Encapsulating Knowledge For Intelligent Automatic Interaction Objects Selection

Jean M. Vanderdonckt, Francois Bodart

RP-93-005 April 1993

in *Proceedings of ACM Conference on Human Aspects in Computing Systems InterCHI'93* (Amsterdam, 24-29 April 1993), S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, T. White (Eds.), Addison Wesley, Reading (Massachusetts), pp. 424-429.

Encapsulating Knowledge For Intelligent Automatic Interaction Objects Selection

Jean M. Vanderdonckt, François Bodart

Facultés Universitaires Notre-Dame de la Paix, Institut d'Informatique
Rue Grandgagnage, 21, B-5000 NAMUR (Belgium)
Tel. : +32 (0)81-72.49.75 - Fax. : +32 (0)81-72.49.67 - Telex : 59.222 FacNamB
eMail : JVANDERDONCKT@INFO.FUNDP.AC.BE, FBODART@INFO.FUNDP.AC.BE

ABSTRACT

TRIDENT is a set of interactive tools that automatically generates a user interface for highly-interactive business-oriented applications. It includes an intelligent interaction objects selection based on three differents concepts. First, an object oriented typology classifies abstract interaction objects to allow a presentation independent selection. Second, guidelines are translated into automatic rules to select abstract interaction objects from both an application data model and a dialog model. Third, these guidelines are encapsulated in a decision tree technique to make the reasoning obvious to the user. This approach guarantees a target environment independent user interface. Once this specified, abstract interaction objects are mapped into concrete interaction objects to produce the observable interface.

KEYWORDS: Automatic User Interface Generation, Decision Tree, Intelligent User Interface, Interaction Objects, Rule-Based System.

INTRODUCTION

Specifying the semantic of the application with a high-level language in order to automatically generate the user interface is a widespread approach [2,12,17,18] which implies two activities: (i) select adequate interaction objects from the semantic, and, (ii) lay out these objects into a more comprehensive one with a logical arrangement. Different authors have identified the following requirements for an intelligent automatic selection technique:

- *rule-based automatic generation* [18]: this can assess whether a user interface meets consistent selection rules;
- *explicit rules* [12]: built-in, code-imbedded rules are implicit in the system and, therefore, invisible and unmodifiable by the designer (textually or graphically);
- *application semantic involvement* [8]: interaction objects should vary according to the application data structure, not according to the presentation;
- *dialog model support* [1]: interaction objects should reflect the nature, the structure, the modality and the complexity of the dialog;
- *user model support* [1,9]: interaction objects have to be selected by taking into account the user experience level, the user skill to manipulate interaction media;
- *related interaction objects grouping* [2,9]: important functional groups have to be identified and formed;
- *screen space consideration* [2]: the selection should not generate overcrowded or cumbersome screens;
- *environment independence* [9,17]: the selection should work in any target environment.

Having only a presentation-independent selection [17] is not sufficient: the selection must be independent of presentation tools (e.g. OSF/Motif, Open Look,...) but also independent of the graphical window managers, UIMSs, toolkits (e.g. X-Windows, Ms-Windows,...). Why not concentrate all the knowledge about interaction objects selection into a such module? This paper describes an intelligent interaction object selection based on a decision tree technique that addresses these problems.

AUTOMATIC USER INTERFACE GENERATION

TRIDENT project [5] extends the computer-aided design methodology IDA (Interactive Design Approach) [3,4] in order to generate highly-interactive business-oriented applications along two dialog dimensions: the conversation from the architecture and specifications [13], the presentation from specifications (this is the subject of this paper). The dialog can be either asynchronous or multi-threaded. It is assumed that the semantic of the application is specified within different conceptual models and that interaction styles and modes are already given. Two conceptual models are used for the selection :

1. an information structure model provides specifications of the data structure of the application following an entity-relationship-attribute (ERA) way;
2. a dialog dynamic model provides a function chaining graph (FCG) specifying the data flow during the task performance.

The models are specified with Dynamic Specification Language, a high-level declarative specification language, and are then stored into local ERA and FCG data bases (fig. 1).

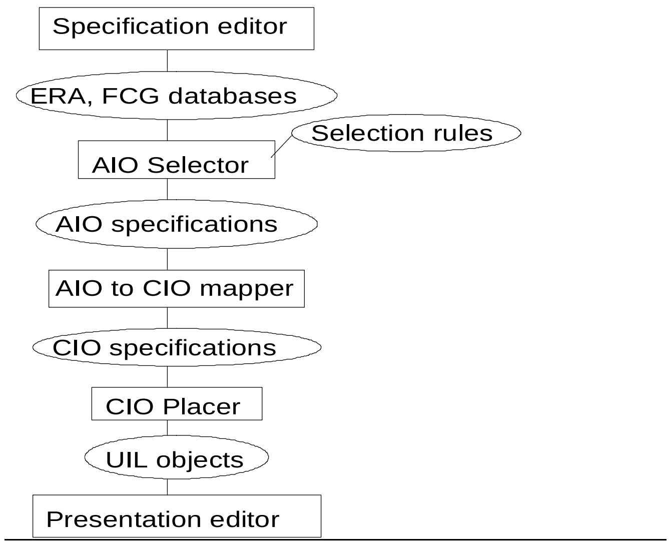


Figure 1. The TRIDENT approach.

By using rules, the AIO selector automatically selects abstract interaction objects (AIO) from the specifications and creates AIO specifications. Because these specifications are target independent, the mapper transforms AIO into concrete interaction objects (CIO) which become target environment dependent. The CIO Placer places them with three techniques inspired from [9]: physical object localisation, appropriate and aesthetic sizing and ergonomical arrangement. The complete resulting interface is translated into UIL objects which can be fully edited by a direct manipulation presentation editor.

ABSTRACT AND CONCRETE INTERACTION OBJECTS

Concrete interaction objects are graphical objects for inputting and displaying data related to the user interactive task; CIO are sometimes called widgets (window gadgets), controls or physical interactors (according to IFIP terminology). CIO specifications cover two major aspects :

1. the graphical appearance univocally determined by both presentation tool (e.g. OSF/Motif) and graphical tool (e.g. X-Windows);
2. the behaviour and constraints to be respected when the user manipulates the object.

When selecting an object, the designer have to pay more attention on the behaviour aspect because user interface communication power and ease of use are vital. Defining the presentation is an annex task to be left to a graphical artist. This observation invites to define interaction objects independently of the presentation, but not independently of their behaviour. The abstract interaction object, or logical interactor (according to IFIP terminology), brings an abstracted view of the CIO where physical characteristics are target environment independent.

AIO sets	AIO elements (some)
action objects	menu, menu item, menu bar, drop-down menu, cascade menu, submenu,...
scrolling objects	scroll arrow, scroll cursor, scroll bar, frame
static objects	label, separator, group box, prompt, icon
control objects	edit box, scale, dial, check box, switch, radio box, spin button, push button, list box, drop-down list box, combination box, table,...
dialog objects	window, help window, dialog box, expandable dialog box, radio dialog box, panel,...
feedback objects	message, progression indicator, contextual cursor

Table 1. Table of abstract interaction objects.

Examining different target environments allows to establish a comprehensive object-oriented AIO typology divided in 6 sets by interactive capabilities (table 1). Each AIO is identified by an unique generic name (e.g. *check box*), general and particular abstract attributes (e.g. *height, width, color, states*), abstract events (e.g. *value selection, mouse click*), primitive functions (e.g. *Pr-EditBoxContent*). By definition, each AIO have no graphical appearance, but each AIO is connected to 0, 1 or many CIO in different environments with different names, presentations (fig. 2).

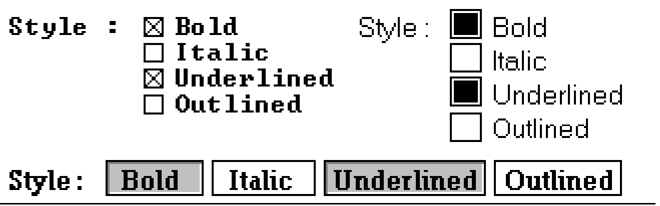


Figure 2. The AIO check box in different environments : Check box in Ms-Windows, XmToggleButton in OSF/Motif, BoxArray in Garnet.

Every AIO can be either *simple* (described in the typology) or *composite* (refinable as containing two or more simple AIO). Scrolling, feedback, control AIO are by nature simple whereas dialog AIO are typically composite: users may input, modify, select the values which may be buffered within the composite objects before transmission to the application.

AUTOMATIC AIO SELECTION

The first idea behind the AIO selection rests on selection rules mapping AIO from the detailed specifications of the semantic [1]. This approach has already been proved successfull with CIO selection rather than AIO selection.

MacIDA [12] selects CIO from database design: a window is mapped as for each entity type and for each relationship, an edit box is mapped for each attribute, a push button is mapped for each function. NIKL [2] includes straightforward selection rules mapping graphical CIO from the application domain model in three phases: realization, selection and redescription. Jade [27] also promotes simple and direct selection rules mapping a CIO from the data behaviour and from information contained in a look-and-feel file. UIDE [3] uses more sophisticated rules by following guidelines provided by different style guides. UIDE's selection rules map a CIO from attributes of the data object and from metadata depending of the attribute type (boolean, integer, real, enumerated and string). Observing that semantic considerations play a crucial role in the CIO selection, Selectors [8] invokes selection rules mapping a CIO from complete task-domain variables. Humanoïd [15] interprets selection rules as a template library mapping CIO from application objects, command and input objects.

Although being more and more rich and complex, we claim that selection rules still suffer from five drawbacks :

1. the selection rules do not take plenty advantage of the full power of guidelines and ergonomical rules: selecting one type of CIO may not be appropriate in another specific context;
2. the selection rules do not rely on dialog, user and screen models as suggested in [1]: complex interactive tasks, unexperienced or expert users, visual density have no influence on CIO selection;
3. the selection rules always work on CIO because they are all environment dependent: one selection rule may not work with others environments;
4. the selection rules are not observable by the designer: though some design tools allow the modifiability of the rules, all of them do not offer any mean to the designer to see the CIO selection process working;
5. the organization of selection rules is mostly formal (*if...then* rules): no graphical representation is provided.

To improve points 1-3, we restarted a full study of selection rules. Guidelines were merged from multiple style guides: the Apple Human Interface Guidelines, the Hewlett-Packard Interface Guide, the IBM Common User Access Style Guide, the OSF/Motif™ Style Guide [11] and the Sun Open Look™ Guide. They were fully rewritten in terms of AIO in a consistent and non-redundant format. Others environment dependent guidelines, namely for Apple // by Toggnazzini [16] and for OSF/Motif™ by Kobara [10], have been transformed in the same way. Precise others selection rules were added from expert ergonomical rules: Arens [1], Brown [6], Card *et al.* [7], Shneiderman [14].

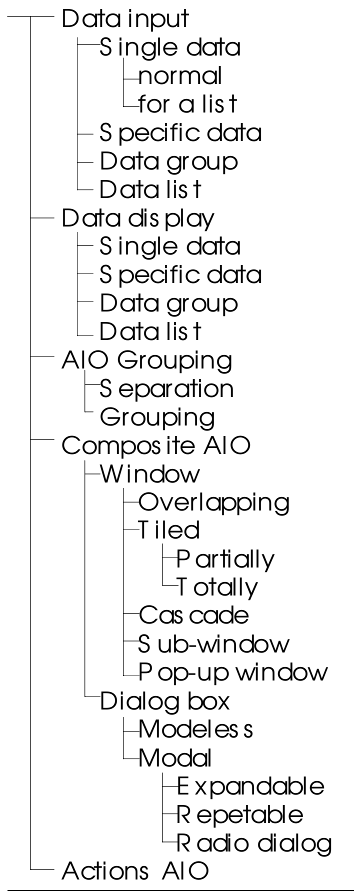


Figure 3. Hierarchy of TRIDENT selection rules

Five sets of rules select simple, composite AIO (fig. 3): rules for selecting simple AIO for data input, simple AIO for data display, static AIO to separate or group (un)related data, composite AIO depending on ERA, FCG and actions AIO.

Sets 1,2 transform a hierarchical data structure into a hierarchy of AIO with (not yet) undetermined root. The data structure supports not only simple attributes, but also group and list of attributes; arrays, trees or others complex data structures are assumed to be equivalent to organizations of groups and lists, like list of groups or group of groups. Sets 1,2 have been splitted for input and display because more suited AIO can be selected for only displaying the same data than for just inputting them on the screen. UIDE [2] also separates editing content (input) and read-only content (display). Set 3 is responsible for introducing separators, group boxes, prompts,... to concretize logical semantic proximity of AIO by surrounding, delimiting and/or separating to improve user guidance.

Set 4 selects the root object which is a composite AIO and/or any sub-composite AIO used in the hierarchy. For example, tiled windows are preferred for simultaned tasks and overlapping windows, for concurrent tasks, pop-up windows for minimal sub-tasks. Radio-dialog boxes simulate exclusive tasks in the FCG, expandable dialog boxes support tasks with frequent and rare sub-tasks, repetitive dialog boxes simulate reiterative tasks with same data. Set 5 introduces dedicated push buttons from the structure of the FCG. For example, "`Ok`" and "`Cancel`" buttons, "`Help`" button for complex task, "`More...`" button for expandable dialog, "`Default`" or "`Reset`" for specific data input or with default values.

INTELLIGENT AUTOMATIC AIO SELECTION

Selection rules generally contain four information sources: data from the application model (all), information on these data (e.g. *metadata* in UIDE [2]), others parameters (as in Selectors [8]), user preferences (as in DON [9]). TRIDENT employs four information sources contained in ERA and CFG databases (fig. 1).

First, every application data is specified as a quadruplet

$$d = (V, dt, nvc, dv)$$

where V is the set of all possible values of d (if known); dt belongs to one of the implemented data types (hour, date, logical, integer, numeric, real, alphabetic, alphanumeric); nvc is the number of values to choose (1 for simple choice, N for multiple choice); $dv \in V$ is the default value of d (if known). V is partitioned as $V = PV \cup SV$ where PV is the sub-set of principal values and SV, the sub-set of secondary values. This characterization allows a future presentation of most frequently used values first, minimizing input and scrolling. The number of principal values and secondary values are given by $npv = \# PV$ and $nsv = \# SV$. Maximum data length is given by $l = max(length\ (v_i))\ \forall\ v_i \in V$.

Second, five metadata refine each data when relevant: *granularity*- whether the data precision is low, moderate or high; *known values*- whether V is well defined; *ordered list*- whether the values of V are sorted according a particular order (numerical, alphabetical, logical, temporal, physical, by frequence); *expandable list*- whether the user can add new vales to V and/or modify V; *continous range*- whether the values are contained in a continous range or interval. If there is some order in the values of V, then an appropriate AIO is selected (e.g. a spin button for chronological data, a scale for a bounded integer, a list box for logically ordered alphabetical data).

Third, one parameter specifies *constrained display space*- whether the selection must consider screen density to avoid too large AIO. The more constrained is the screen space, the less large AIO is selected (e.g. a drowp-down list box in place of a complete list box).

Level	User experience
3	beginner
4	novice
5	intermediate
6	expert
7	master

Table 2. User experience level

Fourth, a common user model is clarified by a *user experience level* provided by Card *et al.* [7] (table 2) and by a *user selection preference* (low or high)- whether the use is more skilled to select one value among a list of values or to enter it with the keyboard. The less is the user experience level, the more sophisticated is the guidance provided by the object (e.g. a profiled edit box for an expert, a list box for an intermediate user, a sequence of spin buttons for a beginner).

AIO SELECTION BY DECISION TREE

To solve points 4-5 above and to really produce intelligent AIO selection, rules must be organized in a proper way which is modifiable and observable. UIDE's [2] and DON [9] selection rules are organized as *if...then* rules contained in a file editable by the designer. Drawing a flowchart benefits to the designer: this organization seems obvious (assimilating rules is easy), concise (the organization is intrinsically kept short and non-redundant) and rapid (following a path is fast). Nevertheless, the flowchart fails to show the entire space of possible values covered by each metadata.

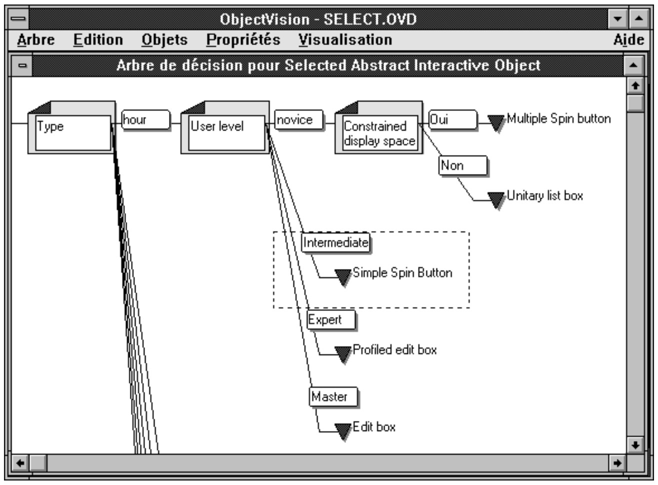


Figure 4. TRIDENT's AIO Selection tree (partial view).

TRIDENT so adopted a decision tree technique showing at each node the possible values space as a partition. A *decision tree* graphically depicts a decision logic with arcs and nodes. Each node consists of a current AIO selected and a set of nodes. Each node states a simple logical condition (e.g. is Level=5?); all arcs starting from a node cover the space of possible values (e.g. Level<5, Level=5, Level>5 in fig. 4; *nvc*=1 for a simple choice, *nvc*>1 for a multiple choice). The first arcs concern the different data types (hour, date, boolean,...). The *selection tree* represents the AIO selection logic where two kinds of nodes are reached :

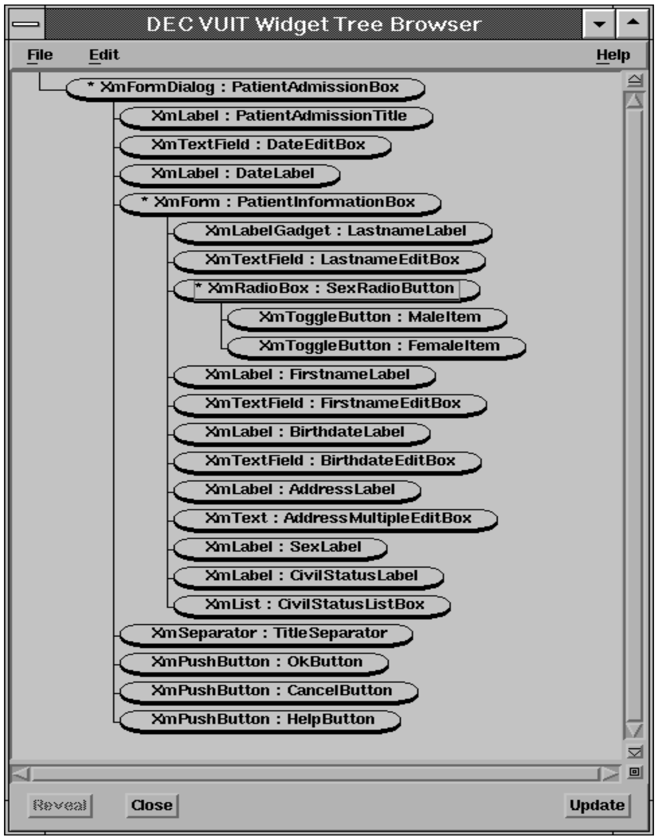
1. *branching nodes*: node evaluating only one parameter to determine the path to follow at lower level and providing the AIO of the current level;
2. *conclusion nodes*: leaf node furnishing final AIO after evaluating the whole decision logic.

AIO on conclusion nodes are supposed to be the most appropriate interaction objects, if all involved parameters are known.

ADVANTAGES AND INCONVENIENCES

Practical experience showed that decision tree has benefits:

- *visibility* : the designer is able to clearly visualize why a particular AIO was choosen progressively and to understand the result of each selection rule; the tree representation has a great influence on the effort needed to implement selection rules and to follow them;
- *easy backtracking* : applying an inappropriate rule may delay successful user interface generation; because the user interface development is iterative, the user can try a rule and, if it later discovers that this rule was inappropriate, can go back and try another instead by using other selection rules;
- *easy reasoning explanation* : trees are useful for keeping track of the intermediate application of rules;
- *fast selection* : this results from a breadth-first ordering because expansion of nodes is similar at each level; this search is guaranteed to find a shortest-length path to a conclusion node, if such a path exists (the completeness of selection rules is to be verified);
- *high modifiability* : identical sub-trees can be duplicated, new or others AIO can be added to tailor the tree to local conventions and user preferences;
- *iterative refinement* : because each node is associated to an AIO, the designer can stop the selection at every node, go back to the previous, select another rule.



```
* XmFormDialog : PatientAdmissionBox
    XmLabel : PatientAdmissionTitle
    XmTextField : DateEditBox
    XmLabel : DateLabel
    * XmForm : PatientInformationBox
        XmLabelGadget : LastnameLabel
        XmTextField : LastnameEditBox
        * XmRadioBox : SexRadioButton
            XmToggleButton : MaleItem
            XmToggleButton : FemaleItem
        XmLabel : FirstnameLabel
        XmTextField : FirstnameEditBox
        XmLabel : BirthdateLabel
        XmTextField : BirthdateEditBox
        XmLabel : AddressLabel
        XmText : AddressMultipleEditBox
        XmLabel : SexLabel
        XmLabel : CivilStatusLabel
        XmList : CivilStatusListBox
    XmSeparator : TitleSeparator
    XmPushButton : OkButton
    XmPushButton : CancelButton
    XmPushButton : HelpButton
```

Figure 5. CIO hierarchy in the presentation editor.

On the other hand, some disadvantages have been observed:

- *rule redundancy* : some identical rules are duplicated at different places in the tree because their application is relevant at different states;
- *excessive size* : the designer prefers trees with small values spaces at each node so that it is vital to reduce the conditions; an explicit tree representation can become impractical for very large selection trees.

MAPPING AIO TO CIO

Once matching AIO hierarchy has been mapped from all data structures, this AIO hierarchy is transformed to CIO hierarchy which is target environment dependant (fig. 5). Different AIO to CIO matching tables are defined. After transformation, CIO hierarchy can be captured as entry of a direct manipulation presentation editor.

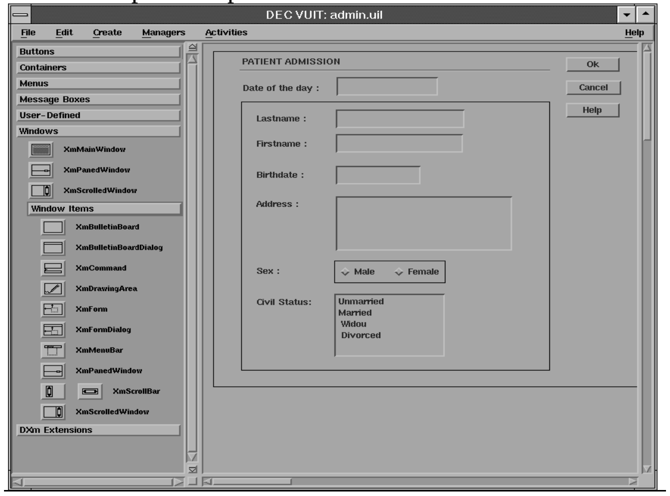


Figure 6. Resulting interface in the presentation editor.

The resulting interface can then be modified by the designer, the human factors specialist using DECVuit™ (*Visual User Interface Tool*) editor in the case of OSF/Motif™ (fig. 6). During this phase, the designer can replace behaviour-equivalent CIO (semantic integrity must be preserved), move or shift CIO whithin the tool to improve presentation, decoration and to respect physiological human limits. This modification is no longer made according to selection rules.

CONCLUSIONS

A decision tree-based AIO selection technique has been introduced to solve several problems and to induce new features which are greatly appreciated by the designers. In order to improve the range of selection rules, many *metadata* were defined and used so that this number becomes the major inconvenience to implement reasonably scaled trees. It seems that is the price to pay when enriching selection rules.

Selection rules are AIO-based rules for data management and business-oriented applications as in insurance and banking companies for instance. Graphical application may draw attention to the AIO typology, but their scope is actually not covered by this work (it was not the intended purpose). The eight supported data types with their metadata is the only investigation of the semantic for the selection rules. This not claims that all important aspects of business applications that affect the interface can be characterized by these informations. In fact, others parts of the semantic are passed to the interface generator in the CIO Placer: functional dependencies, integrity constraints, domain constraints,...

The current selection tree only requires forward chaining, but a future work concerns its extension to fully backward chaining: having a given AIO, which are the selection rules we need to apply, and from which type of data? This question occurs when the final user imperative requires to use a particular AIO which is different from the AIO selected by the tree.

On the other hand, DEC*Vuit* is now under extension for opening actual UIL object hierarchy to home-maded object classes: some interesting CIO are separately implemented and imported (e.g. group box, profiled edit box, option box, simple and extended tables,...)

ACKNOWLEDGMENTS

The auhors would like to thank all members of the TRIDENT project Anne-Marie Hennebert, Jean-Marie Leheureux, Isabelle Provot, Benoît Sacré (especially) and Louis Simon for their collaborative and participative research work. This work was supported by the FIRST (Formation et Impulsion à la Recherche Scientifique et Technologique) research program of "Direction Générale des Technologies et de la Recherche du Ministère de la Région Wallonne", Ref. RASE/SCHL319/Conv. 1487 and by the "Informatique du Futur" project of "Service de la Politique et de la Programmation Scientifique" under contract N°IT/IF/1.
Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the view of the Belgian Government. Copyright © 1993 F. Bodart, Jean M. Vanderdonckt.

REFERENCES

[1] Arens, Y., Miller, L. and Sondheimer, N., "Presentation design Using an Integrated Knowledge Base", in Sullivan, J. and Tyler, S. (eds.), *Intelligent User Interfaces*, Reading, MA: Addison-Wesley, 1991, pp. 241-258.

[2] de Baar, D., Foley, J.D. and Mullet, K.E., "Coupling Application Design and User Interface Design", *CHI'92 Conference Proceedings*, Monterey, May 1992, pp. 259-266.

[3] Bodart, F., Hennebert, A.-M., Leheureux, J.-M. and Pigneur, Y., "Computer-aided specification, evaluation and monitoring of information systems", *Sixth International Conference on Information Systems Proceedings*, Indianapolis, 1985.

[4] Bodart, F. and Pigneur, Y., *Conception Assistée des Systèmes d'Information*, Paris: Masson, 1989.

[5] Bodart, F., Hennebert, A.-M., Leheureux, J.-M., Provot, I., Sacré, B. and Vanderdonckt, J., "The TRIDENT project (Tools foR an Interactive Development ENvironmenT)", Working Conference IFIP WG 2.7, Namur, September 1990.

[6] Brown, C.H., *Human-Computer Interface Design Guidelines*, Berkeley, Ablex Publishing Corp., 1988.

[7] Card, S.K., Moran, T.P. and NEWELL, A., *The Psychology of Human-Computer Interaction*, Hillsdale, NJ: Lawrence Erlbaum Assoc., 1985.

[8] Johnson, J., "Selectors: Going Beyond User-Interface Widgets", *CHI'92 Conference Proceedings*, Monterey, May 1992, pp. 273-279.

[9] Kim, W. and Foley, J., "DON: User Interface Presentation Design Assistant", *UIST'90 Conference Proceedings*, Snowbird, October 1990, pp. 10-20.

[10] Kobara, S., *Visual Design with OSF/Motif*, Hewlett-Packard Press Series, Reading, MA: Addison-Wesley, 1991.

[11] Open Software Foundation, *OSF/Motif*™ *Style Guide*, revision 1.0, Englewood Cliffs, NJ: Prentice Hall, 1990.

[12] Petoud, I. and Pigneur, Y., "An Automatic and visual approach for user interface design", in *Enginnering for Human-Computer Interaction*, Amsterdam: North-Holland, 1990, pp. 403-420.

[13] Provot, I. and Sacré, B., "Proposition d'un langage de spécification de l'interface homme-machine d'une application de gestion hautement interactive", research report, Fac. Univ. N.-D. de la Paix, Namur, December 1991.

[14] Shneiderman, B., *Designing the user interface: strategies for effective human-computer interaction*, Reading, MA: Addison-Wesley, 1987.

[15] Szekely, P., Luo, P. and Neches, R., "Facilitating the Exploration of Interface Design Alternatives: The HUMANOID Model of Interface Design", *CHI'92 Conference Proceedings*, Monterey, May 1992, pp. 507-515.

[16] Tognazzini, B., *The Apple // Human interface guidelines*, Cupertino, CA: Apple Computer, 1985.

[17] Vander Zanden, B. and Myers, B.A., "Automatic, Look-and-Feel Independent Dialog Creation for Graphical User Interfaces", *CHI'90 Conference Proceedings*, Seattle, April 1990, pp. 27-34.

[18] Wiecha, C., Bennett, W., Boies, S. and Gould, J., "Generating Highly Interactive User Interfaces", *CHI'89 Conference Proceedings*, Austin, May 1989, pp. 277-282.