Towards a new approach of model-based HCI Conception

ADEL MAHFOUDHI*,** — WIDED BOUCHELLIGUA** — MOURAD ABED*** — MOHAMED ABID**

* Department of Computer Science,
Science Faculty of Sfax
Rte Soukra km 3,5 BP : 802 -- 3018 Sfax
TUNISIA
** Computer, Electronic and Smart Engineering System Design Laboratory (CES)
National Engineering School of Sfax
Rte Soukra km 3,5 B.P. : w -- 3038 Sfax
TUNISIA
*** LAMIH (UMR CNRS 8530),
Université of Valenciennes.
BP : 311 – 59304 Valenciennes cedex9
FRANCE

Abstract: - This paper presents our contribution to the specification and conception of interactive systems. In this framework, the TOOD+ method (Task Oriented Object Design) proposed in this paper relies on a generic model and based for its description on the language of object modelization UML (Unified Modeling Language). The model-based approach and the used formalism have been chosen, to make a code multiple (C++, JAVA …) and multi platform (Palm, mobile telephone …).Our work is part and parcel of many others which are based on the principle of code generation from the specifications. Among some models, this generation corresponds to apply the MDE approach (Model Driven Engineering. This approach is presented on an example of interactive application (the air traffic control)

Key-Words: - Task model, HCI, formal method, UML, MDE, Interactive systems conception, Interactive systems specification.

1. Introduction

The appearance of the interactivity has developed an important and various amount of skills, techniques and tools coming in their majority from software engineering, ergonomy and cognitive psychology. Yet, it remains often difficult, in processing development, to conjointly use models founded on cognitive sciences and others on software engineering because their approaches of the problems are different.

In the traditional systems of computer science, one used to be interested mainly on the carrying out of the software functions. Besides, it was the user who had to provide the adaptation effort to the system. However, if that required effort is beyond the capacities and the user motivations, the realized system will never be used, even if its functionality and appearance are attractive. If the system functions are not of the nature of completing the user facility and their organisation does not reply to the metal structure of the resolution of the human problem, then none of the presentation effects will be able to hide these basic problems.

The cognitive sciences propose theories for studying human behaviour. The integration of these sciences with computer sciences is of great utility in the conception of HCI.

Added to that, the conception of interactive applications in different industrial sectors presents conceptual, technological and methodological problems.

For that reason, the HCI conception is considered, nowadays, a research domain that necessitates developments aiming at the resolution of these problems. The elaborated works in this research axis led to numerous tools, formalisms and methods ensuring a more or less complete cover of the development cycle of the interactive applications.

2. Previous works

In this context, many works were dedicated to user task modelisation, for example, works dedicated to the methods: MAD [17], DIANE [2], GOMS [7]. But the effective use of these tools is far from being a widespread practice. One of the possible reasons is the

lack of the use of really formal methods, which allow to bring conciseness and coherence to the task model. Moreover, these works lack integration in a global process of the conception covering the cycle set of the HCI development. In order to overcome these problems, the present research is oriented to a methodological framework which extends from the upper stage of the activity analysis to the detailed specification stage of the HCI. The methods MAD* [9], DIANE+ [2], GLADIS++ [16], ADEPT [10], TRIDENT [24] go in the same line of thought. In fact, they are based on several methods (task model, user model, interface model) and assisted by the implementation tools of these models.

Other works conducted since mid 1990s rely on the paradigm of the Model-Based user interface Design (MBD) [21]. It is a description of the application semantics and all the necessary knowledge of the specification.

This one described in a high-level specialized language, engenders a total or partial generation of the application code. The environments which are in favour of these approaches are called MB-IDE (Model Based – Interface Development Environment) [19].

Our work enrols in this orientation emphasizing on the formal aspect of the model representations and their transformation following the steps of the design process. In this project, we try to provide reply elements to the conception problems of the HCI. In this way, we have based our works on the MDA model (Model Driven Architecture) (OMG, 1999) and TOOD [12], [25].

3. TOOD+

The TOOD method [25] defines a generic model to the models set used to cover the development cycle. This solution is intended to avoid the information loss and the ambiguity engendered by the representation changes between the development steps. The generic model couples the formalisms of the object approach and the Object Petri Net (OPN). Nevertheless, in the view of automating the implementation and ameliorating the portability of interactive applications, our works are oriented to a platform independent specification to set up a multiple code generation (C++, JAVA, etc.) and multi-platform (Palm, mobile phone, etc.).

The TOOD+ model that we propose is inspired from the MDE approach. The main objective is to supply a generic model integrating the statistic and dynamic aspect for the set of the taken entities in the existing system analysis and the needs such as user tasks, the domain data manipulated by the tasks and the resources permitting the task realisation.

4. Development Cycle of the TOOD+ method

The TOOD+ method follows a methodological approach based on a transformation of a series of models covering the development process of an interactive system. Figure 1 gives an overall view of the development cycle of TOOD+ method.

Three models on the base of which TOOD+ is built :

- Task model (TM) and Domain Objects Model (DOM) in the specification stage ;
- An Operational Model (OP) in the design stage ;
- An Implementation Model (IM) in the realization stage.

The Domain Object Model (DOM) formalizes the system data according to static and dynamic parts.

These objects are introduced later in the Task Model in the form input and output data and resources in order to represent the task structure in a static model and its behavior in the dynamic model.

In the conception stage, the Operational Model (OM) expresses the link between the interface applications of the system. Thus this model describes the interactions between the interactive objects (system resource) and the user (human resource). These are respectively described in the Interface Local Model (ILM) and the User Model (UM).

To test TOOD+ methodology, we have introduced an example taken from the family of complex systems. It is concerned with the air traffic control carried out in the framework of the PHIDIAS project (Harmonious Position and Integrating the Interactive Dialogue) [13].

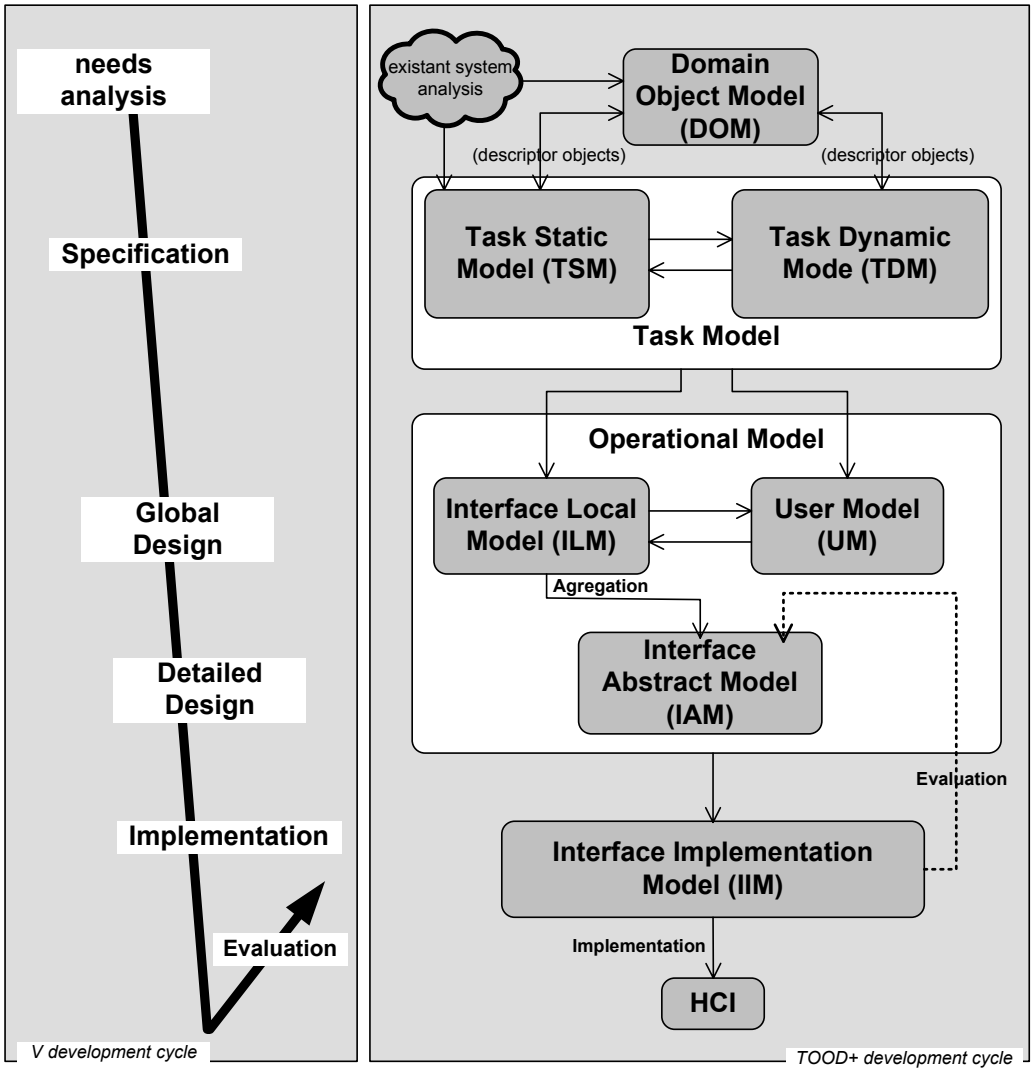

Fig.1 : Development Cycle of the TOOD+ method

We will present afterwards the specification stage of the interactive system which defines the Task Model (TM). Then, we will describe the Operational Model (OM) which depends on the construction of three models: the Interface Local Model (ILM), the User Model (UM) and the Interface Abstract Model (IAM).

5. Specification of the interactive system : Task Model (TM)

In TOOD+, the specification stage of the interactive system is based on the Task Model

The Task Model (TM) represents the task that the user must be able to carry out through the interface. This task is generally decomposed in a hierarchy of the task and sub-task.

In TOOD+, the Task Model is obtained from the decomposition of the user work to significant elements that we call tasks. Each task is characterized by an aim that the user wants to reach. Such a task can reach its objective through the execution of a procedure that defines a set of sub-tasks necessary to its accomplishment.

By applying the specialisation principle from the generic model, the Task Model is based on the class called Task-Class derived from the root class of all the entities manipulated by TOOD+, the TOOD+_Class. A task is then an instantiation of the Task_Class (see fig.2).

The particularity which characterizes the task entity regarding other taken entities is the fact that it necessitates a set of objects which supports its execution, called resource. This resource is represented by a component Class of Task class.

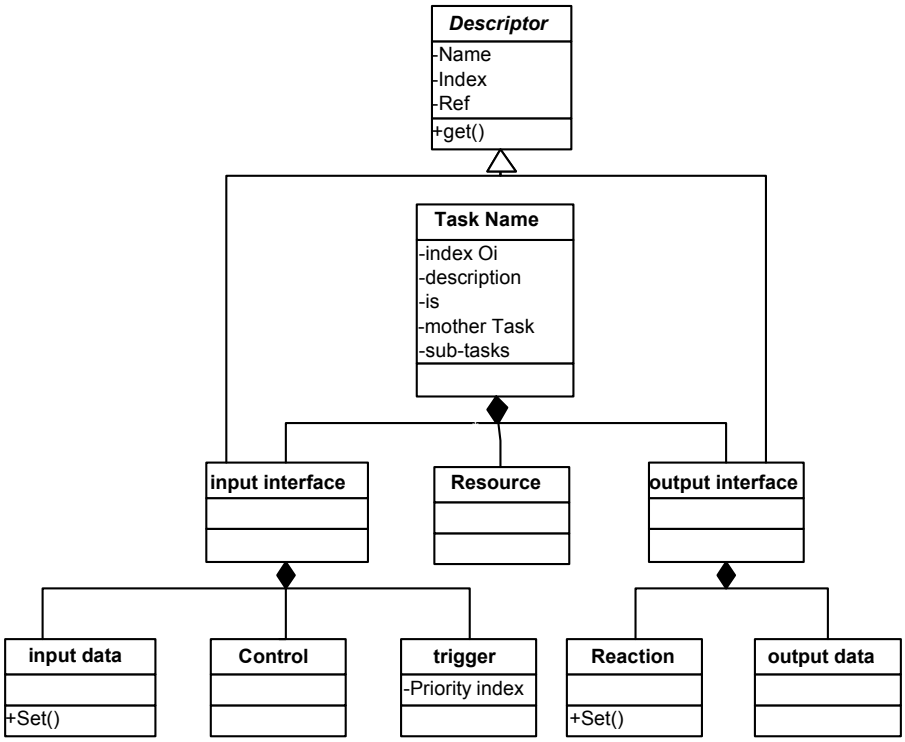

Fig.2 : Task-Class Structure and a Hierarchy of the Descriptor Objects Class

As figure.2 demonstrates, each task possesses:

- A name representing the treatment which is to be carried out, given in the form of a verb followed by a complement (for example : "Select an operation");
- An expressed description in a natural language explains the aim that the user wants to reach through the task ;
- A clue that identifies the task has the form Tij with i as the number of the mother task and I as a number among the daughter tasks.
- A type which specifies the nature of the task (manual, automatic, interactive or cooperative) ;
- A mother task also called control task ;
- A set of sub-tasks that contribute in the realisation of the specified tasks.

In the framework of the task modeling, the Input Interface is composed of the descriptive objects on which the task is realized. These objects specify the parameters of the task input. To model a task, we need to identify:

- The triggers : are the events which instigate the execution of the task. These events may be of two types :
  - formal or explicit producing observable tasks in the work environment (for example, information on screen) and considered as obligatory tasks ;
  - informal or implicit instigated on the decision of the user inciting the facultative task.
- The control data are the validated information at the beginning or during the task execution ;
- The input data are the necessary information of the task realization.

For the input Interface class, it is composed of descriptive objects resulting from the task realization. The execution of the task brings about set of reactions at a physical order capable of modifying the work environment or mental order contributing in the formation of new view of the situation by the user. The realization of a task also supplies transformed or created data.

A task then possesses a set of resources modeled by the Resource Class. These resources are classified in two categories; human and system resources (components of the application). The distribution of the resources allows to determine the task type.

The TM construction goes along the following processes :

1. to elaborate the diagram of the user case which picks up the system functionalities.
2. from the user case, to identify the class of the root or global task ;

3. to specify the descriptive objects and the resources of the global Class-Task ;
4. to decompose the Class-Task into sub Class-Task ;
5. for each sub-class, to specify the descriptive objects necessary for its accomplishment ;
6. to use the constructors of table. 1 in order to identify the relations inter-task ;
7. to recapitulate the stage 4 to 6 to identify all the final Class-Task.

Constructor	*Symbol*	*Description*
Sequence	---	The daughter tasks of the control task are executed in sequence.
Choice	+	The daughter tasks are executed without preferential order.
Parallelism	//	The daughter tasks are all executed at the same time.

Table 1 : Inter-tasks Relations

Most often, the final tasks correspond to the directly realizable operations by the application.

In our example of the control system of aerial traffic, the diagram of the user case comprises three user cases: traffic Control, traffic Planification and traffic Management. The user case traffic Control is linked to the other use cases by the user relation (« use »). The corresponding static model in our example, specifies an interactive root task "T0: to control the traffic", which itself is decomposed into two sub-tasks "T1: to plan the traffic" and "T2: to manage the traffic". These two tasks are carried out concurrently. The task "T1: to plan the traffic" is a control task and it is composed of three sub-tasks: the task "to help the radar controller" and is executed in parallel with the two sequential tasks "to configurate the flight entry" and "to configurate the flight exit" (see fig.3) etc.

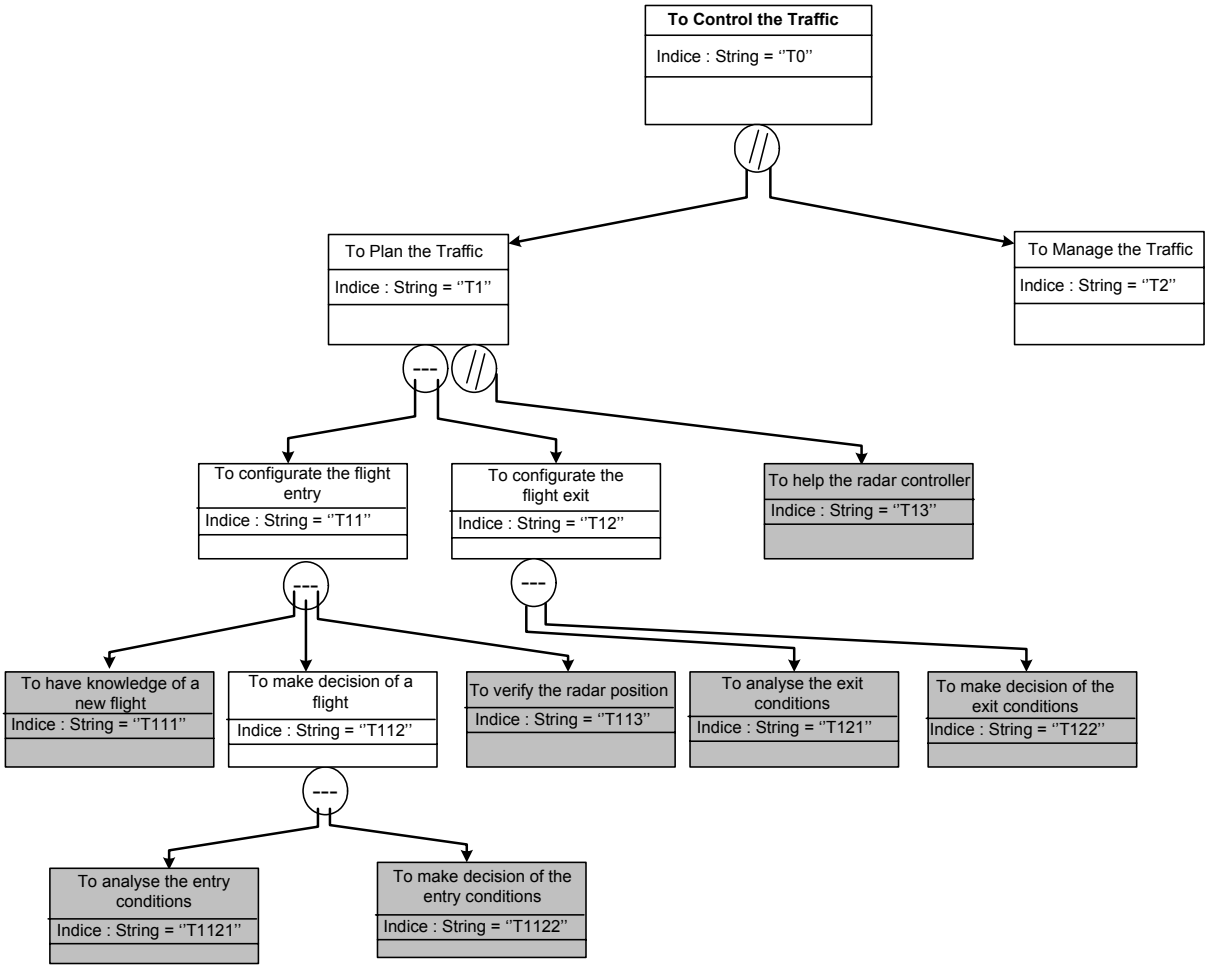

Fig.3 : Hierarchical decomposition of the task T_1 "to plan the traffic"

Once this decomposition is realized, the following stage is the resolution of the descriptive objects, and consequently, the identification of necessary domain objects for the accomplishment of each task.

6. Conception of the interactive system : Operational Model (OM)

The Operational Model allows to carry out the specification transition to the conception aiming at describing the user interface. The Operational Model consists in three models and it is based on two steps:

The Interface Local Model (ILM) to specify the HCI components (i.e. the interactive objects) for each final task relying on a User Model.

The Interface Abstract Model (IAM) whose components are aggregated from MLI.

Based on the same principle of generic model specification, these three models are presented in two parts : Static and dynamic parts.

6.1. User Model and Interface Local Model

6.1.1. User Model (UM)

The integration of the user in the interface conception allows, certainly, to produce interfaces of better quality. Through the User Model, TOOD+ formalizes the user behavior.

We will devote this paragraph to describe the dynamic behavior of this model facing a task. The User Dynamic Model is inspired fundamentally from the qualitative model of Rasmussen [15]. The decisional scale of the Model of Rasmussen consists of four steps: to detect, to evaluate, to decide, to execute. The user behavior either based on the rules to solve a current situation, or on the knowledge to solve a new situation. As figure 4 demonstrates, the states of the diagram State/Transition can be labeled by three states of operators: Perception (Lecture), Cognition (Evaluation) or Action. These states correspond to three chief sub-systems of human operators (visual, cognitive and motive). Thus, the transition models the action that the user comes to undertake in order to execute the final task in question, and which allows to develop an operator state or another.

6.1.2. Interface Local Model (ILM)

The Interface Local Model describes the behavior of the interactive object. These objects help the user in achieving the states/transitions UML defines the dynamic of these interactive objects. The states of the diagram represent the different states of the objects. The behavior of the interactive objects is defined by the chain of states and transitions. In fact, the transition is conditioned by the guard having the form Event [Condition]/Action, which can execute an action and generate the event. The field Event and Condition of the guard of the transition are formed from the descriptive objects identified for the final task (Task Static Model) to which the interactive object belongs.

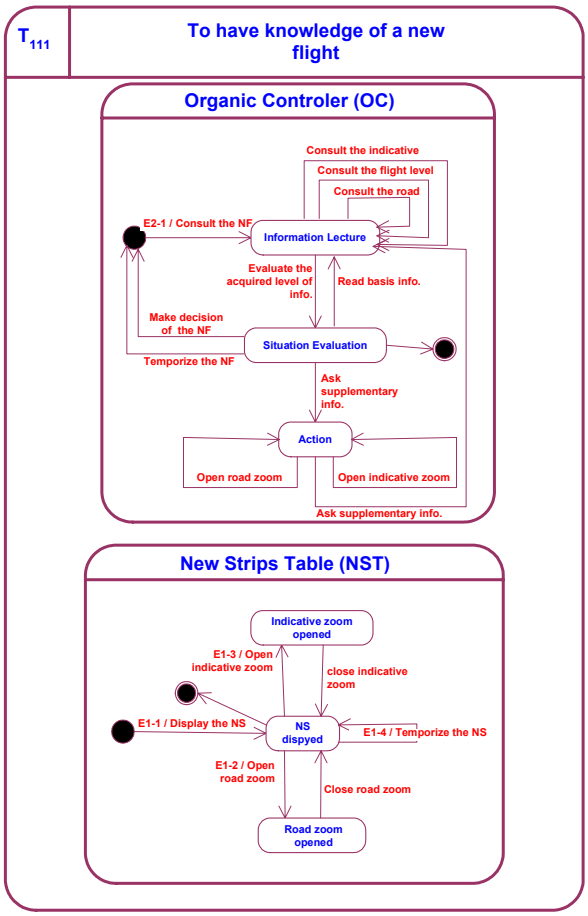

Fig.4 : Operational Model of final task object T_{111} : "to have knowledge of a new flight"

6.2. Interface Abstract Model (IAM)

The Interface Local Model is built by the specification of the interactive objects related to the final task independently from one another. Actually, the user interface, and therefore each interactive object, is not limited to a specific task or a particular transition. In order to do that, we should define the Interface Abstract Model which describes the HCI classes. The construction of an object class HCI suggests the aggregation of the interactive objects with the same name of Interface Local Model.

The procedure of aggregation is incremental, i.e. it takes the two first local models of the entry/input in order to produce an aggregated model which will be eventually integrated with a third local model of the interface etc…

7. The implementation of the HCI : Interface Implementation Model (IIM)

The implementation model of the Interface of TOOD methodology is the specification of low level abstraction and the presentation of the final interface as it will be seen by the user.

The construction of the Implementation Model (IM) is deduced from the task model and the operational model. In order to do that, a set of rules is established such as :

Rule 1 : to associate to each root and terminal task a window ;

Rule 2 : to hide the useless windows which correspond to control tasks (managing task found at a hierarchical level);

Rule 3 : the associated buttons to each window are issued from the user model (for each possible action by the user, we associate a button which holds the name of the action) ;

Rule 4 : the interactive objects which support each terminal task will be put in containers before being placed in window ;

Rule 5 : the user objects will equally be placed in the window associated to each terminal task. They will be grouped in the tab of the interactive object wich manipulate them;

Rule 6 : for the root task and the control task, we base our work on the inter-task relations (see table 1). In the case where we have a choice relation, we associate radio buttons which allows the selection of the task which is done. In the case where we have a sequencing relation, the window associated to a mother task will be useless and therefore it will be hidden. However, for a parallelism relation, the windows associated to daughter tasks are shown at the same time and the window associated to the mother task will be hidden.

An example of the application of these rules is given in the fig.5.

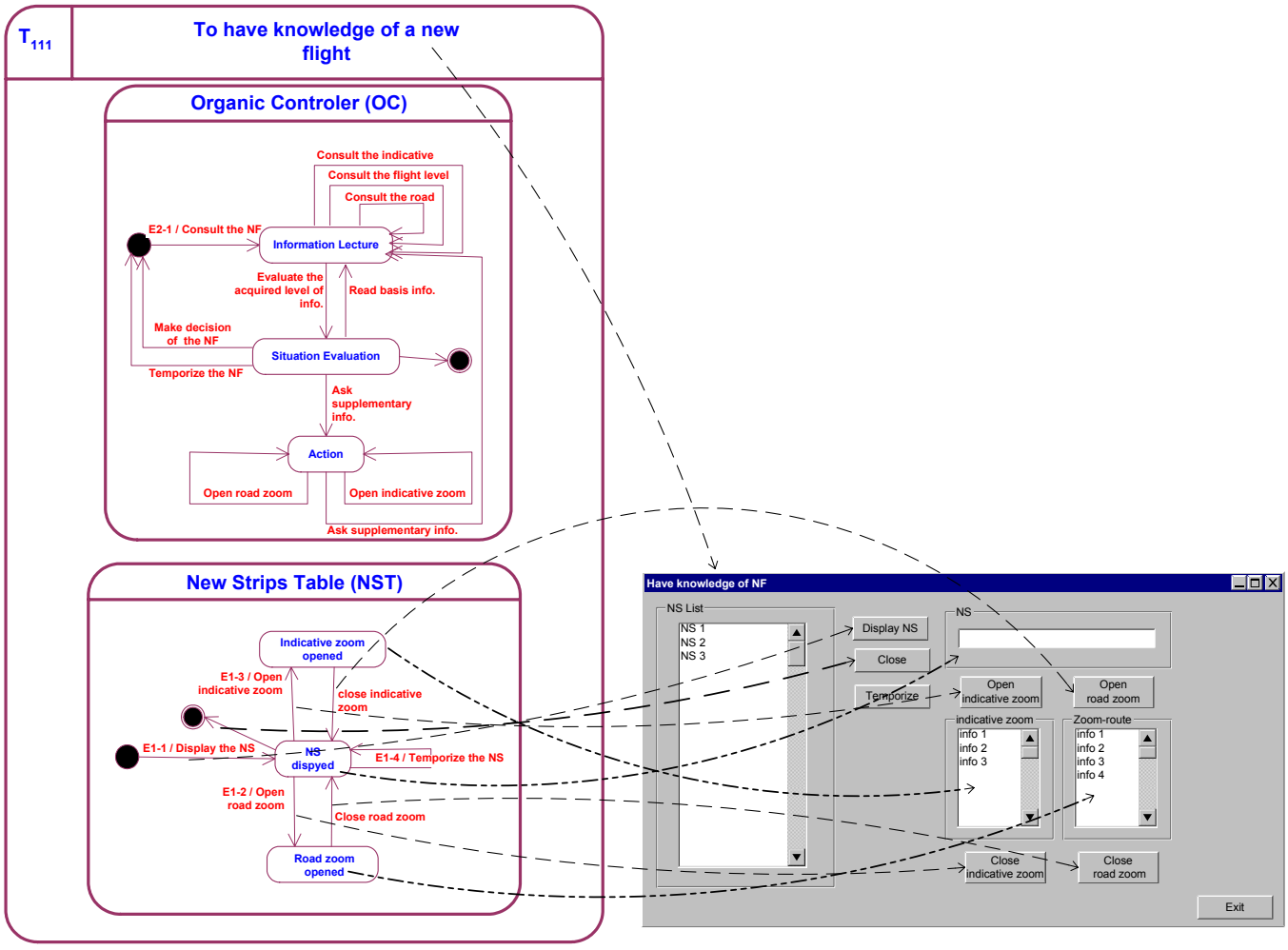

Fig.5 : The generation of the Implementation Model

The objective of this work is the amelioration of the specification model and the TOOD conception based on the new conception approaches and the development based on the models. For that reason, and in order to inspire the MDA approach, we have specified the behavior of the man-machine interaction from the UML specification. The following stage of the MDA defines a set of operations in order to make the models pass from the additional stage to the productive stage. Among these operations, we opt for the succession of the models which correspond to the passage of the graph from a model to a linear model. As a result, the XMI standard allows the representation of the UML models in the form of XML. In fact, XMI defines a set of rules which allows to build a DTD (Document Type Definition) from the UML model. Therefore, the operation of the concession allows to have a specification in a light language of the type XML. This type of specification allows to establish a generation of multiple code and multi-platform.

8. Conclusion

We have demonstrated in this article that the development of the Man-Machine Interface based on the models presents many advantages, especially, when we consider the plasticity of the HCI for which the change in platform is dynamic. Our crucial objective was the unification of model-directed works taken from engineering with those of Man-Machine Interface.

The stress on the correspondences between the models is not sufficient and does not mean here an operational vision. It is interesting to use a transformation language for modelling and implementing this transformation.

Moreover, the TOOD+ method can contribute to help the communication between the different interventions in the conception. The operational model leads to the specification then to HCI generation. This model is considered as a continuation of the structural model using the same formalisms, which privilege the semantic stability of TOOD+ method.

9. Bibliographie

[1] Atkinson C., T. Kühne: Model-Driven Development: A Metamodeling Foundation. IEEE Software. Septembre 2003.

[2] Barthet, M.F. (1988), Logiciels interactifs et ergonomie : modèles et méthodes de conception, Dunod.

[3] Bézivin J., M. Blay, M. Bouzeghoub, J. Estublier, and J.-M. Favre. Rapport de synthèse. Action spécifique CNRS sur l'Ingénierie Dirigée par les Modèles, janvier 2005.

[4] Bézivin J.: In Search of a Basic Principle for Model-Driven Engineering. Journal Novatica,Issus Spécial. Mars- Avril 2004.

[5] Bézivin J., X. Blanc . MDA : Vers un important changement de paradigme en génie logiciel. Développeur Référence – juillet 2002

[6] Bézivin J., O. Gerbé: Towards a Precise Definition of the OMG/MDA Framework. ASE'01 Automated Software Engineering, San Diego, USA, 26-29 Novembre, 2001.

[7] Card, S.K, Moran, T.P, Newell A. (1983), The Psychology of HCI. In Lawrence Erlbaum Ass (Ed.). London.

[8] Favre J.M., NGuyen T., Towards a Megamodel to Model Software Evolution Through Software Transformation, Workshop on Software Evolution through Transformation, SETRA 2004, Rome, Italy, Electronic Notes in Theoritical Computer Science, Volume 127, Issue 3, ENTCS ELSVIER , October 2, 2004.

[9] 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.

[10] 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.

[11] Kleppe A., Warmer J., Bast J.: MDA Explained-The Model Driven Architecture: Practice and Promise. Addison-Wesley, 2003.

[12] Mahfoudhi A., Abed M., Tabary D. From the Formal Specifications of Users Tasks to The Automatic Generation of the HCI Specifications. In People and Computer XV – Interaction without Frontiers. Blandfort A., Vanderdonckt J., Gray P. (eds) pp. 331-348. ISBN : 1-85233-515-7. Springer 2001.

[13] Mahfoudhi, A. (1997), TOOD : Une méthodologie de description orientée objet des tâches utilisateur pour la spécification et la conception des interfaces homme-machine. Thèse en automatique. Université de Valenciennes.

[14] OMG, 1999 Object Management Group : The Common Object request Broker : Architecture and Specification. CORBA IIOP 2.4.1 / Format/00-11-07, Framingham, MA.

[15] Rasmussen, J : The human as a system component. In Smith, H.T.Green, TRG (Eds.), J., Rouse, W.B. (Eds.), Human detection and diagnostic of system failures. Plenum Press, N.Y. 1980.

[16] Ricard, E. & Buisine, A. (1996), Des tâches utilisateur au dialogue homme-machine: GLADIS++, une démarche industrielle. IHM96, pp71-76.

[17] 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.

[18] Seidewitz E.: "What Models Mean". IEEE Software. Septembre 2003

[19] Selic B.. « The Pragmatics of Model-Driven Development ». IEEE Software, (5):19–25, 2003

[20] Sottet J-S., Calvary G., Favre J-M. « Ingénierie de l'interaction homme-machine dirigée par les modèles ». IDM'05 Premières Journées sur l'Ingénierie Dirigée par les Modèles, Paris 30 juin, 1 juillet 2005.

[21] Szekely, P. (1996). Retrospective and challenge for Model Based Interface Development. [CADU'96], Namur, pp.xxi-xliv.

[22] Tarby, J-C & Barthet, M-F. (1996), The Diane+ Method in CADUI'96, pp95-119.

[23] Terrase M., Savonnet M., E. Leclercq, T. Grison, G. Beker. « Points de vue croisées sur les notions de modèle et métamodèle ». IDM'05 Premières Journées sur l'Ingénierie Dirigée par les Modèles, Paris 30 juin, 1 juillet 2005.

[24] 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.

[25] Mahfoudhi A., Abed M., Abid M., *Towards a User Interface Generation Approach Based on Object Oriented Design and Task Model* TAMODIA'2005 : 4th International Workshop on TAsk MOdels and DIAgrams for user interface design For Work and Beyond Gdansk, Poland • September 26-27, 2005