Towards a User Interface Generation Approach Based on Object Oriented Design and Task Model

Adel Mahfoudhi
Department of Computer Science, Science Faculty of Sfax
Rte Soukra km 3,5 BP : 802
3018 Sfax (TUNISIA)
adel.mahfoudhi@fss.rnu.tn

Mohamed Abid
National Engineering School of Sfax
Rte Soukra 3018 Sfax (TUNISIA)
mohamed. abid@enis.rnu.tn

Mourad Abed
Christophe KOLSKI
LAMIH (UMR CNRS 8530)
Université of Valenciennes
BP : 311 – 59304 Valenciennes cedex9
(France)
mourad.abed @univ-valenciennes.fr
Christophe.Kolski@univ-valenciennes.fr

ABSTRACT

This paper presents an approach to generate the user interface from the task model. Our works are situated in the course of approaches based on models. This approach called TOOD (Task Object Oriented Design) is based on a formal notation, which gives quantitative results which may be checked by designers and which provide the possibility of performing mathematical verifications on the models. The modelling formalism is based on the joint use of the object approach and high level Petri nets. The TOOD method integrates different models (task model, user model, local model of the interface, abstract model of the interface, model of the implementation) and their relations. An example, extracted from the air traffic control, is presented to illustrate TOOD methodology.

Keywords
Task Model, User Interface Specification, Formal Method, Object Approach, Petri Nets.

INTRODUCTION

Several research projects have been dedicated to the modelling of user tasks in the field of interactive system design (see, for example, the work concentrating on the following methods: MAD [18], DIANE [1], GOMS [3]). However, their actual use is far from being a widespread practice. One of the possible reasons for this is that they do not use truly formal methods, which make it possible to provide the task models with conciseness, coherence and non-ambiguity [14]. What is more, these projects suffer not only from their lack of integration into a global design process covering the entire life cycle of the User Interface (UI) but also from the lack of modelling support software.

In order to overcome these problems, current research projects are oriented towards a methodological framework which covers all stages from the first activity analysis stage up to the stage of the detailed specification of the UI: The methods MAD* [6], DIANE+ [21], GLADIS++ [22], ADEPT [8] and TRIDENT [24] go in this direction. These design methodologies are based on several models (task model, user model, interface model) and are aided by tools for the implementation of these models.

Our research work falls into this category, but we emphasise the formal aspects of model representation and their transformation throughout the stages of the design process. The TOOD method is based on the representation that the user has of the task, apart from the considerations of computer processing. Like the UML/PNO method [4], HOOD/PNO [16] , [4], [7], [17] and ICO [15], the TOOD method uses the object approach and the object Petri nets to describe, on the one hand, the functional aspects and the dynamics of the user tasks, and on the other hand the behavioural aspects of the HCI and of the user in order to specify how the tasks are performed. Its formalism aims at covering the entire development cycle from the analysis of what exists, up to the detailed design and implementation.

TOOD AND THE CYCLE OF DEVELOPMENT OF USER INTERFACE

The TOOD design process can be divided into four major stages [11], [20] (Figure 1).

- The analysis of the existing system and of the need is based on its user's activity and it forms the entry point and the basis for any new designs.
- The Structural Task Model (STM) is concerned with the description of the user tasks of the system. It makes it possible to describe the user task in a coherent and complete way.
- The Operational Model (OM) makes it possible to specify the UI objects in a Local Interface Model (LIM), as well as the user procedures in a User Model (UM) of the

system to be designed. It uses the needs and the characteristics of the structural task model in order to result in an Abstract Interface Model (AIM) which is compatible with the user's objectives and procedures.

- The realisation of the UI is concerned with the computer implementation of the specifications resulting from the previous stage, supported by the multi-agent software architecture defined in the Interface Implementation Model (IIM). Analysis of the existing system

To know what the operator is presumed to do using the new system, we must know what is achieved in real work situations (the activity analysis) using an existing version of the system or a similar system.

STRUCTURAL TASK MODEL (STM)

After the stage of the existing system analysis and its user's activity, the structural task model (STM) makes it possible to establish a coherent and complete description of tasks to be achieved on the future system, while avoiding the inconveniences of the existing system and adding the new required functions and features. For that, two types of model are elaborated: a static model (SSTM) and a dynamic model (SDTM).

The construction of the structural model is composed of four iterative stages:

- Hierarchical decomposition of tasks.
- Identification of objects and their components.
- Definition of the dynamics of the elementary tasks.
- Integration of the task competition

Static Structural Task Model (SSTM)

The structural model enables the breakdown of the user's stipulated work with the interactive system into significant elements, called tasks. Each task is considered as being an *autonomous* entity corresponding to a goal or to a sub-goal, which can be situated at various hierarchical levels. This goal remains unchanged in the various work situations. In order to perfect this definition, TOOD formalises the concept of tasks using an object representation model, in which the task can be seen as an *Object*, an instance of the *Task Class*. This representation, consequently, attempts to model the task class by a generic structure of coherent and robust data, making it possible to describe and organise the information necessary for the identification and performance of each task.

Two types of document (graphical and textual), as shown in Figure 2, define each task class.

The task class is studied as an entity using four *components*: the Input Interface, the Output Interface, the Resources and the Body. We also associate a certain number of *identifiers* to these describers, which makes it possible to distinguish the Task Class amongst the others: Name, Goal, Index, Type and Hierarchy. This parallel with software engineering guarantees a strong link between a user-centred specification based on ergonomic models and the software design based on the object model. There are defined as follows:

A **name**, action verb followed by a complement (object treated by the task), reflecting the treatment to be performed by the task. It is preferable for the name to include vocabulary used by the users in order to respect the terminology during the development of the interface.

A **goal,** explanation in natural language of the goal which the user or application wishes to reach via the task.

An **index,** formal identifier of the task formed using the number of the master task, to which the sequential number corresponding to the said task is added.

A **type,** nature of the task; this designates its category: human, automatic or interactive.

A **hierarchy**, number of task classes composing it; represented by a series of small squares.

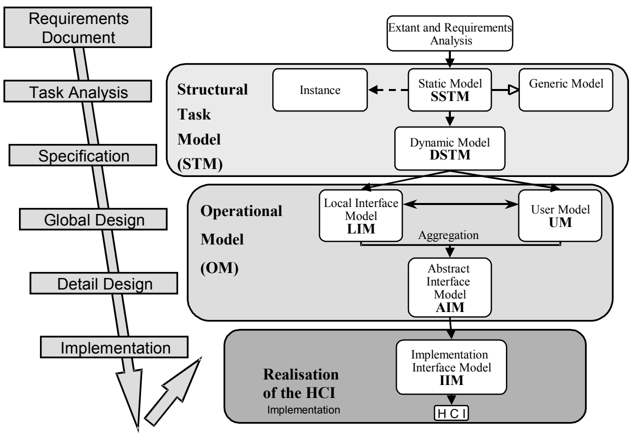
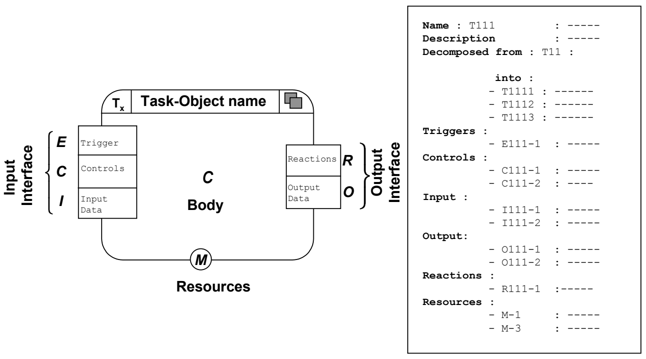


Figure 1. TOOD and the development cycle for the interface



```
Name : T111           : -----
Description           : -----
Decomposed from : T11 :

           into :
          - T1111 : ------
          - T1112 : ------
          - T1113 : ------
Triggers :
          - E111-1  : -----
Controls :
          - C111-1  : -----
          - C111-2  : ----
Input :
          - I111-1  : -----
          - I111-2  : -----
Output:
          - O111-1  : -----
          - O111-2  : -----
Reactions :
          - R111-1  :-----
Resources :
          - M-1     : -----
          - M-3     : -----
```

Figure 2. Generic structure of the class-task

		Description
Input Interface	***Triggers***	*Events which bring about the performance of the task. They are classed into two categories :* • *Formal or explicit trigger events, which correspond to external triggers. They appear in an observable way in the work environment (information on screen, button press, communication, ...). The tasks triggered by this type of event are considered as being compulsory; that is, their performance is vital.* • *Informal or implicit trigger events, which correspond to triggers, brought about following a user decision, from information characterising its work situation. Unlike the formal events, they are not visible to an outside observer, but may be expressed verbally*
	Contextual conditions	*Information which must be checked during the performance of the task. These conditions affect the way in which the task is performed.*
	Input data	*Information necessary during the performance of the task.*
Output Interface	***Reactions***	*Results produced by the performance of the task. Their content indicates the following type of modification :* • *Physical and, in this case, it indicates the modification of the environment (application call, change of state,)* • *Mental, indicating the modification or a new representation of the situation by the user.* *The Reactions thus determine whether the aims are attained or not and, in such a case, the task will be repeated after a possible development of the situation.*
	Output data	*Data transformed or created by the performance of the task..*

Table 1. Input and Output Interface components

The **body** : central unit of the task class. For intermediate or hierarchical tasks, it gives the task procedure diagram, that is to say the logical and temporal relations of the sub-tasks. These relations reflect, in a certain way, the user's work organisation.

- **Resources,** human users and/or interactive system entities involved in the performance of the task

The input interface specifies the initial state of the task. It defines the necessary data to the task execution. These data are considered as the initial conditions to be satisfied at the beginning of the task. It is composed of three categories of information (table 1).

The output interface specifies the final state of the task. It is composed of two types of data (table 1).

The resources and the information of the input and output interfaces are modeled by objects, called "describer objects", instances of describer classes (Figure 2).

Once all future system 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 attributes. The resulting document of this specification includes two kinds of descriptions: a graphic description and a textual one (figure 3).

Dynamic Structural Task Model (DSTM)

The Dynamic Structural Task Model (DSTM) aims at integrating the temporal dimension (sequencing, synchronisation, concurrency, and interruption) by completing the static model. The dynamic behaviour of tasks is defined by a control structure, called TCS (Task Control Structure), based on an Object Petri Net (OPN). It is merely the transformation of the static structure. This TCS describes the input interface's describer objects, the task activity, and the release of describer objects from the output interface as well as the resource occupation.

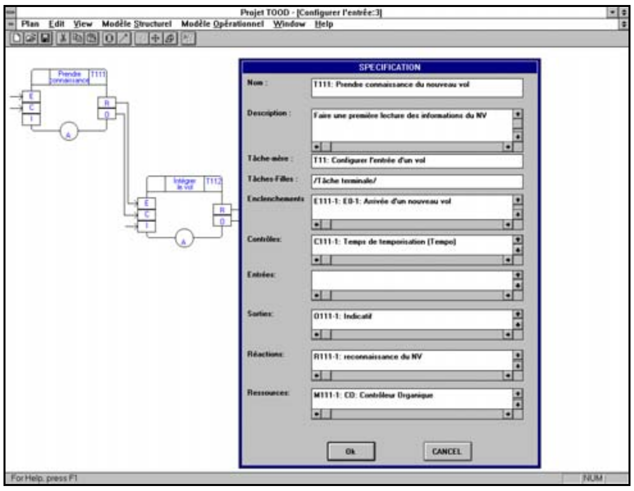
Each TCS has an input transition t1 and an output transition t2 made up of a selection part and an action part. The functions associated with each transition allow the selection of objects and define their distribution in relation to the task activity (Figure 4).

The selection part of transition t1 is made up of three functions: δ, β, χ

- **Priority function** δ makes it possible to select the highest priority trigger for the task. This function is the basis of the interruption system. It allows the initiation of a task performance, even if another lower priority task is being carried out. However, the performance of the task in relation to this trigger remains subject to the verification of the completeness and coherence functions.
- **Completeness function** β checks the presence of all the describer objects relating to an observed event, that is to say the input data, the control data and the resources used to activate the task class in relation to a given trigger event.
- **Coherence function** χ assesses the admissibility of these describers in relation to the conditions envisaged for the task. This function is a set of verification rules which use simple logical or mathematical type operators and which obey a unique syntax making their formulation possible.

The selection part of transition t2 has a **completeness function** ρ which checks the presence of output data and resources associated with the reactions released by the body of the task.

The hierarchical tasks are considered to be **control tasks** for the tasks of which they are composed. Consequently, the action parts of the input and output transitions of their TCS possess respectively an emission function ϕ and a synchronisation function σ**.**


Projet TOOD - [Configurer l'entrée:3]

SPECIFICATION

Nom :	T111: Prendre connaissance du nouveau vol
Description :	Faire une première lecture des informations du NV
Tâche-mère :	T11: Configurer l'entrée d'un vol
Tâches Filles :	/Tâche terminale/
Enclenchements	E111-1: E0-1: Arrivée d'un nouveau vol
Contrôles:	C111-1: Temps de temporisation (Tempo)
Entrées:	
Sorties:	O111-1: Indicatif
Réactions:	R111-1: reconnaissance du NV
Ressources:	M111-1: CO: Contrôleur Organique

Figure 3. Graphic and textual specification of the task-object « T11 : to configure the flight entry »

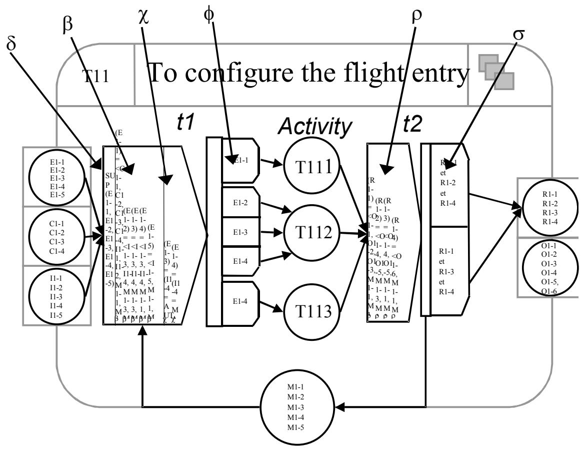


Figure 4. TCS : Task Control Structure

Function ϕ defines the **emission rules** (constructors of the input transition) for transition t1, for the activation of the sub-tasks, as well as the distribution of data used by these sub-tasks. Function σ defines the **synchronisation rules** (constructors of the output transition) for the sub-tasks.

OPERATIONAL MODEL (OM)

This stage has as an objective the automatic passage of the user tasks description to the specification of the HCI. It completes the external model describing the body of terminal task-objects in order to answer the question how to execute the task? (in terms of objects, actions, states and control structure).

At this level we integrate resources of every terminal task-object in its body. These resources become, in this way, component-objects, belonging to the classes Interface, Machine, Application and Human Operator. The modeling of the class application is not addressed in this paper.

The specification of the UI passes through two stages. The first corresponds to the specification of component-objects of every terminal task, and by a process of aggregation of these component-objects. The second stage makes it possible to specify the UI objects.

Specification of components-object

All the component-objects cooperate in a precisely defined manner in order to fulfill the aim of the terminal task-object in response 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 the change of 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), as shown in figure 5. 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 [16].

Graphically, the component-object is presented in an identical structure to the one of a task-object in the structural model. However its internal control structure called Object Control Structure "ObCS" is modeled by an Object Petri Net "OPN". The OPNs are characterized by the fact that the tokens which constitute the place markings are neither atomic nor similar entities, but they can be distinguished from each other and take values, making it possible to describe the characteristics of the system.

In addition to its formal aspect, the ObCS enjoys a simple and easily understood graphical representation; making is possible 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 [15].

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"(interface) can be read as the component-object "X" executes its operation reaction corresponding to the query 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.

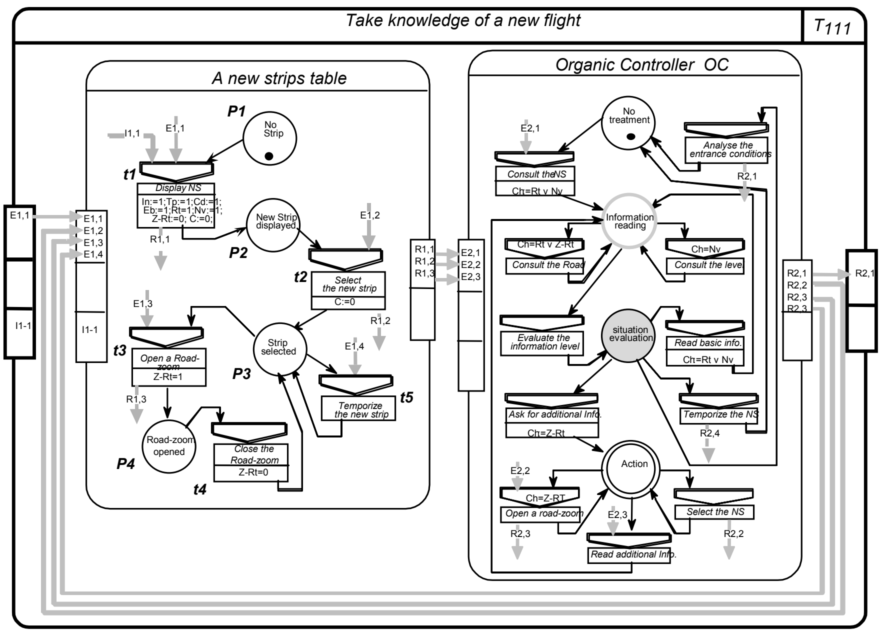


Figure 5 : A graphic Specification of the component-objects "New Strips Table" and "Organic Controller"

An example from air traffic control, corresponding to the terminal task-object “take knowledge the new flight” taken from [10], needs the use of two component-objects: “a New Strips Table: NST” and “Organic Controller: OC” (figure 5). The behaviour 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 should be noted 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 the information again, or "to temporize the NS" or to invoke the terminal task-object "T112: to analyze the entrance conditions".

Aggregation Mechanism

In order to realize the HCI in its real structure, the construction of the object classes of the HCI 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 HCI is built according to the duplication of all the elements (triggers, contextual conditions, input data, reactions, output data and ObCSs) of the component-objects which have the same name.

The explanatory example in figure 5 corresponds to the class “new strips table” constructed by aggregation of the component-objects “new strips table” of the terminal task-object " T111: To take knowledge of a new strip ", and the one of the terminal task-object " T1122: To take decision on conditions of entrance ".

HCI IMPLEMENTATION

The HCI implementation model in the TOOD methodology is the presentation specification of the final interface as it will be seen by the user. It corresponds to the specification of the Presentation components of the Seeheim model or presentation and action languages.

The construction of this model takes place through the translation of objects, states, actions and ObCS to screens, menus, windows, icons, This translation depends on a collection of criteria and ergonomic rules [2], of guides [23] and of heuristics [12].

The following figure (figure 6) schematizes the prototype of simulation of the future objects oriented interface of the PHIDIAS system (HEGIAS) that corresponds to the development of the Implementation Model. This development, made by the CENA, concerns the position of the Organic Controller (OC). It includes four objects:

- A **radar picture** that displays the limits of the controlled sector, the plane tracks, and labels associated with the plane tracks. A clock (HH:MM) is presented in a permanent way,
- A **new strips table**, situated in the upper left part of the screen. Strips are presented according to an automatic ordering by geographical flow.
- A **built-in strips table**, situated in the left bottom part of the screen,
- A **work zone**, situated in the right bottom part of the screen. It is reserved for displaying one of the following entries: the list of flights in account, help in entrance, help in exit or strips withdrawn by anticipation.

There are four input tools: a mouse, two tactile screens, and a mini-keyboard. With these tools, the OC has the possibility to act directly on the interface. He can integrate a new flight, consult a road-zoom, consult help in entrance or in exit for a flight, etc.

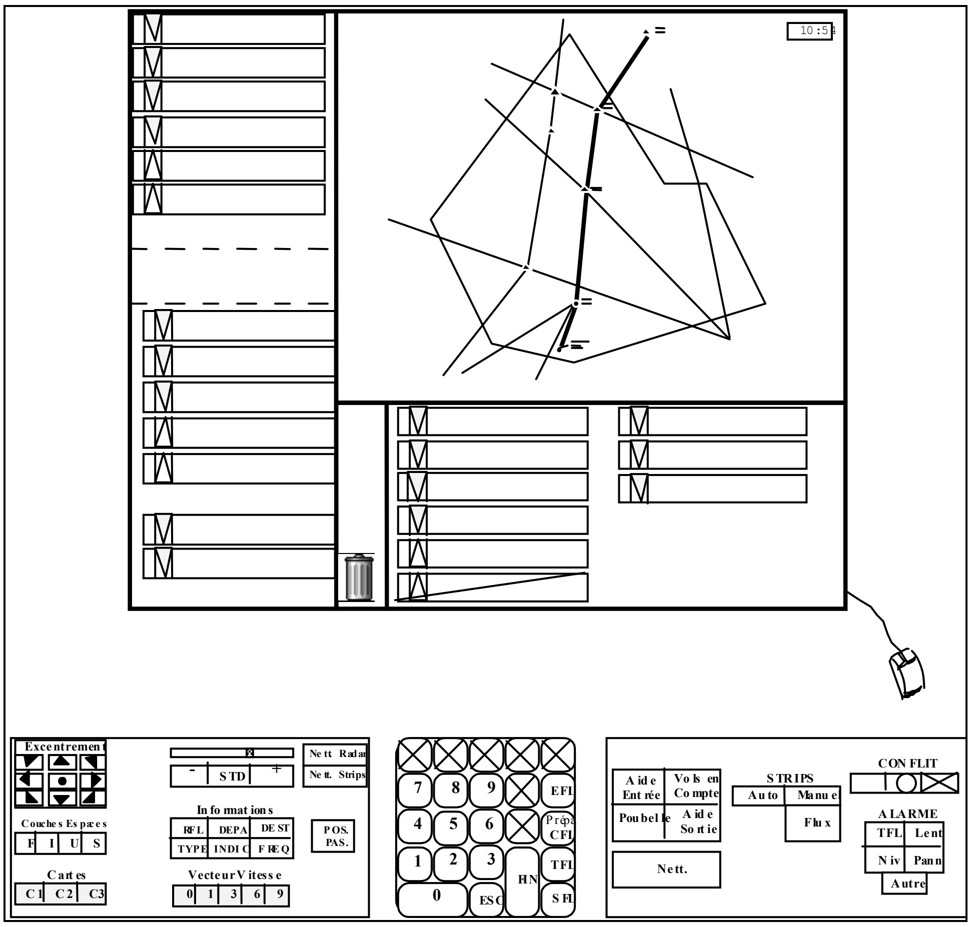


Figure 6. Simulation Prototype of the future air traffic control interface (HEGIAS) specified by TOOD

CONCLUSION

The use of the object oriented approach and object Petri nets presents several advantages for the modeling of the user task. Indeed, the TOOD task model, through its static and dynamic description, allows the modularity of specifications, the expression of interruptions and concurrency. The addition of describer objects to the task entity enables a connection to a programming language, which simplifies the passage to implementation.

Moreover, the TOOD method can contribute towards helping with communication between the different actors in the design process through its formal description.

The operational model leads to the specification then to the generation of the HCI. This model is developed from the structural model while using the same formalisms which ensures the semantic stability of the TOOD method.

REFERENCES

1. Barthet, M.F. (1988), Logiciels interactifs et ergonomie: modèles et méthodes de conception, Dunod.
2. Bastien J.M.C. & Scapin D.L. (1995), Evaluating a User Interface With Ergonomic Criteria. International Journal Of Human-Computer Interaction, 7(2), pp. 105-121.
3. Card, S.K, Moran, T.P, Newell A. (1983), The Psychology of HCI. In Lawrence Erlbaum Ass (Ed.). London.
4. Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F. et Jeremaes, P. (1994). Object-Oriented Development : The Fusion Method. Prentice Hall.
5. Delatour, J. & Paludeto, M. (1998), De HOOD/PNO à UML/PNO : Une méthode pour les systèmes temps réels basée sur UML et objets à réseaux de Petri. Rapport LAAS N° :98248.
6. Gamboa-Rodriguez, F. (1998), Spécification et implémentation d'ALACIE: Atelier Logiciel d'Aide à la Conception d'Interfaces Ergonomiques. Thèse en sciences: Université Paris XI.
7. Jaaksi, A. (1995). Object-Oriented Specification of User Interfaces. Software – Practice & Experience, Vol.25, No.11, pp.1203-122.
8. Johnson, P., Johnson, H. Wilson, S. (1995), Scenario-based design and task analysis. In Carroll, J.M. (Ed.). Scenario-based design: Envisioning work and technology in system development. Willey.
9. Mahfoudhi A., Abed M. & ANGUE J-C. (1995), TOOD : Task Object Oriented Description for Ergonomic Interfaces Specification. IFAC/IFIP/IFOR/IEA SYMPOSIUM on Analysis, Design and Evaluation of Man-Machine Systems at MIT-Combridge, MA USA, June 27-29.
10. Mahfoudhi, A (1997). TOOD: Une méthodologie de description orientée objet des tâches utilisateur pour la spécification et la conception des interfaces homme-machine. PhD dissertation, University of Valenciennes, France.
11. Mahfoudhi A., Abed M., Tabary D., (2001) From the formal specification of users tasks to the automatic generation of the specification of MMI. IHM-HCI 2001 : Interaction without borders. 10-14 September 2001 Lille, France.
12. Nielsen J. & Molich R. (1990), Heuristic evaluation of user interfaces. Proceedings CHI'90, Seatle, ACM New Yoek, pp. 349-356.
13. Norman, D. & Draper, (1986), User centered system design, Lawrence Erlbaum Associates, Publishers.
14. Palanque, P. Spécifications formelles et systèmes interactifs, Habilitation à diriger des recherches, university of Toulouse I, France (1997)
15. Palanque, P., Bastide, R & Paterno, F. (1997), Formal specification as a tool for objective assessment of safety-critical interactive systems. In proceedings of the IFIP TC13 conference on HCI, Interact'97, pp 323-330. Sydney.
16. Paludetto, M. & Benzina, A. (1997), Une méthodologie orientée objet HOOD et réseaux de Petri. : Concepts et outils pour les systèmes de production in J-C. Hennet (ed.) Cépadués, p293-325.
17. Pinheiro da Silva, P., Paton, N.W. (2000). UMLi : The Unified Modeling Language for Interactive applications. 3rd Int. Conference on the Unified Modeling Language. LNCS, 1938, pp. 117-131.
18. Scapin, D.L. & Pierret-Golbreich, C. (1990), Towards a method for task description: MAD. Work with Display Unit'89 in Berlinguet, L & Berthelette, D. (Eds.) CAP.
19. Sibertin-Blanc, C. High-level Petri nets with Data Structure. 6th EWPNA, Espoo (Finland), (1985)
20. Tabary D., Abed M., Mahfoudhi A. (2001). Towards a design based on the User Task of human interfaces in cockpit. PIC'2001: PEOPLE IN CONTROL - An international conference on human interfaces in control rooms, cockpits and command centers. UMIST, Manchester, UK: 19 – 21 June 2001
21. Tarby, J-C & Barthet, M-F. (1996), The Diane+ Method in CADUI'96, pp95-119.
22. Ricard, E. & Buisine, A. (1996), Des tâches utilisateur au dialogue homme-machine: GLADIS++, une démarche industrielle. IHM96, pp71-76.
23. Vanderdonckt J. (1994), Guide ergonomique de la présentation des applications hautement interactives. Press universitaire de Namur.
24. Vanderdonckt, J. (1997), Conception assistée de la présentation d'une interface homme-machine ergonomique pour une application de gestion hautement interactive. Thèse, Faculté Notre Dame de la Paix Louvain, Belgique.