



Institut d'Informatique

Rue Grandgagnage, 21
B-5000 Namur
BELGIUM

Knowledge-Based Systems for Automated User Interface Generation: the TRIDENT Experience

Jean Vanderdonckt

RP-95-010

March 1995

Knowledge-Based Systems for Automated User Interface Generation : the TRIDENT Experience

Jean Vanderdonckt

Institut d'Informatique, Facultés Universitaires Notre-Dame de la Paix,
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 - URL: <http://www.info.fundp.ac.be/~jvd/index.html>

Key words: Computer-Aided User Interface Design, Interaction Object, Model-based Approach, Multi-windowing, Presentation Unit, Task-based Approach, Window.

1. INTRODUCTION

In this paper, we report over experience gained within TRIDENT project [2,3]. TRIDENT (Tools foR an Interactive Development Environment) consists of both a methodology and a supporting environment for developing highly interactive business oriented applications. It includes several tools using knowledge-based techniques to automatically generate a user interface for this particular class of applications, i.e. tools that automatically:

1. suggest to a designer a single interaction style or several combined interaction styles;
2. select appropriate interaction objects from functional and operational specifications;
3. lay out interaction objects using multiple placement strategies that resulted from the previous step;
4. identify windows for a predefined interactive task according to a task model;
5. provide a guideline document for a particular design situation from task parameters and design options.

For each of these questions, a short definition of the problem of concern is given and a description of the knowledge-based systems supporting them is provided.

The strengths and weaknesses are then discussed with respect to evaluation of these techniques performed with first versions of the systems. Finally, we discuss how these systems are evolving and how they are planned for the future. Rather than describing how these various techniques have been implemented in software tools (for this purpose, we refer to [34] for full details), we will focus on the way these techniques have changed and have been augmented by additional features in order to decrease their weak-

nesses. Knowledge-based techniques that are listed above have two common characteristics:

1. they all work on knowledge bases containing ergonomic rules (i.e., guidelines, design rules, recommendations). However, we will see how these various rules have been coded and used within the tools.
2. they attempt to provide ergonomic guidance at design time as much as possible rather than performing ergonomic evaluation after a user interface has been designed.

2. AUTOMATIC SUGGESTION OF INTERACTION STYLE

2.1 Definition of the Problem

Interaction styles are not numerous since they basically consist of a list of eleven items : natural language, command language, query language, questions & answers, menu selection, form filling, function keys, multi-windowing, direct manipulation, iconic interaction, multimedia interaction. For a particular task and one or many types of users, a designer can choose either one single isolated interaction style (e.g., form filling) or multiple combined interaction styles working together (e.g., form filling with function keys and command language). This problem is addressed in [22], section 3.1, and in [21]. This problem was especially addressed for query interactive tasks in [14].

IF	pre-requisites ARE moderate	AND
	productivity IS high	AND
	objective environment IS existent	AND
	structure IS low	AND
	importance IS high	AND
	complexity IS low TO moderate	
THEN	possible interaction style	IS command language
IF	task experience IS rich	AND
	system experience IS high	AND
	motivation IS high	AND
	complex interaction media experience IS high	
THEN	possible interaction style	IS command language
IF	processing type IS mono-processing	AND
	process capacity IS low	
THEN	possible interaction style	IS command language

Figure 1. Examples of production rules.

2.2 Knowledge-Based Techniques

The software includes **production rules** contained in **matching tables** suggesting a pool of possible interaction styles by deriving them from three categories of parameters resulting from context analysis [2] :

1. task parameters : pre-requisites, productivity, objective task environment, environment reproducibility, task structure, task importance, and task complexity;
2. user stereotypes parameters : task experience, system experience, motivation, complex interaction media experience;
3. work place description parameters : processing type, process type.

Figure 1 shows some examples of production rules that are imbedded in the system. For each class of parameter, the systems asks the right value for the interactive task in concern (fig. 2). Once all answers have been provided to all required questions, the small expert system automatically suggests a list of one or many interaction styles (fig. 3) according to the production rules coded. For example, the first production rule of fig.1 is coded as a first-order predicate formula (fig. 4). The list of possible interaction styles is progressively refined one class after another.

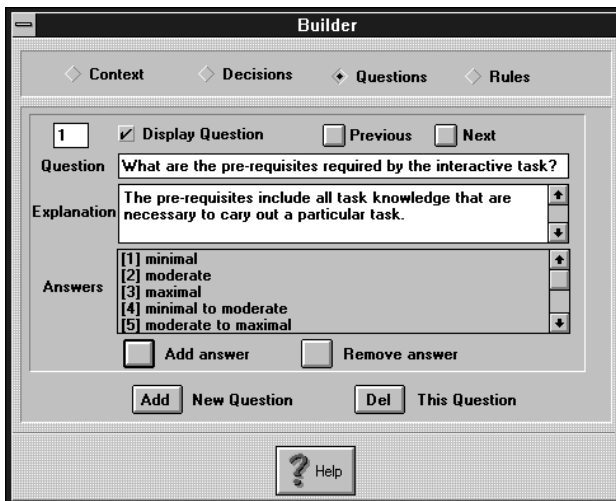


Figure 2. Possible answers for each parameter.

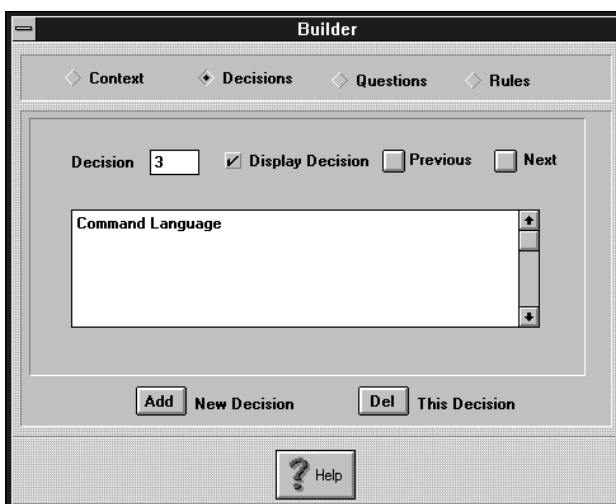


Figure 3. A possible interaction style.

2.3 Discussion

The first steps conducted with this approach highlighted several shortcomings:

- it is not always possible to provide a strict, categorical value for each required parameter since some tasks can combine multiple profiles simultaneously (e.g., being somewhat important at the beginning, but crucial at the end);
- the set of production rules is intrinsically incomplete not because some rules have been unfortunately forgotten but because empirical research has not yet investigated all the conclusions for all the possible values;
- even for a particular domain problem, it seems hard to produce a clear taxonomy of interactive tasks. Tasks are always reduced to a predefined set of parameters.
- the matching tables tested on commercially available software showed that conclusions sometimes needed to be revisited completely according to an emphasis placed on the available technology first, on the task after, and on the user finally. This may be due to the area of business application [32].

2.4 Evolution

The system today evolves to a derivation of interaction styles based on a **multi-valuated fuzzy logic** so that the parameters can be represented with multiple weighted values (e.g., high structure for 60% of actions, weak structure for the rest) resulting to conclusions with multiple success scores (e.g., menu selection with appropriateness score of 75%, command language with 20%, natural language with 5%).

Though most designers tend to solve this problem manually or according to common practice or style guides, they are more likely to use the system if

- they can tailor the fuzzy logic rules to their local conventions (e.g., add a new production rule, edit an existing rule, change the weight of a rule);
- they are not forced to key in a lot of parameters (i.e., the work load of the system does not exceed the human thinking).

What they most like is the mean they can justify their choices by referring to the relevant production rules. The system is also augmented by a technique that **minimizes standard deviation** between parameters values provided by designers and values which are contained in production rules. This case occurs when no actual values are matching the possible values.

Though the system displays the possible values for each parameter, it is rather "passive" in the sense that a designer is forced to provide all the parameters values before receiving suggestion for one or many interaction styles. The production rules are always static and do not evolve with the design's experience and knowledge, unless the expert system engineer maintains the rules consistently with the design issues.

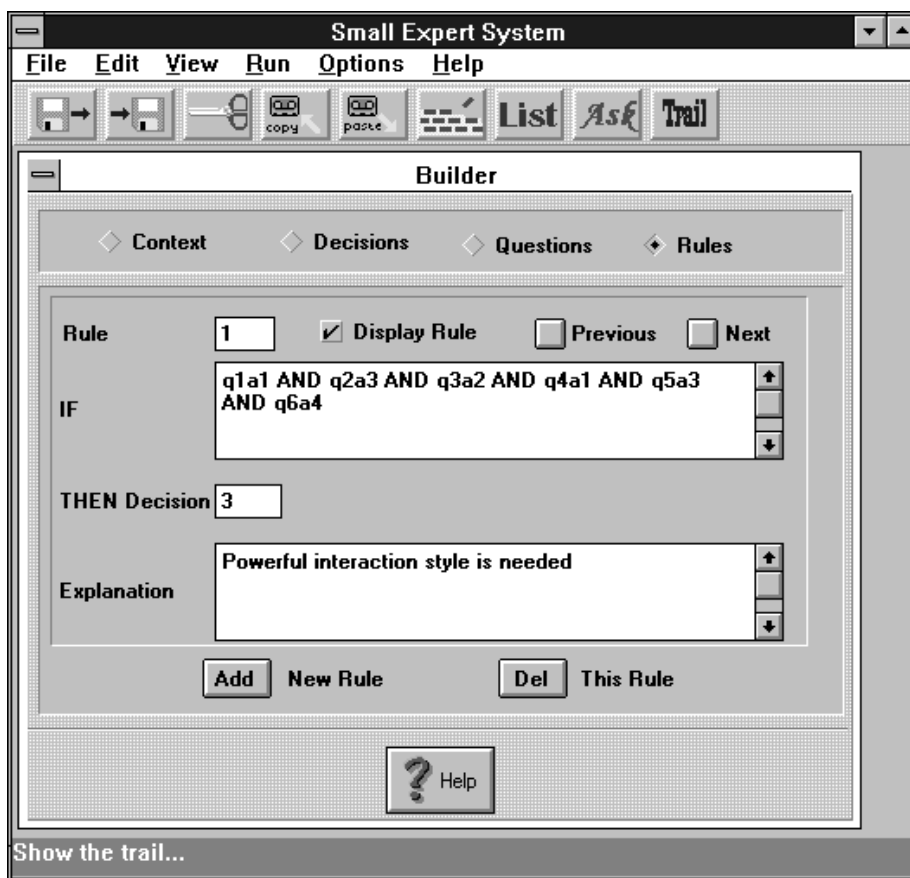


Figure 4. Production rule coded in the small expert system.

3. AUTOMATIC SELECTION OF INTERACTION OBJECTS

3.1 Definition of the Problem

Automatic generation of user interface inevitably raises the question of which interaction objects (IO) (e.g., an edit box, a combination box, an icon, a tool box) should materialize application information and actions. This problem is fully described in [6,8,10,25]. Ergonomic rules for selecting IOs can be found in style guides (e.g., IBM CUA [17]), standards, design guides (e.g., [31]), or some empirical studies (e.g., [24]). A complete corpus of selection rules for choosing IOs is given in [27].

3.2 Knowledge-Based Techniques

A first prototype of the supporting system was grounded on several **algorithms** for selecting IOs from a series of functional and operational specifications (e.g., data type, data length, number of possible values, domain definition, expandability by user). These specifications are partially depicted in TRIDENT in an object-oriented entity-relationship model and partially specified with a specification language called DSL (Dynamic Specification Language). As these algorithms were completely imbedded within the system, they were rendered completely opaque. As ergonomic rules were coded by algorithms that vary from one data type to another, they were virtually unmaintainable.

A second version introduced a **set of selection rules** contained in an expert system. There was a clear need for identifying each possible value for each specification to avoid multiple conditions (e.g., data type = alphanumeric

AND data type <> integer OR ..., THEN interaction object = list box) that are hard to manipulate by designers. The system was able to select a single IO out of twenty-five different IOs from seventeen parameters for nine supported data types. Therefore, the production rules have been made canonical, leading to a better understanding and customization of rules, but also leading to three shortcomings :

1. rule redundancy : the canonicalization of production rules implies redundancy because a same ergonomic rule can be found at different situations;
2. lack of visibility and follow-up : one salient feature of expert system is their ability to fully explain their reasoning. Executing production rules textually is not very representative, expressive for designers;
3. exceeding of specification work : the more specifications are provided, the more

appropriate the selected IO will be. But designers dislike to be forced to specify all details of each information or action before the automatic selection. It is too constraining and unrealistic.

3.3 Discussion

Whereas rule redundancy seems impossible to avoid, the two last problems have been addressed as follows :

1. the lack of visibility of production rules invited us to order rules into **decision tables** that can be graphically represented by **decision trees** techniques. The production rules in the expert system do not need to adhere to particular execution order since the flow of control is not determined by the order in which the selection rules have been coded. This is no longer the case with decision tables and decision trees. Decision trees [11] illustrate search in a **state-space representation** where a preferred order of rule processing is imposed. This order might not be optimal in all cases. But the fact that the selection for an appropriate IO through a decision tree can be illustrated graphically is far most appreciated by designers. Each node represents a state where a current IO is selected, and the links represent a change from this state to another representing a more ergonomic IO. This change results from the examination of the current value of the information specification.
2. this version of selection was rather automatic and straightforward since a IO was selected from all the specification contained in a repository. Designers' attitude was passive. To make them more participative,

11:07 18/10/93 SelectVision - INPUT.OVD Disk C: 15.4 Mb

File Edit Form Field View Help

Selection d'objet interactif abstrait [Complete]
Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique

Selection of an Abstract Interaction Object to input an elementary data

IDENT

Data Name Style Name		Values	
Data Type integer	Length 15	Number of values to choose : 2 from 10	
		Number of principal values : 10	
		Number of secondary values : 0	

User Experience Level
☐ Beginner ☐ Novice ☐ Intermediate
☐ Expert ☒ Master

Numerical Level : **7** Tm **50**
 High Screen Density ☐ Low ☒ High

Input Preference ☐ Selection ☐ Typing

Domain **known**
 Parameters
☐ Expandable Domain
☐ Continuous Domain
 Precision ☒ Low ☐ High

Orientation
 Undefined

Selected Abstract Interaction Object
Selection list

Figure 5. Control panel for computer-aided selection of interaction objects.

it was suggested not to force them to input all required specifications before generation. Instead, they provide minimal specification for each information (fig. 5). From this, the software automatically selects a first proposal. According to the current state in the decision tree, the software becomes more interactive by asking designers one question at a time. The different possible answers, corresponding to the allowed future paths in the decision tree, are presented.

When the designer provides an answer, not only a more ergonomic IO is selected but the corresponding specification is added in the repository. Designers are more likely to see the direct impact of multiple specifications. Therefore, the decision tree could be explored with forward chaining, backward chaining, or bi-directional chaining. As long as the designer wants to proceed, a more ergonomic IO is selected.

objects (CIO, [25]), the system is based on an abstraction of behaviour, leaving presentation aspects outside. This abstraction is called an Abstract Interaction Object (AIO, [25]). Figure 7 details the abstraction aspects for the Push Button AIO. All AIOs are of course arranged in a hierarchy (fig. 8) according objectoriented definitions. Thus, decision rules (fig. 9) contained in the decision tree no longer work on particular instances of interaction objects belonging to different environments, but rather select an AIO.

3.4 Evolution

Further evaluation of these techniques showed that designers rapidly acquire the ergonomic rules contains in the knowledge bases. They sometimes know the right IO even before going through the decision tree. But, another problem emerged. Though extensible, decision trees

- deliver a predefined traversal,
- could only select an IO which is already reachable by selection rules,
- is based on ergonomic rules that select absolute results.

At the leaf nodes of the decision tree, ergonomic rules can be added but they are competitive or conflicting. Therefore, a relative selection should be performed if we want to increase the expressiveness of the selection technique.

Today, we are trying today to use **theory of argumentation** [18,23] to extend this power. At these stages, the software suggests multiple solutions. Each solution is argued in the sense that it

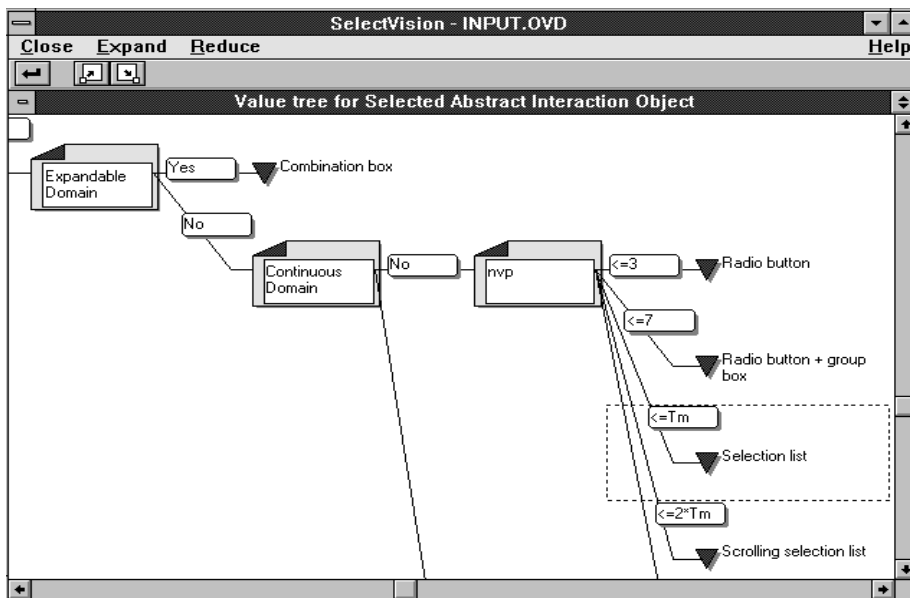


Figure 6. Graphical Decision tree.

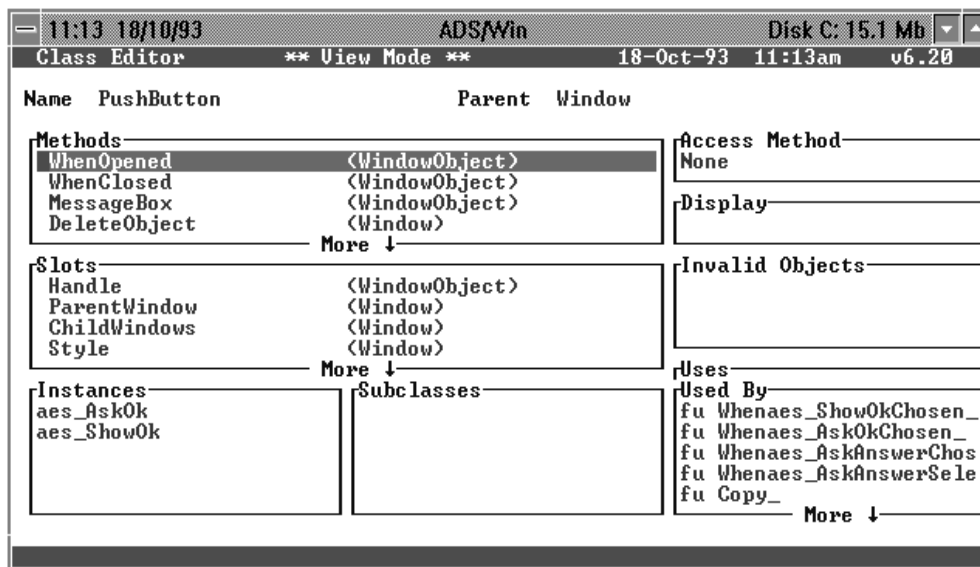


Figure 7. Object-oriented representation of an Abstract Interaction Object.

preserves some cited ergonomic criteria (e.g., compatibility, consistency, dialog control, work load, adaptiveness) but violates some other.

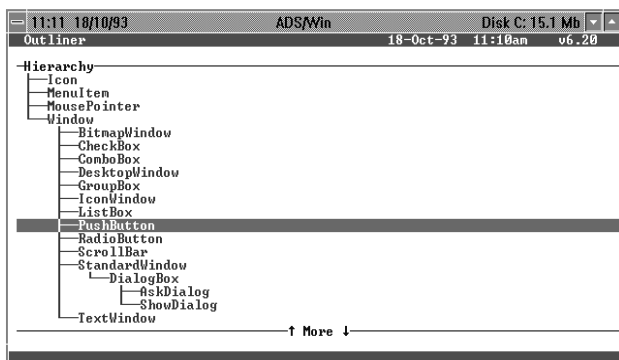


Figure 8. Hierarchy of Abstract Interaction Objects.

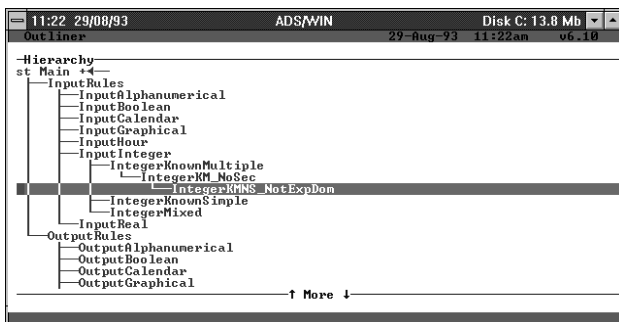


Figure 9. Hierarchy of decision rules.

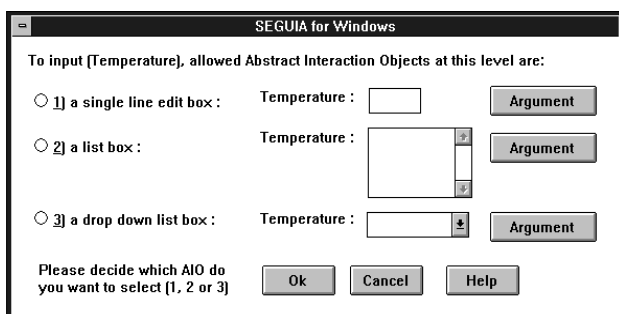


Figure 11. hierarchy of abstract interaction objects.

Figure 10 exemplifies a case where the designer has to select an AIO to input/display the temperature of the human body. It shows the logical steps through which the user may pass before reaching a final solution. On each step, a more appropriate AIO is selected by privileging a particular criterion among several dimensions (e.g., ergonomic criteria, performance, representativeness).

The argumentation displays the pro and contra for selecting this or that IO (fig. 11). The designer

is consequently responsible for the selection, but (s)he is aware of strengths and weaknesses of each choice. The trade-off could therefore be supported, but arguing all steps, even in the narrow scope of computer-aided selection, represents a tremendous amount of work.

We conclude that including such a high number of rules in a system is unable to automatically generate a usable user interface without designer intervention. Moreover, the real ergonomic IO are reached at the end of decision tree where convention, user preference, technology availability, ergonomic criteria have to be considered to get a usable result [32].

4. AUTOMATIC PLACEMENT OF INTERACTION OBJECTS

4.1 Definition of the Problem

Once IOs have been selected in order to input/display application's information, they need to be laid out in a more encompassing IO (e.g., a window, a dialog box, a panel). The usability of the resulting composite IO heavily depends on the way its IOs are placed by taking into account visual design, aesthetics, user preference, task performance, clarity of layout, consistency,...

4.2 Knowledge-Based Techniques

Having a hierarchy of abstract interaction objects [25], the goal is to transform them into concrete interaction objects with their complete coordinates within the composite IO. The first investigated strategy for complete automatic placement, called **two-balanced column strategy**, was based on a generic grid to filled in by IOs following visual continuity. This strategy is quite static and passive since it renders full automatic placement : the designer triggers the strategy and receives the output. It is based on the assumption that IOs in the hierarchy are already arranged in a sequence to be reproduced in the Window. The IOs are equally shared among two balanced columns according to a book metaphor. This strategy is fully described in [1].

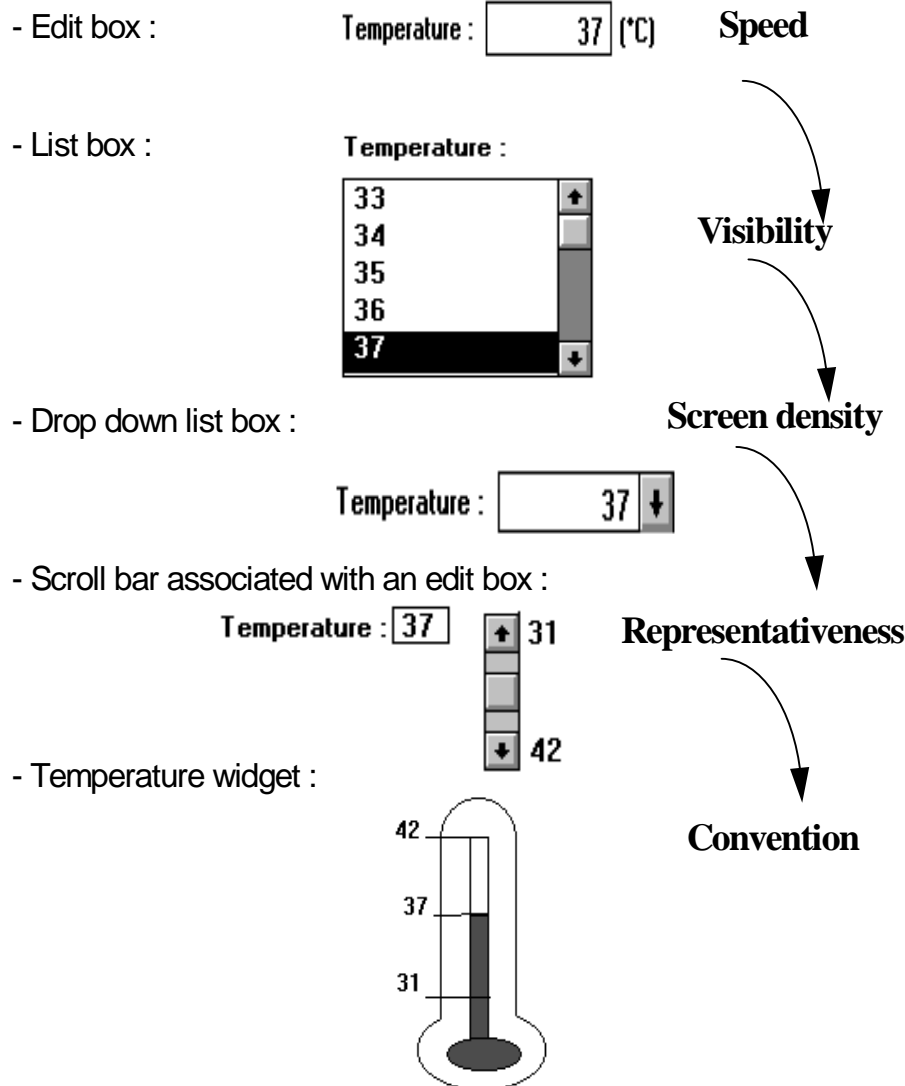


Figure 10. Example of the temperature.

4.3 Discussion

Testing this strategy identified that it was appropriate for form filling user interfaces with moderate amount of IOs [29]. However, flexibility is almost non existent (the strategy is qualified static) since the solution is always unique and is not controlled by designer. Moreover, this strategy is believed to generate unused blank screen regions in order to preserve visual continuity and consistency, to minimize fragmentation.

The lack of designer intervention and the rigidity of the unique solution lead us to consider another strategy, called **Right/Bottom strategy** [1]. The assumptions are the same as above, but rather than placing all IOs through one pass, the strategy displays the first IO in the sequence. The second IO can be placed either at the right or underneath the previous one, depending on relationships and dimensions. The strategy proceeds with these two alternatives for all other IOs to the end of the sequence. The strategy consequently induces a **binary tree of placement alternatives** which is traversed in a computer-aided way. This is why this strategy is qualified dynamic (fig. 11). At each step, the designer is able to

see the two alternatives (fig. 12 and 13) before deciding which one to choose (fig. 14). According to **placement heuristics**, the strategy suggests (fig. 14) the designer which alternative seems the most promising. These heuristics are based on relationships (e.g., justification, centering, uniformity, equilibrium, proportional/total equilibrium) [29]. They attempt to suggest the alternatives that maximizes these relationships and to avoid branches of the binary tree that lead to unpleasant, unappealing layouts. For this purpose, "cut branches" heuristics have been defined. Fig. 17 shows the final solution recommended for the same case as above.

This strategy has been proved much more flexible (2^{n-1} solutions for n IOs - # unappealing solutions) and more active since it let the designer to visually decide which alternative to choose at each step. Though active, the software also provides guidance namely by suggesting the most likely alternatives, by removing all unappealing solutions. Application of this Right/Bottom strategy greatly reduces the

amount of unused spaces, is able to manage large amount of IOs, but increases screen fragmentation.

In a more general study [29], we compared six strategies along several dimensions and mathematical relationships with respect to three points of view : the designers' point of view, the human factors expert's point of view, and the user's point of view.

We concluded that no strategy is able to produce a usable layout in all cases, even if a lot of ergonomic rules have been embodied in the system as for the Right/Bottom strategy [29]. But it always remains at least one strategy that produces a good starting point for future refinement. This conclusion is partially explained by the fact that each strategy automatically preserves some ergonomic criteria but also inevitably violates other criteria. Satisfying all criteria is impossible [32].

For example, our two-balanced column strategy preserves visual continuity, consistency, equilibrium, fragmentation, but is not flexible, static, not active and consumes a lot of screen space (reducing screen density).

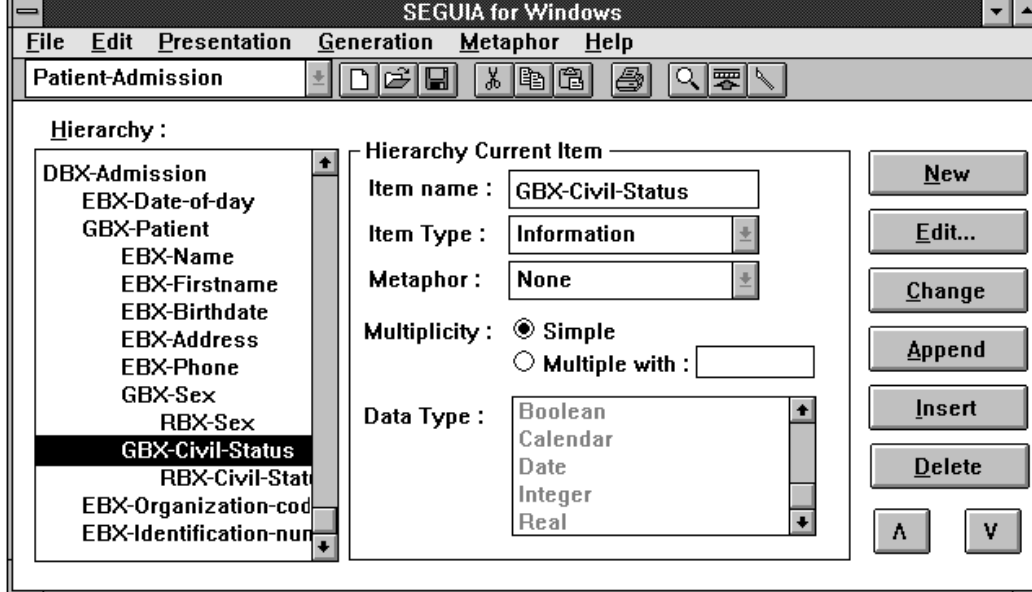


Figure 11. Control Panel displaying the source for placement generation.

Figure 15. Final result.

Figure 12. Bottom placement alternative.

Figure 13. Right placement alternative.

Figure 14. Active control by designer.

4.4 Evolution

Our Right/Bottom strategy also preserves visual continuity, equilibrium, is flexible, dynamic, active, but increases fragmentation and screen density. The conclusion is therefore that the designer has to prioritize the ergonomic criteria (s)he wants to focus on. Thus, we are evolving to **multi-criteria strategies** where one or many strategies that are compatible with these criteria will be suggested.

Another evolution that could be considered is to release the constraint that selection and placement activities are separated. Conversely, they could work hand in hand by combining Right/Bottom strategy at each step for placement and by allowing multiple IOs to be placed. Once again, each solution should be argued.

In the same way, rather than constructing again part of user interface that are similar, existing templates could be automatically retrieved from a template base according to **pattern-matching techniques**. For example, one particular set of information (e.g., personal information) could be materialized by several metaphors : a form filling, a graphic presentation, a multimedia presentation.

For this last interaction style, we found that traditional placement strategies are no longer valid since they assume that IOs are rectangular. In multimedia user interfaces, IOs can be of any other shape : triangular, circular,... We think that **visual techniques** coming from the area of visual literacy should replace traditional placement strategies in this specific area [28,30].

5. AUTOMATIC IDENTIFICATION OF WINDOWS

5.1 Definition of the Problem

This problem arises typically in automated generation of user interface. Once information and actions have been defined for a particular interactive task, they need to be divided into one or several windows that are inter-related or not. The question is : how to distribute information and actions into windows?

5.2 Knowledge-Based Techniques

In a recent study [4], we show that **model-based approaches** solve this problem easily by deciding the repartition on the underlying model. For instance, GENIUS [13] splits information and action according to links between entities in an entity-relationship model. Mecano [19] distributes information and actions according to domain relations between classes in a domain model. The resulting windows are always unique and restricted to the same category of interactive classes of task (e.g., input, display, list, browse). The underlying knowledge is therefore limited and passive.

On the other hand, **task-based approaches** rely on a task model in order to separate information and actions into windows. They are therefore able to support various interactive tasks, not just a restricted set. ADEPT [35] uses a **hierarchical tree** of task decomposed into sub-tasks to actions and information. Information and actions

are first grouped on equal tree levels and merged into windows from one level to another as long as the window is not overcrowded. Though this stopping criteria is physic and not semantic, the reliance on task analysis is guaranteed. But identification of windows is not flexible nor active, partially motivated by the wish of a high degree of automation.

TRIDENT [3] also belongs to task-based design trend in the sense that a task model is used to identify windows. One of the results of task analysis is a model describing the interactive task the user has to carry out which can be graphically represented with an **activity chaining graph** (ACG). An ACG depicts an information flow between function chained to define a business logic compatible with the task's goal (fig. 15).

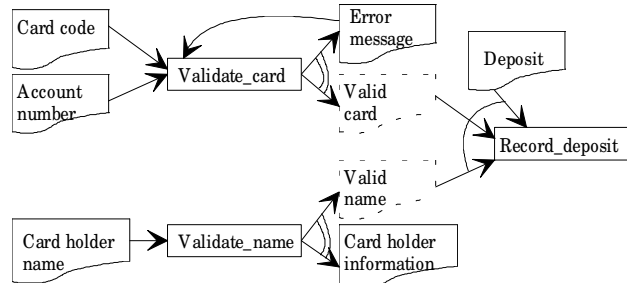


Fig. 15. Example of an Activity Chaining Graph.

5.3 Discussion

From the point of view of graph theory, an ACG is an oriented 1-graph. Nodes represent either information or action, whereas links represent either input/output or chaining. Links are augmented by graphical notation expressing logical conditions (i.e., AND, OR, XOR). Various graph algorithms based on the configuration of the ACG have been proposed [4] for different interaction styles. Each algorithm is based on one of the following identifications :

1. *maximal identification*: only one window is identified by covering all external input/output informations for all functions present in the ACG. This identification is particularly suited from small or medium information set which are highly inter-related or for which it is important to provide a wide overview of all informations together. Wmax covers external input information (i.e., Card code, Account number, Deposit, and Card holder name) and external output informations (i.e., Error message and Card holder information) (fig. 16).

Make a deposit

Card code :

Account number :

Card holder name :

Deposit :

OK

Cancel

Message area

Fig. 16. Maximal identification (Wmax).

2. *minimal identification*: a window is identified for each external information: Wmin1 for Card code, Wmin2 for Account number, Wmin3 for Deposit, Wmin4 for Card holder name, Wmin5 for Error message, Wmin6 for Card holder informations (fig. 17).

Fig. 17. Minimal identification (Wmin1 to Wmin6).

3. *input/output identification*: a window is identified for all input information of all functions having external informations and another window for all output information: Wio1 for both input informations of Validate_card (i.e., Card code and Account number), Wio2 for input information of Record_deposit (Wio2=Wmin3), Wio3 for input information of Validate_name, Wio4 for output information of Validate_card (Wio4=Wmin5), Wio5 for output informations of Validate_name (Wio5=Wmin6) (fig. 18).

Fig. 18. Input/output identification (Wio1, Wio3).

4. *functional identification*: a window is selected for all information of all functions having external informations. Wfc1 for informations of Validate_card (i.e., Card code, Account number, Error message), Wfc2 for informations of Record_deposit (Wfc2=Wmin3), Wfc3 for information of Validate_name (Wfc3=Wio3) (fig. 19).

Fig. 19. Functional identification (Wfc1).

5. *free identification*: a window is identified for every information set resulting from a partition of the ACG information set. This partition can be made arbitrarily. For example, Wfr1 covers Card code, Account number, Deposit, and Error message; Wfr2 covers input/output of Validate_name (Wfr2=Wio3) (fig. 20).

Fig. 20. Free identification (Wfr1).

Grouping techniques have also been introduced, based on a stopping criteria in order not to exceed cognitive load of each window. Each grouping is proposed to the designer so that (s)/he can accept or refuse such groupings, making the decision more active.

Again, these techniques can produce a moderately acceptable identification for the cases that have been investigated so far. But they are not very knowledge-intensive since they are only based on syntactic information contained in the ACG.

Up to now, we never saw a much more elaborated technique because, in order to make a significant breakthrough, the solution should encompass semantic aspects which are hard to formalize and for which designers are not likely to make much more specification than often required in projects.

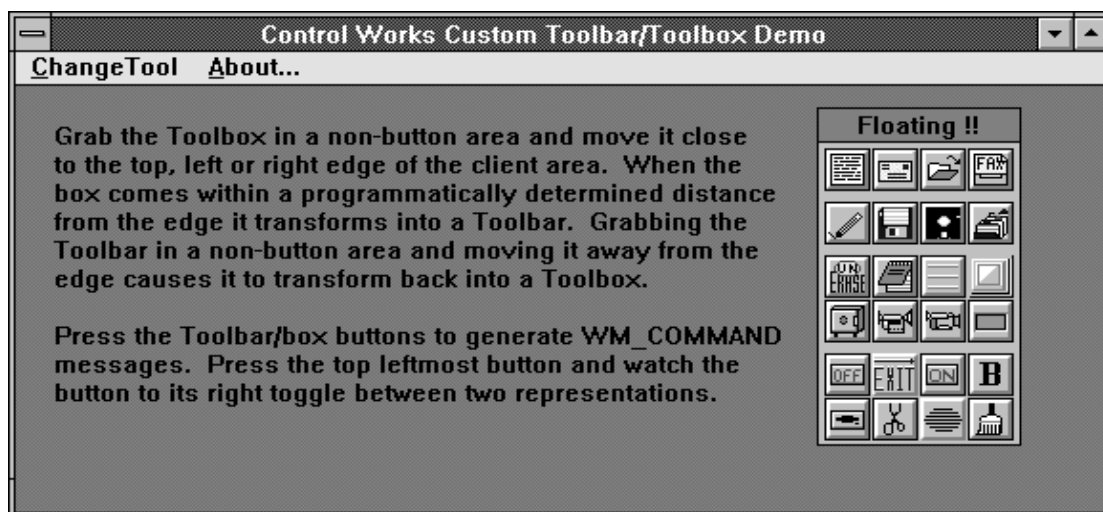
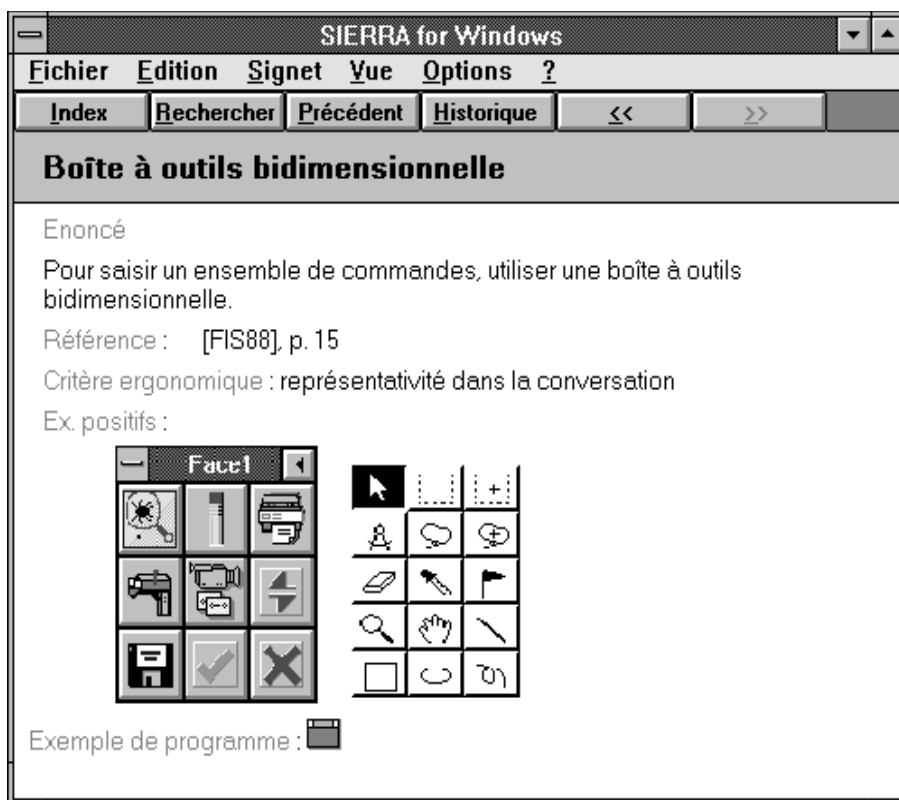


Figure 21. On-line tool for accessing guidelines.

5.4 Evolution

It seems hard to improve significantly our identification based on ACG. But its ability of flexibility lets it open to other aggregation techniques, other stopping criteria (e.g., loops, undo arrows, complexity metrics for each task). Finally, this work should be integrated with other design options related to multi-windowing, e.g., window type (pop-up/secondary/multiple), chaining (modal/modeless), display (overlapping/tiling).

6. AUTOMATIC GENERATION OF A GUIDELINE DOCUMENT

6.1 Definition of the Problem

According to some studies, general ergonomic rules are not very useful for designing user interfaces, but specific rules are. The idea is to produce as automatically as possible a dedicated guideline document containing specific

ergonomic rules for a particular interactive task. Figure 21 highlights an example of a guideline that have been extracted and included automatically in an on-line document. This hypermedia can be browsed, searched, listed, printed,...

6.2 Knowledge-Based Techniques

We are currently investigating a **derivation technique** for gathering specific ergonomic rules which can be summarized as follows [9] :

- a series of task parameters (as above) is given;
- specific ergonomic criteria are derived from task parameters;
- specific ergonomic rules are derived from specific ergonomic criteria and some design options (e.g., interaction style).

6.3 Discussion

The problem of retrieving ergonomic rules heavily depends on the ways they are organized. The problem is that a variety of ways to represent ergonomic rules comes from artificial intelligence, no single representation being all-encompassing and definitive. The object is to choose a right knowledge representation of ergonomic rules.

This may extend to combining different representations within a single system or even to developing a new representation. The chosen representation structures the problem domain. The problem itself can then be worked out using the representation, various operators, and control techniques. Ergonomic rules can be organized in several ways :

- according to functional areas (e.g., data input, data display, dialog control, data transmission) as in the Smith & Mosier document [22];
- according to interaction objects (e.g., control objects, menu bar and pull down menus) [26] as in the IBM CUA style guide [17];
- according to ergonomic criteria (e.g., compatibility, consistency) as in the Scapin design guide [20];
- according to linguistic levels (e.g., semantic, syntactic, lexical) [5] as recommended by Marcus [15];
- according to a predefined taxonomy of tasks (e.g., in query tasks);
- according to the time they are used in the development life cycle (e.g., at requirement time, at design time, at programming time, at evaluating time);
- according to the background of people who have to use them (e.g., designers, psychologist, human factors expert);
- according to high-level factors, dimensions (as in ISO [12]), or heuristics [16];
- ...

Because of this multiplicity of possible organizations, it remains impossible to choose only one knowledge representation that best fits into all cases [32]. In our design guide [31], ergonomic rules have been organized by user interface areas (e.g., information input, visual design, user guidance), by ergonomic criteria and by linguistic level successively. We believe that this organization minimizes gaps between goal and what is available for most cases. This organization can accommodate various demands.

This leads us to consider **semantic networks** for including ergonomic rules knowledge into a software. Of course, semantic networks can be illustrated by diagrams consisting of nodes and arcs. The nodes represent ergonomic rules. The connecting arcs, called links, represent the relations between ergonomic rules. For this reason, we implemented this knowledge as a hypermedia, called SIERRA [33] (fig. 21) with no graphic representation, but with a precise definition of all links : "generalizes", "specializes", "is similar to", "is not similar to", "is conflicting with", "is more important than", "is less important than",...

Links can therefore be one to one, one to many or many to one relationships, one way or two-way. Accessing guidelines information easily through semantic networks is influenced by the quality and completeness of links according to the various users demands [33]. Implementing all these links, which is the key to success, represents a huge time-consuming activity, even automatically.

6.4 Evolution

Such a system could only be usable if it is flexible enough to tailor various demands and local needs. Therefore, customizability of semantic networks (e.g., reducing or enlarging link's scope, introducing typed links, deleting unnecessary links) is highly important.

7. CONCLUSION

With these five examples, we attempted to argue that no unique approach involving a unique knowledge-based technique could solve a particular problem, even in a narrow range of tasks and applications. A multi-strategy approach should be definitely be considered and used. It is the responsibility of our future to understand where and why which technique should be preferred instead of another one. To meet this tremendous challenge, long experimental validation can be performed.

REFERENCES

1. Bodart, F., Hennebert, A.-M., Leheureux, J.-M. and 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, 1994, pp. 78-87.
2. Bodart, F., Hennebert, A.-M., Leheureux, J.-M., Provot, I. and Vanderdonckt, J., *A Model-based Approach to Presentation: A Continuum from Task Analysis to Prototype*, in Proceedings of Eurographics Workshop on Design, Specification, Verification of Interactive Systems (Bocca di Magra, 8-10 June 1994), F. Paterno (Ed.), Eurographics Series, 1994, pp. 25-39.
3. Bodart, F., Hennebert, A.-M., Leheureux, J.-M., Provot, I., Zucchinetti, G. and Vanderdonckt, J., *Key Activities for a Development Methodology of Interactive Applications*, Chapter 4 in Critical issues in User Interface Systems Engineering, D. Benyon and Ph. Palanque (Eds.), Springer-Verlag, 1995.
4. Bodart, F., Hennebert, A.-M., Leheureux, J.-M., and Vanderdonckt, J., *Computer-Aided Window Identification in TRIDENT*, in Proceedings of the Fifth IFIP TC13 Conference on Human-Computer Interaction INTERACT'95 (Lillehammer, 27-29 June 1995), S.A. Arnesen & D. Gilmore (Eds.), Chapman & Hall, London, 1995.
5. Bodart, F., Lesuisse, R. and Vanderdonckt, J., *A Proposition for Layered Ergonomic Criteria in Design/Evaluation*, in Abridged Proceedings of Human-Computer Interaction International HCI'93, 5th Int.

- Conf. on Human-Computer Interaction jointly with 9th Symposium on Human Interface (Orlando, 8-13 August 1993), M.J. Smith, G. Salvendy (eds.), Elsevier Science Publishers B.V., Amsterdam, 1993, p. 19.
6. Bodart, F., Noirhomme-Fraiture, M., and Vanderdonckt, J., *Guidelines for Choosing Interaction Objects*, in Proceedings of Vienna Conf. on Human-Computer Interaction VHCI'93 "Fin de siècle" (Vienna, 20-22 September 1993), T. Grechenig and M. Tscheligi (eds.), Lecture Notes in Computer Science, Vol. 733, Springer-Verlag, Berlin, 1993, pp. 431-432.
7. Bodart, F. and Vanderdonckt, J., *Expressing Guidelines into an Ergonomical StyleGuide for Highly Interactive Applications*, in Adjunct Proceedings of INTERCHI'93 (Amsterdam, 24-29 April 1993), S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel and T. White (eds.), ACM Press, New York, 1993, p. 35-36.
8. Bodart, F. and Vanderdonckt, J., *On the Problem of Selecting Interaction Objects*, in Proceedings of 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.
9. Bodart, F. and Vanderdonckt, J., *Using Ergonomic Rules for User Interface Evaluation by Linguistic Ergonomic Criteria*, in Proceedings of HCI International'95 (Yokohama, 9-14 juillet 1995), Y. Anzai and K. Ogawa (eds.), Elsevier Science B.V., 1995.
10. de Baar, D.J., Foley, D.J., AND Mullet, K.M., *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.
11. Gettys, D., *IF You Write Documentation, THEN Try a Decision Table*, in IEEE Transactions on Professional Communication, PC-25, Vol. 2, 1982, pp. 86-90.
12. ISO/WD 9241 *Ergonomic requirements for Office Work with Visual Displays Units*, International Standard Organization, 1992.
13. Janssen, C., Weisbecker, A. and 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.
14. Jarke, M. and Vassiliou, Y., *A Framework for Choosing a Database Query Language*, ACM Computing Surveys, Vol. 17, No. 3, September 1985, pp. 313-370.
15. Marcus, A., *Graphic Design for Electronic Documents and User Interfaces*, ACM Press Books Tutorial Series, New York, 1991.
16. Nielsen, J. and Molich, R., *Heuristic Evaluation of User Interfaces*, in Proceedings of the Conference on Human Factors in Computing Systems CHI'90 "Empowering People" (Seattle, 1-5 April 1990), J. Carrasco, J. Whiteside (Eds.), ACM Press, New York, 1990., pp. 249-256.
17. Object-Oriented Interface Design: IBM Common User Access Guidelines, Document SC34-4399, IBM Corp., Publisher Que Corp., 1993.
18. Perelman, C. and Olbrechts-Tyteca, L., *The New Rhetoric. A treatise on Argumentation*, University of Notre Dame Press, South Bend, 1958.
19. Puerta, A., Eriksson, H., Gennari, J.H., and Musen, M.A., *Beyond Data Models for Automated User Interface Generation*, in Proceedings of 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.
20. Scapin, D.L., *Guide ergonomique de conception des interfaces homme-ordinateur*, Research report INRIA N°77, Institut National de Recherche en Informatique et en Automatique, Le Chesnay, October 1986.
21. Shneiderman, B., *A Taxonomy and Rule Base for the Selection of Interaction Styles*, in Human Factors for Information Usability, B. Shackel and S. Richardson (eds.), Cambridge University Press, Cambridge, 1991, pp. 325-342.
22. Smith, S.L. and Mosier, J.N., *Design guidelines for the user interface software*, Technical Report ESD-TR-86-278 (NTIS No. AD A177198), U.S. Air Force Electronic Systems Division, Hanscom Air Force Base, Massachusetts, 1986.
23. Toulmin, S., *The Uses of Argument*, Cambridge University Press, London, 1980.
24. Tullis, T.S. and Kodimer, M.L., *A Comparison of Direct-Manipulation, Selection, and Data Entry Techniques for Recording fields in a Table*, in Proceedings of the Human Factors & Ergonomics Society - 37th Annual Meeting "Designing for Diversity", Human Factors and Ergonomics Society (Seattle, 11-15 October 1993), pp. 298-302.
25. Vanderdonckt, J. and Bodart, F., *Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection*, 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 and T. White (eds.), ACM Press, New York, 1993, pp. 424-429.
26. Vanderdonckt, J. and Bodart, F., *An Object-Oriented Model for Organizing Guidelines in Human-Computer Interface Design*, in Abridged Proceedings of Human-Computer Interaction International HCI'93, 5th Int. Conf. on Human-Computer Interaction jointly with 9th Symposium on Human Interface (Orlando, 8-13 August 1993), M.J. Smith, G. Salvendy (eds.), Elsevier Science Publishers B.V., Amsterdam, 1993, p. 260.
27. Vanderdonckt, J., *A Corpus of Selection Rules for Choosing Interaction Objects*, Technical report 93/3, Institut d'Informatique, Namur, August 1993. Electronically available via anonymous ftp at

- arzach.info.fundp.ac.be [138.48.4.5] in /pub/papers/-jvd as Selection.ps.Z
28. Vanderdonckt, J. and Gillo, X., *Visual Techniques for Traditional and Multimedia Layouts*, 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, 1994, pp. 95-104.
 29. Vanderdonckt, J., Ouedraogo, M. and Yguetengar, B., *A Comparison of Placement Strategies for Effective Visual Design*, in Proceedings of 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. 125-143.
 30. Vanderdonckt, J., *Information Presentation for Multimedia Business Oriented Applications*, Paper presented at Workshop on Standardisation of Multimedia User Interface Design during meeting of ISO/TC 159/SC 4/WG 5 (Paris, 29th August - 2nd September 1994), 1st september 1994. Electronically available via anonymous ftp at arzach.info.fundp.ac.be [138.48.4.5] in /pub/papers/jvd as iso94.doc
 31. Vanderdonckt, J., *Guide ergonomique des interfaces homme-machine*, Presses Universitaires de Namur, Namur, 1994, ISBN 2-87037-189-6.
 32. Vanderdonckt, J. and Bodart, F., *Jusqu'au bout avec nos règles ergonomiques*, in Actes des Sixièmes Journées sur l'Ingénierie des Interfaces Homme-machine IHM'94 (Lille, 8-9 décembre 1994), pp. 231-236.
 33. Vanderdonckt, J., *Accessing Guidelines Information with SIERRA*, in Proceedings of the Fifth IFIP TC13 Conference on Human-Computer Interaction INTERACT'95 (Lillehammer, 27-29 June 1995), S.A. Arnesen & D. Gilmore (Eds.), Chapman & Hall, London, 1995.
 34. Vanderdonckt, J., *Automatic Generation of a User Interface for Highly Interactive Business-Oriented Applications*, Ph. D., Institut d'Informatique, Facultés Universitaires Notre Dame de la Paix, Namur, 1995.
 35. Wilson, S., Johnson, P., Kelly, C., Cunningham, J. and Markopoulos, P., *Beyond hacking: a model-based approach to user interface design*, in Proceedings of the HCI'93 Conference, Cambridge University Press, pp. 215-231, 1993.