_{t-1}	*Low*			*Medium*			*High*		
Parent(X_t)	*Low*	*Medium*	*High*	*Low*	*Medium*	*High*	*Low*	*Medium*	*High*
$P(X_t = Low)$	0.930	0.066	0.004	0.815	0.174	0.011	0.724	0.248	0.028
$P(X_t = Medium)$	0.815	0.174	0.011	0.595	0.381	0.024	0.467	0.480	0.053
$P(X_t = High)$	0.724	0.248	0.028	0.467	0.480	0.053	0.335	0.555	0.110

Table B.2: Parameters for Vibration nodes in the Bayesian network $\mathscr{B}_b$ for the Alternative behaviour 1.

(a) Parameters for V_1.

H_1X	*Low*	*Medium*	*High*
$P(V_1 = Low)$	0.029	0.194	0.777
$P(V_1 = Medium)$	0.010	0.198	0.792
$P(V_1 = High)$	0.004	0.123	0.873

(b) Parameters for V_t where $t > 1$.

V_{t-1}	*Low*			*Medium*			*High*		
H_t)	*Low*	*Medium*	*High*	*Low*	*Medium*	*High*	*Low*	*Medium*	*High*
$P(V_t = Low)$	0.220	0.390	0.390	0.086	0.457	0.457	0.040	0.345	0.614
$P(V_t = Medium)$	0.086	0.457	0.457	0.030	0.485	0.485	0.014	0.355	0.631
$P(V_t = High)$	0.040	0.346	0.614	0.014	0.355	0.631	0.006	0.234	0.760

able Vibration, which are 0.33 for each one of its three possible values (*Low*, *Medium* and *High*).

In Table B.1 we show the parameters for the network $\mathscr{B}_b$. There are three tables. In the first one (Table B.1a) we show the parameters for nodes without parents, that is AP_1 and H_1. In Table B.2 we show the parameters for nodes with only one ascendant, which are $\forall t > 1, X \in \{T_1, V_1, AP_t, H_t\}$. Parent($T_1$) refers to AP_1, Parent(V_1) to H_1, Parent(AP_t) to AP_{t-1} and Parent(H_t) to H_{t-1}. Finally, the third table (Table B.1c) shows the parameters for nodes with exactly two ascendants, which are $\forall t > 1, X \in \{T_t, V_t\}$. Parent($T_t$) refers to AP_t and Parent(V_t) to V_t.

In Table B.2 we show the parameters for $\mathscr{B}_b$ used exclusively for the Alternative behaviour 1. It contains two tables. In Table B.2a we show the parameters for V_1, while Table B.2b shows the parameters for V_t where $t > 1$.

References

1. C. Athanasopoulou and V. Chatziathanasiou, *Intelligent system for identification and replacement of faulty sensor measurements in thermal power plants (ippamas: Part 1)*, (Expert Systems With Applications, 36(5):8750–8757, 2009).
2. V. Chandola, A. Banerjee and V. Kumar, *Anomaly detection: A survey*, (ACM Comput. Surveys, 41(3):15:1–15:58, 2009).
3. S.-P. Cheon, S. Kim, S.-Y. Lee, C.-B. Lee, *Bayesian networks based rare event prediction with sensor data*, (Knowledge-Based Systems, 22(5):336 – 343, 2009).
4. A. Darwiche, *Modeling and reasoning with Bayesian networks*, (Cambridge University Press, 2009).
5. M. C. Garcia, M. A. Sanz-Bobi and J. del Pico, *SIMAP: Intelligent system for predictive maintenance: Application to the health condition monitoring of a windturbine gearbox*, (Computers in Industry, 57(6):552–568 2006).
6. E. Gilabert and A. Arnaiz, *Intelligent automation systems for predictive maintenance: A case study*, (Robotics and Computer-Integrated Manufacturing, 22(5):543–549, 2006).
7. D. Heckerman, D. Geiger and D. M. Chickering, *Learning Bayesian networks: The combination of knowledge and statistical data*, (Machine Learning, 20(3):197–243, 1995).
8. M. Henrion, *Propagating uncertainty in Bayesian networks by probabilistic logic sampling*, (In Uncertainty in Artificial Intelligence 2 Annual Conference on Uncertainty in Artificial Intelligence (UAI-86), pages 149–163. Elsevier Science, 1986).
9. D. J. Hill, B. S. Minsker and E. Amir, *Real-time Bayesian anomaly detection in streaming environmental data*, (Water Resources Research, 45(4):1–16, 2007).
10. F. V. Jensen, B. Chamberlain, T. Nordahl and F. Jensen, *Analysis in hugin of data conflict*, (Uncertainty in Artificial Intelligence, volume 6, pages 519–528. Elsevier, 1991).
11. F. V. Jensen and T. D. Nielsen, *Bayesian networks and decision graphs*, (Springer, 2nd edition, 2007).
12. U. B. Kjaerulff and A. L. Madsen, *Bayesian networks and influence diagrams: A guide to construction and analysis*, (Springer Publishing Company, Incorporated, 1st edition, 2010).
13. D. Koller and N. Friedman, *Probabilistic graphical models: Principles and techniques*, (The MIT Press, 2009).
14. K. Korb and A. E. Nicholson, *Bayesian artificial intelligence*, (CRC Press, Inc., Boca Raton, FL, USA, 2003).
15. A. Kusiak and W. Li, *The prediction and diagnosis of wind turbine faults*, (Renewable Energy, 36(1):16–23, 2011).
16. H. Liu, F. Hussain, C. L. Tan and M. Dash, *Discretization: An enabling technique*, (Data Mining and Knowledge Discovery, 6(4):393–423, 2002).
17. C. D. Manning, and H. Schütze, *Foundations of statistical natural language processing*, (MIT press, 1999).
18. N. Mehranbod, M. Soroush and C. Panjapornpon, *A method of sensor fault detection and identification*, (Journal of Process Control, 15(3):321 – 339, 2005).
19. A. Muller, M.-C. Suhner and B. Iung, *Formalisation of a new prognosis model for supporting proactive maintenance implementation on industrial system*, (Reliability Engineering & System Safety, 93(2):234–253, 2008).
20. K. Murphy, *Dynamic Bayesian Networks: Representation, Inference and Learning*, (PhD thesis, UC Berkeley, Computer Science Division, 2002).
21. R. E. Neapolitan, *Learning Bayesian networks*, (Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003).
22. T. D. Nielsen and F. V. Jensen, *On-line alert systems for production plants: A conflict based approach*, (International Journal of Approximate Reasoning, 45(2):255–270, 2007).
23. J. Pearl, *Probabilistic reasoning in intelligent systems: Networks of plausible inference*, (Morgan Kaufmann Publishers Inc, 1988).
24. M. Pecht, *Prognostics and health management of electronics*, (Wiley Online Library, 2008).
25. J. Rabatel, S. Bringay and P. Poncelet, *Anomaly detection in monitoring sensor data for preventive maintenance*, (Expert Systems with Applications,

38(6):7003–7015, 2011).
26. S. Verron, T. Tiplica and A. Kobi, *Fault diagnosis of industrial systems by conditional Gaussian network including a distance rejection criterion*, (Engineering Applications of Artificial Intelligence, 23(7):1229 – 1235, 2010).
27. G. Weidl, A. Madsen and S. Israelson, *Applications of object-oriented Bayesian networks for condition monitoring, root cause analysis and decision support on operation of complex continuous processes*, (Computers & Chemical Engineering, 29(9):1996 – 2009, 2005).
28. B. G. Xu, *Intelligent fault inference for rotating flexible rotors using Bayesian belief network*, (Expert Systems with Applications, 39(1):816 – 822, 2012).
29. S. Jakubek and T. Strasser, *Fault-diagnosis using neural networks with ellipsoidal basis functions*, (American Control Conference, 5:3846–3851, 2002).
30. G. Williams, R. Baxter, H. He, S. Hawkins and L. Gu, *A comparative study of RNN for outlier detection in data mining*, (IEEE, 709–709, 2002).
31. C. De Stefano, C. Sansone and M. Vento, *To reject or not to reject: that is the question-an answer in case of neural classifiers*, (Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 30:849–94, 2000).
32. D. Barbará, J. Couto, S. Jajodia and N. Wu, *ADAM: a testbed for exploring the use of data mining in intrusion detection*, (ACM Sigmod Record, 30(4):15–24, 2001).
33. M. Ester, H. Kriegel, J. Sander and X. Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*, (Kdd,96(34):226–231, 1996).
34. L. Ertöz, M. Steinbach and V. Kumar, *Finding topics in collections of documents: A shared nearest neighbor approach*, (Clustering and Information Retrieval, 83–84, 2003).
35. Smith, Bivens, Embrechts, Palagiri and Szymanski, *Clustering approaches for anomaly based intrusion detection*, (Proceedings of intelligent engineering systems through artificial neural networks, 579–584, 2002).
36. M. Ramadas, S. Ostermann and B. Tjaden, , *Detecting anomalous network traffic with self-organizing maps*, (Recent Advances in Intrusion Detection, 36–54, 2003).
37. A. Pires and C. Santos-Pereira, *Using clustering and robust estimators to detect outliers in multivariate data*, (Proceedings of the International Conference on Robust Statistics, 2005).
38. M. Otey, S. Parthasarathy, A. Ghoting, G. Li, S. Narravula and D. Panda, *Towards nic-based intrusion detection*, (Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 723–728, 2003).
39. Das and Schneider, *Detecting anomalous records in categorical datasets*, (Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, 220–229, 2007).
40. D. Barbara, N. Wu and S. Jajodia, *Detecting Novel Network Intrusions Using Bayes Estimators*, (SDM, 1–17, 2001).
41. Yeung and Chow, *Parzen-window network intrusion detectors*, (Pattern Recognition, 2002. Proceedings. 16th International Conference on, 4:385–388, 2002).
42. L. Eskin and Stolfo, *Modeling system calls for intrusion detection with dynamic window sizes*, (DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings, 1:165–175, 2001).
43. Agarwal, *An empirical bayes approach to detect anomalies in dynamic multidimensional arrays*, (Data Mining, Fifth IEEE International Conference on, 8–pp, 2001).
44. Abraham and Chuang, *Outlier detection and time series modeling*, (Technometrics, 31(2):241–248, 1989).
45. Solberg and Lahti, *Detection of outliers in reference distributions: performance of Horns algorithm*, (Clinical chemistry, 51(12):2326–2332, 2005).
46. Breunig, Kriegel, Ng and Sander, *Optics-of: Identifying local outliers*, (Principles of data mining and knowledge discovery, 262–270, 1999).
47. Byers and Raftery, *Nearest-neighbor clutter removal for estimating features in spatial point processes*, (Journal of the American Statistical Association, 93(442):577–584, 1998).
48. P.A. Aguilera, A. Fernández, F. Reche and R. Rumí, *Hybrid Bayesian network classifiers: application to species distribution models*, (Environmental Modelling & Software, 25(12):1630–1639, 2010).