The DIANE+ Method

Jean-Claude Tarby[1] and Marie-France Barthet[2]
[1]TRIGONE Laboratory, CUEEP Institute, Université Lille 1,
F-59655 Villeneuve d'Ascq Cedex, France
Phone: +33-20.43.32.62 – Fax: +33-20.43.32.79
E-mail: Jean-Claude.Tarby@univ-lille1.fr
WWW: http://www-trigone.univ-lille1.fr/jean_claude/Welcome.html
[2]Laboratory for Information Science, Université Toulouse I, Place Anatole
France, F-31042 Toulouse Cedex, France
Phone: +33-61.63.36.03 – Fax: +33-61.63.37.98
E-mail: barthet@cict.fr - WWW: http://lis.univ-tlse1.fr/~barthet

Abstract

The DIANE method has been created to solve malfunctions in the use of interactive software, leading to trouble in the information systems and difficulties in the user learning and memorisation. The DIANE method aims to integrate the user and his interaction capability into the current process of designing an interactive software. DIANE+ extends the DIANE method to make possible the automatic generation of user interface. This extension concerns the model of dialogue control, and the integration of an OPAC object data model extending the PAC model. This work is based upon a key concept: the control sharing between man and machine. Our approach complements the object methods by integrating aspects relating to tasks and work stations, and concepts such as the user's level and activity.

Keywords

User interface design, task analysis, computer-aided generation, automatic contextual help, automatic user interface management.

Introduction

With actual UIMSs, user-friendly interfaces can be created with greater decisional latitude[1], direct manipulation, prototyping facilities and code generation. These two last features can be executed from screen layouts or specifications of the application. On the other hand, UIMSs have a major default: they do not integrate ergonomics into the life cycle. These limitations occur at four distinct levels:

1. the user is not modelled in the application, so interactions are treated independently of him, and do not take into account his level of knowledge of the application (from beginner to expert);
2. UIMS do not have any specification method. They are used after specifications have been made;

[1] The decisional latitude is the user's freedom of action within the application.

3. human engineering is rarely integrated. In general, it is applied to specific cases;
4. the evaluation of the ergonomic aspects is impossible. The application can only be tested to see if it corresponds to the specifications.

These remarks depend basically on the application domain. We can distinguish three types of tasks: procedural (e.g., in information systems), expert (e.g., in knowledge based systems [Vogel88, Hickmann89, Brunet91]) or creative (e.g., in drawing applications). Our interest is in applications with procedural processes and decisional latitude, i.e., applications where user intentions are predefined. Our objective is to design and create a CASE tool which possesses the advantages of UIMS while reducing the ergonomics problem.

Task-oriented approach and object-oriented approach are both used in application development. The second approach was first used in implementation but current object-oriented methods show that it can be integrated in design and specification [Schlaer88, Bailin89, Colbert89, Coad90, Gibson90, Rumbaugh91]. This has been true for the task-oriented approach for several years, but the rising need of interaction revealed the limits of this approach. It has, however, proved itself and the human engineers know that it is easier to describe a job through tasks and goals rather than objects to manipulate [Sebillotte88, Sebillotte91].

Moreover, a job described through tasks and goals allows extracting and validating the cognitive user model more completely. So, the task-oriented approach is advantageous in the first phases of application development. It can be used, for example, with hierarchical decomposition [Sacerdoti74, Sacerdoti77]. The design and implementation can then be performed with the object-oriented approach. Our work is based on the first phases of application development, so we use the task-oriented approach with the DIANE+ method [Tarby93], which allows us to specify human-computer dialogue. From these specifications, we generate the user interface and a part of the application's code. The dialogue controller of our tool runs as an inference engine. It is responsible for the management of the interface, of the application and of the help module.

The DIANE and DIANE+ (extension of DIANE) methods are presented in this article. The first part is a presentation of the objectives of our work. The second part presents original concepts of DIANE and their evolution along DIANE+, supported by the case of preplanified tasks. The third part shows the formalism and its application through the electronic mail example. The fourth part presents the mock-up tool associated to DIANE+. The last part is a discussion beside similar works.

1 Objectives

The DIANE method has been created to solve malfunctions in the use of interactive software, leading to trouble in the information system and difficulties in the user learning and memorisation. The DIANE method [Barthet88] aims to integrate the user and his interaction capability into the current process of designing an interactive software.

According to the Seeheim model [Pfaff85], the Diane method, based on the analysis of tasks and users (aims, decision margin, experience), brings a model and formalism to describe the dialogue control and its interfacing with the core application.

DIANE+ [Tarby93] extends the DIANE method to make possible the automatic generation and the automatic management of the user interface. These extensions concern the model of dialogue control and the integration of an object data model called OPAC (sub-section 2.6).

This work is based upon a key concept: **the control sharing between man and machine**. This sharing does exist in any application and is comprised between two extreme cases: a complete control by machine or a complete control by man. Applications installed on satellites correspond to the first case, and the second case is close to creative applications such as drawing software. Any current case needs an accurate description of the control sharing. But the object methods do not highlight this sharing which is diluted through objects, making difficult a clear evaluation of control sharing between man and machine.

Our approach complements the object methods by integrating aspects relating to tasks and work stations, and concepts such as the user's level, activity, etc. In order to make this approach applicable, we need the concept of data, which is provided by the OPAC data model referring to current object concepts (inheritance, encapsulate, etc.), separately from the "task" aspect. Consequently, an application will be described first in terms of tasks specifying the control sharing, complemented in a next step by the manipulated OPAC data.

An objective of this work is to show that an application can be described by means of aims associated to tasks, these tasks being associated to the work stations of various kinds of users. This description could make possible to build up the skeleton of the application, including a provisional complete user interface and the code necessary to manage automatically the application (user interface, key functionalities and contextual help). In a first stage, our work limits to the domain of preplanified applications [Rasmussen83] and does not cover the field of expert or creative applications. Any application with preplanified tasks can be designed with DIANE+, for example the management of electronic mail that is presented in this paper.

2 Concepts of the DIANE+ Method

The DIANE+ method covers the specification phase of an interactive application; it makes possible to integrate the results of the analysis phase and can be used during the phases of analysis and specification as a formalism to describe an interactive application; it provides a detailed specification which can be used during the design phase or on automatic generation purpose, as shown in section 4.

The main characteristics of the DIANE+ method are presented below:

- the various representations of an interactive application (sub-section 2.1),
- the abstraction levels (sub-section 2.2),

- the aims and the user's logics (sub-section 2.3),
- the dialogue control sharing between man and machine (sub-section 2.4),
- the adaptation of dialogue to users (sub-section 2.5),
- the OPAC data model (sub-section 2.6).

All these characteristics make possible a quite complete description of the aspects relating to tasks and users.

2.1 The Various Representations of an Interactive Application

In order to integrate the human factors into the design of an interactive application, the DIANE+ method proposes a model of the interactive application including three viewpoints: the analyst's, the user's and the programmer's viewpoint. The links between these viewpoints and the concepts of cognitive psychology and ergonomics are presented in figure 1.

The analyst's viewpoint, called *Conceptual Representation*, describes for a workstation, first the general logics of the new information system, then the logics of the interactive processing, that is the dialogue control and the interface with the core application as defined in the Seeheim model.

During this phase starts the integration of the cognitive psychology elements relating to the user (role, experienced, beginner,...) into the characteristics of the task (effective task, hierarchical planification...) or into the interaction between both (user's logics). The Conceptual Representation covers the concept of utility [Senach90].

The user's viewpoint, called *External Representation*, corresponds to the software as it will be seen and operated by the user. This External Representation is made of two main parts; the first one translates the elements of the Conceptual Representation involved in the man-machine interaction; the second one takes into account all the specific elements of the External Representation (which corresponds to the presentation objects). It integrates all ergonomics elements such as those presented in a style guide [Scapin93, Smith84] corresponding to the software usability .

The programmer's viewpoint, called *Internal Representation*, corresponds to the implementation of both Conceptual and External Representations. No new user specification is generated during this phase which integrates no additional element of ergonomics. Therefore, it will not be described in this article.

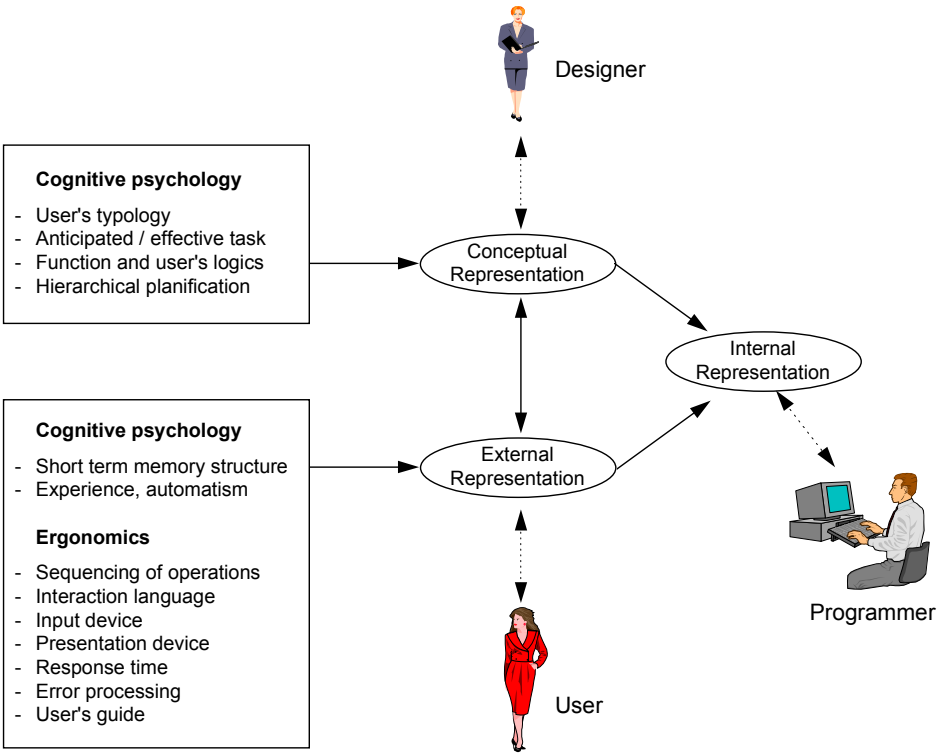

Figure 1. Cognitive ergonomics and the various representations of an interactive application

DIANE+ covers all these three representations. The design with DIANE+ covers the Conceptual Representation; the automatic generation of the user interface and the automatic management of the application cover the three Representations.

2.2 Abstraction Level

With DIANE+, an interactive application is represented not only by means of three viewpoints (sub-section 2.1) but also according to various abstraction levels.

The highest abstraction level includes the goals of the application and provides the most general view of its objectives. In fact, these goals are split up into sub-goals, sub-sub-goals, etc., corresponding to lower abstraction levels. This splitting ends at the elementary processes which correspond to the lowest abstraction level.

Whatever the abstraction level is, the designer is always provided with a complete view of the level. He does not need to know the lower levels to understand the functioning of the application. The more extended the splitting is, the more accurate the description of man-machine dialogue is. Thus, concerning the goals, the designer will specify for instance that goals 1 and 2 are independent from each other, or on the contrary strongly dependant. Then, in the lower abstraction levels, he will consider required process sequences, constraints on process, modes associated to these

process, etc. As a result, each process will be completely described at the lowest abstraction level.

2.3 The Aims and the User's Logics

Because DIANE+ is based on user's task analysis, identifying the aims of the various users is the starting point of the method. The aims reflect the functions that the organisation assigns to the workstations. An aim can be concrete like "delete a message" or subjective like "use the e-mail".

The aims can be defined:

- top-down, starting from the general objectives (of the user, the company or the department) and specialising them on the workstation;
- bottom-up, gathering the system functions aiming to the same objective on the workstation.

Then, for each aim, the user's logics is defined separately from the technical logics. In the case of preplanified or procedural tasks, this user's logics results in various procedures leading to the aim (sub-section 2.4).

Example: the management of the electronic mail may be considered as an aim with several sub-aims (send messages, read messages, organise the messages, etc.) which can be split up (organise may be split up into order the messages, delete messages, etc.), and so on.

2.4 Dialogue Control Sharing Between Man and Machine

Defining the control of the dialogue between man and machine is based on four questions that determine all necessary informations to manage later this control sharing. These informations relate mainly to two DIANE+ concepts which are the *operations* and the *precedences*. A precedence is a sequencing link between operations. An operation is either a process which can be performed (e.g. print the screen) or a set of operations, called sub-operations, which can be processes or sets of sub-operations, and so on.

The four questions are:

1. **Who triggers an operation ?** The triggering is *optional* when it is the user, and *automatic* when it is the computer. In the first case, only the user can trigger the operation and decides when to trigger it. In the second case, the user can absolutely not decide to trigger the operation.
2. **Who performs an operation ?** The operation is *manual* if it is the user, (e.g., sign a document), *automatic* if it is the computer (e.g., disconnect), and *interactive* if it is both (e.g., enter a name).
3. **Who checks the performing of an operation ?** The operation is *optional* when the user checks, and *required* when the computer checks. Example: for the "Record a client" aim, the "Enter the name of client" operation is required when the

"Print a client" operation is optional. All operations of consultation, printing, etc., are in general optional. Therefore, an optional operation always needs an optional triggering achieving the associated aim does not depend on the fact this operation has been performed or not. On the contrary, a required operation may be associated or not to an optional triggering, but it is absolutely necessary for the achievement of the aim.

Another kind of operation exists in DIANE+: the *constrained* operation. A constrained operation results of the splitting of an operation into sub-operations, when a constraint is associated in order to define how many sub-operations must be performed.

4. **Who controls the sequence of operations ?** The precedence is *indicative* if the user controls the sequence, and *permanent* if it is the computer. When an operation refers to no precedence, this means that it can be performed at any moment.

   Example: figure 2 schematises two equivalent DIANE+ specifications (ellipses have no particular signification here). Figure 2.a is a simplified example of a real DIANE+ specification. Figure 2.b shows a strictly equivalent representation of figure 2.a. In these figures there is only one permanent precedence between operations 1 and 2.

   This means that operations 1, 2, 3 and 4 can be performed in any sequence, as far as the constraint of the permanent precedence between 1 and 2 is fulfilled: operation 2 can be performed only after operation 1. So, a lot of sequences are possible, for example (**1**,**2**,3,4), (**1**,3,4,**2**), (3,**1**,**2**,4), (**1**,4,**2**,3),..., (1,2,4,2,3,1,4,3,2,4, 2),... We can see in figure 2.a that the DIANE+ formalism is very concise because we need ten additional arrows to represent the same possibilities in figure 2.b.

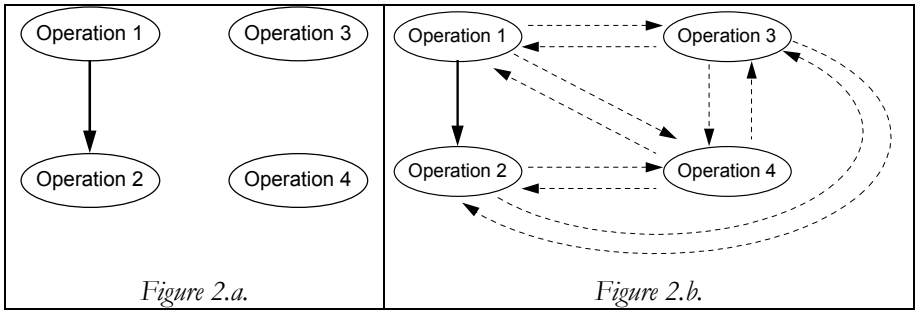

Figure 2. Example of a simplified DIANE+ specification (2.a) and its prescriptive equivalent (2.b)

The gathered informations are sufficient to specify the dialogue sharing beside the tasks, but they are not sufficient to manage data. Hence, the dialogue sharing is also described in the associated *OPAC data model* (sub-section 2.6). One role of the OPAC model is the elementary data processing, such as enter a name, display a post code, etc. which is taken into account through data and not through the DIANE+ operations.

2.5 Adaptation of Dialogue to Users

When the user's aims are determined, the next step is their detailed representation through the operations. For this stage, we use *procedures* which are formal and detailed descriptions of the manner to realise an aim. Thus, a procedure is a set of operations which may be linked by precedences.

The objective of this stage is to make possible an adaptation of the man-machine dialogue to the various kinds of users (experienced, beginner, level of responsibility, work habits, etc.), rather than constraining the dialogue control through a single standard procedure.

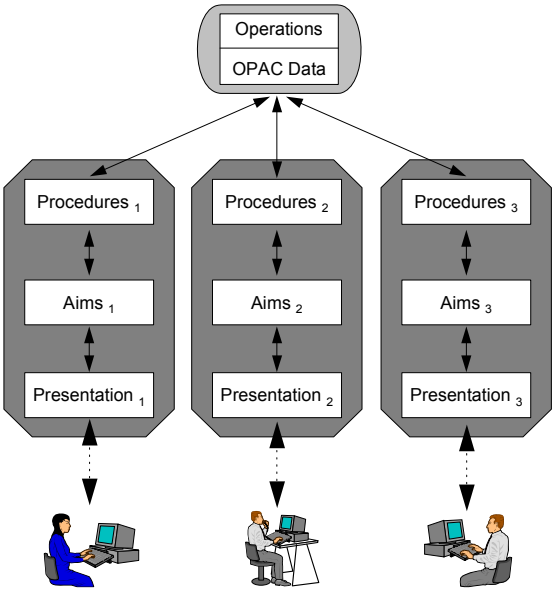

Figure 3. Adaptation of the man-machine dialogue to various kinds of users

Figure 3 shows such an adaptation; operations and OPAC data are used by three different users through procedures, aims and presentations which are dedicated to these users. The decomposition in operation/OPAC, procedures, aims and presentation may be compared to the Seeheim model (core application, application interface, dialogue controller and presentation).

To reach this goal, we define a *minimal procedure* presenting the more flexible dialogue control for the user, since it includes the constraints resulting only from the organisational and management rules.

The other kinds of dialogue, *effective* and *forecast procedures*, correspond to current cases, work habits, and optimisation by the experience. These various procedures must be compatible with the minimal one.

To represent these procedures, we use a formalism close to the one used in MERISE [Tardieu83] to describe the procedures and operations. Nevertheless, their semantics

is slightly different since they describe the sequence of operations necessary to the achievement of a goal by a user and not a domain of the information system.

2.5.1 Forecast Procedures

For each goal, we describe first the forecast procedure which results "naturally" from interviews and questionnaires essentially with persons who are responsible for carrying out the procedure. Describing the current forecast procedure is indispensable for the next stages of the study. More, they may be used for learning and for help because they reflect a coherent use of the application.

2.5.2 Effective Procedures

Effective procedures reflect real activity "in situ" or during simulations. To describe effective procedures corresponding to particular cases or to different work habits, it is necessary either to interview users more thoroughly, or to carry out observations or set up sensors or have self-observation forms filled up. Since describing the current effective procedures is a highly time consuming task, this analysis is to be undertaken only when the current procedures are not impacted or little impacted by the system change.

2.5.3 Minimal Procedures

Interviews cannot highlight spontaneously the minimal procedure which includes information generally not explicit. The point is to define explicitly the decision margin allowed to the users to do the work assigned to the station. This decision margin is represented by the kind of triggerings, precedences and operations.

To define the minimal procedure, we start from the forecast procedure previously collected and we ask questions in order to identify the automatic triggerings, permanent precedences and required operations.

2.6 The OPAC Data Model

The three kinds of procedures mentioned above make easier the adaptation of procedures to the users for processing purpose. To be complete, the procedures must include data. The current version of DIANE+ incorporates an object-oriented data model called OPAC[2] [Tarby93] derived from the PAC[3] model [Coutaz88].

The OPAC model structures data into elementary or compounded classes whom instances are capable of providing and managing their external and internal representations. This model also provides a set of methods (in the object context) for their manipulation.

---

[2] OPAC = natural Object PAC. An natural object is an object which have a sense for the user.
[3] PAC = Presentation, Abstraction, Control.

The aim of the model is to unload the basic data management into the data themselves. The OPAC model manages data in an elementary way, and DIANE+ manages data in the context of the application.

For example, OPAC data can display a client number, select characters in a text or record a date. However, the date validity control, with regard to the application's data, is processed by the DIANE+ procedures and operations.

The OPAC model provides a set of OPAC classes. These classes are specified once for all by the designer. The data processed in the DIANE+ specifications are instances of these classes and comprise three parts:

- an *Abstraction* which:
  - contains the data represented by the OPAC, for example a name of person, a question, a list of book titles, etc.
- a *Presentation* which:
  - proposes external representatives with regard to the Abstraction and links between the OPAC object and the DIANE+ operations. These external representatives are used during the user interface generation;
  - manages, in an elementary way, the associated external representation (selection, scrolling, etc.).
- a *Control* which:
  - provides a set of basic methods (in the object context) to manage the OPAC data (creation, suppression, display, etc.) independently of the DIANE+ operation;
  - maintains the consistency between the Abstraction and the Presentation.

An OPAC may be used partially through external views. These views limit the reachable data that contains the Abstraction.

For example, an OPAC which represent a person may have an external view with only the first name and the last name of the person. This have important consequences during the user interface generation because only widgets associated with this external view will be generated.

For example, if the OPAC is in input for an operation, two static text fields will be generated; if the OPAC is in output or input/output, it will be two entry text fields.

An OPAC may be split up into sub-OPACs. This feature is only interesting for the Internal Representation, i.e. for the designer and the programmer. For the user, the OPAC remains indivisible.

3 Formalism

DIANE+ uses a formalism (figure 4) created to minimise the designing work on the two following points:

- the procedures describe only the characteristics specific to an application, separately from the standard actions common to any application such as quit, cancel, etc. This assumes that the supposed to be standard actions, previously defined, are really common to any application. Only the cases where they are not applicable are to be described.
- the described procedures are not mandatory; what is not forbidden is allowed (figure 2).

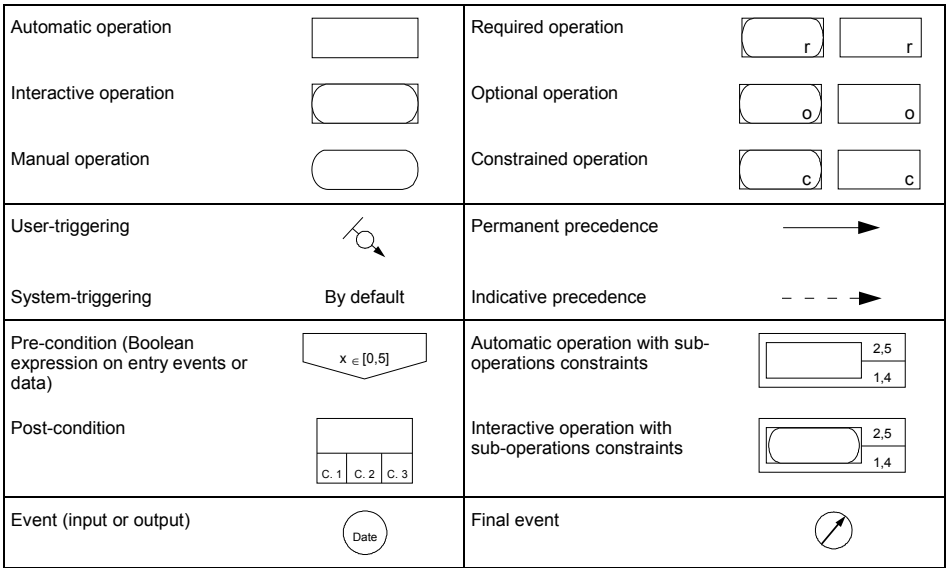

Figure 4. The DIANE+ formalism

3.1 Breakdown of Operations

The breakdown of an operation is submitted to no limit and no constraint. All types of operations and precedences can be mixed on an unlimited level of abstraction. The "electronic mail" example illustrates this possibility (figure 8).

3.2 Constrained Operations

When an operation is split into sub-operations, it is possible to apply constraints on this operation beside its sub-operations. This constraint is indicate with the [a,b] interval on figure 5. It means that at least 'a' and at most 'b' constrained sub-operations must be performed. All the sub-operations which are not constrained (required and optional) are not concerned by this interval (a required sub-operation must always be performed).

Any operation may have a personal constraint which concerns its number of required executions. This constraint is indicate with the [c,d] interval on figure 5. It means that the operation must be performed at least 'c' and at most 'd' times.

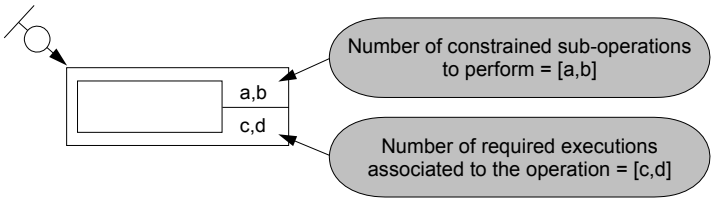

Figure 5. Graphical representation of an operation with constraint

In the following example, the Book number operation is completed after the [1,1] constraint have been fulfilled and all the required sub-operations have been performed.

The user must choose between a bar code entry and a keyboard entry (only one operation between Bar code and Keyboard operations), the two others remaining available to him (optional operations). The Book number operation must be performed at least one time and at most three times.

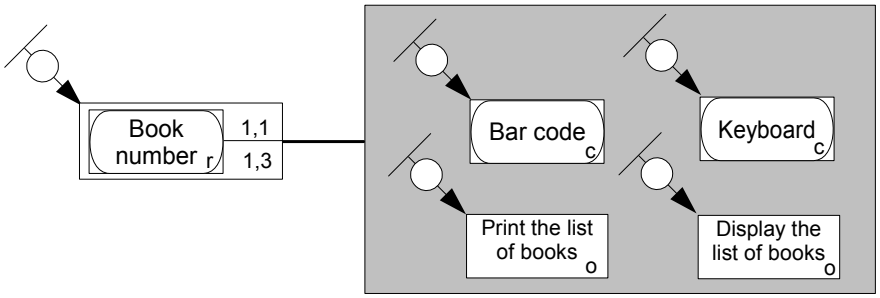

Figure 6. Example of operations with constraints

3.3 Link with OPAC Data

Figure 7 shows an example of an OPAC data split up into sub-OPAC data (name, address, phone and title) which can as well be split up (e.g. the address OPAC data). During the dialogue specification, DIANE+ operations are associated with OPAC data.

These associations precise the kinds of link with the data (input, output, input/output) and may be completed with an external view on the OPAC data.

When an operation uses a sub-abstraction (e.g., the phone number), this sub-abstraction must come from the OPAC data of the highest abstraction level. This OPAC data gets the requested sub-abstraction from the associated sub-OPAC which provides the methods to perform the operation.

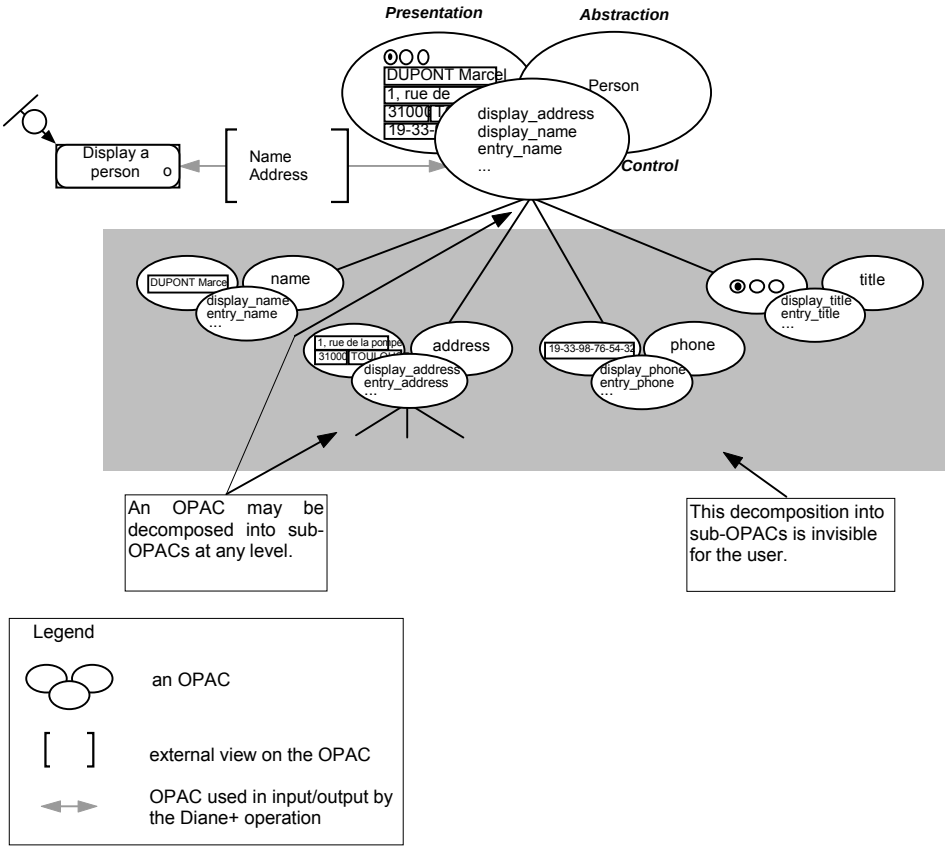

Figure 7. Example of compounded OPAC

3.4 Electronic Mail Example

The following example presents the specification of an electronic mail application. More constraints than in real have been considered in order to exemplify DIANE+. A normal session could be as follows:

1. The user must first connect himself to the system. To do this:
   1.1. he must be identified by the system (name + password)
   1.2. he must enter the e-mail command
   1.3. the mailbox is automatically opened by the system
2. Then, the user can work. He can choose between:
   - select a message. To do this, he must choose between:
     * select the first unread message
     * choose himself a message
   - read a message,
   - delete a message,
   - send a message. To do this, he must:

  * first, define the message by writing the subject (only one time) and the text (at will) in any order,
  * second, send the message by choosing a recipient. This last operation brings on automatically sending the message.

- reply to a message.

3. The user must disconnect from the system, which is done at time of closing the mailbox.

We note that DIANE+ can represent all the constraints of the above specifications. All the algorithmic structures do exist in DIANE+. More precisely:

- the *ordered sequence* is represented by the precedences, e.g., name→password,
- the *unordered sequence* is represented by the required operations and by a lack of precedence, e.g., enter the subject and enter the text in define the message,
- the *loop* is implicitly represented. An unconstrained user-triggered DIANE+ operation means that the user may execute it as often as he wants, e.g., send a message,
- the *required choice* is represented by an operation with constraint on its sub-operations, e.g., select a message with constraint [1,1],
- the *free choice* is represented by an operation without constraint on its sub-operations, e.g., use the e-mail,
- the *parallelism* is represented through the triggers and a lack of constraint. The loops makes possible to perform the same operation many times in parallel, and several loops can be performed in parallel, e.g., reply to the message may be performed in parallel with send a message, enter the subject, or enter the recipient, etc.
- the *default operations*. In the select a message operation, the select the first unread message operation is the default one. When the user presses the enter key (this key was chosen by the designer), the select the first unread message operation will be automatically performed. Consequently, the first element of the list of messages will be displayed in inverse video.
- the *number of times an operation must be performed*, e.g., identification must be performed at least one time and at most three times.
- the *number of constrained sub-operations to perform*, e.g., the [1,1] constraint for select a message.

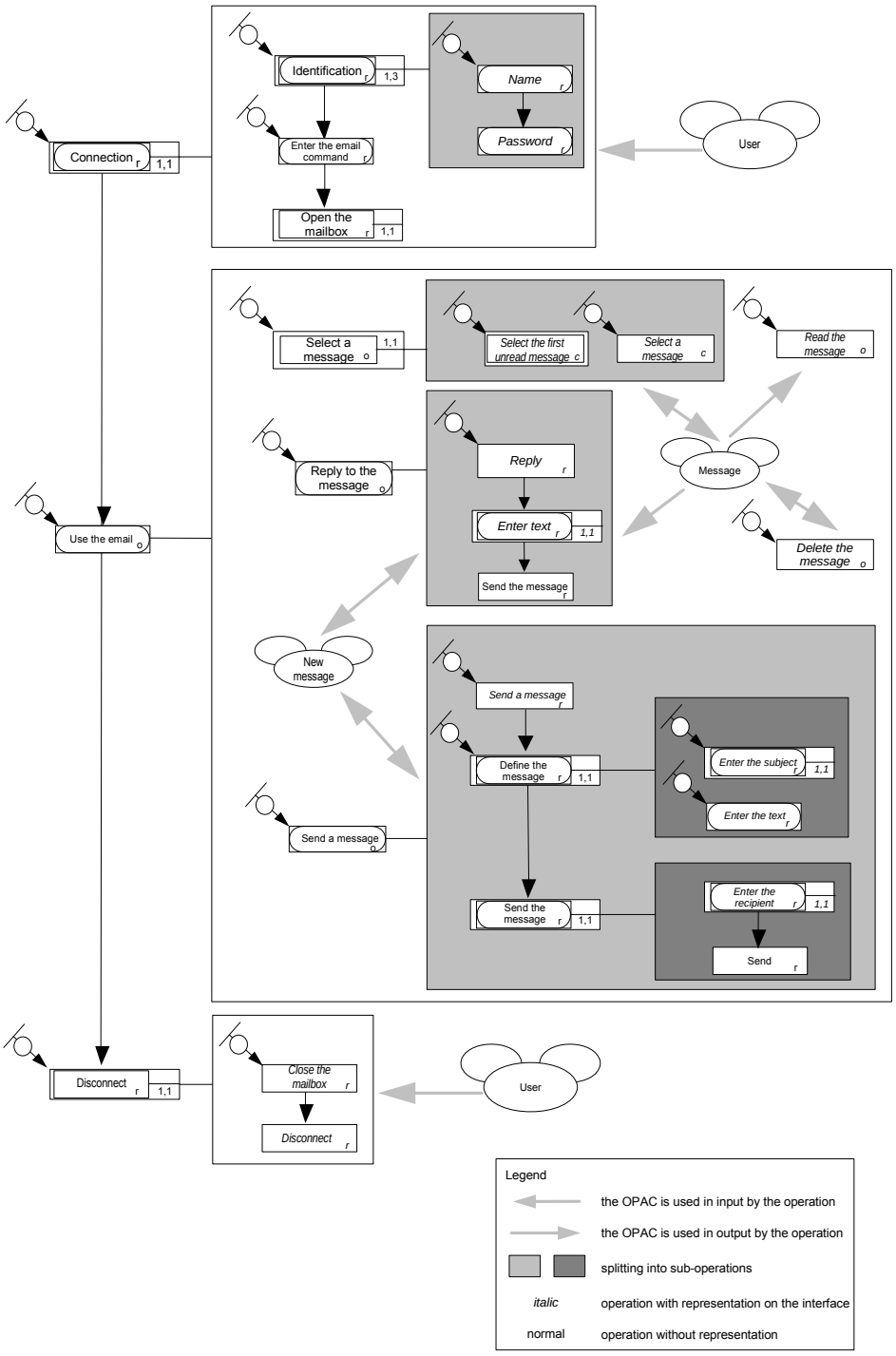

Figure 8. Representation of an electronic mail with DIANE+

4 Tool

In order to test the DIANE+ method, we are developing a tool (figure 9) making possible the automatic generation and management of the user interface from conceptual specifications. These specifications are described in detail with the DIANE+ formalism. We precise in this part the expected functionalities and the current limits of this tool.

4.1 Functionalities of the Tool

The tool must make possible:

- to enter the specification of the human-computer dialogue. This dialogue is described through DIANE+ procedures, as shown in the previous parts. The DIANE+ specifications are entered with a dedicated graphic editor.
- to test an application at each level from specification to generation. These tests consist in:
    - verifying the syntax of the DIANE+ procedures i.e. they are able to detect illegal constraints, errors in the description of an operation, etc. Simple syntactic checks may be made, for example it is impossible to design an optional operation without user-triggering, or to design a system-triggering operation which is split up into user-triggering sub-operations. Before testing the application, other checks concerning the DIANE+ semantics are performed, for example an operation with constraint on sub-operations must have constrained sub-operations.
    - checking the consistency between the generated user interface and the interface expected by the designer. For example, the tool must check the chaining of windows, the management of menus, the accessibility of operations (widgets), etc.
- the interface generation. The aim of this generation is not to obtain automatically a perfect interface but to provide a basic interface capable to manage itself its external view[4], with all the required elements for a correct behaviour of the application. This interface fulfils general criteria of human factors. It is possible to modify its code by using an UIMS or a resource editor. For example, it must be possible to modify the spatial disposition of entry fields, reorganise the menus or the options in the menus. We are thinking about interface generation rules from DIANE+ specifications. A possibility is to translate goals and sub-goals in menus and sub-menus, procedures in main windows, and highest level operations in

[4] At t_1 time, the user activates a widget; this action produces an E event. At t_2 time, the interface sends the message associated to the E event to the application. Between t_1 and t_2, the interface managed alone the event. For example, if the user clicks on a menu label, the interface intercepts the click event, opens the menu and select by default the first option in this menu. More, if an operation becomes disabled, the dialogue manager notifies the interface that reflects the state on the associated widget.

child windows. The lower level operations may be concerned by other rules such as 'user-triggering operations without constraint are represented by push-buttons'.

- to provide several External Representations. OPAC objects provide widgets beside the external views on their Abstraction and the nature on link with the DIANE+ operations (sub-section 2.6). If there is several possible Presentations for an association (OPAC data/DIANE+ operation), two possibilities may be envisaged: an automatic selection or a selection by the designer.

- the generation of the application objects (Internal Representation). After the operations and procedures have been described, it must be possible to generate objects which represent them. For example, each operation is an instance of the Operation class. Likewise, the generation of the interface creates objects used by this interface (windows, menus, buttons, etc.). The generation of objects representing data in computer uses the OPAC data objects that represent these data. OPAC data have been designed and implemented; due to their subjective aspect, they cannot been generated.

- the automatic management of the application behaviour. Some links are created during the generation of the interface and of the objects of the application. These links connect for example an operation to its external representations. The dialogue manager have always to know whether an operation is enabled or not for the user. For each modification of the operation's state, the dialogue manager updates automatically the external view of this operation. As a result, a menu or a push button may become enabled or disabled.

- the automatic management of the contextual help. The specification of the dialogue allows to manage automatically and entirely the contextual help according to a user's logics, i.e., from the user's viewpoint and not from the designer's viewpoint.

4.2 Implementation Results

A first version of the tool was developed in Smalltalk/V under Windows 3.1. Its two main components are a DIANE+ editor and a dialogue manager implemented as an inference engine. The inference engine has to manage the behaviour of the application and the contextual help (figure 9).

4.2.1 DIANE+ Editor

The DIANE+ editor is currently a textual editor (figure 10) that makes possible to create DIANE+ procedures and operations by using specialised editors (not presented here). It takes into account:

- the splitting of operation into sub-operations,

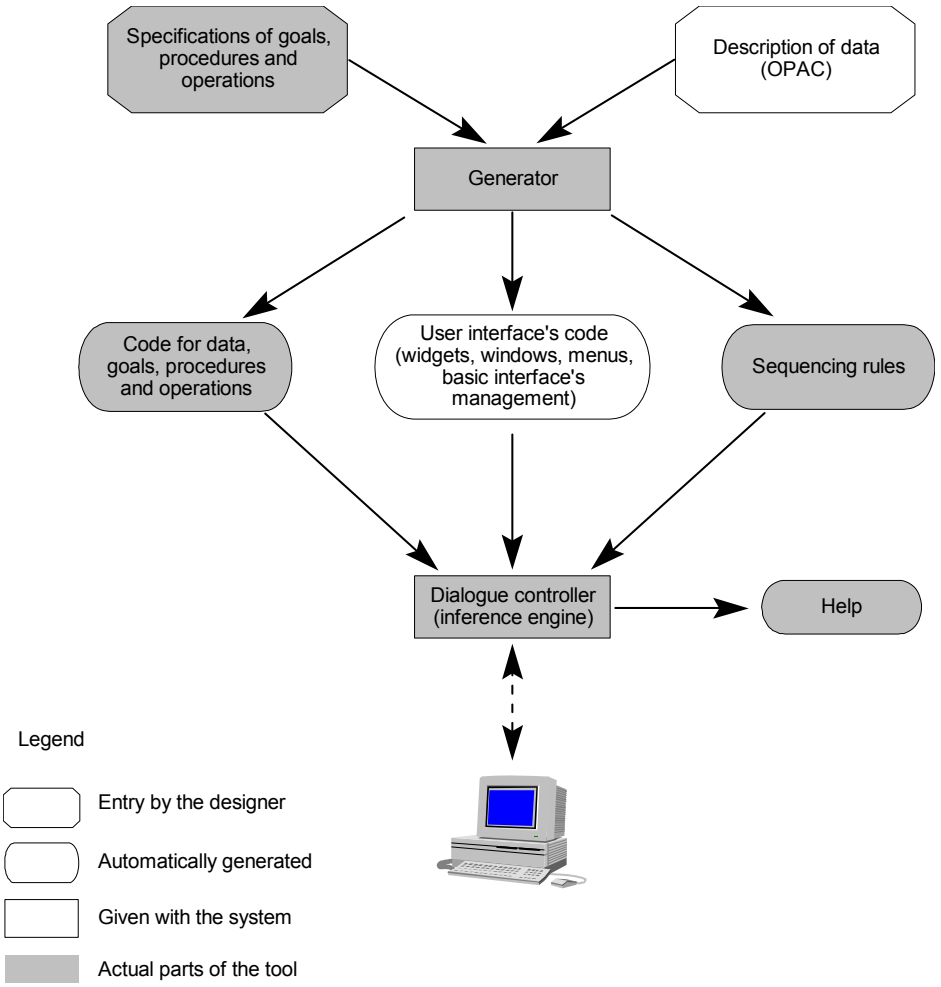

Figure 9. General working of the tool

- the management of constraints on operations and sub-operations,
- the pre- and post-conditions,
- the chaining rules which are the translation of precedences,
- the trigger (human or computer),
- the mode (interactive or automatic),
- the type (required or optional),
- the state (an operation can be active, locked, etc.),
- the list of the external representatives,
- the owner of an operation (an operation can belong to many owners, and for each owner there is an instance of this operation with its proper context).

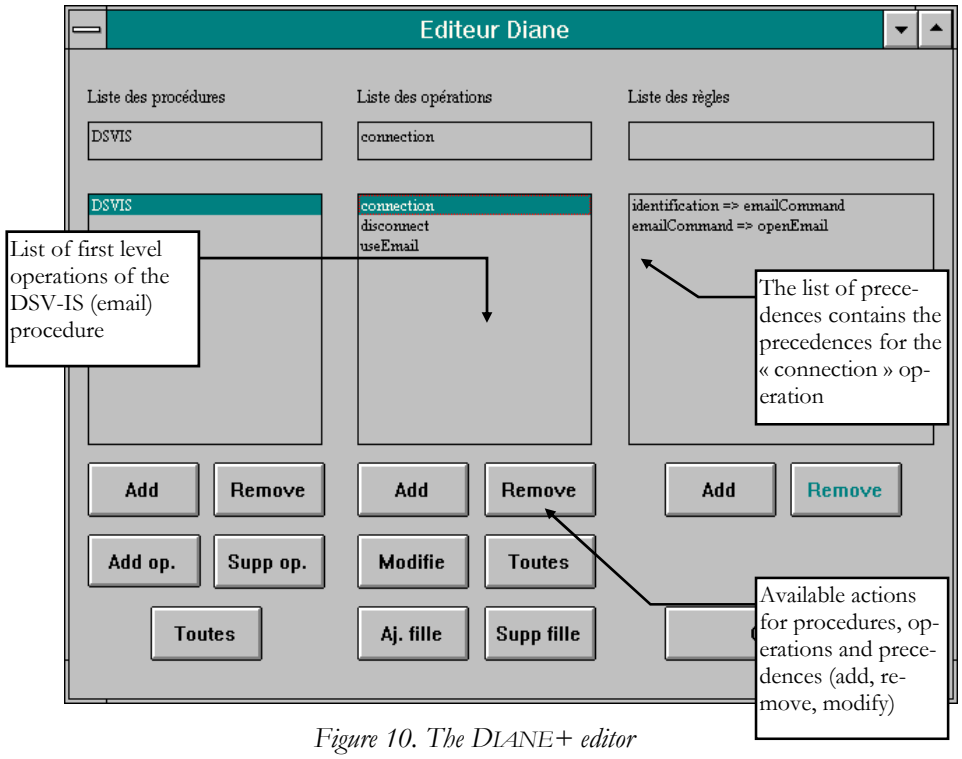

Figure 10. The DIANE+ editor

Figure 11 shows an example of instance for the name operation which belongs to the identification operation in the e-mail example.

By using this DIANE+ editor, we can modify in **real time** the chaining rules and all the attributes presented before (except the owner and the external representations) with immediate repercussions on the application.

Since the current version of the tool does not take the generation of the external view into account, we have to build it entirely. We use Window Builder which is an UIMS running under Smalltalk/V. This UIMS contains a window editor and a generator of Smalltalk code (a window = a class). Once the screen layouts are developed, we connect every widget to its associated DIANE+ operations and procedures with a simple Smalltalk code line. After this step, the system is ready to run.

4.2.2 Inference Engine

As soon as the DIANE+ specifications and the connections between these specifications and the user interface layout have been described, the inference engine can manage automatically the behaviour and the contextual help. In its current version, the inference engine cannot manage the "what is going on if ...?" question; this limit is due to technical constraint of Smalltalk concerning the copy of complex objects.

Three other questions are managed. These are: "how to ...?", "why ... is disabled ?", and "how to end ... ?".

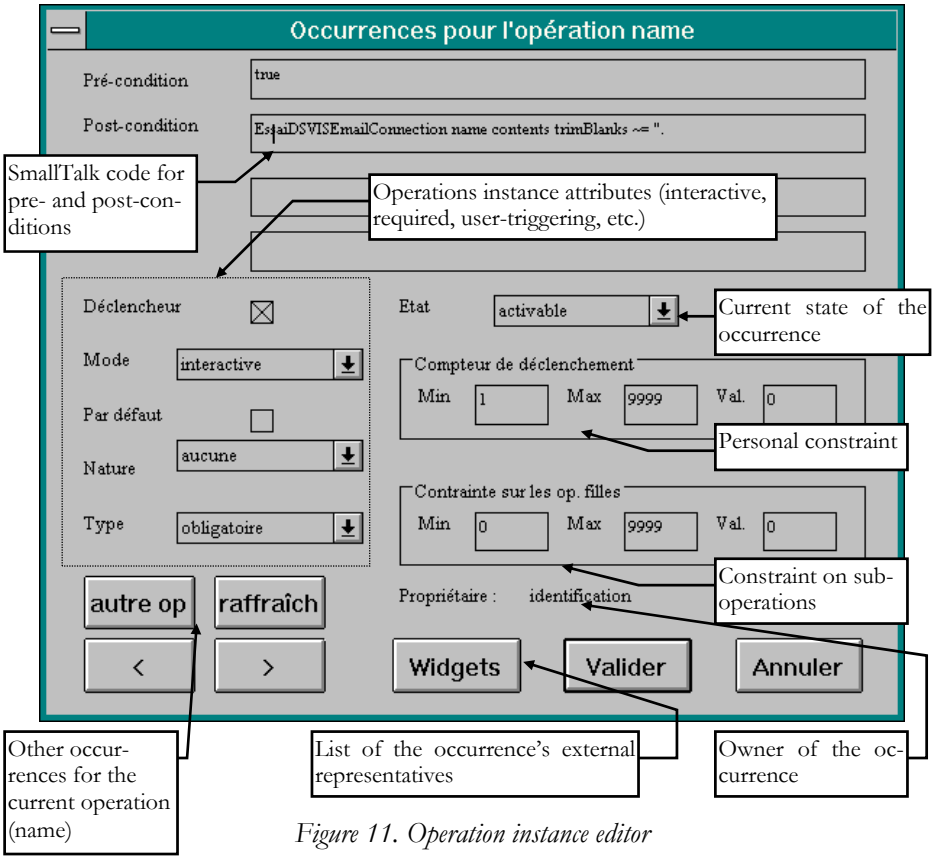

Figure 11. Operation instance editor

The results validated by the tool are:

- the automatic management of the behaviour for the external view and for the internal view (sequences, state transition, etc.),
- the automatic management of the contextual help for the three questions,
- the permanent consistency between the conceptual view and the contextual help and between the conceptual view and the behaviour, since every modification in the specifications (chaining rules, description of operation, etc.) is immediately taken into account in the behaviour and in the help.

Example: figures 12.a-f show screen captures of the electronic example (figure 8). In figure 12.a, the user is at the beginning of the procedure. Only name is enabled[5]. The user wants to know how to enable the push-button. The answer is given in

[5] It is possible to disable entry field AND label by adding the label widget to the list of external representatives of password operation occurrence.

figure 12.b: the identification operation must be ended, and for this DIANE+ says that the user must end the name operation and after the password operation.

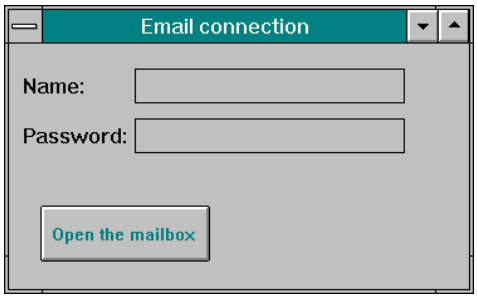

Figure 12.a

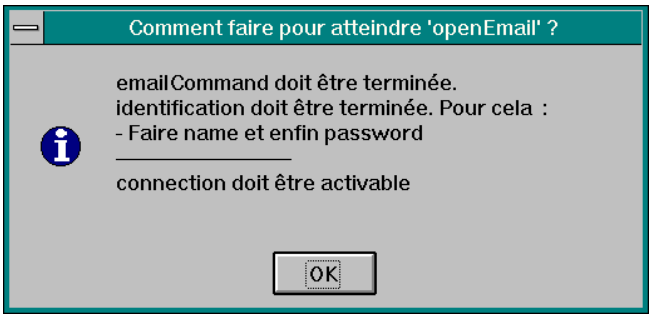

Figure 12.b

In figure 12.c, the user asked why password is disabled. The result shows that the postcondition of name is not verified (we choose "at least one character in the name field" as post-condition).

Figure 12.d presents the same situation later. The two entry fields are filled correctly. The user wants to know why name is dimmed (he wants enter an other name for example). The systems answers (figure 12.e) that the name operation will never be enabled for this session (identification is ended).

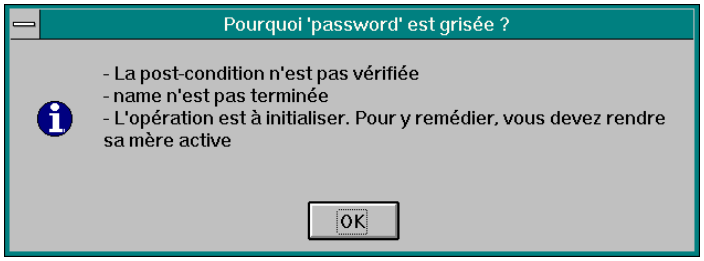

Figure 12.c

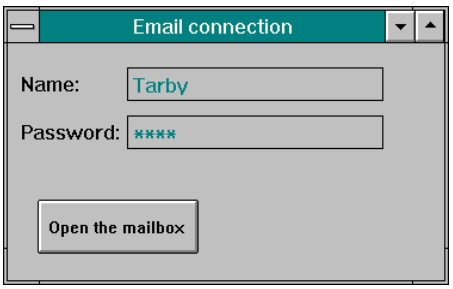

Figure 12.d

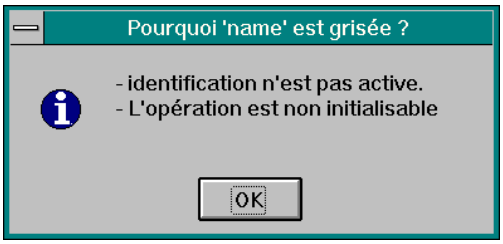

Figure 12.e

Figure 12. Example of automatic contextual help management

5 Related Work

We present here a comparison with related works through four topics:

1. design method,
2. user interface generation,
3. automatic user interface management,
4. automatic contextual help management.

5.1 Design Method

MUSE [Lim94] and DIANE+ have both the aim to integrate human factors in design methods but MUSE is more human factor oriented whereas DIANE+ is computer science oriented [Palanque94c].

DIANE+ is both task oriented and user oriented. It uses the user's logics through a task analysis which will become executable [Copas94], but it does not distinguish kinds of tasks and goals such as interface goals or social goals [Gilmore95]. DIANE+ integrates dialogue sharing between man and machine like the most design methods, e.g., MAD in its last version [Hammouche93] or UAN [Hix93], but it does not itemise process such as UAN.

DIANE+ uses both the concepts of tasks and objects like [Rosson95] as opposite of [Benyon95] which mainly uses the concept of data as support of task description.

The concepts of precedences is more powerful in DIANE+ as MAD and UAN, but a recent evolution of UAN (XUAN) [Gray94] may be compared with DIANE+.

In XUAN, the temporal constraints are more powerful than in DIANE+ which is explicit for the start of operations but not with the end of operation.

5.2 User Interface Generation

In its current version, DIANE+ does not integrate a module for the automatic UI generation, but proposes generation rules. However, works such as DON [Kim90, Kim93], UIDE [de Baar92], GENIUS [Janssen93], or TRIDENT [Bodart94a] are very close with DIANE+.

To be complete, a UI must integrate application domain knowledge [Gulliksen95] like MECANO [Puerta94b, Puerta96]. This may be possible with DIANE+ through OPAC data model, OOA [Balzert95a], or TRIDENT [Bodart93, Bodart94c, Bodart 95b, Bodart95d].

DIANE+ generates UI from task specifications, but an reverse approach may be interesting. For example, [Lauridsen95] generates the UI like DIANE+, but generates elementary dialogue specifications from UI layouts. This may be attractive in the case of reverse engineering.

DIANE+ uses both the concepts of task and object. DIGIS [de Bruin94a, de Bruin-94b] is very close with DIANE+ because:

1. It represents data through a data object model called D-PAC. This model is based on OMT data specification [Rumbaugh91].
2. It uses a task model that is based on UAN [Hix93].
3. Both DIANE+ and DIGIS are aimed to be used by non-progammers.

5.3 Automatic User Interface Management

DIANE+ and UIDE [Foley91] propose both an UI management partly based on pre- and post-conditions. In both cases, the dialogue management is a dedicated module which inspects regularly the state modifications of operations (Action objects in UIDE) and reflects these modifications onto the UI.

Several works use Petri Nets (PN) to represent dialogue specifications. PNs provide mathematical checking. Coupling such systems with ERA diagrams allows a more precise dialogue specification and management [Pettersson95]. PNs and DIANE+ may describe dialogue very precisely, but DIANE+ does not aim to describe elementary process such drag and drop or cut and paste.

The process are either represented in OPAC data or assimilated as standard actions, but we can specify them in DIANE+ when these two cases are not logical with regard to the dialogue specification.

5.4 Automatic Contextual Help Management

Contextual help is more and more present in interactive systems like in UIDE [Gieskens92] and H3 [Moriyón94]. UIDE answers only two questions (why an interaction object is disabled ? How to do ... ?) but integrates animation.

This last alternative should be incorporated into DIANE+ without major difficulties because DIANE+ inference engine is able to find the sequence of operations and their external representatives to answer a question.

In its current version, DIANE+'s answers are still limited as opposite as H3 which displays help through an hypertext. This characteristic is possible with DIANE+ supposing that some parts of text are entered by the designer (like H3), e.g., the meaning of the operations.

Conclusion

In this part, we present the advantages, the limits, and the foreseen extensions of our method.

Advantages and Limits

The main advantage of DIANE+ is the merging between a human factor approach and a software engineering design method, resulting in the adaptation of the design method formalism and the integration of characteristics of users and tasks.

The second advantage is a rigorous description of the dialogue control providing the largest flexibility to the users with a perfect consistency in regard to the management rules of the organisation. This advantage is really relevant in a context of preplanified or procedural tasks with organisational constraints. On the contrary, in the case of creative and individual tasks, defining the dialogue control becomes minor and we recommend to use object methods.

In its first version, DIANE did not include explicit links with objects. This lack was a major disadvantage for the graphic user interface design. This disadvantage disappeared in DIANE+ through the OPAC objects.

DIANE+ may also be reproached a waste of time in the stage of detailed specification, compared to RAD processes. This critic relates to the field of application of DIANE+. The RAD approach applies preferably to creative and computer-aided decision-making tasks, since little information on the current situation is available and there is no predefined procedures. DIANE+ fits more efficiently to the other kinds of task.

Extensions

Experimentation of DIANE+ has lead to highlight a designing process starting from the conceptual level until the external level. This is relevant when the dialogue con-

trol is more complex than the interaction objects. We intent now to identify applications for which a designing process starting from the interaction objects until the dialogue control would be more efficient.

In its current version, the tool manages the user interface and the help. But, the answers provided by the help are displayed in a poor style. We intent to improve the help messages, first by adding more text to make the meaning easier to understand, secondly by increasing the use of the operation attributes, for example to write "you **must** do...", "you **may** do...".

The current version of DIANE+ does not perform the ergonomic evaluation of a software, but DIANE+ has already been applied to the prior evaluation of an application in the field of air traffic control [Zorola95]. We intend to link our tool to human factors evaluation tools like ERGOVAL [Barthet94, Farenc96] to make the evaluation possible during the development phase and to provide advice during the specification phase.

Acknowledgements

The authors would like to thank the anonymous referees for helpful comments on earlier versions of this paper.

Reference

[Bailin 89] Bailin, S.C., *An Object-Oriented Requirements Specification Method*, Communications of the ACM, Vol. 32, No. 5, May 1989, pp. 608-623.

[Balzert95a] Balzert, H., *From OOA to GUI - The JANUS-System*, in [Interact95], pp. 319-324. http://www.swt.ruhr-uni-bochum.de/forschung/janus/lillehammer. html

[Barthet88] Barthet, M.-F., *Logiciels interactifs et ergonomie*, Ed. Dunod Informatique, Paris, 1988.

[Barthet94] Barthet, M.-F., Liberati, V., Ponamale, M., *ERGOVAL - A Software User Interface Tool*, in Proceedings of the 12th Triennal Conference of International Ergonomics Association IEA'94 (Toronto, 15-19 August 1994), Vol. 4, Human Factors Association of Canada, Toronto, 1994, pp. 428-431.

[Benyon95] Benyon, D., *A Data Centred Framework for User-Centred Design*, in Proceedings of the 5th IFIP TC13 Conference on Human-Computer Interaction INTERACT'95, Lillehammer, 25-29 June 1995, K. Nordbyn, P.H. Helmersen, D.J. Gilmore and S.A. Arnesen (Eds.), Chapman & Hall, London, 1995, pp. 197-202.

[Bodart93] Bodart, F., Hennebert, A.-M., Leheureux, J.-M., Sacré, I., Vanderdonckt, J., *Architecture Elements for Highly-Interactive Business-Oriented Applications*, in Proceedings of the East-West International Conference on Human-Computer Interaction EWHCI'93 (Moscow, 1993), L. Bass, J. Gornostaev and C. Unger (Eds.), Lecture Notes in Computer Science, Vol. 753, Springer-Verlag, Berlin, 1993, pp. 83-104.

[Bodart94a] Bodart, F., Hennebert, A.-M., Leheureux, J.-M., Vanderdonckt, J., *Towards a Dynamic Strategy for Computer-Aided Visual Placement*, in Proceedings of 2nd Workshop on Advanced Visual Interfaces AVI'94 (Bari, 1-4 June 1994), T. Catarci, M.F. Costabile, S. Levialdi, G. Santucci (Eds.), ACM Press, New York, 1994, pp. 78-87.

[Bodart94c] Bodart, F., Vanderdonckt, J., *On the Problem of Selecting Interaction Objects*, in Proceedings of British Conference on Human-Computer Interaction HCI'94 « People and Computers IX » (Glasgow, 23-26 August 1994), G. Cockton, S.W. Draper, G.R.S. Weir (Eds.), Cambridge University Press, Cambridge, 1994, pp. 163-178. http://www.info.fundp.ac.be/cgi-bin/pub-spec-paper? RP-94-018

[Bodart95b] Bodart, F., Hennebert, A.-M., Leheureux, J.-M., Vanderdonckt, J., *Computer-Aided Window Identification in TRIDENT*, in Proceedings of the 5th IFIP TC13 Conference on Human-Computer Interaction INTERACT'95, Lillehammer, 25-29 June 1995, K. Nordbyn, P.H. Helmersen, D.J. Gilmore and S.A. Arnesen (Eds.), Chapman & Hall, London, 1995, pp. 331-336. http: //www.info.fundp.ac.be/cgi-bin/pub-spec-paper?RP-95-021

[Bodart95d] Bodart, F., Vanderdonckt, J., *Using Ergonomic Rules for User Interface Evaluation by Linguistic Ergonomic Criteria*, in Proceedings of 6th International Conference on Human-Computer Interaction HCI International'95 (Yokohama, 9-14 July 1995), Y. Anzai, K. Ogawa and H. Mori (Eds.), Advances in Human Factors/Ergonomics Series, Vol. 20A Symbiosis of Human and Artifact: Future Computing and Design for Human-Computer Interaction, Elsevier Science B.V., Amsterdam, 1995, pp. 367-372. http://www. info.fundp.ac.be/cgi-bin/pub-spec-paper?RP-95-023

[Brunet91] Brunet, E., *KADS: Engineering Knowledge Method*, Génie Logiciel et Systèmes Experts, No. 23, June 1991, pp. 24-34.

Coad90

[Colbert 89] Colbert, E., *The Object-Oriented Software Development Method: A Practical Approach to Object-Oriented Development*, in Proceedings of TRI-ADA'89 (Pittsburgh, 23-26 October 1989), C. Engles and J. Foreman (eds.).

[Copas94] Copas, C.V., Edmonds, E.A., *Executable Task Analysis: Integration Issues*, in Proceedings of British Conference on Human-Computer Interaction HCI'94 « People and Computers IX » (Glasgow, 23-26 August 1994), G. Cockton, S.W. Draper, G.R.S. Weir (Eds.), Cambridge University Press, Cambridge, 1994, pp. 339-352.

[Coutaz88] Coutaz, J., *Human-Computer Interface: Design and Implementation*, Ph.D. thesis, Université Joseph Fourier, Grenoble, France, 1988.

[de Baar92] de Baar, D.J.M.J., Foley, J., Mullet, K.E., *Coupling Application Design and User Interface Design*, in Proceedings of the Conference on Human Factors in Computing Systems CHI'92 « Striking a balance » (Monterey, 3-7 May 1992), P. Bauersfeld, J. Bennett, G. Lynch (Eds.), ACM Press, New York, 1992, pp. 259-266. ftp://ftp.gvu.gatech.edu/pub/gvu/ tech-reports/91-10.ps.Z.

[de Bruin94a] de Bruin, H., Bouwman, P., van den Bos, J., *A Task Oriented Methodology for the Development of Interactive Systems as used in DIGIS,* in Proceedings of the 15th Interdisciplinary Workshop on Informatics and Psychology, Interdisciplinary Approaches to System Analysis and Design (Schaerding, 1994).

[de Bruin94b] de Bruin, H., Bouwman, P., van den Bos, J., *Modeling and Analyzing Human-Computer Dialogues with Protocols*, in Proceedings of 1st Eurographics Workshop on Design, Specification, Verification of Interactive Systems DSV-IS'94 (Bocca di Magra, 8-10 June 1994), F. Paternó (Ed.), Focus on Computer Graphics Series, Springer-Verlag, Berlin, 1995, pp. 95-116. ftp://ftp.cs.few. eur.nl/pub/doc/papers/digis/diamodel.ps.Z

[Farenc96] Farenc, Ch., Liberati, V., Barthet, M.-F., *Automatic Ergonomic Evaluation: What are the Limits?*, in this volume, pp. 159-170.

[Foley91] Foley, J.D., Kim, W.C., Kovacevic, S., Murray, K., *UIDE - An Intelligent User Interface Design Environment*, in « Intelligent User Interfaces », J.W. Sullivan, S.W. Tyler (Eds.), Addison Wesley, ACM Press, 1991, pp. 339-384.

[Gibson 90] Gibson, E., *Objects. Born and Bred*, Byte, October 1990, pp. 245-254.

[Gieskens92] Gieskens, D.F., Foley J.D., *Controlling User Interface Objects through Pre- and Postconditions* , in Proceedings of the Conference on Human Factors in Computing Systems CHI'92 « Striking a balance » (Monterey, 3-7 May 1992), P. Bauersfeld, J. Bennett, G. Lynch (Eds.), ACM Press, New York, 1992, pp. 189-194.

[Gilmore95] Gilmore, D., *Interface Design: Have we got it wrong?*, in Proceedings of the 5th IFIP TC13 Conference on Human-Computer Interaction INTERACT'95, Lillehammer, 25-29 June 1995, K. Nordbyn, P.H. Helmersen, D.J. Gilmore and S.A. Arnesen (Eds.), Chapman & Hall, London, 1995, pp. 173-184.

[Gray94] Gray, P., England, D., McGowan, S., *XUAN: Enhancing to Capture Temporal Relationships among Actions*, in Proceedings of British Conference on Human-Computer Interaction HCI'94 « People and Computers IX » (Glasgow, 23-26 August 1994), G. Cockton, S.W. Draper, G.R.S. Weir (Eds.), Cambridge University Press, Cambridge, 1994, pp. 301-312.

[Hammouche93] Hammouche, H., *De la Modélisation des Tâches à la Spécification d'Interfaces Utilisateur*, Research report INRIA No. 1959, July 1993.

[Hickmann 89] Hickmann, F.R., Killin, J.L., Land, L., Porter, D., Taylor, R.M., *Analysis for Knowledge Based Systems. A Practical Guide to the KADS Methodology*, Ellis Horwood, Chichester, 1989.

[Hix93] Hix, D., Hartson, H.D., *Developing User Interfaces - Ensuring Usability Through Product and Process*, John Wiley & Sons, New York, 1993.

[Janssen93] Janssen, C., Weisbecker, A., Ziegler, J. , *Generating User Interfaces from Data Models and Dialogue Net Specifications*, in Proceedings of the Conference on Human

Factors in Computing Systems INTERCHI'93 « Bridges Between Worlds » (Amsterdam, 24-29 April 1993), S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, T. White (Eds.), ACM Press, New York, 1993, pp. 418-423.

[Kim90] Kim, W., Foley, J.D., *DON: User Interface Presentation Design Assistant*, in Proceedings of the 3rd Annual Symposium on User Interface Software and Technology UIST'90 (Snowbird, 3-5 October 1990), ACM Press, New York, 1990, pp. 10-20.

[Kim93] Kim, W.C., Foley, J.D., *Providing High-level Control and Expert Assistance in the User Interface Presentation Design*, in Proceedings of the Conference on Human Factors in Computing Systems INTERCHI'93 « Bridges Between Worlds » (Amsterdam, 24-29 April 1993), S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, T. White (Eds.), ACM Press, New York, 1993, pp. 430-437.

[Lauridsen95] Lauridsen, O., *Generation of user interfaces using formal specification*, in Proceedings of the 5th IFIP TC13 Conference on Human-Computer Interaction INTERACT'95, Lillehammer, 25-29 June 1995, K. Nordbyn, P.H. Helmersen, D.J. Gilmore and S.A. Arnesen (Eds.), Chapman & Hall, London, 1995, pp. 325-330.

[Lim94a] Lim, K.Y., Long, J., *The MUSE Method for Usability Engineering*, Cambridge University Press, Cambridge, 1994.

[Moriyón94] Moriyón, R., Szekely, P., Neches, R., *Automatic Generation of Help from Interface Design Models*, in Companion of the Conference on Human Factors in Computing Systems CHI'94 « Celebrating Interdependence » (Boston, 24-28 April 1994), C. Plaisant (Ed.), ACM Press, New York, 1994, pp. 225-231. http://www.isi.edu/isd/CHI94-Help.ps

[Palanque94c] Palanque Ph., Long, J., Tarby, J.-C., Barthet M.-F., Lim, K., *Ergonomic Application Design: a Method for Computer Science and a Method for Human Factors*, in Proceedings of ERGO-IA'94 (Biarritz, October 1994), Imprimerie Andre Larre, Bayonne, 1994, pp. 394-405. http://www.cenatls.cena.dgac.fr/~palanque/Ps/ ergo-ia94.ps.gz

[Pettersson95] Pettersson, M., *Designing the User Interface on Top of a conceptual Model*, in Proceedings of 7th International Conference on Advanced Information Systems Engineering CAISE'95 (Jyväskylä, 12-16 June 1995), G. Goos, J. Hartmanis, J.van Leeuwen (Eds.), Lecture Notes in Computer Science Vol. 932, Springer-Verlag, Berlin, pp. 231-242.

[Pfaff85] Pfaff, G. (Ed.), Proceedings of Eurographics seminar (November 1983), *Tutorial and perspectives in computer graphics; user interface management system*, Springer-Verlag, Berlin, 1985.

[Puerta94b] Puerta, A.R., Eriksson, H., Gennari, J.H., Musen, M.A., *Beyond Data Models for Automated User Interface Generation*, in Proceedings of British Conference on Human-Computer Interaction HCI'94 « People and Computers IX » (Glasgow, 23-

26 August 1994), G. Cockton, S.W. Draper, G.R.S. Weir (Eds.), Cambridge University Press, Cambridge, 1994, pp. 353-366. http://www-ksl.stanford.edu/KSL_Abstracts/KSL-93-62.html

[Puerta96a] Puerta, A.R., *The MECANO Project: Enabling User-Task Automation During Interface Development*, in Proceedings of AAAI'96 Spring Symposium on Acquisition, Learning & Demonstration: Automating Tasks for Users (Stanford, March 1996), AAAI Press, pp. 117-121.

[Rosson95] Rosson, M.B., Carroll, J.M., *Integrating Task and Software Development for Object-Oriented Applications*, in Proceedings of the Conference on Human Factors in Computing Systems CHI'95 « Mosaic of Creativity » (Denver, 7-11 May 1995), I.R. Katz, R. Mack, L. Marks, M.B. Rosson, J. Nielen (Eds.), ACM Press, New York, 1995, pp. 377-384.

[Rumbaugh91] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., *Object-Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs, 1991.

[Sacerdoti 74] Sacerdoti, E.D., *Planing a Hierarchy of Abstraction Spaces*, Artificial Intelligence, Vol. 5, No. 2, Summer 1974, pp. 115-135.

[Sacerdoti 77] Sacerdoti, E.D., *A Structure for Plans and Behavior*, Elsevier Computer Science Library, New York, 1977.

[Scapin93] Scapin D.L., Bastien, J.M, *Ergonomics Criteria for the Evaluation of Human-Computer Interfaces*, Report INRIA No. 156, June 1993.

[Schlaer88] Schlaer, S., Mellor, S.J., *Object Life Cycles: Modeling the World in States*, Yourdon Press, Prentice Hall, Englewood Cliffs, 1991.

[Sebillotte 88] Sebillotte, S., *Hierarchical Planning as Method for Task Analysis: the Example of Office Task Analysis*, Behaviour and Information Technology, Vol. 7, No. 3, 1988, pp. 275-293.

[Sebillotte 91] Sebillotte, S., *Task Description according User's Objectives,* Le Travail Humain, Vol. 54, No. 3, 1991, pp. 193-223.

[Senach90] Senach, B., *Evaluation ergonomique des interfaces homme-machine: une revue de la litterature*, Report INRIA No. 1180, March 1990.

[Smith84] Smith, S.L., Mosier, J. N., *A design evaluation checklist for user-system interface software*, Report MTR-9480 EDS-TR-84-358, The MITRE Corporation, Bedford, 1984.

[Tarby93] Tarby, J.-C, *Gestion Automatique du Dialogue Homme-Machine à partir de Spécifications Conceptuelles [Automatic Human-Computer Dialogue Management from Conceptual Specifications]*, Ph.D. thesis, Université de Toulouse I, Toulouse, September 1993. http://www-trigone.univ-lille1.fr/jean_claude/publis.htm

[Tardieu83] Tardieu, H., Rochfeld, A., Coletti, R., *La méthode MERISE, Principes et outils*, Ed. Organisation, Paris, 1983.

Vogel88

[Zorola95] Zorola Villareal R., *L'évaluation des IHMs Multi-utilisateurs dans le Travail Coopératif*, PhD. thesis, Université Toulouse I, October 1995.