# Using structural decomposition methods to design gray-box models for fault diagnosis of complex industrial systems: a beet sugar factory case study

Belarmino Pulido[1], Jesus Maria Zamarreño[2], Alejandro Merino[3], and Anibal Bregon [4]

[1,4] *Dept. de Informática, University of Valladolid, Valladolid, Spain*
*belar@infor.uva.es, anibal@infor.uva.es*

[2] *Depto. Ingeniería de Sistemas y Automática, University of Valladolid, Valladolid, Spain*
*jesusm@autom.uva.es*

[3] *Depto. Ingeniería Electromecánica, University of Burgos, Burgos, Spain*
*alejandromg@ubu.es*

## Abstract

Reliable and timely fault detection and isolation are necessary tasks to guarantee continuous performance in complex industrial systems, avoiding failure propagation in the system and helping to minimize downtime. Model-based diagnosis fulfils those requirements, and has the additional advantage of using reusable models. However, reusing existing complex non-linear models for diagnosis in large industrial systems is not straightforward. Most of the times the models have been created for other purposes different from diagnosis, and many times the required analytical redundancy is small. In this work we propose to use Possible Conflicts, which is a model decomposition technique, to provide the structure (equations, inputs, outputs, and state variables) of minimal models able to perform fault detection and isolation. Such structural information can be used to design a gray box model by means of state space neural networks. We demonstrate the feasibility of the approach in an evaporator for a beet sugar factory using real data.

## 1. Introduction

Prognostics and Health Management are very important tasks for continuous operation and to comply with safety requirements of large industrial systems. In such systems, monitoring and early fault diagnostics are also fundamental tasks to avoid fault effects propagation, to prevent failures, and to minimize downtime. Hence, reliable and fast fault detection and isolation are needed, providing additionally an accurate input for the prognostic stage.

Model-based reasoning provides different kinds of methods to fulfil those requirements. Model-based diagnosis uses a model of the system to estimate the proper behaviour and to compare with current observations in order to detect anomalies. In the last three decades model-based diagnosis has been approached by two different communities DX (Hamscher, Console, & Kleer (Eds.), 1992) –using Artificial Intelligence techniques–, and FDI (Gertler, 1998; Blanke, Kinnaert, Lunze, & Staroswiecki, 2006; Patton, Frank, & Clark, 2000) –based on Systems Theory and Control. Both communities provide different but complementary techniques, as demonstrated by recent works (Cordier, Dague, Lévy, Montmain, & Travé-Massuyès, 2004).

Our proposal elaborates on the similarities of both approaches and focus on consistency-based diagnosis using numerical models (Pulido, Alonso-González, & Acebes, 2001). Consistency-based diagnosis proceeds in three stages: first, fault detection is performed by detecting minimal conflicts in the system (minimal set of equations or components involved in predicting a discrepancy); second, fault isolation is achieved by computing the minimal hitting-sets of the conflicts; third, fault identification requires using fault models to predict the faulty behaviour (Reiter, 1987; Dressler & Struss, 1996), and rejecting those fault modes that are not consistent with current observations. In this work, we use Possible Conflicts (Pulido & Alonso-González, 2004), PCs for short, that are computed off-line and are the complete set of minimal redundant models that can become conflicts. PCs provide the structural model– equations, input, output, and state variables– that can be used for fault detection and isolation, or can be also used to simplify the fault prognostics stage (Daigle, Bregon, & Roychoudhury, 2011).

While using PCs we need to build off-line simulation or state-observer models (Pulido, Bregon, & Alonso-González, 2010) to track the subsystem behaviour. This step requires the analysis of the model, and sometimes to rewrite the original equations for diagnosis purposes. Main advantage of model-based diagnosis is reusing existing models, but this is also its main difficulty. Frequently the models were created for purposes different from diagnosis, and the required analytical redundancy in the system is small, due to the price of additional sensors, and because they allocation is related to process control. Both problems exist in large industrial systems where complexity comes from the highly non-linear models required to mimic system performance. Consequently, reusing existing non-linear models for diagnosis in those systems is not straightforward. We propose to use the structural information in each Possible Conflict to design different kind of executable models. In this work, where precise analytical models[1] is difficult to handle, we propose to build grey-box models based on a state space neural network architecture derived from that structural information.

Preliminary results in an evaporation unit for a beet sugar factory in Spain using real data show the feasibility of the approach. The system has slow dynamics and due to the high costs of the start-up mode, it should work for weeks uninterrupted. Main difficulty in the existing models comes from the number of unknown parameters to be identified in the model, and the presence of non-linearities that requires expert manipulation in order to derive diagnosis-oriented models. An additional problem to test any approach is that there is few information about faults actually happening. Hence, any feedback from the model-based diagnosis system will be very helpful for the system operators.

The organization of this paper is as follows. First, we present the real system to be studied. Second, we introduce the Possible Conflicts technique used to find minimal models. Third, we introduce the state space neural network approach to obtain grey box models for the Possible Conflicts. Next, we test the first principle and the neural network models in the case study, drawing some conclusions.

## 2. Description of the case study: an evaporation unit in a beet sugar factory

We will test our proposal in an evaporation station of a beet sugar factory. In such processes there are four main stages: diffusion, purification, evaporation and crystallization. Evaporation is the stage in which the water contained in a juice with low sugar concentration is evaporated in order to obtain higher sugar concentration. Afterwards, the resulting syrup is used to obtain sugar crystals in a set of vacuum pans. Figure 1 shows the main elements in an evaporation plant: the evaporation units.

Figure 1. Five evaporation units for the evaporation section in a beet sugar factory in Spain.

Each evaporator has two chambers. The heating chamber surrounds a set of vertical tubes that contain boiling juice. A flow of steam enters these chambers and transfers heat to the juice providing the energy needed for boiling. The steam condenses around the tubes and leaves the evaporator as condensate. The interior tubes, plus the evaporator upper and bottom spaces, it is named the juice chamber. A sugar solution of low concentration (juice) flows continuously into the base of the evaporator and starts boiling. Consequently, we get a solution of higher concentration at the output. The steam produced from the water evaporation reaches the upper space and leaves the juice chamber by a pipe at the top.

### 2.1. The simulation models

The simulated plant consists on a set of five effects interconnected through pipelines and valves. Each effect is formed by one or several evaporation units. The steam generated in one effect is used to provide energy to the heating chambers of the evaporators of the next effect, while the juice flows from one effect to another increasing the sugar concentration. In this multiple-effect arrangement only the first effect is fed with boilers steam and purified juice. In the last effect, the evaporated steam escapes from the juice chamber to the condensers and then to atmosphere.

The use of dynamic modelling and simulation techniques in the process industry is an activity mainly oriented towards the design of installations and the training of the working staff but it can be also used to test new control or diagnosis strategies. For this factory there is a training simulator developed at the University of Valladolid, Spain (Merino, Alves, & Acebes, 2005). The main console of the training simulator for the evaporation section is shown in Figure 2.

[1]Based on Physics first principles, usually a collection of ODEs.

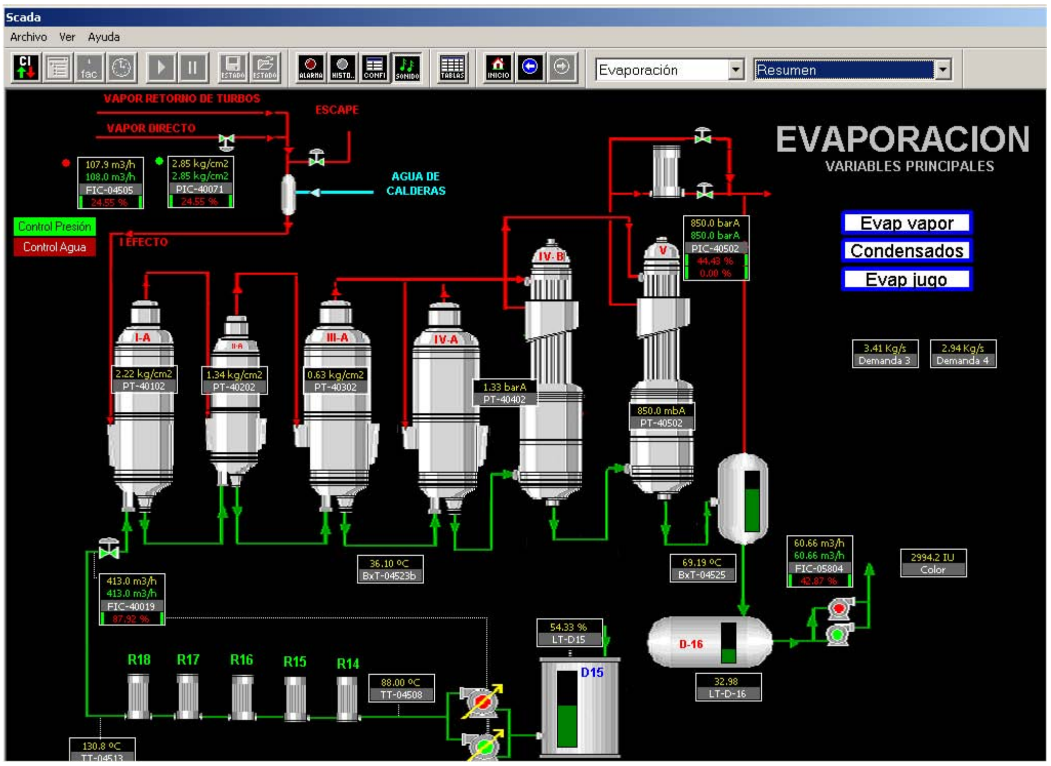


Figure 2. Schematic of the available simulator for training operators.

The simulator is articulated in two big systems: a simulation program and a distributed control system, where one of the control units works as an instructor console. The objective of the simulation program is to reproduce in the most reliable way the global dynamic performance of the process of sugar production. The simulation is made using the EcosimPro (EcosimPro, 2012) simulation language, and the model is developed using libraries of elemental units developed in an object oriented modelling approach (Acebes, Merino, Alves, & Prada, 2009). Additionally, the simulation code must work in real time and use an OPC (OLE for Process Control) interface (Alves, Normey-Rico, A., Acebes, & Prada, 2008) to communicate with the distributed control system. OPC is a de facto standard for communications on Windows applications in industrial processes and it is included in almost every modern SCADA. The OPC simulation program performs two tasks in parallel: solve the dynamic mathematical models of the process in real time, and attend requests from OPC clients. The SCADA system, that can be configured as operator or as instructor console, acts as an OPC client, receives data from the simulation, changes the boundary conditions and activate faults in the simulation program.

When building a model, the degree of complexity is variable depending on the use of the model. In the case of evaporation units, it is possible to use different approximations to the model (Luyben, 1990; Merino, 2008). In the training simulator, a detailed model is used, including dynamics in the liquid and vapour phase and complex phenomena such as accumulation of incondensable gases or the absence of juice in the evaporator, which allows steam flow via the juice pipes. These features are necessary in order to provide training capabilities to the simulator. As an example of the type of equations used in the simulator, energy balances to the juice chamber are shown:

$$\frac{dT_{jo}}{dt} = \frac{W_{jo}(H_{jo} - H_{jo}) - \frac{\partial H}{\partial T} W_{ji}(C_{si} - C_{so})}{m_t \frac{\partial H_{jo}}{\partial T_{jo}}} - \frac{E\left(H_E - H_{jo} + \frac{\partial H_{jo}}{\partial C_{so}} C_{so}\right)}{m_t \frac{\partial H_{jo}}{\partial T_{jo}}}$$

Where:

$$\frac{\partial H_{jo}}{\partial C_{so}} = -0.025104 \cdot T_{js} + 3.6939 \cdot 10^{-5} \cdot T_{jo}^2$$

$$\frac{\partial H_{jo}}{\partial T_{so}} = 4.06 - 0.025104 \cdot C_{so} + \left(5.418936 \cdot 10^{-4} \cdot C_{so}^2\right) T_{so}$$

Together with these equations, mass balances, heat transmission equations, state equations, etc., must be added for the liquid and vapour phases, resulting in a very complex nonlinear model. Furthermore, in the physical model, there are several parameters that must be adjusted dynamically. In the relatively simple case studied in this article, only four parameter were necessary. These were the set of equations that we need to analyse and modify in order to perform model-based diagnosis, to obtain from the model the value of a variable that is being measured, so that it is possible to compare both values. For this to occur, it is necessary that the number of measured variables in the process is sufficiently large to allow calculating one measured variable from other ones. On the other hand, there is no need of a match between the physical causality of the modelled system and the causality imposed by the availability of the measures. This involves the symbolic manipulation of the mathematical model, which is usually complex, even when using object oriented modelling languages that allow non-causal modelling. For example, in the case of evaporation, the juice level is a measured variable. This variable, from the point of view of the physical modelling, is a state variable that is calculated by numerical integration. In the case of fault diagnosis, this variable is a measured one that cannot be calculated by the model by integration without appearing a high index problem. This makes it necessary to manipulate the model so that the present binding disappears.

## 3. Possible Conflicts for structural model decomposition

### 3.1. Possible Conflicts

The computation of the set of Possible Conflicts (PCs) (Pulido et al., 2001; Pulido & Alonso-González, 2004) is a system model decomposition method from the DX community, which searches for the whole set of submodels of a given model with minimal redundancy (the number of equations in the submodel equals the set of unknown variables plus one). PCs provide the minimal analytical redundancy neccesary to perform fault diagnosis. PCs are computed off line and they can be used on line to perform consistency based diagnosis of dynamic systems. PCs also provide the computational structure of the constraints that generate redundancy. This structure can be used to build a simulation model, or –as we will show later– to obtain the structure of a state space neural network.

Off-line PCs computation requires three steps:

1. To generate an abstract representation of the system as a hypergraph. The nodes of the hypergraph are system variables and the hyperarcs represent constraints between these variables. These constraints are abstracted from the equations that relate system variables.

2. To derive *Minimal Evaluation Chains* (MECs), which are minimal connected over constrained subsystems. The existence of a MEC is a necessary condition for analytical redundancy to exist. MECs have the potential to be solved using local propagation (solving one equation in one unknown) from the measurements.

3. To generate *Minimal Evaluation Models* (MEMs) assigning causality[2] to the constraints of the MEC. MEMs are directed hypergraphs that specify the order in which equations should be locally solved starting from measurements to generate the subsystem output.

In Consistency-based diagnosis (Reiter, 1987; Kleer & Williams, 1987) a conflict arises given a discrepancy between observed and predicted values for a variable. Hence, conflicts are the result of the fault detection stage. But they also contain the necessary structural information for fault isolation. Possible Conflicts were designed to compute off-line those subsystems capable to become minimal conflicts online. Under fault conditions, conflicts are observed when the model described by a MEM is evaluated with available observations, because the model constraints and the input/measured values are inconsistent (Reiter, 1987; Kleer & Williams, 1987). This notion leads to the definition of a Possible Conflict:

**Definition 1 (Possible Conflict)** *The set of constraints in a MEC that give rise to at least one MEM.*

Recent works have demonstrated the equivalence between MECs, Analytical Redundancy Relations (ARRs), and other structural model decomposition methods (Armengol et al., 2009).

### 3.2. Inclusion of temporal information in the models

There are two kinds of contraints in the model: *Differential* constraints, those used to model dynamic behaviour, and *instantaneous* constraints, those used to model static or instantaneous relations between system variables.

Differential constraints represent a relation between a state variable and its first derivative $(x, \frac{dx}{dt})$. These constraints can be used in the MEMs in two ways, depending on the selected causality assignment. In integral causality, constraint is solved as $x(t) = x(t-1) + \int_{t-1}^{t} \dot{x}(t)dt$. In derivative causality, $\dot{x}(t) = \frac{dx}{dt}$ assumes that the derivative can be computed based on present and past samples for $x$. Integral causality usually implies using simulation, and it is the preferred approach in the DX field. Derivative causality is the preferred approach in the FDI approach. Both have been demonstrated to be equivalent for numerical models, assuming adequate sampling rates and precise approximations for derivative computation are available, and assuming initial conditions for simulation are known (Chantler, Daus, Vikatos, & Coghill, 1996). PCs can easily handle both types of causality, since they only represent a different causal assignment while building MEMs (Pulido et al., 2010).

Special attention must be paid to loops in the MEM (set of equations that must be solved simultaneously). Loops containing differential constraints in integral causality are al-

[2]In this context, by causality assignment we mean every possible way one variable in one equation can be solved assuming the remaining variables are known.

lowed, because under integral causality the time indices are different to both sides of the differential constraint (Dressler, 1994, 1996). It is generally accepted that loops containing differential constraints in derivative causality can not be solved (Blanke et al., 2006).

Summarizing, each MEM for a PC represents how to build an executable model to monitor the behaviour of the subsystem defined by the PC. Such executable model can be implemented as a simulation model or as a state-observer (Pulido et al., 2010). However, building such model for complex non-linear systems it is not a trivial task. In Section 5 we will show the set of PCs obtained for our case study, and we will derive a simulation model for one of the PCs. In subsection 5.3 we will show how a grey-box model using neural networks can be obtained for the same PC. Next section shows the fundamentals for the type of neural network model used in this work.

## 4. State space neural networks for behaviour estimation

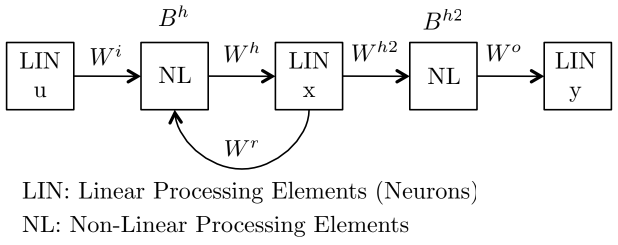
State Space Neural Networks (ssNN) (Zamarreño & Vega, 1998) is a great tool for modelling non-linear processes as shown in several cases (González Lanza & Zamarreño, 2002; Zamarreño, Vega, García, & Francisco, 2000); even in the sugar industry (Zamarreño & Vega, 1997). Main advantages of such modelling approach are its ability for representing any non-linear dynamics, and what is called a *parallel model*. This model represents the cause-effect process dynamics without considering past inputs and/or past outputs. The dynamic relation is modelled by the state layer, which calculates the internal state of the network using just current inputs of the model and internal state values from the previous time step.

The architecture of the ssNN (see figure 3) consists of five blocks, and each block represents a neural network layer. From left to right, the number of neurons at each layer is $n$, $h$, $s$, $h2$ and $m$. The third layer represents the state of the system (the dynamics). As can be seen in the figure, there is a feedback from the state layer to the previous layer, which means that the current state depends (in a non-linear way) on the state at the previous time step. The second and fourth layers model the non-linear behaviour: from the inputs to the states and from the states to the outputs, respectively. The first and the fifth layers provide linear transformations from inputs and to outputs, respectively. The ssNN is implemented by the following mathematical representation:

$$\begin{aligned} \hat{\vec{x}}(t+1) &= W^h \cdot f1(W^r \cdot \hat{\vec{x}}(t) + W^i \cdot \vec{u}(t) + B^h) \\ \hat{\vec{y}}(t) &= W^o \cdot f2(W^{h2} \cdot \hat{\vec{x}}(t) + B^{h2}) \end{aligned}$$

where the parameters are weight matrices, $W$, and bias vectors, $B$:

- $W^i$, $W^h$, $W^r$, $W^{h2}$, $W^o$ are matrices with dimension *h* x *n*, *s* x *h*, *h* x *s*, *h2* x *s* and *m* x *h2*, respectively.
- $B^h$ and $B^{h2}$ are bias vectors with *h* and *h2* elements respectively.
- $f_1$ and $f_2$ are two functions (non-linear, in general) which are applied elementwise to a vector or matrix. They are usually of sigmoid type.



$B^h$
$B^{h2}$
LIN u
$W^i$
NL
$W^h$
LIN x
$W^{h2}$
NL
$W^o$
LIN y
$W^r$
LIN: Linear Processing Elements (Neurons)
NL: Non-Linear Processing Elements

Figure 3. Generic state space Neural Network architecture

For some processes, where some *a priori* knowledge about the first principle equations can be obtained, a *black box* model could be too generic to obtain good results. But this knowledge can be used to restrict the *architecture* of the model, so we end up with a *grey box* model that can be better adjusted to mimic the process. Next section will illustrate the training process to obtain a specific grey box model for a PC related to an evaporation unit.

## 5. Results on the case study

### 5.1. PCs for the evaporation unit

As mentioned in Section 2.1 the evaporation section of the sugar factory is made up of five effects working sequentially to increase the sugar concentration in the syrup. All the evaporation units in the same effect share the same steam output conduit, and provide the steam for the next effect, thus partially coupling the behaviour of all the units. For our tests we have focused on the first evaporation unit in the first effect.

There are several assumptions that must be made in order to simplify the original model used in the training simulator, and to use those first principles equations for diagnosis. In our case, we simplified the dynamic processes actually happening inside the evaporation chamber, and we assumed the system was in only one operation mode. The dynamic processes considered in the evaporation unit were: conservation law for the amount of sugar and no-sugar products, global balance of matter in the evaporation chamber, sugar balance, level in the chamber, energy balances, steam volume balance, interchanged heat, and pressures in the chamber. As a result of this simplification process, our model was made up of 40 equations based on first principles of physics, 44 unknown variables , and 12 measured variables. Only 5 of these equations were used to model the evolution of 5 state variables: $C, T, M, juice\_out.T, Mvh$.

The algorithms used to compute the set of PCs provided 1058 MECs, and 775 MEMs. The total number of PCs in this system were 237, but most of them shared the same fault isolation capabilities since only 8 of the original 40 equations model relevant faulty behavior.

In the original model there are several equations containing partial derivatives, and several non-linear functions. As a consequence, most of the generated MEMs can be hardly implemented, although it is analytically possible. The problem we faced at that point was to implement the relevant MEMs, because it would be necessary to write by hand each simulation model. The process needs to be supervised by the modelling expert, thus producing a bottleneck in the development of the diagnosis system. As a consequence, to test the approach, we have modelled only one of the PCs, $PC_{195}$, whose MEM is graphically described in Figure 4. The MEM is a directed hypergraph which represents how the equations must be used to compute the output, $steam\_out.P$, using just measurements as inputs, and how the inputs are used to compute the intermediate unknown variables. Each solid arc represent an instantaneous constraint. Each dashed arc represent a differential constraint. In this system we use integral causality, hence each dashed arc means that we must perform integration to obtain the value of the state variable.

We selected the subsystem because it contains 16 equations, several input measurements –$juice\_in.W$, $juice\_in.Brix$, $juice\_out.T$, and $level\_juice.signal$–, and several state-variables –$M$, $C$, and $Mvh$–. The observed output variable is $steam\_out.P$. Hence, it has enough complexity to be a good test for the state space neural model.

## 5.2. The experimental data-set

$PC_{195}$ was implemented in EcosimPro (EcosimPro, 2012). We run a set of five experiments using real data from the factory for an intermediate month in the five month campaign. Experiment 1 consisted of 900 data points taken from 9 measurements in the system every 30 seconds. Experiments 2, 3, 4 and 5 consisted of 2800 data points taken from the same 9 measurements every 30 seconds[3]. Data sets 1, 2, and 4 represent nominal behaviour. Data set 3 represent a fault in the output sensor.

In order to monitor the nominal behavior and perform fault detection, we empirically determined a threshold. Figure 7 shows the performance of the model on the four scenarios. It can be seen that the simulation model is able to monitor the nominal behaviour and also to detect the fault, but as shown in experiments 2 and 3 the estimations obtained are not very accurate. This is due mainly to the assumptions made regarding unknown parameters and boundary conditions.

[3]Since the first experiment is shorter than the other four, we do not show its results.

## 5.3. State space neural network models for PCs

$PC_{195}$ graphically specifies in Figure 4 the relations between the inputs ($juice\_level.signal$, $juice\_in.W$, $juice\_out.T$, and $juice\_in.Brix$) and the states ($M$, $C$, $Mvh$ and $steam\_out.P$). Moreover, the last input ($juice\_in.Brix$) can be considered constant along time, so it can be removed from the model. The output of the model is $steam\_out.P$, so there is a direct relation between the output and one of the states. Taking this into account, the ssNN architecture can be customized to represent the process characteristics in a better way, as described in Figure 5. The non-linear (hidden) layer is split into four parts, and each part (represented by NL inside a square) has a number of neurons (h1, h2, h3, h4) that must be adjusted by trial and error to represent the nonlinear dynamics of each state.

This simplified ssNN architecture can be viewed as removing some of the weights between layers, or setting zeros in some specific elements of the weight matrices (the matrices can be seen in figure 6). Dimension of matrices $W^i$, $W^r$, $W^h$, and $W^o$ is $(h1 + h2 + h3 + h4) \times 3$, $(h1 + h2 + h3 + h4) \times 4$, $4 \times (h1 + h2 + h3 + h4)$, and $1 \times 4$, respectively.

### 5.3.1. Training

Training is the process of modifying the parameters (weights and bias) of the neural network to adjust its output to the process output. Error between the neural network output and process output has to be minimized, so the training procedure is an optimization task where some index, Sum Squared Error (SSE) in our case, has to be minimized.

A feedforward network is quite easy to train, using the backpropagation method or some of its variants. But a recurrent neural network (such as ssNN) is more difficult to train due to the recurrent connections. Stochastic methods are an alternative for this kind of neural network, which results in easier to implement training algorithms. The Modified Random Optimization Method (Solis & Wets, 1981) has been selected in this work, but with some modifications to improve convergence as shown in (González-Lanza & Zamarreño, 2002).

For training this ssNN architecture, we used the experiments explained in Section 5.2. Experiments 1, 3 and 5 were the training set, and experiments 2 and 4 were used for validation. The only parameter to tune in this *ad hoc* ssNN is the number of hidden neurons at the second layer. With 5 neurons at each part is enough to represent the data, thus a total of 20 sigmoid neurons.

Figure 8 shows the evolution of the estimated and measured variable for $PC_{195}$ for the selected 4 experiments. To use the ssNN model for fault detection, a new threshold was empirically calculated. Again, the ssNN model is able to track the nominal behaviour (experiments 1, 2, and 4 in Figure 8), and to correctly detect the fault in experiment 3 when the residual

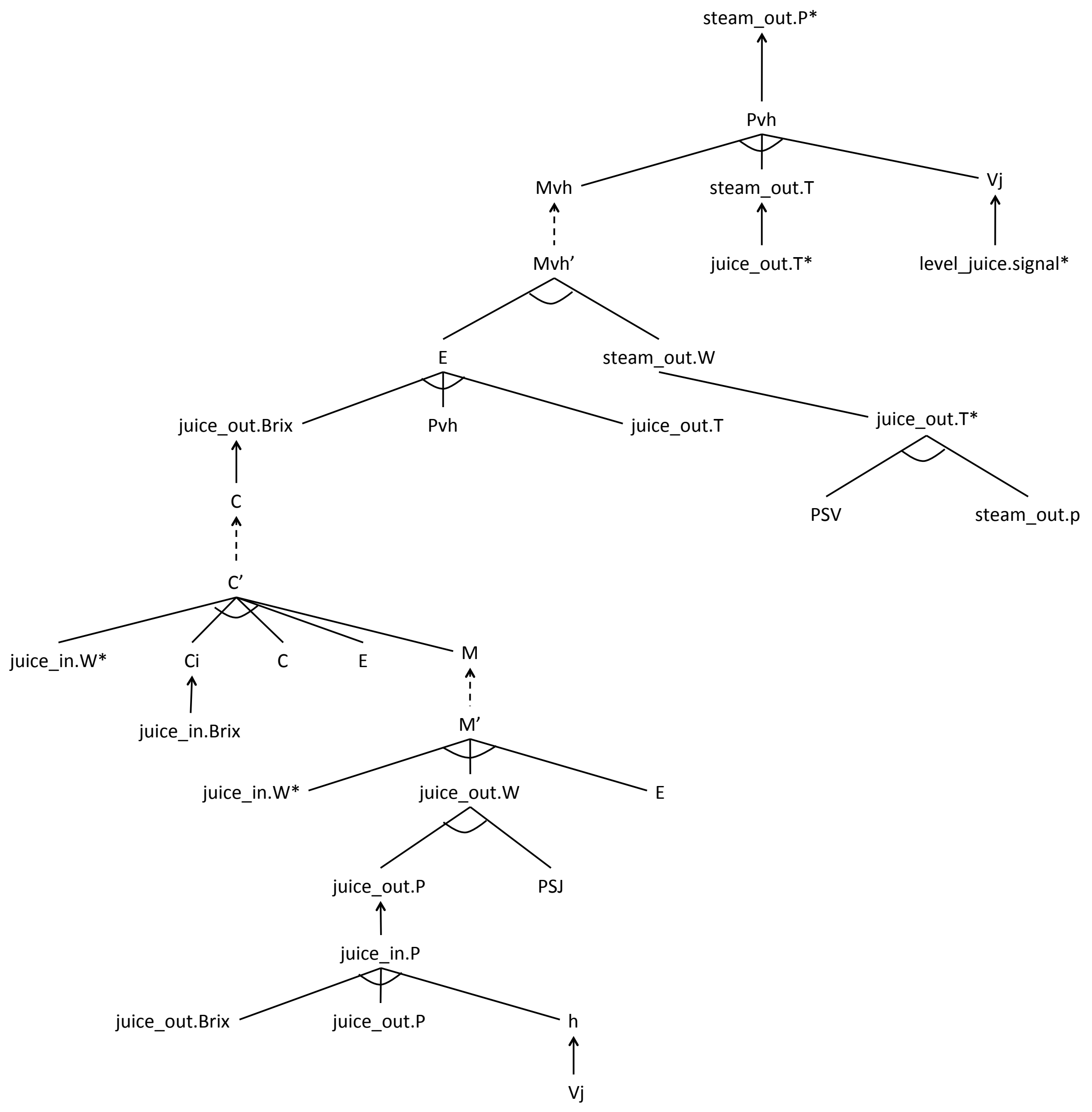


Figure 4. Minimal Evaluable Model schematics for Possible Conflict $PC_{195}$. The estimated variable is $Pvh$. The corresponding measured variable is $Steam\_out.P$.

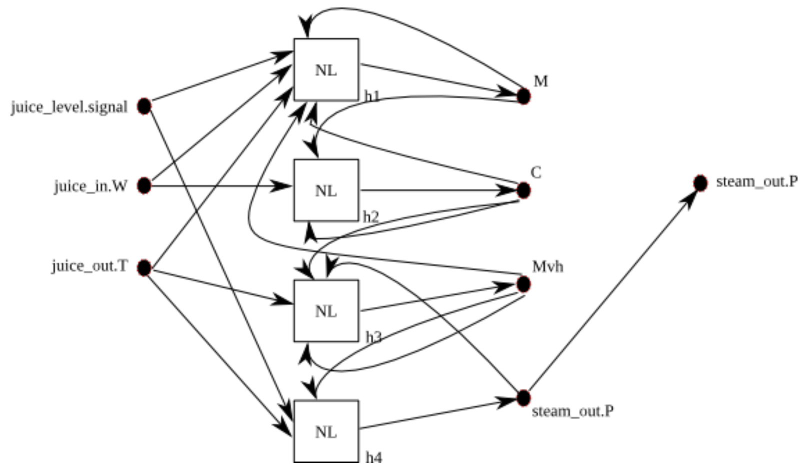


Figure 5. Simplified ssNN to better represent the model for the PC.

$$W^i = \begin{bmatrix} \begin{bmatrix} i_{1,1} & i_{1,2} & i_{1,3} \\ \vdots & \vdots & \vdots \\ i_{h1,1} & i_{h1,2} & i_{h1,3} \end{bmatrix} \\ \begin{bmatrix} 0 & i_{h1+1,2} & 0 \\ \vdots & \vdots & \vdots \\ 0 & i_{h2,2} & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & i_{h2+1,3} \\ \vdots & \vdots & \vdots \\ 0 & i_0 & i_{h3,3} \end{bmatrix} \\ \begin{bmatrix} i_{h3+1,1} & 0 & i_{h3+1,3} \\ \vdots & \vdots & \vdots \\ i_{h4,1} & 0 & i_{h4,3} \end{bmatrix} \end{bmatrix}$$

$$W^r = \begin{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & 0 \\ \vdots & \vdots & \vdots & \vdots \\ r_{h1,1} & r_{h1,2} & r_{h1,3} & 0 \end{bmatrix} \\ \begin{bmatrix} r_{h1+1,1} & r_{h1+1,2} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ r_{h2,1} & r_{h2,2} & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & r_{h2+1,2} & r_{h2+1,3} & r_{h2+1,4} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & r_{h3,2} & r_{h3,3} & r_{h3,4} \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & r_{h3+1,3} & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & r_{h4,3} & 0 \end{bmatrix} \end{bmatrix}$$

$$W^h = \begin{bmatrix} \begin{bmatrix} d_{1,1} & \cdots & d_{1,h1} \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \end{bmatrix} & \begin{bmatrix} 0 & \cdots & 0 \\ d_{2,h1+1} & \cdots & d_{2,h2} \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \end{bmatrix} & \begin{bmatrix} 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ d_{3,h2+1} & \cdots & d_{3,h3} \\ 0 & \cdots & 0 \end{bmatrix} & \begin{bmatrix} 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ d_{4,h3+1} & \cdots & d_{4,h4} \end{bmatrix} \end{bmatrix}$$

$$W^o = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 6. Simplified weight matrices $W^i$, $W^r$, $W^h$, and $W^o$ for the ssNN implementing $PC_{195}$.

exceeds the threshold.

Looking at results in Figures 7 and 8 we can see that both models can be used to monitor the evolution of variable $Steam\_out.P$. Main difference comes from the bias introduced by the parameters in the first principles models in Fig. 7 leading to a higher threshold for fault detection. Nevertheless, the model can still be used for monitoring and fault detection.

The ssNN model used only 3 experiments for training, and was able to track the nominal behaviour more accurately, and it was capable to detect the fault in the sensor. However, more training is necessary considering data from different months.

## 6. CONCLUSIONS

In this work we have proposed to use Possible Conflicts to decompose a large system model into smaller models with minimal redundancy for fault detection and isolation. Possible Conflicts provide the structural models (equations, inputs, outputs, and state variables) required for model-based fault detection and isolation, and these models can be implemented as simulation or state-observer. Since deriving such models for complex non-linear models it is not straightforward and requires the participation of modelling experts, we have proposed to use the structural information in the model to design a neural network grey box model using a state space architecture.

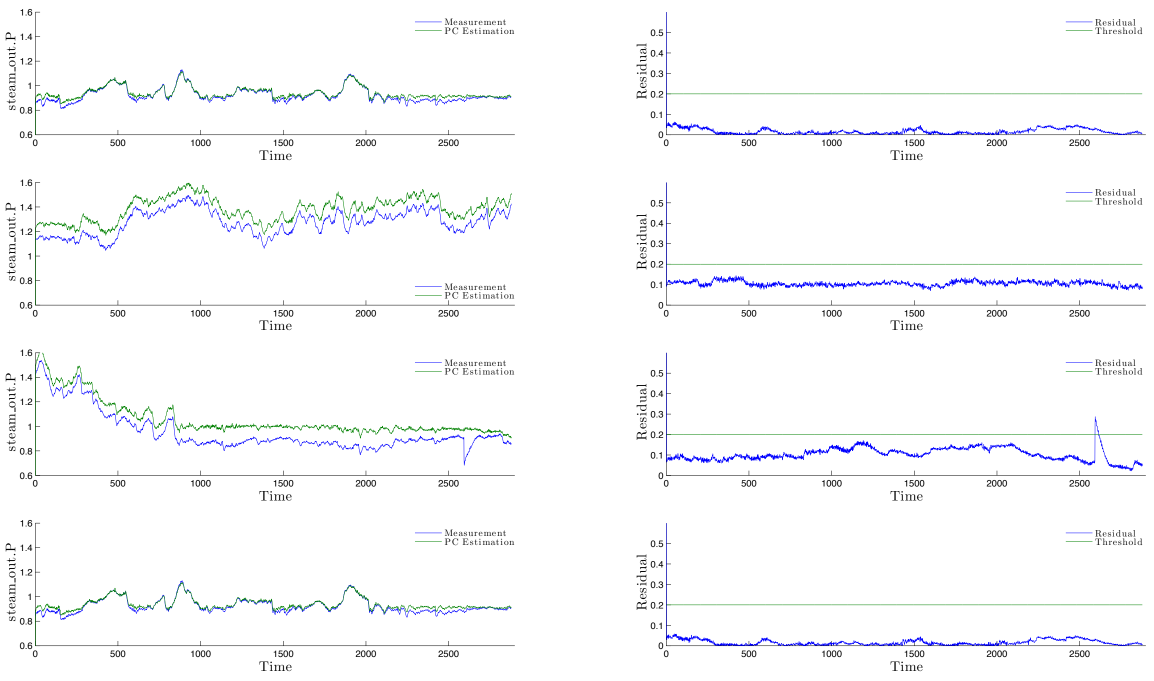


Figure 7. Results for the PC tracking the system using the first principles model. The figure represent 4 experiments with real data. On the left we represent the estimated and the real value of the magnitude. On the right we represent the evolution of the residual, and the threshold.

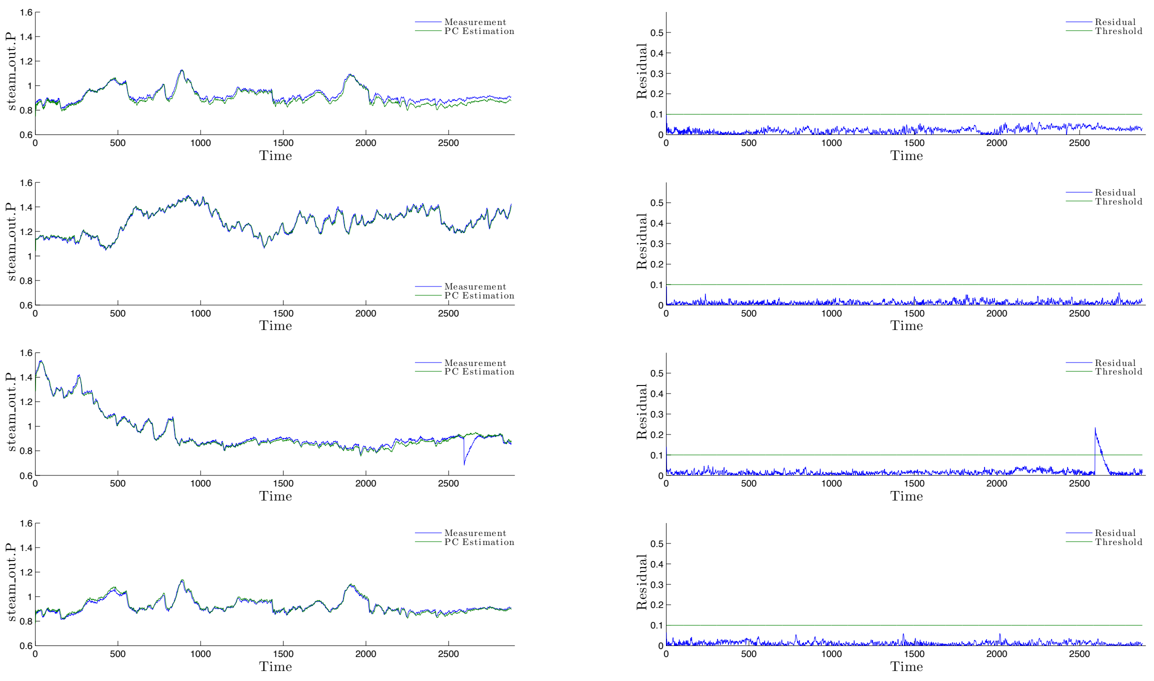


Figure 8. Results for the PC tracking the system using the ssNN model. The figure represent 4 experiments with real data. On the left we represent the estimated and the real value of the magnitude. On the right we represent the evolution of the residual, and the threshold.

The main conclusion is that the structure of the Minimal Evaluable Model for a Possible Conflict can guide the design of the state space model of the neural network, reducing its complexity and avoiding the process of multiple unknown parameter estimation in the first principles models. Comparing results of this approach in an evaporation unit of a beet sugar factory we have observed that the ssNN is able to obtain similar of even better results than a simulation model manually derived by an expert. Both types of models were used to successfully monitor the process and to detect faults.

As further work, we plan to derive additional ssNNs and to test on a larger experiment data-set. Additionally, we need to test the approach at different times of the season, because this is a very slow evolving process whose parameters vary over time. Moreover, we can test more abstract models that will produce fewer PCs, but still containing same structural information. Finally, once we introduce larger data sets, we will use statistical tests to perform fault detection, and to determine the threshold to guarantee a maximum percentage of false positives and false negatives.

## Acknowledgment

This work has been supported by Spanish MCI grants TIN2009-11326 and DPI2009-14410-C02-02. We would also want to thank personnel from “Azucarera Española” for the data provided for these experiments, and three anonymous reviewers for their comments that have help us to improve this paper.

## References

Acebes, L., Merino, A., Alves, R., & Prada, C. de. (2009). Online energy diagnosis of sugar plants (in Spanish in the original). *RIAI - Revista Iberoamericana de Automatica e Informatica Industrial*, *6*(3), 68-75.

Alves, R., Normey-Rico, J., A., M., Acebes, L., & Prada, C. de. (2008, June). Distributed Continuous Process Simulation: An Industrial Case Study. *Computers and Chemical Engineering*, *32*(6), 1203-1213.

Armengol, J., Bregon, A., Escobet, T., Gelso, E., Krysander, M., Nyberg, M., et al. (2009). Minimal Structurally Overdetermined sets for residual generation: A comparison of alternative approaches. In *Proceedings of the 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS09* (p. 1480-1485). Barcelona, Spain.

Blanke, M., Kinnaert, M., Lunze, J., & Staroswiecki, M. (2006). *Diagnosis and Fault-Tolerant Control*. Springer.

Chantler, M., Daus, S., Vikatos, T., & Coghill, G. (1996). The use of quantitative dynamic models and dependency recording engines. In *Proceedings of the Seventh International Workshop on Principles of Diagnosis, DX96* (p. 59-68). Val Morin, Quebec, Canada.

Cordier, M., Dague, P., Lévy, F., Montmain, J., & Travé-Massuyès, M. S. L. (2004). Conflicts versus Analytical Redundancy Relations: a comparativeanalysis of the Model-based Diagnosis approach from the Artificial Intelligence and Automatic Control perspectives. *IEEE Trans. on Systems, Man, and Cybernetics. Part B: Cybernetics*, *34*(5), 2163-2177.

Daigle, M., Bregon, A., & Roychoudhury, I. (2011, September). Distributed Damage Estimation for Prognostics Based on Structural Model Decomposition. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2011* (p. 198-208).

Dressler, O. (1994). Model-based Diagnosis on Board: Magellan-MT Inside. In *Working Notes of the International Workshop on Principles of Diagnosis, DX94.* Goslar, Germany.

Dressler, O. (1996). On-line diagnosis and monitoring of dynamic systems based on qualitativemodels and dependency-recording diagnosis engines. In *Proceedings of the Twelfth European Conference on Artificial Intelligence, ECAI-96* (p. 461-465). John Wiley and Sons, New York.

Dressler, O., & Struss, P. (1996). The Consistency-based approach to automated diagnosis of devices. In G. Brewka (Ed.), *Principles of Knowledge Representation* (p. 269-314). CSLI Publications, Standford.

Empresarios Agrupados Internacional. (2012). *EcosimPro.* http://www.ecosimpro.com/. Madrid, Spain.

Gertler, J. (1998). *Fault detection and diagnosis in Engineering Systems*. Marcel Dekker, Inc., Basel.

González-Lanza, P., & Zamarreño, J. (2002, january). A hybrid method for training a feedback neural network. In *First International ICSC-NAISO Congress on Neuro Fuzzy Technologies NF 2002.* Havana - Cuba.

González Lanza, P., & Zamarreño, J. (2002). A short-term temperature forecaster based on a state space neural network. *Engineering Applications of Artificial Intelligence*, *15*(5), 459 - 464.

Hamscher, W., Console, L., & Kleer (Eds.), J. de. (1992). *Readings in Model based Diagnosis*. Morgan-Kaufmann Pub., San Mateo.

Kleer, J. de, & Williams, B. (1987). Diagnosing multiple faults. *Artificial Intelligente*, *32*, 97-130.

Luyben, W. (1990). *Process modeling, simulation, and control for chemical engineers*. McGraw-Hill.

Merino, A. (2008). *Librería de modelos del cuarto de remolacha de una industria azucarera para un simulador de entrenamiento de operarios*. Unpublished doctoral dissertation, Universidad de Valladolid.

Merino, A., Alves, R., & Acebes, L. (2005). A training simulator for the evaporation section of a beet sugar production process. In *Proceedings of the 2005 European Simulation and Modelling conference.*

Patton, R. J., Frank, P. M., & Clark, R. N. (2000). *Issues in fault diagnosis for dynamic systems.* Springer Verlag, New York.

Pulido, B., & Alonso-González, C. (2004). Possible conflicts: a compilation technique for consistency-based diagnosis. *"IEEE Trans. on Systems, Man, and Cybernetics. Part B: Cybernetics"*, *34*(5), 2192-2206.

Pulido, B., Alonso-González, C., & Acebes, F. (2001). Consistency-based diagnosis of dynamic systems using quantitative models and off-line dependency-recording. In *12th International Workshop on Principles of Diagnosis (DX-01)* (p. 175-182). Sansicario, Italy.

Pulido, B., Bregon, A., & Alonso-González, C. (2010). Analyzing the influence of differential constraints in Possible Conflict and ARR computation. In *Current Topics in Artficial Intelligence, CAEPIA 2009 Selected Papers. P. Meseguer, L. Mandow, R. M. Gasca Eds.* Springer-Verlag Berlin.

Reiter, R. (1987). A Theory of Diagnosis from First Principles. *Artificial Intelligence*, *32*, 57-95.

Solis, F., & Wets, R. J.-B. (1981). Minimization by Random Search Techniques. *Mathematics of Operations Research*, *6*, 19–30.

Zamarreño, J., & Vega, P. (1997). Identification and predictive control of a melter unit used in the sugar industry. *Artificial Intelligence in Engineering*, *11*(4), 365 - 373.

Zamarreño, J., & Vega, P. (1998). State space neural network. Properties and application. *Neural Networks*, *11*(6), 1099–1112.

Zamarreño, J., Vega, P., García, L., & Francisco, M. (2000). State-space neural network for modelling, prediction and control. *Control Engineering Practice*, *8*(9), 1063 - 1075.

## BIOGRAPHIES

**Belarmino Pulido** received his Licenciate degree, M.Sc. degree, and Ph.D. degree in Computer Science from the University of Valladolid, Valladolid, Spain, in 1992, 1995, and 2001 respectively. In 1994 he joined the Dept. of Computer Science at the University of Valladolid, where he is Associate Professor since 2002. His main research interests are Model-based and Knowledge-based reasoning, for Supervision and Diagnosis. He has worked in several national and European funded projects related with Supervision and Diagnosis, and is the coordinator of the Spanish Network on Supervision and Diagnosis of Complex Systems since 2005.

**Jesús Maria Zamarreño** has a degree in Physics, and PhD in Physics from the University of Valladolid, Spain, where he is Lecturer (Associate Professor). He is with the Dept. of System Engineering and Automatic Control. He is a member of the IFAC Spanish section CEA, and also is Advisor of the ISA student section at Valladolid. His research interests are Artificial Neural Networks, Agent-based modelling, Model-based predictive control, and OPC Applications.

**Alejandro Merino** received has a degree in Chemical Engineering and M.Sc. and PhD degree in Process and Systems Engineering from the University of Valladolid, Spain. He is currently Assistant Professor at University of Burgos, Spain, and also senior researcher at the Centre of Sugar Technology. He has worked in different projects related to modeling and optimization of complex industrial processes. His research interest is modelling and optimization of dynamic processes.

**Anibal Bregon** received his B.Sc., M.Sc., and Ph.D. degrees in Computer Science from the University of Valladolid, Valladolid, Spain, in 2005, 2007, and 2010, respectively. Currently he is Assistant Professor and Research Assistant at the Department of Computer Science from the University of Valladolid. From September 2005 to June 2010, he was Graduate Research Assistant with the Intelligent Systems Group at the University of Valladolid, Spain. During that time he was visiting scholar at the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA; the Dept. of Electrical Engineering, Linkoeping University, Linkoeping, Sweden; and the Diagnostics and Prognostics Group, NASA Ames Research Center, Mountain View, CA, USA. His current research interests include model-based reasoning for diagnosis, prognostics, health-management, and distributed diagnosis of complex physical systems.