To link to this article: **DOI:10.1016/j.compchemeng.2009.04.016**

http://dx.doi.org/10.1016/j.compchemeng.2009.04.016

doi:10.1016/j.compchemeng.2009.04.016

Model-based fault diagnosis for hybrid systems: Application on chemical processes

Nelly Olivier-Maget[a,*], Gilles Hétreux[a], Jean Marc Le Lann[a], Marie Véronique Le Lann[b,c]

[a] Laboratoire de Génie Chimique (PSI – Génie Industriel), UMR-CNRS 5503, INPT-ENSIACET, 118, Route de Narbonne, Toulouse F-31077, France
[b] LAAS-CNRS, Université de Toulouse, 7, avenue du Colonel Roche, Toulouse F-31077, France
[c] Université de Toulouse, INPT, INSA, France

Keywords:
Fault detection and isolation
Extended Kalman filter
Signature
Distance
Dynamic hybrid simulation
Object differential Petri nets

A B S T R A C T

The complexity and the size of the industrial chemical processes induce the monitoring of a growing number of process variables. Their knowledge is generally based on the measurements of system variables and on the physico-chemical models of the process. Nevertheless, this information is imprecise because of process and measurement noise. So, the research ways aim at developing new and more powerful techniques for the detection of process fault. This article presents a method for the fault detection based on the comparison between the reference model evolution and the real system generated by the extended Kalman filter. The reference model is simulated by the dynamic hybrid simulator, *PrODHyS*. It is a general object-oriented environment which provides common and reusable components designed for the development and the management of dynamic simulation of industrial systems. The use of this method is illustrated through a didactic example relating to the field of Chemical Process System Engineering.

1. Introduction

With the evolution of the computer power, dynamic simulation has become an efficient study tool in chemical process design and analysis. Indeed, it is exploited with success, for instance, for the studies of the system behaviour faced with disturbances around a set point (sensitivity to the parameters), or for the initialization of a steady-state simulation (i.e. distillation operation). However, numerous operation states – such as batch production mode, material physical state changes or also abrupt evolutions – make the use of purely discrete or purely continuous models difficult. Indeed, the modelling of these discontinuous phenomena leads to the implementation of piecewise continuous models, which require the use of hybrid dynamic simulator for their resolution.

Thanks to their large application field, numerous works on hybrid dynamic systems (*HDS*) deal with modelling, stability and control (Birouche, 2006; Zaytoon, 2001). Among them, some research works focus on the monitoring of these systems and many methods have been developed for fault detection and isolation (*FDI*) (Zhao, Koutsoukos, & Haussecker, 2005). For instance, important research ways of diagnosis approaches for hybrid systems are based on:

- discrete and/or temporal abstractions of the continuous dynamics (Lunze, 2000; Mosterman & Biswas, 1999);
- or particle filtering methods (Koller & Lerner, 2001; McIlraith, Biswas, Clancy, & Gupta, 2000).

Commonly, fault diagnosis tools are commonly divided into model-based methods or without model methods. In our case, a model-based approach has been developed. This exploits an extended Kalman filter adapted to hybrid dynamic systems. The main idea is to reconstruct the outputs of the system from the measurement using observers or Kalman filters and using the residuals for fault detection (Mehra & Peschon, 1971; Simani & Fantuzzi, 2006; Welch & Bishop, 1995). The purpose is to detect the presence of a fault and to locate the occurrence time. The estimations are compared to the normal variable values and so, deviations are interpreted as faults. Next, the problem is similar to a pattern recognition problem.

This article is organized as follows. The first section underlines the significant works in fault detection and isolation and in hybrid dynamic systems. The second part presents the proposed model-based methodology. Next, its implementation is underlined and a general overview of the simulation environment and its software structure are presented. Then, the main fundamental concepts of the *ODPN* formalism are described. This is followed by a presentation of a modelling within *PrODHyS*. These concepts are illustrated through the simulation of a typical process example. Composed of

* Corresponding author. Tel.: +33 05 62 88 58 57; fax: +33 05 62 88 56 00.
E-mail address: Nelly.Olivier@ensiacet.fr (N. Olivier-Maget).

a heat tank, an energy feed and a material feed, the goal of this system is the change of solvents. A fault in the functioning of the energy system is introduced at an unknown moment. The energy system does not provide enough heat quantity. Finally, Section 8 summarises the contributions and achievements of the paper and some future research works are suggested.

2. Significant works

2.1. Fault detection and diagnosis methods

In a very competitive economic context, the reliability of the production systems can be a decisive advantage. In this context, a simple failure is considered as prejudicial. This is why, the fault detection and diagnosis are the purpose of a particular attention in the scientific and industrial community. The major idea is that the defect must not be subjected but must be controlled. Nowadays, the fault detection and diagnosis remain a large research field. The literature quotes a great number of fault detection and diagnosis methods in various domains of application (Venkatasubramanian, Rengaswamy, Yin, & Kavuri, 2003). A notable number of works has been devoted to fault detection and isolation, and the techniques are generally classified as:

- *methods without models* such as quantitative process history based methods—neural networks (Venkatasubramanian et al., 2003), statistical classifiers (Anderson, 1984), or qualitative process history based methods—expert systems (Venkatasubramanian et al., 2003);
- and *model-based methods* which are composed of quantitative model-based methods—such as analytical redundancy (Chow & Willsky, 1984), parity space (Gertler & Singer, 1990), state estimation (Willsky, 1976), or fault detection filter (Frank, 1990), and qualitative model-based methods—such as causal methods: digraphs (Shih & Lee, 1995), or fault trees (Venkatasubramanian et al., 2003).

The above mentioned articles underline the necessity to have precise knowledge of the normal process states during *fault detection and isolation*. It has been recognized that fault detection using all measurements and models by a single filter is weighty and sometimes inappropriate for real-time use (Lee & White, 1998; Zhao et al., 2005). Nevertheless, while model-based techniques require greater computing power and data storage, the amazing progress in computer technology has made them feasible at low costs. Moreover, model deviation, nonlinearities and measurement disturbances are unavoidable in industrial systems. They are often the cause of false detection in fault detection and isolation algorithms. Despite the complexity of such algorithms, most of the time, their robustness is insufficient in industrial context.

2.2. Simulation of hybrid dynamic systems

The simulation of unit operations and physico-chemical evolution of products often necessitates the implementation of phenomenological models. So, the traditional tools such as the discrete events simulators (which require fixed operational durations) or traditional formalisms integrating only the concept of temporal events – timed automata (Alur & Dill, 1994), timed Petri nets (Sifakis, 1977), or crossing speeds–continuous Petri nets (David & Alla, 1990), hybrid Petri nets (David & Alla, 2001) – are not well adapted to these problems. In this context, the use of hybrid dynamic simulators seems to be a better solution (Zaytoon, 2001).

In order to model them, two dynamic schemes have to be described: on the one hand, the continuous dynamic, which is generally represented by a Differential and Algebraic Equations (DAE) set, and on the other hand, the discrete one, which is represented by a set of discrete states and transition sets. Several formalisms have been defined to combine the continuous and discrete elements. In the literature, these formalisms are generally classified as:

- approaches which extend models of the continuous field, such as unified models (Branicky, 1995), bond-graphs with switches (Buisson & Cormerais, 1998);
- approaches which extend models of the discrete field, such as hybrid Petri nets (Le Bail, Alla, & David, 1991), batch Petri nets (Demongodin, 2001), time Petri nets (Berthomieu & Menasche, 1983), and timed automata (Alur & Dill, 1994);
- and finally mixed approaches, in which discrete and continuous models are exploited in the same structure (the hybrid aspects are taken into account in the interface between the two parts): hybrid automata (Alur et al., 1995), hybrid statecharts (Kesten & Pnueli, 1992), mixed Petri nets (Valentin-Roubinet, 1998), and differential predicate-transition Petri nets (Champagnat, Esteban, Pingaud, & Valette, 1998).

At the same time, several softwares have been developed for the simulation of hybrid systems, such as *gPROMS* (Barton & Pantelides, 1994), *Omsim* (Andersson, 1994), *BaSIP* (Wöllhaf, Fritz, Schulz, & Engell, 1996), *Shift* (Deshpande, Gollu, & Semenzato, 1998), and *Chi* (Fábián, van Beek, & Rooda, 1998). In these softwares, the hybrid aspect is described via an imperative language.

3. Description of the proposed *FDI* methodology

Nowadays, for reasons of safety and performance, monitoring and supervision have an important role in process control. The complexity and the size of industrial systems induce an increasing number of process variables and make difficult the work of operators. In this context, a computer aided decision-making tool seems to be wise.

The approach proposed in this article to handle the problem of fault detection and isolation is a model-based approach. The methodology, called *SimAEM* (*Simulation Abnormal Event Management*) is more particularly designed to treat batch and semi-continuous processes which are the prevalent mode of production for low volume of high added value products. Such systems are composed of interconnected and shared resources, in which a continuous treatment is carried out. For this reason, they are generally considered as hybrid systems in which the discrete and continuous aspects are handled. Moreover, the recipe is more often described with state events (temperature or composition threshold, etc.) than with fixed processing times.

In this context, based on the research works performed for several years on process modelling and simulation, we have developed the object-oriented environment *PrODHyS* dedicated to the hybrid dynamic simulation of chemical processes (Jourda, Joulia, & Koehret, 1996; Moyse, 2000; Olivier-Maget, Hétreux, LeLann, & LeLann, 2008; Perret, Hétreux, & Le Lann, 2004; Sargousse, 1999). The adopted hybrid formalism is based on a mixed approach and object concepts: the *Object Differential Petri nets* (*ODPN*). In these works, this platform has been exploited for monitoring studies: a simulation model is used as a reference model to implement the functions of detection and diagnosis. So, our *FDI* system is based on various simulations of the process achieved with *PrODHyS*. The on-line system identification and fault detection are based on the *Extended Kalman Filter*. Then, fault indicators with continuous values are generated. Finally, these instantaneous fault indicators are compared with the theoretical fault indicators by means of a distance.

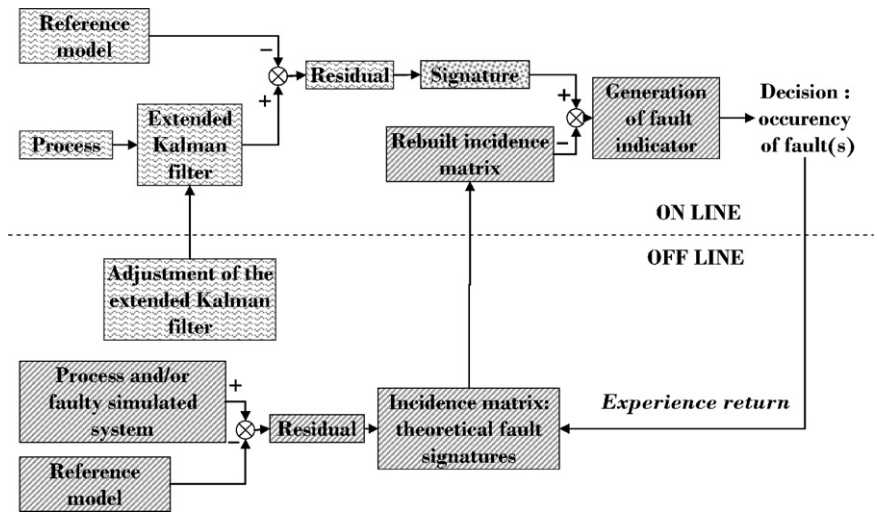

Fig. 1. Supervision architecture.

3.1. Functional architecture of the monitoring tool

The supervision module must be able to detect the faults of the physical systems (leak, energy loss, etc.) and the faults of the control/command devices (actuators, sensors, etc.). As defined in De Kleer and Williams (1987), our approach is based on the hypothesis that the reference model is correct. The global principle of this system is shown in Fig. 1, where the sequence of the operations is underlined. Moreover, a distinction between the on-line and off-line operations is made. Our approach is composed of three parts:

- The first part concerns the *residual generation* (waved pattern in Fig. 1). It consists in the comparison between the predicted behaviour obtained by simulation of the reference model and the real observed behaviour.
- The second part is the *signature generation* (dotted pattern in Fig. 1). This detection stage determines the presence or not of a fault by using a simple threshold.
- The last part is the diagnosis of the fault (hatched pattern in Fig. 1). The signature obtained in the previous part is compared with the theoretical fault signatures by means of distance. Then, a fault indicator is generated.

3.2. Generation of the residuals

The first part concerns the generation of the residuals (waved pattern in Fig. 1). In order to obtain an observer of the physical system, a real-time simulation is done in parallel. So, a complete state of the system will be available at any time. Thus, it is based on the comparison between the predicted behaviour obtained owing to the simulation of the reference model (values of state variables) and the real observed behaviour (measurements from the process correlated by *means of* the extended Kalman filter). The main idea is to reconstruct the system outputs from the measurement and to use the residuals for fault detection (Mehra & Peschon, 1971; Simani & Fantuzzi, 2006; Welch & Bishop, 1995). A description of the extended Kalman filter can be found in Olivier-Maget et al. (2008). Note that the extended Kalman filter takes into account the parametric and measurement disturbances. Moreover, the residual is defined according to the following equation:

$$r_i^r(t) = \hat{X}_i(t) - X_i(t) \quad \text{with} \quad i \in \{1, n\} \tag{1}$$

where X_i is the state variable, obtained by the reference model, $\hat{X}_i$ is the state variable estimated with the extended Kalman filter and n is the number of state variables.

Next, the last generated residual is determined by Eq. (2):

$$r_i^r(t) = \frac{\hat{X}_i(t) - X_i(t)}{X_i(t)} \quad \text{with} \quad i \in \{1, n\} \tag{2}$$

It has to be added that the generated residual $r_i^r(t)$ is relative difference and not binary. As a matter of fact, this allows the comparison between the residual of a variable and the residual of another one, since the residual is dimensionless.

Fig. 2 illustrates the residual concept. In Fig. 2(a), the evolutions of the reference and estimated states are represented. At the date t_0, we note that the estimated state deviates from the reference state. The graph (b) represents the relative residual. A normal zone was defined by the study of the model disturbances and it will be

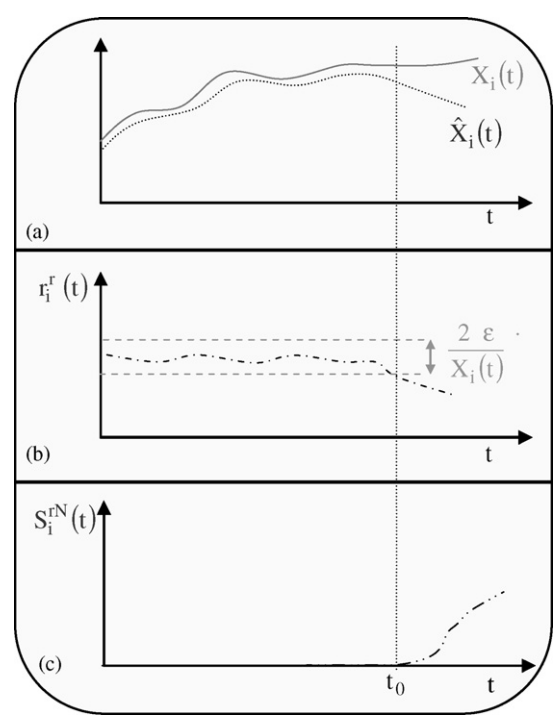

Fig. 2. Principles of the residual and signature structures.

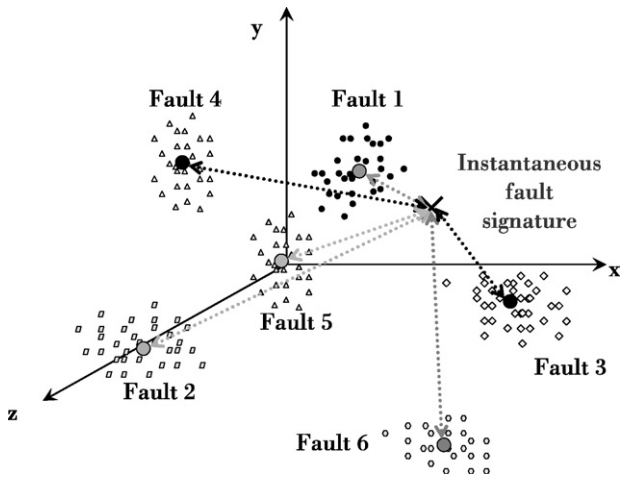

Fig. 3. "Distance" notion.

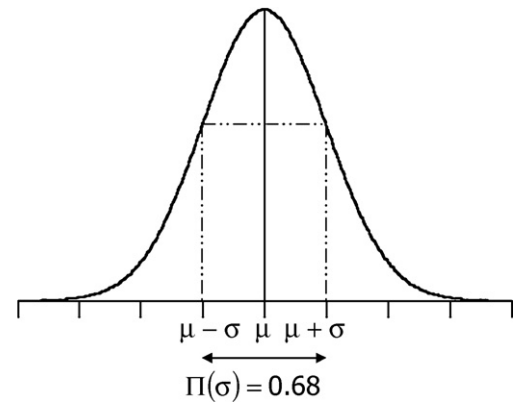

Fig. 4. Normal distribution.

explained below. We notice that the relative residual deviates from the domain of the nominal performance at the date t_0.

3.3. Generation of the signatures

The second part is the generation of the signatures (dotted pattern in Fig. 1). This is the detection stage. It determines the presence or not of a fault. This is made by a simple threshold, $\varepsilon_i(t)$. The generated structure $S_i^{rN}(t)$ is denoted by the following equation:

$$S_i^{rN}(t) = \frac{Max[(|r_i^r(t)| - \varepsilon_i'(t)); 0]}{\sum_{k=1}^{n} Max[(|r_k^r(t)| - \varepsilon_k'(t)); 0]} \quad \text{with} \quad i \in \{1, n\} \tag{3}$$

with $\varepsilon_i'(t) = \varepsilon_i(t)/X_i(t)$, where ε_i is the detection threshold and n the number of state variables.

The value of ε_i is evaluated according to the model error covariance matrix of the extended Kalman filter.

For example, the last graph (c) of Fig. 2 illustrates the signature notion. This synthetic structure reveals the symptoms of the abnormal behaviour. Then, while the relative residual is in the area of the nominal performance, the generated signature is zero. On the contrary, from the date t_0, this structure is nonzero. This underlines the presence of a symptom of the abnormal behaviour. Besides, it allows quantifying this symptom.

3.4. Generation of the fault indicators

The last part deals with the diagnosis of the fault (hatched pattern in Fig. 1). This is similar to a pattern recognition problem. The aim is to use distance metrics to calculate the distance of a given pattern from the means of various classes and classify the pattern to the class from which it is closest. This pattern is characterized by the signature obtained in the previous part (the instantaneous fault signature) and the various classes are represented by the theoretical fault signatures. Thus, the signature obtained in the previous part – the instantaneous fault signature – is compared with the theoretical fault signatures by means of distance. This principle is shown in Fig. 3.

Otherwise, a theoretical signature $T_{\bullet j}$ of the fault j is obtained by experience or, in our case, by simulations of the faulty process with numerous occurency dates of this fault. Note that all the theoretical fault signatures are listed in the same structure: the incidence matrix. Then, a fault indicator is generated. For this, two distances have been defined: the relative Manhattan distance and the improved Manhattan distance. The first distance is denoted by Eq. (4). The relative Manhattan distance is calculated component by component for each variable. It characterizes the similarity between the instantaneous fault signature and the theoretical fault signatures:

$$D_j^{\mathrm{Mr}}(t) = \frac{\sum_{i=1}^{n} S_i^{rN}(t) - T_{ij}}{n} \tag{4}$$

The second distance is the improved Manhattan distance. In this case, the instantaneous fault signature is not compared component by component with the theoretical fault signatures. The idea is to only consider the symptoms of the faulty behaviour – the nonzero components – and to characterize the similarity of symptoms. In this way, we only test that the nonzero components of the theoretical fault signatures match up to the nonzero components of the instantaneous fault signature, in order not to increase the distance between these signatures in the case of simultaneous faults. So, the improved Manhattan distance allows the diagnosis of many simultaneous faults and is denoted by Eq. (5):

$$D_j^{\mathrm{Ma}}(t) \frac{\sum_{i=1}^{n} |S_i^{rN}(t) \times m' - T_{ij} \times n'| \cdot T_{ij}}{n'} \tag{5}$$

where n' is the number of nonzero elements of the theoretical fault signature $T_{\bullet j}$ and m' is the number of nonzero elements of the fault signature $S_i^{rN}(t)$.

Since both distances are defined in the interval [0;1], the fault indicators are defined as the complement to 1 of these distances. These indicators represent the probability of the occurency of a particular fault. Next, these generated indicators are exploited to take a diagnosis of the system. For this, we suppose that:

- The minimum value of the indicator, for which the fault can be considered, is 0.68—this value has been chosen according to the normal distribution properties. This threshold corresponds to the probability $\Pi(-1 \leq f \leq 1)$, where f is the centred reduced normal variable (Fig. 4).
- The number of faults, which can simultaneously take place, is limited to three.

4. Implementation of the diagnosis module

The implementation of the diagnosis module is made within the environment *PrODHyS*. This environment provides a library of classes dedicated to the dynamic hybrid simulation of processes. Based on object concepts, *PrODHyS* offers extensible and reusable software components allowing a rigorous and systematic modelling of processes. The primal contribution of these works consisted in determining and designing the foundation building classes. An important evolution of *PrODHyS* is the integration of the dynamic hybrid simulation kernel (Hétreux, Théry, Olivier, & Le Lann, 2007;

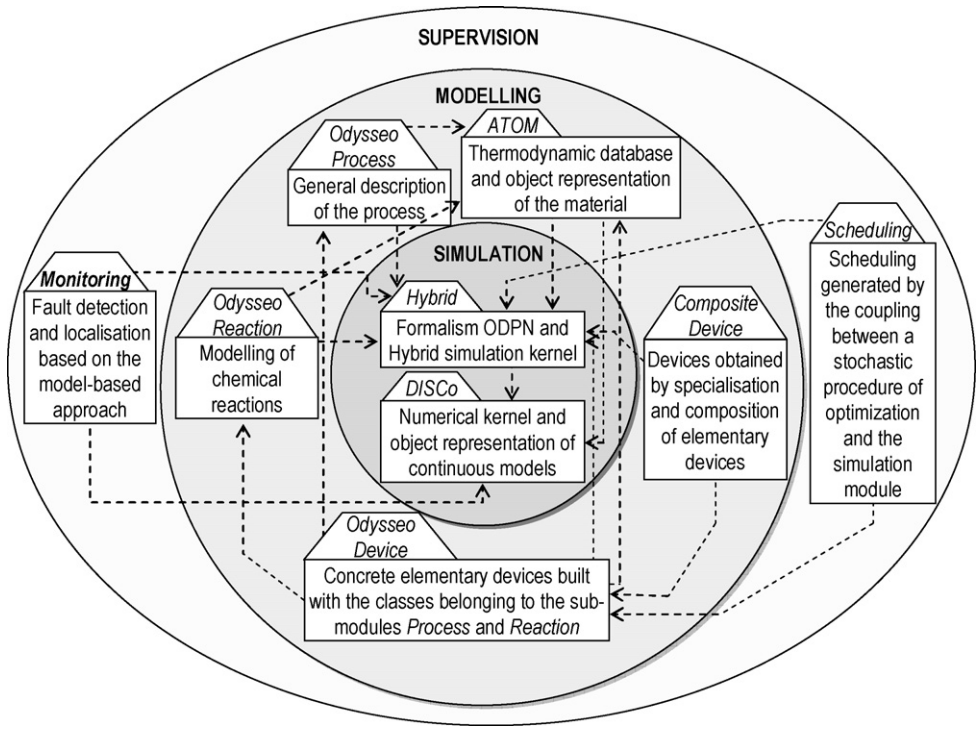

Fig. 5. Software architecture of *PrODHyS*.

Olivier-Maget et al., 2008; Perret et al., 2004). The object differential Petri nets (*ODPN*) formalism is used to describe the simulation model associated with each component. It combines in the same structure a set of *DAE* systems and high level Petri nets and has the ability to detect state and time events. The mathematical properties of this formalism are not developed here but can be found in Perret et al. (2004).

4.1. PrODHyS software architecture

Currently, this library is made up of more than one thousand classes distributed into three independent functional layers (simulation/modelling/supervision) and nine modules (Fig. 5):

- The internal layer corresponds to the simulation kernel of the platform. It provides the basic elements allowing the simulation of any dynamic systems. This layer includes the module *Disco* (Le Lann, 1999; Sargousse, 1999) and the module *Hybrid* (Perret et al., 2004).
- The middle layer includes a set of classes allowing the modelling of processes. The "modelling" layer rests on the "simulation" layer and provides a set of general and autonomous entities which can be exploited by any user who wishes to build its own simulation system or prototype. This layer includes the module *ATOM* (Jourda et al., 1996), the module *Odysseo* (Moyse, 2000) and the module *CompositeDevice* (Moyse, 2000).
- The higher layer has been added to the PrODHyS architecture. It corresponds to a set of classes dedicated to process supervi-

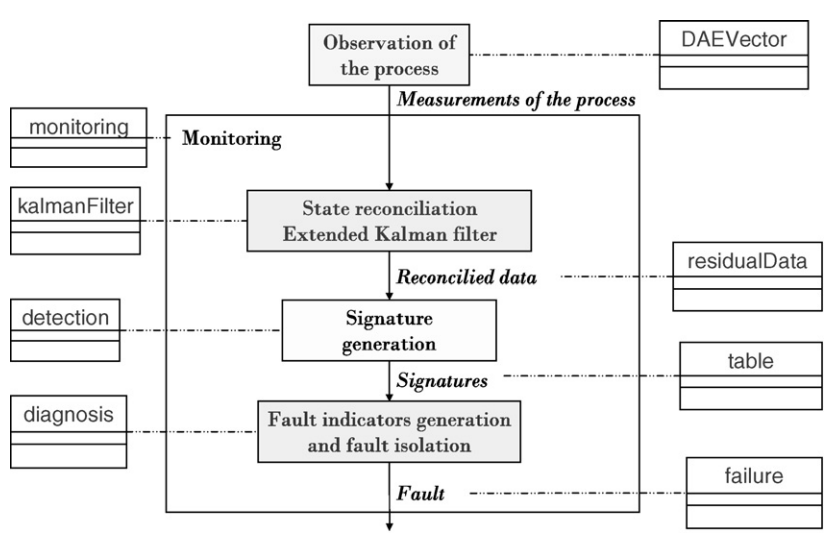

Fig. 6. Architecture of the "Monitoring" module.

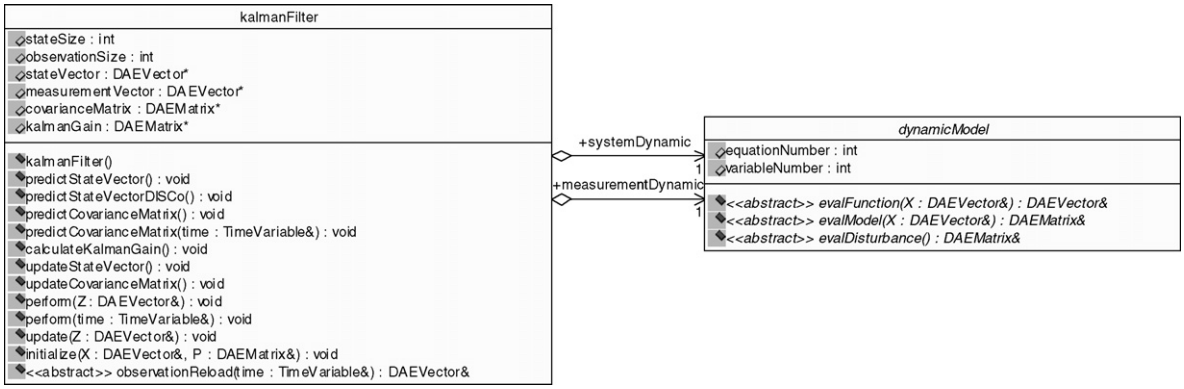

Fig. 7. State reconciliation UML diagram.

sion. The "supervision" layer rests on both layers "simulation" and "modelling" and provides general and autonomous entities which allow studies of the control of batch processes. It is composed of the *Scheduling* module and the *Monitoring* module. Developed during the present works, the *Monitoring* module, called *PrODHySAEM*, performs the studies of the process monitoring tasks.

4.2. PrODHySAEM module

The *PrODHySAEM* module is based on our methodology. Thus, each fundamental element of our approach (filtering, residual generation, detection, etc.) is described by an object class (Fig. 6). In this context, the next section presents a detailed description of these elements and their implementations, by means of UML diagrams. The various interactions which characterize them are underlined.

4.2.1. State reconciliation module

The first step of our diagnosis approach is the state reconciliation owing to an extended Kalman filter. The class diagram (Fig. 7) gives its general structure. The instantiation of an object *kalmanFilter* does not require any argument. Its initialization is carried out by the call of the method *initialize*. The *kalmanFilter* class have two attributes: *systemDynamic* and *_measurementDynamic* of *dynamicModel* type. The first attribute represents the system dynamics and the second, the observation dynamics. A dynamic model has a number of equations (*_equationNumber*) and a number of variables (*_variableNumber*) and three methods *evalModel*, *evalFunction* and *evalDisturbance*. The *dynamicModel* class allows taking into account all the types of system: linear system or not. Thus, the calculation established in the class *kalmanFilter* is independent of the studied system. Its running is accomplished through the call of one of the available possibilities of the method *perform*. A description of the extended Kalman filter can be found in Olivier-Maget et al. (2008).

4.2.2. Detection module

The following step consists of the generation of the residuals and the signatures. The detection is represented by the class *detection* (Fig. 8). It has three tables: *_signature*, *_residual* and *_epsilon* which respectively represent the observation window of the signatures, of the residuals and of the detection thresholds. These three tables are created respectively by the call of the methods *signatureGeneration*, *residualGeneration* and *epsilonUpdate*. The method *perform* gathers the calls of the previous methods. The detection is started by a call to the method *perform*.

In the methodology *SimAEM*, a residual, a detection threshold or a signature is an entity carrying information represented by the class *residualData* (Fig. 8). This class is built by specialization of the class *residual*. Thus, the class *residualData* has three attributes: the current time (*_time*) and two inherited attributes, the size of the state vector (*_stateSize*) and the data characterizing an object *residualData* (*_dataValue*).

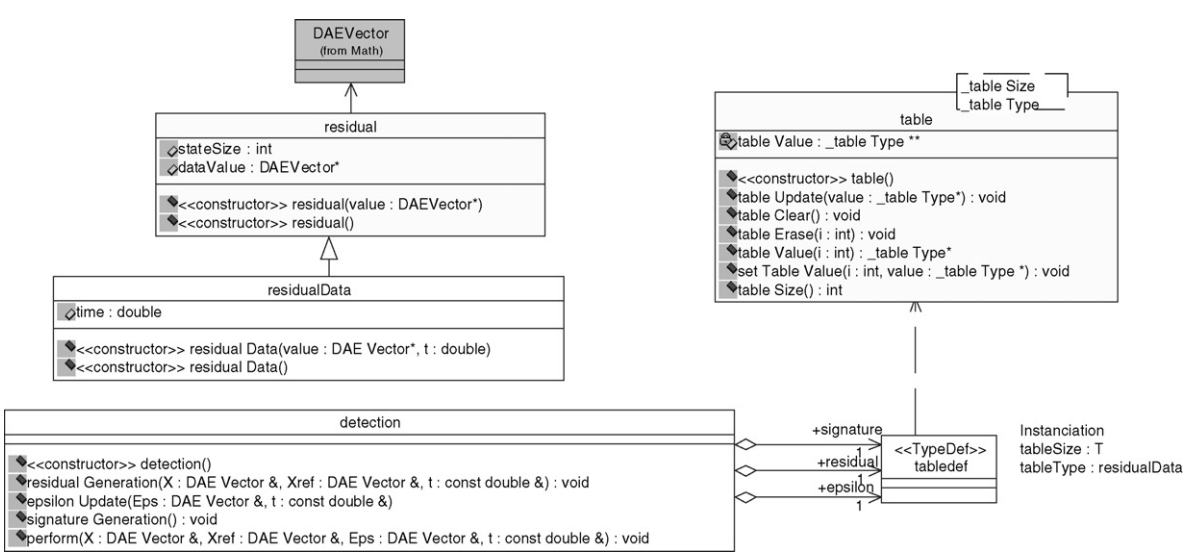

Fig. 8. UML diagram of the detection structure.

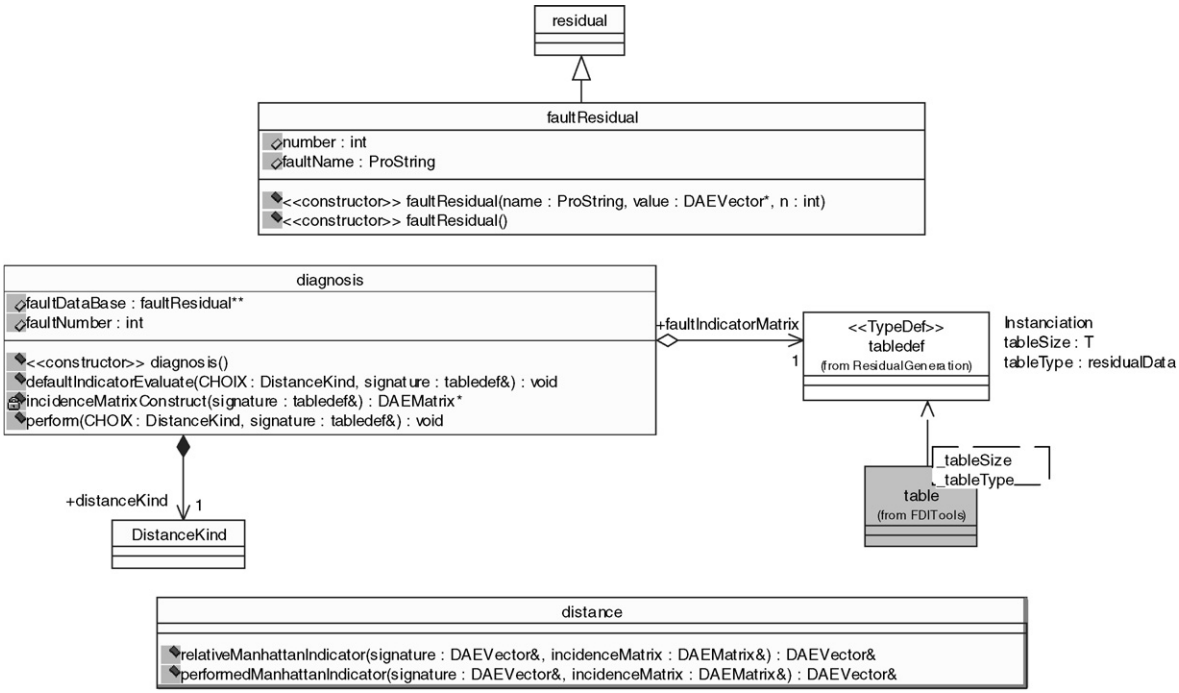

Fig. 9. UML diagram of the diagnosis structure.

The class *table* is defined to describe a matrix (Fig. 8). It is a generic class. It is characterized by its dimension *_tableSize* and its data type *_tableType*. These two attributes are generic parameters. The goal is to define a class independent of the handled object type. When an object is instantiated, its type is given. Then, various tables can be created. Moreover, the class has an attribute which is a pointer to the object *_tableValue*. The method *tableUpdate* allows adding an element to the table by shifting the observation window. The method *tableErase* erases an element of the table, while the method *tableClear* erases all the elements of the table.

4.2.3. Diagnosis module

The diagnosis will provide the causes of the detected failure owing to the instantaneous signature generated during the next step. It uses the real process *instantaneous fault signature* and the modelled (or previously obtained by experiments) *theoretical fault signatures*. A fault signature of defect is represented by the class *faultResidual*, which inherits the class *residual* (Fig. 9). Its specialization involves the addition of two new attributes: an integer *_number* which represents the population size and a character string *_faultName* which is the name of the fault.

The last step of the methodology *SimAEM* consists of the localization and the identification of the fault(s). The class diagram of the diagnosis module is illustrated in Fig. 9. The diagnosis is based on the incidence matrix (structure which contains all the theoretical fault signatures). This matrix is built owing to the call of the method *incidenceMatrixConstruct* and the fault database (*_faultDataBase*). The method *defaultIndicatorEvaluate* generates the fault indicators and updates the table *_faultIndicatormatrix*. The indicators are evaluated by means of distance between the theoretical fault signatures (incidence matrix) and the instantaneous fault signature obtained during the detection. The method *perform* gathers the calls of the two methods *incidenceMatrixConstruct* and *defaultIndicatorEvaluate*. The call to this method starts the diagnosis.

The utility class *distance* (Fig. 9) contains all the distance calculation methods. The choice of one of them is done through the argument *CHOIX* of type *DistanceKind* in the method *perform* of the class *diagnosis*. Currently, only the relative (4) and improved Manhattan (5) distances are developed.

4.2.4. Monitoring module

The monitoring module is represented by the class *monitoring* (Fig. 10). It has a set of attributes and methods in order to implement the monitoring function. The monitoring is composed of three main stages: the creation of the monitoring module, its initialization and its performance. This is carried out by the calls respectively to the constructor, to the method *initialize* and to the method *perform*. The call to the constructor of the monitoring class induces the call to the constructor of the classes *detection*, *diagnosis* and *kalmanFilter*. The monitoring module is initialized by the call to the method *initialize*. The method perform of the class *monitoring* contains the call to the methods *perform* of these objects: *_kalmanFilter*, *_detection* and *_diagnosis*.

In this very general level, the components of the monitoring module (*_detection*, *_diagnosis* and *_kalmanFilter*) are described by high level classes; these classes can be specialized in the classes of the derived classes *monitoring*.

5. Modelling of the monitored process

5.1. Process modelling with PrODHyS

5.1.1. General structure of the simulation model

The simulation of a discontinuous process necessitates to model separately the control part (the supervisory control) and the operative part (the process).

Concerning the operative part, the specification of any device of *PrODHyS* is always defined according to two axes: a topological axis and a phenomenological axis. The topological axis defines the structure of the process (system vision): physical connections (material, energy, information) between the different parts of the process and hierarchical decomposition of the devices. The phenomenological axis rests on the *ODPN* formalism, which manages a set of mathematical models based on mass and energy balances and, thermodynamic and physico-chemical laws. Thus, the models

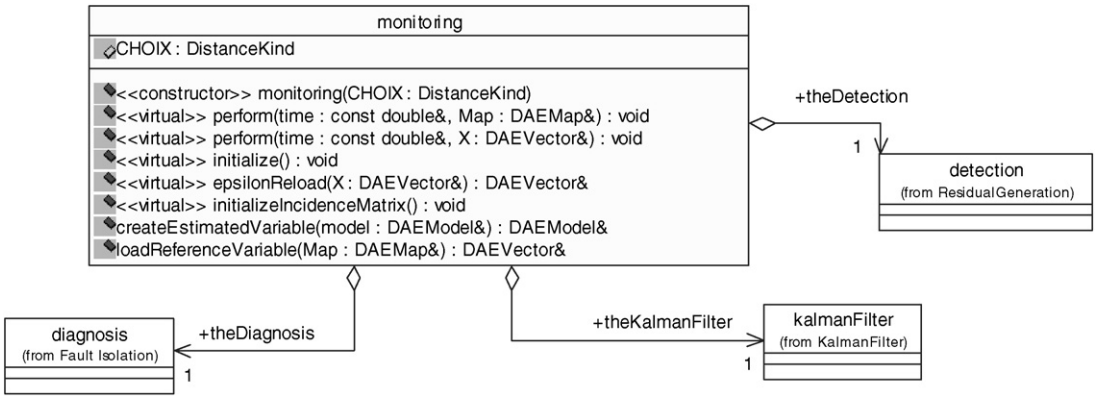

Fig. 10. Monitoring UML diagram.

of devices are reusable whatever the context. In addition, the combined approach is used to dissociate the model of material from the model of devices which contains the material. Therefore, object tokens are reusable and reduce the complexity of the devices Petri nets. More details on the modelling of devices and material can be found in previous articles (Hétreux et al., 2007; Perret et al., 2004).

On the other hand, the model of the command part is specific to the recipe and the process topology. It consists in describing the procedure of manufacture of each product. So, it specifies the assignments of resources and the sequence of tasks ordered in time necessary to the realization of each batch.

5.1.2. Connections between "devices" PN and "recipe" PN

The exchanged signals between the command part and the operative part are modelled by a discrete place. The state of a signal state is associated with the marking of the corresponding place. In this framework, an entity is either an active device if it has one or more signal places (such as valves, pumps, feeds, column, sensors) or a passive device if there is no direct relation with the recipe (such as simple tanks or reactors). These notions are illustrated in Fig. 11. It represents an operative sequence which permits the feed of a tank until a fixed volume is reached. In this example, the *DAE* systems corresponding of each differential places are the following ones:

- $\begin{cases} F_{\text{out}}^{\text{V1}} = 0 \\ F_{\text{out}}^{\text{V1}} - F_{\text{in}}^{\text{V1}} = 0 \end{cases}$ for the differential place "ON" of the valve V1;
- $\begin{cases} F_{\text{out}}^{\text{V1}} = F_{\text{O}} \\ F_{\text{out}}^{\text{V1}} - F_{\text{in}}^{\text{V1}} = 0 \end{cases}$ for the differential place "OFF" of the valve V1;
- and $\mathrm{d}U_{\text{l}}/\mathrm{d}t - F_{\text{in}}^{\text{T1}} + F_{\text{out}}^{\text{T1}} = 0$ for the differential place "NOT EMPTY" of the tank ST1.

Note that the model does not have explicit connection equation such as: $F_{\text{out}}^{\text{V1}} = F_{\text{in}}^{\text{T1}}$. This relationship is implicit through the connection of ports. Thus, for example, $F_{\text{out}}^{\text{V1}}$ and $F_{\text{in}}^{\text{T1}}$ are the same variable in the simulation model.

The marking of the signal place of an active entity induces the evolution of its Petri net. This Petri net can itself induce the evolution of active or passive entities in cascade through the net composed with the connection of different material or energy ports.

5.2. FDI simulation model

This methodology has firstly been tested by using the simulation of the faulty process which is needed anyway to build the incidence matrix. As a matter of fact, during the learning of the incidence matrix, this simulation provides some knowledge of the abnormal behaviour, which we do not have a priori. Moreover, for safety reasons, it is interesting to test and validate the monitoring module by simulation rather than in a real industrial context.

5.2.1. Representation

Our monitoring module requires the simulation in parallel of three process recipes (Fig. 12): one for the reference simulation, another one for the state reconciliation by the extended Kalman filter, and the last one for the representation of the real process. Note that the simulation of the real process takes by default into account the parametric and measurements disturbances. So the signal of a sensor is perceived as noisy by the monitoring module.

Moreover, the real process is potentially faulty. For its representation, its flowsheet is only composed of potentially faulty devices. A faulty device is described by a specific Petri net, which inherits the Petri net of the corresponding non-faulty device. In this Petri net, a

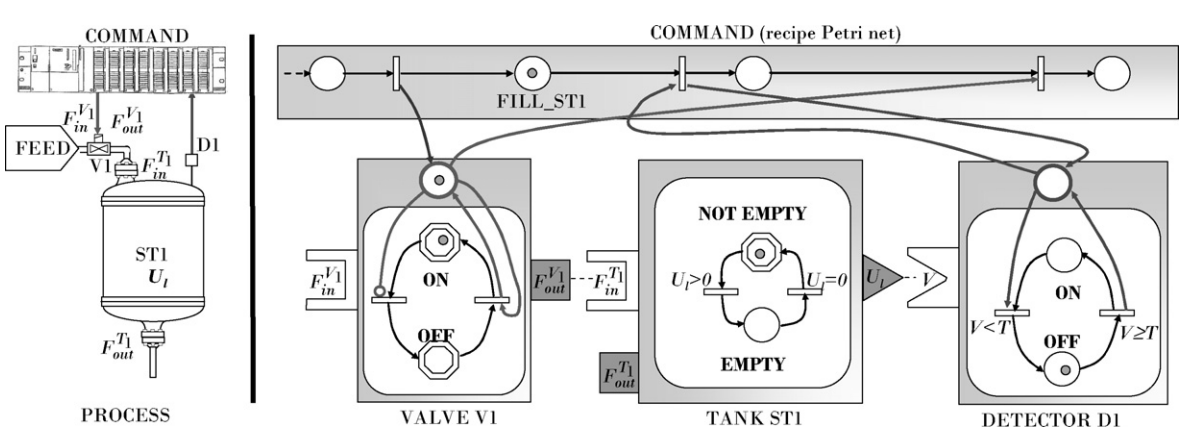

Fig. 11. Interactions between the command level and the process level.

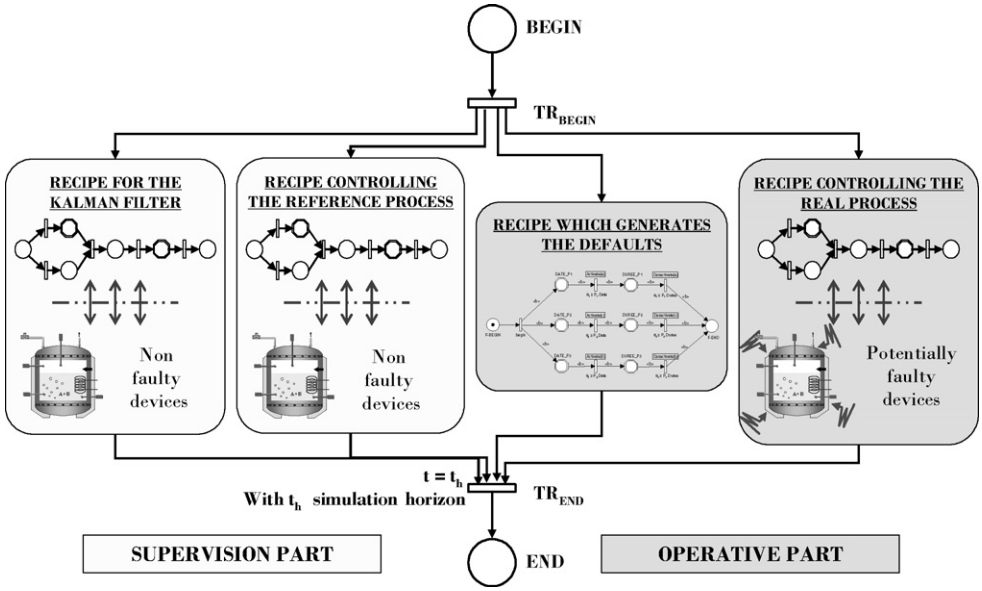

Fig. 12. Main Petri net.

set of places and transitions are added to represent the abnormal behaviour of the device.

Next, the device faults are generated by a fourth recipe. Finally, to perform a study of the process monitoring, four recipes are used: both of them for the operative part and the other two for the supervision part. Notice that when we will use the monitoring module with the real process, there will only be the recipes of the supervision part. In order to play the four Petri nets, they are gathered within the same Petri net. The fire of the transition T_{RBEGIN} leads to the autonomous play of each recipe. However, a recipe can interact with the other ones owing to actions on its transitions.

5.2.2. Modelling of a faulty device

In order to build the simulation model of the monitored process, the fault appears in the modelling of specific *Device* objects in which the faults are defined in an intrinsic way. Moreover, for a system analysis, the failures are generated by a recipe owing to a fault calendar which lists the faults, their occurrence times and their durations. Next, the failures are obviously generated on autonomous and random commutations. Then, a failure appears uncertainly and the monitoring must be able to detect and isolate the fault. In fact, the first stage aims at validating the adjustment of the monitoring system, which is used during the test step.

Fig. 13 represents the modelling of a faulty sensor. In the studied case, we consider the occurency of the fault "*a*": the sensor stays in off position instead of being in on position.

At the occurency date, within the failure generator Petri net, the fire of the transition t_2 leads to the activation of the fault "*a*". So, within the faulty sensor Petri net, that allows the fire of the transition t_{R1} and the marking of the place *F_off*. When the duration of the fault is passed, the transition t_3 is fired and so the fault "*a*" is deactivated. Next, within the faulty sensor Petri net, the deactivation of the fault "*a*" allows the fire of the transition t_{R2} and then the return to the normal conditions.

Notice that the parameter duration of a fault allows the taking into account of intermittent and permanent faults.

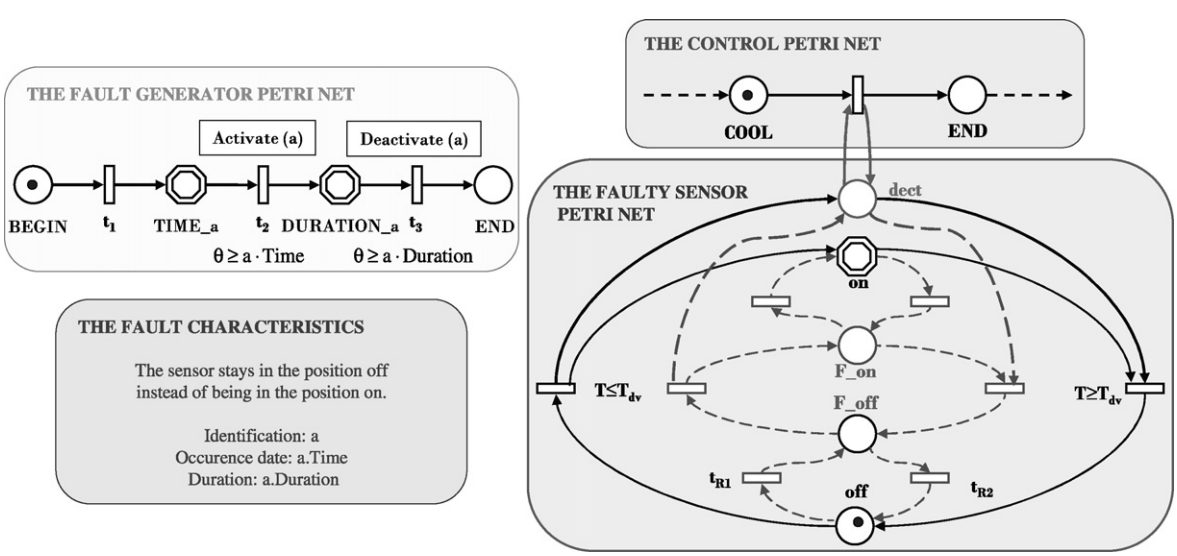

Fig. 13. Modelling of a faulty sensor.

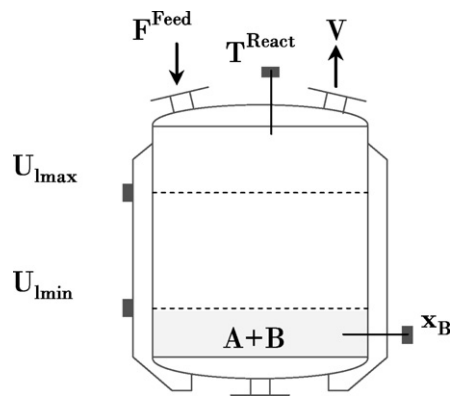

Fig. 14. Studied process.

Table 1
Geometrical parameters.

Tank height	100 cm
Tank diameter	50 cm
Heating energy	3,000 W
Cooling energy	−20,000 W

6. Application: a process unit operation

6.1. Description

The process of addition–evaporation is generally used to change solvents. Its recipe describes a succession of evaporations and additions of the new solvent. This process is illustrated in Fig. 14. This system is composed of a heat tank, four sensors, an energy feed and a material feed. The geometrical parameters of this system are listed in Table 1. The values of the minimum and maximum holdups are respectively 200 and 800 moles. Before each addition of solvent, the reactor is cooled up to the temperature of 300.15 K. The pressure is supposed to be constant during this operation. The goal of this process is to have a molar composition of methanol in the reactor at 0.95. The operation conditions are listed in Table 2. According to the operation condition, the initial liquid phase may vapourise with a flow rate V.

6.2. Modelling of the system

The Petri net of the whole system is presented in Fig. 15. The recipe Petri net is presented by a hatched pattern; it is the control part. The grey Petri net represents the functioning of a device. This is the operative part. The places with waved and dotted patterns are the signal places. They represent the exchanged signals between the command and operative parts, when a command is sent (in dotted pattern) or when information is received (in waved pattern). For example, the marking of the place *m-ON* of the composition sensor means that the molar composition of methanol has reached 0.95 and the marking of the place *m_Open* of the material feed activates the command of the opening of the material feed.

Table 2
Operating conditions.

	Reactor	Material feed
T (K)	298.15	298.15
P (atm)	1	1
$x_{A=\text{water}}$	0.6	0.01
$x_{B=\text{methanol}}$	0.4	0.99
U_l (mol)	300 (8.1 L)	–
Flow rate (mol/min)	–	5

Next, we present the description of the mathematical model of each device.

6.2.1. Modelling of a sensor

The sensor recipe is only composed of discrete places. These places represent a state of the device:

- The state becomes *ON* when the desired value (v_{dv}) is reached by the variable v.
- On the contrary, the state remains *OFF*.

The monitored variable v is different for each sensor:

- v is the temperature of the reactor T^{reac} for the temperature sensor.
- It is the molar composition of methanol x_B in the reactor for the composition sensor.
- And it is the holdup U_l of the reactor for the holdup sensors.

6.2.2. Modelling of a feed

The feed Petri net is composed of a discrete place, which represents the state *CLOSED*, and a differential place for the state *OPEN*. The *DAE* system of the place *OPEN* is:

- $F^{feed} - F_O = 0$ for the material feed,
- $Q^{heat} - Q_O^{heat} = 0$ for the heating energy system of the reactor,
- and $Q^{cool} - Q_O^{cool} = 0$ for the cooling energy feed. Quote that Q^{cool} is negative to follow the energy sign convention.

6.2.3. Modelling of the reactor

Finally, the reactor has a material input and two energy inputs. When the place *NOT_EMPTY* is marked, a *DAE* system is instantiated. The outlet vapour is opened on the outside. The pressure P^{react} is supposed to be constant. Consequently, the mathematical model of the reactor at the thermodynamic equilibrium and in its maximal state (i.e. liquid/vapour) is composed of:

- the global material balance: $\frac{dU_l}{dt} - F^{feed} + V = 0$;
- the partial material balances: $\frac{d(U_l \cdot x_j)}{dt} - F^{feed} \cdot x_j^{feed} + V \cdot y_j = 0$ for $j = A, B$;
- the energy balance: $\frac{d(U_l \cdot h)}{dt} - F^{feed} \cdot h^{feed} + V \cdot H - Q^{heat} - Q^{cool} = 0$;
- the liquid level: $h_l - \frac{U_l \cdot V_{ml}}{S_t} = 0$;
- the liquid/vapour equilibrium: $\begin{cases} y_j = K_j \cdot x_j, & \text{for } j = A, B \\ x_A + x_B - y_A - y_B = 0 \end{cases}$;
- the model of the liquid/vapour equilibrium constant: $K_j - mK_j(T, P, x, y) = 0, \quad \text{for } j = A, B$;
- the constraint on the liquid enthalpy: $h - mh(T, P, x) = 0$;
- the constraint on the vapour enthalpy: $H - mH(T, P, y) = 0$;
- the constraint on the liquid molar volume: $V_{ml} - mV_{ml}(T, P, x) = 0$;
- the constraint on the vapour molar volume: $V_{mV} - mV_{mV}(T, P, y) = 0$;
- the constraint on the pressure: $P - mP(p) = 0$.

Parts of these models are only merged when a simulation model is instantiated. Thus, the size and the structure of the resulting DAE systems change during the simulation, according to the actual state of the process.

This established model is used by the extended Kalman filter in order to build both the state and the system output. Then, the estimation is compared with the real process and the residual is analyzed in order to detect the faults.

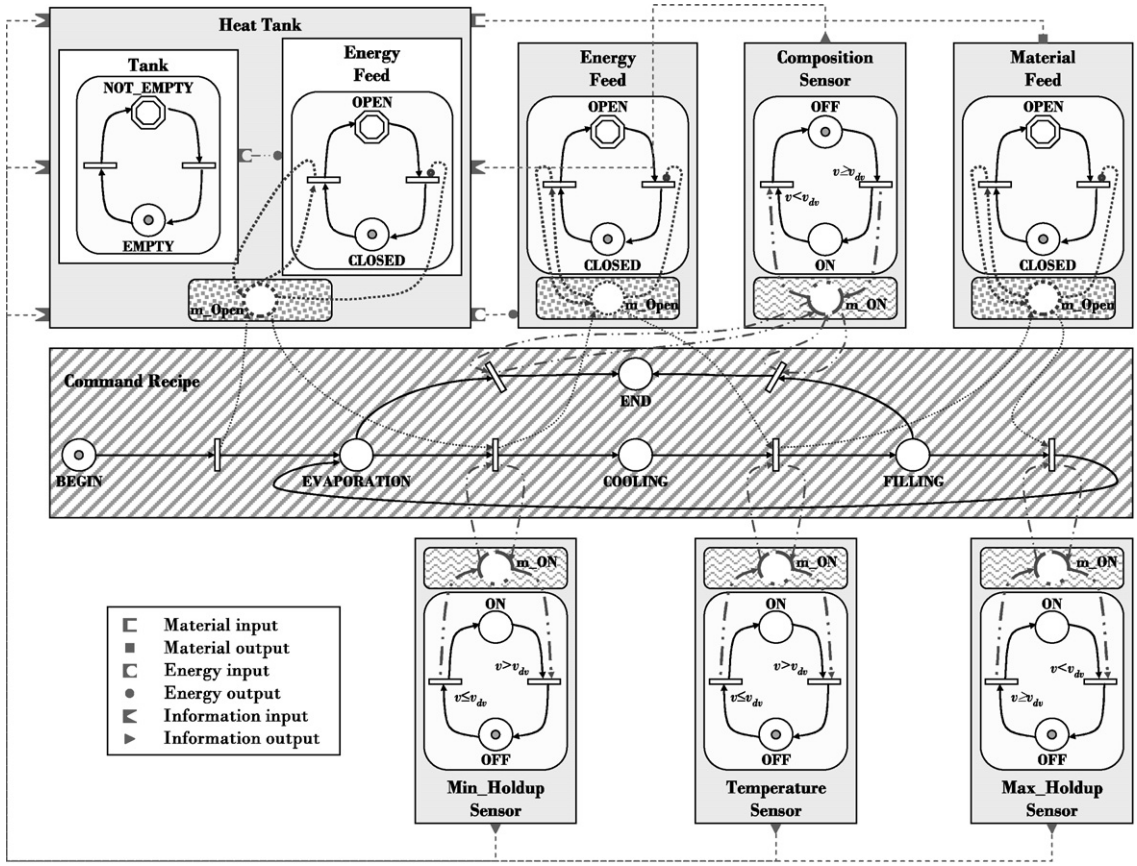

Fig. 15. System Petri net.

6.3. Adjustments of the monitoring module

To perform a monitoring of a process, some off-line adjustments must be made. Firstly, we evaluate the covariance matrices of the model and measurement disturbances. While the measurement noises are well-known by experience or by the sensor manufacturer, the model disturbances are estimated by an "ensemble method". We make numerous simulations during which a model parameter is disturbed. This allows the estimation of statistics of the model mistakes. Then, if the behaviour of the system goes beyond this distribution, its behaviour is abnormal. So, the detection thresholds are determined according to the model disturbances.

Next, the second adjustment is the learning of the incidence matrix. It is based on the same "ensemble theory". For this, we make a set of simulations, during which a fault is introduced at different occurrence dates, for each potential state of the hybrid dynamic system. For example, Fig. 16 represents the results obtained for the introduction of a fault: the material feed provides material with a degraded flow rate. The signatures of this fault are represented for different occurency dates. They have the same pattern. The centroid is then calculated providing the theoretical signature of this fault.

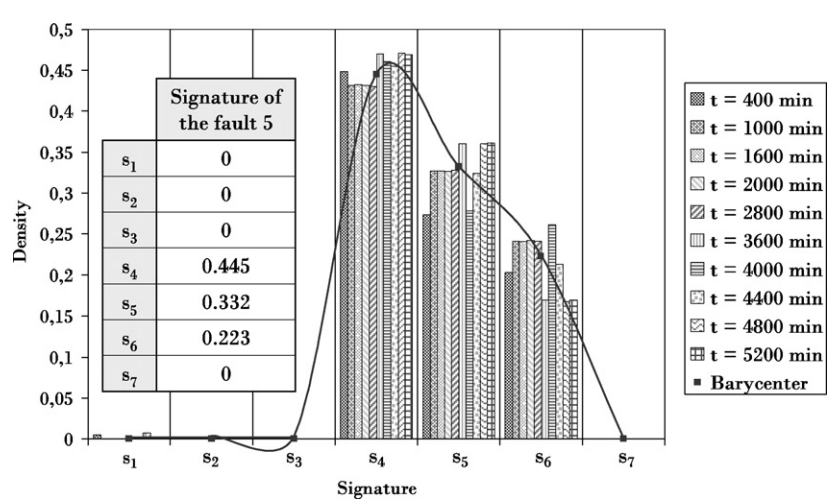

Fig. 16. Learning of the incidence matrix.

Table 3
Incidence matrix.

	Fault 1	Fault 2	Fault 3	Fault 4	Fault 5	Fault 6	Fault 7	Fault 8	Fault 9
T_{tank}	0.005	0.18	0.003	0.147	0	0	0.012	0.128	0.016
y_{water}	0	0	0	0	0	0	0.476	0	0
$y_{methanol}$	0	0	0	0	0	0	0.484	0	0
x_{water}	0.395	0.23	0.427	0.338	0.445	0.428	0.01	0.377	0.531
$x_{methanol}$	0.4	0.345	0.324	0.341	0.332	0.324	0.018	0.286	0.175
U_{ltank}	0.2	0.245	0.246	0.174	0.223	0.248	0	0.209	0.278
U_{vtank}	0	0	0	0	0	0	0	0	0

Next, this previous study is made for all the considered faults of the system and we obtain the following incidence matrix (Table 3):

Note that the faults 1, 2, 3, 4, 8 and 9 would be confused, if we were working with binary values. The work with real and non-binary values aims at distinguishing the importance of the symptoms between different faults. Therefore, for a given fault, the norm allows that the main symptoms are underlined.

7. Results

7.1. Detection results

This process is a system based on thermal phenomena. A fault on the tank thermal system is a risk for the success of this operation. This is the reason why it is important to detect it as soon as possible.

We recall that the thresholds for the detection correspond to the model uncertainties obtained through the extended Kalman filter computation. As example, let us consider that a fault on the heating energy feed of the reactor takes place at $t = 20$ min. This energy feed provides a heat quantity lower than the nominal one. Fig. 17 shows the detection stage. It illustrates the time evolution of the residuals for the liquid composition of both water and methanol. From $t = 80$ min, the both residuals no longer characterize the nominal performance. The diagnosis is launched at $t = 95$ min.

7.2. Diagnosis results

Table 4 presents the construction of the synthetic structures, the signature, which is relative and normed.

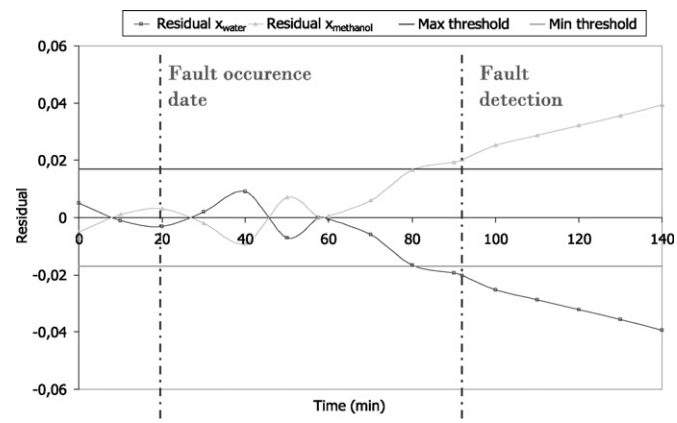

Fig. 17. Evolutions of the composition residuals during the evaporation stage.

Consider only the residuals obtained by Eq. (1). If we compare the residuals 1 (the tank temperature T_{tank}) and 2 (the water vapour molar composition in the tank y_{water}), we can suppose that the deviation of the variable 1 (T_{tank}) is greater than the one of the variable 2 (x_{water}). However, these residuals are not without dimensions: the residual of the variable 1 (T_{tank}) is in Kelvin and the one of the variable 2 (x_{water}) is dimensionless. So, to compare the residuals, both residuals must be dimensionless. The use of relative residuals (Eq. (2)) allows the comparison and then proves the opposite case: the deviation of the variable 2 (x_{water}) is greater than the one of the variable 1 (T_{tank}).

Table 4
Construction of the signature.

	Variable	Residual	Relative residual	Relative detection threshold	Relative normed signature
1	T_{tank}	6.5	−0.01864371	0.010038922	0.0044098
2	y_{water}	0.272512	−1	0.036695632	0.49367559
3	$y_{methanol}$	0.727488	−1	0.020618897	0.50191462
4	x_{water}	0.01461	−0.02377117	0.024405721	0
5	$x_{methanol}$	−0.01461	0.03790965	0.038921612	0
6	U_{ltank}	(12.611	0.04388129	0.104388129	0
7	U_{vtank}	0	0	5.00E−20	0

Table 5
Fault indicators of the example.

	Fault	Manhattan relative indicator	Manhattan improved indicator
1	The up holdup sensor detects a value higher than the nominal value.	0.71428571	0.605
2	The up holdup sensor detects a value lower than the nominal value.	0.71554566	0.7254961
3	The temperature sensor detects a value higher than the nominal value.	0.71428571	0.64
4	The temperature sensor detects a value lower than the nominal value.	0.71554566	0.7104961
5	The material feed provides material with a degraded flow rate.	0.71714286	0.645
6	The heating energy feed of the reactor has a temperature lower than the nominal one.	0.71428571	0.645
7	The heating energy feed provides a heat quantity lower than the nominal value.	0.99819303	0.75330735
8	The energy feed used for the cooling of the reactor has a temperature higher than the nominal one.	0.71554566	0.7104961
9	The energy feed used to the cooling of the reactor provides a heat quantity lower than the nominal value.	0.71428571	0.585

Table 6
Diagnosis results.

	Fault								
	1	2	3	4	5	6	7	8	9
Fault 1									
Manhattan relative indicator	0.993	0.907	0.964	0.950	0.945	0.936	0.714	0.950	0.893
Improved Manhattan indicator	0.983	0.888	0.915	0.923	0.872	0.856	0.5	0.923	0.781
Faults 5 and 6									
Manhattan relative indicator	0.857	0.849	0.856	0.857	0.857	0.857	0.857	0.856	0.857
Manhattan improved indicator	0.856	0.859	0.922	0.898	0.940	0.941	0.875	0.898	0.863

When the relative residuals are compared to the relative detection thresholds, only three residuals characterize the abnormal behaviour (Table 4). Then, the instantaneous fault signature underlines this fact, since for the other variables the value of the signature is equal to zero. The symptoms of the abnormal behaviour are the deviations of the variables 1 (T_{tank}), 2 (y_{water}) and 3 (y_{methanol}).

Next, we compare the instantaneous fault signature (Table 4) with the theoretical fault signatures (Table 3), by calculating the relative and improved Manhattan distances (Eqs. (3) and (4)). Then, the fault indicators are generated (Table 5).

The relative Manhattan indicator detects the presence of the fault 7 with a probability of 99.8%. Nevertheless, all the other possible faults are not rejected, since indicators of some of them are higher than 0.68. On the opposite, with the improved Manhattan indicator, the faults 1, 3, 5, 6 and 9 are eliminated, since their indicators are lower than 0.68. So, the possibilities are the faults 2, 4, 7 and 8. This example underlines the importance to use both indicators to conclude. So, by combining the results of both indicators, we can rule on the presence of the fault 7, because its indicators are the highest. For this reason, this fault is the most probable. Then, the fault is located on the energy feed of the reactor. Furthermore, it has been identified: the heating energy feed of the reactor provides a heat quantity lower than the nominal value. Notice that the fault comes from an actuator.

7.3. Other results

In this last part, we present other results of diagnosis (Table 6).

During the first simulation, the fault 1 is introduced. According to both hypotheses used for the decision (explained in Section 3.4) and by combining the results of both indicators, we can rule on the presence of the fault 1. This is a fault of a sensor.

Finally, during the last simulation, two faults (5 and 6) take place simultaneously. The results of the relative Manhattan indicator cannot distinguish the faults. Conversely, the results of the improved Manhattan indicators underline the faults 3, 5 and 6. The fault 3 has the lowest indicators, so the faults 5 and 6 are emphasized. Nevertheless, we cannot be sure of the presence of one or both faults. Notice that the simulation of these three scenarios could remove the ambiguity.

8. Conclusion

In this study, the feasibility of using the simulation as a tool for fault detection and diagnosis is described. The developed method lies on the hybrid dynamic simulation software *PrODHyS*, which is based on an object-oriented approach. Our fault diagnosis approach is a general method for the detection and isolation of the occurency of a fault. Besides, this approach allows the detection of numerous types of fault and has the ability to underline the simultaneous occurency of many faults. The works in progress aim at integrating this simulation model within a model-based supervision system. The goal is to define a recovery solution following the diagnosis of a fault. For this, we exploit the results of signatures in order to generate qualitative information. Moreover, the computation of real indicators instead of binary ones make possible to distinguish a simple degradation from a failure. This information will be used as information in the supervisory control structure to improve the recovery solution. Finally, present works are devoted to develop a generalised diagnosis structure combining the model-based diagnosis approach presented in this paper with other methods, such as classification and case based reasoning.

Appendix A. Nomenclature

A	water
B	methanol
D	distance
D^{Mr}	relative Manhattan distance
D^{Ma}	performed Manhattan distance
ε	detection threshold
F	liquid flow rate (mol s^{-1})
g	gravity constant (m s^2)
h	liquid enthalpy (J/mol)
H	vapour enthalpy (J/mol)
h_{l}	liquid level in the tank (m)
in	input
K	liquid/vapour equilibrium constant
mh	model for the liquid enthalpy
mH	model for the vapour enthalpy
mK	model for the liquid/vapour equilibrium constant
mP	model for the pression
mV_{ml}	model for the liquid molar volume
mV_{mV}	model for the vapour molar volume
n	number of state variables
O	order
out	output
p	set of the operative parameters
P	pressure (Pa)
Q	energy quantity (W)
r	residual
r^r	relative residual
S	signature
S^{rN}	relative normed signature
S_{t}	area (m^2)
t	time
T	temperature (K)
$T_{\bullet,\text{j}}$	theoretical signature of the fault j
$T_{i,j}$	incidence matrix
U_{l}	liquid holdup (mol)
v	variable
V	vapour flow rate (mol s^{-1})
V_{ml}	liquid molar volume (m^3/mol)
V_{mV}	vapour molar volume (m^3/mol)
x	liquid molar composition
$\hat{X}$	estimated system state vector
X	system state vector
y	vapour molar composition

References

Alur, R., & Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science*, *126*(2), 183–225.

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T. A., Ho, P. H., Nicollin, X., et al. (1995). The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, *138*, 3–34.

Anderson, T. W. (1984). *An introduction to multivariate statistical analysis*. New York: Wiley.

Andersson, M. (1994). *Object-oriented modelling and simulation of hybrid systems*. PhD thesis, Lund Institute of Technology, Sweden.

Barton, P. I., & Pantelides, C. C. (1994). The modelling of combined discrete/continuous processes. *AIChE Journal*, *40*, 966–979.

Berthomieu, B., & Menasche, M. (1983). *IFIP congress series* (Vol. 9, pp. 41–46).

Birouche, A. (2006). *Contribution sur la synthèse d'observateurs pour les systèmes dynamiques hybrids*. Thèse de doctorat, Institut National Polytechnique de Lorraine, Nancy, France.

Branicky, M. S. (1995). *Studies in hybrid systems: Modeling, analysis and control*. PhD thesis, MIT, MA, USA.

Buisson, J., & Cormerais, H. (1998). Descriptor systems for the knowledge modelling and simulation of hybrid physical systems. *Journal Européen des systèmes automatisés*, *32*(9–10), 1047–1072.

Champagnat, R., Esteban, P., Pingaud, H., & Valette, R. (1998). Modeling and simulation of a hybrid system through Pr–Tr PN-DAE model. In *ADPM'98* Reims, France, (pp. 131–137).

Chow, E. Y., & Willsky, A. S. (1984). Analytical redundancy and the design of robust failure detection system. *IEEE Transactions on Automatic Control*, *29*(7), 603–614.

David, R., & Alla, H. (1990). Autonomous and timed continuous Petri nets. In *11th international conference on application and theory of Petri nets* Saragosse, Spain, June.

David, R., & Alla, H. (2001). On hybrid Petri nets. *Journal of Discrete Event Dynamic Systems: Theory and Applications*, *11*, 9–40.

De Kleer, J., & Williams, B. C. (1987). *Artificial Intelligence*, *32*, 97–130.

Demongodin, I. (2001). *Discrete Event Dynamic Systems: Theory and Applications*, *11*(1–2), 137–162.

Deshpande, A., Gollu, A., & Semenzato, L. (1998). The shift programming language for dynamic networks of hybridautomata. *IEEE Transactions on Automatic Control*, *43*(4), 584–587.

Evensen, G. (2003). The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean Dynamics*, *53*, 343–367.

Fábián, G., van Beek, D. A., & Rooda, J. E. (1998). Integration of the discrete and the continuous behaviour in the hybrid chi simulator. In *European simulation multiconference* Manchester.

Frank, P. M. (1990). Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy—a survey and some new results. *Automatica*, *26*(3), 459–474.

Gertler, J., & Singer, D. (1990). A new structural framework for parity equation based failure detection and isolation. *Automatica*, *26*(2), 381–388.

Hétreux, G., Théry, R., Olivier, N., & Le Lann, J. M. (2007). Exploitation de la simulation dynamique hybride pour la conduite de procédés semi-continus. *Journal Européen des Systèmes Automatisés JESA, Supplément MOSIM*, *41*(5), 585–616.

Jourda, L., Joulia, X., & Koehret, B. (1996). Introducing ATOM, the applied thermodynamic object-oriented model. *Computer & Chemical Engineering*, *20A*, S157–S164.

Kesten, Y., & Pnueli, A. (1992). Timed and hybrid statecharts and their textual representation. *Lecture Notes in Computer Science (LNCS)*, *571*.

Koller, D., & Lerner, U. (2001). Sampling in factored dynamic systems. In *Sequential Monte Carlo methods in practice, ser. Statistics for engineering and information science*. New York: Springer., pp. 445–464.

Le Bail, J., Alla, H., & David, R. (1991). Hybrid Petri nets. In *Proceedings of the European control conference* Grenoble, (pp. 1472–1477).

Lee, S. K., & White, P. R. (1998). The enhancement of impulsive noise and vibration signals for fault detection in rotating and reciprocating machinery. *Journal of Sound and Vibration*, *217*(3), 485–505.

Lunze, J. (2000). Diagnosis of quantised systems by means of timed discrete-event representations. *Computer Science*, *1790*, 258–271.

McIlraith, S., Biswas, G., Clancy, D., & Gupta, V. (2000). Hybrid systems diagnosis. *Computer Science*, *1790*, 282–295.

Mehra, R. K., & Peschon, J. (1971). An introduction approach to fault detection and diagnosis in dynamic systems. *Automatica*, *5*, 637–640.

Mosterman, P., & Biswas, G. (1999). Diagnosis of continuous valued systems in transient operating regions. *IEEE Transactions on Systems Man and Cybernetics Part A: Systems and Humans*, *29*(6), 554–565.

Moyse, A. (2000). *Odysseo: Plate-forme orientée-objet pour la simulation dynamique des procédés*. PhD thesis, INP de Toulouse, France.

Olivier-Maget, N., Hétreux, G., LeLann, J. M., & LeLann, M. V. (2008). Integration of a failure monitoring within a hybrid dynamic simulation environment. *Chemical Engineering and Processing*, *47*(11), 1942–1952.

Perret, J., Hétreux, G., & Le Lann, J. M. (2004). Integration of an object formalism within a hybrid dynamic simulation environment. *Control Engineering Practice (Elsevier)*, *12*(10), 1211–1223.

Sargousse, A. (1999). *Noyau numérique orienté-objet dédié à la simulation des systèmes dynamiques hybrides*. Thèse de Doctorat, INPT, France.

Shih, R., & Lee, L. (1995). Use of fuzzy cause–effect digraph for resolution fault diagnosis for process plants. *Industrial and Engineering Chemistry Research*, *34*(5), 1688–1717.

Sifakis, J. (1977). *Use of Petri nets for performance evaluation, measuring, modelling and evaluating computer systems*. Amsterdam: North Holland., pp. 75–93.

Simani, S., & Fantuzzi, C. (2006). Dynamic system identification and model-based fault diagnosis of an industrial gas turbine prototype. *Mechatronics*, *16*, 341–363.

Valentin-Roubinet, C. (1998). *Proceedings of Automation of Mixed Processes (ADPM 98)* Reims, France, March 19–20, (pp. 142–149).

Venkatasubramanian, V., Rengaswamy, R., Yin, K., & Kavuri, S. N. (2003). A review of process fault detection and diagnosis. *Computers & Chemical Engineering*, *27*(3), 293–346.

Welch, G., & Bishop, G. (1995). *An introduction to the Kalman filter*. Technical Report TR 95-041, University of North Carolina.

Willsky, A. S. (1976). A survey of design methods for failure detection in dynamic systems. *Automatica*, *12*, 601–611.

Wöllhaf, K., Fritz, M., Schulz, C., & Engell, S. (1996). BaSiP—batch process simulation with dynamically reconfigured process dynamics. *Computers & Chemical Engineering*, *20*(2), S1281–S1286.

Zhao, F., Koutsoukos, X., & Haussecker, H. (2005). Monitoring and fault diagnosis of hybrid systems. *IEEE Transactions on Systems Man and Cybernetics Part B: Cybernetics*, *35*(6), 1225–1240.

Zaytoon, J. (2001). *Systèmes dynamiques hybrides*. Hermès Sciences Publications.

Further reading

Evensen, G. (2003). The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean Dynamics*, *53*, 343–367.