# A GENERIC MODEL FOR DIALOG SPECIFICATION

**Laurence ROUILLE, Patrick BOSC, Alain CHAUFFAUT**

Campus Universitaire de Beaulieu
35042 RENNES Cédex - FRANCE
Telephone : 99 36 20 00 (p. 495) - Telex : 950473F
Telecopy : 99383832 - Network : chauffauirisa.irisa.fr

Abstract :

This article deals with the dialog specification for interactive applications. Most of existing models are based on transition diagrams, but they give too attention to implementation considerations. Therefore, we recommend a new model completely independent of any implementation problem. The dialog is broken down into a series of queries, responses, results. This generic model is the main tool for the dialog specification in EREDIA, a new integrated environment for the user interface development of interactive applications. EREDIA discerns three steps : design, which is the definition of the logical dialog, layout which is the translation of the logical dialog into the physical dialog dedicated to an access method and programming which is the integration of the application functions. EREDIA is a friendliness environment, it takes into account the graphical aspects of the generic model on bit-mapped workstations.

Keywords :
user interface, user interface management system, transition diagrams, graphical specification

# 1 Introduction.

An interactive application is an application which communicates directly with the user to obtain the data necessary to its execution. Two parts may be discerned as regards interactive application: the interface which ensures communication between the application and the user, and the functions of the application.

The interface serves as a showcase of the application. A favorable reaction on the user's part garantees success. Its design therefore occupies a preponderant position. It must be up to users' expectations.

An application may be used by several kinds of users, beginners or experts, in which case the interface must be suited to user's knowledge level, without the processing of the application having to change. In addition, the application may be set up on different types of equipment, thereby influencing its communication with the user. Therefore, it is better to provide several interfaces for the same application, whenever there is not standardization in the use of the application.

The design of an application is a pluridisciplinary undertaking, which involves psychology, ergonomy... Moreover, with the current tendency towards optimal exploitation of graphic possibilities with access terminals, consulting a layout specialist is advisable. Consequently, the functional design of the application and that of the user interface need distinguishable qualities and will not necessarily be implemented by the same persons.

All the above motives make it essential to distinguish between the development of the interface and that of the application functions. The present article primarily deals with the analysis of methodologies and the development of user interface for interactive applications. We then outline a new development environment, termed EREDIA, and based on a generic model for interface specification.

# 2 Analysis of the extant.

## 2.1 Towards dialog management systems.

With the technological improvements in the field of I/O, bit-mapped display, videotex, mouse,..., the interactivity between the application and the user has been made more concrete. Its becomes truly a dialog. Nowadays an interactive dialog is a series of basic exchanges constitued with three steps: the application makes a query, the user keys in a reply and then the application provides him with a result. The first interfaces often allowed applications to be independent of all management of the physical terminal, by fostering a degree of abstraction in communication by means of the notion of virtual terminal. But this first stage was not sufficient to ensure a well laid-out display. Management tools for elementary dialog exchanges display came into being. They dealt with the display of query, result, or user responses, by grid management, state editors... Today, another problem has come up the coherence in the series of dialog exchanges.

It is what dialog management systems try to come to grips with. Dialog management systems attempt to obtain a entire specification of the dialog. On the one hand, there is the syntax of I/O. One may distinguish two specification levels: concrete syntax (basic communication display) and abstracted syntax (distinguished by its application). On the other hand, there is a description of the communication dynamics between the application and the user. It is dialog management. From these specifications, dialog management systems generate the execution core and establish links between the interface and the application functions.

## 2.2 Presentation of some models for dialog specification.

The transition diagram were used very at a very early date in the specification of interfaces. We prefer diagrams because they facilate understanding with their graphical representation, which is not true of grammars or object-oriented

languages. A graphical representation is especially worthwhile if a specification model is to be usable by non-specialists of computer-science. We present some examples of models which seem us representative

### 2.2.1 Parnas

Parnas [5] was one of the first people who suggested a use for command language specification. These languages infer restricted dialog, the application query is always the same implicit query: "What is your command?". Interface specification problems are simplified. Nodes represent the application states. Arcs contain the user response and the application result. Through syntact analysis of the command, Parnas determines what terminal operation has to be run and what the updated state of the terminal is like. It is a preliminary definition of the use of diagrams for language specification. His model is too succinct, but it is a widespread reference for the topic.

### 2.2.2 Kieras and Polson

As a result of their work in the field of command languages, Kieras and Polson [4] have indicated that a semantic analysis of the user response is essential. Then they added conditions to transitions. The application result is defined by the user response, but also by the semantic interpretation which can be deduced. However, they do not recommend any method to achieve a rigorous and coherent analysis of the user response. They recommend a description of the theoretical functionning of the condition evaluation. Several conditions can be true at the same time. They test them one at a time until reaching an exit node. However, the dialog can end up in a dead-lock (no passable route to an ultimate state). Moreover, they added an essential element whenever more important dialogs must be specified: that is the sub-diagram. A sub-diagram can be integrated to several levels (condition, action, state). This notion allows them to structure the dialog. In particular, the sub-diagrams are used to carry out the concrete syntactic analysis of the dialog. They are integrated into the principal diagram, which represents the dynamics of the dialog and the abstracted syntax. In their model, the notion of application query is treated superficially. The nodes are more or less classified by the input mode which they expect from the user: keyboard entry, function key...

### 2.2.3 Wasserman

In the Wasserman model [6] this notion of query is explicit. Besides, he does focus solely on command languages, but also recommends a broader model. Wasserman uses the transition diagram to specify the dynamics of dialog exchanges. There is no specification of abstracted syntax; he directly favours specification of concrete syntax with a language for the application and control characters as regards user inputs. His model is oriented towards an immediately deduced specification production. The operator who uses his model must prove his competence in all fields involving interface design. His diagram definition is very different from the afore-mentioned. Nodes represent application ouputs, queries and results. Arcs contain user inputs and application actions. An action is a condition or any other way of processing the application. He introduces the notion of sub-diagram structuring the dialog, as well his model allows a series of ouputs, responses, actions to be built. The operator must deliberately use the model to specify a coherent dialog. For this purpose, the model does not warrant a breakdown of the application dialog in query, user response and application result.

## 2.3 Analysis of the presented models.

There is a development of the recommended models through an increasingly more subtle analysis of the dialog, which approaches the definition that has previously been given. However, they do not solve all the problems brought up in the first part of the introduction. We have infered the conclusion that for the same appli-

cation several interfaces must coexist according to hardware, or to the user's experience. By several interfaces, we mean the form which changes with regard to the physical display of the dialog or a mode more or less condensed according to the user. Even if the form of the dialog should change, the logic is always the same. In a dialog two points of view stand out: the logical dialog independent of any implementation problem, and the physical dialog specially dedicated to a material environment and a grade of users. The models that we have studied closely blend the design of the logical dialog and the appearance of the physical dialog. Only one physical dialog is adapted to a logical dialog. The whole specification must be revised if the display layout is to be changed. This separation is possible with the model of Kieras and Polson, but it does not seem to be one of their objectives. Moreover, this separation is particularly desirable, because a proper display can be ensured only by a specialist.

The recommended models are designed for an immediately deduced specification production. Within this framework, they do not attempt to reason at a sufficiently abstracted level to be used by non-computer scientists. That aside, the necessary abilities for the specification of an interface are varied and not exclusively computerized. The dialog breakdown in term of series of queries, responses, results, is never explicit. Nevertheless, this definition expands the model to thereby include the specification of any type of dialogs.

### 2.4 Our solution.

Our solution to these problems lies in a generic model, which describes the dialogs in terms of queries, responses, results, accessible by non-computer scientists. It suits the addition of specific informations the description of different aspects of the dialog, logical and physical. The basic model is independent of any implementation. We give a description of this model in the chapter III. Afterwards, we evoke in the chapter IV its use in the framework of EREDIA (Environnement de REalisation des DIAlogues) for general public interactive applications. EREDIA is a design for logical dialog completely independent of physical dialog. As a matter of fact, several physical dialogs may refer to a same logical dialog. Lastly, we finish with the perspectives that offered by our model faced with new workstations, including multiple windows, as well as new designation devices like the mouse, icons...

## 3 The generic model.

The model is concerned with the specification of a dialog composed of basic exchanges. It shows the dynamics process communication and the abstracted syntax of dialog. It allows a large position for the graphical representation of the specification.

### 3.1 An example of dialog.

We take the example of the interactive application (server) "leisures". This server offers the user the choice between several services : "cinema", "theater", "concert". The function of the service "cinema" is to inform him about the cinema, timetables, films. To that purpose, the user chooses among four criteria of selection: "cinema", "timetable", "film", "type". The combinations of these criterions lead to different responses. The combination "cinema" and "film" will give the list of the timetables of the film for the cinema, "film" and "timetable" all the theater where the film is showing at that time ... The combinations of the two criteria "film" and "type" lead to an illicit query. For this purpose, we cannot ask at one and the same time, information about a particular film and type of films. When the list of the responses is displayed to the user, he can interrupt at any time to obtain an abstract of the film, by giving the corresponding number. We do not detail the other services, "theater" and "concert".

## 3.2 The basic dialog exchange.

### 3.2.1 The questionnaire.

The application query may have different forms, it may be especially implicit as we have already seen in command languages. The application waits for a user request or data. When the dialog is guided by the application, it decides to recommend menus to him, or to query him. In some case, both can be combined. For instance (fig. 1), in a menu which gives the user the choice between several leisure activities, he may be asked to give the name of the theater at the same time as he gives his selection of a service. Therefore, to each request are associated data fields which can have possible initial values. A questionnaire is composed of a collection of requests. In particular, when the application puts a single query to the user, the user request is the choice of answering this query. Later on, we indicate that it is sometimes interesting to have at one's disposal a classification of questionnaires, and to attach a type to the questionnaire.

For instance, the first exchange asks the user to choose a type of leisure activities. The questionnaire is called "leisures", its type is "menu" and it has three requests: "cinema", "theater", "concert". Any one of these three requests needs a data field. In the second questionnaire the application asks the user for his criteria of selection. That questionnaire is called "criteria", its type is "grid" and it has two requests: "selection" which corresponds to giving the criteria and "return" which corresponds to a return to the first menu. In the first request are associated four data fields: "cinema", "timetable", "film", "type". We consider that they have no initial value. On the other hand, the user's option to interrupt an edition of the result to ask for an abstract of the film cannot be modelized by a questionnaire. We will come back to this problem subsequently.

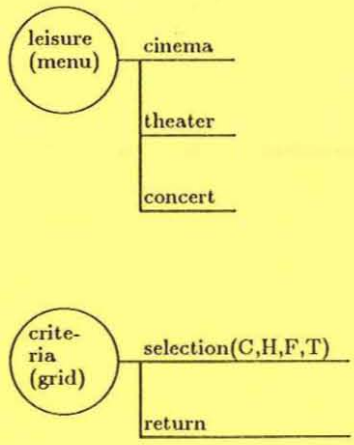


Figure 1 : The questionnaires

### 3.2.2 The interpretation of the user response.

The user response consists of a possible request with data. We need a semantic analysis of this response, whatever the type of the dialog considered. The model of Kieras and Polson has brought to the fore the risk of the dialog locking. We want to avoid this at all costs.

The semantic analysis depends on a context: request choice, user's previous data or permanent application data. This analysis can be more or less complex, that is why we have chosen to break it down into a combination of elementary conditions. This breakdown works in favour of a methodical analysis, but also assures us against a possible locking in the dialog. A elementary condition can be written under the form of a boolean function on data. So as to provide always a single true global condition, a solution is a binary tree of the interpretation. A boolean function gives back two values "true" and "false", these two possibilities must always appear in the analysis. Then, there is no locking in the dialog, the branch with non-conditions is a way out of this situation. On the other hand, at any given time just one interpretation is thrue, but this seems to us to reflect reality.

The solution that we come up with, seems suitable to us and not too impractical, as can be seen from its application to the example.

In the example, the choices of the requests of

the first questionnaire do not need particular interpretation. To this purpose, we do not ask the user for data and we do not consider that the requests depend on certain context, like the number of times the user has demanded this service during a session... On the other hand, the "selection" request of the "criteria" questionnaire needs an interpretation of data provided by the user, to determine what the new actionis and the pursuit of dialog. The user has the possibility of entering four data "cinema", "timetable", "film", "type". Nevertheless, the combination of the fields "film" and "type" leads to an illicit query. In this case, we must verify if this combination exists, and if the contrary were true, consider if there are responses to satisfy the request of the user. In this example (fig. 2), there are five possibilities, that we can translate into the conjunctions of following boolean functions :

film (F) and type(T)
film (F) and no (type(T)) and responses (C, H, F, T)
film (F) and no (type(T)) and no (responses (C, H, F, T))
no (film (F)) and responses (C, H, F, T)
no (film (F)) and no (responses (C, H, F, T))

So, we have the previously desirable binary tree, so that there is no locking in the functionning of the diagram.

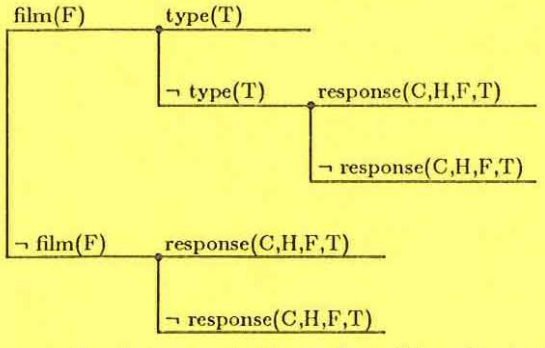


Figure 2 : Interpretation of multi-criteria in the "leisure" server

### 3.2.3 The result.

An interpretation of the response of the user leads to a result of the application. The result is composed of two parts: processing and edition. It is symbolized by a segment.

In the interpretations of the "selection" request, the results are the sending of warning messages such as "the query is illicit" or "there is no response", but also the display of a list of data, when there are responses.

## 3.3 The dialog.

### 3.3.1 The start questionnaire.

This is the starting point of the diagram, it corresponds to a single fictitious questionnaire. It is defined by an identifier and a request which corresponds to the user's wish to connect to the server. The request can have several interpretations, if there are several contexts of connection: proprieties linked to a user, connection constraints ... The start questionnaire is symbolized by a square.

### 3.3.2 Dialog pursual.

Once the application has answered the user request, the dialog goes on through another basic dialog exchange, or stops. It is a statically determined continuation of dialog. It is graphically modelized by the link of a result object with a state object.

We have added some dynamic dialog pursuits (fig. 3). We do not find them in the other models.

We have introduced the typed-return which is worthwhile when several paths in the diagram lead to a same state. If a return to an anterior questionnaire is desired, which depends on the path followed by the user, it is better to indicate just the type of the questionnaire. On execution of the last questionnaire put to the user, which has the type required by the typed-return, will be the new state. In the example we have given types to questionnaires: "menu", "grid"... but there is no restriction to the classification of the questionnaires. A typed-return is determined by the type of the questionnaire. It is symbolized on the diagram by a hexagon.

The second element we have introduced, is the event-salvage. The elements that we have given up to now partly allow us to deal with the system events. A system event can be treated as a particular request of the questionnaire or as an additional level of the interpretation of the user

response. A event salvage enables taking into account a system event during the display of a result.

A event-salvage is defined by an ident and by the set of events. An event is defined as a request. The event-salvage is symbolized by an oval.

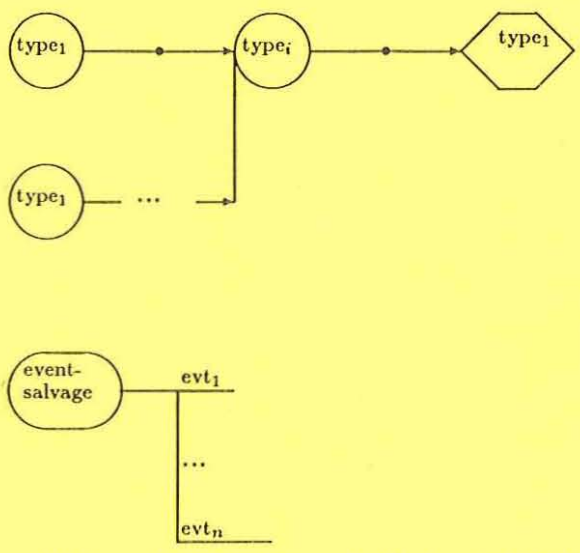


Figure 3 : The dynamic dialog pursual

### 3.3.3 The final mark.

This means the end of the dialogs. It can be duplicated on the diagram. It is characterized by an identifier and symbolized by a triangle.

### 3.3.4 The sub-dialog.

This is a concept that we find in all models. Beyond their methodical aspect, the sub-dialogs by allowing concision in writing contribute to an improvement in the legibility of the diagram.

A sub-dialog can be composed of all the elements of the model; it can especially call another sub-dialog. The difference between a principal dialog and a sub-dialog is that for the latter there may be several output contexts. The final marks possess moreover semantic information. A sub-dialog is a pseudo-questionnaire at the level of principal dialog. That's why it has requests (or pseudo-requests) which are its final marks. They can be interpreted according to the context of the principal dialog. A sub-dialog is symbolized by a rectangle at the level of a diagram.

In the example (fig. 4), we can assume that there are three sub-dialogs : "cinema", "theater" and "concert". There are two possible outputs for each sub-dialog, we can come back to the first questionnaire to choose a new service or stop the dialog.

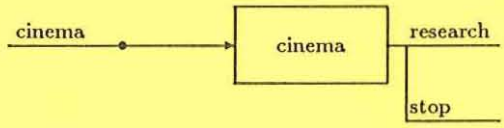


Figure 4 : The integrated "cinema" sub-dialog in the principal dialog

### 3.3.5 A particular result : intervention.

Intervention is an original element in our model. It is a particular result, used for editions liable to be long. In this case, the application stays open to the user. The long editions can be cut into elementary results. That cutting up serves as a landmark for the application, which is informed of a possible intervention of the user as soon as the latter receives an elementary result. If the user does not intervene, the next result is displayed. In this particular case, the user has a condensed version of the results. Any intervention of the user except a display stop request is a digression of the edition (a sub-dialog). At the output of this sub-dialog, the iteration of the edition goes on, or can be abandonned according to different contexts. In an intervention, the iteration is not disturbed at each edition of an elementary result to put a query explicitly to the user. The user knows the requests which are possible. During an intervention, it is the user who has the initiative of putting a request. Intervention is a possibility in the dialog, which is not already feasible to implement. It depends on the access method available.

User interventions are definyed in a manner similar to the requests of a questionnaire. In all cases, there is a standard request which is "non-intervention". We associate a cutting up of the edition with an intervention. The intervention is modelized by a rhombus.

In the example (fig. 5), the intervention has, as well as the standard request the request "abstract" which has a data field, the number of the film "number", and a request "stop".

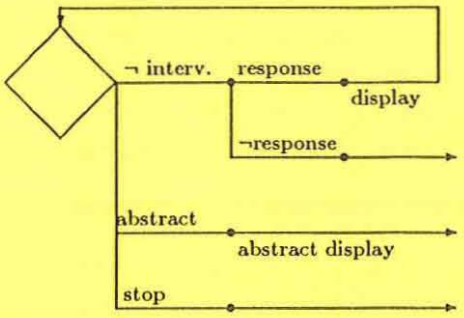

Figure 5 : The intervention

### 3.4 Conclusion.

Our model gives the possibility of specifying all dialogs in term of query, response, result. It does not refer to any computerized pre-requisite and so it is easily accessible to non-computer experts. Moreover, it is independent of any problem of implementation, presentation or programming.

Our model is an extension of transition diagrams, representation of finite state machines. We have modified the structure, but nevertheless there are analogies. In the transition diagrams, a node represents a state of the machine and a transition allows the passage from one state to another, it has an input and an output. For us, it is not a question of considering the states of the machine any more, however we can recognize some points in the dialog as being steps in the dialog user. At these steps, the user has the possibility of intervening, or these are radical changements in the dialog. In our model, we have assimilated some elements as being states of dialog. These are the questionnaires, the interventions, the sub-dialogs, but also the start questionnaire, the final marks, the event-salvages and the typed-returns. We can say that we have broken down the transition diagrams into three elements : the request, the interpretation and the result.

The dialogs which can be studied with our model are numerous and various; dialogs of query/response, command languages and why not object-oriented dialogs. These last types of dialogs are user initiative. This is a capital feature which may modify the design of the interface. The user has at a given time a lot of possibilities, because he can direct of the dialog. The notion of logic in the series of exchanges is not always perceptible, because the number of scenarios is too vaste. Our model seems less adapted to the specification of this type of dialog. We obtain very important diagrams, but it is not without interesting. As a matter of fact, the notion of series is still present; all the possible actions of the user are not continuously permitted.

The multiple windows allow parallelism at physical I/O device level and at the level of given services by the application, which then can be used simultaneously. We have not looked for this new possible aspect of the dialog. We think it is important to study the possibilities of our model to solve this problem.

## 4 Uses of the model.

### 4.1 The uses in EREDIA.

We now consider the use of the model in the framework of EREDIA, [1], [2]. EREDIA recommend to develop an interactive application around the definition of its dialogs with the user. Therein, EREDIA discerns three phases in the development of an interactive application: design of the logical dialog, layout(definition of the real dialog) and the programming. These three steps of the development are entrusted to experts: the designer, the model maker, and the programmer. Although all three work on the same dialog, but they have not the same perception, because they have not the same interest. Later on, we going indicate how the same model can be used for the specification of dialogs for the three operators. The model is a core. Supplementary specifications must be added to elements, to specialize the model in the description of a particular aspect of the dialog. Having a basic model is interesting: it allows a communication between the different operators, who have then the same references to work from.

### 4.2 The conceptual schema.

The conceptual schema must account for the logical dialog between the application and the

user, without being preoccupied with any physical representation of the exchanges. It presents their semantics. It is the first of the three schemas which is established and it is therefore the reference through the development of the application.

Designing a conceptual schema reverts to finding what the exchanges between the application and the user are. It means indicating what the characteristics of the questionnaires are, suggesting a set of requests, putting queries, analysing the user responses and the application results. There is neither reference to the exchanges presentation, nor to the programming of the interpretations and application results. It is the approach which has been presented all through the model description with the presentation of examples.

## 4.3 The presentation schema.

From this conceptual schema, several schemas of presentation can be deduced, which integrate physical attentions on the dialog. There may be several representations, because there can be different types of available equipment or several kinds of users to satisfy and who do not expect the same characteristics from an interface.

Changing from the conceptual schema to a presentation schema consists of a translation of logical exchanges into physical exchanges accounting for the real dialog between the application and the user. Therefore, there is a modification of the structure of the conceptual schema, elements are added and the specifications of the elements are completed. This transformation is carried out in two parts. First, there is a translation of conceptual requests into physical requests. In the example, for the first questionnaire the user has the choice between three requests: "cinema", "theater", "concert". He can have access to these requests in different manners according to the choice of the model maker; by typing a number between 1 and 4 or the ident of the request, by using function keys, by clicking in a menu... If for a same function key, there are several values to determine a choice of request, (1 + envoi = cinema, 2 + envoi = theater...) supplementary interpretation levels are created. Moreover, the choice of an access method (terminal management) adds some secondary exchanges to the principal dialog described by the designer. For instance, in the case of videotex access methods, every function key must be affected to at least one action.

In addition, a presentation format must be joined to every questionnaire and to every result. The model such as it is defined does not integrate elements of presentation specification. According to the possible complexity of the presentation it is certainly desirable to dispose of a specification language, which gives a direct view. For instance, in EREDIA there is the LCF (Langage de Composition de Formats) to specify videotex pages [3].

## 4.4 The programming schema.

For each presentation schema corresponds a programming schema. It is an extension of the presentation schema to which we add routines to elements, state routine, interpretation routine, result routine. The programming of the core of the application is deduced from the programming schema. The programmer must give in addition subroutines for the functions of the application.

## 4.5 The help of EREDIA in the specification

EREDIA recommends a set of integrated tools and methodologies for the development of an interactive application. The friendliness in EREDIA shows itself through several choices: bitmap workstations, a graphical specification of dialog by "graphical echo", dialogs with the operator through a set of adapted menus according to the operator's work... The graphical echo is a new concept totally different from computer aided drawing tools. The operator builds the interface in the concrete terms of dialog: question, response, result. EREDIA gives him back a graphical echo of his specifications in the form of a diagram. The graphical echo has several advantages. It preseves in all cases a lisible

drawing; the elements are placed according to rules. It allows for the interactive verification of some of the consistency properties of the diagram. EREDIA runs a checking process and can lead the specifications of the operator, by refusing him incorrect choices. Lastly, for a same dialog between EREDIA and the operator, we can have several forms of echos adapted to the operator.

### 4.6 Conclusion.

We have presented an use of our model in the framework of the EREDIA environment. We develop a prototype of EREDIA on SPS7/BULL with a bit-mapped workstation. But as we have just shown the model is sufficiently broad to be integrated to another context of use. We simply have to change the textual specifications linked to each element.

## 5 Conclusion.

The separation between the user interface and the functions of the application is one of essential guidelines for the design of interactive application. We have particulary focused on the problem of the user interface development. An interactive dialog is a series of basic exchanges, which consists of three components: the application query, the user response and the application result. The interfaces management systems fit together to give a coherence in the series of exchanges. The basis to put this guideline into practice is to dispose of a specification model of the communication dynamics.

First, the model has to be understood and used by non computer experts. This is the principal motivation that made us choose a graphical representation with our extension of the transition diagrams. The originality of our model lies in the transitions represented by the tree-like arcs and the addition of new elements to extend the power of the diagrams, such as typed-return, intervention, event-salvage.

On the one hand, our definition of the dialog, (questionnaire, response-interpretation, result), is simple and does not refer to any computerized knowledge.

On the other hand, the model must allow for the specification of different viewpoints of the dialog while preserving a coherence. In our opinion, it achieves its purpose being an generic model independent of any implementation problem, to which some elements of textual specification must be added to specialize it.

Up to now, the model has been principally used to specify query/response dialogs of videotex application in the framework of EREDIA. However, we think that its definition allows the more general specification of any dialog (query, response, result), in which the series of exchanges need a description. Its modularity makes it easilly adaptable for applications of which the presentation differs from this one for the videotex applications. Thus, it is the specification of I/O languages which evolves, but the design of logical dialog, is always the same.

## References


[1] Patrick BOSC, Alain CHAUFFAUT, and Bertrand HARDY. EREDIA : a system to develop servers including text processing technics. In *PROTEXT II (BOOLE PRESS LIMITED)*, pages 133–144, 2nd International Conference on Text Processing Systems, Dublin, 23-25 october 1985.

[2] Bruno CHERON and Laurence ROUILLE. *Etude et réalisation d'outils d'aide à la spécification graphique pour le concepteur et le programmeur de l'atelier EREDIA.* DEA Report, University of RENNES I, september 1985.

[3] Christian COUEPEL and Colette TANGUY. *EVA : Le Logiciel de Composition de Formats.* Technical Report 78, INRIA - RENNES, februar 1987.

[4] David KIERAS and Peter G. POLSON. A generalized transition network – representation for interactive systems. In *Proceedings of CHI'83, Human Factors in Computer Systems*, pages 103–106, 1983.

[5] David L. PARNAS. On the use of transition diagrams in the design of a user interface for an interactive computer system. *Proc. 24th National Conference ACM*, 379–383, 1969.

[6] Anthony I. WASSERMAN. Extending state transition diagrams for the specification of human–computer interaction. *IEEE Transaction on Software Engineering*, SE.11(8):699–712, august 1985.