TOOD : TASK OBJECT ORIENTED DESCRIPTION FOR ERGONOMIC INTERFACES SPECIFICATION

A. Mahfoudhi, M. Abed, J-C. Angué

L.A.M.I.H., University of Valenciennes,
B.P. n 311 59304 Valenciennes Cedex FRANCE
Tel. : + (33).27.14.14.61 E-mail : mahfoudhi@univ-valenciennes.fr

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 Description methodology (TOOD). It tries to relate the characteristics of the user's task with those of the interface. The introduction of ergonomic concepts allows to take the human factors into account. And the joint use of the object oriented techniques and the High Level Petri Nets supplies complete, coherent and re-usable entities allowing to give a formal description of the interactive systems' characteristics and an appropriate specification of the user interface. An example, extracted from the air traffic control, is presented to illustrate this new methodology.

Keywords : Task Analysis, Man-Machine Interface, Specification, Co-operation, Object Oriented Techniques, Object Petri Nets, Human Factors.

1. INTRODUCTION

The technological evolution and increase of economic constraints have given birth to high-performance systems. However, several ergonomic problems are still present during the complex systems design where the presence of operators is and will remain necessary. These problems can be essentially related to the lack of taking into consideration the users' knowledge and human factors which are either implicit or explicit during the evaluation phase, and the absence of formal method for the Man-Machine Interface specification. So, it's very important to have models and methods that allow, on the one hand to make more accessible the users' knowledge, and mainly to make more formal and more detailed their description. On the other hand to allow the specification of the communication interface (Man-Machine Interface : MMI).

Over the last few years, the domain of man-machine interaction has been enriched with new methods and techniques. Some of them try to exploit the knowledge acquired by the cognitive theories in order to describe the operator tasks and its strategies (Richard *et al.*, 1992). The others use the simulation to study the decisional behaviour of the human operator (HO) (Cacciabue, 1988). At the same time, studies are carried out in order to modelize the man-machine interaction (Norman *et al.*, 1986) and to modelize the user interface (Coutaz, 1987) and (Green *et al., 1986*). Recently several users' tasks analysis and description methods, such as MAD proposed by Scapin and Pierret-Golbreith (1989) or the SADT/Petri method proposed by Abed (1990), have tried to describe the task as a set of operations and, at the same time, they integrate implicit information about human factors.

Our contribution consists in the proposition of a global methodology, called Task Object Oriented Description (TOOD), beginning from the task analysis and description to the MMI specification, based on the object oriented techniques and Object Petri Nets (OPN). TOOD has three goals : (1) to assure the transition between the different phases of the system development, (2) to allow the taking into

consideration of the appropriate knowledge of the three system's components (the process or application, the MMI and the user), and (3) to offer a framework of efficient collaboration between the different contributors in the system development (ergonomists, computer specialists and users).

This article is structured into two sections : the first one presents a brief discussion of the tasks collection and modelling formalism used by TOOD, called "External Model". The second section explains the transition from the external model to the MMI specification called "Internal Model". The examples presented along this article are provided in the air traffic control context (Mahfoudhi and Abed, 94).

2. USERS' TASKS DESCRIPTION : "EXTERNAL MODEL"

Like the majority of task description methods, TOOD advocates the hierarchical approach for the tasks description. Although it's highly structured, the hierarchical approach allows to represent the user's cognitive model. Indeed, it allows to identify all the tasks to be carried out by the user with different choice of actions and possibilities of sequences.

Before presenting the different stages of the external model, it is advisable to define the different used terms. The generic term "Task-Object" 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. Let us consider the air traffic control, "to plan the traffic" can be regarded as a task-object. In order to reduce its abstraction, this task-object can be decomposed into two children task-objects : "to plan the traffic in the sector" and "to plan the traffic in the sector's frontiers".

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.

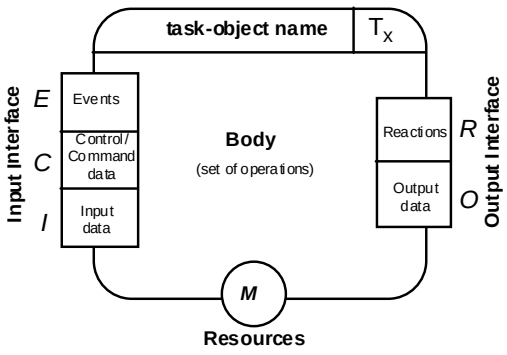


Fig. 1. graphic structure of the task-object

A task-object is also defined by a set of attributes, called "descriptors", that defines the execution conditions and the effects of the task, as well as the actions or sub-tasks to be carried out to reply to a given functional context.

Table 1. the task-object's descriptors.

Descriptor	Definition
Name	a name which identifies the task.
Input Interface	identifies the initial state of task-object. It is composed of three elements (E, C, I).
E	define the necessary Events for the task release.
C	Control/Command data : define the constraints to be respected at the time of the task execution.
I	Input data : define the list of data and information transformed by the task execution.
Output Interface	identifies the final state of task-object. It is composed of two elements (R, O).
R	Reactions : define the reports of the task-object execution (action realized or service demanded from other task-object).
O	Output data : defines the list of modified input data and/or a new data created after task-object execution.
Body	describes the operational model of the task-object.
Resource	defines the necessary human and material entities for the task-object execution.

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, the notion of TCS (Task Control Structure) exploiting the task object's descriptors, allows to describe precisely the different task-object's states.

To establish the external model, three stages must be realized :

- task-objects identification,
- task-objects specification,
- description of the TCS (**T**ask **C**ontrol **S**tructure).

2. 1. Task-objects identification

In order to identify all the tasks to be designed in the

future system, TOOD begins by analyzing the existing system and the operator's current tasks. It allows to avoid the disadvantages of the existing system and to add the new desirable characteristics.

By a hierarchical decomposition, TOOD 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).

Once all future system's tasks are identified, TOOD defines the constraints and relations between them. It also makes the distinction between the users' tasks and the computer's tasks. Indeed, it attributes all the interactive and manual tasks to the user, while the automatic tasks are attributed to the computer. Once the allocation bas been effected, TOOD takes an interest in the users' tasks because only this category can be used for specifying the users' interfaces.

2. 2. Task-objects specification

This specification stage defines all the execution conditions and the effects of each task-object. Based on the encapsulation concept, it describes what the task-object can do through the input/output interfaces.

For each task-object, the specification consists in :

- listing and identifying all the enclenchement events and the reactions. Indeed, the task execution can be asked for by the emission of an event, while the treatment report of the task-object is provided by a reaction.
- listing and identifying all the input, output and control/command data required and supplied by the task-object.

2. 3. Task Control Structure : TCS

The last stage describes the dynamic task-object behaviour. To that aim, a Task Control Structure "TCS" is used. The TCS is modelized by a Coloured Petri Net (Jensen,1987ت) called "Petri Net Task Control Structure: PNTCS" (Fig. 2).

In order to remove any lack of determinism, two functions of data distribution (f and g) and a priority function (δ) are associated to the input and output transitions. Indeed, the functions f and g group together all the Pre and Post functions usually combined with the arcs of a coloured Petri net. Their aim is, on the one hand to select the necessary input and control/command data to activate the task-object with a given enclenchement event; on the other hand to specify the output data produced with the reaction. The priority function (δ) arranges the enclenchement events of the task according to their importance (alarms, interruption, temporal constraints, etc.).

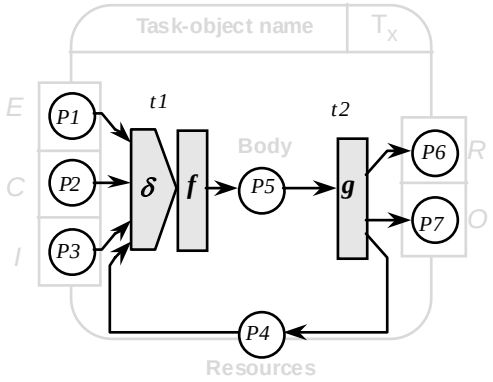


Fig. 2. Task Control Structure (TCS)

The formal aspect of the TCS allows to identify, at any time, the current state of the task-object (treatment authorised, waiting for an enclenchement event, producing a reaction, etc....).

The resulting document of the external model includes two kinds of description : a graphic specification for a clean, legible and exploitable representation, and a textual specification for a complete description (Mahfoudhi *et al.*, 94).

3. USER INTERFACE SPECIFICATION : "INTERNAL MODEL"

The aim of this stage is the automatic passage from the users' tasks description (external model) to the MMI specification (internal model) Fig. 3. It completes the external model by defining the operational level of each terminal task-object. Indeed, 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 belonging to the classes : Interface, Application and Operator (Fig. 3).

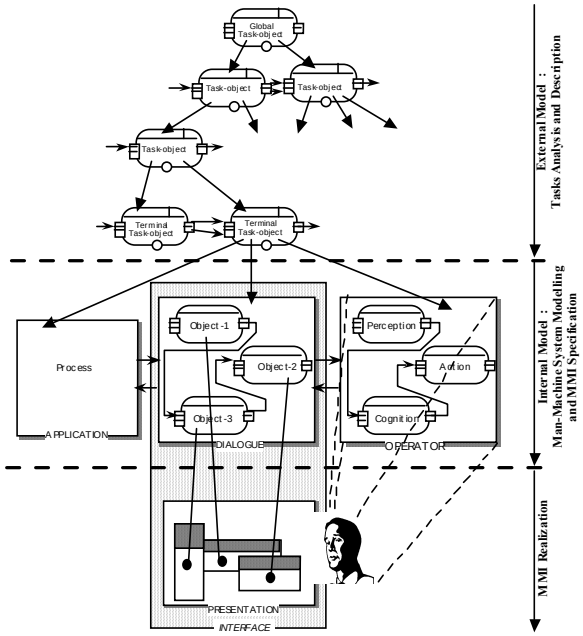


Fig. 3. From the users' tasks description to the MMI specification.

The MMI specification is carried out on two stages. The first one consists in specifying of the component-objects of each terminal task-object, with an aggregation of these component-objects; the second stage allows to specify the user interface.

3. 1. Component-objects specification

All the component-objects co-operate in a precisely manner in order to fulfil the aim of the terminal task-object in reply to a given functional context. 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. For example, from the P3 state (strip selected) of the component-object "new strips table" the operator has the possibility to carry out two actions : t3 (open a road-zoom) or t5 (temporize the new strip) Fig.4. On the other hand, the set of states and operations of an Operator component-object represents the different possible procedures for the execution of the terminal task. Indeed, the procedure represents the different activity phases of a human operator : situation apprehension, goals identification, preparation of an action plan, application of this action plan, control of the situation, correction (Norman *et al.*, 86).

Graphically, the component-object is presented in an identical structure that the one of a task-object in the external model. However its internal control structure called Object Control Structure "ObCS" is modelized by an Object Petri Net "OPN" (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.

In addition to its formal aspect, the ObCS enjoys a simple and easily understandable graphic representation, allowing to represent - with the places of the OPN - all the possible states of the component-object, and with the transitions, to represent all the operations and actions that can be taken from these states. The graphic representation used for the ObCS is inspired by the cooperative and interactive objects formalism proposed by Palanque (1992).

The different states of the operator component-object come down to the three states : Perception, Cognition and Action. The ObCS allows the distinction between these states (table 2).

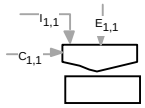
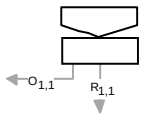
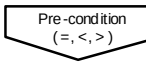
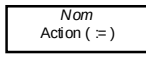
Table 2. the states of the operator component-object

Place	State
○	perception
●	cognition
◎	action (an operator action on the interface)

The communication between the component-objects is carried out through their input and output interfaces. So, an action "A" executed by a component-object "X" (operator) on the component-object "Y"the :can be read as (interface) execute "X"object -components his reaction operation corresponding to his quiry of the action A. This execution is rendered by a reaction R in the output interface of the component-object X. The output interface transmits the reaction R to the input interface of the component-object Y. So the reaction R becomes an event E. And lastly this event activates the service operation of the component-object Y corresponding to the action A asked by the component-object X.

Graphically, the ObCS allows the distinction between the different kinds of operations (private operation, service operation and reaction operation) table 3.

Table. 3. the different transitions in the ObCS

Transition	Description
	**tP**: private transition does not communicate with the interfaces of the component-object.
	**tS**: service transition. The Ei, Ci and Ii communicate with the input interface.
	**tR**: reaction transition. The Ri and Oi communicate with the output interface.
	The Pre-condition part of the transition which has a test about the input data of the transition.
	The action part of the transition which has the name and the expression of the operation

The execution of a service (service operation) by the component-object is equivalent to the fire of the service transition (tS) associated to this service.

The fire of a reaction transition (tR) renders the fact that the component-object executed the reaction operation (produced a reaction) associated with this transition. The ObCS can also contain transitions that are not associated to any services nor to a reaction; these transitions are called “Private Transition” (tP). They modelize the internal changes of the states of the component-object.

An example taken from the air traffic control, corresponding to the terminal task-object “take knowledge the new flight” taken from (Mahfoudhi and Abed, 94), needs using two component-objects : “a New Strips Table : NST” and “ Organic Controller : OC” (figure 4). The comportment of the component-object “a New Strips Table” 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 “Organic Controller”, the set of states and operations represents the different possible procedures to execute the terminal task “Take knowledge of a new flight” in reply to a given functional context. So, the display of a New Strip NS in the component-object "new strips table" invokes, by the event E2,1, the operation service "Consult the NS" of the component-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".

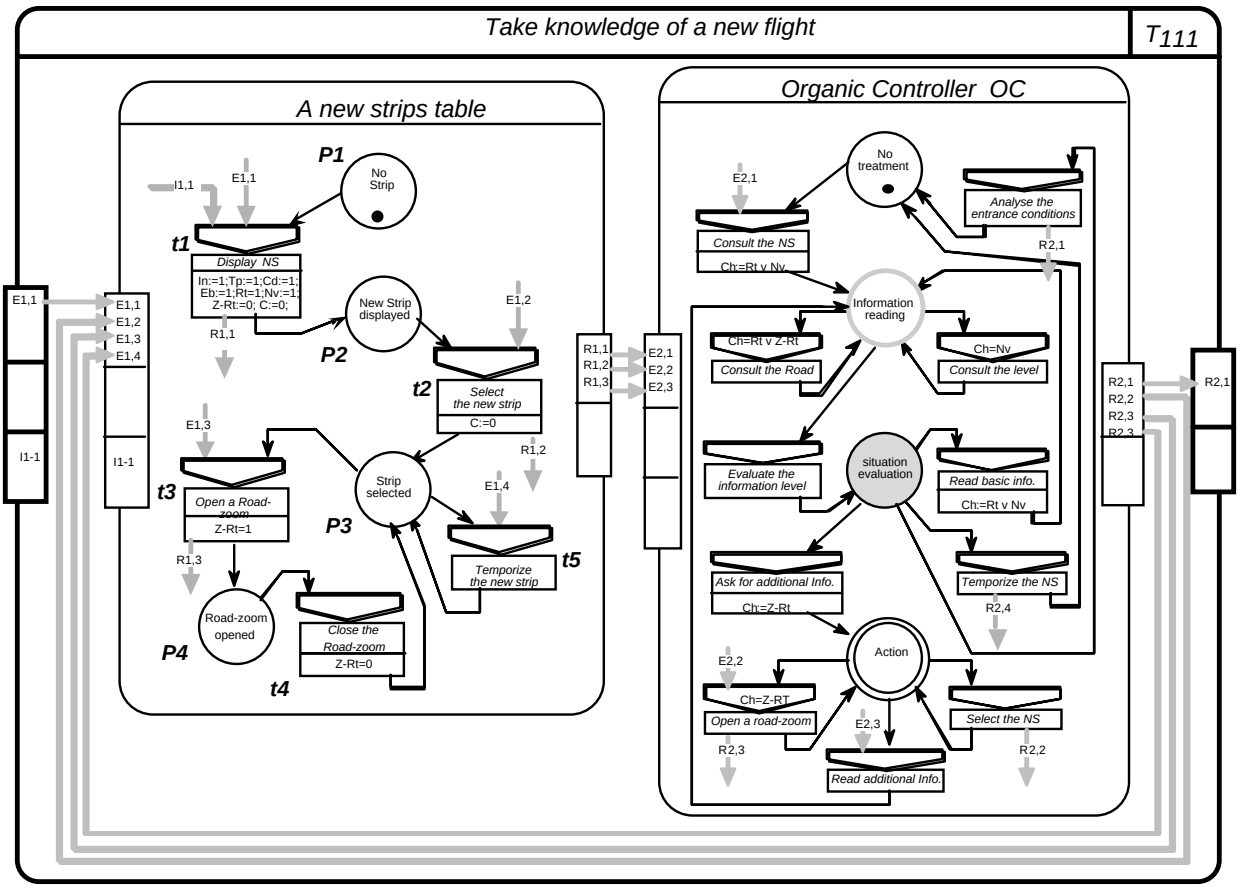


Fig. 4. A graphic Specification of the component-objects "New Strips Table" and "Organic Controller"

3. 2. Aggregation mechanism

In order to realize the MMI in its real structure, the construction of the object classes of the MMI suggests the aggregation of the different component-objects which have the same name, specified during the description of the internal model of each terminal task-object. This aggregation mechanism is comparable to the composition relation of the HOOD method called the parent/child relation.

Thus, an object class of the MMI is built according to the duplication of all the elements (events, control/command data, input data, reactions, output data and ObCSs) of the component-objects which have the same name. Yet, there would be some modifications which are detailed in (Mahfoudhi and Abed, 94).

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 modˆlisation de la t‰che par des outils de spˆcification exploitant les mouvements oculaires: application ˆ la conception et l'ˆvaluation 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 orientˆe objet des t‰ches du contr™leur pour la spˆcification 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 orientˆe objet : Pratique de la mˆthode 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, Modˆlisation par objets Coopˆratifs Interactifs d'interfaces homme machine dirigˆes 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, Universitˆ de Paris 8.

Scapin, D.L. and Pierret-Golbreith, C., 1989, MAD : Une Mˆthode Analytique de Description de T‰ches [MAD : a tasks analytic description method], Actes du Colloque sur l'ingˆnierie 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.