A Small Knowledge-Based System for Selecting Interaction Styles

Jean Vanderdonckt [1,2,3]

[1] Université catholique de Louvain, Place des Doyens, 1
B-1348 Louvain-la-Neuve, Belgium
vanderdonckt@qant.ucl.ac.be
[2] Stanford University, Stanford CA 94305, USA
jeanvdd@cs.stanford.edu

Abstract. SDISelect consists of a small knowledge-based system teaching and assisting designers of interactive applications in selecting appropriate interaction styles for a particular context of use. As any other tool for working with guidelines, five development milestones have been browsed. Guidelines for selecting interaction styles have been captured in a knowledge-base system as selections among an available set of possible interaction styles, a set of parameters characterizing the context of use, questions to provide the parameters' values, and selection rules. Each rule selects a candidate interaction style candidate according to values assigned to parameters of the context of use according to a rule-based language, which basically consists of a first-order predicate logic formula. The values of these parameters are either stored or prompted to the designer through questions to form a final set of possible interaction styles.

1 Introduction

An *interaction technique* is usually referred to as any basic way in which users are interacting with the user interface (UI) of an interactive application (e.g., drag/drop, selecting, input/output). We shall use the term *interaction style* as any logical combination of such interaction techniques to organize the interaction into a consistent, recognizable and unified manner. The interaction style contributes to the *interaction genre* by expressing conceptual uses to which the technical means are put (e.g., interaction metaphor, agent character). Numerous interaction styles are available to select from, but each holds its unique dialog and presentation schemes.

1.1 The Design Option of Selecting Interaction Styles

It is the designer's responsibility to select interaction styles that satisfy the parameters of the context of use. Although this selection is seldom covered as an explicit design option in the development life cycle, it plays a major role in the UI final usability. At first glance, this design option may appear relatively trivial: the variety of interaction styles exists, but is finite and interaction styles mostly remain the same from one application to another in the same domain. Indeed, the selected interaction styles are

[3] J. Vanderdonckt is supported as Visiting Associate Professor by the Fulbright-Hayes Program and by NATO research fellowship under reference P61.90/A/02.01/99-911.

J. Vanderdonckt et al. (eds.), *Tools for Working with Guidelines*

likely the ones allowed by the toolkit (e.g., multi-windowing, menu selection) and old interaction styles (e.g., command languages, questions/answers) tend to vanish in applications, as they are no longer considered usable for end users. But at second glance, different reasons make this design option more complex than imagined:

- Design reasons: despite the variety of available interaction styles, the selection may fail to match the context of use, thus potentially decreasing the wanted usability. As task analysis informs this design option, its progressive nature may not produce simultaneously all information required for the selection, thus inviting the designer to think about them: in particular, they can tailor the task by refining it into task parameters.
- Methodological reasons: understanding how the different parameters influence the selection [24] and assessing how these parameters may engender conflictual selections are important issues though.
- Technical reasons: current tools offer little or no guidance with decision regarding the variety of interaction styles and their possible combination. Moreover, believing that there is a need to conform to the basic interaction styles of the computing platform or the toolkit is fallacious.

Our primary goal is therefore to seriously consider this design option by a method and a tool that supports this method.

1.2 A tool for Selecting Interaction Styles

SDISelect is another instance of a tool for working with guidelines. Goals of this research can be thus presented according to five development milestones of such a tool:

1. *Guidelines collecting*: the first task to accomplish is to collect guidelines for selecting interaction styles to become aware of any knowledge existing for this purpose, to gather as many knowledge on this topic as possible, and to process it.
2. *Guidelines organization*: as guidelines are extracted from heterogeneous sources with their own proprietary format, the need to organize them into a comprehensive and consistent framework rapidly became obvious. It also turned out to be extremely useful to investigate how this knowledge can be systematically expressed to this framework and how it fits.
3. *Guidelines incorporation into approach*: as the selection of interaction styles can be clearly identified as a design option, it makes sense to develop a systematic method for deciding this design option, based on the framework.
4. *Guidelines operationalization*: to provide some supportive character of this strategy, a knowledge-based approach was adopted in order to make this knowledge explicit, persistent, analyzable and incremental. The knowledge was therefore captured in a repository as selections, questions on parameters of the context of use, values of these parameters as answers to these questions, and selection rules.
5. *Guidelines usage*: finally, it was intended to assess to what extent the method for of selecting interactions developed in step (3) can be supported by the tool imagined in step (4), manipulating the knowledge of step (2). The final goal was consequently to act as a test bed for the method and to demonstrate its feasibility.

The rest of this paper is structured as follows: sections 2 to 6 are corresponding to the above milestones to reach a tool for working with guidelines. Section 7 reviews related work by summarizing solutions investigated in other methods and by discussing what the differences are between them and the current method. Section 8 ends up with some shortcomings of this method and a research agenda to overcome them.

2 Guidelines collecting

Although our theoretical basis is yet unable to rigorously classify all interaction styles, they can be roughly divided into fourteen families of interaction based on the involvement they induce between the user and the UI, sometimes with sub-styles: command language, programming language, natural language, query language, questions/answers, function keys, menu selection (with instances like embedded menus, pie menus), form filling (with instances like simple forms, multi-tab forms, spreadsheets), multi-windowing, direct manipulation, iconic interaction, graphic interaction, multimedia interaction, and virtual reality.

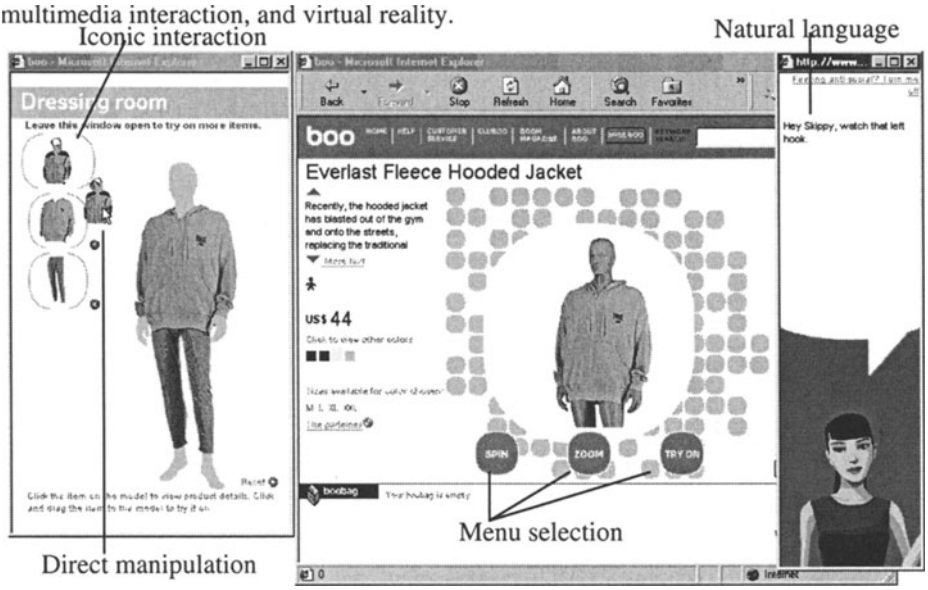

Fig. 1. Electronic commerce web site with multiple interaction styles (source: www.boo.com).

A UI assigned to a single interaction becomes more and more rare as the variations of the context of use are enlarging. Therefore, a UI is likely to have multiple interaction styles either one style at a time, depending on the circumstances, or several styles at a time. In particular, a same task can be carried out according to different interaction styles depending on the user, the platform, and other possible parameters.

For example, the task "place an order for clothes" can be carried out:

- Through function keys and questions/answers for an ATM-like ordering system.
- Through a phone-based system with voice recognition and speech synthesis based on a limited natural language to enable customers to order by phone.
- Through an operator-guided system with direct form filling, multi-windowing ca-

pabilities and menu selection to allow customer assistance.
- Through an electronic web site with direct manipulation, virtual reality, and iconic interaction to enable customers to "experience" the clothes before buying (Fig. 1).

Guidelines for selecting interaction styles are relatively numerous. Representative examples are [2,6,7,8,9,11,12,16,17,18,19,23]. Brown delivers isolated guidelines for selecting interaction styles based mostly on accumulated experience rather than empirical evidence [6]. When, how and why are they useful? Smith & Mosier report on guidelines that are based on task requirements and system response time (Table 1) [20]. Although today's computers have much faster response time, their reasoning about the user training is still valid. Bono and Ficorelli [5] showed how queries input with a graphic interaction style can be re-expressed in natural language. Eberleh *et al.* compared command language and direct manipulation from the viewpoint of mental workload required by the style and the user subjective preference for each [10].

Table 1. Estimated user training and response time for interaction styles.

Interaction style	Required User Training	Tolerable System Response Time
Question and Answer	Little/None	Moderate
Form Filling	Moderate/Little	Slow
Menu Selection	Little/None	Very Fast
Function Keys	Moderate/Little	Very Fast
Command Language	High	Moderate/Slow
Query Language	High/Moderate	Moderate
Natural Language	Moderate/Little	Fast
Graphic Interaction	High	Very Fast

Shneiderman describes interaction styles by emphasizing what their consequences are on the user and on the task by emphasizing the benefits and concerns of each style. These guidelines are expressed as rules in a framework (Fig. 2) based on information about the task domain, user skill level, and user skill level combined with frequency of use of the application [16,17,18]. The author agrees that respecting these guidelines can bring a certain presence of usability, but also that violating them does not necessarily bring a certain absence of usability.

```
IF modest knowledge of task domain with some computer skills
=> menu selection OR direct manipulation OR form filling.
```

Fig. 2. Some rules for selecting interaction styles.

Cristiano enumerated appropriate interaction styles for accessing a database [7]. Jarke and Vassiliou are more systematic as their framework for selecting an interaction style for querying a database is explicitly expressed in terms of user parameters, task parameters, and system parameters [13]. We kept that idea since such a knowledge can be progressively captured on the different circumstances where a single or multiple styles have been shown usable and its representation can be made explicit. In particular, they emphasize the difference between syntactical knowledge and semantic knowledge for users regarding to their tasks and to the system. Wexelblat is focusing on the selection of one particular interaction style: virtual reality [23].

3 Guidelines organization

Nowadays almost every designer has to first learn and understand what the parameters of the context of use are and then to decide on the best interaction styles possible. While deciding that design option, they have to take into consideration what the physically available styles are, what the style guide is recommending, what are the styles the user are familiar with. Considering this and using the framework principle of Jarke & Vassiliou [13], we focus on three sets of parameters: task parameters, user parameters, and environment parameters. These sets are defined in the following subsections with a global compromise on precision (sufficiently expressing parameters) and concisions (keeping them easy to learn and cheap to manipulate).

3.1 Task parameters

An interactive task is characterized by the following seven parameters:

- *Pre-requisites* (minimal, moderate, maximal): task prerequisites denote the amount of knowledge required to the user to properly carry out the task with the intended UI. In particular, this parameter may include the training needed to carry out the task. For instance, the prerequisites of an ATM should be minimal, whereas the UI for an air-traffic control system would surely be maximal.
- *Productivity* (low, moderate, high): task productivity is summing up the average amount of task accomplishments per unit of time. This parameter can also subsume the frequency of use (e.g., daily to two times per month). For instance, the productivity of a letter composition task in an insurance company is high for insurance producers (more than 50 letters a day). The productivity of a monthly reporting task is considered as low.
- *Objective task environment* (existent, non-existent): this parameter depicts the presence or the absence of domain objects manipulated in order to properly carry out the task. These objects include paper document, folders, and physical objects. For instance, a document verification task is primarily based on the document itself, whereas recording a product's price in a supermarket does not require the product itself, except maybe for scanning the bar code.
- *Environment reproductibility* (feasible, unfeasible): this parameter is a consequence of the previous one: reproducing the task environment is feasible or not provided that such an environment already exists. Environment reproducibility is said feasible if it is useful to represent domain objects as manipulable. For instance, the IBM RealThings system is intended to reproduce feasible objects that belong to our real world so that they can behave like in the real world (e.g., phone system, card filer). Conversely, reproducing into a UI the complete physical navigation in a library has been shown feasible, but not necessarily usable.
- *Task structuration* (low, moderate, high): this parameter expresses the degrees of freedom or constraints that the user has in carrying out the task. For instance, calculating the roots of a second-degree equation is highly structured since a deterministic algorithm governs the process. On the inverse, an advice-giving task for

loans may re-order subtasks according to currently available information, to the choices of the customer.

- *Task importance* (low, moderate, high): this parameter reveals the fundamental, crucial, if not vital, character of the task. For instance, setting up an alarm-system for a control room is considered highly important, whereas editing a simple statistical report is not.
- *Task complexity* (low, moderate, high): this parameter manifests the complexity degree of a task from the cognitive, articulatory, and intellectual viewpoints and skills that are required to properly achieve the task. For instance, a radar-tracking task is highly complex, whereas an advertisement composition is not complex.

Assigning values to these parameters is expected to encourage designers to observe, interview end user on site and report on their task as precisely as possible. Moreover, guidelines can be expressed as appropriate only for certain values of these parameters. Table 2 provides such expressions. In particular, due to the lack of guidelines regarding 4 interaction styles (i.e. programming language, virtual reality, graphic interaction, and multimedia interaction), these values have been assigned by introspection, which is sensible to variation.

3.2 User parameters

It is generally recognized convenient to decompose the whole user population into user stereotypes. A *user stereotype* gathers users repeating similar action patterns with respect to the same prescribed task definition. As they share a same induced task to become the same interactive task, they usually have comparable cognitive profiles and preferences [15]. Each stereotype is characterized by four parameters, which are varying for each stereotype regarding a same interactive task:

- *Task experience* (elementary, regular, rich): this parameter combines syntactic and semantic task knowledge. Syntactic knowledge often refers to the task allocation and its position in the complete chain, including terminology, whereas semantic knowledge refers to domain objects, actions and procedures embedded in the task. If a user decently integrated these from both an intellectual and practical point of view, then the task experience is said rich.
- *System experience* (elementary, regular, rich): this parameter gives an idea of the experience level required by technological means in order to carry out the task. These means may include computer, printer facilities, file management, word processing.
- *Task motivation* (low, moderate, high): this parameter translates the psychological attitude f the user with respect to the task. If the user is readily disposed carry out the task, then the motivation is said high. A constrained user has a low value.
- *Experience with modern interaction devices* (low, moderate, high): this parameter reflects how a user is apt to use one modern interaction device at a time or several one simultaneously. For instance, gesture recognition devices are considered to require some substantive experience from users.

Several other similar and equally expressive parameters can be imagined, but their increase may reduce the concision needed to work with the intended framework.

Table 2. Expression of interaction styles in terms of task parameters.

Interaction style	Prerequisites	Productivity	Objective environment	Reproductibility	Task structuration	Task importance	Task complexity
Command lang.	moderate	high	non existent	unfeasible	low	high	low to moderate
Programming lang.	maximal	low	non existent	unfeasible	low	low	moderate to high
Natural language	minimal	low	non existent	unfeasible	low	low	low to moderate
Query language	moderate	moderate	non existent	unfeasible	low	low	low
Questions/Answers	minimal	low	existent	feasible	high	low	low
Function keys	minimal	high	non existent	unfeasible	low to moderate	moderate	low to moderate
Menu selection	minimal	moderate	non existent	unfeasible	moderate to high	low	moderate
Form filling	moderate	moderate	existent	feasible	high	high	moderate
Multi-windowing	moderate	moderate	existent	feasible	low to moderate	high	high
Direct manipulation	minimal to maximal	moderate	existent	feasible	low	high	high
Iconic interaction	moderate	high	existent	feasible	moderate	moderate	low
Graphic interaction	moderate to maximal	moderate	existent	feasible	low	low to moderate	low to
Multimedia	low	low	existent	feasible	moderate	low to	moderate to high
Virtual reality	low	low	existent	feasible	low to moderate	low	low to moderate

Similarly to Table 2, guidelines can be expressed as appropriate only for certain values of these user parameters. Table 3 provides such expressions.

Table 3. Expression of interaction styles in terms of user parameters.

Interaction style	Task experience	System experience	Task motivation	Experience with modern interaction devices
Command Language	moderate to rich	rich	high	moderate
Programming language	rich	rich	rich	moderate
Natural Language	rich	moderate	low	high
Query language	rich	moderate	moderate	high
Questions/Answers	elementary	elementary to moderate	low	moderate to rich
Function keys	moderate to rich	elementary	low	low
Menu selection	elementary	elementary	low	low
Form filling	elementary to rich	elementary to rich	low to moderate	elementary to moderate
Multi-windowing	elementary	elementary	low	low
Direct manipulation	elementary	moderate	low	low
Iconic interaction	elementary to moderate	moderate	low to moderate	low
Graphic interaction	elementary	moderate	low to moderate	moderate
Multimedia interaction	elementary	moderate	low	moderate
Virtual reality	elementary	moderate	low	high

3.3 Environment parameters

Similarly here, several parameters describing the ambient conditions of task accomplishment, regarding or not computer facilities, can be exploited. We retain only two:

- *Processing type* (mono-processing, multi-processing): this parameter specifies whether the user can focus on one task at a time on the same workstation (mono-processing) or several tasks simultaneously (multi-processing). For instance, a same terminal can be dedicated to a predefined task most of the time, but it can still allow user to access other interactive applications in case of need.
- *Processing capacity* (low, moderate, high): this parameter considers the interruption, parallelism and concurrent aspects of tasks on a same workstation. For instance, if a user can suspend and resume a task, during this interval, other tasks may be operated, thus having a high processing capacity.

Similarly to Table 3, guidelines can be expressed as appropriate only for certain values of these environment parameters. Table 4 provides such expressions.

Table 4. Expression of interaction styles in terms of environment parameters.

Interaction style	Processing type	Processing capacity
Command Language	mono-processing	low
Programming language	mono-processing	low
Natural Language	mono-processing	low
Query language	mono-processing	low
Questions/Answers	mono-processing	low
Function keys	multi-processing	moderate
Menu selection	multi-processing	moderate to high
Form filling	multi-processing	moderate
Multi-windowing	multi-processing	high
Direct manipulation	multi-processing	high
Iconic interaction	multi-processing	high
Graphic interaction	multi-processing	moderate to high
Multimedia interaction	mono-processing	moderate
Virtual reality	mono-processing	low

Remark. Values set in Tables 2 to 4 are the most general possible. For example, in the second line of Table 2, the value "moderate" is assigned to the parameter "Prerequisites", meaning of course that the complete line is applicable if the task prerequisites are moderate, but also if the prerequisites are moderate or rich. Conversely, in line 7 of Table 2, the value "low to moderate" is assigned to the parameter "Task structuration", restricting the use to this interval only.

4 Guidelines incorporation into approach

To use the above framework, the method consists of performing the following steps:

1. Consider the three tables sequentially.
2. For each table,
   - Assign a value when possible to each descriptive parameter.
   - Identify *all* possible interaction styles.
   - Add a positive score for each possible style, a negative score to forbidden ones.
   - Keep possible interaction styles in an ordered list of candidates sorted by their score reflecting how many times their selection has been considered usable.
3. Select one or several styles from the resulting pool.

The experience of using this systematic method raised several questions.

What happens if values are changed? If we change the value of one or more parameters, we can notice the impact of the whole selection process without looking at the whole set of rules. This gives the designer with the opportunity to explore alternative styles and to check these alternatives for the design option for fit with the final UI.

Does the sequence of tables matter? The initial sequence reflect the hypothesis according which task parameters could prevail over user parameters, which in turn could prevail over environment parameters. This sequence needs to be revisited and

weighted for any case according to the requirements. For instance, if a requirement states the goal on an integrated workstation, it might be helpful to consider Table 3 before. If a task should be carried out by a stereotype of users with special needs, Table 1 could be relegated after Table 1. And so forth.

Does the method univocally select one interaction style? Not necessarily, since multiple lines can cover parameters having similar or close values, multiple styles may fit with the same usability. Moreover, if the same task should be carried out by many user stereotypes, it is likely that multiple interaction styles will appear.

Does the method select at least one interaction style? This depends on the guidelines expressed in the tables. In Tables 2 to 4, no completeness is guaranteed. A particular combination of values may probably select no interaction style, as there is no matching. In this case, interaction styles that minimize the deviation with respect to all lines can be selected. Or the values can be tuned to be closer to possible interaction styles.

Are selected interaction styles definitive? Since the method is a guideline-based approach, selected styles can be still revisited according to other criteria, but in principle, they can be applied by relying on the guidelines. The method may lay the design option with more confidence than merely with an "ad hoc" decision without any justification. An explicit reasoning may convince design stakeholders. Sometimes the guidelines are consistent (as in [13]), allowing a possible selection; sometimes they are inconsistent, thus yielding to a conflict between possible interaction styles.

4 Guidelines operationalization

The format of Tables 2 to 4 naturally leads to a guidelines operationalization as valued production rules. For example, Fig. 3 depicts the first line of Tables 2, 3, and 4.

```
IF   prerequisites          ARE moderate AND
     productivity           IS high      AND
     objective environment  IS existent  AND
     task structuration     IS low       AND
     task 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
```

Fig. 3. Examples of production rules.

The main requirements for a tool for working with these guidelines are its ease of use, its ability to represent the guidelines in a knowledge base, and its incorporation of a set of facilities for setting up the knowledge base, modifying it, supporting their execution, and the identification of multiple or conflictual or non existent solutions.

SDISelect, which addresses these requirements, is based on three components:

1. A knowledge base containing the different possible interaction styles (*selections*), the definition of task, user, and environment parameters for which a value is stored or prompted to the designer (*questions*) and the production rules that identify possible decisions according to the answers to the questions (*rules*).
2. A set of facilities for manipulating this knowledge base.
3. An inference engine which fires the production rules (according to short circuit optimization or decision depth optimization), guides the designer in the selection process and prompt for values when additional information is needed.

6 Guidelines usage

This section details how guidelines have been implemented as production rules in the system and how they are finally used. First, all possible selections are identified in a list (Fig. 4). Then, each parameter is associated with a related question (Fig. 5): if the value of the parameter is not already stored for the current case, this question will be asked to the designer to provide it when required. Each guideline, i.e. each line of Tables 2 to 4, is represented as a production rule of the form: $\wedge q_i a_i \Rightarrow s_j$ where the q_i are the different questions, a_i, the different answers, s_j, any possible selection. Fig. 6 represents the production rules related to the first line of Table 2. Once all answers have been provided to all required questions, SDISelect automatically suggests a list of one or many interaction styles according to the coded production rules. Any other type of first-order predicate logic formula can be written equally. When a possible interaction style has been identified, the user can still proceed with the selection to see whether other production rules can be fired according to the available parameters.

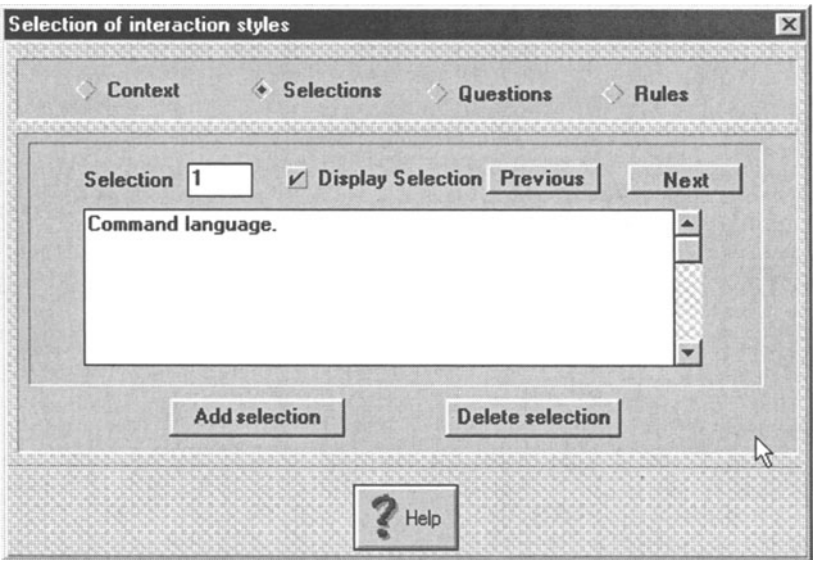

Fig. 4. Possible selections.

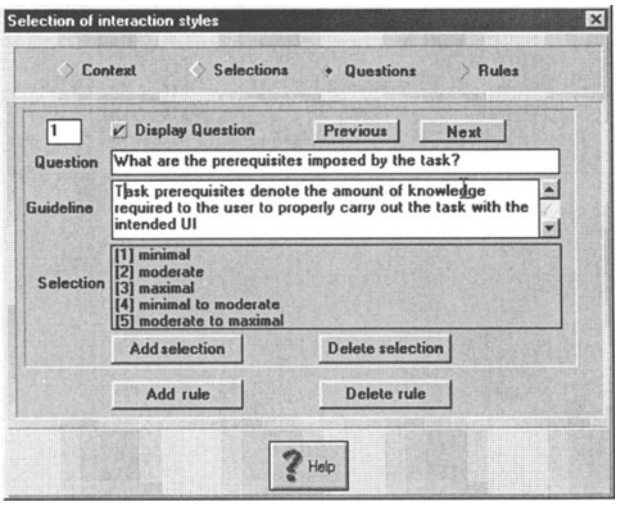

Fig. 5. Question about a parameter.

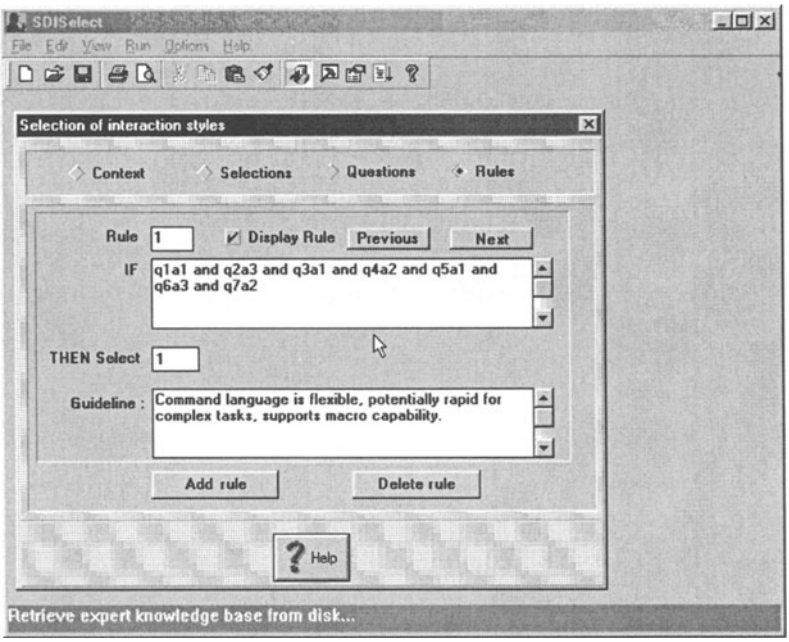

Fig. 6. Implementation of guidelines as production rules.

7 Related work

The closest tool to SDISelect is TOTO, a tool for selecting interaction techniques with a knowledge-based approach. Interaction techniques are more located at a fine-grained interaction level than the interaction styles, which are more at a coarse-

grained level, yet a clear relationship between them exists. Another major difference is that a special emphasis is given to the physical characteristics of input devices and the pragmatics of their use, which are not covered by SDISelect. An extensible object-oriented taxonomy of task concepts and physical actions is also provided.

MIDAS consists of a tool providing designers with methodological assistance to design user interface models before rushing to code [14]. It does not select interaction techniques or styles, but foster their exploration by reporting on their advantages and disadvantages. The techniques range from very basic ones (e.g., information selection) up to more sophisticated techniques (e.g., gesture commands).

SDISelect is very similar to ADEPT [25] regarding the selection of interaction styles, except that ADEPT is applying their guidelines to select abstract interaction techniques for domain objects. In this sense, they more focus on abstract interaction objects [22].

NIKL [1] and SCOPE [3] directly work on the selection of any interaction technique by coding them in a rule-based language. In this language, the interaction techniques are first selected at a logical level, then matched onto physical ones.

8 Conclusion and Future Work

The first steps conducted with this approach highlighted some shortcomings:

- Providing a strict, categorical value for each parameter is not always straightforward since some tasks can combine multiple profiles simultaneously or dynamically in time (e.g., being somewhat important at the beginning, but crucial at the end).
- The set of production rules is intrinsically incomplete not because some guidelines have been unfortunately forgotten but because empirical research has not yet investigated all the conclusions for all possible values.
- But the knowledge base is able to grow up as the experience is increasing: if a project was successful or unsuccessful in selecting one or many interaction styles, it would be interesting to capture this experience in the rules component. As the experience is increasing, the contents of the knowledge base can be refined, it is expected that the incompleteness may decrease, more conflicts can be clarified,...
- Even for a particular domain, it seems hard to produce a clear taxonomy of interactive tasks. Tasks are always reduced to a predefined set of parameters, which is always a restriction of its definition.
- 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 concern (e.g., information systems).

The system today evolves to a derivation of interaction styles based on a multivaluated fuzzy logic so that the parameters can be represented with multiple weighted

values (e.g., high structuration for 60% of actions, low structuration 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 ability to justify a selection by referring to the relevant guidelines. The system can also be augmented by a technique minimizing standard deviation between parameters values provided by designers and values that 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": 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, which can be done manually. An adaptation system that records any deviation and any final decision with respect to parameters would also be welcomed.

Acknowledgments

The author would like to thank Ms. Margaret Nicholson, of the Commission for Educational Exchange, Brussels (Belgium) and Mr. Christophe Goyvaerts, of the Ministère des Affaires étrangères, du Commerce extérieur et de la Coopératioon international au Développement, Brussels (Belgium) for their constant support during his visiting scholarship at Stanford University. The author also thanks the anonymous reviewers for their insightful comments.

References

1. Arens, Y., Miller, L., Sondheimer, N.: Presentation Design Using an Integrated Knowledge Base. Chapter 1. In Sullivan, J.W., Tyler, S.W. (eds.). Intelligent User Interfaces. ACM Press, New York (1991) 241–258
2. Banks, W.W., Gilmore, W.E., Blackman, H.S., Gertman, D.I.: Human Engineering Design Considerations for Cathode Ray Tube-Generated Displays. Vol. Ii. Document CR-3003/EGG-2230. U.S. Dept. of Energy, Idaho National Engineering Laboratory, Idaho Falls, Idaho (July 1983).
3. Beshers, C.M., Feiner, S.K.: SCOPE: automated generation of graphical interfaces. In Proc. of ACM Symposium on User Interface Software and Technology UIST'89 (Williamsburg, November 13-15, 1989). ACM Press, New York (1989) 76–85
4. Bleser, T.W., Sibert, J.: TOTO: A Tool for Selecting Interaction Techniques Interaction Techniques. In Proc. of ACM Symp. on User Interface Software and Technology UIST'90 (Snowbird, 3-5 October, 1990). ACM Press, New York (1990) 135–142
5. Bono, G., Ficorelli, P.: Natural Language Restatement of Queries Expressed in a Graphi-

cal Language. In Pernul, G., Jjoa, A.M. (eds.): Proc. of 11th International Conference on the Entity-Relationship Approach (Karlsruhe, October 7-9, 1992). Lecture Notes in Computer Sciences, Vol. 645. Springer-Verlag, Berlin (1992) 357–373

6. Brown, C.M.: Human-Computer Interface Design Guidelines. Ablex Publishing Corporation, Berkeley (1988)
7. Cristiano, L.: Methodology for Comparative Selection of Informative Database Interface Styles. SIGCHI Bulletin. Vol. 21, No. 1 (July 1989) 29–36
8. Draper, S.W.: Interface Styles. (May 12, 1996). Accessible at http://www.psy.gla.ac.uk/~steve/IntStyles.html
9. Dumas, J.: Designing User Interfaces for Software. Prentice Hall, Englewood Cliffs (1988)
10. Eberleh, E., Korfmacher, W., Streitz, N.A.: Thinking or Acting? Mental Workload and Subjective Preferences for a Command Code and a Direct Manipulation Interaction Style. International Journal of Human-Computer Interaction, Vol.4 No. 2 (1992) 105–122
11. Gaines, B.: The Technology of Interaction Dialogue Programming Rules. International Journal of Man-Machine Studies. Vol. 14 (1981) 13–150
12. Hutchins, E., Hollan, J.D., Norman, D.A.: Direct Manipulation Interfaces. In Norman, D.A., Draper, S.W.: User Centered Design-New Perspectives on Human-Computer Interaction, Lawrence Erlbaum Associates Publishers, Hillsdale (1986) 87-124.
13. Jarke, M., Vassiliou, Y.: A Framework for Choosing a Database Query Language. ACM Computing Surveys. Vol. 17, No. 3 (September 1985) 313–370
14. Luo, P.: MIDAS: A Model-Based Approach to Managing Conceptual Design of User Interface Software. In Paternò, F. (ed.). Proceedings of 1st Eurographics Workshop on Design, Specification, Verification of Interactive Systems DSV-IS'94 (Carrara, June 8-10 1994), Focus on Computer Graphics Series. Springer-Verlag, Berlin (1995) 129–147
15. Rich, E.: Stereotype and User Modeling. In: Kobsa, A., Wahlster, W. (eds.): User Models in Dialog Systems. Springer-Verlag, Berlin (1988) 35–51
16. Shneiderman, B.: We Can Design Better User Interfaces: a Review of Human-Computer Interaction Styles. Ergonomics, Vol. 31, No. 5, 1988, pp. 699-710.
17. Shneiderman, B.: A Taxonomy and Rule Base for the Selection of Interaction Styles. In Shackel, B., Richardson, S.: Human Factors for Informatics Usability. Cambridge University Press, Cambridge (1991) 325–342
18. Shneiderman, B.: A Taxonomy and Rule Base for the Selection of Interaction Styles. In Baecker, R.M., Grudin, J., Buxton, W.A.S, Greenberg, S.: Readings in Human-Computer Interaction: Toward the Year 2000. Morgan Kaufmann Publishers (San Francisco) 1995 401–410
19. Shneiderman, B.: Designing the User Interface: Strategies for Effective Human-Computer Interaction (3rd ed.). Addison-Wesley, Reading (1997)
20. Smith, S.L., 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 (1986)
21. Sutcliffe, A.G.: Human-Computer Interface Design. MacMillan, London (1988)
22. Vanderdonckt, J., Bodart, F.: Encapsulating Knowledge for Intelligent Interaction Objects Selection. In Proc. of ACM Conf. on Human Aspects in Computing Systems InterCHI'93 (Amsterdam, 24-29 April 1993). ACM Press, New York (1993) 424–429. Accessible at http://www.qant.ucl.ac.be/membres/jv/publi/InterCHI 93-Encaps.pdf
23. Wexelblat, A.: An Approach to Natural Gesture in Virtual Environments Special Issue on Virtual Reality Software and Technology. ACM Transactions on Computer-Human Interaction, Vol. 2 No. 3 (1995) 179–200
24. Wiedenbeck, S., Davis, S.: The Influence of Interaction Style and Experience on User

Perceptions of Software Packages. International Journal of Human-Computer Studies, Vol. 46 No. 5 (1997) 563–588

25. Wilson, S., Johnson, P.: Bridging the Generation Gap: From Work Tasks to User Interface Designs. In: Vanderdonckt, J. (Ed.), Proc. of 2nd Workshop on Computer-Aided Design of User Interfaces CADUI'96 (Namur, June 5-7, 1996). Presses Universitaires de Namur, Namur (1996) 77–94