Modeling complex systems with Heterogeneous Petri Nets (HtPN)

Amaury Vignolles, Elodie Chanthery, Pauline Ribot

Modeling complex systems with Heterogeneous Petri Nets (HtPN)

Amaury Vignolles[a,b], Elodie Chanthery[a,b], Pauline Ribot[a,c]

[a] *CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France*
[b] *Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France*
[c] *Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France*

Abstract

This article presents a new formalism based on Petri nets to model heterogeneous systems and take uncertainties into account: the Heterogeneous Petri Nets (HtPN). A formal definition of heterogeneous systems is introduced: a class of systems that can present continuous, discrete or hybrid dynamics by subparts. After presenting the formalism and how it allows to specify the behavior of heterogeneous systems, the formalism is applied on an hybrid example, a production system from Motorola.

Keywords: Modeling, Simulation, Heterogeneous Systems, Petri Nets

1. Introduction

Modeling correctly a system is paramount, as it allows, among other things, to precisely specify, control and monitor the considered system. However, with the global complexification of systems, specially manufacturing systems, it is necessary for modeling formalisms to evolve as well. For this purpose a new formalism based on the well-known Petri Nets formalism [1] is introduced in this paper. This formalism, named Heterogeneous Petri Nets (HtPN), was created to fulfill specifics needs, although we claim that it can be used to represent everything the usual Petri Net can do.

Email addresses: avignolles@laas.fr (Amaury Vignolles), echanthe@laas.fr (Elodie Chanthery), pribot@laas.fr (Pauline Ribot)

The specific needs we focus on are modeling and monitoring the health of a heterogeneous system under all kinds of uncertainty. In short, a heterogeneous system, according to our definition, is a system in which purely discrete, purely continuous or hybrid parts (i.e. parts mixing discrete and continuous aspects) are all linked and communicate with each other. For example, in the domain of manufacturing systems, cyber-physical systems [2] main characteristic is the different nature of their elements. They integrate various heterogeneous devices which have heterogeneous dynamics. Hence, we need a formalism able to represent any type of system. We also need to be able to represent different types of uncertainty (on modeling or observations, for example) through parallelism or noise functions associated to places of the Petri Net. Finally, monitoring the health of manufacturing systems has become such an important challenge for the industry that we wanted to define a formalism that is easy and natural to understand and appropriate. The formalism must therefore also be able to represent the aging of the system, through degradation dynamics for example. To sum up, the new formalism should make it possible to:

- model and monitor any kind of system, be it discrete, continuous, hybrid or heterogeneous. This needs includes parallelism representation for multi-component systems;
- represent uncertainty on the system, be it of modeling or because of problems on observations (noise or communication problems);
- monitor the health of the system and follow its degradation process through dynamics associated to places of the network, for example.

A survey of the existing formalisms will exhibit that classical Petri Nets or their usual extensions (colored Petri Nets, Hybrid Nets...) do not comply with these needs. We thus propose the Heterogeneous Petri Nets based on the work of [3]. This formalism is not only able to fulfill our needs for health monitoring, but also to simulate a control system. We implemented a software to simulate systems modeled with HtPN. This implementation was applied on a production

system from Motorola already defined in the literature [4].

To sum up, our main contributions in this article are:

- the definition of heterogeneous systems;
- the specification of the new HtPN formalism based on Petri Nets able to monitor the health of complex heterogeneous systems under uncertainty as well as simulate control systems;
- a software implementation to simulate models of heterogeneous systems in the proposed formalism.

This paper is organized as follows. Section 2 defines Heterogeneous systems. Section 3 surveys related works and shows the need for a new formalism. Section 4 introduces our new formalism to represent Heterogenous systems, the Heterogeneous Petri Nets (HtPN). Finally, Section 5 shows how this new formalism was applied to simulate the model of a control system taken from the litterature, which represents a production system from Motorola.

2. Heterogeneous Systems

2.1. Definitions

Although the notion of heterogeneous systems (HtS) exists in the literature, the formal definition of this type of systems based on their dynamics has not yet been given to our knowledge or is often too restrictive. For example, in [5], authors consider systems where discrete models and continuous models have to communicate during simulation. Hybrid behaviors are not considered. Other works are not dynamics-based and simply consider heterogeneous systems as the integration of diverse specific components in various domains such as the electrical, mechanical and optical fields [6, 7]. This section aims at proposing a dynamics-based definition of Heterogeneous systems (HtS). To better understand this definition, the definitions of Discrete Event Systems (DES), Continuous Systems (CS) and Hybrid Systems (HS) must be recalled.

Definition 1 (Discrete Event System). *A Discrete Event System [8] is a system which will only manage discrete data:*

- *the state space is a discrete set;*
- *the state transition mechanism is event-driven.*

That is its state evolution depends entirely on the occurrence of asynchronous discrete events over time.

Some events are observable, whereas some are unobservable (like some spontaneous fault events for diagnosis purpose for example). If continuous data are encountered by a DES, they will be abstracted to generate discrete events.

Definition 2 (Continuous System). *A Continuous System [8] is a system with continuous time dynamics.*

The evolution of such a system can be described by a dynamic equation C of the form:

$$C = \begin{cases} x_{k+1} &= \mathbf{f}(x_k, u_k) &+ \mathbf{v}(x_k, u_k) \\ y_k &= \mathbf{h}(x_k, u_k) &+ \mathbf{w}(x_k, u_k) \end{cases} \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$ is the continuous state vector of n state variables at time k, $u_k \in \mathbb{R}^{n_u}$ is the vector of n_u continuous input variables at time k, $\mathbf{f}$ is the noise-free continuous evolution function, $\mathbf{v}$ is a noise function, $y_k \in \mathbb{R}^{n_y}$ is the vector of n_y continuous output variables at time k, $\mathbf{h}$ is the noise-free output function and $\mathbf{w}$ is the noise function associated with observation.

Definition 3 (Hybrid System). *A Hybrid System [9] is a system that will encounter both discrete and continuous data at any time.*

Now that the definitions of these three kinds of systems have been reminded, we can introduce our definition of a HtS.

Definition 4 (Heterogeneous System). *A Heterogeneous System is a system that can be divided into different sub-systems. These sub-systems can be either purely discrete, purely continuous, or hybrid, and communicate together.*

From a data point of view, a HtS will sometimes be affected solely by discrete data, sometimes only by continuous data and sometimes by both continuous and discrete data.

In the context of health monitoring, the aging of the system plays an important role on the evolution of its health state. We then define aging Heterogeneous Systems (aHtS):

Definition 5 (Aging Heterogeneous System). *An aHtS is a HtS which includes the aging of the system as a continuous time function. This aging process is usually represented through degradation dynamics.*

2.2. Running Example of an HtS

Our running example is a heterogeneous system that can be found in any manufacturing process involving a water tank. A water pump has two operating modes: either it is on and pumps water, or it is off. In this system, the pump can get stuck and the system will enter a faulty state and shut down. When the pump is on, there are three ways for the system to stop: the user manually turns it off, the observed (i.e. measured) water level exceeds a given threshold (50 liters here) or a fault occurs on it. The only way to turn on the pump is for the user to start it by pressing the ON button. The system can enter the faulty state from both the on and off states. When in the faulty state, the pump is considered unavailable and can not be started. When a repair action is made, the system returns into its off state.

This example falls under our definition of a Heterogeneous System. Indeed, one part of the system is hybrid (when the system is on, we consider both discrete and continuous data or observations) and another part is purely discrete (when the system is off, we only consider discrete data and observations). Both parts communicate with each other. An illustration of how this system works will of course be given with the new HtPN formalism we propose.

3. Related Work

This section aims at studying existing solutions and formalisms to deal with heterogeneous systems. Since few works deal with heterogeneous systems, we will also study the formalisms developed for hybrid systems. We will try to identify *a priori* solutions likely to satisfy the needs stated in the introduction: parallelism in various systems, representation of uncertainty both in the model and in the observations, representation and monitoring of system aging.

The theory of hybrid automata has been published in [9] but the preliminary version was published in 1996 in the Proceedings of the 11^{th} Annual IEEE Symposium on Logic in Computer Science (LICS96). In hybrid automata [10], each discrete state represents a possible state of the automaton, i.e. of the system. It is associated with continuous dynamics defining the evolution of the continuous space. In this model, only one state can be active at a time. By definition, this is incompatible with the idea of parallelism. The transitions are defined by 5-tuples of the form $(q, Guard, \sigma, Jump, q')$ with q the state before the transition, q' the state after the transition, $Guard$ the condition to fulfill in order to fire the transition, σ the event received or emitted during the transition firing and $Jump$ the changes on the variables taking place during the firing (whether it is a reset to zero or the application of a function to compute a new value). Concepts such as $Guard$ and $Jump$ are very interesting. However, even if hybrid automata composition is possible, they cannot share a common state. Therefore it is impossible to represent uncertainty concerning observations or on the current system state.

Petri nets are widely used to manage manufacturing systems [11, 12, 13, 4], even for hybrid aspects. Petri nets have the advantages to be very intuitive for modeling and designing systems and are recognized for their compactness and their relevance in decision-making and system monitoring. They are also used for proving some properties on systems. This is why this section will now focus on Petri Net based formalisms. They all have in common the use of places, transitions and tokens.

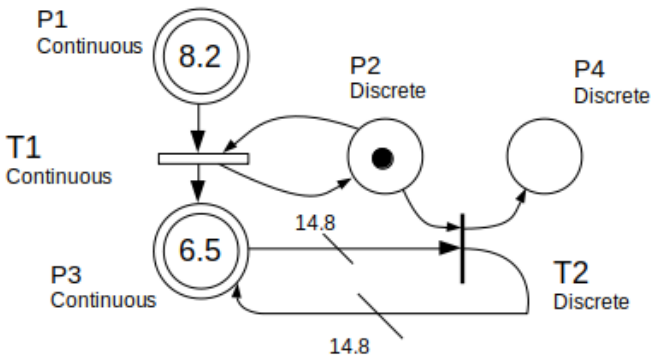


Figure 1: Example of Hybrid Petri net [4]

In hybrid Petri nets [4] there are two types of places: continuous places, represented by double line, and discrete places as shown on Figure 1. Tokens situated in continuous places are real numbers, whereas tokens in discrete places are integer. Two types of transitions can be distinguished. Continuous transitions (as a rectangle) and discrete transitions. In the case of continuous transitions, a crossing quantity is defined and acts like a weight on the arcs. It is possible for transitions to have both types of places as inputs, but the discrete place must be an input and an output of the transition, and the weights of the ingoing and outgoing arcs must be the same. This can be seen on Figure 1 with $T1$ and $P2$. In case of conflict between continuous and discrete transitions, the discrete transition has the priority. The idea of parallelism is applicable as different tokens can evolve simultaneously in the model. However, continuous places are not associated with any dynamics. It is therefore impossible to associate to a state variable an evolutionary dynamic according to the state of the system and to monitor it, or to make a degradation variable evolve according to a level of stress associated with a state of the system.

In mixed Petri nets, proposed by [14], a place can be continuous and associated to dynamics. In this case, one, or few, differential equation is associated to the place. A place can also represent a discrete phenomenon. The continuous variables evolution is modeled through two sets giving the set of equations acti-

vated when a place is marked or when a marking is true. One contains the set of the equations activated when a place is marked whilst the other contains a set of equations activated for a specific marking. The idea of *Jump* and *Guard* from hybrid automaton is also present. The idea of parallelism is applicable as long as the marked places do not change the value of the same variable through equations. The continuous variables are shared with the whole system and evolve following the active equations. This is still kind of restricting according to our needs, with the idea of parallelism and uncertainty, as this formalism does not allow two places in parallel to modify the same variables.

A Petri net based model to represent heterogeneous embedded systems (PRES) was introduced in [15]. In this formalism, a token is a pair $k = (v_k, r_k)$, where v_k is the token value, of any type, and where r_k is the token time. Places are associated with a single token type. This means that a place containing a token whose value is an integer will be considered as an integer type place: this type will not change during the entire dynamic behavior of the net. In PRES, a transition is associated with an output function, which will make the token value evolve, a function delay, which will make the token time evolve and a guard, which is the condition for the transition to be fired. Although interesting ideas can be found in this work, such as the *output function*, the *function delays* and the *guard*, some aspects do not fit our requirements. For example, a place in PRES is of a particular type, and will not accept a token which is not of the type of the place. This implies that the combination of discrete and continuous behaviors is difficult. Moreover, the token value does not evolve in the places, it only evolves using the transitions and their output functions. We would like the value of the tokens to evolve even while they are in a place.

In previous works, we proposed a formalism named Hybrid Particle Petri Nets (HPPN) in [16, 17]. A diagnostic method was developed based on this formalism that can be applied only on hybrid systems. It represents uncertainty both in the model and in the observations and system aging can be represented and monitored. However, the HtPN formalism was hard to figure out, so we worked on the simplification of the concepts for a better understanding and on

References	Formalism	Hybrid? Heterogeneous?	Parallelism?	Model uncertainty?	Observation uncertainty?	System aging representation?
[9] [10]	Hybrid Automata	no	no	no	no	no
[11] [12] [13]	Petri Nets	no	✓	no	no	no
[4]	Hybrid Petri nets	Hybrid only	✓	no	✓	no
[14]	Mixed Petri nets	Hybrid only	$\simeq$	no	✓	✓
[15]	PRES	$\simeq$	✓	no	no	$\simeq$
[16] [17]	HPPN	Hybrid only	$\simeq$	✓	✓	✓

Table 1: Related Work Summary

the evolution of HPPN towards design and monitoring of heterogeneous systems.

Table 1 summarizes the work described in this section, focusing on important criteria. It indicates if the cited formalism is able to represent and monitor hybrid systems and/or heterogeneous systems; if parallelism is possible; if uncertainties on the model (resp. on the observations) may be taken into account ; if the system aging may be represented and monitored. The symbol $\simeq$ means that the property is well taken into account in the formalism but in a way that does not fullfill our needs for our health monitoring problem and for encompassing any type of Petri nets.

It emphasizes that even if each formalism can bring some interesting ideas for our approach, the definition of a new formalism dedicated for aging heterogeneous systems will answer a need.

4. The HtPN formalism

As we have seen with the related works, the existing formalisms do not fulfill our particular needs. Hence, we had to define a new formalism, extended from the classical Petri Nets. This section presents this new formalism, the Heterogenous Petri Nets (HtPN), their semantics and their firing rules.

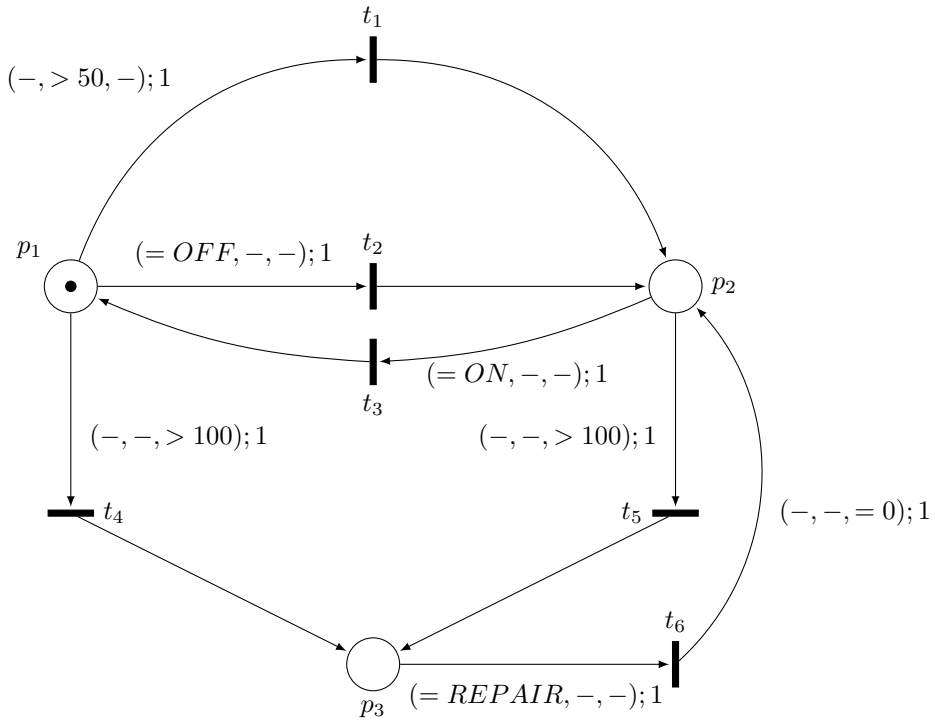
The HtS presented in Section 2.2 was modeled using HtPN and will be used as a running example to illustrate the different notions of the proposed formalism (see Figure 2).

4.1. General Presentation

A HtPN is formally defined as follows.

Definition 6 (HtPN). *A HtPN is a set of 11 elements:*
$< P, T, A, Guard, Jump, E, X, \Gamma, C, D, \mathbb{M}_0 >$ *gathering information to describe discrete, continuous and degradation dynamics through places and the relationships linking these places through transitions:*

- P *is the set of places;*
- T *is the set of transitions;*
- $A \subset (P \times T \cup T \times P)$ *is the set of arcs;*
- *Guard is the set of conditions associated with incoming arcs connecting the places to the transitions;*
- *Jump is the set of assignments associated with outgoing arcs connecting the transitions to the places;*
- E *is the set of event labels. It is the union of the set* E_o *of the labels of observable events and the set* E_{uo} *of the labels of unobservable events:* $E = E_o \cup E_{uo}$.
- $X \subset \mathbb{R}^{n_N}$ *is the state space of the continuous state vector, where* $n_N \in \mathbb{N}_+$ *is the finite number of continuous state variables;*



(a)

places	continuous dynamics	degradation dynamics
p_1	$x_{k+1} = x_k + 1$	$\gamma_{k+1} = \gamma_k + 3$
p_2	-	$\gamma_{k+1} = \gamma_k + 1$
p_3	-	-

(b)

Figure 2: Example of an HtPN (a) and dynamics associated to its places (b)

- $\Gamma \subset \mathbb{R}^{n_D}$ *is the state space of the degradation state vector, where* $n_D \in \mathbb{N}_+$ *is the finite number of degradation state variables;*
- C *is the set of continuous dynamics of the system;*
- D *is the set of degradation dynamics of the system;*

- $\mathbb{M}_0$ *is the initial marking of the network.*

4.1.1. Places

In HtPN, a place is an object with a continuous dynamic, and a dynamic of degradation. Discrete information is represented by the place itself.

Hence, to a place p are associated a set of equations $C_p \in C$ modeling continuous dynamic of the system and the associated noise (uncertainties on the evolution and on the measurements) as well as a set of equations $D_p \in D$ modeling degradation dynamic of the system:

$$p = \begin{Bmatrix} C_p \\ D_p \end{Bmatrix} \tag{2}$$

The set of equations C_p is defined as:

$$C_p = \begin{cases} x_{k+1} = f(x_k, u_k) + v(x_k, u_k) \\ y_k = g(x_k, u_k) + w(x_k, u_k) \end{cases} \tag{3}$$

where $x_k \in X$ is the continuous state vector at time k, $u_k \in \mathbb{R}^{n_u}$ is the vector of the n_u continuous input variables, f is the noiseless continuous evolution function, v is the noise function of the continuous evolution, $y_k \in \mathbb{R}^{n_y}$ is the vector of the n_y continuous output variables, g is the noiseless continuous output function, and w is the noise function of the continuous output. Functions f, v, j and w are dependent on the considered p place.

The set of equations D_p is defined as:

$$D_p = \left\{ \gamma_{k+1} = d(\gamma_k, b_k, x_k, u_k) + z(\gamma_k, b_k, x_k, u_k) \right. \tag{4}$$

where $\gamma_k \in \Gamma$ is the degradation state vector and d is the noiseless hybrid degradation function. It depends on discrete events represented by b_k and continuous events represented by x_k, and z is the noise function of the degradation evolution. The functions d and z are dependent on the considered p place. In the example in Figure 2, there are 3 places: p_1, p_2 and p_3 and the associated continuous and degradation dynamics are explained in Table 2 (b).

A place may have no associated continuous and/or degradation dynamics. In this case, the $-$ symbol is used. This is illustrated in Table 2 (b) with the continuous dynamics of p_2 or the degradation dynamics of p_3.

By default, if no dynamics are specified, the place p is defined as:

$$p = \left\{ \begin{matrix} - \\ - \end{matrix} \right\} \tag{5}$$

4.1.2. Tokens

A place p contains $n_H(p)$ tokens ($n_H(p) \geq 0$). Each token h have three attributes: a discrete attribute, a continuous attribute and a degradation attribute. These three attributes are represented as a set $< \delta_k, \pi_k, \phi_k >$, with δ_k representing discrete information at time k, π_k representing continuous information at time k and ϕ_k representing degradation information at time k. These attributes evolve according to discrete events and dynamics associated to the place p the token belongs to.

The discrete information carried by a token h is called a **configuration**. The configuration of a token is the set of events that have occurred in the system up to the time k and whose occurrence explains the existence of the token. More formally, δ_k is the set b_k of events that occurred up to time k:

$$b_k = \{(v, \kappa) | \kappa \leq k\}$$

where (v, κ) represents an event $v \in E$ that occurs at time κ.

The continuous information carried by a token is called the **state of the token**. The state π_k represents the continuous state vector $x_k \in X$ of the system at time k. The state of a token h evolves according to the continuous dynamics C_p (see Equation 3) of the place it belongs to. If no continuous dynamic is specified, the state of the token will not evolve and will therefore remain constant.

The degradation information carried by a token is called the **status of the token**. The status ϕ_k represents the degradation status vector $\gamma_k \in \Gamma$ at time k. The status of a token h evolves according to the degradation dynamics D_p (see

Equation 4) of the place it belongs to. If no degradation dynamic is specified, the status of the token will not change and will therefore remain constant.

Definition 7 (Marking). *The marking $\mathbb{M}_k$ of a HtPN at time k is the distribution of tokens in the different places of the network.*

Initial marking $\mathbb{M}_0$ represents the initial conditions of the system. Each token carries its initial configuration (the set of events that have occurred until time 0), its initial continuous state and its initial degradation status.

4.1.3. Arcs

The set of arcs is divided into two subsets: $A_{\bullet t}$ which contains all incoming arcs connecting the places to the transitions and $A_{t\bullet}$ which contains all outgoing arcs connecting the transitions to the places:

$$A = A_{\bullet t} \cup A_{t\bullet} \tag{6}$$

Incoming arcs. An arc $a_{p,t} \in A_{\bullet t}$ connecting a place $p \in P$ to a transition $t \in T$ wears a set $\Omega_{p,t} \in Guard$. This set is composed of two elements:

$$\Omega_{p,t} = \left\{(\Omega^S_{p,t}, \Omega^N_{p,t}, \Omega^D_{p,t}); \rho_{p,t}\right\} \tag{7}$$

- a triplet of conditions (a symbolic condition $\Omega^S_{p,t}$, a numerical condition $\Omega^N_{p,t}$ and a degradation condition $\Omega^D_{p,t}$)
- a weight $\rho_{p,t} \in \mathbb{N}^+$.

By default, this set is $\Omega_{p,t} = \{(\top, \top, \top); 1\}$, which means that if nothing is specified, the symbolic, numerical and degradation conditions are set to TRUE (i.e. they are basically satisfied), and the weight to 1. If an element is omitted in the definition of the arc, either the triplet of conditions or the weight, this element will take its default value.

The **symbolic condition** $\Omega^S_{p,t}$ is a condition related to the configuration of the tokens located in the input places of a transition t. This condition can be set to TRUE ($\top$), FALSE ($\bot$) or tests the occurrence of one (or more, in

the case of a logical equation) event $v \in E$. In this case, it takes the form $\Omega^S_{p,t}(\delta_k) = occ(b_k, v)$ (which is true if $v \in b_k$).

The **numerical condition** $\Omega^N_{p,t}$ is a condition related to the state of the tokens in the input places of the transition t. It can be set to TRUE ($\top$), FALSE ($\bot$) or represents a constraint on the continuous state vector. In this case, $\Omega^N_{pt}(\pi_k) = c(x_k)$ is a test on the continuous state vector x_k.

The **degradation condition** $\Omega^D_{p,t}$ is a condition related to the status of the tokens in the input places of the transition t. It can be set to TRUE ($\top$), FALSE ($\bot$), or represents a constraint on the degradation status vector. In this case, $\Omega^D_{p,t}(\phi_k) = c(\gamma_k)$ is a test on the degradation state vector γ_k.

Examples of guards can be seen in the running example (see Figure 2 (a)):

1. Ω_{p_1,t_1} contains a numerical condition to test if the continuous state vector is $x_k > 50$ and no symbolic or degradation conditions,
2. Ω_{p_1,t_2} contains a symbolic condition to check if the event OFF occurred and no numerical or degradation conditions,
3. Ω_{p_1,t_4} contains only a degradation condition to test if the degradation state vector is $\gamma_k > 100$.

Pre is the matrix containing the values of weights of arcs connecting the places to the transitions, of dimensions $P \times T$. $Pre(t)$ is thus a vector containing the values of weights of arcs connecting the input places to a given transition t. $Pre(p,t)$ represents the value of the weight of the arc connecting a place p to a transition t. The value of $Pre(p,t)$ allows to know if an arc $a_{p,t}$ exists in the HtPN:

$$(Pre(p,t) \neq 0) \equiv \exists a_{p,t} \tag{8}$$

Definition 8 (Accepted token). *A token h is said to be accepted by an incoming arc if it satisfies:*

- *either the set of symbolic and numerical conditions of the arc,*
- *or the degradation condition of the arc.*

More formally, let $p \in P$ be a place such that $p \in P \wedge Pre(p,t) \neq 0$:

$$\begin{aligned} &\forall h \in p, Accept(h, a_{p,t}) \equiv \\ &\big(< \delta_k, \pi_k, \phi_k > \ | \ ((\Omega^S_{p,t}(\delta_k) = \top) \wedge (\Omega^N_{p,t}(\pi_k) = \top)) \vee (\Omega^D_{p,t}(\phi_k) = \top)\big) \end{aligned} \tag{9}$$

We note $H_a(a_{p,t}, p)$ the set of tokens in the place p which are accepted by the arc $a_{p,t}$:

$$h \in H_a(a_{p,t}, p) \equiv (Accept(h, a_{p,t}) = \top) \tag{10}$$

The weight $\rho_{p,t}$ of an arc connecting a place p to a transition t represents the minimum number n_{Ha} of accepted tokens required to validate the arc t.

Definition 9 (Validated arc). *Let consider an arc $a_{p,t}$, n_{Ha} the number of tokens accepted by $a_{p,t}$ present in the input place of the arc t and $\rho_{p,t}$ the weight of the arc. The arc $a_{p,t}$ is said to be validated if $n_{Ha} \geq \rho_{p,t}$.*

Outgoing arcs. An arc $a_{t,p} \in A_{t\bullet}$ connecting a transition $t \in T$ to a place $p \in P$ carries a set $\Omega_{t,p} = \big\{(\Omega^S_{t,p}, \Omega^N_{t,p}, \Omega^D_{t,p}); \rho_{t,p}\big\} \in Jump$. As for the incoming arcs, this set has two elements:

- a triplet of assignments (a symbolic assignment $\Omega^S_{t,p}$, a numeric assignment $\Omega^N_{t,p}$ and a degradation assignment $\Omega^D_{t,p}$),
- and a weight $\rho_{t,p} \in \mathbb{N}^+$.

The symbol $-$ for an assignment means that no change is made to the concerned attribute. By default, $\Omega_{t,p} = \{(-,-,-); 1\}$, which means that no assignment is specified. A weight equals to 1 means that only one token will be put in the output place of t.

Configuration evolution The symbolic assignment $\Omega^S_{t,p}$ concerns the configurations of the tokens passing through the arc $a_{t,p}$. Let δ_k be the configuration of a token h passing through this arc at time k and wearing the value b_k:

- if $\Omega^S_{t,p} = v$, where $v \in E$, the event v is concatenated with the current configuration of the token passing through the arc:

$$b_{k+1} \longleftarrow b_k \cup (v, k+1) \tag{11}$$

- if $\Omega^S_{t,p} = b_{new}$, where b_{new} is a set of timed events, the configuration is completely reset and only contains b_{new}:

$$b_{k+1} \longleftarrow b_{new}, \tag{12}$$

- else if $\Omega^S_{t,p} = -$:

$$b_{k+1} \longleftarrow b_k. \tag{13}$$

State assignment The numerical assignment $\Omega^N_{t,p}$ concerns the state of the tokens passing through the arc $a_{t,p}$. Let π_k be the state of a token h crossing the arc at time k. Suppose that π_k carries x_k,

- if $\Omega^N_{t,p} = x_{new}$, where x_{new} represents a new numerical value for the token state π_k then:

$$x_{k+1} = x_{new}, \tag{14}$$

- else if $\Omega^N_{t,p} = -$:

$$x_{k+1} = x_k. \tag{15}$$

The numerical assignment $\Omega^N_{t,p}$ provides the initial condition for the state of the token passing through the arc, then the set of equations $C_p \in C$ defined in Equation 3 determines the evolution of the state of the token in the output place p.

Status assignment The degradation assignment $\Omega^D_{t,p}$ concerns the status of tokens passing through the arc $a_{t,p}$. Let ϕ_k be the status of a token h crossing the arc at time k and γ_k be the value of ϕ_k,

- if $\Omega^D_{t,p} = \gamma_{new}$, where γ_{new} is a numerical value:

$$\gamma_{k+1} = \gamma_{new}, \tag{16}$$

- else if $\Omega^D_{t,p} = -$:

$$\gamma_{k+1} = \gamma_k. \tag{17}$$

The degradation assignment $\Omega^D_{t,p}$ provides the initial condition for the status of the token passing through the arc, then the set of equations D_p defined in Equation 4 determines the evolution of the status of the token in the output place p.

An example of a status assignment can be seen in Figure 2 (a) : Ω_{t_6,p_2} sets the status of the token to 0.

Weights $Post$ is the matrix containing the values of weights of arcs connecting the transitions to the places, of dimensions $P \times T$. $Post(t)$ is thus a vector containing the values of weights of arcs connecting the given t transition to the output places. $Post(t,p)$ corresponds to the weight of the arc connecting the transition t to the place p. The value of $Post(t,p)$ allows to know if an output arc $a_{t,p}$ exists:

$$(Post(t,p) \neq 0) \equiv \exists a_{t,p} \tag{18}$$

The weight $\rho_{t,p}$ defines the number of tokens to be put in the output place of the arc $a_{t,p}$, i.e. whether tokens will be duplicated or destroyed, and, if so, in what quantity. A weight $\rho_{t,p}$ less than the number of tokens used to fire the transition will cause the deletion of some of those tokens. A weight greater than the number of tokens used to fire the transition will result in the duplication of some of the tokens. This deletion or duplication will be performed randomly. In Figure 3, we observe both a case of deletion of tokens (a-b) and a case of duplication of tokens (c-d). In the first case, three tokens (h_1, h_2 and h_3) are used to fire the transition t, but the arc a_{t,p_2} having a weight ρ_{t,p_2} equal to 1, two tokens will be deleted during the transition firing. In the second case, only one token (h) is used to fire the transition, but the arc a_{t,p_2} having a weight ρ_{t,p_2} equal to 3, this token will be duplicated, until 3 tokens (h, h_1 and h_2) are drawn.

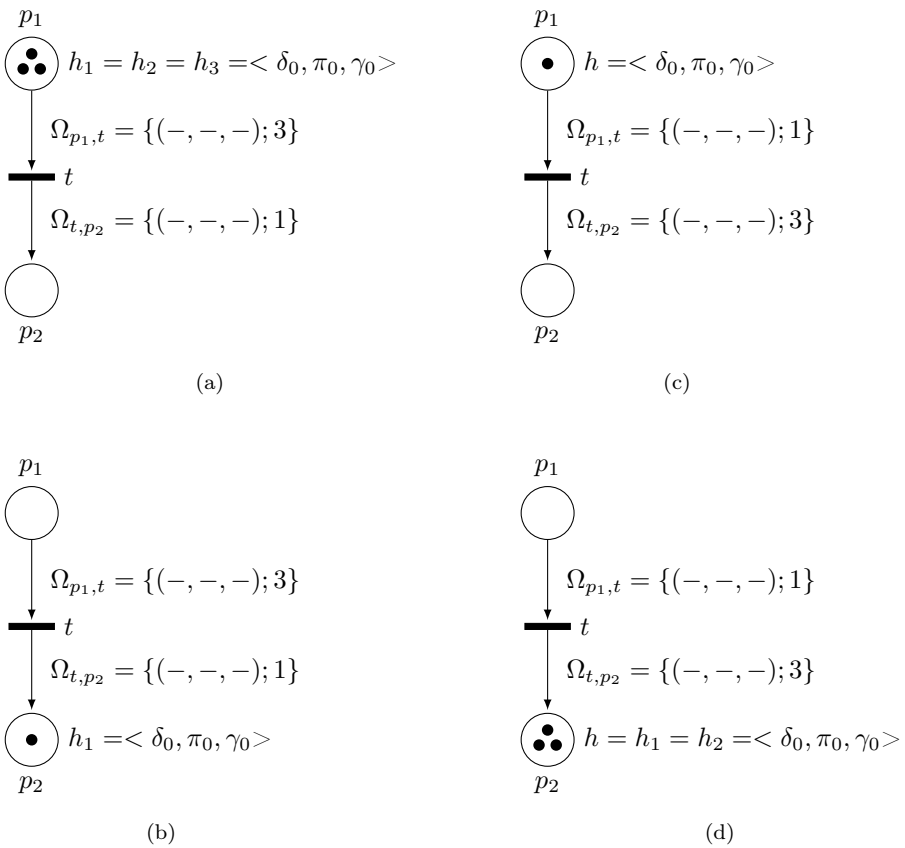


Figure 3: Weights $\rho_{t,p}$ examples

4.2. Firing rules

4.2.1. Enabled Transition

A transition $t \in T$ is said to be enabled at time k if all incoming arcs of t are validated (see Definition 9):

$$enabled(t) \equiv (\forall p \text{ s.t } a_{p,t} \in A_{\bullet t},\ n_{Ha} \geq \rho_{p,t}) \tag{19}$$

where n_{Ha} is the number of tokens accepted by the arc $a_{p,t}$, and $\rho_{p,t}$ is the weight carried by the $\Omega_{p,t}$ condition set present on $a_{p,t}$.

4.2.2. Set of fired tokens

Let $Card(H_a(a_{p,t}, p))$ be the number of accepted tokens in the place p by the arc $a_{p,t}$. When $Card(H_a(a_{p,t}, p)) > \rho_{p,t}$, a choice function has to be defined to select $\rho_{p,t}$ tokens to be fired among the tokens $H_a(a_{p,t}, p)$:

$$^{\bullet}\zeta : \mathbb{N}^{+} \times H_a \rightarrow H_a.$$

Let p such that $p \subset P \wedge Pre(p,t) \neq 0$, the set of selected tokens in the place p among accepted tokens is noted $\Psi(p,t)$ and is formally defined as follows:

$$\Psi(p,t) = \begin{cases} ^{\bullet}\zeta(\rho_{p,t}, H_a(a_{p,t}, p)) & if \quad Card(H_a(a_{p,t}, p)) > \rho_{p,t} \\ H_a(a_{p,t}, p) & otherwise. \end{cases} \tag{20}$$

From $\Psi(p,t)$ can be defined $\Psi(\bullet t)$ which is the set of tokens fired by the transition t.

Let $p_1, p_2, ..., p_i$ be the set of input places of transition t:

$$\Psi(\bullet t) = \Psi(p_1, t) \cup \Psi(p_2, t) \cup ... \cup \Psi(p_i, t) \tag{21}$$

Another choice function $\zeta^{\bullet}$ can be defined to select which tokens will be kept, duplicated or deleted among the set of fired tokens $\Psi(\bullet t)$:

$$\zeta^{\bullet} : \mathbb{N}^{+} \times \Psi(\bullet t) \rightarrow \Psi(\bullet t).$$

4.2.3. Transition firing

During a transition firing, the tokens fired by the transition t are moved into the output places of t. The attributes of fired tokens are either kept or updated. As specified in Section 4.1.3, this update, as well as the possible deletion or duplication of tokens, are defined by the set of assignments $\Omega_{t\bullet} \in Jump$ carried by the outgoing arc. As a reminder, if $\Omega_{t\bullet}$ carried by the outgoing arc carries no information, the attributes of the tokens are kept and no duplication will take place. However, deletion of tokens may occur if the number of tokens fired is greater than the weight $\rho_{t\bullet} = 1$.

The firing of a transition t at time k is formally defined as follows: $\forall p \in P \wedge Pre(p,t) \neq 0$ and $\forall p' \in P \wedge Post(t,p') \neq 0$,

$$\begin{aligned} M_{k+1}(p) &= M_k(p) - \rho_{p,t} \\ M_{k+1}(p') &= M_k(p') + \rho_{t,p'} \end{aligned} \tag{22}$$

where $\rho_{p,t}$ is the weight carried by the arc connecting the place p to the transition t, $\rho_{t,p'}$ is the weight carried by the arc connecting t to the place p', and $M_k(p)$ is the number of tokens in the place p at time k. $\mathbb{M}_k(p)$ represents all the tokens in the place p at time k:

$$\begin{aligned} \mathbb{M}_{k+1}(p) &= \mathbb{M}_k(p) \setminus \Psi(p,t) \\ \mathbb{M}_{k+1}(p') &= \mathbb{M}_k(p') \cup \zeta^{\bullet}(\rho_{t,p'}, \Psi(\bullet t)) \end{aligned} \tag{23}$$

THe HtPN formalism has been formally defined and can be used to model an heterogeneous system. In the next section, the HtPN formalism is used to model and simulate the behavior of a production system from Motorola.

5. Application

5.1. System description

To exhibit the formalism and show that it can represents any type of Petri Nets, an example of an Hybrid Petri Nets taken from [4] was represented. To reduce the computation time, the numbers in the original example were divided by 10. The chosen system represents a production system from Motorola. This production system can take care of two types of pieces coming in batch, which are called L-type and R-type. When a L-type batch arrives, it is immediately transformed into 3000 pieces, which will be continuously taken care of, and put in an upstream buffer. When 50 pieces are in this buffer, the processing of the L-type pieces will begin after waiting 30 time units (which corresponds to the set-up of the system for an L-type batch). Once all pieces of the batch are taken care of, the system goes back into an idle state and is available to process another batch.

For an R-type batch, the process is almost the same. The main difference is

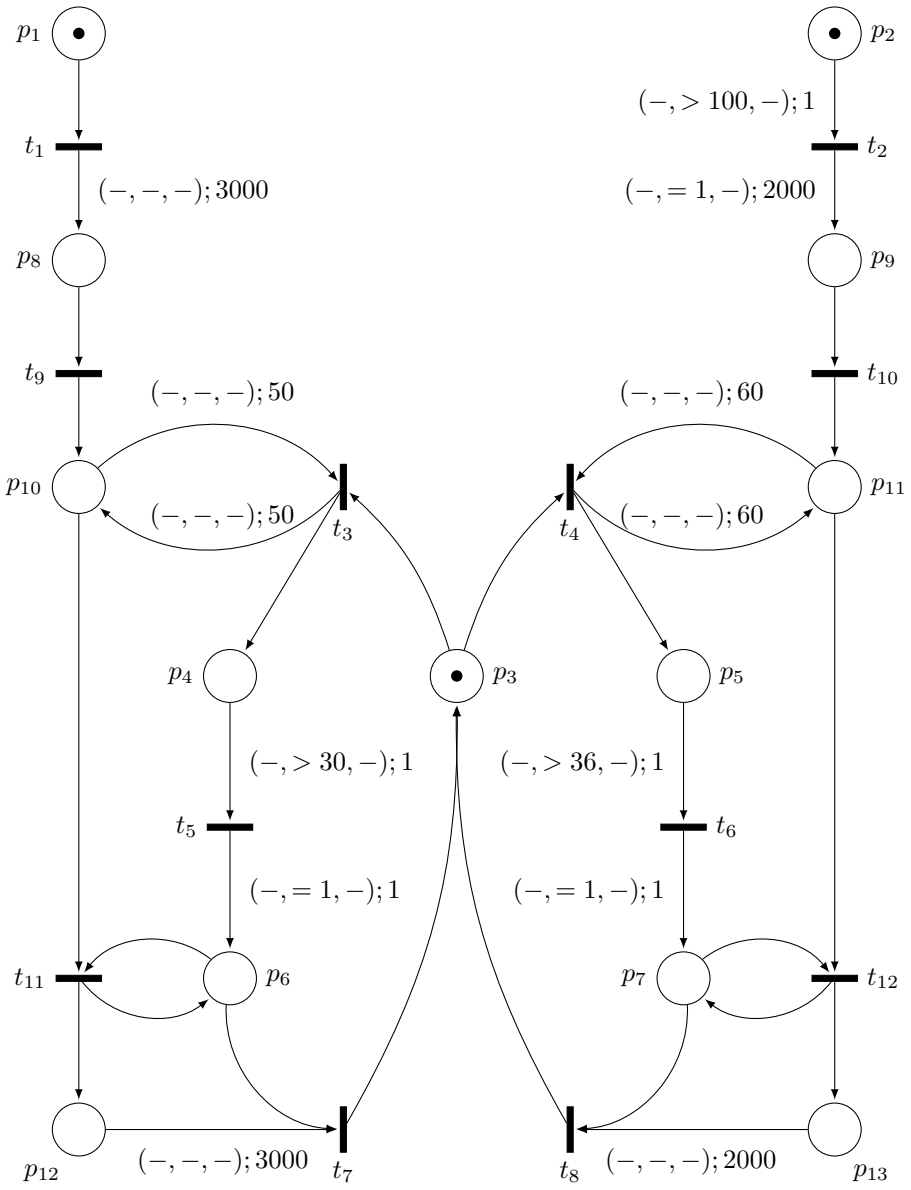


Figure 4: HtPN modeling the behavior of a production system

that the beginning of the process is delayed by 100 time units, and that the delay time before processing is set to be 36 time units.

455 The system is neither purely discrete, as the time is represented by continuous

dynamics, nor purely continuous, as the system cannot be solely represented by continuous dynamics and is in need of discrete events: it is a hybrid system. As this model was initially developed for system control, the monitoring aspect with degradation dynamics is not considered and is not represented in this model.

5.2. Modeling with HtPN

5.2.1. Model description

This system was modeled with HtPN and can be seen in Figure 4. The left part of the figure represents the L-type pieces, while the right part of the figure represents the R-type pieces. The set P of places is composed of 13 places: $P = \{p_1, \ldots, p_{13}\}$. The model is heterogeneous, as it is composed of places without any continuous dynamics (purely discrete) communicating with places having continuous dynamics (p_2, p_4 and p_5).

The initial marking is $\mathbb{M}_0 = [p_1, p_2, p_3]^T$. At $\mathbb{M}_0$, the tokens have an empty configuration (which will stay empty as the set of events E is empty), a state $\pi = [1]$ which represent the time elapsed in each place: x_k^h represents the time that the token h has passed in the place p at time k. As there are not any degradation dynamics (as the system is being controlled and its health state is not being monitored), the status ϕ of the token is set to 0.

The set of continuous dynamics C is composed of continuous dynamics incrementing the tokens state by 1 in the places p_2, p_4 and p_5, which are the places concerned by the elapsed time requirement:

$$C_{p_i} = \Big\{ \pi_{k+1} = \pi_k + 1, \quad i = 2, 4, 5 \tag{24}$$

The set T of transitions is composed of 12 transitions: $T = \{t_1, \ldots, t_{12}\}$.

A numerical condition $\Omega_{p,t}^N$ was used to represent the elapsed time since the arrival of the token in the place requirement. It is for example the case for conditions Ω_{p_2,t_2}, Ω_{p_4,t_5}, Ω_{p_5,t_6} which will be detailed latter in this section.

To simulate the numbers of pieces, the weights on the arcs were set to the needed values. For example, ρ_{t_1,p_8} was set at 3000, to simulate the fact that the batch is transformed into 3000 pieces.

The set of conditions *Guard* is mostly set at $-$, except for the following ones:

- $\Omega_{p_2,t_2} = \{(-, \pi_k > 100, -); 1\}$ to represent the 100 time units waiting in p_2
- $\Omega_{p_{10},t_3} = \{(-,-,-); 50\}$ to represent the fact that 50 pieces have to be in p_{10} before the system begins processing the L-type pieces
- $\Omega_{p_{11},t_4} = \{(-,-,-); 60\}$ to represent the fact that 60 pieces have to be in p_{11} before the system begins processing the R-type pieces
- $\Omega_{p_4,t_5} = \{(-, \pi_k > 30, -); 1\}$ to represent the 30 time units waiting in p_4
- $\Omega_{p_5,t_6} = \{(-, \pi_k > 36, -); 1\}$ to represent the 36 time units waiting in p_5
- $\Omega_{p_{12},t_7} = \{(-,-,-); 3000\}$ to represent the end of the processing of the 3000 L-type pieces
- $\Omega_{p_{13},t_8} = \{(-,-,-); 2000\}$ to represent the end of the processing of the 2000 R-type pieces.

The set of conditions *Jump* is:

- $\Omega_{t_1,p_8} = \{(-,-,-); 3000\}$ to represent the transformation of an L-type batch into 3000 pieces
- $\Omega_{t_2,p_9} = \{(-, \pi = 1, -); 2000\}$ to represent the transformation of an R-type batch into 2000 pieces and reset the state of the tokens
- $\Omega_{t_3,p_{10}} = \{(-,-,-); 50\}$ is associated with Guard Ω_{p_{10},t_3} and is the second part of the verification of the number of pieces in p_{10}
- $\Omega_{t_4,p_{11}} = \{(-,-,-); 60\}$ is associated with Guard Ω_{p_{11},t_4} and is the second part of the verification of the number of pieces in p_{11}

- $\Omega_{t_5,p_6} = \{(-, \pi = 1, -); 1\}$ to reset the state of the token
- $\Omega_{t_6,p_7} = \{(-, \pi = 1, -); 1\}$ to reset the state of the token.

The duration of the simulation is 6000 time units, as it is sufficient for the system to process both R-type and L-type batch, and return to its idle state.

5.2.2. Model evolution

When the system is idle, a token is in the place p_3. A token in p_1 represents the fact that a L-type batch is available and waiting to be transformed into pieces. This batch is turned immediatly into 3000 pieces, which will be put into the place p_8. This action is represented through the firing of the transition t_1. Transition t_9 occurs and represents the placement in the upstream buffer represented by p_{10}. When the number of pieces in p_{10} is 50, transition t_3 is fired leading to the initialisation of the system for the L-type pieces, represented by the place p_4. Then, t_5 will be fired 30 time units later, which corresponds to the duration of the initialisation for a L-type batch. A token in p_6 shows that the system's initialisation for the L-type pieces is over and that the system is ready to deal with the L-type pieces. Transition t_{11} is then enabled, meaning that the L-type pieces will be processed. Once all 3000 pieces have been processed, t_7 is fired and the systems goes back into an idle state.
For the R-type batch, the process is quite similar putting aside the delay of a 100 time units before the beginning of the process.

5.3. Results

The results of the HtPN model simulation can be found as a video available on this link: `https://www.youtube.com/watch?v=RZ8yhZum_Pw&feature=youtu.be`.

After the software is launched, we can notice that the marked places are p_1, p_2 and p_3 which represent that a L-type and a R-type batchs are waiting to be processed and that the system is available. Then, the processing of the L-type

batch begins as it is divided into 3000 pieces, placed in p_8. These pieces are immediatly transferred, one by one, to the place p_{10}. Then, when 50 pieces are in p_{10}, the transition T_3 is fired, leading to a token in p_4 (at 0min04secs on the video). After 30 time units in p_4, T_5 is fired, and a token is placed in p_6, which will lead to the repeated firing of T_{11}. The L-type batch pieces are then being processed. When the 100^{th} time unit has passed, T_2 is fired (at 0min06secs on the video). 2000 pieces from the R-type batch are placed into p_9 and transferred to p_{11} one at a time. However, the R-type pieces will stop here as the system is still manufacturing the L-type pieces. At 2min15sec, all the R-type pieces are waiting in p_{11} for the system to be available. It still has 900 L-type pieces to process before. At 3min12sec, all the L-type pieces have been manufactured. T_7 is then fired, and the system is available. As the 2000 R-type pieces are waiting to be processed, T_4 is fired immediatly and a token is placed into p_5. The system waits 36 time units and T_6 is fired, leading to the processing of the R-type pieces represented by the firing of T_{12}. After all the pieces are processed, T_8 is fired. The system is then back into its idle state and the simulation stops, as 6000 time units have passed.

6. Conclusions and future work

A definition for a heterogeneous system has been provided in this paper. For health monitoring purposes, this definition has been extended to aging Heterogeneous Systems (aHtS) in order to take into account the degradation dynamics of the system.

A new formalism based on well-known Petri Nets has been introduced and specified, the Heterogeneous Petri Nets (HtPN). This formalism can represent everything the usual Petri Net can do and more. It can represent the behavior of a complex heterogeneous system to simulate control systems for example or monitor the system health state as well. This representation allows to take into account different types of uncertainty about modeling and observations.

A software implementation (HeMU) has been realized to simulate models

of heterogeneous systems in the proposed formalism and an application to a production system from Motorola has been proposed.

A benchmark with photovoltaic panels charging batteries is being developed to apply this formalism on a real case study, simulate it and compare the simulation results to real observations.

Future work will focus on the development of the health monitoring function of such complex heterogeneous systems under uncertainty. This function will integrate diagnostic and prognostic capabilities to estimate the current health state of the system and to predict its remaining useful life.

References

[1] J. L. Peterson, Petri nets, ACM Computing Surveys (CSUR) 9 (3) (1977) 223–252.

[2] A. Napoleone, M. Macchi, A. Pozzetti, A review on the characteristics of cyber-physical systems for the future smart factories, Journal of Manufacturing Systems 54 (2020) 305 – 335. doi:https://doi.org/10.1016/j.jmsy.2020.01.007.

[3] Q. Gaudel, E. Chanthery, P. Ribot, Health Monitoring of Hybrid Systems Using Hybrid Particle Petri Nets, in: Annual Conference of the Prognostics and Health Management Society 2014, Proceedings of The Annual Conference of the Prognostics and Health Management Society 2014, 2014, p. 51.

[4] H. Alla, R. David, Continuous and hybrid petri nets, Journal of Circuits, Systems, and Computers 8 (01) (1998) 159–188.

[5] F. Bouchhima, G. Nicolescu, E. M. Aboulhamid, M. Abid, Generic discrete–continuous simulation model for accurate validation in heterogeneous systems design, Microelectronics journal 38 (6-7) (2007) 805–815.

[6] M. B. Ayed, F. Bouchhima, M. Abid, Codis+: Co-simulation environment for heterogeneous systems, Journal of Control Engineering and Applied Informatics 20 (1) (2018) 98–107.

[7] P. Ribot, E. Bensana, A generic adaptive prognostic function for heterogeneous multi-component systems: application to helicopters, in: European Safety & Reliability Conference, Troyes, France, 2011.

[8] C. G. Cassandras, S. Lafortune, Introduction to discrete event systems, Springer Science & Business Media, 2009.

[9] T. A. Henzinger, The theory of hybrid automata, in: Verification of digital and hybrid systems, Springer, 2000, pp. 265–292.

[10] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, S. Yovine, The algorithmic analysis of hybrid systems, Theoretical computer science 138 (1) (1995) 3–34.

[11] M. Zhou, F. DiCesare, A. A. Desrochers, A hybrid methodology for synthesis of petri net models for manufacturing systems, IEEE transactions on robotics and automation 8 (3) (1992) 350–361.

[12] F. DiCesare, G. Harhalakis, J.-M. Proth, M. Silva, F. Vernadat, Practice of Petri nets in manufacturing, Springer, 1993.

[13] F. Balduzzi, A. Giua, C. Seatzu, Modelling and simulation of manufacturing systems with first-order hybrid petri nets, International Journal of Production Research 39 (2) (2001) 255–282.

[14] C. Valentin-Roubinet, Hybrid systems modelling: mixed petri nets, in: IEEE Conference CSCC, Vol. 99, 1999.

[15] L. A. Cortés, P. Eles, Z. Peng, A petri net based model for heterogeneous embedded systems, in: Proc. NORCHIP Conference, 1999, pp. 248–255.

[16] Q. Gaudel, E. Chanthery, P. Ribot, M. J. Daigle, Diagnosis of hybrid systems using Hybrid Particle Petri nets: theory and application on a planetary rover, in: M. Sayed-Mouchaweh (Ed.), Fault Diagnosis of Hybrid Dynamic and Complex Systems, Springer Verlag, 2018, pp. 209–241.

[17] Q. Gaudel, E. Chanthery, P. Ribot, Hybrid Particle Petri Nets for Systems Health Monitoring under Uncertainty, International Journal of Prognostics and Health Management 6 (Jun. 2015).

URL https://hal.archives-ouvertes.fr/hal-01229083