An Object Oriented Methodology for Man-Machine Systems Analysis and Design

A. Mahfoudhi, M. Abed, J-C. Angué

Laboratoire d'Automatique et de Mécanique Industrielles et Humaines
URA CNRS 1775   Université de Valenciennes et du Hainaut-Cambrژsis
B.P. n، 311 59304 VALENCIENNES CEDEX-FRANCE

Abstract

Despite the recent progress in the domain of Man-Machine Interface engineering, several problems concerning the incompatibily between the information presentation to the user and his cognitive representation are still present. This paper presents a new Task Object Oriented Description methodology (TOOD), specially adapted to the taking into account of the human factors for the specification of the Man-Machine Interfaces (MMI). Aconcrete application of this methodology was presented in the air traffic control context.

1. INTRODUCTION

The interactive systems development presents always ergonomic problems, essentially, bound to the man-machine communication. These problems can be generally related to the taking of human factors into consideration which are either implicit or explicit during the evaluation phase and the presentation of the information to the human operator (what, when and how). So, it's very important to have models and methods that permit, on the one hand to make more accessible the users' knowledge, and, mainly, more formal and more detailed their description. On the other hand to permit the specification of the communication interface (Man-Machine Interface : MMI).

There are two main approaches aiming appropriate solutions for these problems. The first one concerns the users' tasks analysis and description methods. These methods, such as MAD proposed by Scapin and Pierret-Golbreith (1989) or the SADT/Petri method proposed by Abed (1990), try to describe the task as a set of operations and, at the same time, they integrate implicit information about human factors. However, most of these methods present the same disadvantage; the managed treatments are not detailed, and the MMI specification are not completely integrated. The second approach, aiming to offset the disadvantages of task analysis and description methods, concerns the object oriented methods. An example of these methods is the OOA method (Coad and Yourdon, 1991) or the interactive cooperative objects formalism (Palanque, 1992). These methods have several advantages; they are easily usable, clear and favour the reusability and the modularity. However, they also, present their appropriate disadvantages. Account of their partial use (most often in the end, just before the programming), the object oriented methods take into account only the software engineering aspects in which case the considerations about users and human factors are completely absent.

It's only natural that our decision enjoys advantages of those two approaches. So, we propose a global Task Object Oriented Description (TOOD) methodology beginning from the task analysis and description to the MMI specification, based on the object oriented techniques and Object Petri Nets (OPN). Our goal was to test this methodology in a real complex system context (air traffic control). So, this paper provides a formal description of the air trafic controllers' task and the exploitation of this description for an ergonomic and complete specification for the futur air trafic control interface PHIDIAS (Position Harmonisant et Integrant les Dialogues Interactifs, Assistances et Secours).

2. DESCRIPTION OF THE AIR TRAFIC CONTROLLERS' TASK

The interface design to be implemented in a new system begins by analyzing the existing or similar system and the operator's current tasks. and raising both negative and positive interface aspects. These aspects are determined by taking into consideration the factors and relationships between the user, the tasks and the interface (Free and Brodbeck 1989). The relationship *accoplish tasks*describes how a user can carry out the tasks. The relationship *usability* describes haw difficult it is for the user to use the interface. The relationship *functionality* or *utility* describes how well the interface supports the tasks and allows the user to reach the task goals. So, it's very important to have models and methods that permit, on the one hand to make more accessible the users' knowledge, and, mainly, more formal and more detailed their description. On the other hand to permit the specification of the communication interface (Man-Machine Interface : MMI). For this, the TOOD methodology was found to be suittable not only for a simple task but also for a complexe tasks such as the air trafic controllers' task.

2. 1. TOOD : Task Object Oriented Description

TOOD is a new ergonomic methodologie which tries to relate the characteristics of the user's task with those of the Man-Machine Interface. It uses two complementary approaches : the first one for the users' tasks analyzing and describing, which advocates the hierarchical approach for the users' knowledge organization and the second approach for the MMI specification (cf. 3). For that TOOD uses The generic term زTask-Objectس which indicates each task of the hierarchy. Indeed, the task-object is defined as an independent entity and responsible for a treatment, whatever his complexity level, to reply to a goal to be carried out with given conditions. The task-object has a graphic form, inspired from the HOOD formalism (Michel, 1991) and Extended SADT method (Feller and Rucker, 1990), presented in Fig. 1.

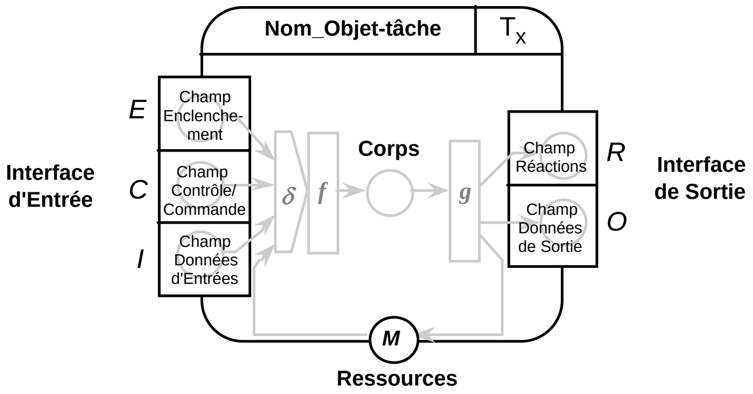


Figure 1. Structure d'un objet-t‰che

A task-object is also defined by a set of attributes, called "descriptors or attributsس, that defines the execution conditions and the effects of the task, as well as the actions or sub-tasks to be carried out : a *name* which identifies the tasks, a set of *events* (E) whiche defines the necessary events for the task release, a set of *control/command data* which defines the constraints to be respected, a set of *Input data* (I) which defines the list of data and information transformed by the task execution, a set of *Reactions* :(R) which defines the reports of the task execution, a set of *Output data* (O), a set of *Ressource* (R) which defines the necessary human and material entities for the execution of the task, and a *body* which identifies the actions or sub-tasks to be carried out.

2.2. Identification et specification of the air trafic controllers' taskd

The first stage of the TOOD methodology is the identification of the tasks of the futur system. By a hierarchical decomposition, it organizes the identified tasks-objects in a hierarchical tree form. It starts from the global task-object (the hierarchical tree's root) passing through the least abstract task-objets (the knots) and finishes with the terminal task-objects (the leaves). Let us consider the air traffic control, ژto configure the flight entryس can be regarded as a task-object. In order to reduce its abstraction, this task-object can be decomposed into three children task-objects : ژT_{111} : to take knowledge of a new flightس (terminal task-object), ژT_{112} : to take adecision about flightس and ژ T_{113} : to verify the position on radar screenس (terminal task-object). It is to be noticed that the events which activate the same task-object are shared out among the children task-objects. As shown by the figure 2, the task-object ژT11 : to configure the flight entryس can be activate by two events ژE11-1 : Arrival of a new flightس and ژE11-2 : Proposition of an entry level (EFL)س. Yet thoseevents activate two different children task-objects..Which means that both events ask for two different processing of the task-object ژT11 : to configure the flight entryس. Thus the event E11-1 activate the task-object ژT_{111} : to take knowledge of a new flightس to read information about the new flight while the event E11-2 suppose that the flieht information have been readen and activate the task-object ژT_{112} : to take adecision about flightس.

Once all future system's tasks are identified, the second stage of TOOD concerns the specification which defines all the execution conditions and the effects of each task-object. It consists in listing and identifying all the descriptors or attributs. The resulting document of this specification includes two kinds of description : a graphic description for a clean, legible and exploitable representation, (figure 2) and a textual one for a complete description of the descriptors of each task-object (figure 3).

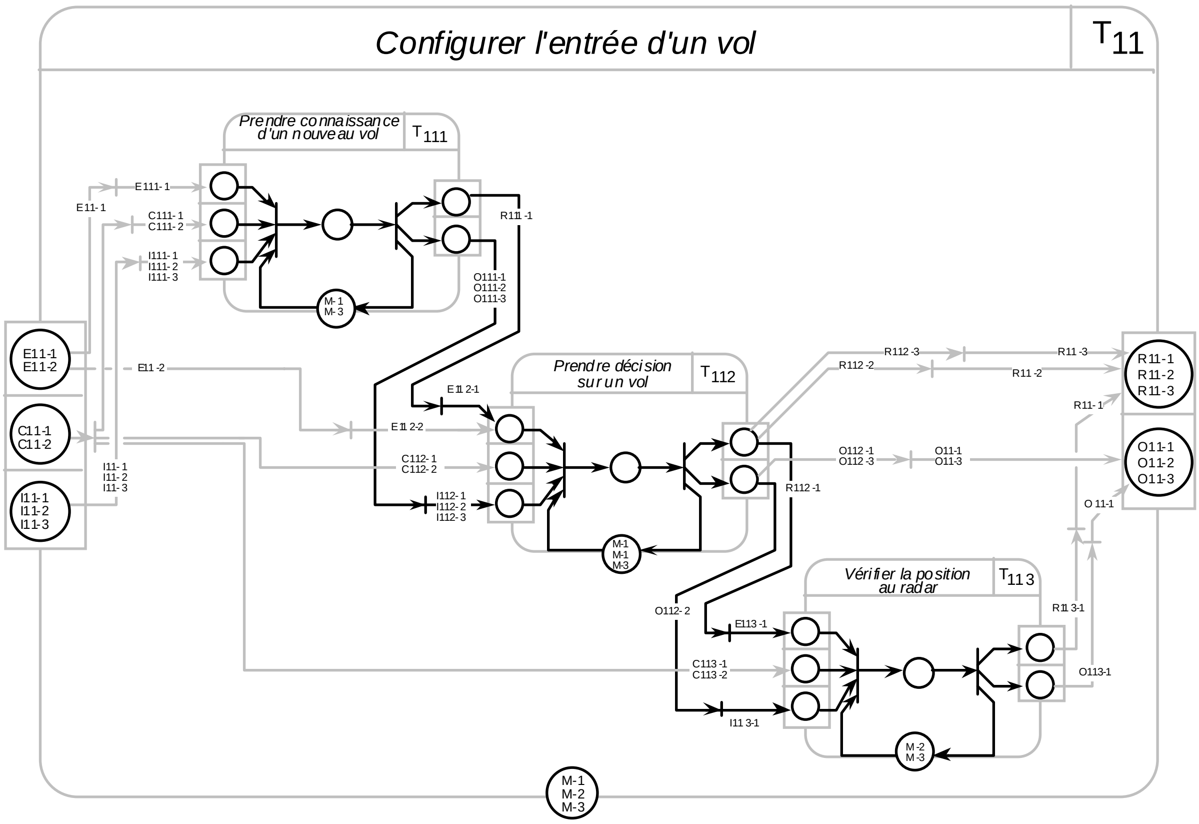


Figure 2. a graphic specification of the task-object ژT_{11} : to configure the flight entryس

As for the rest of this paper, we consecrate our study to the task-object ژT_{111} : to take knowledge of a new flightس. So we give the formal description and the simulation of its execution.

2.2. Task Control Structure (TCS)

The majority of methods presents the disadvantage that their descriptors are not exploited and their treatments are not detailed. TOOD has found a solution for this problem. Indeed, it adds a TCS (Task Control Structure) to each task-object (a gray net of figure 1). The TCS is modelized by a Coloured Petri Net (Jensen,1987ت) which we add three functions : (f) the input distribution function which selects the necessary input and control/command data to activate the task-object with a given event, (g) the output distribution function which selects the produced output data with a given reaction, and (δ) a priority function which arranges the enclenchement events of the task according to their importance (alarms, interruption, temporal constraints, etc...).

At any moment, the TCS must determine the task-object state; (treatment authorised, waiting for an enclenchement event, waiting for a resource, producing a reaction, etc...). At a given moment, the task-object state is given by the current marking of the TCS in the equation (1).

$$M' = M_0 + W \,.\, S \qquad (1)$$

$*$ **M’** : current marking of the TCS (vector with **7** dimensional)

$*$ **M_0** : initial marking of the TCS (vector with **7** dimensional)

$*$ **W** : incidence matrix of the TCS (matrix with a 7$*$2 dimensional). The (i, j) element of this matrix is equal to the difference between the Post (Pi, tj) and Pre (Pi, tj) functions

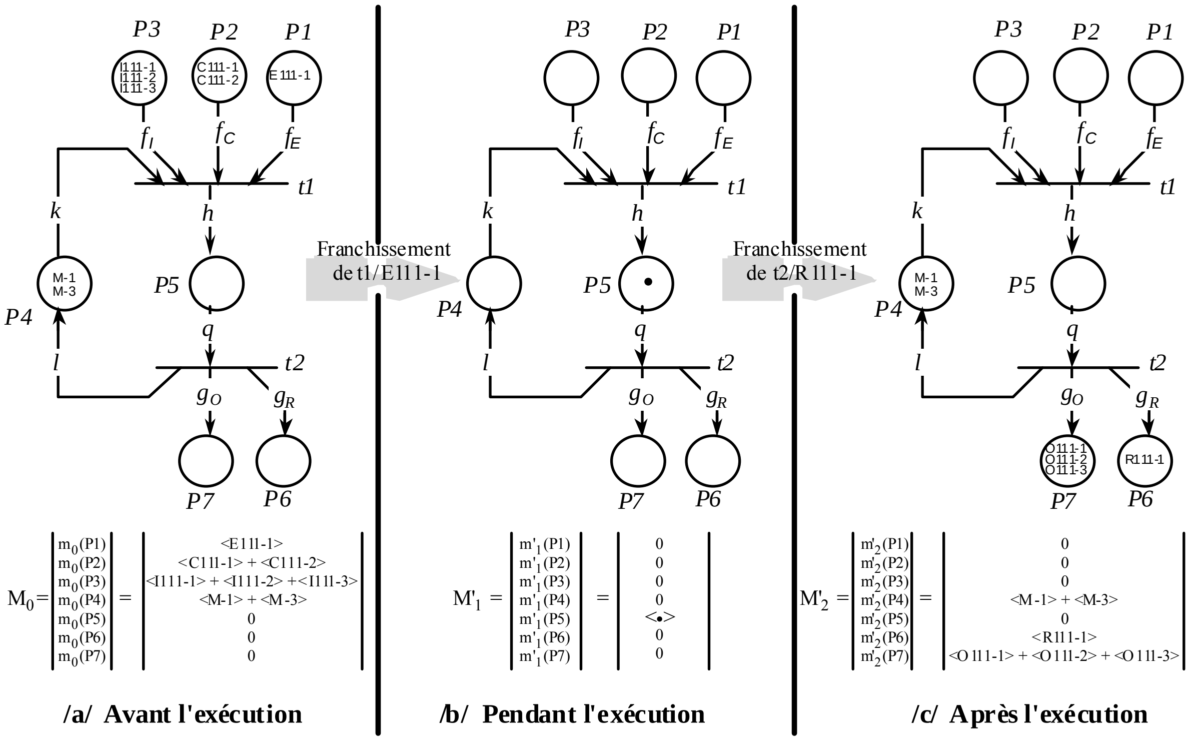
$$W = \begin{array}{|cc|c} t1 & t2 & \\ -f_E & 0 & P1 \\ -f_C & 0 & P2 \\ -f_I & 0 & P3 \\ -k & l & P4 \\ h & -q & P5 \\ 0 & g_R & P6 \\ 0 & g_O & P7 \end{array} \qquad (2)$$

$*$ **S** : A firing sequence of the task-object T_k. It’s a vector with 2 dimensional indicating with which event the firing of the input transition t1 must be effected and with which reaction the firing of the output transition t2 must be effected.

$$S = \left| \begin{array}{c} \langle E_{k,j} \rangle \text{ such as } \delta(E_{k,j}) = \sup(\delta(E_{k,1}), \delta(E_{k,2}),...) \\ \\ \langle R_{k,j} \rangle \end{array} \right| (3)$$

Si on reprend l'exemple de l'objet-tâche T_{111}, le *contr™leur organique* ne peut l'exژcuter qu'avec la prژsence de l'ژvژnement "E111-1 : Arrivژe d'un nouveau vol" c'est-ˆ-dire le contr™leur organique ne peut ouvrir et lire les informations d'un nouveau vol qu'aprڈs l'affichage du Strip correspondant sur le *tableau des nouveaux strips*. Une fois dژcidژ de lire les informations donc d'exژcuter l'objet-t‰che "T_{111} : Prendre connaissance d'un nouveau vol", il doit lire les donnژes d'entrژe I_{111-1}, I_{111-2}et I_{111-3} tout en se refژrant au donnژe de contr™le commande "C_{111-2} : Direction et type d'avion" qui influe sa faچon de lire et de traiter les donnژes d'entrژe. Mais aussi en respectant la donnژe de contr™le/commande "C_{111-1} : Le temps de temporisation " c'est-ˆ-dire qu'il a un temps limitژs pour lires les donnژes du nouveau vol.

Mathژmatiquement l'exژcution de l’objet-t‰che T_{111} se traduit par la prژsence d’un jeton du type < • > dans la place P5 (Figure 4). Cette prژsence ne peut گtre obtenue que par le franchissement de la transition d’entrژe **t1**. Or cette derniڈre est validژe dڈs la prژsence de l'ژvژnement d’enclenchement E_{111-1} dans la place P1, les donnژes de contr™le/commande nژcessaires dans la place P2, les donnژes d’entrژes nژcessaires dans P3 et les ressources nژcessaires dans P4.



$$M_0 = \begin{vmatrix} m_0(P1) \\ m_0(P2) \\ m_0(P3) \\ m_0(P4) \\ m_0(P5) \\ m_0(P6) \\ m_0(P7) \end{vmatrix} = \begin{vmatrix} <E111\text{-}1> \\ <C111\text{-}1> + <C111\text{-}2> \\ <I111\text{-}1> + <I111\text{-}2> + <I111\text{-}3> \\ <M\text{-}1> + <M\text{-}3> \\ 0 \\ 0 \\ 0 \end{vmatrix}$$

$$M'_1 = \begin{vmatrix} m'_1(P1) \\ m'_1(P2) \\ m'_1(P3) \\ m'_1(P4) \\ m'_1(P5) \\ m'_1(P6) \\ m'_1(P7) \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \\ 0 \\ <\bullet> \\ 0 \\ 0 \end{vmatrix}$$

$$M'_2 = \begin{vmatrix} m'_2(P1) \\ m'_2(P2) \\ m'_2(P3) \\ m'_2(P4) \\ m'_2(P5) \\ m'_2(P6) \\ m'_2(P7) \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \\ <M\text{-}1> + <M\text{-}3> \\ 0 \\ <R111\text{-}1> \\ <O111\text{-}1> + <O111\text{-}2> + <O111\text{-}3> \end{vmatrix}$$

/a/ Avant l'exécution **/b/ Pendant l'exécution** **/c/ Après l'exécution**

Figure 4. Simulation de l'objet-t‰che "T_{111} : Prendre connaissance d'un nouveau vol"

Le franchissement de la transition **t1** donne le marquage M’1 de la figure 4-b. Ce marquage peut گtre dژterminer par l’ژquation (1):

3. USER INTERFACE SPECIFICATION

The aim of this stage is the automatic passage from the users’ tasks description to the MMI specification. It allows to define all the necessary action plans and manipulated objects for the task executing. So, the resources of each terminal task-object become its component-objects which include MMI objects to be implementated in the future Interface, Application objects and Operator objects.

All the component-objects co-operate in a precisely manner in order to fulfil the aim of the terminal task-object. A component-object shall be defined from its class (Interface or Operator) and provided with a set of states and a set of operations (or actions) which allow to change these states.

Graphically, the component-object is presented in an identical structure that the one of a task-object. However its internal control structure called Object Control Structure زObCSس,inspired by the cooperative and interactive objects formalism proposed by Palanque (1992), is modelized by an Object Petri Net زOPNس (Sibertin, 85).The OPNs are characterized by the fact that the tokens which constitute the place markings are not atomic nor similar entities, but they can be distinguished from each other and take values allowing to describe the characteristics of the system.

The terminal task-object زT111 : to take knowledge the new flightس needs using two component-objects : MMI object زa New Strips Table : NSTس and operator object ز Organic Controller : OCس (figure 4). The comportment of the MMI object زa New Strips Tableس is defined by four states P1, P2, P3 and P4. From each state the Organic Controller can carry out a group of actions (transitions). From the P3 state (strip selected), for example, he has the possibility to achieve two actions : t3 (open a road-zoom) or t5 (temporize the new strip).

For the component-object زOrganic Controllerس, the set of states and operations represents the different possible procedures to execute the terminal task زT111 : to take knowledge of a new flightس. So, the display of a New Strip NS in the MMI object "new strips table" invokes, by the event E2,1, the operation service "Consult the NS" of the operator object "Organic Controller OC". According to his selection "Ch=", the organic controller carries out a first reading of the NS information ("Consult the road" or "Consult the level"). After this reading, he changes his state into cognition in order to evaluate his information level. Then he decides to "read again the basic information" or "to ask for additional information". The asking for additional information expresses itself by a change of his state into "Action" in order to "select the NS" and to "open the Road-Zoom". Both actions transmit R2,2 and R2,3 reactions to the component-object "new strips table". It is to be noticed that the organic controller carries out the action "open a road-zoom" only after receiving the event E2,2 confirming that the action "Select the NS" has been carried out. Once the Road-Zoom has been opened, the Organic Controller changes his state into "information reading" in order to read the additional information and then into the "situation evaluation" state to decide either to read again the information, or "to temporize the NS" or to invoke the terminal task-object "T112 : analyse the entrance conditions".

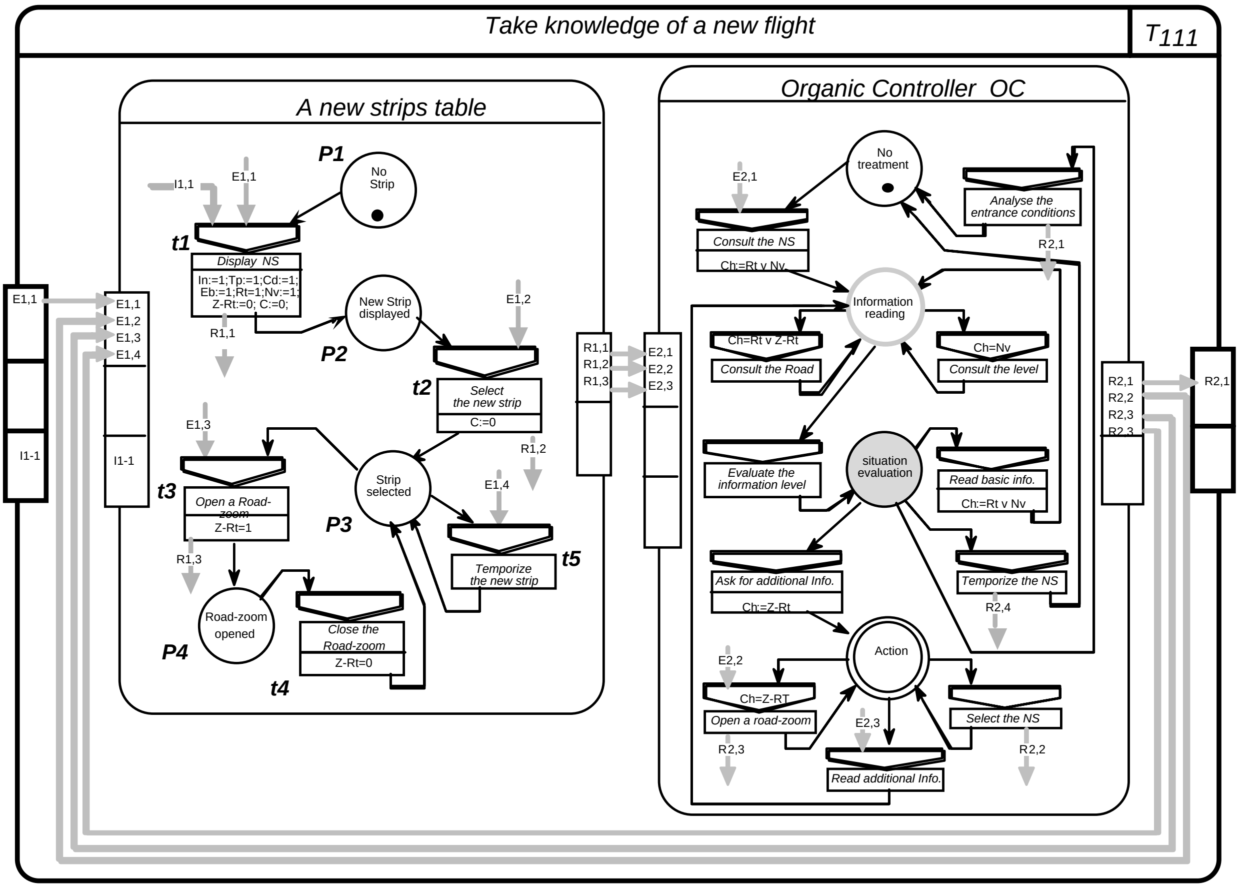


Fig. 4. A graphic Specification of the component-objects "New Strips Table" and "Organic Controller"

4. CONCLUSION

The TOOD methodology enjoys the contributions of methods and concepts taken from cognitive sciences and ergonomy domains together with those of the software engineering domain. It provides a framework of efficient collaboration between various users and between ergonomists and computer specialists. Its formalism allows on the one hand to define in a formal, coherent and structured way, the different entities intervening in the task model, and on the other hand to specify an adapted interface to the users' characteristics. Moreover, its mathematical formalism allows it to have a tool for the validation and the simulation. There is still to develop a language leading to its exploitation on a large scale.

5. REFERENCES

Abed, M., 1990, Contribution ˆ la modژlisation de la t‰che par des outils de spژcification exploitant les mouvements oculaires: application ˆ la conception et l'ژvaluation des Interfaces Homme-Machine [contribution to the task modelling with a specification tools exploiting the ocular activities : application to the man-machines interfaces design and evaluation], Doctorate thesis, University of Valenciennes France.

Cacciabue, 1988, Modelling Human Behaviour in the context of a simulation of Man-Machine Systems, Trainaing Human Decision Marking and Control, Elsevier Science Publishers B.V., North Holland,

Coutaz, J., 1987, PAC : An Implementation Model For Dialog Design. Interact'87, pp 431-436, Stuttgart.

Feller, A., and Rucker, R., 1990, Extending Structured Analysis Modelling with A.I.: An Application to MRPII Profiles and SFC Data Communications Requirements Specifications, IFIPS Conference.

Green, 1986, A survey of three dialogue models. ACM Transaction on graphics. **Vol. 5, Nb. 3** pp. 244-275.

Jensen, K., 1987, Coloured Petri Nets, In Petri Nets : Central models and their properties, LNCS **Nb. 254**, Spring Verlag.

Mahfoudhi, A. and Abed, M., 1994, Description orientژe objet des t‰ches du contr™leur pour la spژcification et la conception des interfaces [a controler's tasks object oriented description for the interfaces specification and design], Contract research report CENA 94, University of Valenciennes France.

Michel, L., 1991, Conception orientژe objet : Pratique de la mژthode HOOD [object oriented design : practice of the HOOD method], Dunod Press Paris.

Norman, D. and Draper, 1986, User centered system design, Lawrence Erlbaum Associates, Publishers.

Palanque, P, 1992, Modژlisation par objets Coopژratifs Interactifs d'interfaces homme machine dirigژes par l'utilisateur [modelling of the man machine interfaces managed by the users using interactive and cooperative objects], Doctorate thesis, University of Toulouse I France.

Richard J.F., Poitrenaud S., Tijus C.A., Barcenilla J., 1992 Semantic of action networks : a cognitive model for human and system interaction. HCI 92, Universitژ de Paris 8.

Scapin, D.L. and Pierret-Golbreith, C., 1989, MAD : Une Mژthode Analytique de Description de T‰ches [MAD : a tasks analytic description method], Actes du Colloque sur l'ingژnierie des Interfaces Homme-Machine, Sophia-Antipolis, France.

Sibertin, B.C, 1985, High-level Petri nets with Data Structure. 6th European Workshop on Petri Net and application, Espoo, Finland.