7

# Use of Petri Nets for Modeling an Agent-Based Interactive System: Basic Principles and Case Study

Houcine Ezzedine and Christophe Kolski
*LAMIH, University of Valenciennes*
*France*

## 1. Introduction

Several architecture models of interactive systems have been put forward by researchers over the past twenty years. Two main types of architecture can be distinguished: architectures with functional components: Langage (Foley & Vandam, 1982), Seeheim (Pfaff, 1985) and ARCH (Bass et al., 1991) and architectures with structural components: PAC (Coutaz, 1987), PAC-Amodeus (Nigay, et al., 1997), MVC (Goldberg, 1983), AMF (Tarpin-Bernard & David, 1999), H⁴ (Guittet, 1995),…. The approaches currently used in interactive system design adopt a modular structuring aimed towards a better apprehension of the reactivity, flexibility, maintainability and re-use. Agent-based approaches are promising in this way.

In the agent-based architecture proposed, we suggest using a division into three functional components: the *application agents* which handle the field concepts and cannot be directly accessed by the user; the *interactive agents* (or *interface agents*, or presentation agents) which , unlike the application agents, are in direct contact with the user (they can be seen by the user); the *dialogue control agents* which are also called mixed agents (Ezzedine & Trabelsi, 2005). Each agent therefore plays a role within its group; this role can be expressed in the form of the services it offers in the interactive system.

We use so-called agent Petri Nets (PN) to model a priori the services offered by each interface agent: a service is defined as being a quadruplet S ={E, C, R, P}, with E: the event which triggers the service, C: the conditions to be met in order to perform this service, R: the resources necessary for the service to be performed, P: the property of this service, which can be either an operation concerning the agent alone (with or without a change of state for the interactive agents), or a call for the service of another agent. The succession of various calls for services gives rise to the succession of page-screens in the human-computer interface.

This chapter begins with a state of the art about the use of Petri nets in Human-Machine Interaction. Then we explain the problem relating to agent-based architectures of interactive systems, and we propose a solution for the modeling of the interface agents of such architectures. Lastly, we illustrate our approach by a case study.



Source: Petri Net,Theory and Applications, Book edited by: Vedran Kordic, ISBN 978-3-902613-12-7, pp. 534, February 2008, I-Tech Education and Publishing, Vienna, Austria

## 2. Use of PN in the field of human-computer interaction

Petri nets allow the modeling and visualization of behaviors comprising parallelism, synchronization and resource sharing. Their power lies in their formal aspect; they allow the modeling of discrete systems evolving in parallel, which represents a great contribution for the modeling of various facets relating to human-machine interactions, particularly in complex systems. Various complementary approaches are found on this subject in the literature, based on the use of various types of PN in the field of human-machine interaction, a subject which has been developed ever since the end of the Eighties (Williem & Biljon, 1988). They are given here in a list which is not intended to be exhaustive, but rather to be representative.

Thus, PN were used: (1) before design phases (with an aim of human-computer interaction specification), for task modeling (also called prescribed or theoretical task); it corresponds to the task, envisaged by the designer(s), to be carried out by the user and/or the machine (*a priori* modeling); (2) for the modeling of the human activities (*a posteriori* modeling); this modeling follows the phase of evaluation of the interactive system in a real or simulated situation with users. By a confrontation of the *a priori* and *a posteriori* models, and by an analysis of the differences between these two complementary sets of PN, it is possible to detect design errors, lacks of information, and so on, and to put forward proposals, especially concerning the improvement of human-computer interaction (Abed, 1990; Abed et al. 1992; Abed, 2001). Figure 1 shows an example of modeling with PN of the human-machine interaction planned for part of an interactive application relating to a post of transport network supervision (Ezzedine et al., 2003). The places in the PN represent the actions carried out by the user whereas the transitions represent the reactions from the user interface; the graphic components present on it (bottom left part on figure 1) rise directly from the elements described on the PN (right part of figure 1).

Very important basic research was undertaken by P. Palanque, R. Bastide and their colleagues on the use of the PN for the checking and validation of interactive systems (Palanque & Bastide, 1990; Palanque et al., 1995; Navarre et al., 2003; Winkler et al., 2006…). For instance, they proposed rule-based mechanisms for the automatic evaluation of PN-based models of interactive systems (Palanque et al., 1999).

In the works on the ICO (Interactive Cooperative Objects) (Palanque, 1992; Palanque 1997) and the TOOD method (Task Object Oriented Design) (Mahfoudhi et al., 1995; Tabary & Abed, 2002), Object Petri nets are used to model human tasks in a HCI context and to specify and then design object oriented interactive systems.

Ezzedine and Kolski (2005) present a method for the modeling of cognitive activity also using object Petri nets: the method includes the recognition of the various classes of situation (normal and abnormal) which human operators are likely to meet whilst performing their tasks; each of these classes is described according to the characteristics of the state of the system (Kaddouri et al., 1995).

From a description of normal and abnormal situations possible in process control applications, Moussa et al. (2002, 2006) use interpreted Petri Nets for human-machine dialogue specification.

Kontogianis (2003) chooses to use colored Petri nets for ergonomic task analysis and modeling with emphasis on adaptation to system changes. Gomes et al. (2001) propose an interesting approach based on reactive Petri nets (inherited from colored Petri nets) for human-machine interface modeling.

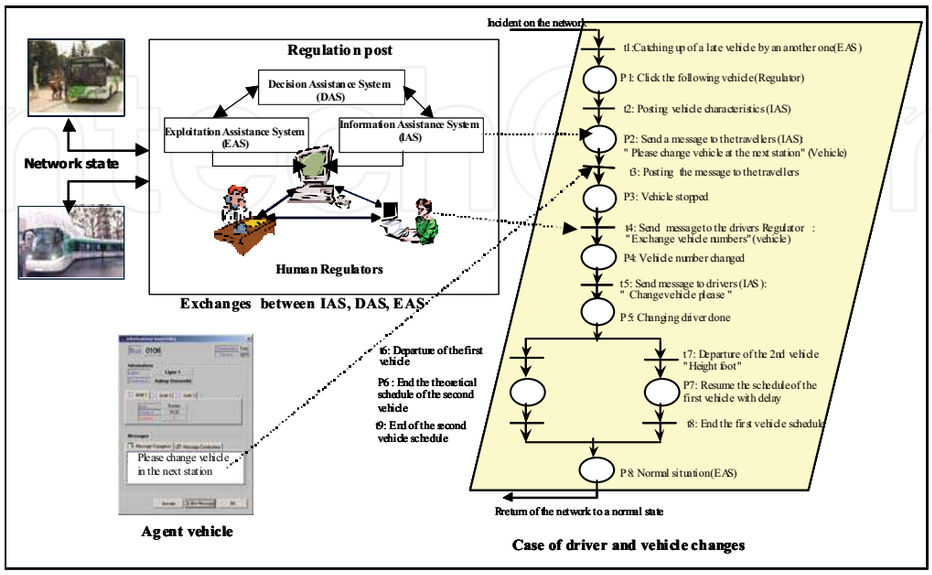



Fig. 1. Human-computer interaction modeling using PN

Bernonville et al. (2006) propose a method to facilitate the re-engineering of existing interactive software by proposing a common framework for Software Engineers and Human Factor specialists: their method explicitly combines Petri Nets and ergonomic criteria.
To our knowledge, none of these works is interested in modeling the agents which make up agent-based interactive systems, by establishing a direct link with the software architecture.

## 3. Problem of modeling related to agent-based architectures of interactive systems

The architecture of a computer system is a set of structures, each including: components, outside visible properties of these components and relations which the components maintain (Bass et al., 1991). We are only interested in interactive systems: in this context, the architecture models aim to provide a framework for the design and the realization of the complete system, emphasizing clearly the part with which the user interacts. Existing architectures break up the interactive system into modules and define specific roles for each module, contributing to the correct execution of the complete system. Two main types of architecture can be distinguished: architectures with functional components (Langage, Seeheim and Arch) and architectures with structural components (PAC, PAC-Amodeus, MVC…). It should also be noted that certain classifications emphasize three categories (centralized models, distributed or agent-based model, hybrid models).
The classic models of interactive systems distinguish three essential functions *(presentation, control* and *application)*. Some models, such as the Seeheim (Pfaff, 1985) and ARCH models, consider these three functions as being three distinct functional units. Other approaches using structural components, and in particular those said to be distributed or agent

approaches, suggest grouping the three functions together into one unit, the agent. The agents are then organised in a hierarchical manner according to principles of composition or communication: for example PAC (Coutaz, 1997) or its variants, or the MVC model (Model-View-Controller) of Smalltalk and its recent evolutions (Goldberg, 1984), AMF and its variants (Ouadou, 1994), H⁴ (Guittet, 1995)…

These architecture models preach the same principle based on a separation between the system (application) and the human-computer interface (HCI). Thus, an architecture must separate the application and the HCI, define a distribution of the services of the interface and define a protocol of information exchange. One of the interests in separating the interface and the application is to make it easier to modify the interface without changing the application (Coutaz, 1997).

The architecture adopted can be considered as being intermediate as it borrows elements for its principles from both types of model given above at the same time whilst being functional and structural (Ezzedine et al., 2001). In (Ezzedine et al., 2003) and (Ezzedine et al., 2005), we proposed an architecture ensuring separation in three functional components, which we called respectively: *interface with the application* (connected to the application), *dialogue controller* and *presentation* (this component is directly linked to the user), figure 2.

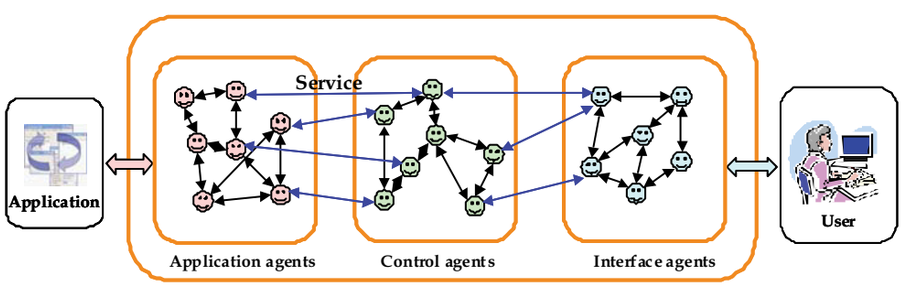



Fig. 2. Agent-based Architecture of interactive system

These three components group together agents:

- the *application agents* which handle the field concepts and cannot be directly accessed by the user. One of their roles is to ensure the correct functioning of the application and the real time dispatch of the information necessary for the other agents to perform their task;
- the *control* (or *dialogue controller) agents* which are also called mixed agents; these provide services for both the application and the user. They are intended to guarantee coherency in the exchanges emanating from the application towards the user, and vice versa;
- the *interactive agents* (or *interface agents*, or *presentation* agents); unlike the application agents, these are in direct contact with the user (they can be seen by the user). These agents co-ordinate between themselves in order to intercept the user commands or requests, and to form a presentation which allows the user to gain an overall understanding of the current state of the application. In this way, a window may be considered as being an interactive agent in its own right; its specification describes its presentation and the services it has to perform.

In the following part, we explain how the agents which make up such agent-based interactive systems are modeled. We focus on the interface agents.

## 3. Principles of modeling by PN of interactive systems with agent-based architecture

Initially, we will point out the basic principles of the parametrized Petri nets which inspired our modeling approach. Then we will explain how the services of the various interface agents of interface can be modeled.

### 3.1 Parametrized Petri nets

Parametrized Petri nets are classified amongst the high level PN. They allow the modelling of dynamic systems (Gracanin et al., 1994) according to the following principle: it is possible to link, in one parameter of these PN, a coherent set of objects or of values taken by the objects. This makes it possible to handle sets of objects, thus reducing the complexity of the representation.

A parameterized Petri net is a n-tuple: (C, D, Pp, T, I, O) where:

- C: the set of the values of the parameters; a parameter is a class of objects or values taken by the objects.
- D: the set of all the vectors built starting from the values. In such a network, in fact the vectors are produced or consumed (tokens).
- Pp: the set of the places of the network, called parameterization descriptor.
- T: the set of all the transitions from vectors representing all the actions which can be carried out by the system.
- I: the set of consumed tokens (input)
- O: the set of produced tokens (output)

### 3.2 Modeling of interface agents according to a set of services

The concept of an agent's service such as it was introduced by (Maoudji et al., 2001) considers the service as an action which involves the agent itself or other agents. For a service to be started, the agent needs:

- the appearance of the trigger event
- the checking of the condition in connection with the service,
- the checking of some resources which may be necessary for the establishment of the service.

Formally the service is defined by a quadruplet (Moldt et Wienberg, 1997), Figure 2, Service={E, C, R, P} where:

- E: the event which triggers the service, for example a user action or a display of an alarm or anomaly message coming from an application.
- C: a condition to check in order to carry out the service, such as the exceeding of a threshold as regards the value of one of the application's variables, or the presence of any risk for the application
- R: the resources necessary for the service to be performed.
- P: the property of the service which can be internal (the resulting action affects the agent itself) or external (the resulting action relates to other agents); the latter will be interpreted as an event.

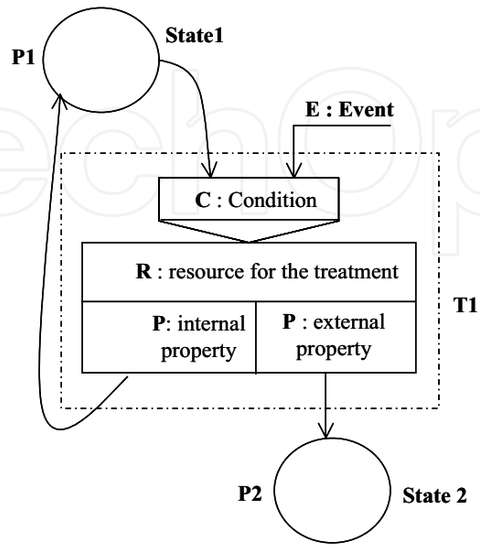



Fig. 2. Modeling of an agent's service (Maoudji et al., 2001)

### 3.3 Concept of PN used to model the agents of the interactive system (Agent PN)

Taking the concept of service of an agent introduced by (Maoudji et al., 2001) as a starting point, we propose the modeling of the behaviour of an agent by the modeling of the set of its services, Figure 3. The services of each agent can be represented by external actions on

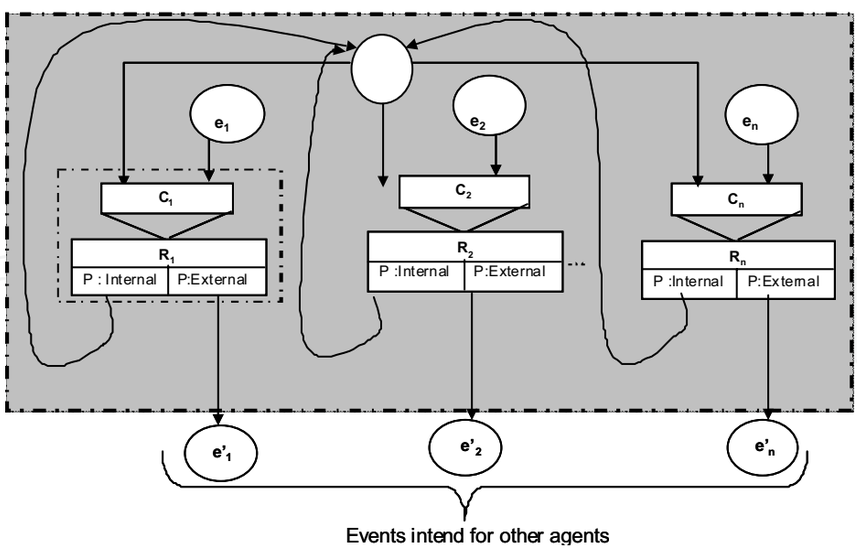



Fig. 3. Modeling of agent's services with agent PN (Ezzedine et al., 2001)

other agents; they are then interpreted as trigger events for other agents such as $e'_1$, $e'_2$ and up to $e'_n$, or by actions on the agents themselves such as looping on the place which is at the entry of their condition C. The service of an agent can be achieved by the execution of an internal or external action only if the condition C is true: i.e. if the place upstream of the condition C is marked (and thus has at least one token), and if event $e_i$ is triggered and the resource necessary to achieve the action is available.
We distinguish three types of place in the PN agent which are:

1. An *agent* place*:* which always contains the current view of the agent (result of a service).
2. An *event* place*:* which contains an event which triggers the service.
3. An intermediate place*:* which contains the events bound for another agent.

If we model an interface with n agents and m services for each agent, we will obtain a less readable PN agent, because of this and in order to mitigate the problem of the complexity of the PN, we increase the model by the capacity to abstract the set of services of an agent in one single form, Figure 4.

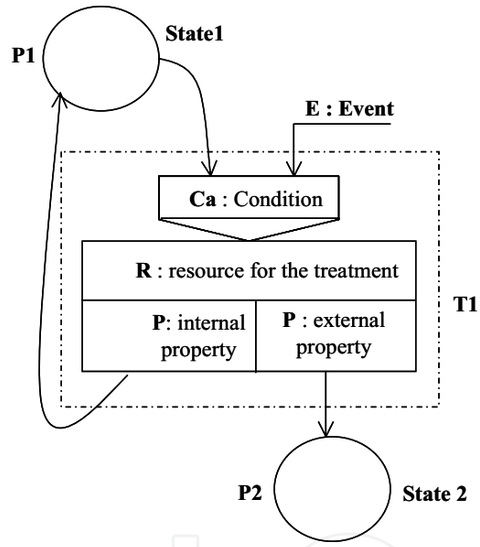



Fig. 4. Abstraction of the services of an agent.

Formally, an agent (set of n services) is defined by a quadruplet as follows:
Agent={E, C, R, P} where:

- E= {$e_1$, $e_2$,..., $e_i$,…, $e_n$}, set of events which trigger agent services (**n** events for **n** services). An event is characterized by two fields: its identifier and the moment of the appearance. For example, an alarm may be triggered when the threshold of a variable to be supervised in an industrial process is exceeded (temperature too high in a chemical process, late bus in a transport system…).
- C= {$c_1$, $c_2$,..., $c_i$, ..., $c_n$}, set of conditions necessary for the establishment of the services. Each service must check a condition so that it can trigger itself off. For example: an event may appear following the triggering of an alarm. A condition can be made of several elementary conditions. It at least includes the elementary condition: presence of

the service trigger event.

- R= {$r_1$, $r_2$, ..., $r_i$, ..., $r_p$}, set of resources which may be necessary for the establishment of the services (if the services have visible actions). A resource can be made of several elementary resources. For example, it may be necessary to have a display screen, a printer or another peripheral device (possibly a sound device) in order to inform the user of the type of alarm message. The information contained in the fields of each resource relates particularly to its size, color(s) and all kinds of information contributing to the characteristics of the human-computer interface.
- P: Properties of the services. Each service results from the execution:
    1. either of a non visible action $ac_N$ (an action which affects the agent itself): for example the service which deals with the displaying of a value of a process variable where the display service alone is involved in updating the value of the variable in question; we then speak about an internal property of the service.
    2. or of a visible action $ac_V$ (an action which relates to another agent); for example actions of the human operator which consist in writing a message with a goal to send it to another person, or the change of a value of a process parameter; we then speak about an external property of the service.

A service can have two properties at the same time: in other words, following the appearance of an event, an agent can act on itself (such as the service which is in charge of the reactualization of a value of a variable) and on another agent at the same time (for instance in the case of exceeding a tolerated threshold of the value of the variable); in this case it will be necessary to generate and display an alarm message, which is the subject of another service.

An external action can relate to several agents (for instance, sending the same message to several agents); it will be regarded as a vector which includes the number of the action and the list of the identifiers of the agents concerned. If several screens of the human-machine interface are concerned with the same message, the agents of message acquisition, treatment and display each perform a different service.

### 3.4 Mathematical model

A mathematical formulation of the PN agent can be put forward, using (Moldt et Wienberg, 1997) as a starting point:

- An agent $a_j$ is a set $S_j$ of n services, $S_j$= {$s_1$, $s_2$, ..., $s_i$, ..., $s_n$}.
- For each service is associated an event $e_i$ belonging to the set E of the events, E= {$e_1$, $e_2$, ..., $e_i$, …, $e_n$}.
- To each service, a condition $c_i$ is associated, belonging to the set C of the conditions, C= {$c_1$, $c_2$, ..., $c_i$, ..., $c_n$}; a condition can be composed of several elementary conditions.
- If the service comprises a visible action, a resource $r_i$ belonging to the set R of the resources is necessary, R= {$r_1$, $r_2$, ..., $r_i$, ..., $r_p$}.
- the set of the actions of the agent is composed of two subsets; $AC_V$ is the set of visible actions and $AC_N$ corresponds to the set of non visible actions.
  $AC_V$ = {$ac_{V1}$, $ac_{V2}$,…, $ac_{VP}$}, $AC_N$= {$ac_{N1}$, $ac_{N2}$,…, $ac_{Nq}$} with p ≤ n and q ≤ n.

We have:

Card (E) = n, Card (C) = n, Card (R) = p, Card ($AC_V$) = p, Card ($AC_N$) = q.

A service results from a minimum of one action (which can be visible in the form of a

display, input coming from a keyboard, a click on the mouse, speech acquisition, …, or non visible, for instance when there are interactions between internal agents, such as the control agents) and a maximum of two actions (visible and non visible).
The number of all the services of the agent is defined by:
Nb_Services = Card (E) = n.
The number of all the actions of the agent is defined by:
Nb_Actions = Card ($AC_V$) + Card ($AC_N$)= p + q.
It is included in the margin: n ≤ Nb_Actions ≤ 2×n.
We define the result of the service with a couple of actions ($AC_{Vk}$, $AC_{Nt}$), where the indices k and t take the zero value if the service does not contain a visible action or a non visible action. The number of the couples of actions is defined by:
Nb_Couples = Nb_Actions – Card (E)
We will order for example the couples of actions so that the couples which contain visible and non visible actions are the first, then the couples which contain only visible actions are in second place, while those which contain only non visible actions are in last place.
Then the services are defined by the following parameterized function (the parameter j being the identifier of the agent):

$S_j$: $E_j \times C_j \times R_j \longrightarrow AC_V \times AC_N$.
$(e_{i,j}; c_{i,j}; r_{k,j}) \longrightarrow (ac_{vk,j}; ac_{Nt,j})$. j = 1,…, number of agents
i = 1,...,n.
0 ≤ k ≤ p such as if i ≤ p then k=i.
else k=0.
0 ≤ t ≤ q such as if i ≤ Nb_Couples then t=i
else
if i > p then t = i – p + Nb_Couples.
else t=0.

j: identifier of the agent
i: number of the event
$AC_V$: to express that it is a Visible action.
$AC_N$: to express that it is a Not Visible action.
k,t: number of the action (if the service does not result by a visible action, then k=0; if the service does not result by a non visible action, then t=0).
Then the specification of an agent $a_j$ consists in the definition of the sets $E_i$ , $C_i$, $R_k$ and the mathematical specification of a task of an agent consists of the definition of a sequence of triplets ($e_i$, $c_i$, $r_k$) associated with their couples of actions ($ac_{vk,j}$; $ac_{nt,j}$), by chronological order of appearance of the events.

## 4. Case study

The case study relates to an application of supervision of a complex process. It is supposed that a human operator (or group of human operators) is located in a control room: he or she must supervise it by the intermediary of an interactive system. The human operator intervenes in various normal situations as well as in abnormal ones. In the abnormal situations, it must intervene following the arrival of disturbances, or even anticipate them (Stanton, 1994; Moray, 1997). It is supposed that the application relates to the supervision of an urban transport network (such as tramways, buses...).
The architecture suggested for the interactive system consists of three modules: *Application,*

*Control* and *Presentation* (figure 5). Each module is composed of agents interacting between themselves and/or with the agents of other modules. Let us recall that an agent (set of n services) is defined by a quadruplet as follows:
Agent={E, C, R, P}.
It is the presentation module (composed of interactive agents) which we will model according to the following scenario of disturbance:

1. An event $e_1$ comes from a *dialogue controller* agent, following a disturbance in the application.
2. There is $Ac_8$ action of agent $a_1$ (a request for service of the agent $a_1$ to the agent $a_2$), following an analysis of the condition $c_1$ for the execution of the action $a_1$; this $Ac_8$ action in its turn becomes the event release $e_5$ for the agent $a_2$ (for example for the display of a message).
3. Agent $a_2$ carries out the $Ac_3$ action which consists in presenting the message to the user if the resource "Window" is available.
4. The event $e_3$ occurs; it represents the reaction of the user, following the display of the message. It consists of an input of a text with the keyboard or an acknowledgement of the message with the keyboard or the mouse.
5. The agent $a_3$ carries out the $Ac_5$ action following an event trigger $e_3$ which can be a return of an acknowledgement message or the validation of an action provided that $c_3$ is checked and that the display resource is available.

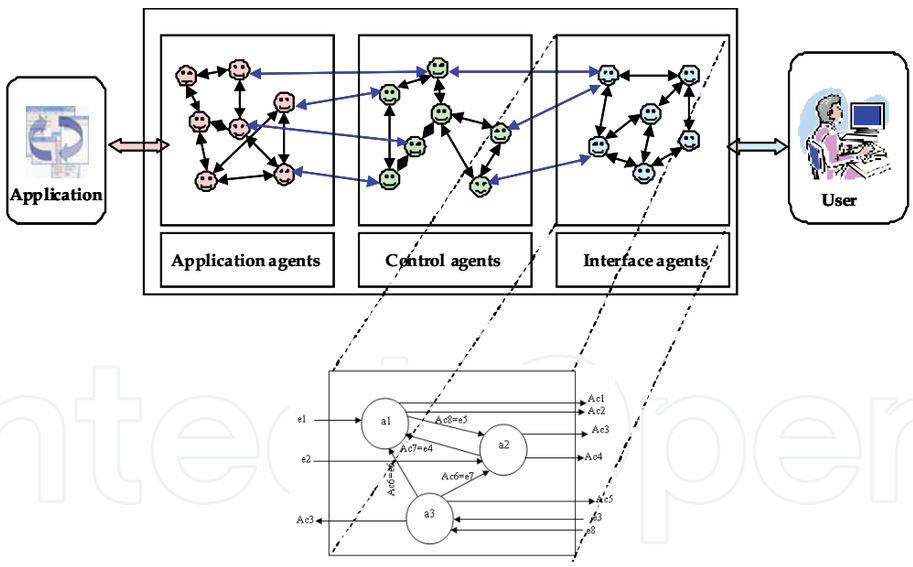



Fig. 5. Global view of the agent-based architecture, with arrival of a disturbance (Rehim, 2005).

While reformulating more formally, using (Moldt & Wienberg, 1997) as a source of inspiration, the presentation module can be defined by the following elements (figure 6):
A= {a1, a2, a3}: set of the presentation agents (in the example of figure 6, all three of them are

visible to the user).

E= {$e_{1,1;}$ $e_{1,2;}$ $e_{1,3,;}$ $e_{2,3,;}$ $e_{2,1;}$ $e_{2,2;}$ $e_{3,1;}$ $e_{3,2}$}: set of events coming (1) from the agents of the *interface with the application* module (via the *dialogue controller* agents which send $e_{1,1}$ and $e_{1,2}$ events to the presentation module), such as alerts, dysfunctional events, incidents, and so on (2) from user actions such as: commands or confirmations ($e_{1,3;}$ $e_{2,3)}$, (3) from non visible actions which become events for other agents ($e_{2,1;}$ $e_{2,2;}$ $e_{3,1;}$ $e_{3,2}$).

AC= { $ac_{v1,1;}$ $ac_{v2,1;}$ $ac_{v1,2;}$ $ac_{v2,2;}$ $ac_{N1,1;}$ $ac_{N1,2;}$ $ac_{N1,3;}$ $ac_{N2,3;}$ $ac_{N3,3}$}: set of agent actions; these actions can be visible ($ac_{v1,1;}$ $ac_{v2,1;}$ $ac_{v1,2;}$ $ac_{v2,2;}$ $ac_{v1,3}$), such as the display of information on a screen, or non visible ($ac_{N1,1;}$ $ac_{N1,2;}$ $ac_{N1,3;}$ $ac_{N2,3;}$ $ac_{N3,3}$), such as a request for service to other agents.

R= {$r_{1,1;}$ $r_{2,1;}$ $r_{1,2;}$ $r_{2,2;}$ $r_{1,3}$}: set of resources (such as keyboard, mouse, windows…) necessary for the visible actions of the agents, in order to carry out their actions ($r_{1,1}$ is related to $ac_{v1,1}$, $r_{2,1}$ is related to $ac_{v2,1}$, and so on).

C= {$c_{v1,1;}$ $c_{v2,1;}$ $c_{v1,2;}$ $c_{v2,2;}$ $c_{N1,1;}$ $c_{N1,2;}$ $c_{N1,3;}$ $c_{N2,3;}$ $c_{N3,3}$}: set of conditions related to the execution of the visible or non visible actions ($c_{v1,1}$ is related to $ac_{v1,1}$, $c_{v2,1}$ is related to $ac_{v2,1}$, and so on).

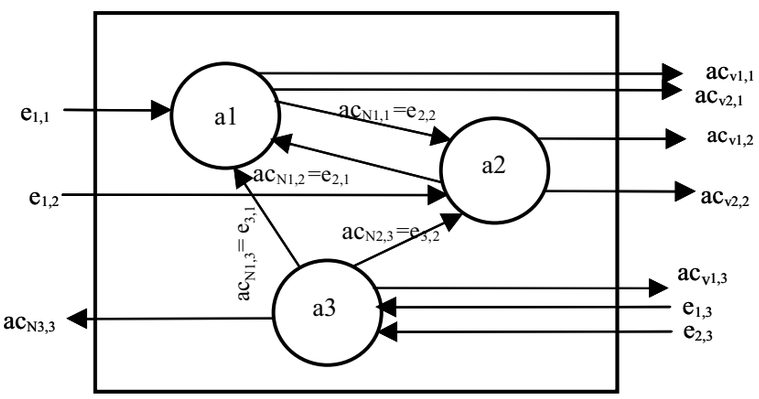



Fig. 6. Reformulated example related to the disturbance (*presentation* module)

For a better representation of the example visible in figure 6, one associates an agent identifier i,j with all the information present in the example; i is the event, action or necessary resource number; j is the agent number (from 1 to N). The possible couples constituted with visible ($ac_{vi,j}$) and non visible ($ac_{Ni,j}$) actions are obtained following the appearance of an $e_{i,j}$ event, under the conditions that $c_{i,j}$ is true and the $r_{i,j}$ resource is available. The modeling of services of each of the three agents forming the presentation module can be expressed as follows, with $r_{0,j}$: resource not necessary for the execution of the non visible action ($ac_{N0,j}$) by an agent j, and with $ac_{v0,j}$ and $a_{N0,j}$: visible and non visible actions which are not performed by the agent j.

**Agent1:**

E1 = {$e_{1,1;}$ $e_{2,1;}$ $e_{3,1}$}

$Ac_{v1}$ = {$ac_{v1,1;}$ $ac_{v2,1}$}, $Ac_{N1}$ = {$ac_{N1,1}$},

R = {$r_{1,1;}$ $r_{2,1}$},

C = {$c_{v1,1;}$ $c_{v2,1;}$ $c_{N1,1}$}.

Card ($E_1$) = 3, Card ($Ac_{v1}$) = 2, Card ($Ac_{N1}$) = 1.
Nb_Actions = Card ($Ac_{v1}$) + Card ($Ac_{N1}$) = 2+1=3.
Nb_Couples = Nb_Actions – Nb_Services = 3 – 3 = 0.

S1: $E_1 \times C_1 \times R_1 \longrightarrow Ac_{v1} \times Ac_{N1}$
($e_{1,1}$; $c_{v1,1}$; $r_{1,1}$) $\longrightarrow$ ($ac_{v1,1}$; $ac_{N1,1}$)
($e_{2,1}$; $c_{v2,1}$; $r_{2,1}$) $\longrightarrow$ ($ac_{v2,1}$; $ac_{N1,1}$)
($e_{3,1}$; $c_{N1,1}$; $r_{0,1}$) $\longrightarrow$ ($ac_{v0,1}$; $ac_{N1,1}$)

**Agent2:**

$E_2 = \{e_{1,2}; e_{2,2;} e_{3,2}\}$,
$Ac_{v2} = \{ac_{v1,2}; ac_{v2,2}\}$, $Ac_{N2} = \{ac_{N1,2}\}$,
$R_2 = \{r_{1,2}; r_{2,2}\}$,
$C_2 = \{c_{v1,2}; c_{v2,2}; c_{N1,2}\}$.

Card ($E_2$) = 3, Card ($Ac_{v2}$) = 2, Card ($Ac_{N2}$) = 1.
Nb_Actions = Card ($Ac_{v2}$) + Card ($Ac_{N2}$) = 2+1=3.
Nb_Couples = Nb_Actions – Nb_Services = 3 – 3 = 0.

$S_2$: $E_2 \times C_2 \times R_2 \longrightarrow Ac_{v2} \times Ac_{N2}$
($e_{1,2}$; $c_{v1,2}$; $r_{1,2}$) $\longrightarrow$ ($ac_{v1,2}$; $ac_{N1,2}$)
($e_{2,2}$; $c_{v2,2}$; $r_{2,2}$) $\longrightarrow$ ($ac_{v2,2}$; $ac_{N1,2}$)
($e_{3,2}$; $c_{N1,2}$; $r_{0,2}$) $\longrightarrow$ ($ac_{v0,2}$; $ac_{N1,2}$)

**Agent3:**

$E_3 = \{e_{1,3}; e_{2,3}\}$,
$Ac_{v3} = \{ac_{v1,3}\}$, $Ac_{N3} = \{ac_{N1,3}; ac_{N2,3;} a_{N3,3}\}$,
$R_3 = \{r_{1,3}\}$,
$C_3 = \{c_{v1,3}; c_{N1,3}; c_{N2,3;} c_{N3,3}\}$.

Card ($E_3$) = 2, Card ($Ac_{v3}$) = 1, Card ($Ac_{N3}$) = 3.
Nb_Actions = Card ($Ac_{v3}$) + Card ($Ac_{N3}$) = 1+3=4.
Nb_Couples = Nb_Actions – Nb_Services = 4 – 4 = 0.

S3: $E_3 \times C_3 \times R_3 \longrightarrow Ac_{v3} \times Ac_{N3}$
($e_{1,3}$; $c_{v1,3}$; $r_{1,3}$) $\longrightarrow$ ($ac_{v1,3}$; $ac_{N1,3}$)
($e_{2,3}$; $c_{N2,3}$; $r_{0,3}$) $\longrightarrow$ ($ac_{v0,3}$; $ac_{N2,3}$)
($e_{3,3}$; $c_{N3,3}$; $r_{0,3}$) $\longrightarrow$ ($ac_{v0,3}$; $ac_{N3,3}$)

From the mathematical modeling above, which made it possible to formulate the interactions of the agents in figure 6, each agent is modeled with the determination of its inputs and outputs in terms of condition, action, event and resource necessary to achieve the service. In figure 7, we present a model of interaction between the agents of the *presentation* module of the human-machine interface.

Figure 8 shows an example of release of the service $s_{7,1}$ "Change_Delay_Threshold_Vehicle" of the agent called *Traffic_State* belonging to the presentation module of the human-machine interface. Following the appearance of the event $e_{7,1}$ "Delay_Threshold_Vehicle" the interface agent *Vehicle* checks the corresponding condition $c_{7,1}$ "Appearance of the event and delay > 5". If the condition is verified, the service is started; the visible action $ac_{v7,1}$ is activated. The activation of this action exploits the resource $r_{7,1}$ "Dialog_Box" and reveals an alarm message bound for the human operator.

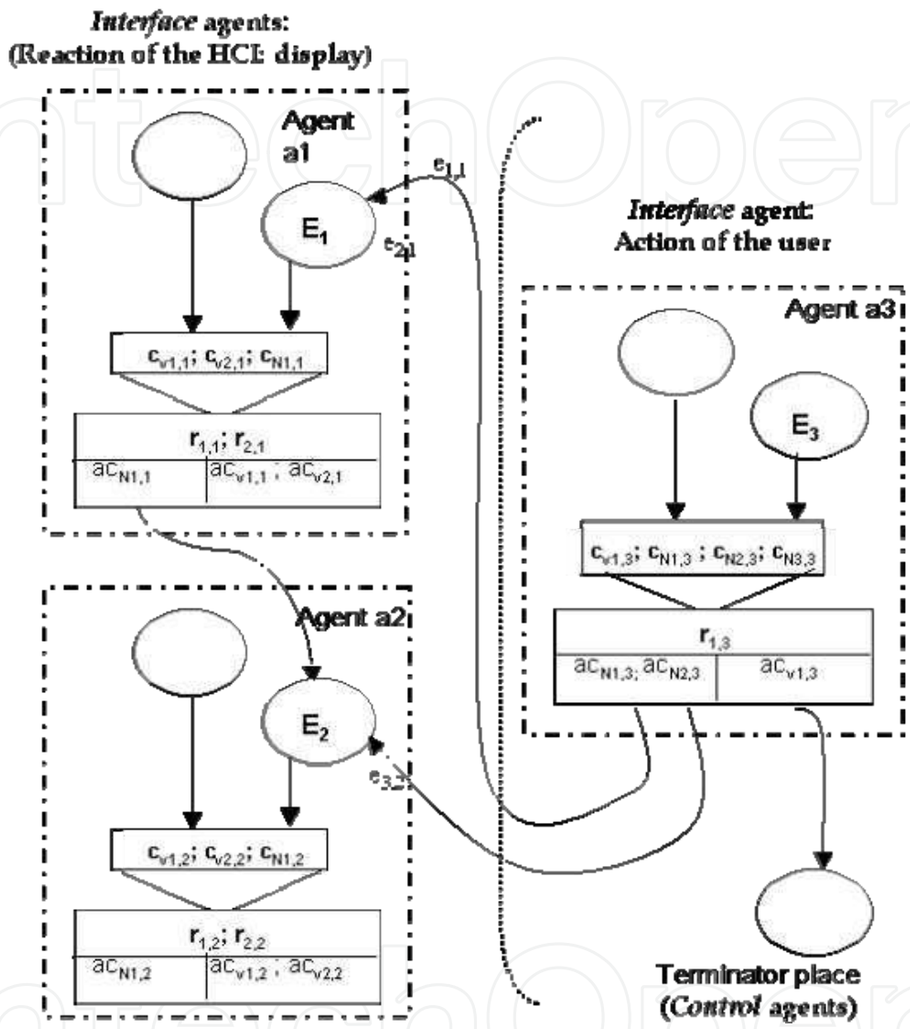



Fig. 7. Modeling of the example related to the disturbance (Rehim, 2005)

We notice, with the example presented above (figure 8), that there is no non visible action ($ac_{N0,1}$) of agent 1 in transition T1, i.e. that there is no interaction with other internal agents; but on the other hand, thanks to the visible action $ac_{v7,1}$, agent 1 acts on an external agent which is the window n°2 belonging to the presentation module of the interactive system.

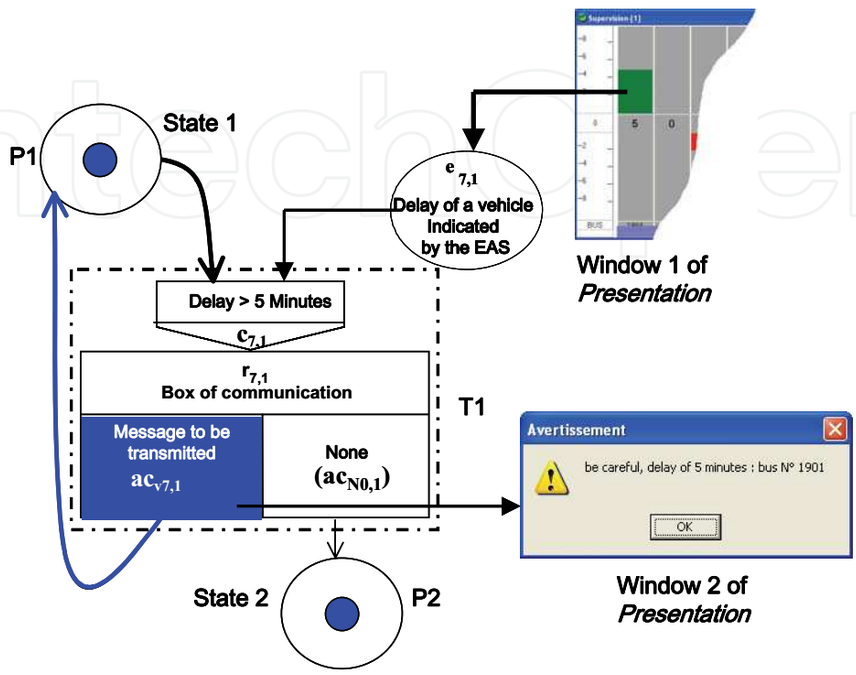



Fig. 8. Example of release of a service of the *Traffic_State* agent (Trabelsi, 2006)

## 4. Conclusion

Through their formal aspect and their capacity to model the dynamics of systems, Petri Nets have been bringing complementary and significant contributions in the human-computer interaction domain since the end of the Eighties. In this chapter, we have explained their utility for the modeling of agents which make up interactive systems with an architecture containing agents. A scenario made it possible to illustrate the approach proposed.

The PN used represent a promising tool for the modeling of such interactive systems. Their originality and their power reside in (1) their capacity to visualize the behavior of each agent (external or internal actions), as well as (2) their capacity of abstraction which makes it possible to keep the same information without any visual complexity, and (3) their formal aspect also enables them to be potentially efficient at the time of the evaluation and validation phase (which is not dealt with in this article).

There are several perspectives with this work. We are currently studying and developing an assistance tool for the evaluation of interactive systems. This tool makes it possible to connect to each interface agent, evaluation agents intended to analyze their behavior at the time of situations of use. The PN must make it possible to

reconstitute the human activities performed (Trabelsi, 2006; Tran et al., 2007). A second perspective relates to generalization with the agents of the two other modules: (1) interface with the application, (2) dialogue controller. Another perspective relates to the evaluation of the approach suggested in various application domains (for instance design and evaluation web sites).

## 5. Acknowledgements

The authors thank the FEDER, the GRRT and the Nord-Pas de Calais region for their financial support (SART, EUCUE and MIAOU projects). They also thank André Péninou, Hacène Maoudji, Aïssam Rehim and Abdelwaheb Trabelsi for their contribution to various modeling aspects described in this chapter.

## 6. References

Abed, M. (1990). Contribution à la modélisation de la tâche par des outils de spécification exploitant les mouvements oculaires : application à la conception et à l'évaluation des interfaces homme-machine. Thèse de doctorat, Université de Valenciennes et du Hainaut-Cambrésis, septembre.

Abed, M, Bernard, J.M, Angué, J.C (1992). Method for comparing task model and activity model. Proceedings 11th European annual conference Human Decision Making and Manual Control, Valenciennes, France.

Tabary, D., Abed, M. (1998). TOOD: an object-oriented methodology for describing user task in interface design and specification - An application to air traffic control. La Lettre de l'Intelligence Artificielle, 134, pp. 107-114.

Abed, M. (2001). Méthodes et modèles formels et semi-formels pour la conception et l'évaluation des systèmes homme-machine. Habilitation à diriger des recherches, Université de Valenciennes et du Hainaut-Cambrésis, 02 mai 2001.

Benaïssa, M.L., Ezzedine, H., Angué, J.C. (1993). An interface Specification Method for industrial processes. XII European annual conference on human decision making and manual control, Kassel, Germany, juin.

Bernonville, S., Leroy, N., Kolski, C., Beuscart-Zéphir, M. (2006). Explicit combination between Petri Nets and ergonomic criteria: basic principles of the ErgoPNets method. Proceedings of the 25th Edition of EAM'06, European Annual Conference on Human Decision-Making and Manual Control (September 27-29, 2006, Valenciennes, France), PUV.

Coutaz, J. (1987). PAC, an Object-Oriented Model for Dialog Design. In: Bullinger, Hans-Jorg, Shackel, Brian (ed.): INTERACT 87 - 2nd IFIP International Conference on Human-Computer Interaction. September 1-4, Stuttgart, Germany. p.431-436.

David, R. & Alla, H. (2004). Discrete, continuous, and hybrid Petri Nets. 1er ed. Springer Verlag, 2004, XXII, 524 p. Hardcover ISBN 3-540-22480-7

Ezzedine, H., Kolski, C. (2005). Modelling of cognitive activity during normal and abnormal situations using Object Petri Nets, application to a supervision system. Cognitive, Technology and Work, 7, pp. 167-181.

Ezzedine, H., Trabelsi, A., Kolski, C. (2006). Modelling of agent oriented interaction using Petri Nets, application to HMI design for transport system supervision. P. Borne, E.

Craye, N. Dangourmeau (Ed.), *CESA2003 IMACS Multiconference Computational Engineering in Systems Applications (Lille, France, July 9-11, 2003)*, Ecole Centrale Lille, Villeneuve D'Ascq, pp. 1-8, janvier, ISBN 2-9512309-5-8.

Ezzedine, H., Trabelsi, A., Kolski, C. (2006). Modelling of an interactive system with an agent-based architecture using Petri nets, application of the method to the supervision of a transport system. *Mathematics and Computers in Simulation*, 70, pp. 358-376.

Ezzedine, H., Trabelsi, A. (2005). From the design to the evaluation of an agent-based human-machine interface. Application to supervision for urban transport system. P. Borne, M. Benrejeb, N. Dangoumeau, L. Lorimier (Ed.), *IMACS World Congress "Scientific Computation, Applied Mathematics and Simulation" (July 11-15, Paris)*, ECL, pp. 717-725, juillet, ISBN 2-915913-02-1.

Ezzedine, H., Maoudji, H. & Péninou A. (2001). Towards agent oriented specification of Human-Machine Interface : Application to the transport systems. 8th IFAC Symposium on Analysis, Design, and Evaluation of Human-Machine Systems (IFAC-HMS 2001), pp. 421-426, Kassel, Germany, 18-20 September.

Foley, J.D & Van Dam, A. (1982). Fundamentals of Interactive Computer Graphics, Addison-Wesley (IBM Systems Programming Series), Reading, MA.

Goldberg, A. (1980). Smalltalk-80, the interactive programming environnement. Addision-Wesley.

Gomes, L., Barros, J.P., Coasta, A. (2001). Man-machine interface for real-time telecontrol based on Petri nets specification. In T. Bahill, F.Y. Wand (Eds.), IEEE SMC 2001 Conference Proceedings (e-Systems, e-Man and e-Cybernetics), Arizona, USA: IEEE Press, pp. 1565-1570.

Gracanin, D., Srinivasan, P. & Valavanis, K.P. (1994). Parametized Petri nets and their applications to planning and coordination in intelligent systems. IEEE Transactions on Systems, Man and Cybernetics, Vol. 24, pp. 1483-1497.

Guittet, L. (1995). Contribution à l'Ingénierie des IHM - Théorie des Interacteurs et Architecture H4 dans le système NODAOO, Thèse de l'Université de Poitiers, 1995.

Jensen, K. (1980). Coloured Petri Nets and Invarient Method. Daimi PB 104, Aarhus University.

Jensen, K. (1996). Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use. 2nd edition, vol n°2, Springer-Verlag.

Kontogiannis, T. (2003). A Petri Net-based approach for ergonomic task analysis and modeling with emphasis on adaptation to system changes. Safety Science, vol. 41 n°10, pp. 803-835.

Kaddouri, S.A., Ezzedine, H., Angué, J.C. (1995). Task modelling using object Petri Nets. In Anzaï Y., Ogawa K., Mori H. (Eds.), Symbiosis of Human and Artefact, HCI International'95: 6th International, Tokyo, Japan. (pp. 988-994). Amsterdam: Elsevier.

Mahfoudhi, A., Abed, M., Angué, J.C. (1995). An Object Oriented Methodology for Man-Machine systems analysis and design. Anzai Y., Ogawa K., Mori H. (Ed.), Symbiosis of Human and Artefact, HCI International'95: 6th International, Tokyo, Japan, Elsevier, Amsterdam, pp. 965-970, janvier.

Maoudji, H., Ezzedine, H. & Péninou A. (2001). Agents oriented specification of interactive systems. In M.J. Smith, G. Salvendy, D. Harris, R. Koubek (Ed.), Usability

evaluation and Interface design: Cognitive Engineering, Intelligent Agents and Virtual Reality, volume 1. (pp. 71-75). London : Lawrence Erlbaum Associate Publishers.

Maoudji, H., Ezzedine, H., Péninou, A. & Kolski, C. (2000). Amélioration de la qualité des correspondances dans les réseaux de transports urbains. Rapport d'étude à mi-parcours du projet coopératif GRRT, Juillet.

Moldt, M., Wienberg, F. (1997). Multi-Agent-Systems based on Coloured Petri Nets. In Proceedings of the 18th International Conference on Application and Theory of Petri Nets, Toulouse.

Moray, N. (1997). Human factors in process control. In Handbook of human factors and ergonomics, G. Salvendy (Ed.), John Wiley & Sons, INC., pp. 1944-1971.

Moussa, F., Riahi, M., Kolski, C., Moalla, M. (2002). Interpreted Petri Nets used for Human-Machine Dialogue Specification in Process Control : principles and application to the Ergo-Conceptor+ tool. Integrated Computer-Aided Engineering, 9, pp. 87-98.

Moussa, F., Kolski, C., Riahi, M. (2006). Analyse des dysfonctionnements des systèmes complexes en amont de la conception des IHM : apports, difficultés, et étude de cas. Revue d'Interaction Homme Machine (RIHM), 7, pp. 79-111.

Navarre, D., Palanque, P., Bastide, R. (2003). A Tool-Supported Design Framework for Safety Critical Interactive Systems, Interacting with computers, 15 (3), pp. 309-328.

Nigay, L., Coutaz, J. (1997). Software architecture modelling: Bridging Two Worlds using Ergonomics and Software Properties. In Formal Methods in Human-Computer Interaction, P. Palanque & F. Paterno (Eds.), Springer-Verlag: London Publ., ISBN 3-540-76158-6, 1997, pp. 49-73.

Ouadou, K. (1994). AMF : Un modèle d'architecture multi-agents multi-facettes pour Interfaces Homme-Machine et les outils associés. Thèse de l'Ecole Centrale de Lyon. 1994.

Palanque, P. & Bastide, R. (1995). Design, specification and of interactive systems. Springer Verlag 1995, ISBN 3-211-82739-0. 370 pages.

Palanque, P., Bastide, R. (1990). Petri nets with objects for specification, design and validation of user-driven interfaces. In proceedings of the third IFIP conference on Human-Computer Interaction, Interact'90, Cambridge,UK, 27-31 August.

Palanque, P. (1992). Modélisation par objets coopératifs interactifs d'interfaces homme-machines dirigées par l'utilisateur. Ph.D. Thesis, University of Toulouse 1, France.

Palanque, P., Bastide, R. (1997). Synergistic modelling of tasks, system and users using formal specification techniques. Interacting With Computers, 9 (12), pp. 129-153.

Palanque, P., Bastide, R., Sengès, V. (1995). Task model–system model: towards an unifying formalism. In proceedings of the HCI international (EHCI'95), Chapman & Hall, pp. 189-212.

Palanque, P., Farenc, C., Bastide, R. (1999). Embedding Ergonomic Rules As Generic Requirements in a formal Development Process of Interactive Software. In Proceeding Interact'99, Sasse A., Jonhson C. (Eds), IOS Press, pp. 408-416.

Pfaff, (1985). User interface management system. Springer-Verlag.

Rehim, A. (2005). Etude des outils exploitant des réseaux de Petri agent pour l'évaluation des systèmes interactifs. Mémoire de Master recherche.

Sibertin-Blanc, C. (1985). High-level Petri nets with Data Structure. Proceedings 6th EWPNA, June, Espoo, Finland, 1985.

Stanton, N. (1994). Human factors in alarm design. Taylor & Francis Ltd, London.

Tabary, D., Abed, M. (2002). A software Environment Task Object Oriented Design (ETOOD). Journal of Systems and Software, 60, pp.129-141.

Tarpin-Bernard, F. & David, B. (1999). AMF : un modèle d'architecture multi-agents multi-facettes Techniques et Sciences Informatiques. Hermès. Paris Vol. 18 No. 5. pp. 555-586. Mai . Thèse de doctorat, Université Joseph Fourier Grenoble 1, Mars.

Trabelsi, A. (2006). Contribution à l'évaluation des systèmes interactifs orientés agents. Application à un poste de supervision de transport urbain. Thèse de doctorat, Université de Valenciennes et du Hainaut-Cambrésis, 25 septembre.

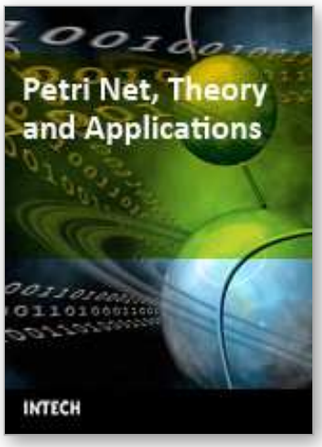
Trabelsi, A., Ezzedine, H. & Kolski, C. (2006). Un mouchard électronique orienté agent pour l'évaluation de systèmes interactifs de supervision. CIFA2006, Bordeaux, France, 30-31 Mai et 1 juin. Université de Valenciennes et du Hainaut-Cambrésis, juillet.

Tran, C.D., Ezzedine, H. & Kolski, C. (2007). Towards a generic and configurable model of an electronic informer to assist the evaluation of agent-based interactive systems.

ICEIS'2007, 9th International Conference on Entreprise Information Systems. 12-16 June, Funchal, Madeira- Portugal

Williem, R., Biljon, V. (1988). Extending Petri Nets for specifying Man-Machine dialogues. International Journal of Man-Machine Studies, vol. 28, pp. 437-45.

Winckler, M., Barboni, E., Palanque, P., Farenc., C. (2006). What Kind of Verification of Formal Navigation Modelling for Reliable and Usable Web Applications? 1st Int. Workshop on Automated Specification and Verification of Web Sites, Valencia, Spain. Electronic Notes Theoretical Computer Science, 157(2), pp. 207-211.




**Petri Net, Theory and Applications**

Edited by Vedran Kordic

ISBN 978-3-902613-12-7

Hard cover, 534 pages

**Publisher** I-Tech Education and Publishing

**Published online** 01, February, 2008

**Published in print edition** February, 2008

Although many other models of concurrent and distributed systems have been de- veloped since the introduction in 1964 Petri nets are still an essential model for concurrent systems with respect to both the theory and the applications. The main attraction of Petri nets is the way in which the basic aspects of concurrent systems are captured both conceptually and mathematically. The intuitively appealing graphical notation makes Petri nets the model of choice in many applications. The natural way in which Petri nets allow one to formally capture many of the basic notions and issues of concurrent systems has contributed greatly to the development of a rich theory of concurrent systems based on Petri nets. This book brings together reputable researchers from all over the world in order to provide a comprehensive coverage of advanced and modern topics not yet reflected by other books. The book consists of 23 chapters written by 53 authors from 12 different countries.



**InTech Europe**

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen