Proceedings of the IASTED International Conference
HUMAN-COMPUTER INTERACTION
November 14-16, 2005, Phoenix, AZ, USA

FROM TASK MODEL TO THE USER INTERFACE GENERATION: AN APPROACH BASED ON MULTI-AGENT SYSTEM

Adel Mahfoudhi[1], Jamel Slimi[1], Mourad Abed[2], Mohamed Abid[3]

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

[2] *LAMIH (UMR CNRS 8530) Université of ValenciennesBP : 311 – 59304 Valenciennes cedex9 (France)*
mourad.abed @univ-valenciennes.fr

[3] National Engineering School of Sfax Sfax (TUNISIA)
Mohamed.Abid@enis.rnu.tn

ABSTRACT
The User Interface (UI) plays a crucial role in the development of the interactive applications; its simplicity of use can sometimes be an important criteria of the application assessment. A good application that is represented by a non adequate interface can be judged as non successful one, and thus the utility of UI generator can guarantee a legible, intuitive and easy interface to manipulate by any users.
The multi agent systems are endowed with an interesting capacity liable to help the specialists of the UI to the conception and the development of the interactive systems. In order to take advantage of these domains, we proposed a method for the UI generation based on a multi agent model. Our approach is based on a set of rules assuring the passage from the task model to the multi agent model and from the latter toward the user interface generation.

KEY WORDS
Formal Method, Task Model, Multi-Agent Model, PAC Model, UI Generation, Petri nets.

1. 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 [1], DIANE [2], 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 formal methods truly. In fact the latter make it possible to provide the task models with conciseness, coherence and non-ambiguity [4]. 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 that of the detailed specification of the UI [5]: The methods MAD* [6], DIANE+ [7], GLADIS++ [8], ADEPT [9] Unified User Interface Design [10], OBSM [11] and TRIDENT [12] 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 [13] [14] is based on the representation the user has of the task, apart from the considerations of computer processing. Like the UML/PNO method [15], HOOD/PNO [16] and ICO [17]), 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 [18].
In this paper, we propose a set of rules allowing the user-interface generation based on multi-agents PAC architecture.
The integration of the **PAC** model in the **TOOD** methodology takes place via two passages: the first one from the task model to the **PAC** model and the second from the operational model to the **PAC** model. After the integration of the **PAC** model, we define the rules of the user-interfaces generation.

2. TOOD and the cycle of development of User Interface

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

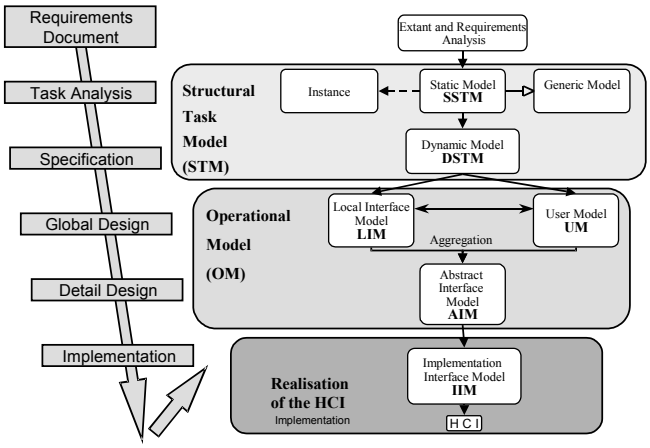

Figure 1. TOOD and the development cycle for the interface

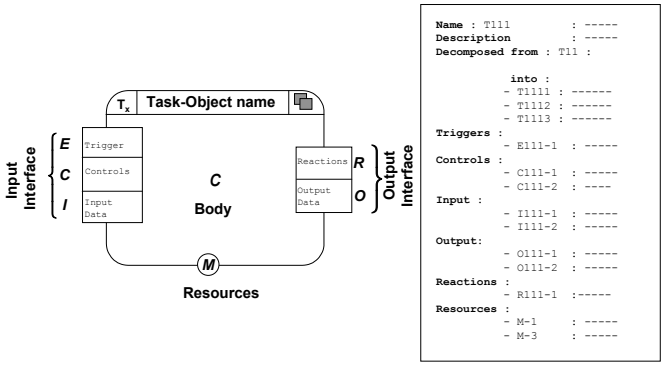

Figure 2. Generic structure of the class-task

- 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 to result in an Abstract Interface Model (**AIM**) 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, which is supported by the multi-agent software architecture defined in the Interface Implementation Model (**IIM**).

3. 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.

4. 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 in the future system, while avoiding the inconveniences of the existing system and adding the new required functions and features. So, 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:

1. Hierarchical decomposition of tasks.
2. Identification of objects and their components.
3. Definition of the dynamics of the elementary tasks (terminal task).
4. Integration of the task competition

4.1. 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 formalizes 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.

4.2. Dynamic Structural Task Model (DSTM)

The Dynamic Structural Task Model (**DSTM**) (figure 3) aims at integrating the temporal dimension (sequencing, synchronisation, concurrency, and interruption) by completing the static model.

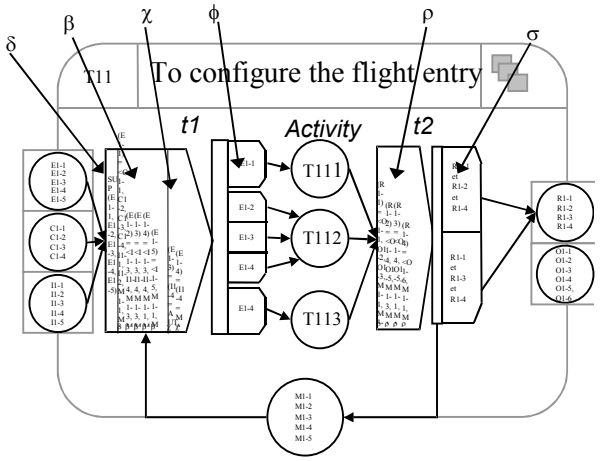

Figure 3. TCS : Task Control Structure

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 the 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 3).

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 related to an observed event, 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 uses simple, logical or mathematical type operators and obeys 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**. 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.

5. Operational Model (OM)

This stage has as an objective the automatic passage of the description of the user tasks 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.

In the TOOD method, the conception of the interactive systems is supported by the Operational Model (**OM**) that has, as objectives, the description of the user-interface to a high level of abstraction and the automation of the transition of the specification to the application conception [13].

The Operational Model (OM) is based on two steps:

- The UI composing specification for each terminal task supported by the User Model (**UM**) and the Interface Local Model (**ILM**).
- The global interface specification supported by the Interface Abstract Model (**IAM**).

The objective of the user's model is to understand and to formalize the user's behavior and therefore to establish a centered user interface conception.

The ILM describes the Interactive Objects (**IO**) behavior: the Object Control Structures (**ObCS**). The ObCS defines the dynamics of these IO in terms of states, offered service, and internal operation. These IO helps the user in the achievement of his task placing at his disposal a set of services. The ObCS is based on the Object Petri Net (OPN) formalism that facilitates their graphic representation (figure 4).

The **IAM** describes classes of user-interface objects. The construction of a user-interface object class suggests the aggregation of all **IO** in the same name, of the **ILM**. This mechanism of aggregation is comparable to the relation of composition of the HOOD method (Hierarchical Object Oriented Design).

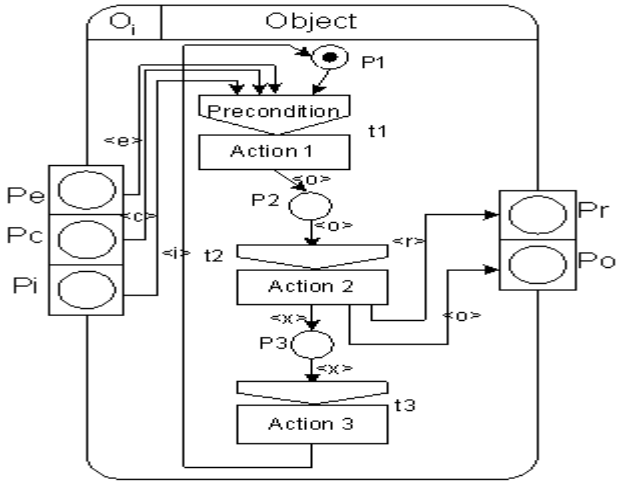

Figure 4. Interactive Object ObCS.

6. From the Task Model towards Multi-Agents Model

6.1. Multi-agents model: principle and objectives

Agent-based models structure an interactive system as a collection of specialised computational units called agents. An agent has a state, possesses an expertise, and is capable of initiating and reacting to events [19] [20] [21]. A system based on the multi-agent model is composed of a certain number of specialized agents communicating between each other and reacting to events which are considered as stimuli in order to produce stimuli for other agents [22].An agent is a complete information processing system, it includes event receivers and event transmitters, a memory to maintain a state, and a processor which cyclically processes input events, updates its own state, and may produce events or change its interest in input event classes. Agents communicate with other agents including the user.

The properties of modularity and reuse offered by the multi-agent models permit to reduce the complexity of the conception of the interactive systems, the transition toward the models of oriented object conception. The multi-agent models provide a support to the structuring and the organization of the dialogue [23]. The PAC model is software architecture model.

The PAC model structures an interactive system in three components: the Presentation, the Abstraction and the Control. PAC (Presentation, Abstraction, and Control): the facets of an agent are used to express different but complementary and strongly coupled computational perspectives. A PAC agent has a Presentation (i.e., its perceivable input and output behaviour), an Abstraction (i.e., its functional core), and a Control to express dependencies. The Control of an agent is in charge of communicating with other agents as well as expressing dependencies between the Abstract and Presentation facets of the agent. In the PAC style, no agent Abstraction is authorized to communicate directly with its corresponding Presentation and vice versa.

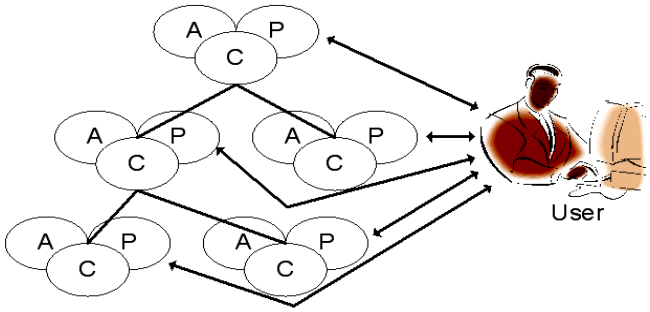

Figure 5. Structuring of an interactive system according to PAC.

In PAC model, dependencies of any sort are conveyed via Controls. Controls serve as the glue mechanism to express coordination and formalism transformations that lie between abstract and concrete perspectives [24].

PAC provides a setting of a systematic construction applicable to all the levels of an interactive system abstraction; it admits a straightforward separation between its components: Abstraction, Presentation and Control.

The passage from the task model to the multi-agent model is composed of two big parts: the first consists in integrating the PAC model in the TOOD methodology that is based on the model of the task and the operational model and the second consists in the passage from the agent's specification to the UI generation.

6.2. The integration of PAC agent in TOOD

The integration of the PAC model in **TOOD** will follow two stages: first it is a passage of the task hierarchy specifying the **UI** toward an agent's hierarchy, and second it is a passage of the interface **OM** and **ILM** toward the **PAC** model. These two passages are automated via a set of rules we will explain progressively.

We take as a basis, in these two passages, a set of generic rules allowing the **PAC** model to support the task model so that we can complete the phase of user-interface generation.

6.2.1 Construction of the agent's ramification

In the TOOD methodology, the specification of **UI** is based on the task model to represent the interface following a ramification of tasks going from the task root which includes the whole interactive system, to the control tasks to arrive at the terminal tasks. As a matter of fact the agent's **UI** specification will take the same structure as that of the task model. Indeed, we associate an agent root to the task root, control agents to the control tasks and terminals agents to the terminal tasks.

Rule 1*: to associate to the task root an agent root.*

Rule 2*: to associate to each control task a control agent.*

Rule 3*: to associate to each terminal task a terminal agent.*

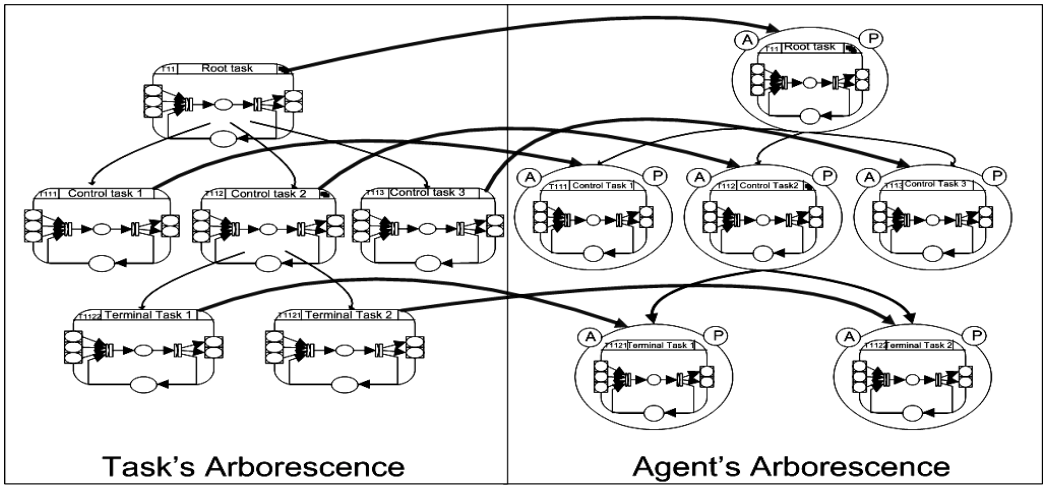

Figure 6. From the Task's ramification to agent's ramification.

To enrich the ramification of agents gotten while applying these three rules, we must take into account the interactive objects and the user's objects, which will be integrated in **PAC** ramification by a passage of the operational model toward the **PAC** model.

6.2.2. From the interface Operational Model and interface local model toward the PAC model

The objective of the **O**perational **M**odel is to describe the user-interface, formalize and automate the passage of the specification to the conception of the interactive systems. In the **I**nterface **L**ocal **M**odel, we have a description of the behavior of the interactive objects and a description of the interactions between these objects and the user. Every interactive object specifies the domain objects that assure the interaction between the user and the application. Thus, we associate to each interactive object an agent that supports it and to each objects of the domain an agent that supports it as well.

Rule 4*: to associate to each interactive object of the Interface Local Model an agent resource.*

Rule 5*: to associate to each object of the domain of the Domain Object Model a domain agent.*

After the application of these five first rules, we get a ramification of agent PAC supporting the user-interface of the interactive application. This ramification is composed of :

- -An agent root that will support the interactive application. It controls the set of the agents.
- The control's agents to organize the dialogue between the agent root and the terminal agents.
- The terminal agents that manipulate the interactive object agents.
- The objects user's agents and the interactive object agents.

6.2.3. Reduction of the agent's hierarchy

The systematic association of resource agents and the domain agents to the interactive objects and the domain objects, presents an inconvenience when it is dealing with the same interactive object used by two different agents; there is redundancy of agent. To surmount this problem, we are going to aggregate the two interactive objects agents to train only one agent (figure 7).

Rule 6*: To aggregate two identical interactive objects agents, if they are supported by two different terminal agents.*

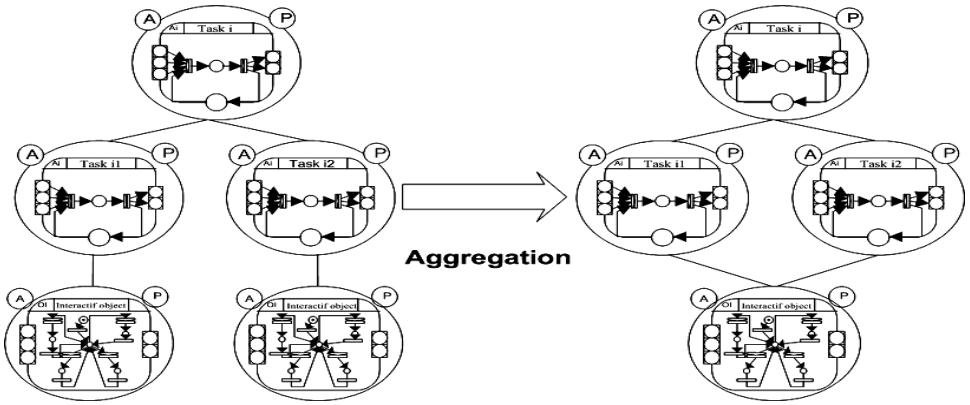

Figure 7. Agent's aggregation.

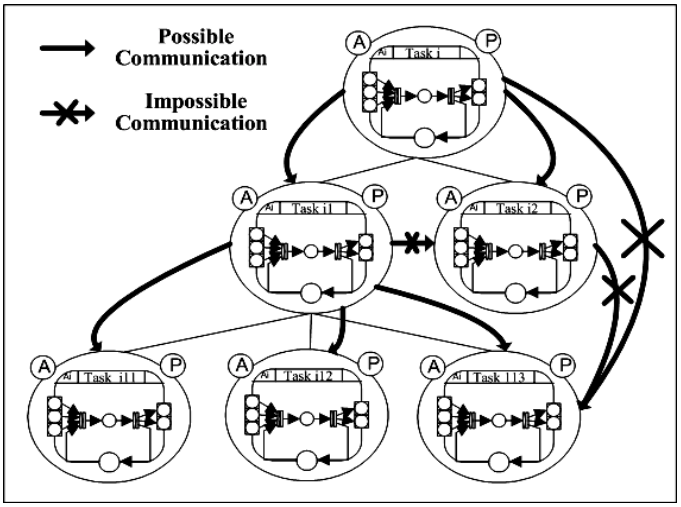

Figure 8. Agent's communication.

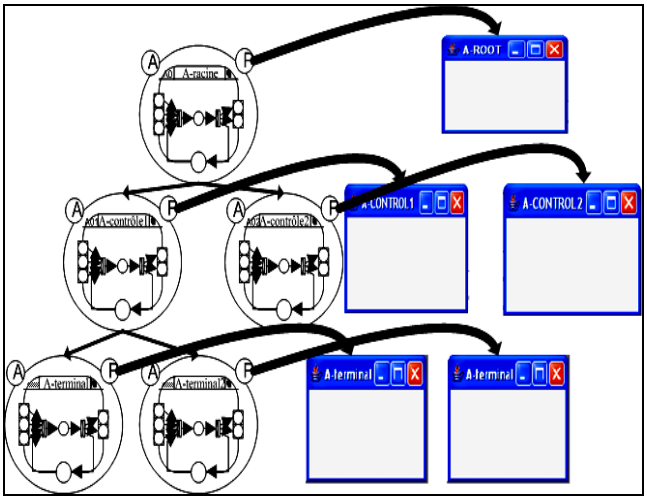

Figure 9. Affectation of the agent's frame.

6.3. Agent's communication

The agent's hierarchy expresses the potential that an agent has to supervise and to coordinate the activities of lower abstraction level agents, and in particular to assure the information consistency shared by itself and its subordinate agents. Indeed, this hierarchy has two direct consequences:

- An agent cannot be invoked by the agent to which it is the direct subordinate.
- The agents of dialogue coins treat the parts of their dialogues to their subordinate agents.

In our approach, this communication inter agents is based on the operational model.
The Object Controls Structure (**ObCS**) that describes an agent's internal and external behavior is represented by its control facet. Every agent's internal behavior is deducted by the execution of each object terminal task that will be achieved by the activity and the behavior of its interactive objects and user objects. Indeed, an interactive object agent can solicit the agent's facet presentation that supports it through the internal fluxes and can solicit the facet presentation of its subordinate agent through the external fluxes.
The establishments of the agent's hierarchy and the communication between its components have as objective, to generate the application **UI**.

7. User interface generation

The concept of interface generator is based on the agent's behavior specification to create an interactive system. The generator has as objectives, to facilitate the development and identify the interface reusable elements.
We will complete our procedure by a set of rules which has already assured the interface generation from agent's ramification.

In the first place, we take on a frame to each presentation facet of agent root, of control agents and terminal agents. Each frame will take the agent's name that supports it.
Rule 7*: To assign to an agent's facet presentation a frame carrying its name. The gent's resources and the agent's domain objects will not follow this rule since they have a very definite presentation (zone of seizure, list of choice, combo-box . . .).*

7.1. Second reduction of the agent's hierarchy

The systematic assignation of a frame to every presentation facet generates an interface with an important frame number. Among these frame, we find those which do not have a meaningful role in the interface. As a matter of fact, we are obliged to merge the two agents that are at the origin of a useless frame provided that they are bound by a hierarchical tie.
In addition, we can solve this problem by a masking of the presentation facet of the most superior agent in the hierarchy.
Rule 8*: To merge the two agents that are at the origin of a useless frame.*
Rule 9*: To conceal the presentation facet of the most superior agent.*
The choice of one of the two rules is left to the designer's common sense. So far, we have a set of frame emptiness representing the presentation facets of the agent's roots, controls and terminals. It is up to the investment of the interactive objects in these frames to finish the application interface generation part.

7.2. Interface construction

Three techniques of the frame's components investment exist:

- The static investment that consists in positioning the components of the interface manually.
- The investment under constraints that positions the components of some in relation to the other.
- The implicit investment that encapsulates the components in containers, which will be positioned in the final interface.

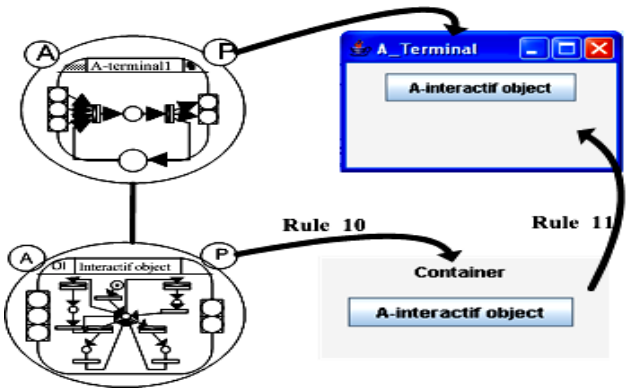

Figure 10. Application of the rules 10 and 11.

In our approach, we will use the implicit investment. Indeed, we will place the interactive objects in containers. Every container will carry the agent's resource name that supports it. Then these containers will be placed in the frame of the terminal agent presentation facet that supports these interactive objects (figure 10).

Rule 10*: to associate a container to the presentation facet of each agent interactive object.*
Rule 11*: to place the containers in the terminal agent's presentation facets.*

In the same way, the objects of the domain will be regrouped in the tab of the interactive object that manipulates them. We defined the facets of the presentation of the terminal agents. What is left is to define the presentation facets of the control agents. We can use the same logic-temporal relations (the sequence, the parallelism and the choice) applied in **TOOD** on the task model, in our agent model.
Rule 12*: if an agent supervises a set of agents that executes them in choice, we associate to the agent presentation facet a container that is under the shape beyond figure 11.*

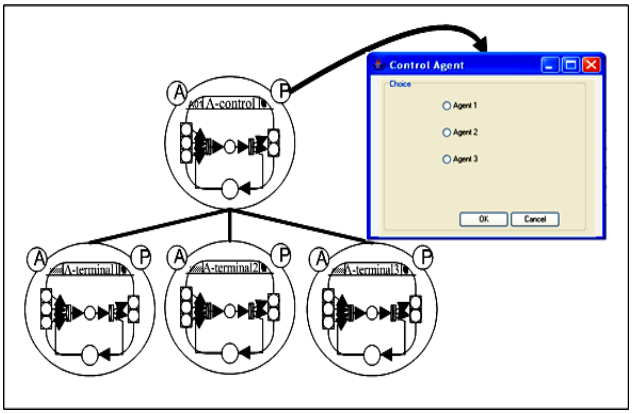

Figure 11. Presentation of control agent.

Rule 13*: if the agent supervises a set of agents that executes them in parallel or in sequence, we apply the rules 6 and 7.*

8. 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.
Moreover, the TOOD method can contribute towards helping with communication between the different actors in the design process through its formal description.
In the objective to generate the user interface, we have described in this paper a set of generic rules. These rules define our approach to integrate the agent model in the **TOOD** methodology. They have as objective to guide the passage of the task model toward the **PAC** model. This model will allow the user interface generation based on agents **PAC**.

References

1. Scapin, D.L. & Pierret-Golbreich, C., Towards a method for task description: MAD. *Work with Display Unit'89* in Berlinguet, L & Berthelette, D. (Eds.) CAP 1990.
2. Barthet, M.F., *Logiciels interactifs et ergonomie : modèles et méthodes de conception,* Dunod 1988.
3. Card, S.K, Moran, T.P, Newell A., *The Psychology of HCI*. In Lawrence Erlbaum Ass (Ed.). London. 1983.
4. Palanque, P. Spécifications formelles et systèmes interactifs, Habilitation à diriger des recherches, university of Toulouse I, France 1997.
5. Hartson, H.R., 1998. Human–computer interaction: interdisciplinary roots and trends. International Journal of Human–Computer Studies 43, 103–118.
6. Gamboa-Rodriguez, F., *Spécification et implémentation d'ALACIE: Atelier Logiciel d'Aide à la Conception d'Interfaces Ergonomiques.* Thèse en sciences: Université Paris XI 1998.
7. Tarby, J-C & Barthet, M-F., *The Diane+ Method* in CADUI'96, 1996 pp95-119.
8. Ricard, E. & Buisine, A., *Des tâches utilisateur au dialogue homme-machine: GLADIS++, une démarche industrielle.* IHM96, 1996 pp71-76.
9. Johnson, P., Johnson, H. Wilson, S., *Scenario-based design and task analysis*. In Carroll, J.M. (Ed.). Scenario-based design: Envisioning work and technology in system development. Willey 1995.
10. Anthony Savidis, Constantine Stephanidis. Unified user interface design: designing universally accessible interactions. Interacting with Computers 16 (2004) 243–270.

11. Kneer, B., Szwillus, G., 1995. OBSM: a notation to integrate different levels of user interface design, Proceedings of the ACM DIS'95 Symposium on Designing Interactive Systems, MI, USA, pp. 25–31.

12. Vanderdonckt, J., *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 1997.

13. Mahfoudhi.A, TOOD: une méthodologie de description orientée objet des tâches utilisateur pour la specification et la conception des interfaces hommes-machine, thèse en automatique humaine, University of Valencienne Hainaut-Combrésis, 1997(in french).

14. Adel Mahfoudhi, Mourad Abed, Dimitri Tabary. From the formal specifications of users tasks to the automatic generation of the HCI specifications. In: Blanford, A., Vanderdonckt, J., Gray, P. (Eds.), People and Computer XV—Interaction without Frontiers. Springer, Berlin 2001.

15. Delatour, J. & Paludeto, M., *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 1998.

16. Paludetto, M. & Benzina, A*., 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 1997.

17. Palanque, P., Bastide, R & Paterno, F., *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 1997.

18. Huhn Kim, Wan Chul Yoon. *Supporting the cognitive process of user interface design with reusable design cases.* Int. J. Human-Computer Studies 62 (2005) 457–486

19. Clavary.G, Coutaz.J, Nigay.L,Klemmer, *From Single-User Architectural Design to PAC*: a Generic Software Architecture Model for CSCWR*, Proceedings CHI, ACM Publi.,pp.242-249, 1997.

20. Hong Liua,, Mingxi Tang, John Hamilton Frazer : Supporting dynamic management in a multi-agent collaborative design system Advances in Engineering Software 35 (2004) 493–502

21. S.K. Lee, C.S. Hwang: Architecture layers and engineering approach for agent-based system. Information and Software Technology 45 (2003) 889–898

22. Buisine.A, *vers une démarche industrielle pour le dévelopment d'Interfaces Homme-Machine*, thesis, university of Rouen, 1999(in french).

23. Coutaz.J, *interface homme ordinateur: conception et realisation*, Dunod computer Ed, Paris, 1990 (in French).

24. Coutaz.J, Nigay.L, Salber.D, *Agent-Based Architecture Modelling for Interactive Systems*, published in critical issues in User Interface Engineering, pp 191-209, 1995.