The Knowledge Engineering Review

Graphical interfaces for knowledge engineering: an overview of relevant literature

SARA JONES

Department of Computer Science, City University, Northampton Square, London ECIV 0HB

Abstract

Literature relevant to the design and development of graphical interfaces for knowledge-based systems is briefly reviewed and discussed. The efficiency of human-computer interaction depends to a large extent on the degree to which the human-machine interface can answer the user's cognitive needs and accurately support his or her natural cognitive processes and structures. Graphical interfaces can often be particularly suitable in this respect, especially in cases where the user's "natural idiom" is graphical. Illustrated examples are given of the way in which graphical interfaces have successfully been used in various fields with particular emphasis on their use in the field of knowledge-based systems. The paper ends with a brief discussion of possible future developments in the field of knowledge-based system interfaces and of the role that graphics might play in such developments.

1 Introduction

Interest in the field of knowledge-based system interfaces has been increasing over the past few years. This interest is reflected in the growing body of literature in that area (see, for example, Hendler, 1988). Below is an attempt to organize and briefly discuss literature relevant to the design and development of knowledge-based system interfaces with particular emphasis on key areas such as human factors and knowledge engineering.

2 The importance of human factors

Computer systems and applications are becoming ever more powerful, complex and sophisticated. Many of today's systems are also highly interactive, relying heavily on human input in order to perform their functions. In recent years, it has been the aim of human factors specialists to ensure that these rapid developments are matched by an equally rapid evolution of the human computer interface.

Interface design has been shown to have a significant influence on factors such as learning time, performance speed, error rates and user satisfaction. Numerous studies report that information display format significantly affects performance in various tasks such as problem solving (Wright and Reid, 1973, Senach, 1983), decision-making (Benbasat and Dexter, 1985, Benbasat, Dexter and Todd, 1986), and fault-location (Brooke and Duncan, 1981). While good interface design can lead to significant improvements in task performance, poor designs can actually hold the user back. Work on the human-computer interface is therefore of crucial importance. Human factors engineering, which has been thought of as "the paint put on at the end of a project" is according to Shneiderman now increasingly viewed as "the steel frame on which the structure is built".

The situation with knowledge-based systems is similar to that with many other types of application. They have evolved rapidly over the last few years and the general move to what have been called "expert advisory systems" rather than classical expert systems such as MYCIN, Prospector and R-1, means that users are now very much involved in the decision making loop. Interactions

with early systems tended to be mainly system-driven: the user would simply respond to the system's prompts and then be given an answer. The trend now is for users to make more of the decisions themselves on the basis of help and information provided by the advisory system, so that user and system both take part in the decision making process (Stelzner and Williams, 1988).

There is a feeling that the lack of acceptance of early expert systems was due, in part at least, to the lack of transparency and comprehensibility which resulted from the poor design of their interfaces (Hendler and Lewis, 1988). Work on knowledge-based system interfaces can therefore be seen as crucial in order to accommodate more user-driven interactions and to lead to increased user acceptance of future systems.

2.a Current perspectives in human factors engineering

In recent years, the field of human factors engineering has seen a shift in emphasis from the consideration of the human or the computer in isolation to that of what Woods has called the "joint human machine cognitive system", or the human-computer system as a whole (Coats and Vlaeminke, 1987).

Although some amount of training must always be expected in order for a user to become maximally proficient with a given tool, it is, in general, easier to modify the characteristics of a computer system than those of its users. One of the main aims of human factors engineers or "software ergonomists" has therefore been to match the interface characteristics of computer software to the user's cognitive needs in the same way as traditional ergonomists match hardware to human physical needs.

Current theories of interface design (for example Card, Moran and Newell, 1983, Foley and Van Dam, 1982) all suggest the importance of inducing an appropriate "conceptual model" in the user in order that he or she can learn to use an application as quickly as possible and then perform comfortably and efficiently with it. If this conceptual model can be based on users' existing models and the real world experiences and expectations which they bring to any system, then so much the better.

The views of those concentrating on the design of knowledge-based system interfaces are again similar to those of human factors engineers in general (Stelzner and Williams, 1988). Experimental evidence suggests that the user's ability to build an appropriate cognitive model of system processing is a dominant factor in determining the quality of user-expert system interaction (Hendler and Lewis, 1988, Lehner and Kralj, 1988). Apart from the importance of an accurate conceptual model of the system's reasoning processes, Lehner and Kralj also suggest that the expert system's problem solving procedures, knowledge structures and knowledge elements should appear to be consistent with the user's own. It is of course the interface which determines how these things will appear to the user.

See section 1 of the bibliography for general texts on Human-Computer Interaction, for example Baecker and Buxton, 1987; Coats and Vlaeminke, 1987; Schneiderman, 1980, 1987 for a general overview and Card, Moran and Newell, 1983; Foley and Van Dam, 1982; Norman and Draper, 1986 for three classic approaches. For discussion of how the interface should look, see section 2 (Foley, Wallace and Chan, 1984; Sanders and McCormick, 1987; Smith and Mosier, 1986 give more specific guidelines). For a cognitive perspective, see section 3: Gardiner and Christie, 1987 is a general text, see also Carroll, 1987; Murch, 1984. For overall coverage of HCI issues in relation to knowledge-based systems, see Hendler, 1988.

3 The argument for the use of graphics

It has been suggested above that the design of the human-computer interface can be vital to the efficient functioning of the system as a whole. It has also been suggested that the best design will probably be one which reflects, as far as possible, the user's natural cognitive processes and structures. How, then, can the use of computer graphics contribute to the development of better interfaces?

There are many good arguments for the use of graphical interfaces to at least some applications.

According to Foley and Van Dam, "pictorial communication is a medium that is both natural and efficient to human beings and yet is sufficiently precise for computer manipulation". Graphical representations are well-suited to the display of multi-dimensional information and allow a much greater bandwidth of human-computer communication than is possible with text alone. There is evidence to suggest that recall is better for information that is presented pictorially (Marshall, Nelson and Gardiner, 1987) and it is often claimed that people enjoy interacting graphically with a computer more than they do using traditional alphanumeric techniques (Foley and Van Dam, 1982).

In terms of supporting the cognitive processes and structures of the user, a graphical display can be highly appropriate in cases where the user's "natural idiom" is graphical (Stelzner and Williams, 1988). The claim that "people think in pictures" is often heard, and while it is certainly not true that all people think pictorially about all things, there is startling evidence to show that people do work with mental images in a way that is closely analogous to the manner in which they work with actual images of the real world (Shepard and Metzler, 1977). Rumelhart et al. (1986) have claimed that one of the most effective problem-solving strategies people use involves the creation of virtual physical representations or mental images of physical structures which can then be examined and understood using powerful pattern-matching abilities.

Taking the approach one step further, it can be seen that we should not only aim to represent information in the natural idiom but that we should also aim to allow users to act on information displayed in the same way as they act on mental images. This is the philosophy behind direct manipulation interfaces.

Direct manipulation interfaces are usually based on real world metaphors, such as the desktop metaphor, which attempt to model sections of the physical world. They present the user with graphical representations of objects within that world (for example, files and mail trays in the case of the desktop metaphor) which he or she can then act on in a direct manner, often using a mouse. There is therefore no need for a mental decomposition of tasks into verbal commands with complicated syntax: the user can simply perform an action on a symbolic representation of the object in question and observe its results in a natural and realistic way.

The success of direct manipulation interfaces using the techniques of interactive computer graphics has been widespread. They are now used with many applications and are found to promote high levels of task performance and user satisfaction. The question remains as to how useful such interfaces could be in the domain of knowledge-based systems applications.

Many of the general advantages of graphical interfaces also apply in the knowledge-based systems domain. Tsuji and Shortliffe (1983) argue that the way in which graphical representations can assist in comprehension of complex systems and structures could be especially useful here and Woods (1986) reminds us that increased comprehension as a result of visualization can lead to better problem structuring and solving. McAleese (1987) also claims that visual representations can lead to increased comprehension of the kinds of logical information structures which lie at the heart of knowledge-based systems. It seems possible, then, that with the use of direct manipulation techniques, graphical interfaces could have a similar impact on the field of knowledge-based systems to that which they have already had on other application domains.

Looking in a little more detail at the requirements for a knowledge-based system interface, we can see that there are three distinct user groups which it must support. Domain experts, involved in the initial creation of the system, knowledge engineers, who design, implement and maintain it, and end users all have different requirements which must be catered for.

The end user of the knowledge-based system will be mainly concerned with the knowledge domain itself rather than its internal representation in the knowledge base. In many domains, graphical representations are the natural medium for communication and can therefore be used to maximum effect in the interface (see section 5.a).

The knowledge engineer, and to some extent the domain expert, will be more concerned with the system's internal representations of the domain and its reasoning processes (Stelzner and Williams, 1988). The question then arises as to whether a graphical representation of a knowledge-based system's internal knowledge structures is a natural one in terms of accurately modelling or support-

ing the user's own internal representations. This is a difficult issue—as there is no real world physical counterpart to the kind of knowledge structures we are considering, we can only guess at the way people internally represent such structures. Kidd and Cooper (1983) and Tsuji and Shortliffe (1983) are optimistic that the kinds of graphical representations we see in today's knowledge-based system applications (see section 6.a, 6.b) are sufficiently similar to the user's own conceptual representations to be useful. Kidd and Cooper report that a diagrammatic representation of the inference network in a Prospector-like expert system shell helped experts to develop an accurate conceptual model of how knowledge would be represented in the system, and hence an idea of the sort of knowledge a knowledge engineer was trying to collect and why. Kearney and Hunt (1987) suggest that this kind of representation can also help the end user to develop an accurate conceptual model of how the system works and hence use it more efficiently. Finally, Tsuji and Shortliffe suggest that the use of such graphical representations would be a substantial improvement over the kinds of explanation facilities commonly found today.

Examples of applications developed over the past few years with highly graphical interfaces will now be discussed. Section 4 presents some examples of applications in domains in which graphical interfaces have been found to be particularly useful and suitable. Section 5 concentrates on graphical interfaces to knowledge-based system applications, both for the end user, and for the knowledge engineer.

See section 4 of the bibliography for articles generally concerning the use of graphics in the interface. Foley and Van Dam, 1982 is a classic work on the use of interactive graphics. For more about direct manipulation interfaces, see Schneiderman, 1987; Hollan et al., 1987; Hollan, Hutchins and Weitzman, 1984. For discussion of the use of graphical interfaces with knowledge-based systems, see Baroff et al., 1988; Dodson, 1988, 1987; Kidd and Cooper, 1983; McAleese, 1987; Tsuji and Shortliffe, 1983; Woods, 1986.

4 Graphical interfaces for specific applications

This section looks briefly at some examples of the kinds of representations that have been developed for applications in three domains in which there has been considerable interest in the use of graphical interfaces. The section ends by looking briefly at attempts to develop application independent graphical interfaces.

4.a Programming tools and environments

The use of graphical techniques as an aid to programming has been under discussion for many years. There was a great deal of reserach in the late 1960s and early 1970s on the usefulness of flowcharts in programming. The results of this research were at best inconclusive and at worst contradictory. Shneiderman et al. (1977) presented a review of work in that area together with results of some experiments of their own intended to lay the matter to rest once and for all. Testing the usefulness of flowcharts in program composition, debugging and modification for subjects of low or intermediate skill produced negative results. Schneiderman et al. suggested that flowcharts were redundant or even disadvantageous to programmers except possibly when dealing with reasonably large programs, of greater than 1000 lines, for which they may reduce anxiety and confusion.

The more recent use of graphical techniques in programming has been a little more successful (Myers, 1986). Halbert (1984) has shown that non-programmers can create fairly complex programs with little training using visual programming systems which provide the user with graphical programming languages. Even professional programmers can apparently gain by using program visualization systems which provide graphical representations of the run-time state of a program specified in conventional textual form.

There are now too many such systems to provide an exhaustive list. Recent examples include Pict (Glinert and Tanimoto, 1984, see Fig. 1) which allows the user to work with conventional flowcharts

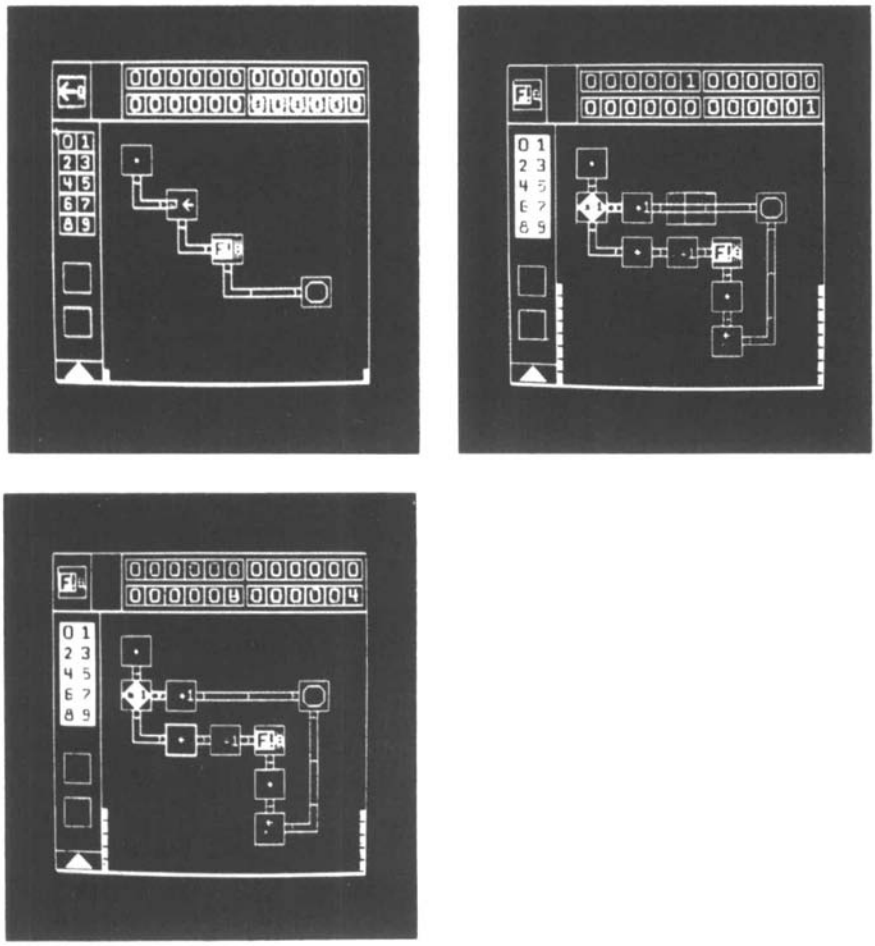

Figure 1 Three frames from Pict (Glinert and Tanimoto, 1984). Original in colour

using colour pictures or icons rather than text in the flowchart boxes, and PROGRAPH (Pietrzykowski, Matwin and Muldner, 1983, see Fig. 2) which provides a slightly more novel graphical representation of a functional data flow language. A visual programming interface has also recently been provided for Thinglab (a tool for creating and using virtual instruments such as springs and pulleys, see Borning, 1979, 1981) which allows users to program visually in the object-oriented para-

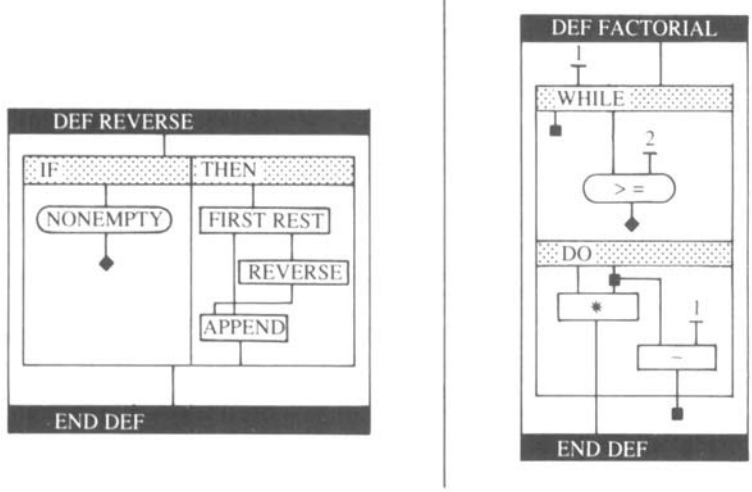

Figure 2 Two procedures from PROGRAPH (Pietrzykowski, Martin and Mulder, 1983)

digm (Borning, 1986). Suggestions for graphical interfaces to programming languages of particular interest to the knowledge engineering community, such as Lisp and Prolog will be discussed in section 5.b.

Despite recent advances, visual programming and program visualization techniques still have a long way to go before they can claim to be superior to conventional alternatives. Many systems still suffer from all or several of the drawbacks outlined by Myers, 1986 as follows. As he pointed out, visual representations tend to be physically larger than the text they replace and a method must be found to get around this using scrolling or abstraction. Many visual programming systems work only in limited domains and run programs very slowly. With large programs, it is even debatable whether a visual representation provides the increased comprehensibility which is its greatest claim: static representations which may be ideal for small scale demonstrations can begin to look like a "maze of wires" when used with more complex applications.

4.b Database management

A large number of tools incorporating graphical techniques have also been developed for work with databases.

Yasdi (1985) described a graphical representation facility for "knowledge engineering", that is, development of the Conceptual Requirement, Conceptual Behaviour and Conceptual Structure graphs which form the basis of the conceptual knowledge model for a data base (see Fig. 3). Zhang and Mendelzon (1983) and Frasson and Er-radi (1986) suggest graphics based and iconic command languages for querying relational databases while Goldman et al. (1985) and Bryce and Hull (1986) present ISIS and SNAP respectively: two more general tools for design, manipulation, browsing and querying of databases.

The graphical paradigms used in the above systems are all relatively similar in that they are based firmly in the traditions of paper representations, and exploit the possibilities of graphical representation only to a limited extent. More recently, Reid (1988) has proposed the use of 3-dimensional representations based on the "molecular model metaphor" (see Fig. 4) which could form the basis of future tools by making better use of today's technologies and providing the user with greater control over large and highly complex data bases.

The use of graphical interfaces in a domain in which paper graphical representations are already widely used is obviously highly appropriate. With greater use of techniques such as those suggested by Reid, the functionality of such interfaces may be still further enhanced.

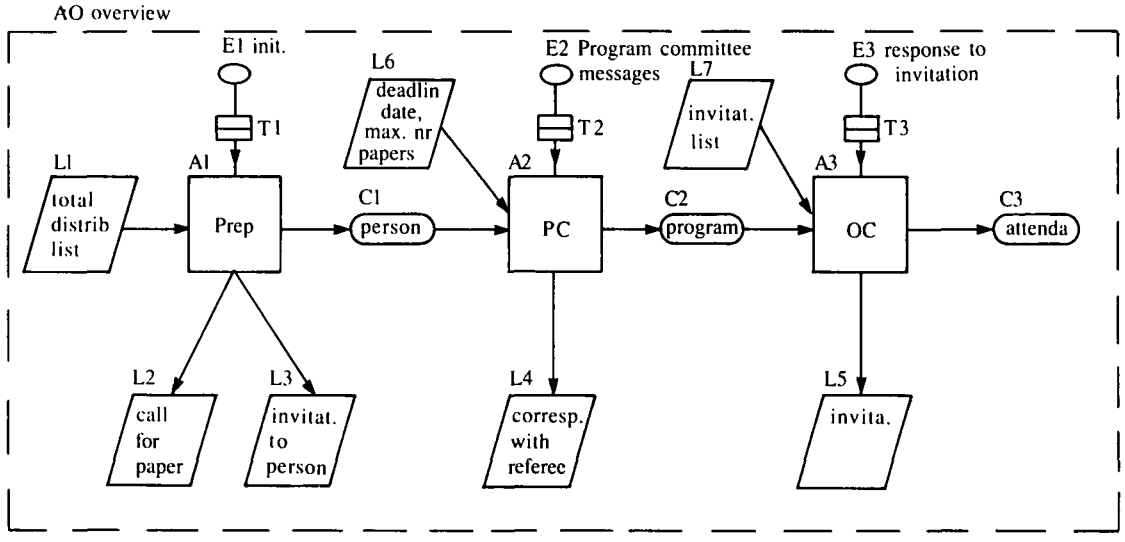

Figure 3 Conceptual requirement graph as suggested by Yasdi (1985)

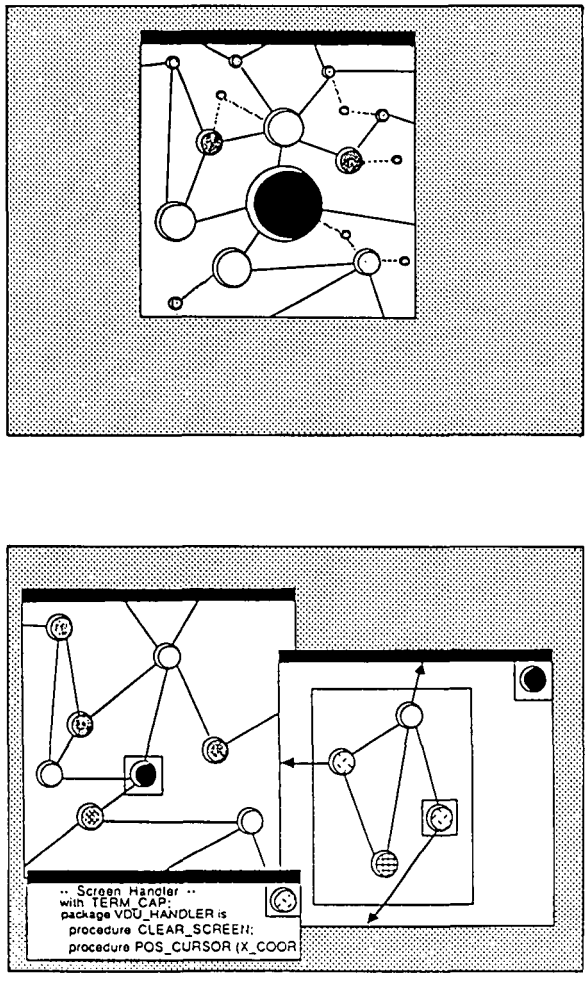

Figure 4 Three-dimensional graphical representation of a large project data base as suggested by Reid (1988). Above is a fisheye view. Below are flat views showing expansion of individual nodes

4.c Simulation and training environments

One of the areas in which the possibilities of graphical interfaces have been most fully exploited has been that of simulation and training systems. Many systems, notably those used in CAD, employ more or less sophisticated graphical representations of objects and processes. Below are a few examples.

Thinglab (Borning, 1979, 1981) and LabVIEW (Vose and Williams, 1986, see Fig. 5) are systems allowing scientists and engineers to create virtual instruments such as springs and pulleys which are represented on the display by graphical images. The user can observe the interactions of the graphical objects which apparently behave as real objects would according to the laws of physics and mathematics. In Thinglab, the laws themselves can be modified by the user (for instance, the value of the gravitational constant g can be reset) in order to observe the effects of such changes.

Knapp, Moses and Gellman (1982) describe a battlefield display system for training of military personnel. Here, the emphasis is on displaying complex information in such a way that it can be understood as quickly and efficiently as possible. Interface designers have therefore concentrated on clutter reduction and the use of effective coding techniques (see Fig. 6).

Finally, STEAMER (Hollan, Hutchins and Weitzman, 1984, Hollan et al., 1987) is a tool intended for training purposes which consists of a graphical interface to a mathematical model of a steam propulsion plant (see Fig. 7). Graphical icons and animation are used to construct a simulation of the plant which is rich in information so that students can watch the evolution of plant states and the effects of their actions on these states.

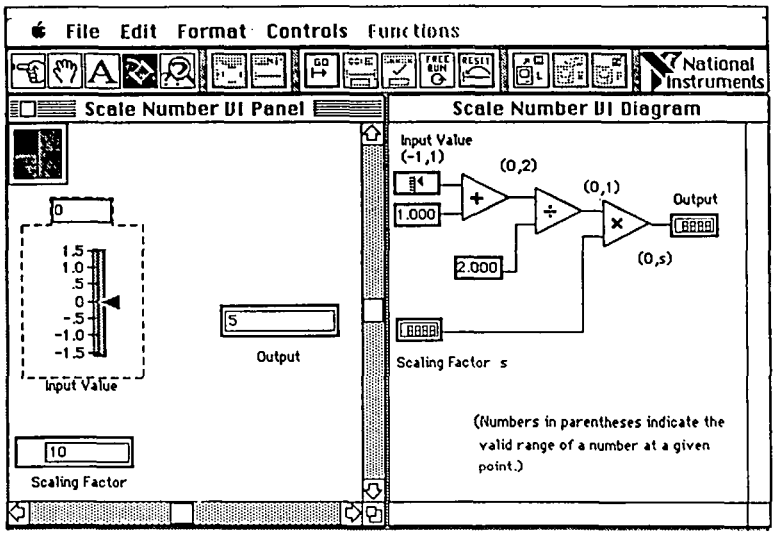

Figure 5 Screen from LabVIEW (Vose and Williams, 1986). The left of the screen shows the front panel for the virtual instrument represented on the right

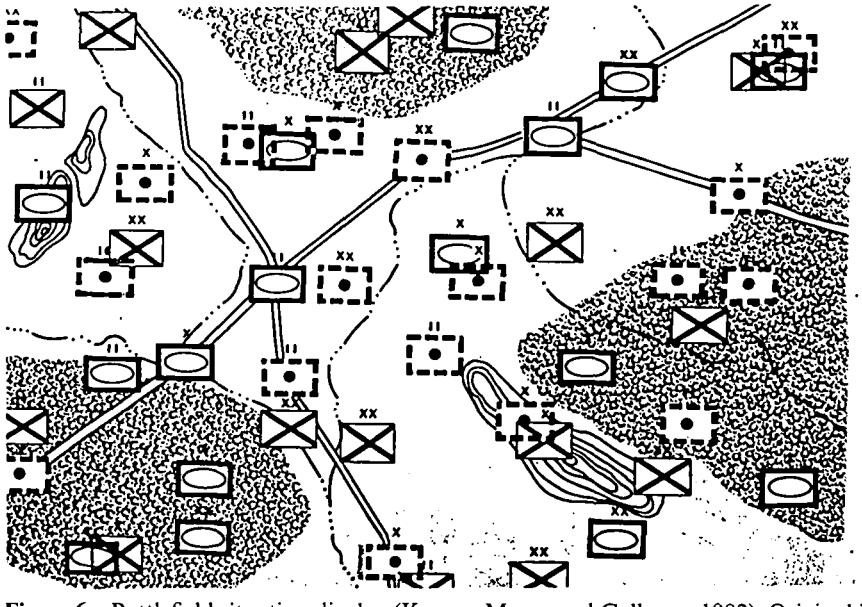

Figure 6 Battlefield situation display (Knapp, Moses and Gellman, 1982). Original in colour

Graphical interfaces can obviously be extremely useful in the development of sophisticated training and simulation environments. Users of such interfaces are able to obtain a much more immediate impression of what is happening to the object or system in question by looking at a graphical representation than a textual or symbolic representation of a mathematical model. People can be taught to operate vast and complex systems, such as nuclear plant control systems, without fear of the disastrous consequences of mistakes, and of course, the more realistic the training environment, the more likely it is that a trainee's experience will be carried over into real world situations.

4.d Application independent graphical interfaces

Several systems have been developed which aim to provide application independent graphical interfaces. Such systems generally rely on the use of heuristic knowledge about the kinds of graphical

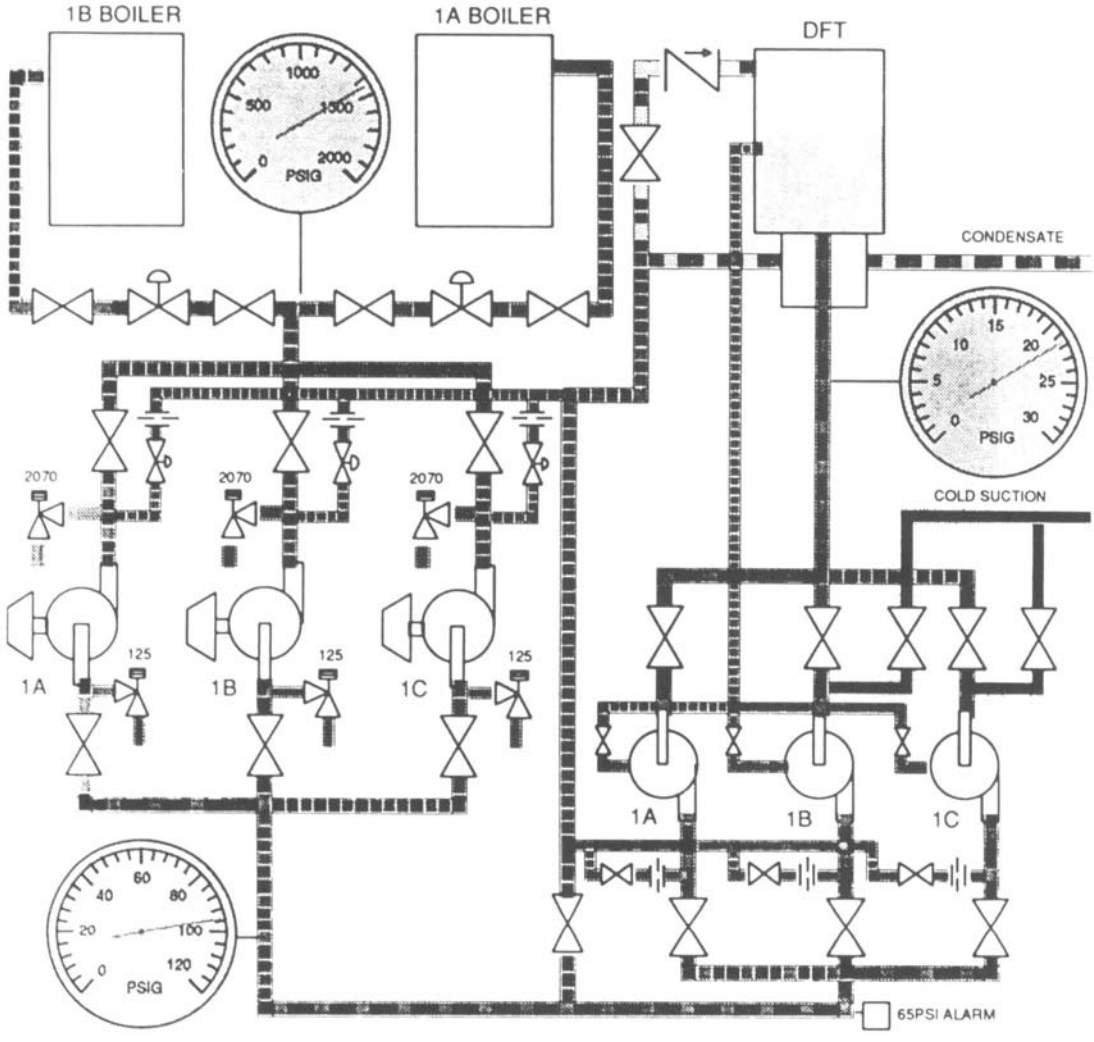

Figure 7 Main feed system of the steam propulsion plant represented by STEAMER (Hollan et al., 1987). Original animated in colour

representations which are suitable for particular uses and aim to be able to automatically generate displays for a variety of applications.

APEX (Feiner, 1985) uses knowledge about objects and actions stored as frames to automatically generate a simple 3-dimensional picture or sequence of pictures to illustrate those actions. It is currently only at the stage of a prototypical tool and has not yet been linked to any major application.

APT (Mackinlay, 1986) is another application independent presentation tool which uses coded graphic design criteria to automatically design graphical presentations (such as bar charts and scatter plots) of relational information. APT is also as yet only a prototype.

Finally, SAGE is a system developed by Clemons and Greenfield (1985) to support rapid preparation of graphics interfaces for decision support systems. SAGE uses displays of icons which can be positioned on maps or grow, shrink, rotate or change colour to reflect changes in the values they represent. It has apparently been quite successful as an interface to a restricted class of modelling packages in the field of managerial decision support.

Most such systems for automatic generation of graphical representations are still at an early stage in development and cater only for very restricted domains of application. There is a great deal of work to be done before they can be expected to take over the work of generating graphical interfaces for even a relatively small set of applications, let alone for applications in more than one domain.

For further information on applications (other than knowledge-based systems) with graphical interfaces, see section 5 of the bibliography.

5 Graphical interfaces for knowledge-based systems

Section 3 presented arguments for the use of graphical interfaces to applications in the knowledge-based system domain. The section illustrates examples of the ways in which graphical representations have actually been used both in end user interfaces (to represent the domain itself), and in interfaces for the knowledge engineer to clarify the system's internal representations of domain knowledge.

5.a Graphical interfaces for the knowledge-based system end user

It is becoming more and more common for knowledge-based systems to have at least some graphical, often direct manipulation component in their interfaces. Stelzner and Williams suggest that this trend towards the use of increasingly sophisticated interfaces is due in part to the transition to what they call expert advisory systems (see section 2) and the increased use of deep knowledge. The general increase in size and complexity of expert systems is probably also responsible as is the move to workstation environments and the corresponding increase in availability of powerful interface design tools.

As was suggested above (see section 3), the use of graphical interfaces has probably been most successful for systems in domains such as engineering and medicine in which pictures, maps and diagrams are the natural medium for communication. Below is a brief review of the ways in which graphical representations have been used in end user interfaces to systems in these areas.

The ONCOCIN project (Tsuji and Shortliffe, 1983) which has been going on at Stanford for several years was one of the first to attempt to exploit the possibilities of graphical end user knowledge-based system interfaces. ONCOCIN is a system for use by doctors in determining optimal therapy for cancer patients. At the beginning of a consultation, ONCOCIN's query system displays a window containing a graph of the hierarchical relationship between diseases and chemotherapies (see Fig. 8). The user may then constrain the enquiry to a certain situation by direct manipulation of the graph using the mouse (Fig. 8 shows the appearance of the display after an enquiry restricted to the investigation of the MOPP therapy in the context of Hodgkins disease). Pop-up menus can be used to request further information in the form of further graphical displays (see Fig. 8) or textual descriptions. It can be seen that the graphical and interaction paradigms used in this system are relatively simple. Graphical displays consist only of simple networks of textual nodes and line links with emboldening to illustrate the results of user interactions. Even this simple alternative to text is, however, valuable in increasing user understanding of the complex material involved.

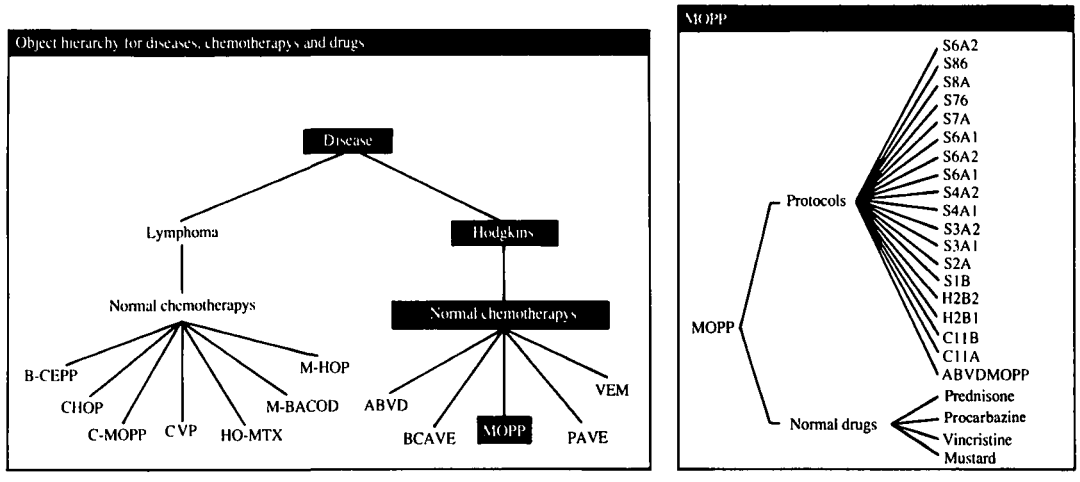

Figure 8 ONCOCIN's graphical query system (Tsuji and Shortliffe, 1983). On the left is a graph showing the relationship between diseases and chemotherapies. On the right is an expansion showing the particular treatment protocols in which MOPP therapy (highlighted on the left as a result of a user request) is involved

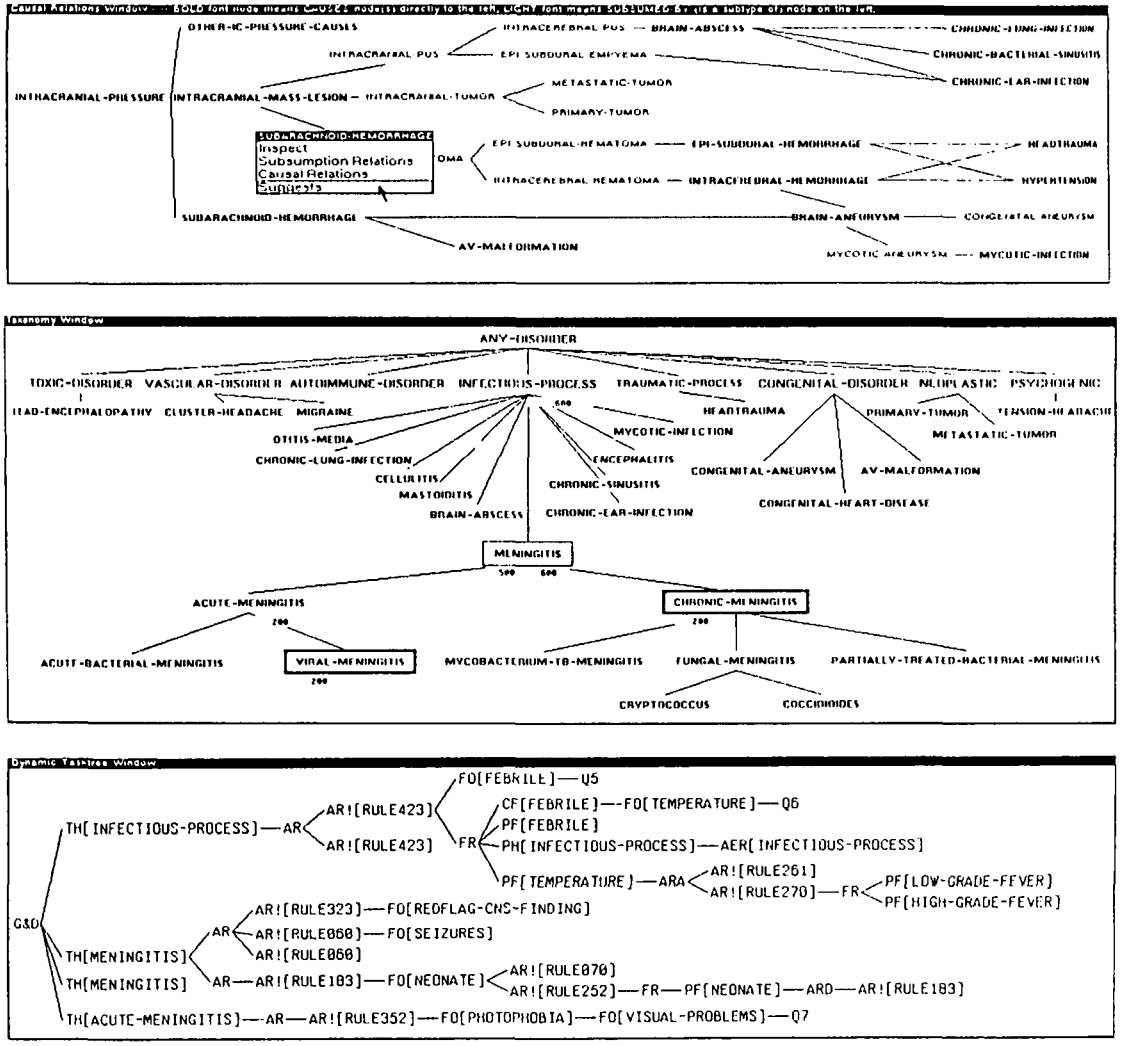

Figure 9 Graphical representations used in GUIDON-WATCH (Richer and Clancey, 1985). Top is a causal relations network with pop-up menus giving options for further information. Middle shows a disease taxonomy—use of boxing, flashing and different font styles illustrate the dynamic search strategy using a static representation of the knowledge structure. Bottom is a dynamic task tree showing multiple calls of a task and the resulting events

GUIDON-WATCH is a graphical interface to NEOMYCIN also developed at Stanford (Richer and Clancey, 1985, see Fig. 9). It allows the user to browse through the NEOMYCIN knowledge base and view reasoning processes during a consultation. It is intended as a tool for both programmers and medical students. It can in fact provide an effective user interface to any HERACLES knowledge base (HERACLES is a software tool which can be used in the development of diagnostic knowledge-based systems in many domains) but has been tuned to display kinds of knowledge structures such as disease taxonomies, causal networks and evidence for hypotheses.

The graphics paradigm used by GUIDON-WATCH is similar to that used by ONCOCIN and is also based on a system of direct manipulation and multiple windows. Causal relations, disease taxonomies and dynamic task trees are all shown as simple graphs consisting of textual nodes and line links with boxing, flashing and different print styles used for emphasis and in the illustration of dynamic processes. For example, in the taxonomy and causal relations windows, nodes representing hypotheses newly added to the set under consideration are then boxed with a heavy line. When hypotheses are no longer under consideration, the boxes are redrawn using a light line leaving the user with a permanent record of those hypotheses which have been considered. In the text-based evidence window, findings with positive values and rules that have succeeded are shown in bold

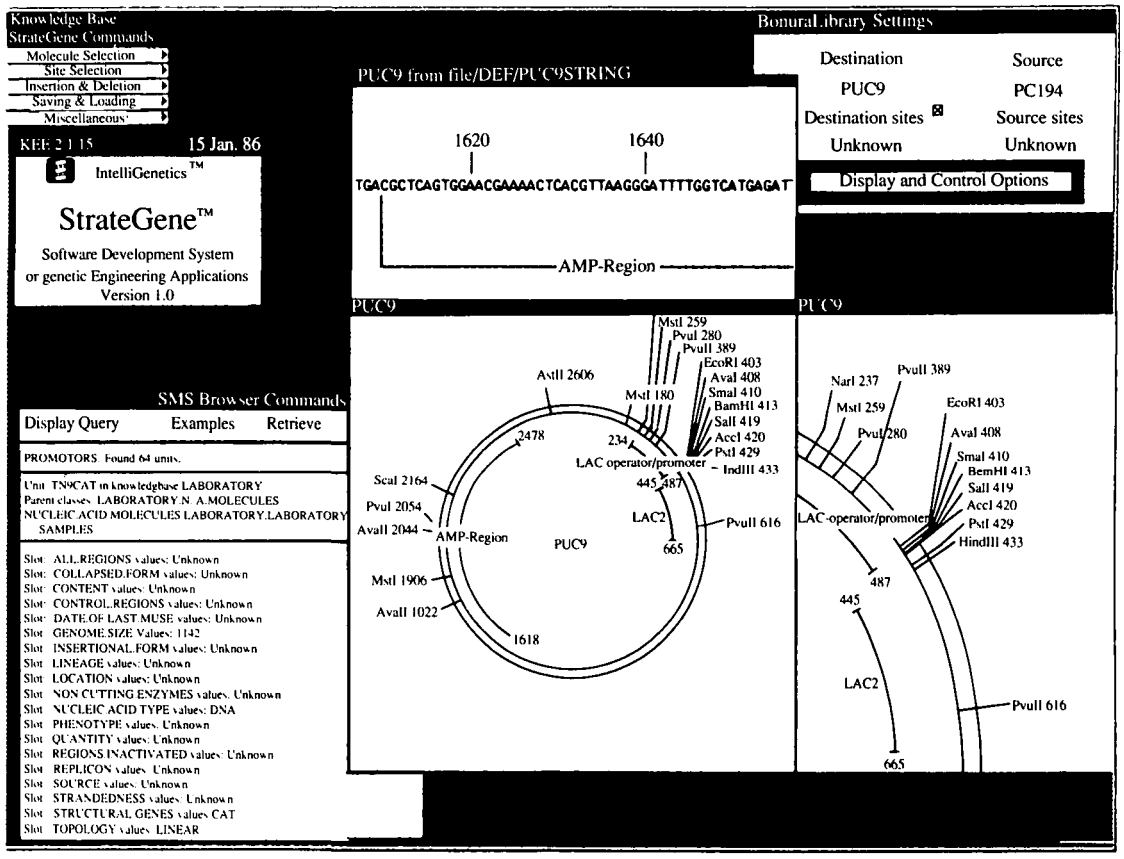

Figure 10 The Strain Management System (Stelzner and Williams, 1988). Genetic engineers can manipulate the graphical representations of genes to investigate the effects of gene splicing

whereas negative findings and failed rules are shown greyed over. Findings and rules not yet considered are shown in normal print. Again, relatively simple graphical techniques have been used to develop an interface which is far more useful and informative than any which could be achieved using text alone.

More recent systems allow users to operate directly on graphical representations of objects in the knowledge domain. For example, the Strain Management System (Stelzner and Williams, 1988) developed at Intellicorp allows genetic engineers to study the effects of gene splicing by manipulating graphical representations (see Fig. 10) and the GRIPE system (Seifert and Rawlings, 1988) developed at the Imperial Cancer Research Fund labs in London allows molecular biologists to investigate the structure of selected proteins by constructing graphical queries to an underlying knowledge base (see Fig. 11).

These systems illustrate the general trend towards expert advisory systems rather than classical expert systems and show how it is more and more often the domain expert who is the principal end user of the system. With this in mind, Baroff et al., 1988 have suggested that the expert should be more involved in the development of the interface as well as the system itself, and have coined the term "interface acquisition" in analogy with knowledge acquisition. Interfaces designed by domain experts used to working and communicating with pictures or diagrams are of course likely to have a large graphical component.

Furthermore, as systems and interfaces become more intelligent and sophisticated, it is likely that the expert's role could extend still further to include development of the knowledge base itself. Here again, graphical interfaces can play a crucial part in allowing experts to enter knowledge by performing operations directly on graphical representations of objects in the domain, rather than by using complex statements in the system's knowledge representation language. There are already some such systems in existence. For example, DETEKTR (Freiling and Alexander, 1984) is a knowledge-based tool which allows experts to develop expert fault-finding systems using simple

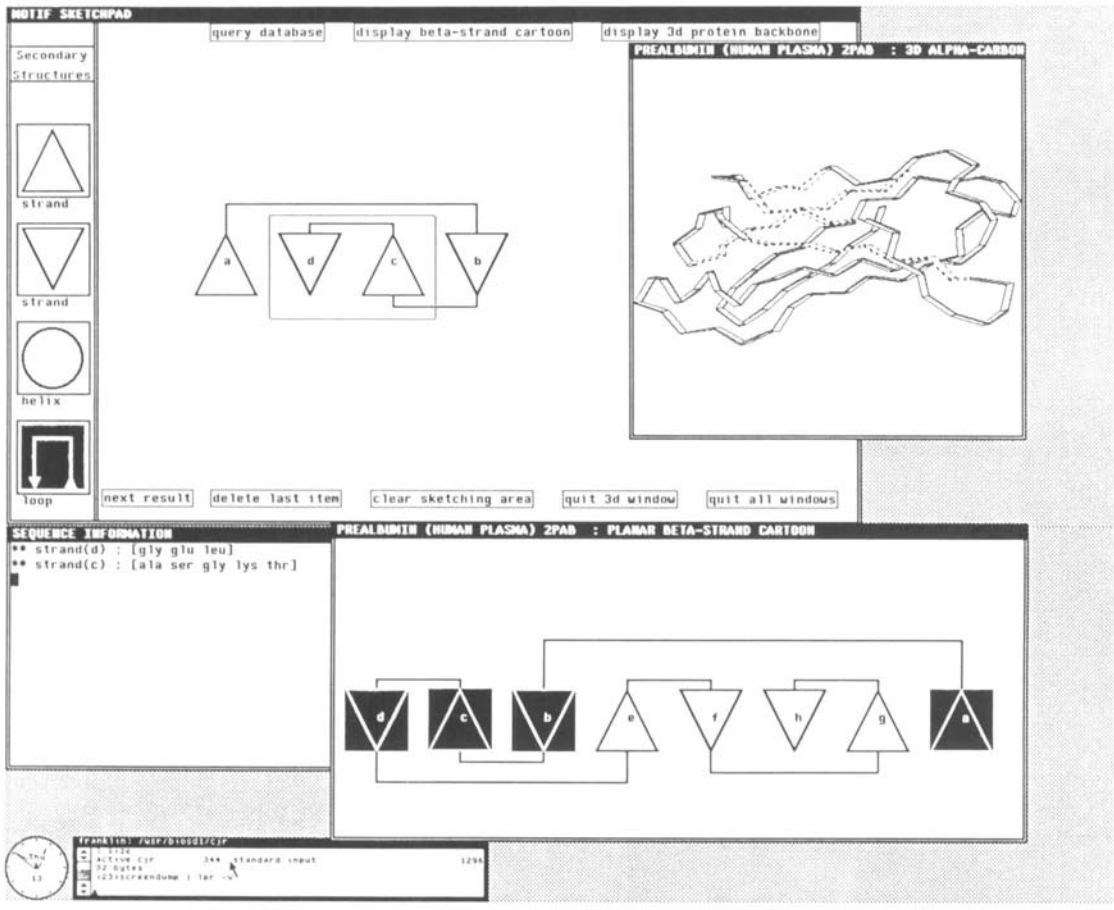

Figure 11 The GRIPE system interface (Seifert and Rawlings, 1988). The display shows the results of a graphical query concerning the structure of the protein shown in the upper right hand corner of the screen

direct manipulation of a diagrammatic representation of a printed circuit board and OPAL (Musen, Fagan and Shortliffe, 1986) is a "knowledge editor" which has been developed as part of the ONCOCIN project to allow doctors to create and modify ONCOCIN knowledge bases by using flow chart representations of treatment programs.

5.b Graphical interfaces for the knowledge engineer

The knowledge engineer can now choose from a wide variety of languages, tools and environments for working with knowledge-based system applications. While standard implementations of languages traditionally used for knowledge engineering such as Prolog and Lisp still have fairly simple text-based interfaces, some of the newer tools and environments, especially those available on workstations, have increasing numbers of graphical components. In this section, I will briefly review the kinds of tools commonly used for knowledge engineering and the types of graphical facilities they provide.

There are many dialects of Lisp (Hasemer, 1987) which run mainly on specialised Lisp machines such as the Symbolics, TI Explorer and Xerox 1180 series, though increasingly also on general-purpose workstations. Lisp environments are often fairly sophisticated in themselves. For example, the Interlisp-D environment on a Xerox 1186 workstation includes such features as function, text and bit-map editors as well as graphic on-line help facilities and "file browsers" which allow users to search through a hierarchical representation of the system's file structure. Moreover, as the environments themselves are generally written in Lisp, the programmer can tailor them to suit his or her individual needs with relative ease.

Several attempts have been made to develop a graphical approach to programming in Prolog.

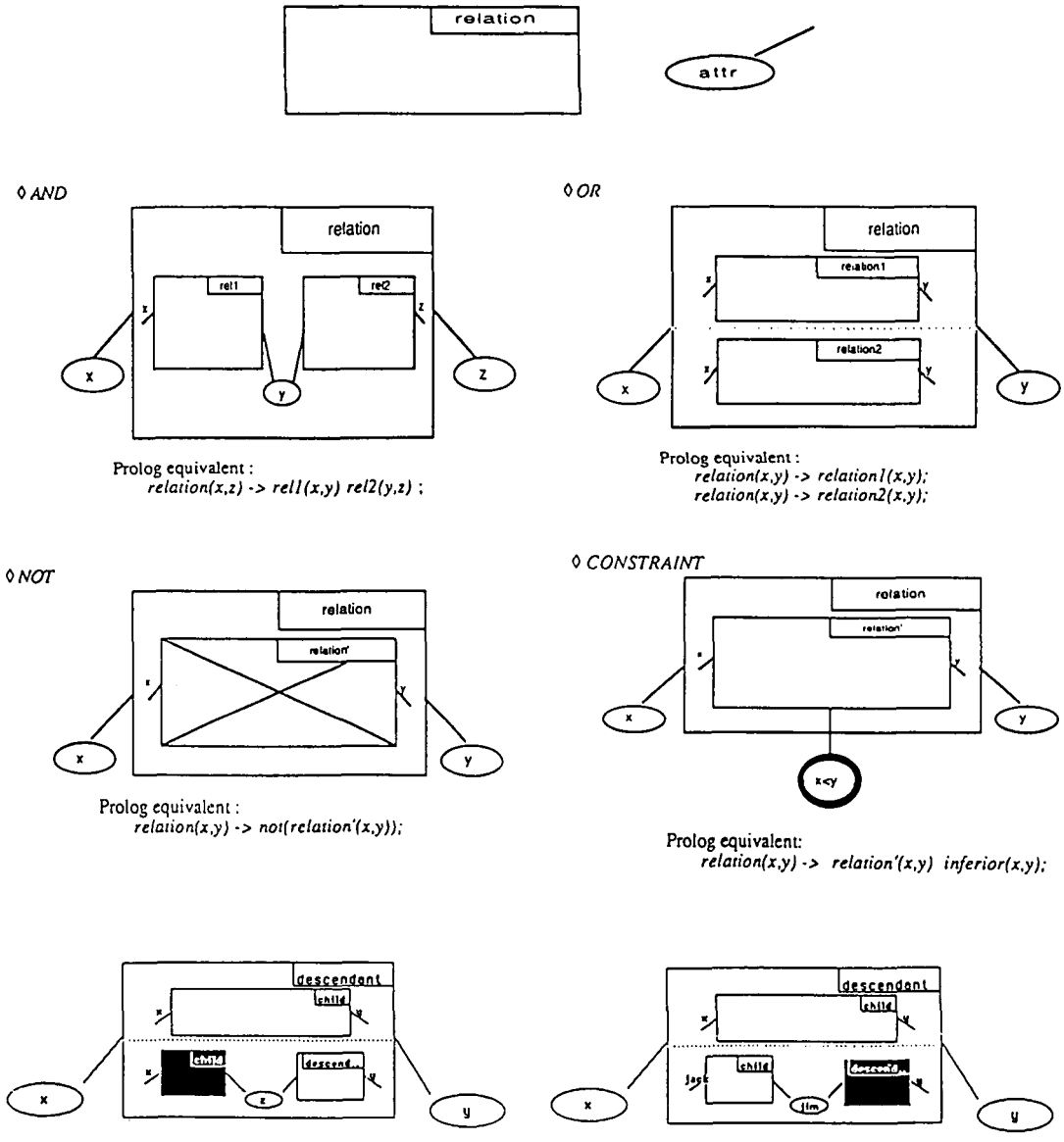

Figure 12 Display of Prolog relations, attributes and operators as suggested by Rueher et al. (1986). Activated relations are shown shaded

Rueher et al. (1986) describe a prototypical system for the graphical construction of Prolog programs based on the representations shown in Fig. 12. With this system, the user can create a program using the graphical symbols shown and watch the process of backtracking during excecution as activated relations are shaded. Eisenstadt and Brayshaw (1988) have also developed a graphical representation of Prolog execution as part of an advanced tracing and debugging facility: the graphical notation shown in Fig. 13 is the basis for a full user environment including a graphical tracer (with "replay" facilities), hierarchical zoom and the ability to cope with search spaces of thousands of nodes. Despite these suggestions, few commercially available implementations yet provide any very sophisticated graphical facilities beyond those available in any workstation windowing environment.

Poplog (Shearer, 1987, Morgan, 1987, du Boulay, 1987) is a multi-language program development environment which lies somewhere between individual languages and their environments and the more sophisticated tools described below. It supports incremental compilers for POP-11, Common Lisp and Prolog and generally uses the kind of WIMP interface common to most workstations. Beyond that, there are generally no graphical facilities apart from those that the knowledge engineer himself may develop for his own use.

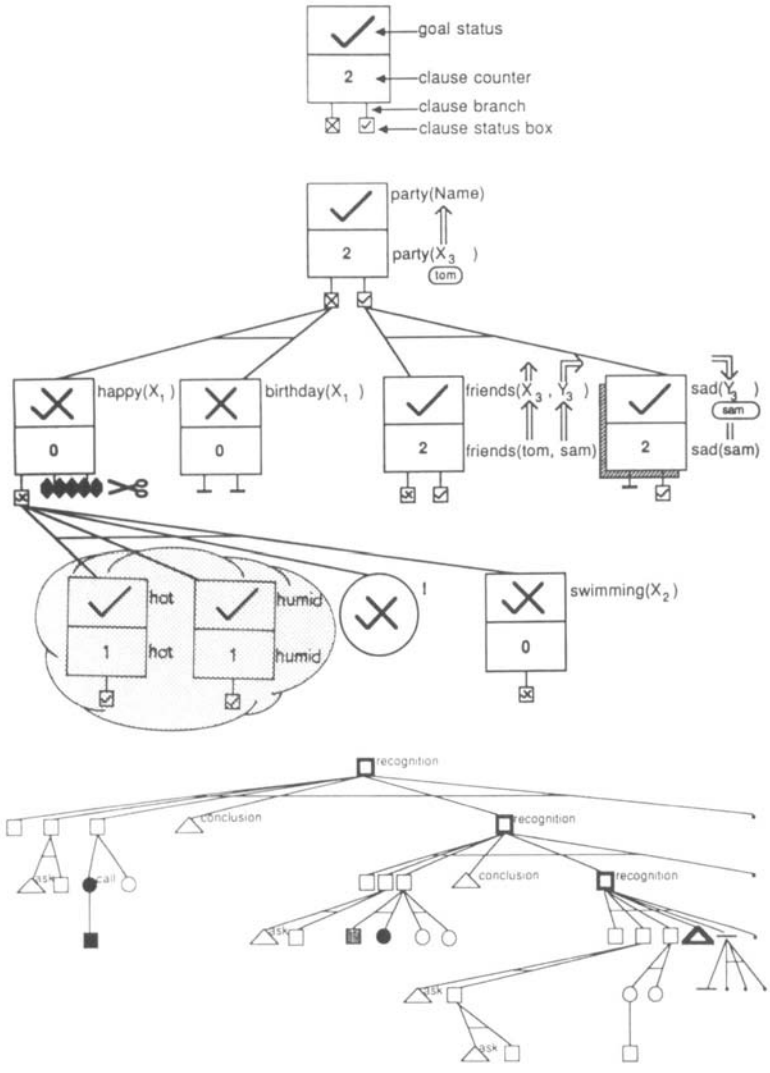

Figure 13 Elements of the graphical tracing and debugging facility suggested for Prolog by Eisenstadt and Brayshaw (1988). State of execution is illustrated by a hierarchy of iconic procedure status boxes which can be collapsed into a "long distance view"

More specialized tools for knowledge engineering range from basic expert system shells run on microcomputers to the powerful knowledge-based systems development tools now available for use on workstations.

The PC-based shells often (though not always) operate in a simple windowing environment which the knowledge engineer may or may not be able to modify to meet his own particular needs. A small number of these shells provide some sort of graphical facilities, though these are usually (with some notable exceptions, such as Nexpert) extremely limited, taking the form of simple "knowledge trees" showing dependencies between rules in the knowledge base or representations of the class object network.

The most widely known tools currently available for knowledge-based system development on workstations are Art, KEE, LOOPS and KnowledgeCraft (Morgan, 1987, Mettrey, 1987, Shearer, 1987, Bobrow and Stefik, 1983, Stefik et al., 1983, Gittins, 1987, Pepper and Kahn, 1986, Baroff et al., 1988, Stelzner and Williams, 1988). Many of the features presented by these systems are broadly similar as all arise originally from Lisp-based environments. All four provide multiple schemes for knowledge representation incorporating versions of frames with inheritance of slots and values as well as rules. They generally support multiple inferencing and control paradigms including rule-based inferencing (forward and backward chaining), procedural control, object-oriented program-

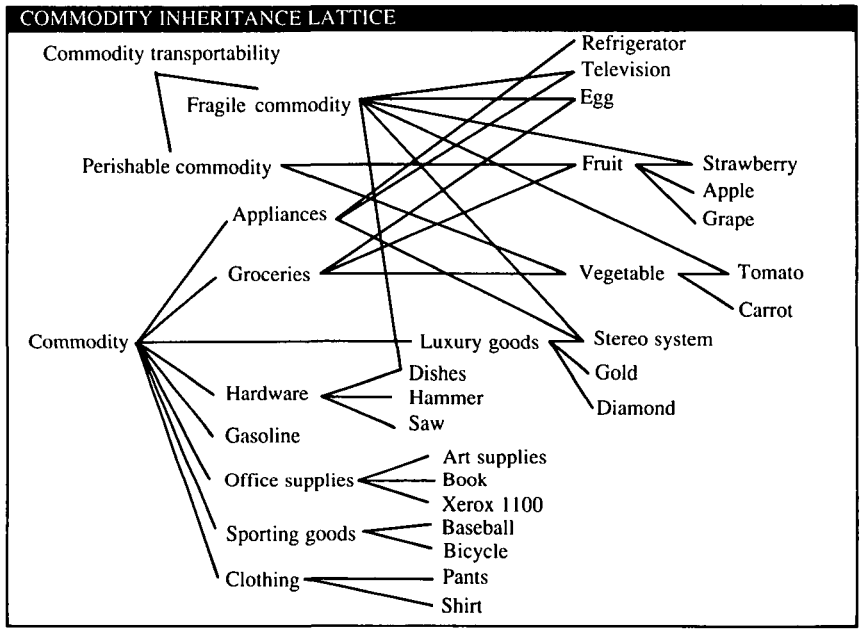

Figure 14 Loops class browser (Stefik et al. (1983)

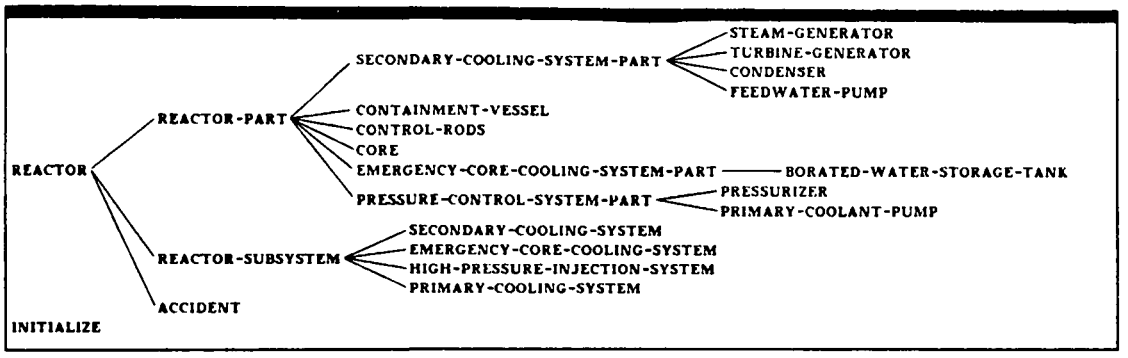

Figure 15 KEE class hierarchy (Kunz, Kehler and Williams, 1987)

ming and access-oriented programming. The development environments are based on the workstation WIMP paradigm using windows and direct manipulation of knowledge graphs to enter and manipulate knowledge in the knowledge base (see Figs. 14 and 15). ART and KEE can also provide graphic tracing of rule activity during inferencing using an active rulegraph (see Fig. 16). KEE and KnowledgeCraft support a graphic display of frames in the form of an inheritance network such that frames can be added to and deleted from the knowledge base by graphically editing the display using a mouse. Furthermore, ART, KEE, KnowledgeCraft and LOOPS all provide graphic toolkits that allow the knowledge engineer to develop custom interfaces consisting of dials, gauges and sliders for the end user (see Fig. 17).

It can be seen that the graphical paradigms used in the features for knowledge engineer support are still relatively simple, consisting of 2-dimensional representations of graphs with textual nodes and simple line links (Figs. 14 and 15), though in KEE's "active rulegraph" for example, some use is made of reverse video to distinguish rules under consideration and "tick" and "cross" icons to show the states of premises and conclusions already considered (see Fig. 16).

A number of other similar tools have also recently been under development, including KEATS (Motta et al., 1986), NEXPERT *OBJECT* (Digital Equipment Corporation, 1988) and Strata (GEC, 1988). These provide broadly similar capabilities to the tools described above as well as some others (for example, KEATS gives help at the knowledge acquisition stage by partly automating the

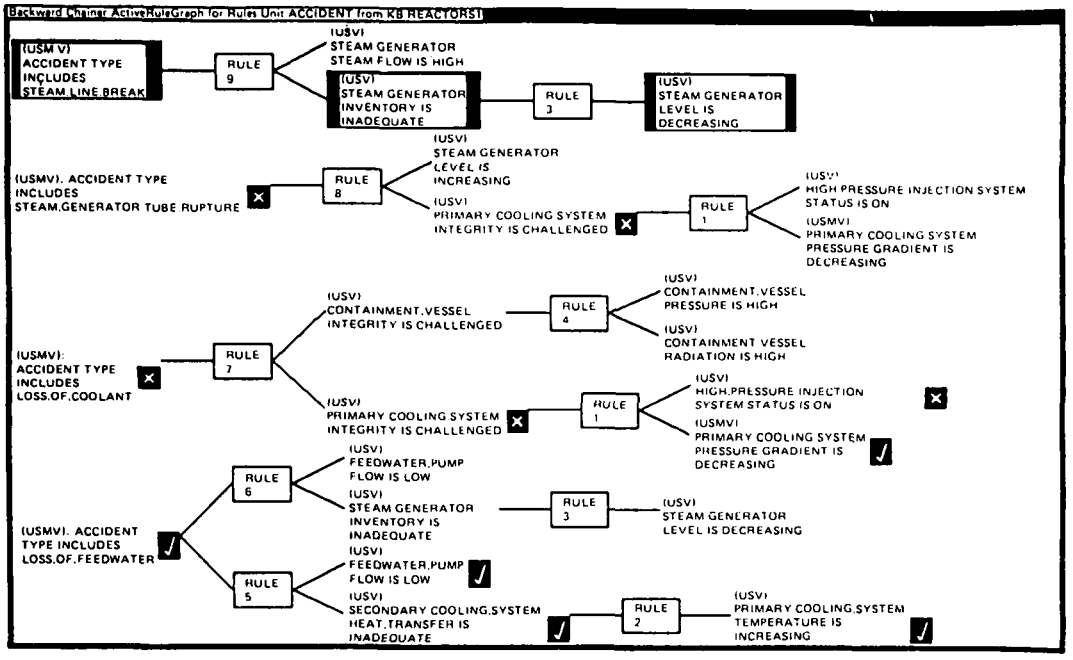

Figure 16 KEE's active rulegraph (Kunz, Kehler and Williams, 1987)

process of transcript analysis). All use similar graphics paradigms involving 2-dimensional representations of graphs with textual nodes and line links. KEATS also allows several different kinds of relations between objects in the knowledge base to be represented using up to eight differently coded link types (consisting of bold lines, dotted lines, dashed lines etc.) and NEXPERT *OBJECT* makes more use of icons similar to the "tick" and "cross" ones used in KEE together with a variety of print styles.

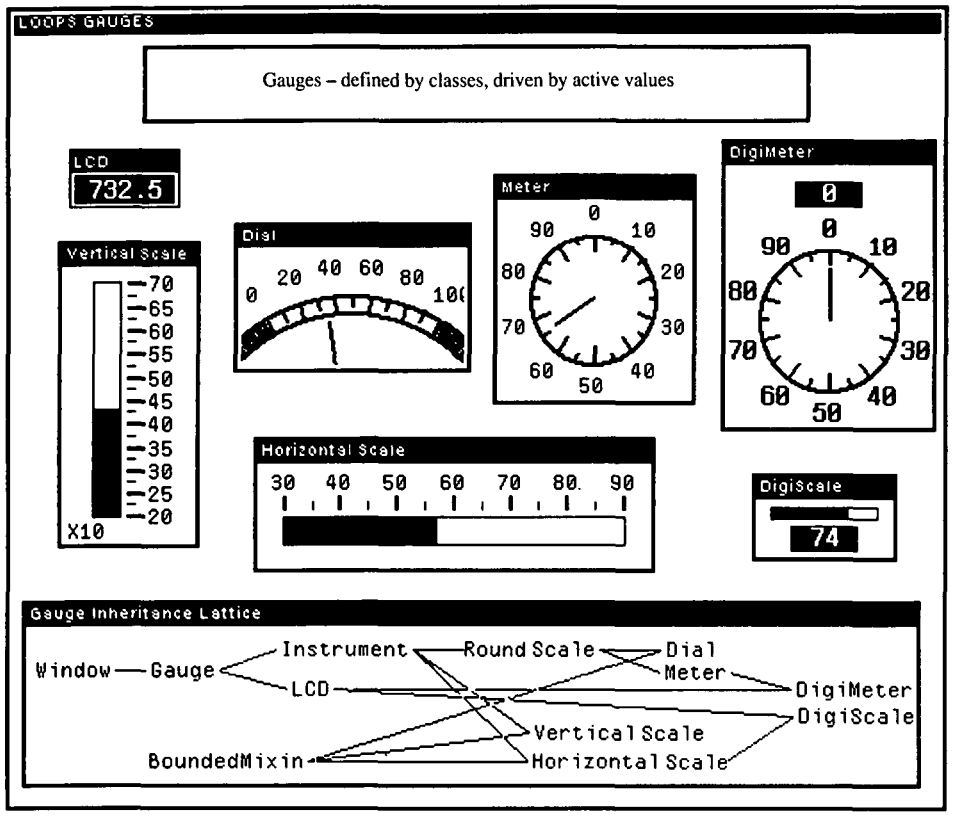

Figure 17 Loops gauges: network shows relationships between classes of gauges (Stefik et al., 1983)

One system which does provide more advanced graphical facilities for knowledge engineering is SemNet which has been developed at Stanford (Fairchild and Poltrock, 1986). This system is not at present commercially available but is intended as a tool for knowledge engineers working on knowledge bases with up to 3,500 elements. SemNet uses a 3-dimensional representation of a structure of nodes and links in "virtual space" around which the user can move. The mouse is used to control movement through the structure in one of three planes according to a "helicopter metaphor" in which the user can also yaw, roll and pitch. Each object node in the initial high level structure may represent several clusters of lower level nodes and can be made to "explode" on approach, revealing its contents. The impression of depth can be enhanced by oscillation of the eye-point as in molecular modelling systems and users can elastically manipulate the structure in order to inspect individual nodes.

Many of the features available in SemNet are still well in advance of those found in the kinds of workstation tools described above. Most of those tools are, however, still under development and it is quite possible that they too will begin to evolve in the direction of more sophisticated graphical facilities as general-purpose workstations become more powerful and more able to cope with the extra processing involved.

See section 7 of the bibliography for further information on knowledge-based system interfaces, especially the collection edited by Hendler, 1988 (which contains Baroff et al., 1988; Hendler and Lewis, 1988; Lehner and Kvalj, 1988; Stelzner and Williams, 1988). Lane, Differding and Shortliffe, 1985; Richer and Clancey, 1985; Tsuji and Shortliffe, 1983, 1985 describe early work on using graphical interfaces for knowledge-based systems. Musen, Fagan and Shortliffe, 1986; Seifert and Rawlings, 1988; Stelzner and Williams, 1988 describe more recent work in that area. For information about graphical interfaces for knowledge engineering, see also section 8 on AI Tools, especially Fairchild and Poltrock, 1986; Gittins, 1987; Kunz, Kehler and Williams, 1987; Stefik et al., 1983.

6 Future directions

Any developments in knowledge-based system interfaces must be viewed in the context of developments in human-computer interfaces in general. I have tried to show how the kinds of issues that must be considered by the designer of a knowledge-based system interface are broadly similar to those considered by any other interface designer: in each case, the main task is to maximise the efficiency of human-computer interaction and hence of the man-machine system as a whole. It is generally argued that this efficiency of interaction is best realized by using interfaces which support the user's natural cognitive processes and structures as far as is possible within the constraints of current technology.

There are two main reasons why this approach cannot yet yield the perfect interface. The first is that understanding of the user's cognitive processes is still relatively limited. Research is being done in this area, but the field is a vast one and progress is relatively slow. It will be a long time before we can say conclusively that we have a complete model of the user's cognitive behaviour.

The second limiting factor is technology. Interaction with computers using today's technology is often far from natural in the sense that it is unlike almost any other form of interaction commonly practised. Work is going on here too with a view to bringing human-computer interaction more into line with other forms. Foley (1987) and Tello (1988) report on experiments with "datagloves" and "datasuits" which translate hand and body movements of the user into a graphical image of the hand or body in a virtual environment giving the user a much stronger sense of direct manipulation than today's mice. Investigations are also being conducted into the use of other senses in human-computer interaction: for example force-feedback can be built into datagloves and datasuits or even more conventional manipulators (Brooks, 1988) so that the user can be given the impression of bumping into objects in a virtual environment. Hearing and speech may of course also be used to a greater extent in future.

The incorporation of most of these developments into everyday interfaces is still some time away. Until then, it seems that vision will probably be the main sense on which we rely for human-

computer interaction. Even so, we are only just beginning to exploit man's advanced visual capabilities and well-developed facilities for hand-eye coordination to anything like the full with today's interactive graphical interfaces and direct manipulation techniques. Substantial progress can be made simply by developing these methods and techniques a little further.

So what does all this mean in terms of the development of knowledge-based system interfaces and interfaces for knowledge engineering in particular?

Again, progress needs to be made on two fronts. Firstly, our understanding of the cognitive behaviour of the knowledge-based system user must improve a great deal before it can form a sound basis for the development of suitable interfaces. The tasks involved in knowledge engineering in particular are at present very poorly understood—knowledge engineers often have only a vague idea of the methods they use in building a knowledge-based system and tend to describe the process as rather ad hoc: an art rather than a science. There has been some work which has attempted to specify formal procedures for knowledge engineering, or at least to define current working practices. None of the proposed methodologies have, however, been very widely adopted to any degree of detail and all are far from becoming a universal standard. In years to come, this kind of work combined with a greater understanding of research by cognitive psychologists into problem solving and decision making strategies might provide a relatively firm foundation on which designers can construct interfaces to appropriately support a range of knowledge engineering tasks. In the meantime, interfaces can be based only on relatively vague working hypotheses about the kinds of tasks to be supported.

Progress can also be made on the technology front. As was suggested above, it is possible that the type of technology on which human-computer interaction is based will be radically different in ten or twenty years' time. In the meantime, there are many small changes to be made which could substantially improve the quality of interaction for all knowledge-based system users. The increasing availability of raw processing power, specialized hardware and high-resolution screens means that graphical representations used in the interface can be made to seem ever more realistic in both appearance and behaviour. This is likely to be of particular benefit to the kinds of direct manipulation interfaces for knowledge-based system end users discussed in section 5.a.

Many more developments in software technology may also be of use in the near future. Work is going on to develop a variety of graphical paradigms with greater expressiveness and flexibility than those illustrated in section 5 (Dodson, 1988). Algorithms and contraints for use in automatic diagram layout are also being investigated with a view to ensuring that computer-generated diagrams of knowledge structures, for example, look good and are easy to understand. Three-dimensional views are of great interest in that they can help the user to make sense of highly complex structures (such as knowledge structures) and fisheye views (Furnas, 1986) are also being considered to allow users to browse over relatively large structures. Brooks (1988) has described investigations into the use of effective depth cues for three-dimensional structures, simulation of realistic motion and use of maps in navigation of complex structures, all of which might be incorporated into future interfaces for knowledge engineers. Finally, the slightly more fine-grained considerations discussed by Verplank (1988) must not be forgotten: graphic design work at the screen as well as the pixel level can substantially improve the overall look and feel of an interface by promoting realism and visual order and providing an appropriate user focus.

Some work which attempts to combine progress on these fronts by attempting an outline of the kinds of tasks involved in knowledge engineering as well as specifying an appropriate type of interface for each is that of Poltrock, Steiner and Nong Tarlton (1986) at the Microelectronics and Computing Corporation in Austin, Texas. They have attempted a basic division of the tasks carried out by a knowledge engineer and suggest a different type of graphical representation as being most appropriate to support each of the three major task types. The first type of representation (see Fig. 18) is a system overview summarizing all rules and their potential interactions. It is suggested that this would be most suitable for exploring an unfamiliar knowledge base. For system validation, the knowledge engineer needs to observe simultaneously the activity of system rules and generated facts. For this purpose, a dynamic display is suggested (see Fig. 19). Finally, to allow the knowledge

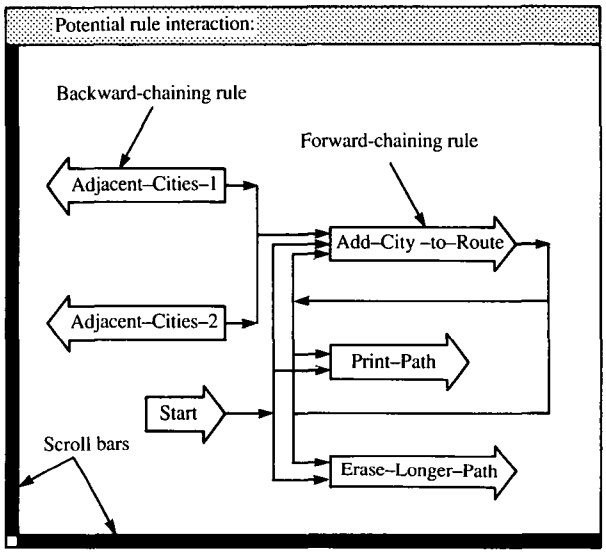

Figure 18 Potential rule interaction display (Poltrock, Steiner and Nong Tarlton, 1986)

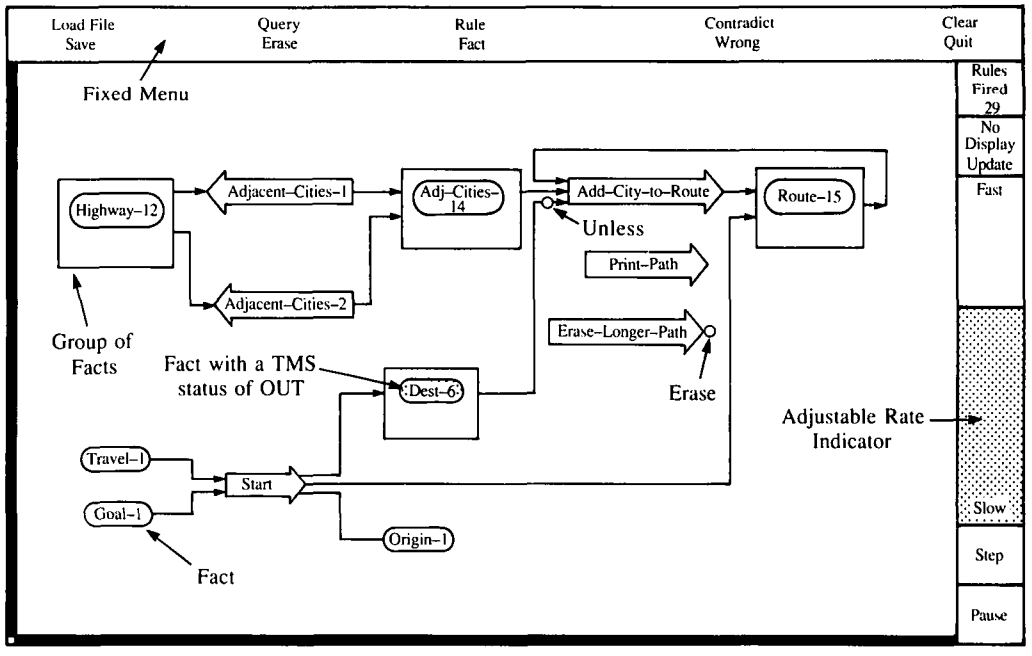

Figure 19 Rule and fact display (Poltrock, Steiner and Nong Tarlton, 1986)

engineer to find out why a particular fact was asserted even when the knowledge base is so large that the relevant rules and facts can't be displayed in the "rule and fact display" described above, a fish-eye display is suggested. Using this facility, the relevant rule and/or fact can be displayed at the centre of the screen with a large number of others shown in the background in less detail (see Fig. 20).

The work of Poltrock et al. seems to be at least a small step forward in the development of suitable interfaces to a knowledge engineer's tool in that it combines some insight into the task of knowledge engineering with a relatively innovative use of interactive computer graphics. The writer is at present aware of no published reports of formal evaluative work aimed at assessing the suitability of graphical interfaces for knowledge engineering. Arguments have, however, been advanced to suggest that graphical interfaces may have several advantages over text-based alternatives in this context. Tools may well benefit from the inclusion of graphical components and options. But in the

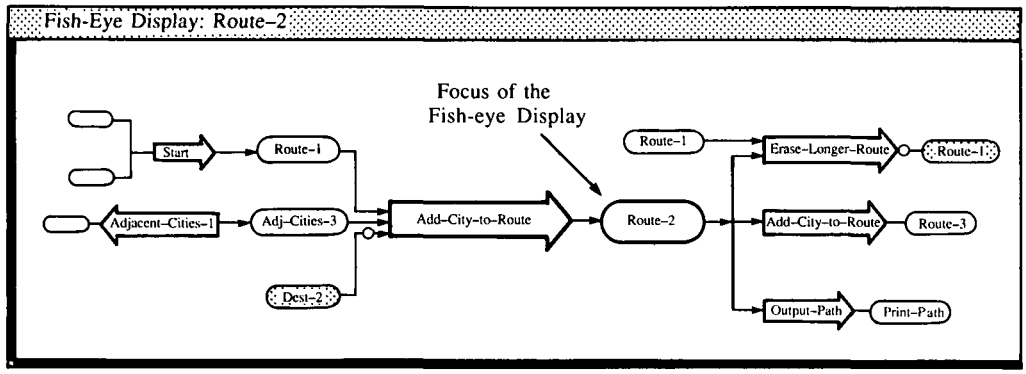

Figure 20 Fish-eye display (Poltrock, Steiner and Nong Tarlton, 1986)

absence of more general research of the kind mentioned above, routine evaluation tests carried out on all aspects of the interface for a new application should include explicit comparisons between graphical and text-based paradigms in order to determine the relative merits of each.

7 Summary

The efficiency of the man-machine system as a whole is determined to a large extent by the efficiency of man-machine interaction, and hence by the quality of the man-machine interface. The aim of human factors engineers is to maximize efficiency of the system as a whole by developing interfaces which accurately fulfil the user's requirements. The emphasis recently in the field of human-computer interaction has been on the user's cognitive requirements and it has been argued that the best interface (all other things being equal) will be the one that best answers the user's cognitive needs and most accurately matches his own natural cognitive processes and structures.

Graphical representations can often be helpful in this respect. In many cases, the user's natural idiom, or the way in which the user naturally thinks of the application domain may itself be graphical. Examples of applications in domains in which graphical interfaces have been found to be particularly useful and suitable have been briefly discussed (see sections 4 and 5). Particular success has been achieved by graphical direct manipulation interfaces based on real world metaphors which allow the user to act in a relatively realistic way on models of the appropriate section of the world to which he is accustomed. These models allow users to exploit experiences and expectations already developed in daily life and therefore give them a head start in comparison with users who must learn a system from scratch.

It is, however, not always easy to determine exactly what the user's "natural" cognitive processes and structures might be. In the case of knowledge-based systems, direct manipulation graphical interfaces are often very suitable for end user interfaces in domains in which graphics is a normal medium for communication, that is, in fields in which heavy use is already made of maps and diagrams. But there is no real world counterpart to the kind of knowledge structure manipulated by the knowledge engineer in designing and developing a knowledge-based system and it is therefore much more difficult to imagine what the knowledge engineer's cognitive representation of such structures might be.

Graphical representations may be highly suitable for use in the interface to knowledge engineering tools and may provide a medium with which the knowledge engineer can work quite easily. Newer knowledge engineering tools increasingly include graphical components which allow the user to work within a highly graphical environment. There is, however, a need for some caution in the move towards increasingly graphical interfaces. There has apparently as yet been little formal evaluative work aimed at comparing the use of graphical and textual interfaces for typical knowledge engineering tasks. Until more general findings are available, each new graphical component of a knowledge engineering tool should be formally evaluated with representative users and tasks to determine its relative merits with respect to its text-based counterparts.

Acknowledgements

I would like to thank British Telecom and the Science and Engineering Research Council for their co-sponsorship of the research program under which this work was carried out. I would also like to thank David Dodson, Lee McCluskey and John Fox for their help in preparing the final draft of this paper.

References

1 Human-computer interaction—general and miscellaneous

Baecker, R M and Buxton, W A S, 1987. *Readings in Human-Computer Interaction* Morgan Kauffman.

Card, S K, Moran, T P and Newell, A, 1983. *The Psychology of Human-Computer Interaction* Erlbaum.

Carroll, J M, (Ed.), 1987. *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction* MIT Press.

Coats, R B and Vlaeminke, I, 1987. *Man-Computer Interfaces: An Introduction to Software Design and Implementation* Blackwell Scientific Publishers.

Foley, J D, 1987. "Interfaces for advanced computing" *Scientific American*, October.

Foley, J D and Van Dam, A, 1982. *Fundamentals of Interactive Computer Graphics* Addison-Wesley.

Foley, J D, Wallace, V L and Chan, P, 1984. "The human factors of computer graphics interaction techniques" *IEEE Computer Graphics and Applications*, November.

Green, M, 1986. "A survey of three dialogue models" *ACM Computer Graphics* **5**(3).

Norman, D A and Draper, S W (Eds.) 1986. *User Centred System Design* Lawrence Erlbaum Associates.

Sanders, M S and McCormick, E J, 1987. *Human Factors in Engineering and Design* McGraw-Hill, 6th edition.

Shneiderman, B, 1987. *Designing the User Interface* Addison-Wesley.

Shneiderman, B, 1980. *Software Psychology: Human Factors in Computer and Information Systems* Winthrop.

Smith, S L and Mosier, J N, 1986. *Guidelines for Designing User Interface Software* Mitre.

Tello, E, 1988. "Between man and machine" *Byte* September.

Woods, D D, 1986. "Cognitive technologies: the design of joint human-machine cognitive systems" *AI Magazine* **6**(4).

2 Perceptual aspects of displays

- See also *Cognitive aspects of interface design.*
- See also *Human-computer interaction—general and miscellaneous.*

Benbasat, I and Dexter, A S, 1985. "An experimental evaluation of graphical and color-enhanced information presentation" *Management Science* **31**(11).

Benbasat, I, Dexter, A S and Todd, P, 1986. "The influence of color and graphical information presentation in a managerial decision simulation" *Human-Computer Interaction* **2**.

Brooke, J B and Duncan, K D, 1981. "Effects of system display format on performance in a fault location task" *Ergonomics* **24**(3).

Christ, R E, 1975. "Review and analysis of colour coding research for visual displays" *Human Factors* **17**(6).

Cleveland, W S and McGill, R, 1987. "Graphical perception: the visual decoding of quantitative information on graphical displays of data" *Journal of the Royal Statistical Society* **150**(3).

Cleveland, W S and McGill, R, 1984. "Graphical perception: theory, experimentation and application to the development of graphical methods" *Journal of the American Statistical Association* **79**(387).

Davidoff, J, 1987. "The role of colour in visual displays". In: *International Reviews of Ergonomics, 1*, D J Osborne (Ed.) Taylor and Francis.

Murch, G, 1984. "The effective use of color: cognitive principles" *TEKniques* **8**(2).

Murch, G M, 1984. "Physiological principles for the effective use of colour" *IEEE Computer Graphics and Applications* November.

Scrivener, S, 1983. "Perceptual factors in the design of information displays". In: *Computer Graphics '83*. Online conference.

Tullis, T S, 1981. "An evaluation of alphanumeric, graphic and color information displays" *Human Factors* **23**(5).

Tullis, T S, 1986. "Optimizing the usability of computer generated displays." In: *Proceedings of HCI'86.*

3 Cognitive aspects of interface design

- See also *Perceptual aspects of displays.*
- See also *Human-computer interaction—general and miscellaneous.*

Allen, R B, 1982. "Cognitive factors in human interaction with computers" In: *Directions in Human/Computer Interaction*, A Badre and B Schneiderman (Eds.) Ablex.

Gardiner, M M and Christie, B, (Eds.) 1987. *Applying Cognitive Psychology to User Interface Design* John Wiley.

Gaylin, K B, 1986. "How are windows used? some notes on creating an empirically-based windowing benchmark test" In: *Proceedings CHI '86* ACM.

Graesser, A C, Lang, K L and Elofson, C S, 1987. "Some tools for redesigning system-operator interfaces" In: *Applications of Cognitive Psychology: Problem Solving, Education and Computing*, D E Berger, K Pedzek and W P Banks (Eds.) Lawrence Erlbaum Associates.

Marshall, C, Christie, B and Gardiner, M M, 1987. "Assessment of trends in the technology and techniques of human-computer interaction" In: *Applying Cognitive Psychology to User Interface Design*, M M Gardiner and B Christie (Eds.) Wiley.

Marshall, C, Nelson, C and Gardiner, M M, 1987. "Design guidelines" In: *Applying Cognitive Psychology to User Interface Design*, M M Gardiner and B Christie (Eds.) Wiley.

Norman, D A, 1987. "Design principles for human-computer interfaces" In: *Applications of Cognitive Psychology: Problem Solving, Education and Computing*, D E Berger, K Pezdek and W P Banks (Eds.) Lawrence Erlbaum Associates.

Norman, K L, Weldon, L J and Schneiderman, B, 1986. "Cognitive layout of windows and multiple screens for user interfaces" *International Journal of Man-Machine Studies* **25**.

Woods, D D, 1986. "Cognitive technologies: the design of joint human-machine cognitive systems" *AI Magazine* **6**(4).

4 Use of graphical interfaces—general

- See also *Applications with graphical interfaces.*

Barth, P S, 1986. "An object-oriented approach to graphical interfaces" *ACM Transactions on Computer Graphics* **5**(2).

Benbasat, I and Dexter, A S, 1985. "An experimental evaluation of graphical and color-enhanced information presentation" *Management Science* **31**(11).

Benbasat, I, Dexter, A S and Todd, P, 1986. "The influence of color and graphical information presentation in a managerial decision simulation" *Human-Computer Interaction* **2**.

Bocker, H-D, Fischer, G and Nieper, H, 1986. "The enhancement of understanding through visual representations" In: *Proceedings CHI '86* ACM.

Bocker, H-D and Nieper, H, 1985. "Making the invisible visible: tools for exploratory programming" In: *Proceedings of the 1st Pan Pacific Computer Conference.*

Brooke, J B and Duncan K D, 1981. "Effects of system display format on performance in a fault location task" *Ergonomics* **24**(3).

Brooks, F P, 1988. "Grasping reality through illusion—interactive graphics serving science" In: *CHI '88 Proceedings* ACM.

Clemons, E K and Greenfield, A J, 1985. "The sage system architecture: a system for the rapid development of graphics interfaces for decision support" *IEEE Computer Graphics and Applications* November.

Fitter, M J and Green T R G, 1981. "When do diagrams make good computer languages?" In: *Computing Skills and the User Interface*, L Alty and M J Coombs (Eds.) Academic Press.

Foley, J D and Wallace, V L, 1974. "The art of natural graphic man-machine conversation" *Proceedings of the IEEE* **62**(4).

Furnas, G W, 1986. "Generalised fisheye views" In: *CHI '86 Proceedings* ACM.

Henderson, D A and Card, S K, 1986. "Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface" *ACM Transactions on Computer Graphics* **5**(3).

Hollan, J D, Hutchins, E L, McCandless, T P, Rosenstein, M and Weitzman, L, 1987. "Graphic interfaces for simulation" *Advances in Man-Machine Systems Research* **3**.

Mackinlay, J, 1986. "Automating the design of graphical presentations of relational information" *ACM Computer Graphics* **5**(2).

Myers, B A, 1986. "Visual programming, programming by example and program visualisation: a taxonomy" In: *Proceedings CHI '86* ACM.

Senach, B, 1983. "Computer-aided problem solving with graphical display of information" In: *Psychology of Computer Use* Academic Press.

Verplank, B, 1988. Tutorial given at HCI88.

Zdybel, F, Greenfield, N R, Yonke, M D and Gibbons, J, 1981. "An information presentation system" In: *Proceedings of the 7th International Joint Conference on AI* AAAI.

5 Applications with graphical interfaces

- See also *Use of graphical interfaces—general.*
- See also *Knowledge based system interfaces.*

Borning, A, 1986. "Defining constraints graphically" In: *Human Factors in Computing Systems: Proceedings SIG-CHI '86* ACM.

Bryce, D and Hull, R, 1986. "Snap: a graphics-based schema manager" In: *Proceedings of the IEEE International Conference on Data Engineering.*

Clemons, E K and Greenfield, A J, 1985. "The sage system architecture: a system for the rapid development of graphics interfaces for decision support" *IEEE Computer Graphics and Applications* November.

Delisle N and Schwartz, M, 1986. "Neptune: a hypertext system for cad applications" In: *Proceedings of ACM SIGMOID '86.*

Eisenstadt, M and Brayshaw, M, 1988. "Aorta: diagrams as an aid to visualising the execution of prolog programs" In: *Graphics Tools for Software Engineering*, A C Kilgour and R A Earnshaw (Eds.) BCS Documentation and Displays Group.

Feiner, S, 1985. "Apex: an experiment in the automated creation of pictorial explanations" *IEEE Computer Graphics and Applications* November.

Frasson, C and Er-radi, M, 1986. "Principles of an icons-based command language" In: *Proceedings ACM SIGMOD International Conference on Management of Data.*

Glinert, E P and Tanimoto, S L, 1984. "Pict: and interactive graphical programming environment" *IEEE Computer* **17**(11).

Goldman, K J, Goldman, S A, Kannelakis, P C and Zdonik, S B, 1985. "Isis: interface for a semantic information system" In: *Proceedings of the ACM SIGMOD International Conference on Management of Data.*

Green, T R G, 1982. "Pictures of programs and other processes, or how to do things with lines" *Behaviour and Information Technology* **1**(1).

Halbert, D C, 1984. *Programming by Example* PhD thesis, Computer Science Division, University of California, Berkeley.

Hollan, J D, Hutchins, E L and Weitzman, L, 1984. "Steamer: an interactive inspectable simulation-based training system" *AI Magazine* **5**(2).

Knapp, B G, Moses F L and Gellman, L H, 1982. "Information highlighting on complex displays" In: *Directions in Human/Computer Interaction*, A Badre and B Shneiderman (Eds.) Ablex.

Pietrzykowski, T, Matwin, S and Muldner, T, 1983. "The programming language prograph: yet another application of graphics" In: *Graphics Interface '83.*

Reid, P, 1988. "Dynamic interactive display of complex data structures" In: *Graphics Tools for Software Engineering*, A C Kilgour and R A Earnshaw (Eds.) BCS Documentation and Displays Group.

Ripka, W, 1988. "Computers picture the perfect drug" *New Scientist*, June.

Rueher, M, Thomas, M-C, Gubert, A and Ladret, D, 1986. "A prolog based graphical approach for knowledge expression" *Microsoftware for Engineers* **24**(4).

Shneiderman, B, Mayer, R, McKay, D and Heller, P, 1977. "Experimental investigations of the utility of detailed flowcharts in programming" *Communications of the ACM* **20**(6).

Vose, G M and Williams, G, 1988. "Labview: laboratory virtual instrument engineering workbench" *Byte* September.

Yasdi, R, 1985. "A conceptual design aid environment for expert-database systems" *Data and Knowledge Engineering.*

Zhang, Z and Mendelzon, A O, 1983. "A graphical query language for entity relationship databases" In: *Entity Relationship Approach to Software Engineering*, C G Davis, S Jahodia, P A Ng and R T Yeh (Eds.) Elsevier.

6 Knowledge-based systems—general and miscellaneous

Black, W J, 1986. *Intelligent Knowledge Based Systems: An Introduction* Van Nostrand Reinhold.

Brachman, R J and Levesque, H J (Eds.) 1985. *Readings in Knowledge Representation* Morgan Kaufmann.

Buchanan, B G and Duda, R O, 1982. *Principles of Rule-Based Systems* Technical Report, Stanford University Heuristic Programming Project, HPP-82-14.

Diaper, D, 1988. "The promise of pomess: a people oriented methodology for expert system specification" In: *Human and Organisational Issues of Expert Systems* May.

Hayes-Roth, F, 1985. "Rule-based systems" *Communications of the ACM* **28**(9).

Hoffman, R R, 1987. "The problem of extracting the knowledge of experts from the perspective of experimental psychology" *AI Magazine.*

Jackson, P, 1986. *Introduction to Expert Systems* Addison-Wesley.

Sell, P S, 1985. *Expert Systems—A Practical Introduction* Macmillan.

Shadbolt, N and Burton, M, 1987. "Tutorial on knowledge elicitation" Course material, 1987. Department of Psychology, University of Nottingham.

Thompson, B A and Thompson, W A, 1985. "Inside an expert system" *Byte* April.

Welbank, M, "A review of knowledge acquisition techniques" British Telecom Memorandum R19/022/83.

7 Knowledge-based system interfaces

- See also *AI tools.*

Badler, N I and Finin, T W, 1985. "Computer graphics and expert systems" *IEEE Computer Graphics and Applications* **5**(11).

Baroff, J, Simon, R, Gilman, F and Shneiderman, B, 1988. "Direct manipulation user interfaces for expert systems" In: *Expert Systems: The User Interface*, A Hendler (Ed.) Ablex.

Berry, D C and Broadbent, D E, 1986. "Expert systems and the man-machine interface" *Expert Systems* **3**(4).

Dodson, D C, 1988. "Interaction with knowledge systems through connection diagrams: please adjust your diagrams" In: *Proceedings Expert Systems '88* (to appear).

Dodson, D C, 1987. "Interaction with knowledge systems through structure diagrams: perspectives and problems" In: *Abridged Proceedings of the 2nd International Conference on Human-Computer Interaction.*

Freiling, M J and Alexander, J H, 1984. "Diagrams and grammars: tools for mass-producing expert systems" In: *Proceedings of the 1st Conference on AI Applications* IEEE.

Carnegie Group, 1986. "Language craft version 3.1" Sales literature.

Hayes, P J, 1988. "Using a knowledge base to drive an expert system interface with a natural language component" In: *Expert Systems: The User Interface*, J A Hendler (Ed.) Ablex.

Hendler, J A, (Ed.) 1988. *Expert Systems: The User Interface* Ablex.

Hendler, J A and Lewis, C, 1988. "Designing interfaces for expert systems" In: *Expert Systems: The User Interface*, J A Hendler (Ed.) Ablex.

Kearney, P J and Hunt, B, 1987. *A Graphical Aid to IKBS Program Development* Technical Report, IKBS Group BAe, Preston.

Kidd, A L. Description of r19.1.2's alvey proposal entitled "graphics interface for expert systems project" R19.1.2. Group Memorandum 84/7, British Telecom.

Kidd, A L and Cooper, M B, 1983. "Man-machine interface for an expert system" In: *Proceedings of Expert Systems '83* BCS Expert Systems Special Interest Group.

Lane, C D, Differding, J C and Shortliffe, E H, 1985. *Graphical Access to Medical Expert Systems: II Design of an Interface for Physicians* Technical Report, Stanford Knowledge Systems Laboratory. KSL-85-15.

Lehner, P E and Kralj, M M, 1988. "Cognitive impacts of the user interface" In: *Expert Systems: The User Interface*, J A Hendler (Ed.) Ablex.

Lewis, J W, 1986. "The inference machine laboratory: graphic tools for knowledge management" In: *Proceedings Graphics and Vision Interface '86.*

McAleese, R, 1987. "The graphical representation of knowledge as an interface to knowledge based systems" In: *Aspects of Educational technology, volume 22*, H Methias and R Budget (Eds.) Kogan Page.

Musen, M A, Fagan, L M and Shortliffe, E H, 1986. *Graphical Specification of Procedural Knowledge for an Expert System.* Technical Report, IEEE Computer Society. Reprinted from the 1986 Computer Society Workshop on Visual Languages.

Poltrock, S E, Steiner, D D and Nong Tarlton, P, 1986. "Graphic interfaces for knowledge-based system development" In: *Proceedings CHI '86* ACM.

Richer, M H and Clancey, W J, 1985. "Guidon-watch: a graphic interface for viewing a knowledge based system" *IEEE Computer Graphics and Applications* **5**(1).

Seifert, K and Rawlings, C, 1988. *GRIPE: A Graphical Interface to a Knowledge Based System which Reasons About Protein Topology* Technical Report, Imperial Cancer Research Fund. Paper presented at HCI88.

Shneiderman, B, 1988. "Series editor's preface" In: *Expert Systems: The User Interface*, J A Hendler (Ed.) Ablex.

Stelzner, M and Williams M D, 1988. "The evolution of interface requirements for expert systems" In: *Expert Systems: The User Interface*, J A Hendler (Ed.) Ablex.

Tsuji, S and Shortliffe, E H, 1983. *Graphical Access to the Knowledge Base of a Medical Consultation System* Technical Report HPP-83-6, Stanford Heuristic Programming Project.

Tsuji, S and Shortliffe, E H, 1985. *Graphics for Knowledge Engineers: A Window on Knowledge Base Management.* Technical Report, Stanford Knowledge Systems Laboratory. KSL-85-11.

Woods, D D, 1986. "Paradigms for intelligent decision support" In: *Intelligent Decision Support in Process Environments*, E Hollnagel (Ed.) Springer-Verlag.

8 AI Tools

Digital Equipment Corporation, 1988. *Nexpert Object: Expert System Development for the VMS Operating System Environment* Sales literature.

GEC Research, 1988. *STRATA* Sales literature.

Bobrow, D G and Stefik, M, 1983. *The LOOPS Manual* Xerox Corporation.

du Boulay, B, 1987. "Poplog for beginners: a powerful environment for learning programming" In: *Artificial Intelligence Programming Environments*, R Hawley (Ed.) Ellis Horwood.

Fairchild, K and Poltrock, S, 1986. *SemNet* Technical Report HI-104-86, MCC. Video.

Freiling, M J and Alexander, J H, 1984. "Diagrams and grammars: tools for mass-producing expert systems" In: *Proceedings of the 1st Conference on AI Applications* IEEE.

Gittins, M, 1987. "Loops: a multi-paradigm environment" In: *Artificial Intelligence Programming Environments*, R Hawley (Ed.) Ellis Horwood.

Carnegie Group, 1987. "Knowledge craft overview" Technical brief.

Hasemer, T, 1987. "Lisp programming" In: *Artificial Intelligence Programming Environments,* R Hawley (Ed.) Ellis Horwood.

Hawley, R (Ed.) 1987. *Artificial Intelligence Programming Environments* Ellis Horwood.

Kunz, J C, Kehler, T P and Williams, M D, 1987. "Applications development using a hybrid ai development system" In: *Artificial Intelligence Programming Environments,* R Hawley (Ed.) Ellis Horwood.

Mettrey, W, 1987. "An assessment of tools for building large knowledge-based systems" *AI Magazine.*

Morgan, T, 1987. "An overview of tools and languages" In: *Proceedings of KBS '87* Online Publications.

Motta, E, Eisenstadt, M, West, M, Pitman, K and Evertsz, R, 1986. *KEATS: The Knowledge Engineer's Assistant. Final Project Report* Technical Report 20, Human Cognition Research Labs.

Pepper, J and Kahn, G, 1986. *Knowledge Craft: An Environment for Rapid Prototyping of Expert Systems* Technical Report, Carnegie Group Inc.

GEC Research, 1987. "An introduction to the strata ai toolkit" reference Y/295/4813.

Richer, M H, 1985. *Evaluating the Existing Tools for Developing Knowledge-Based Systems.* Technical Report, Stanford Knowledge Systems Laboratory. KSL 85-19.

Shearer, C, 1987. "Kee and poplog: alternative approaches to developing major knowledge based systems" In: *Proceedings of KBS '87* Online Publications.

Stefik, M, Bobrow, D G, Mittal, S and Conway, L, 1983. "Knowledge programming in loops: report on an experimental course" *AI Magazine.*

9 Miscellaneous

Atkinson, R L, Atkinson R C, Smith, E E and Hilgard, E R, 1987. *Introduction to Psychology* Harcourt, Brace and Jovonich, 9th edition.

Bainbridge, L, 1988. "Types of representation" In: *Tasks, Errors and Mental Models,* L P Goodstein, H B Andersen and S E Olsen (Eds.) Taylor and Francis.

Bareiss, E R and Porter, B W, 1987. *A Survey of Psychological Models of Concept Representation.* Technical Report, Artificial Intelligence Laboratory, The University of Texas at Austin. AI TR87-50.

Borning, A, 1981. "The programming language aspects of thinglab; a constraint-oriented simulation laboratory" *Transactions on Programming Language and Systems* **3**(4).

Borning, A, 1979. *Thinglab—A Constraint-Oriented Simulation Laboratory* Technical Report, Xerox Palo Alto Research Centre. SSL-79-3.

Brooks, R E, 1980. "Studying programmer behaviour experimentally: the problems of proper methodology" *Communications of the ACM* **23**(4).

Chapanis, A, 1967. "The relevance of laboratory studies to practical situations" *Ergonomics* **10**(5).

Conklin, J, 1987. "Hypertext: an introduction and survey" *IEEE Computer* September.

Foley, J, 1986. "Guest editor's introduction to the special issue on user interface software" *ACM Transactions on Computer Graphics* **5**(3).

Galambos, J A, Abelson, R P and Black, J B, 1986. *Knowledge Structures* Lawrence Erlbaum Associates.

Harel, D, 1988. "On visual formalisms" *Communications of the ACM* **31**(5).

Johnson-Laird, P N and Wason P C (Eds.) 1977. *Thinking: Reading in Cognitive Science* Cambridge University Press.

Kearney, P J and Hunt, B, 1987. *A Graphical Aid to IKBS Program Development* Technical Report, IKBS Group, BAe, Preston.

Kiss, G and Pinder, R, 1986. "The use of complexity theory in evaluating interfaces" In: *Proceedings of HCI '86* BCS HCI Special Interest Group.

Livingstone, M S, 1988. "Art, illusion and the visual system" *Scientific American* January.

Mackinlay, J, 1986. "Automating the design of graphical presentations of relational information" *ACM Computer Graphics* **5**(2).

Mackinlay, J, 1983. *Intelligent Presentation: The Generation Problem for User Interfaces* Technical Report, Stanford University Heuristic Programming Project. HPP-83-34 (thesis proposal).

MacLean, A, Barnard, P and Hammond, N, 1984. "Recall as an indicant of performance in interactive systems" In: *Human-Computer Interaction—Interact '84,* B Shackel (Ed.) Elsevier Science Publishers.

Myers, B A and Buxton, W, 1986. "Creating highly interactive and graphical user interfaces by demonstration" *ACM Computer Graphics* **20**(4).

Rumelhart, D E, Smolensky, P, McClelland, J L and Hinton, G E, 1986. "Schemata and sequential thought processes in pdp models" In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 2: Psychological and Biological Models,* J L McClelland, D E Rumelhart and the PDP Research Group (Eds.) MIT Press.

Shephard, R N and Metzler, J, 1977. "Mental rotation of three-dimensional objects" In: *Thinking, Reading in Cognitive Science,* P N Johnson-Laird and P C Wason (Eds.) CUP.

Tamassia, R, Batini, C and Di Battista, G, 1988. "Automatic graph drawing and readability of diagrams" *IEEE Transactions on System, Man and Cybernetics* To appear.

Winograd, T, 1977. "Formalisms for knowledge" In: *Thinking, Readings in Cognitive Science*, P N Johnson-Laird and P C Wason (Eds.) CUP.

Wright, P and Reid, F, 1973. "Written information: some alternatives to prose for expressing the outcomes of complex contingencies" *Journal of Applied Psychology* **57**(2).