A design methodology and a prototyping tool dedicate to adaptive interface generation

Sébastien Romitti, Charles Santoni, Philippe François

DIAM-IUSPIM
Domaine Universitaire de St Jérôme,
Avenue Escadrille Normandie-Niemen
13397 Marseille Cedex 20, France
pim264@vmesa12.u-3mrs.fr, {charles.santoni, philippe.francois}@iuspim.u-3mrs.fr

Abstract.
In this paper, the authors show a design methodology for adaptive interfaces. This methodology allows to completely specify, and then generate the interface. The specification consists of two stages : a stage of Petri net modelisation of the interface dialogue, from a static task modelisation under MAD or UAN, and a stage of specification of the presentation, using Abstract Interactive Objects. A Petri net interpreter uses the dialogue and the presentation specification in order to generate the interface. With those tools, development of adaptive interface becomes easier.

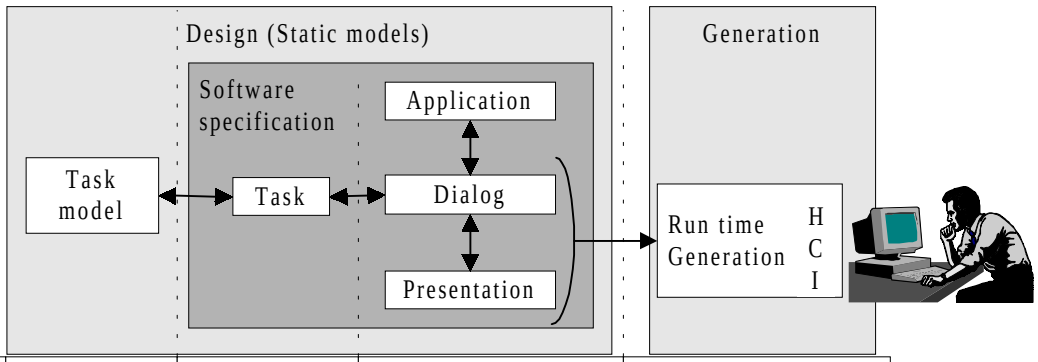
Keywords : design methodology, specification, Petri net, interface generation, adaptive interfaces.

1. INTRODUCTION

Our research project concerns the problem of interaction with complex systems, such as industrial process supervision systems. We define the notion of complex interactive system by a conjunction of certain factors such as : many tasks which can be performed from the interface, many possible states of the system, a lot of data handled, complex task scheduling, a rapid evolving system, and heterogeneity of potential users. From the users' point of view, this complexity generates usability problems. From the design team's point of view, ergonomists and computer scientists, this complexity generates modelisation and software specification difficulties.
Adaptive interfaces is a way to reduce usability complexity [Browne 90], [Shneider 92]. Those interfaces automatically adapt to each user, contrary to adaptable interfaces where the user has to adjust explicitly the interface. Adaptativity increases the quality of interaction, but it increases the cost and the time of conception. An adaptive interactive system has to easily change, and then, has to be strongly dynamical.
In order to reduce design complexity, our goal is to create tools and design methodologies, which allow to easily take adaptativity into account. These methodologies and tools have to integrate human factors and the contributions of cognitive science to human interface engineering [Romitti 97].
Our objective in this paper is not to present an adaptation process, but to present a new design methodology and a prototyping tool dedicate to adaptive interfaces.

2. GLOBAL DESCRIPTION

The work which is presented here concerns a methodology based on a formal specification. Figure 1 shows our design methodology. The first level is a software specification phase, issued from a task model. The second one is an automatic interfaces generation phase in run-time, through the interpretation of the dialog model.



Ergonomist	X		X	
Engineer			X	
Design tool		X	X	X

Figure 1 : The design methodology

The task modelisation uses UAN [Hartson 92] or MAD [Sebillotte 94] formalisms. The software specification consists of four phases where the dialog and the presentation components are functionally independent [Pfaff 85]:

- A phase of task model data reduction into Petri nets.
- A phase of Petri net modelisation of the dialog control, issued from the task model, where the interface states are identified.
- A phase of presentation specification, which uses the dialog model. Interaction objects are chosen in this phase.
- A phase of application modelisation, which is the functional core of the system.

This set of methodology, specification language and generation is a Model-Based user interface development tool [Szekely 96], that is dedicated to adaptive interfaces or complex systems design. It explicitly describes the function of each design actor (ergonomist, engineer and software tool), and takes care of the transition between each models, and of their compatibility.
Our methodology doesn't generate adaptive interfaces, but it is a new designing way for such interfaces. This is why there is no explicit user model. A user modelisation shell such as BGP-MS [Kobsa 93] should be easily plugged in with our generation tool.

3. DESIGN LEVEL

3.1. Task modelisation

In our methodology, the modelisation of user's task is the first phase of requirement definition. This phase should be processed by an ergonomist with MAD or UAN formalism. We distinguish task model of user on system and task model of user on interface, also called activity [Girard 96]. The model of user on system is a first step of task modelisation. Then, it must be translated into the task model of user on interface, by successive modifications, and by merging tasks into workspaces [Normand 92] [Santoni 95]. Figure 2 shows a merging example : different tasks such as making a phone call in a normal or an urgent situation start with the same task "pick up".

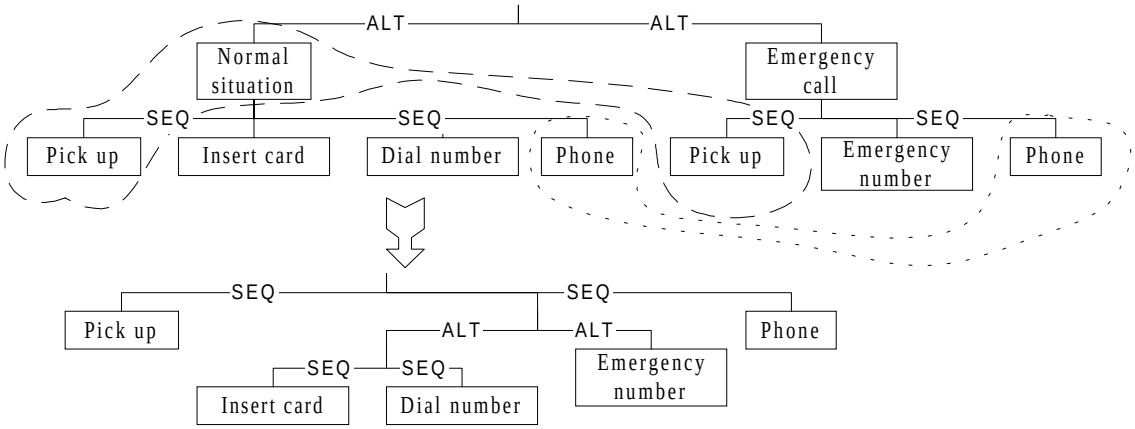


Figure 2 : Example of conversion from system task model to a interface task model.

3.2. Software specification

Formalism

Many specification formalisms for interactive systems already exists [Brun 95]. Characteristics of a formalism for interface generation are, on one hand, executability which allows automatic generation, and on the other hand, formal properties for analyses and verifications. Legibility and graphical representation are important for the usability of models. Frequently, formalism has to support concurrency and parallelism. Therefore, among usual specification formalisms, we choose Petri nets. They support extension such as the Cooperative Objects (C. O.), which extend their expression power [Bastide 92].

Software specification and task model

The task modelisation phase is a requirement analysis phase in which there is not any choice about interaction modality. We think that if task models describe a high abstraction level, and do not come down to the level of interaction objects, then a translation of those models into Petri nets such as described in [Palanque 95] is useful and pertinent. Such a translation lends homogeneity to software specification models. Petri nets which are resulting from this translation are ordinary.

Dialog model

The dialog model, issued from the task model, describes each possible state of the dialog. Each place of the Petri net represents a state of the dialog, while arcs and transitions represent the dynamics of the dialog. The dialog model is issued from the task model, by insertion of data useful to the dialog control as tokens, and of call-backs to functional core into transitions. Petri nets of dialog control are therefore object nets.

System model

The application modelisation phase allows to describe the process of the application. The formalism which is used to describe the dialog control and the presentation models is Petri net formalism. In order to dispose of compatible models, the formalism used to describe the application process is Petri net formalism too. The kind of Petri net used here depends on the domain of application.

Presentation

One of the characteristics of our approach lies in that presentation is issued from the dialog control model, so that in most current design tools such as the ICO (Interactive Co-operative Objects) [Palanque 92], presentation is a basis for dialog specification.
Besides, the presentation model has to specify interactive objects (widgets), when the presentation component of the ICO is only a static front panel. Among HCI classical specification formalisms, only UAN takes into account a description of presentation, nevertheless incomplete [Brun 95].
Our methodology allows (figure 3) to firstly define the semantic level (dialog), and to deduct from it the syntactic elements (interaction objects).
Therefore, the presentation model allows to provide information to the user, and to give him, through interaction objects, a way of changing the dialog model state. Different interaction objects can fit the same dialog model, and this opens the door to a form of adaptativity.

Figure 3 : interface design phases

Widgets are the atomic elements of our specification, and they are chosen by the ergonomist of the design team. A widget is a standard and classic one (a button), or complex and specific to the application (workspace, conspectus…).

4. SPECIFICATION OF THE PRESENTATION

Through lack of adapted formalism, we developed a specification of presentation, parallel with the dialog, but functionally independant of this model, in order to follow a Seehein's model recommendation [Pfaff 85].
At any time, the interface state is a conjunction of understates, which are running concurrently and independently. Petri nets allow to model such a system, considering that each place represents an understate. The whole active places (the marking) defines the state of the dialog at any time. The user creates a new state by firing a transition.
The exploration of the marking tree in the case of complex systems, in order to list the possible states, leads to a combinational explosion. Then, instead of generating the executable code of the interface, we use an interpretation of the dialog model. It implies that presentation has to be linked after each new state, thanks to a composition of Presentation Units related to active places.
Presentation Units are sets of Abstract Interactive Objects (AIO) [Sacre 96] that read as follows :

- A Presentation Unit is associated to one or several places.
- A place is connected to zero, one or several Presentation Units.
- A user action on an interactive object produces an event conveyed to a transition.
- The body of a transition may call methods of interactive objects.
- Instantiation and destruction of interactive objects are automatic.

Presentation units are represented as graphics in Petri nets (figure 4).

We now use synchronised Petri nets [Moalla 78], where an event is associated to transitions. Then, a transition is fired if it is validated (each place contains at least one token), and when its associated event is received. Then an event which is sent to a non-valid transition because of an out-of-context action from the user, will not modify the current dialog state.

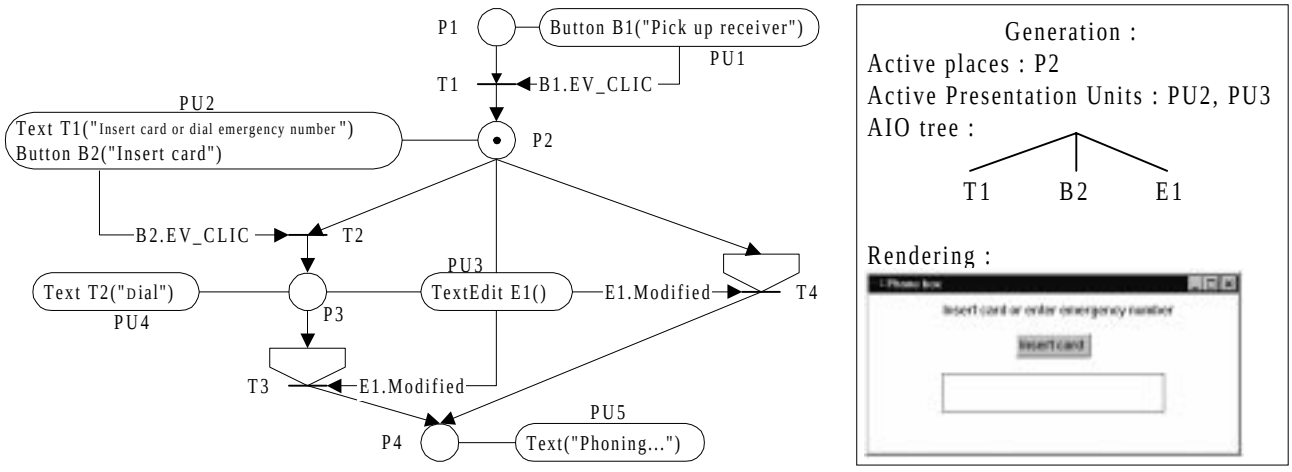


Figure 4 : An example of specification and of generation

5. GENERATION PHASE

The models of the software specification describe all of the interface. Then, the interface generation only interprets those models. A Petri net interpreter runs the dialog model, and lists the active places after each transition firing. The graphical objects that have to be displayed are the ones connected to, and the presentation model contains their attributes. To the right of figure 4 there is an example of generation for the marking {P2}.
Note that the interface is always context adapted, so that only the objects related to the current state are displayed.

7. CONCLUSION

This paper presents a design methodology with, as principal characteristic, continuity and homogeneity of models, from task model to interface generation. Transitions between models are described, and we define the place of each actor of the development (ergonomist and software engineer). With this methodology, the specification of the presentation is elaborated according to the dialog model.
In the absence of a complete interface specification model, we had to develop a specification of the presentation associated to a dialog model.
Once the interface is completely specified, then the automatic generation of an interface prototype is possible. This kind of generation offers a new way of designing adaptive interfaces, thanks to its very dynamical aspect.

BIBLIOGRAPHY

[Bastide 92] Bastide R. *Objets coopératifs : un formalisme pour la modélisation des systèmes concurrents*, Thèse, Toulouse III, 1992.[Browne 90] Browne D., Norman M., Totterdell P. (Eds) *Adaptive User Interfaces*., 1990 Academic press ISBN 0-12-137755-5

[Browne 90] Browne D., Norman M., Totterdell P. (Eds) *Adaptive User Interfaces*., 1990 Academic press ISBN 0-12-137755-5

[Brun 95] Brun P., Beaudouin-Lafon M., *A taxonomy and evaluation of formalisms for the specification of interactive systems*. Proceddings of HCI'95, people and computers X, Huddersfield, U.K., p. 197-212, Cambridge University Press, 1995.

[Girard 96] Girard P., Palanque P., *Compte-rendu de l'atelier Modélisation et conception logicielles et outils*, IHM'96, Cépadues, ISBN 2.85428.443.4

[Hartson 92] Rex Hartson H., Philip D. Gray, *Temporal Aspects of Tasks in the User Action Notation*, Human computer interaction,1992 V(7), p 1-45, Lawrence Erlbaum Associates

[Kobsa 93] Kobsa, A., Pohl, W., *The BGP-MS user modelling system*., User Modelling and user-adapted Interaction, 4(2), 59-106.

[Moalla 78] Moalla M., Pulou J., Sifakis J. Réseaux de Petri synchronisés. Revue RAIRO, vol 12, n°2, 1978.

[Palanque 92] Palanque P. *Modélisation par Objets Coopératifs Intéractifs (ICO) d'interfaces Homme-Machine dirigés par l'utilisateur*, Thèse, Université de Toulouse I, 1992

[Palanque 95] Palanque P., Bastide R., Senges V., *Task model - system model : towards an unifying formalism* Proceedings of HCI International conference, Yokoohama, Japan., 9-14 July 1995, p. 489-494, Elsevier.

[Pfaff 85] Pfaff G. E. (Ed), *Eurographic Seminar ; Tutorial and Perspectives in Computer Graphics ; User Interface Management Systems* ; In proceeedings of the workshop on UIMS held in Seehein FRG, springer-Verlag 1985

[Romitti 97] Romitti S., Santoni C., Francois P., *The scope of cognitive sciences in human-computer interaction*, ME-SELA'97, Loughborough University, UK, 22-24 July 97

[Sacre 96] Sacre B., Provot-sacre I, Vanderdonckt J., *Une description orientée objet des objets interactifs abstraits utilisés dans les Interfaces Homme-Machine* , 92-96, Rapport de recherche Projet TRIDENT, URL : http ://www.info.fundp.ac.be/~jvd

[Santoni 95] Santoni C., François P., Furtado E., Romitti S., *A generator of Adaptive User Interface Oriented Task*, Human Interaction with complex systems : conceptual principles and Design Practice, Kluwer Academic Publisher.

[Szekely 96] Szekely P., *Retrospective and Challenges for Model-Based Interface Development*, CADUI'96, Namur, 5-7 June 1996 J. Vanderdonckt (ed.) Collection "Travaux de l'Institut d'Informatique" n°15 Presses Universitaires

[Sebillotte 94] Sebillotte S. *Note de recherche concernant le formalisme MAD*. INRIA Note de recherche, 1994

[Shneider 92] Shneider-Hufschmidt M., Kühme T. & Malinwski U. *Adaptive user interfaces : principles and practice*, Human factors in technology, ISBN 0-444-81545-7

[Normand 92] Normand V. *Le modèle SIROCO : de la spécification conceptuelle des interfaces utilisateur à leur réalisation*. Thèse de l'université Joseph Fourrier Grenoble 1992