Journal of Chromatography, 592 (1992) 51–57
Elsevier Science Publishers B.V., Amsterdam

CHROMSYMP. 2498

CRISEBOOK, a Hypermedia version of an expert system for the selection of optimization criteria in high-performance liquid chromatography

B. Bourguignon, P. Vankeerberghen and D. L. Massart*

Vrije Universiteit Brussel, Farmaceutisch Instituut, Laarbeeklaan 103, B-1090 Brussels (Belgium)

ABSTRACT

Expert systems built in shells such as KES suffer from a lack of easy maintenance, prototyping and a user-friendly man–machine interface. To solve these problems, the feasibility of using a Hypermedia approach was investigated. The reimplementation of the expert system CRISE, an object-oriented decision tree, in Toolbook was promising as a first application. The structure and features of the reimplemented system are described and similarities and differences with the KES version are discussed.

INTRODUCTION

Our laboratory has been involved in the development of several stand-alone expert systems for chromatographic method development [1–5], which were subsequently integrated in a complex expert system [6,7]. Both the stand-alone systems and the integrated system were developed with the aid of a shell. It was found that such expert systems present some problems [8].

One of these problems is the lack of easy maintenance. Expert systems must be easy to maintain. Especially in "first-guess" expert systems (systems that recommend the first mobile phase to be tried out [3,4], the user should be able to add new information and delete redundant knowledge as much as possible by himself. Indeed, technology changes and expert systems that are not updated would rapidly become obsolete. With the conventional present shells, it is difficult for the user to make changes owing to the way rules are connected to each other by such a system. Adding or changing rules could have repercussions all over the system. Moreover, some shells require advanced skills of knowledge engineering. This also implies that it is not easy to build a prototype without investing significant resources. Another problem concerns the man–machine or user interface, the part of the expert system that guides the user through the various steps of the consultation. Because most often the expert system functions as a dynamic information source for inexperienced occasional users, a user-friendly man–machine interface is very important. In KES, little attention is paid to the presentation and one must write a front end in C to make this more acceptable.

To solve these problems, we decided to examine Hypermedia tools. One of the first applications of Hypermedia tools in the domain of chemical analysis was proposed by Farkas *et al.* [8]. They developed SpecTool, a software package having Hypermedia features, as a help for the interpretation of NMR, MS, IR and UV–VIS spectra. Their system functions primarily as an information source and not as an expert system because it does not make decisions.

HYPERMEDIA

Hypermedia is the combination of multimedia and hypertext. In a hypertext document the user does not have to read all information in a sequential

0021-9673/92/$05.00

way, but is guided, according to his needs and interests, to pieces of information that are linked to each other. Multimedia is a collection of tools producing graphics, sound and animation, to present data in a more flexible way. The prototype of a software construction set with Hypermedia features is Hypercard, which runs on a MacIntosh PC. Hypercard combines the properties of both the relational database and an authoring system. By applying the relational capabilities, links can be created between individual files so that one can access from one file related information in another file. The distinction between Hypercard and a traditional database management system is the way of displaying information, which is provided by the authoring system. This system includes many kinds of tools such as painting, creating fields and buttons which are fundamental to build an efficient user interface. Hypermedia applications use object-oriented languages, such as Hypertalk in the Hypercard environment. In such a language, the author writes instructions to alter the behaviour of created objects (*e.g.*, Hypercard's stacks, collections of cards). In our experience, first-guess expert systems are really decision trees, at least in some instances, and Hypermedia can be used to create such trees. Therefore, it was decided to investigate their use for knowledge-based systems in this application area.

As a first application, we investigated the suitability of Toolbook [9], an IBM version of Hypercard, to reimplement an expert system called CRISE [5] that was built in a traditional expert system shell. The choice of the software was dictated by our IBM hardware. The reimplementation is called CRISEBOOK.

Toolbook features two distinct working levels, namely an author and a reader level. It allows the author to create applications for readers for various purposes, such as information management, entertainment and education [10]. Applications are composed of one or more "books" (which are similar to stacks in Hypercard). Books contain "pages". A page is made up of a background and a foreground. Whereas the former is a template shared by a certain number of pages, the latter is unique to that page. Background and foreground may both contain objects such as buttons, fields and graphical objects. All those objects are part of the object hierarchy (Fig. 1), which determines the order in which

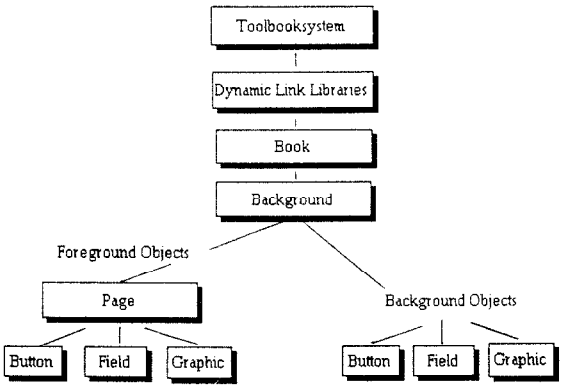

Fig. 1. CRISEBOOK's object hierarchy.

messages pass from one object to another. For each object, the author can write a script, a series of instructions, called statements, in Toolbooks's object-oriented programming language OpenScript [11,12]. As in Hypertalk, scripts may link all kinds of objects, which was most important to us in the reimplementation of CRISE. It may also show a hidden text or a button, do calculations, play sounds or modify pictures to introduce some animation and to customize the user interface to the needs of the end-user.

THE EXPERT SYSTEM

CRISE (*Cri*terion *Se*lection), was an expert system developed to assist the chromatographer in the selection of suitable optimization criteria [5]. As it was built in KES (Knowledge Engineering System, release 2.5), it suffers from the previously described problems. KES is a mid-sized expert system building tool that is mainly rule-based, but also supports frame-like structures and backward and forward chaining. A KES knowledge base is structured into an attribute section, for defining the characteristics of the problem, a rule section and an action section that guides the expert system to seek appropriate information.

CRISE consists of four modules. In the first module the most suitable elemental criterion is selected to quantify the extent of separation between two adjacent peaks in the chromatogram. Resolutions (R_s), separation factors (S), separation factors corrected for plate counts (S_N) but also expressions

such as peak–valley ratios (P, P_v, P_m) can be used. As R_s, S_N and S do not take into account a possible loss in resolution due to non-ideal situations (*e.g.*, large matrix or solvent peaks), corrections may be required and this is investigated in module 2. Module 3 assists the user in the selection of weighting factors to make a difference between peaks according to their relevance or relative importance. In the last module, the elemental criteria are incorporated in the global optimization criterion that is best suited to quantify the separation quality of the entire chromatogram taking into account the purpose of the separation.

Organization

It was decided to keep all the associated information together and thus to create one book. In a book, information can be organized in different ways, such as a series of pages, object networks or a hierarchical form, depending on the content of that information and on the way the reader is expected to use it. We decided to organize CRISEBOOK as a hierarchy and more precisely as a decision tree, first because this kind of organization fits in best with the hierarchical structure of the knowledge and also because it offers advantages for both the user and the author. The user, by choosing between different options, is guided to a particular piece of information, whereas for the author it will be easier to "write" the book. In Toolbook, the representation of knowledge is supported by object-oriented programming and rules. Those, representation facilities are perhaps more limited than in shells such as KES, where also framelike structures can be built [3], but appeared to be powerful enough to implement the knowledge base without problems. The ruleset is subdivided into scripts, smaller rulesets, which is impossible in KES and results in a better structured knowledge base.

The authoring is made easier by preprocessing, which means that a great deal of the programming has already been done. Consider, for instance, the creation of an object such as a button. The author only has to click the button tool, drag a button on the screen and adjust its size with the cursor. For the linking of pages, instead of writing a script, one can use the "link to" or "link with" function of buttons. Toolbook automatically creates a script and for the latter function even a return button is created on the destination page. If one wants to perform unpreprocessed events and needs to write a script, this can be done in a very English-like syntax, as shown in Fig. 2. This syntax is one of the strengths of object-oriented languages such as Hypertalk and OpenScript and allows one to write scripts that are entirely readable even for those unfamiliar with programming. As in the knowledge engineer interface of most expert system building tools [13], tracing facilities allow the author to investigate the order in which statements in a script are executed. Scripts can also be checked on syntax and runtime errors by using the Script window (check syntax command) or the Debug window, respectively.

```
Script for button "a" of page "page 15".

To handle buttonUp
        if button a is false then
                go to page "page Rs"
        else
                go to page "page 30"
        end if
end buttonUp
```

Fig. 2. Example of a button script for two mutually exclusive options.

As an example, let us consider the structure and the way of executing a buttonscript in Fig. 2. The aim of this script is to display the page named "page thirty" if button a is clicked, or else to display the page with a message concerning the resolution. The if/then/else control structure tells Toolbook to execute a specific statement if a specific condition is true. In conditions, functions and logical operators (*e.g.*, or, is not, and) can be called. If the user clicks the button, Toolbook will send the message "buttonUp" to that button. On condition that the button contains one or more statements to be executed on arrival of that message, Toolbook will do so and thus display page 30. If the object's script does not contain such statements, the message will be sent to the object that is the next higher level in the object hierarchy (Fig. 1), which in this example would be the page that contains the button.

To answer most of the questions the user has to select only one of two mutually exclusive options (for instance, yes/no) and an example of such a buttonscript is given in Fig. 2. If there are more than two options and they are not mutually exclusive, a

```
Script for button "answer" of page 1".

To handle buttonUp
        step i from 1 to 4
                if the hilite of button i is false then
                        put "x" after answer
                else
                        put "0" after answer
                end if
        end step
        put "page"&space&quote&answer&quote into comment page
        go to comment page
end buttonUp
```

Fig. 3. Example of a checkbox button script.

different kind of buttons, namely checkbox buttons, is applied. Consider, for instance, the question "Which descriptive parameters can you determine in the chromatogram?" Four options are offered to the user, which means that he can answer one of sixteen possible combinations. In addition to the four buttons with possible answers, a fifth button was created ("this is my answer") which the user has to click after he has clicked one or more of the other buttons. In the script of the fifth button (see Fig. 3), a "loop" with the "step" command was applied. With this command the author can instruct Toolbook to execute one or more statements a certain number of times. "Highlight" is one of the several button properties that can be changed by the author. It means that a button will flash when a reader clicks it, and hence that a visual signal will be shown. The "quote" constant and the "&" operator are used to include quotation marks as part of a string. With the "go to" command the user is referred to the page with the name "answer". All buttons are thus checked in a certain order and the answer is put in the variable "answer" in the form of a binary code, for instance 00xx if buttons 3 and 4 out of buttons 1, 2, 3 and 4 are clicked. The principle of this method is the translation of a number of clicked buttons into a binary number that is the name of the page to which the user has to go.

Altogether, CRISEBOOK includes 195 pages. Creating pages for each possible combination of answers does not necessarily result in a larger total development time and offers the important advantage that conflicting selections are error trapped. Consider, for instance, the selection of the required corrections to the elementary criteria R_s, S_N or S. If one or more corrections are selected together with the option "no correction", the user is informed of his conflicting answers. The question is asked again to determine with more certainty whether "some" or "no" corrections are required.

Presentation

Compared with the end-user interfaces of, *e.g.*,

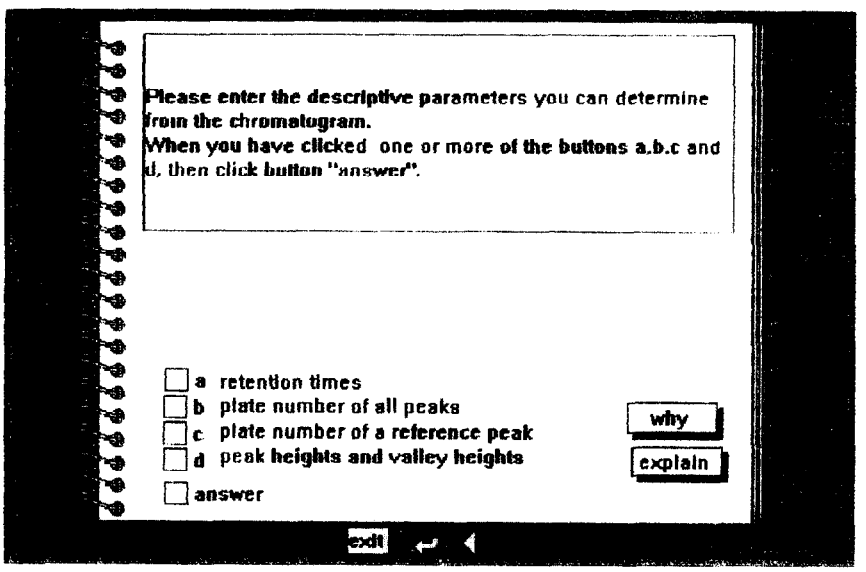

Fig. 4. Example of a page with a question.

Delfi-2, Apes and KES [13], Toolbook's interface is not standardized and hence can be customized to the needs of the user. It was decided to display on each page one question with the possible answers or a message. For each question, a number of buttons are created, equal to the number of possible answers to that question (Fig. 4). The user answers by clicking with the mouse on the button(s) next to the correct answer(s) and the result is that a following page appears with a new question. For pages with a question, the same background with three navigation buttons was always applied. The buttons allow the reader to go back through the pages that have been accessed one at a time (and thus to recall previous questions and change given answers) (arrow in Fig. 4), to go to the first page (and thus to restart the consultation) (triangle) or to leave the system (exit). In the end-user interface of KES it is not possible to recall previous questions. Consider, for instance, the case that the user has changed his mind after answering a question that is nearly at the end of the consultation. In the KES version he has to stop the consultation and start again by answering all previous questions, whereas in CRISEBOOK he only has to click the correct navigation button. Those buttons will appear in the same position and size on each page that shares that background. This enhances the conformity and hence also the user-friendliness of the system. If necessary, "explain" and "why" buttons are provided in addition to answers buttons. Questions and messages are typed in fields, most often record fields, with a scrollbar if not all of the text can be shown in the field. Pull-down menus allow the user, for instance, to print pages or to use the History command (see further). These additional features obviously represent an important advantage compared with the KES version. Another improvement of the man–machine interface was obtained by the introduction of graphics, by applying drawing tools, to explain *p*-type criteria, which cannot be explained well by definition alone (Fig. 5). Graphics cannot be provided in KES. With Toolbook's various representation facilities such as pattern-, drawing-, tool- and line-paletttes and windows, a greatly improved overall representation of CRISE was achieved. As Toolbook runs in Windows, its preprocessed screen layouts (windows, menus, etc.) are in accordance with Extended CUA (Common Usser Access), IBM-defined standard rules to regulate preprocessed screen layouts, among other things [14]. This offers the important advantage that one can switch from one CUA software tool to another because similar icons and menus will always appear at the same place.

An important feature, difficult to achieve with shells, is that at the end of the consultation an overview of intermediate results can be provided with system variables. Their values remain available in the variables while an instance is running and are changed in accordance with the statements in the scripts of CRISEBOOK's objects that are open in that instance. Although the overview was restricted to intermediate suggestions (selected elementary criterion, required corrections, use of weighting factors and selected global criterion), by using system variables it also is possible to provide a kind of "tracer window". Such a window could correspond to a page where all answers of the consultation process are represented so that the user can follow the reasoning process more easily. A list of the names of the last 100 pages accessed in a session can be displayed selecting the History command in a pull-down menu. If all pages are given a descriptive name related to the content of the message or to the question, the reader can also have in this way an overview of the path he has followed.

A validation of CRISEBOOK was accomplished by applying simulated test cases. All possible combi-

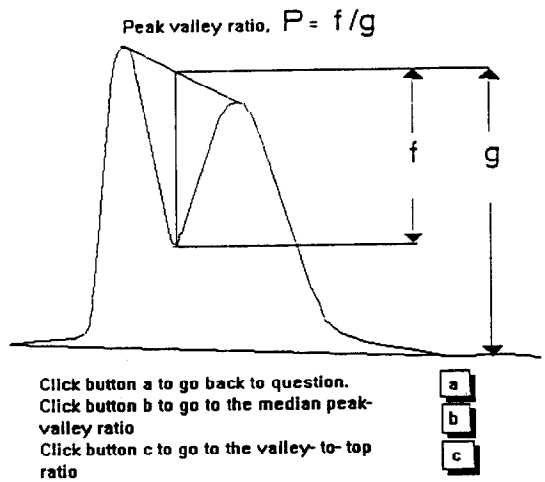

Fig. 5. Example of a graphic: the peak–valley ratio.

nations of answers were tried out and each time it was checked whether the script of the clicked button(s) referred to the correct page. This allowed us to fine-tune and optimize the system further and to test the ease of adding or deleting pages and thus of updating the system. To update one has to change only the previous script that leads to the new page; all following pages are renumbered automatically. This ease of updating is considered to be an important advantage, as explained in the Introduction. Suppose, for instance, that an additional question should be included. In CRISEBOOK one has to create a new page with buttons and with fields for the question and possible answers. The scripts of the buttons on the page that is just before have to be changed. Expanding the KES version with the same question requires a much more detailed and thus time-consuming analysis of the whole program. Changing a rule migth have consequences not only in the part that follows immediately, as is the case in CRISEBOOK, but also in rules that are elsewhere in the program and for which consequence seem much less evident. How easy it is to maintain a hypertext program in the long run is not clear yet. However, as for all types of programs, a good analysis of the system before starting the actual programming is certainly needed.

The system also has some disadvantages. One is that a full listing of all screen displays of the whole knowledge base is not possible unless a program is written, such as Hyperreporter on the MacIntosh, to print out the scripts of all objects in an application. The slow running of CRISEBOOK is not considered to be an important drawback. However, speed probably would become a real problem when external programs (see further) are added.

Algorithmic tools and external programs

CRISE and CRISEBOOK are restricted to the selection of the criterion. Therefore, they do not need any calculations and it is not yet clear how good Hypermedia will be in applications that do require computations. It is possible to perform calculations in Toolbook by means of arithmetic functions, such as average, log, tan. OpenScript has 40 of such built-in functions, each representing a particular operation that returns a value based on the values the user gives it. The author can add capabilities that are not provided in Toolbook, by applying Dynamic Link Libraries (DLL). DLL functions as an interface between Toolbook and any non-Windows application. These are separate programs the author can create with Microsoft C, Pascal or Assembler language to add new functions to OpenScript. To perform tasks that require other Windows applications, one can use Dynamic Data Exchange (DDE), a set of rules that defines communication between Toolbook and any other Windows application. Programming under windows is not easy and this might create problems.

For the practical use of the expert system, a runtime version of Toolbook was created which allows a user who has Microsoft Windows 3 to use CRISEBOOK without having the development version of Toolbook.

CONCLUSIONS

By applying Toolbook to reimplement CRISE, a system was obtained with an attractive user interface that is easy to update. Toolbook has also most of the other essential features desired in the implementation of the knowledge base such as ease of prototyping, the possibility to add external programs, a runtime version on a PC computer, "why" and "history" facilities and a way of dividing the knowledge base into subdomains. The main drawbacks are the lack of speed and the large amount of memory required (2–8 MB, depending on whether sample and help books are also installed). Hypermedia certainly is a convenient way of implementing knowledge that is in the form of or that can be translated to if/then rules, as is the case in CRISE. It is to be expected that further developments in the field of hypermedia will eliminate some of the still existing drawbacks and make it a generally valid alternative to expert system shells. It is our intention to apply this approach to several other applications in the near future.

ACKNOWLEDGEMENT

B. Bourguignon is a research assistant of the National Fund for Scientific Research.

REFERENCES

1 M. De Smet, G. Musch, A. Peeters, L. Buydens and D. L. Massart, *J. Chromatogr.*, 485 (1989) 237.

2 M. De Smet, A. Peeters, L. Buydens and D. L. Massart, *J. Chromatogr.*, 457 (1988) 25.
3 R. Hindriks, F. Maris, J. Vink, A. Peeters, M. De Smet, D. L. Massart and L. Buydens, *J. Chromatogr.*, 485 (1989) 25.
4 F. Maris, R. Hindriks, J. Vink, A. Peeters, N. Vanden Driessche and D. L. Massart, *J. Chromatogr.*, 506 (1990) 211.
5 A. Peeters, L. Buydens, D. L. Massart and P. J. Schoenmakers, *Chromatographia*, 26 (1988) 101.
6 T. Hamoir, M. De Smet, H. Piryns, P. Conti, N. Vanden Driessche, D. L. Massart, F. Maris, H. Hindriks and P. J. Schoenmakers, *J. Chromatogr.*, in press.
7 P. Conti, T. Hamoir, M. De Smet, H. Piryns, N. Vanden Driessche, F. Maris, H. Hindriks, P. J. Schoenmakers and D. L. Massart, *Chemometr. Intell. Lab. Syst.*, 11 (1991) 27–35.
8 M. Farkas, M. Cadish and E. Pretsch, *Scientific Computing and Automation*, Elsevier, Amsterdam, 1990.
9 *TOOLBOOK*, Asymmetrix, Washington, DC, 1989.
10 *Using TOOLBOOK*, Asymmetrix, Washington, DC, 1989.
11 *Using OpenScript*, Asymmetrix, Washington, DC, 1990.
12 J. M. Pierce, *Toolbook Companion*, Microsoft Press, Washington, DC, 1990.
13 J. A. van Leeuwen, *Ph. D. Thesis*, University of Nijmegen, 1990.
14 D. Bartels, *DOS*, 5 (1990) 66.