4th AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop
Banff, Canada, October, 1989.

A HETEROGENEOUS KNOWLEDGE SUPPORT SYSTEM

Brian R Gaines[1], Alain T Rappaport[2] & Mildred L G Shaw[1]

Knowledge Science Institute[1]
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4

Neuron Data[2]
444 High Street
Palo Alto
CA94301, USA

ABSTRACT

A knowledge support system is presented that combines hypermedia, knowledge acquisition, expert system shell and database systems to provide an integrated end-user environment. The integration is achieved through inter-application communication protocols operating between four programs designed independently for stand-alone use. The system supports a wide variety of forms and representations of knowledge from the highly informal to the highly computational. It is extremely non-modal and allows knowledge acquisition and knowledge consultation to be combined in a variety of ways. In particular, the results of a consultation may be used to create a new case that may be added to the case base of the acquisition tool which may then be re-analyzed, possibly resulting in changed advice.

INTRODUCTION

This paper presents the architecture and operation of a knowledge support system providing a diversity of knowledge representations, knowledge acquisition paths and validation procedures. The architecture is essentially heterogeneous, based on the loose coupling of four programs originally written as independent stand-alone systems. A uniform and consistent end-user interface is maintained through the commonality and consistent use of the Macintosh window, control and image structures in all systems. Extreme non-modality is supported in that users can switch between tasks and applications freely, either within the integrated application or through the underlying operating system. Multiple partial elicitations, consultations and edits can be undertaken in parallel and left indefinitely in intermediate states. Interlocks are provided to ensure that consistency is maintained in the underlying distributed, heterogeneous knowledge base. The knowledge support system can be used in a highly focused, goal-directed manner for knowledge consultation, or in a highly exploratory, serendipic manner for knowledge creation, or in combination so that application and enhancement are mutually supportive activities.

Any technology carries with it the danger of the Procrustean bed, that we shape our tasks to the available tools and, in so doing, trivialize them rather than address them. When we bring the automata-theoretic precision of computer technology to bear upon knowledge processes, we may filter from the knowledge only that which is computational and, in so doing, lose the essence of what may be known. Fortunately, computer technology has been applied to so many layers of our social activities that links exist to many of the existing media through which we express knowledge, many of which are not truly computational in that their semantics are, as yet, not subject to computer interpretation. Sounds, images and text can be digitized or placed under computer control, and access to them can be made part of a knowledge-based system. Even the semantics of appropriate access do not need to be subject to computer interpretation in that the human user of the system can decide to do so completely outside any computational context.

These considerations extend beyond computer technology to media and discourse processes involving no computer interaction. Our objective in this development has been to recognize that knowledge support systems are part of the technological infrastructure facilitating improved quality and productivity in the knowledge processes of society. Figure 1 shows a model of the knowledge acquisition process that we have previously described in some detail (Rappaport & Gaines 1988). Our systems development is targeted on providing support for as many of the variety of processes shown as is possible with current technology. A major part of the contribution is the computational formalization of knowledge so that it becomes operational, but, to be effective and acceptable, this has to be linked to the rich infrastructure of informal and semi-formal knowledge that affects human behavior even though it not computer-interpretable.

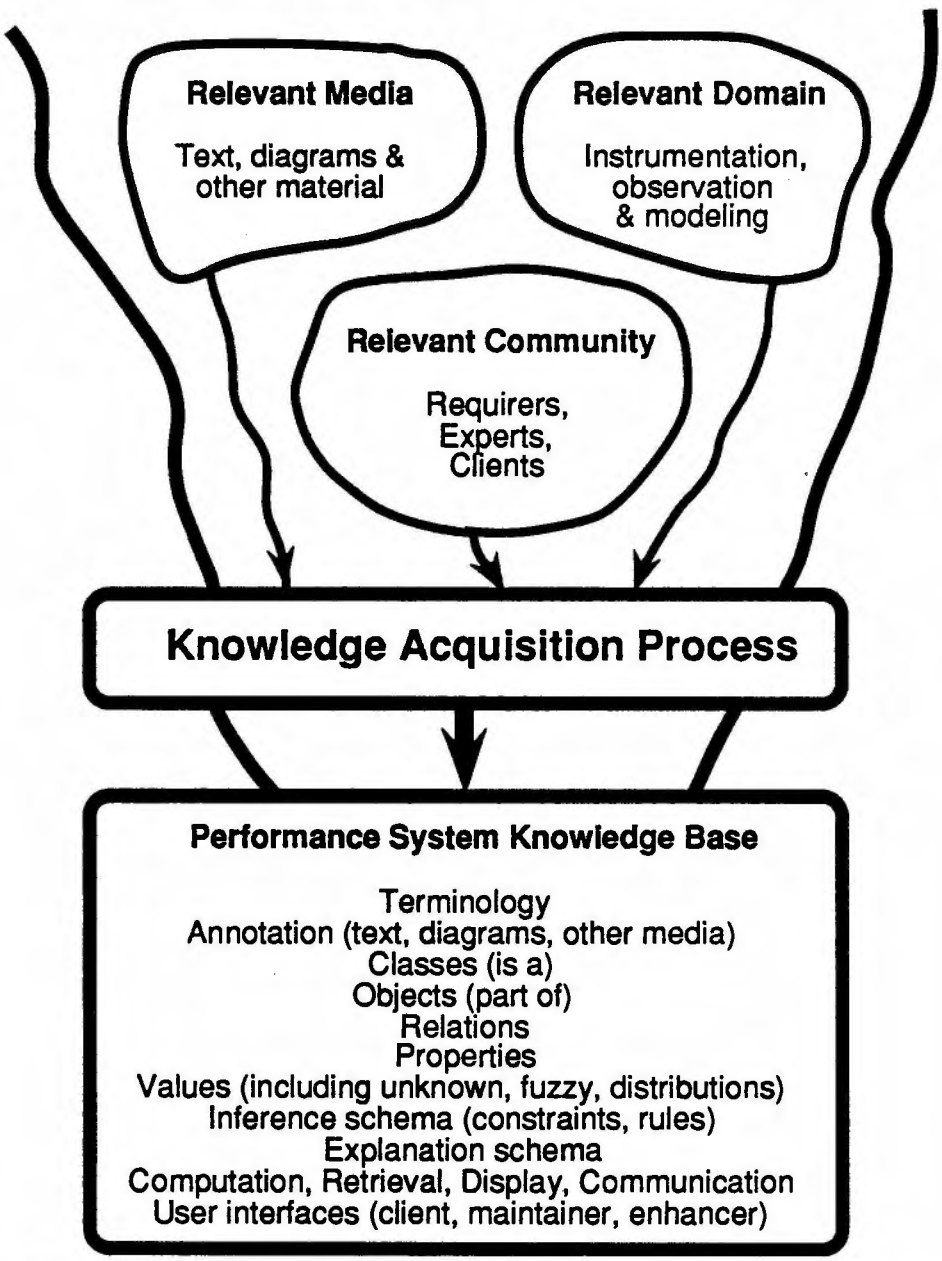

Fig.1 The knowledge acquisition process for knowledge-based systems

KNOWLEDGE SUPPORT SYSTEM ARCHITECTURE

We have previously described two of the tools involved, some of their functionality, and some of the major integration problems in detail (Rappaport & Gaines 1988). Figure 2 was presented as our design objective for a comprehensive knowledge support system. We see a reasonable 'knowledge base' as commencing with fairly informal knowledge that has little structure. Structure may be added to part of this in a variety of ways by annotating it, particularly with cross-references and linkages. Part of this structure may be formalized within various generic ontologies. Part of this formalization may be made operational by encoding it in computational form. Shaw and Woodward (1989) have analyzed the cognitive processes underlying this multi-stage knowledge representation.

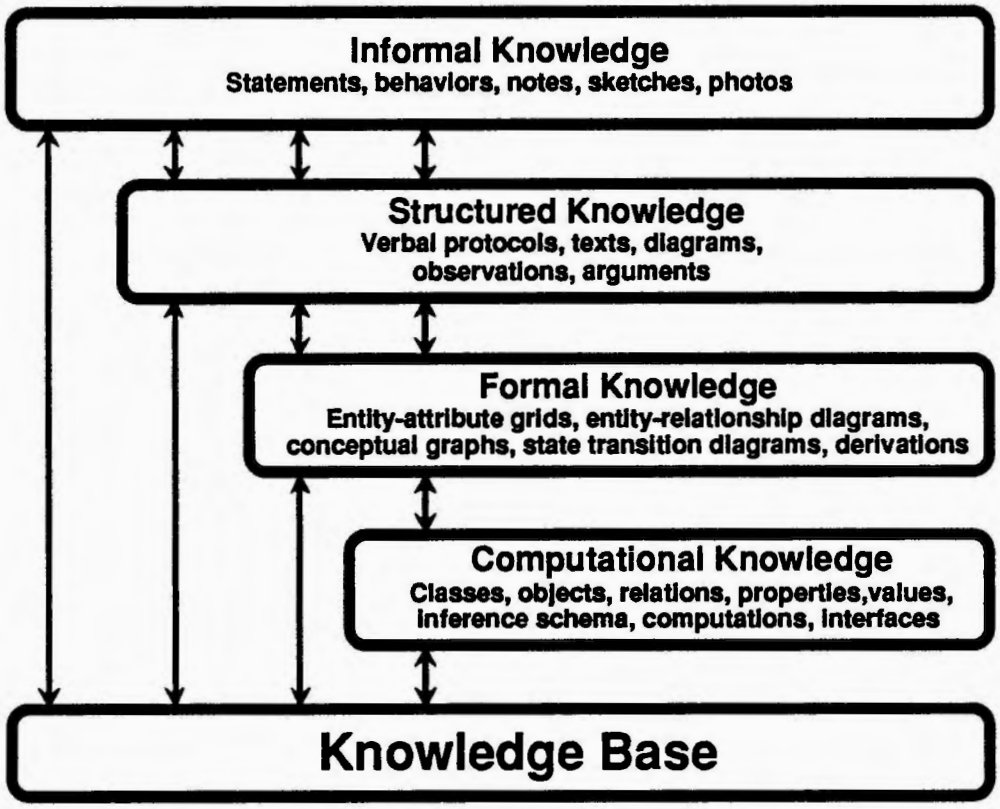

Fig.2 Levels of representation in a knowledge base

In computational terms, it is common to call only the final stage, the computational knowledge, the 'knowledge base.' However, since much of what is captured in the earlier stages is subject to computer storage, if not semantic processing, we see the computer's knowledge base as essentially comprising all these levels of representation. The users' knowledge bases will comprise part of this together with additional material not subject to any form of computer encoding. Even this unencoded material may have computational implications, however, since it may be appropriate to include material about the users' state of knowledge within the knowledge base.

In our system design we have brought together four different existing application packages and integrated them through shared file structures and inter-application communication protocols:

- *HyperCard* (Goodman 1987) is used as a hypermedia system capable of encoding, or controlling access to, some of the informal knowledge and much of the structured knowledge

- *NEXTRA*, an implementation of KSS0 (Gaines 1987b, Gaines & Shaw 1987), is used as a knowledge acquisition system taking the structured knowledge to formal knowledge and encoding it as computational knowledge
- *Oracle* is used as a database holding and processing some parts of the computational knowledge
- *NEXPERT* (Rappaport 1987a,b) is used as an inference engine holding and processing those parts of the computational knowledge most closely associated with expert systems, notably frame structures, inheritance relations and methods.

Figure 3 shows the architecture of the knowledge support system integrating these four application packages. At the bottom of this diagram is the community of users operating in the material, mental and logical worlds (Gaines 1989a). They communicate with the system through an extended workstation. Note that the direct user interface is to only two of the packages. The others are accessed through HyperCard which makes it simpler to provide a uniform and consistent interface. The packages themselves communicate through inter-application communication protocols driving well-defined script languages controlling their functionality. At the top of the diagram, the knowledge base is distributed across the file structures of the four application packages.

This architecture is highly generic and extendable through networked heterogeneous integration (Gaines 1988b, Sharp & Gaines 1989). For example, the 'internal communications' layer could be through local or remote networks, the servers could be on different platforms, and other application servers could be incorporated, such as a simulation package.

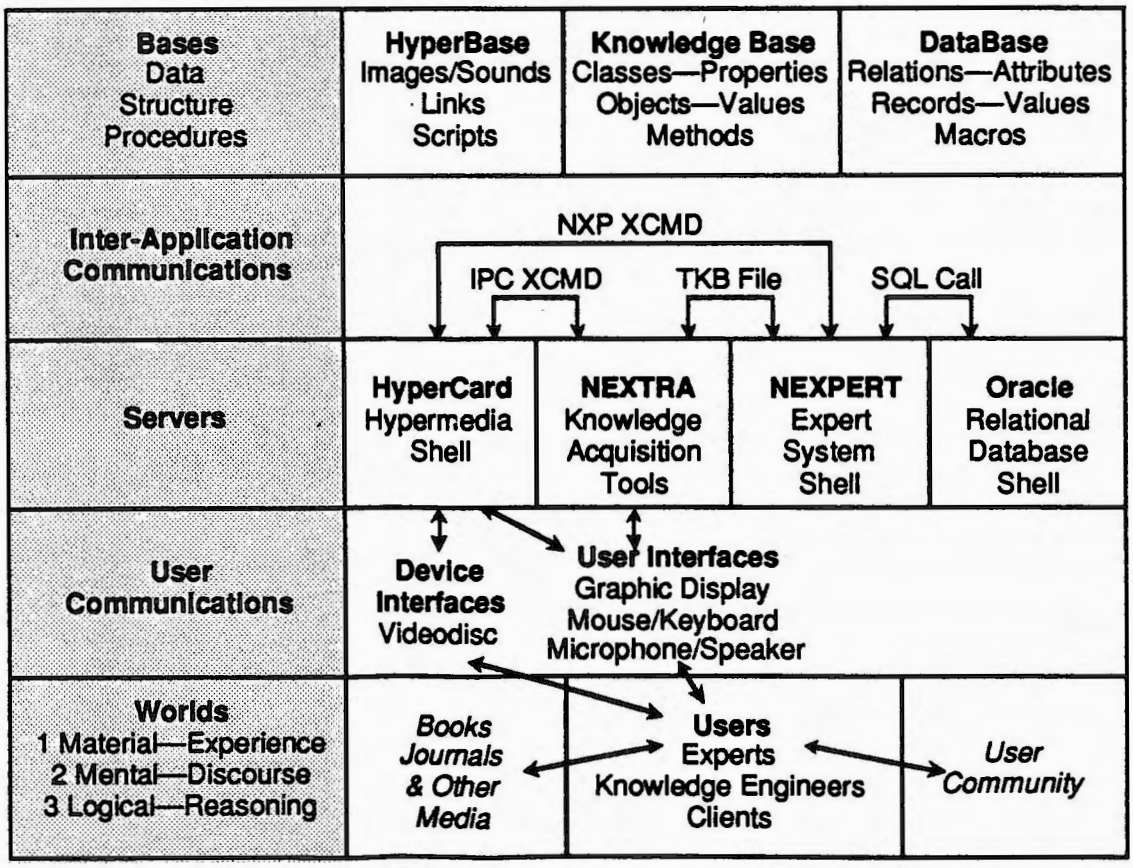

Fig.3 Architecture of the knowledge support system

The primary technical problem in implementing the system of Figure 3 is the maintenance of consistency within the distributed knowledge base including views of it presented to the user, such as edit windows and conceptual clusters. This problem is exacerbated by a design requirement for extremely non-modal operation so that a user may have several editing activities underway and freely move between them. Figure 4 illustrates some of the complexities of the problem as we have previously discussed it (Rappaport & Gaines 1988): if representation B is derived from A then it is clear that editing A may result in the need to propagate changes to B; however, it is less clear how to cope with changes to B that may, or may not, be consistent with the original source A.

The approach we have adopted to this solution currently is to keep track of the state of all data structures and their relations and tag them with version numbers so that the information, for example, that version b of B derived by process X from version a of A is continuously available. Since even the names of structures can be edited, unique accession numbers are generated as structures are created and used for inter-application linkages. Updates are done on a lazy basis and are activity dependent. For example, the knowledge base created by the acquisition tool for use by the expert system shell is automatically updated if necessary when a consultation is initiated with the shell, but not if acquisition is undertaken, as it may be, during the process of consultation.

The most difficult design decisions relating to such updating are those concerned with data in windows for user interaction and visualization. We do not know where the user is looking unless we force an action in a window, so that lazy updating is inadequate. However, continuous updating may be both time-consuming and hinder user interaction, for example if the user is actually attempting to determine the effect of a change in a case upon the results of conceptual clustering or constraint induction. We are experimenting with various forms of user interaction to determine what functionality makes the system most effectively usable.

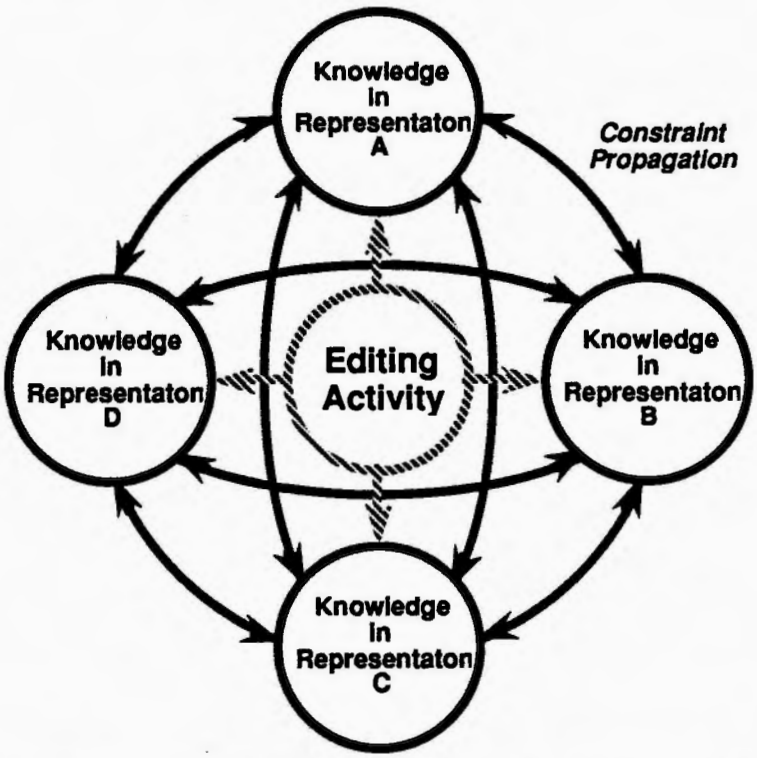

Fig.4 Mutual constraints between knowledge structures in editing

KNOWLEDGE SUPPORT SYSTEM IN ACTION

This section presents in detail the operation of the knowledge support system whose architecture is shown in Figure 3. The starting point for a user is an initial hypertext stack with two backgrounds and associated scripts, three cards, and a stack script. The user can add additional backgrounds and cards for annotation and other purposes. Figure 5 shows this basic stack structure.

The "control" background and cards are used to initiate major activities in the expert system shell such as loading knowledge bases and consulting them, and in the knowledge acquisition system such as conceptual clustering, rule induction and knowledge base creation. Two cards are provided, one blank and subject to user embellishment to give the stack and identity, and the other containing a scrolling log field through which all activities may be monitored. Either of these, or additional user-defined cards, may be used for primary control.

The "data" background and cards are used to provide basic annotation of attributes and entities in the dataset associated with the stack. One card is provided that asks the user whether a new card should be created, and the other cards are created dynamically as new entities and attributes are added. Each card contains two scrolling fields: one set up automatically with the possible values of attributes for attribute cards, and with the actual values of attributes for entity cards; and the other made available to the user for annotation. The background also provides buttons for editing data, adding an entity to the knowledge acquisition dataset, and passing control back to the knowledge acquisition tool. Space is left for user embellishment with graphics and buttons linking to extended annotation.

<table>
<tr><th></th><th>Control</th><th>Data</th><th>Extensions</th></tr>
<tr><td>Backgrounds</td><td>System-defined background for primary control</td><td>System-defined background for primary data</td><td>User-defined backgrounds for annotation structures</td></tr>
<tr><td rowspan="2">Cards</td><td>System-defined initial card subject to user-embellishment</td><td>System-defined "Add a new item?" card</td><td rowspan="2">User-defined cards for annotation</td></tr>
<tr><td>System-defined log card</td><td>System-generated cards for entities & attributes subject to user-embellishment</td></tr>
<tr><td>Background Scripts</td><td>Load knowledge base
Start consultation</td><td>Answer NEXPERT
Edit cases
Post cases to NEXTRA</td><td>User-defined scripts</td></tr>
<tr><td>Stack Script</td><td colspan="3">Inter-program protocols</td></tr>
<tr><td rowspan="2">Code Segments</td><td rowspan="2">Hypertalk interpreter</td><td>NXP XCMD</td><td>IPC XCMD</td></tr>
<tr><td>NEXPERT dynamic library</td><td>Inter-program communication</td></tr>
</table>

Fig.5 Stack structure

The "extensions" background and cards are freely created by the user for purposes such as additional annotation of entities and attributes. Since the complete power of the script languages of all four application packages is available in these extensions, it is possible to build major knowledge acquisition tools within the stack, or called by inter-program communication from the stack. In particular, a knowledge engineer can create as domain-specific an environment as is appropriate.

Figure 6 shows the initial control card of a stack developed as part of knowledge engineering for a domain concerning national science and technology policies. The user has added a world map, a button linking to an additional control card showing world demographics, and buttons linking to the cards created for nations which are the chosen entities. These links are particularly simple because the card creation script names the cards by the entity names, and the stack script contains a generic button script that goes to the card named on the button if the button itself has no script.

The other buttons on this card are provided by the system to control: the creation of a knowledge base from an entity-attribute dataset, its loading and consultation through the expert system shell; and the operation of the knowledge acquisition tool in processing the dataset. Control may be transferred explicitly to the user interface of the acquisition tool, and control is then transferred back through context-sensitive pop-up menus in that tool. The user interfaces of the expert system and database tools are through the hypertext tool so that such explicit transfer of control does not occur.

The inter-program communication protocol allows the acquisition tool also to be used as a library of callable routines rather a separate application with its own user interface. However, HyperCard currently does not provide the variable size colored windows necessary to the full functionality of the acquisition application.

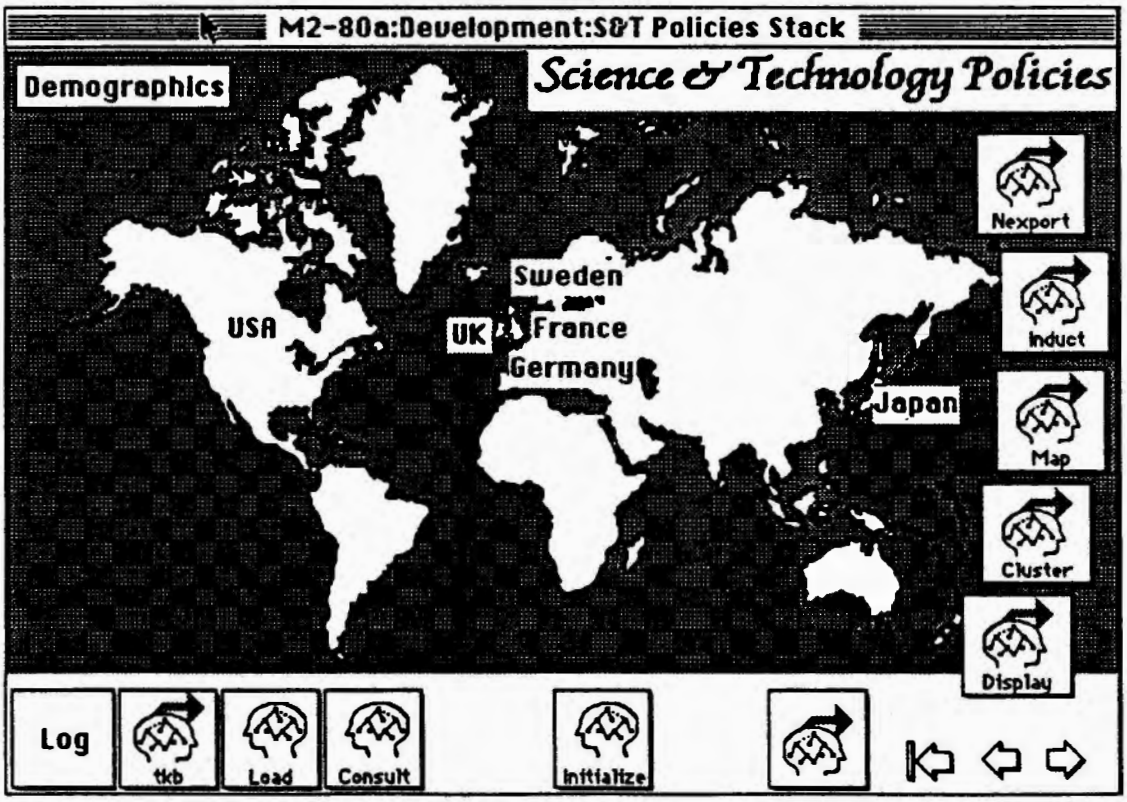

Fig.6 Initial control card of "Science and technology policies" stack

Figure 7 shows the log control card which keeps track of the major activities that have occurred. The knowledge engineer may use this card to keep track of the full complexity of a knowledge acquisition session. The normal question mark indicates user replies to query from the expert system shell during a consultation, and the inverted question mark indicates conclusions drawn. The final action recorded in this log is the adding of the entity resulting from an edited consultation to the dataset used by the acquisition tool.

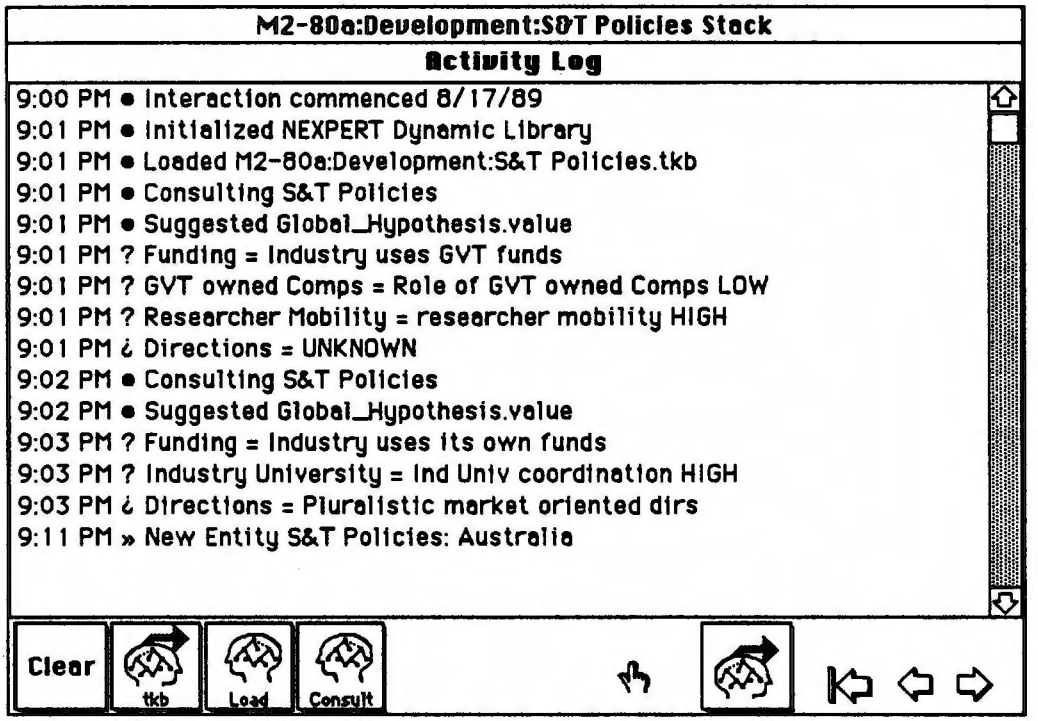

Fig.7 Log control card of "Science and technology policies" stack

Figure 8 shows a data card for the entity "United States." It is created automatically when the stack is accessed from the knowledge acquisition tool with the first request for annotation associated with this entity. The top three fields giving the domain name, entity name and entity values on attributes are created from the knowledge acquisition dataset. The annotation in the scrolling field on the lower left has been added by the user, as has the map on the right.

Figure 9 shows a set of windows forming the user environment at some stage during the acquisition process. For the user only one application is apparent. The two applications keep track of which window is in front and update their data structures appropriately. The user can bring any window to the front, possibly changing the underlying application, without being aware of the interlocks being activated to prevent inconsistencies.

Figure 10 shows how the data card of Figure 8 may be accessed from an interview window used in knowledge elicitation through the knowledge acquisition tool. As the user mouses across the names of structures in the dataset such as entities and attributes, they highlight to show they are selectable and at their extreme left side the cursor changes to a menu shape to indicate that a pop-up menu is available. If the use presses the mouse button the pop-up menu appears as shown in Figure 10, offering a range of actions, one of which is transfer to HyperCard. This option only appears in the menu if HyperCard is running and, if selected, transfers control to the hypertext tool passing information about the object selected, the window and the dataset. This is sufficient for the inter-program communication protocol script to find and transfer control to the appropriate stack and card.

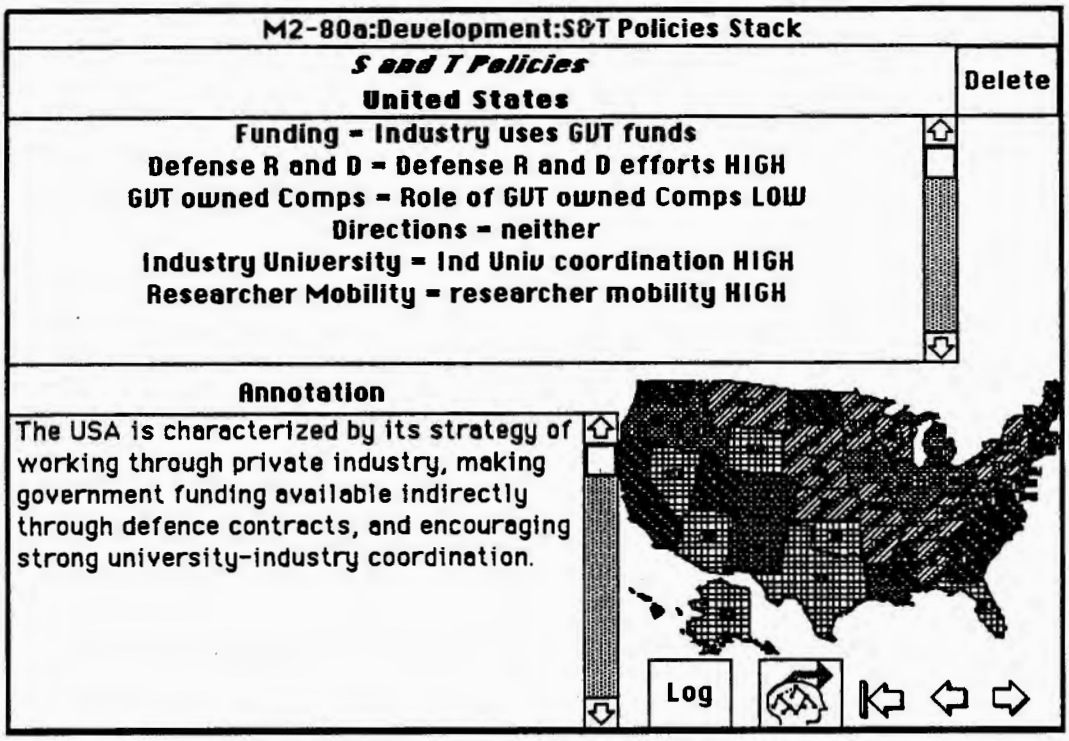

Fig.8 Data card for the entity "United States"

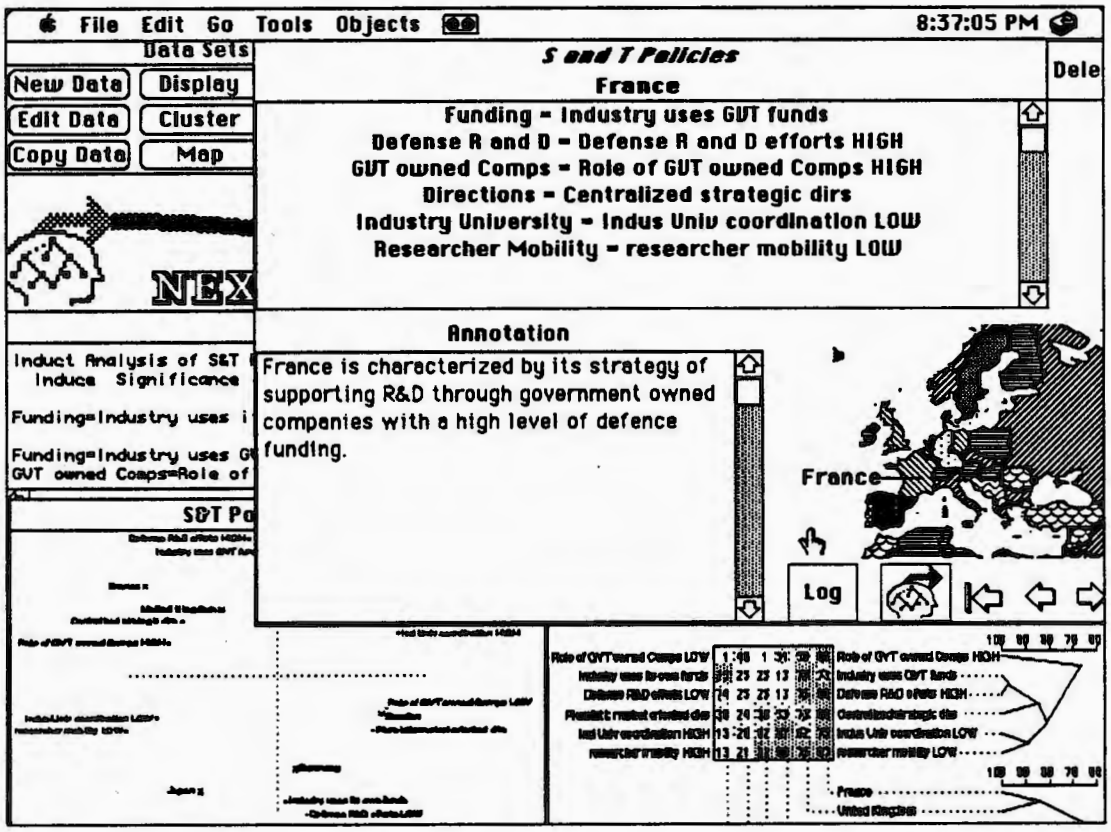

Fig.9 User environment of hypertext and knowledge acquisition windows

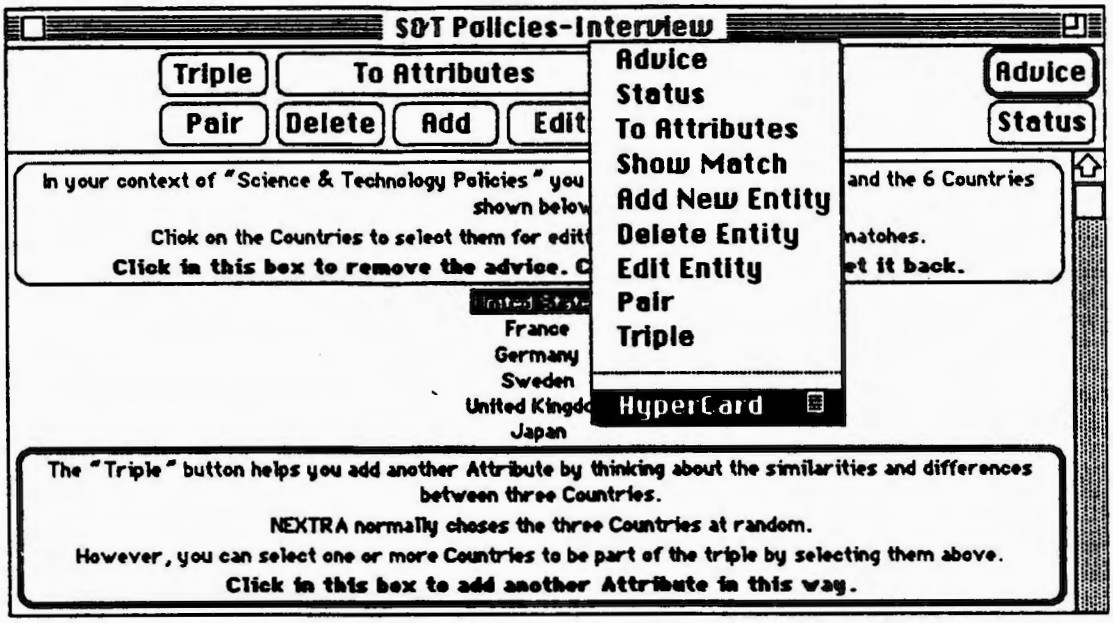

Fig.10 Context-sensitive pop-up menu in the knowledge acquisition tool

The pop-up menus are consistently available in all contexts. Figure 11 shows one being activated by a user browsing a conceptual map of the dataset created by the knowledge acquisition tool. Thus access to the annotation facilities in the hypertext tool is readily available through simple interaction in the knowledge acquisition tool. The expert from whom formal knowledge is being elicited can also enter informal and structured knowledge and link it automatically to the formal knowledge. This can include links to other media accessible through the hypertext tool such as videodiscs and sounds, including voice annotation.

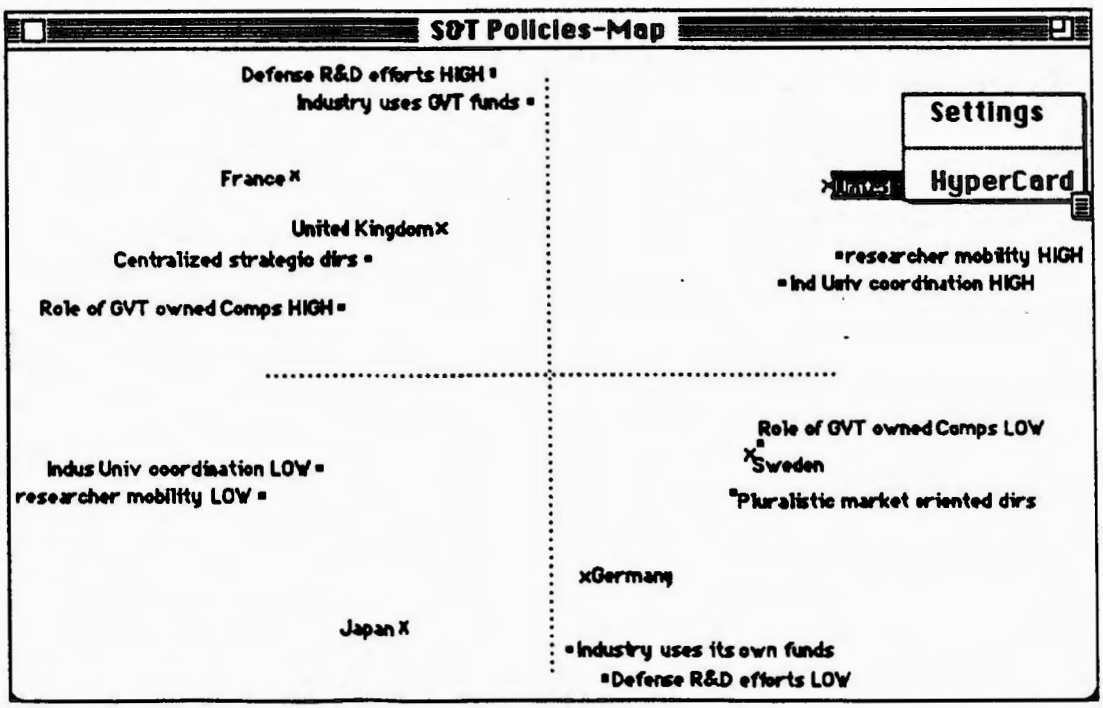

Fig.11 Context-sensitive pop-up menu in conceptual map

The data cards created in the hypertext tool are not only used for annotation. They are also the primary user interface to the expert system shell. The knowledge acquisition tool exports to the expert system shell a knowledge base in which classes, properties, objects, values and methods have been created that correspond to the elicited dataset and the entailments derived from it. This may be consulted through the expert system shell by clicking on the "Consult" icon on the lower left of Figure 6. This leads to the shell checking inference rules in the knowledge base and initiating queries to the user about the values of attributes tested in those rules.

The inter-program communication protocol passes the queries to a script that obtains answers from the user using the appropriate attribute data card as a user interface as shown in Figure 12. The top scrolling field is filled in automatically from the dataset as the set of possible values of the attribute, including the possibility of it not being known. The lower scrolling field is annotation added by the expert at some time, usually when the attribute is elicited. The space on the lower right is left for additional graphics, links, videodisc control, and so on, which may be used for tutorial and explanatory purposes when a question about the attribute is asked.

The upper two fields of the attribute data card are the domain and attribute name. In Figure 12, when the card is being used for a query, these are overlaid with the query. The user answers it by clicking on the appropriate value in the scrolling field below.

When the expert system shell reaches a conclusion it creates a new entity card showing the results of the consultation as shown in Figure 13. The values in the top scrolling field show both the answers to queries and the conclusions reached. The latter are also placed in the annotation field.

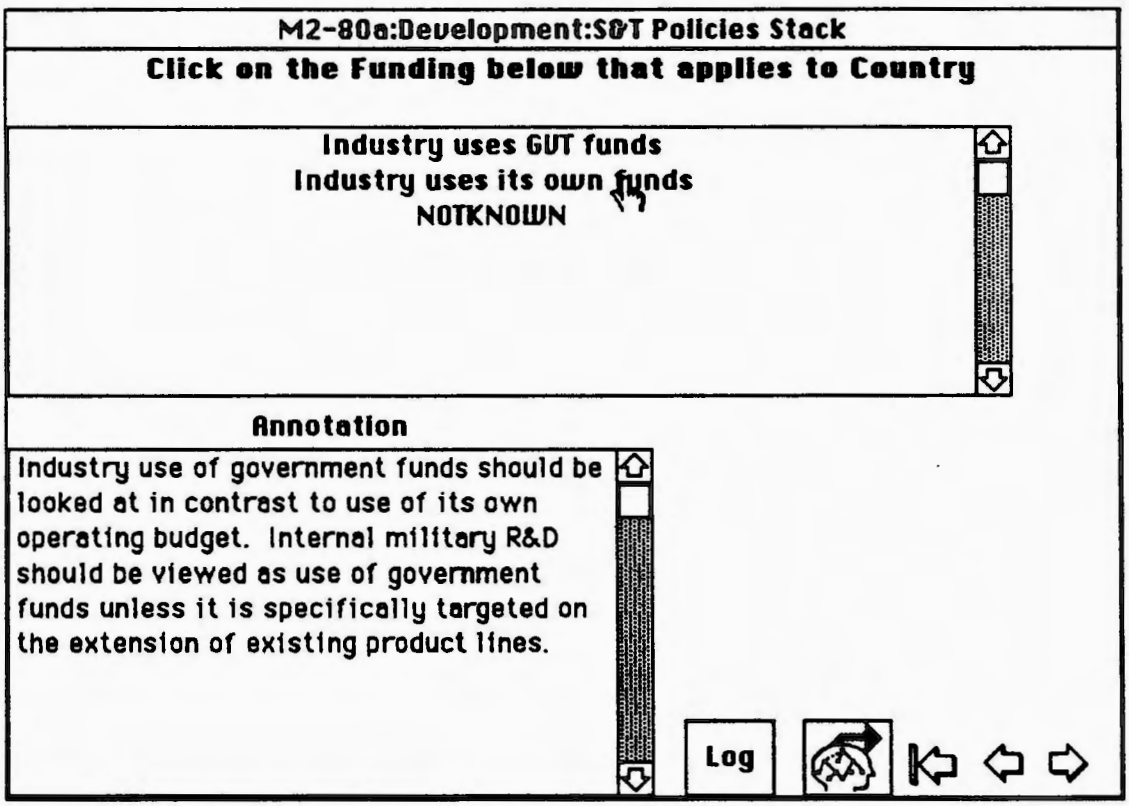

Fig.12 User query through data card in a consultation

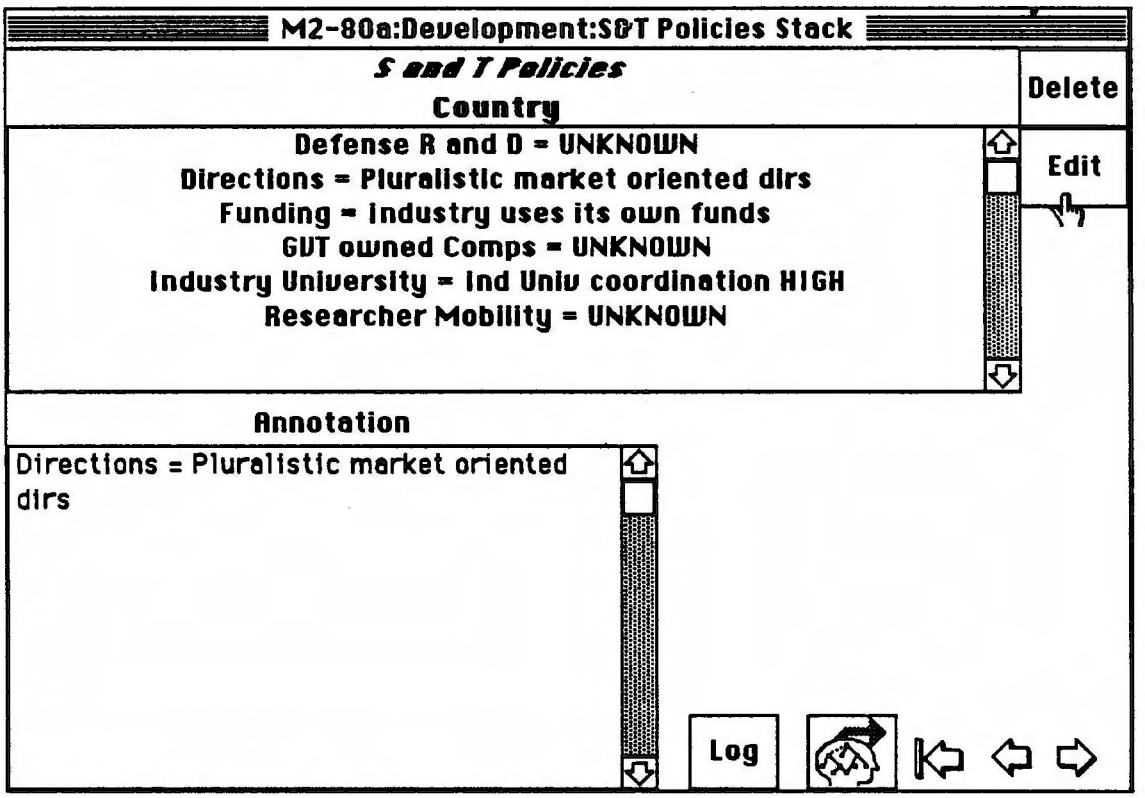

Fig.13 Results of a consultation

One outcome of the consultation, particularly during knowledge acquisition, is that the user is not satisfied with the results and wishes to change them. Clicking on the "Edit" button at the upper right of the card puts the system into an edit mode in which the user can name the test case, fill in values of attributes not requested by the shell during the consultation, and change the conclusion appropriately.

Editing the value of an attribute is carried out by clicking on it in Figure 13 which results in transfer to a query card again using the appropriate attribute data card as shown in Figure 14. Clicking on the required value in Figure 14 transfers control back to the card of Figure 13 with the new value entered.

When the editing is complete a new entity has been created as shown in Figure 15. Clicking on the "Add" icon at the upper right posts this entity to the knowledge acquisition tool where it is added to the dataset. Running entailment induction may result in changed rules in the new knowledge base and hence changed behavior in a consultation with the expert system shell. The effect of the new entity on the conceptual structure may be investigated immediately within the knowledge acquisition tool—for example, Figure 16 shows the resulting conceptual cluster including the new entity.

The non-modal nature of the user interface means that such edits to the dataset have to be done in such a way that no problems arise if the dataset is already open for editing in the knowledge acquisition tool. In particular, edit windows must be automatically updated to show the new data. The inter-program communication protocol will also operate across a network so that the edit windows could be in active use when they are updated. It may be appropriate to ask the other user whether the new entity should be added—the interaction of technical and social protocols for simultaneous editing of the same dataset needs substantial further investigation (Gaines 1986).

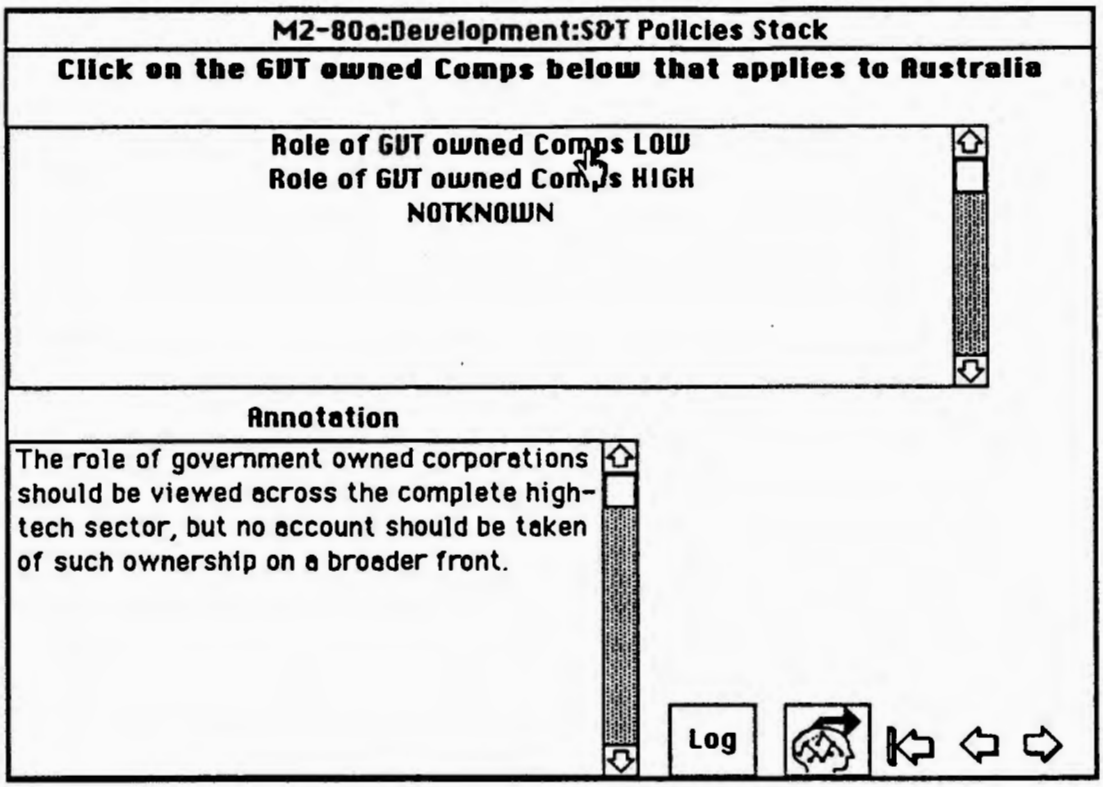

Fig.14 Editing results of a consultation

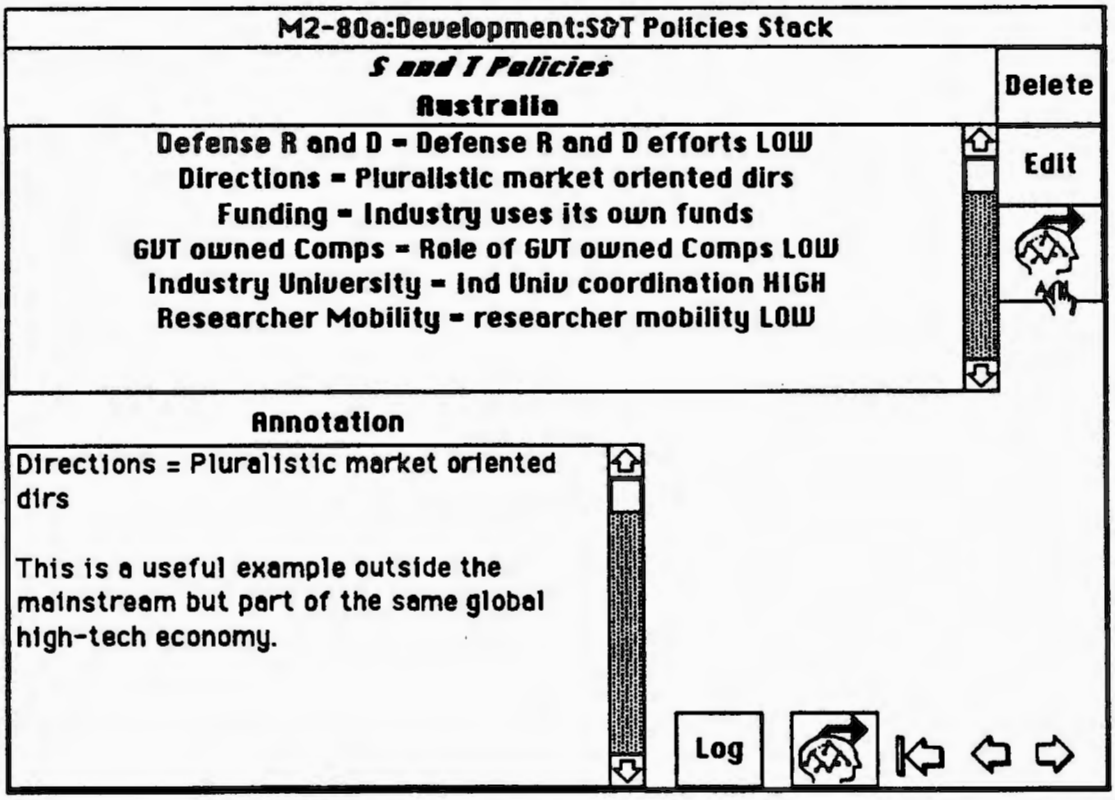

Fig.15 Adding edited results as new entity in the dataset

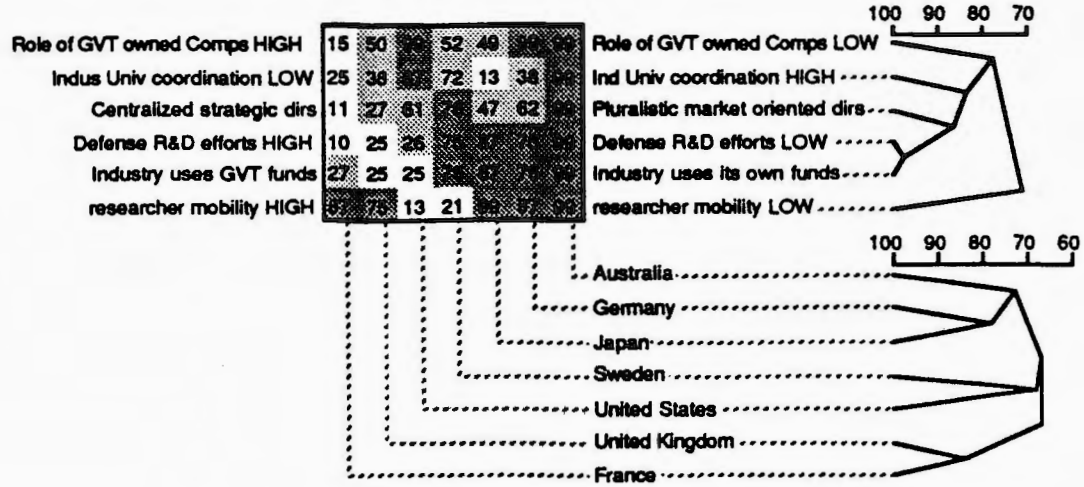

Fig.16 Concept cluster showing new entity added

OVERALL APPLICATION OF THE KNOWLEDGE SUPPORT SYSTEM

Figure 17 shows the modes of knowledge acquisition available in the knowledge support system we have described. We have analyzed previously the many psychological problems that impede knowledge elicitation (Gaines 1987a, Gaines & Boose 1988) and emphasized the need for a wide variety of approaches and technologies to be made available. The system described provides many qualitatively distinct alternatives for elicitation.

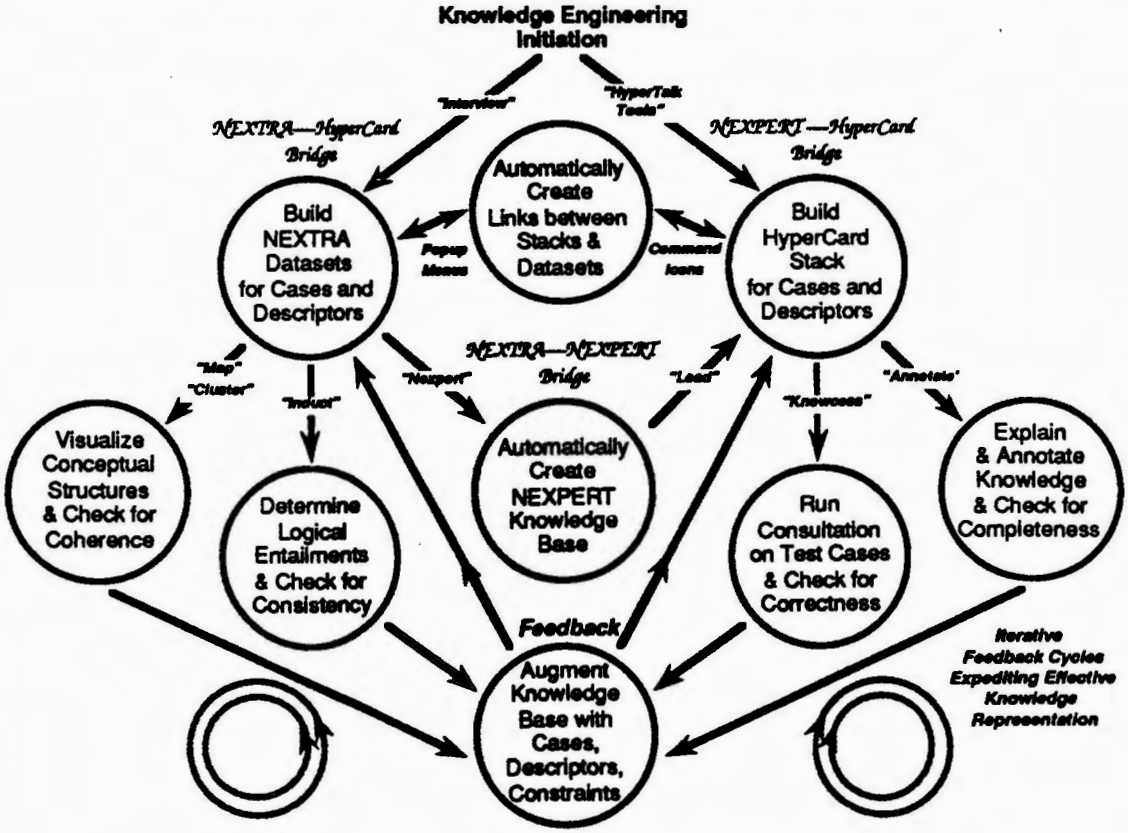

Fig.17 Modes of knowledge acquisition available

The knowledge engineering activity can be initiated either within the knowledge acquisition tool or within the hypertext tool. The expert can commence with structured knowledge elicitation and annotate it, or commence with an informal notebook and structure it. The system will automatically link the stacks and datasets, no matter how generated, and the expert can add further linkages to enrich the knowledge base. The system also automatically makes the computational part of the knowledge base mutually available to the acquisition tool and expert system shell so that consultations are available at any time.

It is interesting to analyze the multiple paths for knowledge elicitation in terms of their impact on the validation of the knowledge elicited:

- The visualization analyses in the acquisition tool allow the conceptual structures elicited from the expert to be examined and checked for *coherence*, that the internal relations between cases, between descriptors, and between cases and descriptors make sense to the expert.
- The logical entailment analyses in the acquisition tool allow the constraints implied in the conceptual structures elicited from the expert to be examined and checked for *consistency*, that they can not lead to contradictory conclusions.
- The consultation facilities in the expert system shell allow the knowledge base derived from the conceptual structures and constraints to be checked for *correctness*, that conclusions acceptable to the expert are generated with test cases.
- The annotation facilities in the hypertext tool allow the computational knowledge base derived from the conceptual structures and constraints to be checked for *completeness*, that what has been made computational adequately encompasses the critical material described less formally.

The terminology above is reminiscent of notions of truth in the philosophy of science and philosophical logic, and the connotations in terms of validation procedures are similar. However, the knowledge elicitation process can not be reduced to formal considerations of truth. Even expertise in a formal domain, such as mathematics, involves relevant knowledge outside the formal domain, such as worthwhile problems, fruitful approaches to unsolved problems, problem-solving techniques, and so on. The notions of truth in formal systems are, however, the end-product of refining validation procedures for proto-truth as shown in Figure 17.

There are also elicitation and validation processes available within the system that are not represented in Figure 17. The inter-expert comparison facilities in the acquisition tool (Shaw & Gaines 1988) allow the grounding of the experts' terminology and its relation to distinctions made to be analyzed. The database tool allows existing sources of information to be used in building the knowledge base, through expert selection and empirical induction rather than expertise transfer (Gaines 1989b). The plurality of sources and processes leading to the creation of an adequate knowledge base is important in allowing some of the variety of knowledge-based systems to be encompassed, and some of the blockages in knowledge elicitation to be bypassed. It increases the managerial load on the knowledge engineer, to make effective use of appropriate tools.

The availability of many modes of elicitation also makes it more difficult to design a comprehensible user interface. We have avoided some problems currently by treating the expert system shell and database tools as libraries callable from scripts in the hypertext tool. In so doing, we have lost access to some significant facilities in the application programming support environments of these tools. Currently, these may be accessed separately but not as part of the integrated system.

EXTENSIONS

The knowledge support system described in this paper, even though it has very powerful functionality, is also completely open in its architecture. The functionality shown is controlled through scripts in the hypertext tool that access the other tools as callable libraries, each having its own script language. The control scripts and the library scripts are easily modified and extended. In particular, new knowledge elicitation and application modules can be added, and new links may be made to other application tools. We see this as the essence of modern information system design, to provide base technology in modular form with an open architecture already delivering powerful functionality, yet extendable to be more general or more specific according to user community requirements.

For example, a knowledge engineer could build a stack in the hypertext tool that was highly targeted to a specific domain area, incorporating structures and vocabulary that were already standard for experts in that domain. Domain specific knowledge acquisition tools have been at the heart of some of the most successful applications of knowledge-based systems (Boose & Gaines 1988, Musen 1989), but it is a major task to create an effective one for a new domain. Customizing existing generic tools to be as domain-specific as required offers a highly productive approach to specialized knowledge acquisition.

The extensions need not be domain-specific but could add elicitation and analysis functionality to the tool at different levels from those already covered, or using different knowledge acquisition paradigms. For example, Woodward's (1988) operationalization of linguistic elicitation of 'implicit knowledge structures' is implemented in HyperCard as his Cognosys tool, and could be incorporated as another starting point for elicitation standing between the unstructured hypertext tool and structured entity-attribute acquisition tool.

The extensions could also be used for integrating different knowledge islands within a domain. Entity-attribute grid tools (Shaw 1980, Gaines & Shaw 1980, Shaw & Gaines 1980, 1987, Boose 1984, 1986, Boose & Bradshaw 1987) may be seen as being most effective at the class or frame level. They allow a single, coherent knowledge island to be thoroughly explored. This may be all that is required in simple classification and selection tasks. However, in more complex domains, knowledge structures for a number of knowledge islands have to be developed and then integrated through appropriate import-export data linkages and control structures to operate as an effective overall system. The acquisition tool can already cope with multiple knowledge islands as separate entities, and we are developing additional facilities within it to cope with the integration. However, the open architecture described in this paper makes it feasible to program the integration facilities within the hypertext tool, and this may be the appropriate place for them since it is not yet clear that they can be provided in a form that is both usable and domain-independent.

One may continue this line of thinking further and contemplate future system designs that are even more modular than that described in this paper. The acquisition tool is written as a loose control framework calling separate modules for graphic elicitation, visualization, expert comparison, induction, and knowledge base creation. These modules could be separated as distinct libraries and the control framework moved into the hypertext tool. Currently, there are weaknesses in existing hypertext tools, notably lack of variable-size color windows, that make this unattractive, but it may become the most effective architecture as these tools develop.

HyperCard shows the potential for system design with an object-oriented development tool supporting a general-purpose interpretive language, with recourse to foreign function calls in other languages when speed is required, and having a powerful user interface. There are weaknesses in the consistency of the design, the language and the interface, but these can be overcome, and are already outweighed by the system design possibilities.

The next generation of such tools will make major changes to system design practice, particularly for highly interactive, intrinsically heterogeneous systems, such as knowledge support systems. Future tools may be based on the evolution of HyperCard-like tools, or on the combination of more principled systems such as the Smalltalk object-oriented programming language coupled with the X-Windows/Motif networked user interface and inter-application protocols. However the technology is provided, it offers opportunities for system designs and implementations that were virtually impossible a few years ago, yet which are fundamental to our aspirations for knowledge-based systems.

We have to be very careful that the technological limitations of the past have not blinkered us to the opportunities now becoming available. The next generation of knowledge support systems will be limited more by our creative imaginations, than by our available technologies.

CONCLUSIONS

This paper describes the architecture and operation of a knowledge support system that combines hypermedia, knowledge acquisition, expert system shell and database systems to provide an integrated end-user environment. The integration is achieved through inter-application communication protocols operating between four programs designed independently for stand-alone use. The system supports a wide variety of forms and representations of knowledge from the highly informal to the highly computational. It is extremely non-modal and allows knowledge acquisition and knowledge consultation to be combined in a variety of ways. In particular, the results of a consultation may be used to create a new case that may be added to the case base of the acquisition tool which may then be re-analyzed, possibly resulting in changed advice, and supporting the essentially iterative process of knowledge-based system development shown in Figure 1.

We have emphasized the knowledge acquisition phase of application of the system in this paper, yet the integration of performance and acquisition tools blurs the distinction between development and application phases. The possibility of acquisition through performance allows knowledge engineering to be treated as an ongoing process with new case data being added from field experience in the application of the system. This is a step towards what Steels (1986) and Gaines (1988a) have termed "second generation" knowledge-based systems able to learn through their performance. Our current implementation requires extension to incorporate anomalous case detection to draw attention to cases where the knowledge base may be inadequate rather than clearly in error (Gaines 1988c). To fully participate in the knowledge processes shown in Figure 1, it is also important for the system to be able to report local updates to other systems through remote communications. Such extensions present no significant technical problems, but they require a different conception of the role of expert systems than is currently prevalent in user communities.

If there is one overall conclusion to be drawn from this paper, it is the importance for modern open system architectures to achieving the aspirations of the knowledge acquisition and knowledge-based systems development communities. We are now able to make available tools that are powerful and well-integrated in their own right, but where we are also able to say to user communities, "What we provide is useful, but you are not bound by it. If you want to change or extend the system, it is open to you do so." This catches the essence of knowledge processes in society, that each of us stands on the shoulders of countless others in making our own contribution, that we take the tools developed by earlier generations and other disciplines and use them in unintended ways for new purposes. Knowledge support tools have to be open, malleable and extensible if they are to have a meaningful role in the knowledge processes of society.

It is interesting to look back at our own aspirations for knowledge support systems and see if we are still on track and whether they have themselves been extended. We state that (Shaw & Gaines 1986):

1. The knowledge support system should be domain independent.
2. The knowledge support system should be directly applicable by experts without intermediaries.
3. The knowledge support system should be able to access a diversity of knowledge sources including text, interviews with experts, and observations of expert behavior.
4. The knowledge support system should be able to encompass a diversity of perspectives including partial or contradictory input from different experts.
5. The knowledge support system should be able to encompass a diversity of forms of knowledge and relationships between knowledge.
6. The knowledge support system should be able to present knowledge from a diversity of sources with clarity as to its derivation, consequences and structural relations.
7. Users of the knowledge support system should be able to apply the knowledge in a variety of familiar domains and freely experiment with its implications.
8. The knowledge support system should make provision for validation studies.
9. As much of the operation of the knowledge support system as possible should be founded on well-developed and explicit theories of knowledge acquisition, elicitation and representation.
10. As the overall knowledge support system develops it should converge to an integrated system.

The system described in this paper satisfies all these objectives, with the greatest incremental contribution being to the final objective, of convergence to an integrated system. However, our concept of what such integration means is certainly very much looser and heterogeneous than it was three years ago. We have also extended each of our objectives, for example, the first would now read, "be domain independent at their kernel but customizable to be as domain specific as is useful." Indeed this customization is now a major requirement in its own right since it impacts all the other objectives—direct applicability may not always be achievable without prior customization—the incorporation of theoretical developments may requires extensions to the tools, and so on.

The degree and quality of satisfaction of each of these objectives is clearly open to major improvement—and this will always be so—knowledge is fallible and evolving and so is our knowledge of how to acquire, process and apply knowledge. Boose (1989) has classified the current repertoire of knowledge acquisition tools, their limited strengths and major weaknesses. We do not claim that the knowledge support system described in this paper goes very far, or is significantly innovative, in overcoming those weaknesses. We do suggest that its heterogeneous architecture illustrates an enabling technology that is fundamental to overcoming those weaknesses in the time-scales, and over the breadth of applications, that are now required.

ACKNOWLEDGEMENTS

Financial assistance for this work has been made available by the Natural Sciences and Engineering Research Council of Canada. We are grateful to many colleagues for discussions over the years that have influenced the research described in this paper, in particular John Boose, Jeff Bradshaw, and other colleagues at the Knowledge Acquisition Workshops world-wide.

REFERENCES

Boose, J.H. (1984). Personal construct theory and the transfer of human expertise. **Proceedings AAAI-84**, 27-33. California: American Association for Artificial Intelligence.

Boose, J H. (1986) **Expertise Transfer for Expert System Design**. New York: Elsevier.

Boose, J H. (1989) A survey of knowledge acquisition techniques and tools. **Knowledge Acquisition 1** (1), 39-58 (March).

Boose, J.H. & Gaines, B.R., Eds. (1988). **Knowledge Acquisition Tools for Expert Systems**. London: Academic Press.

Boose, J.H. & Bradshaw, J.M. (1987) Expertise transfer and complex problems: using AQUINAS as a knowledge acquisition workbench for knowledge-based systems. **International Journal of Man-Machine Studies 26**, 3-28.

Gaines, B.R. (1986). Cognitive models of human-computer interaction in distributed systems. **Proceedings of 1986 International Conference on Systems, Man and Cybernetics.** Vol. IEEE 86CH2364-8, pp. 1358-1362. Atlanta (October).

Gaines, B.R. (1987a) An overview of knowledge acquisition and transfer. **International Journal of Man-Machine Studies 26** (4), 453-472 (April).

Gaines, B.R. (1987b). Rapid prototyping for expert systems. Oliff, M.D., Ed. **Intelligent Manufacturing: Proceedings from First International Conference on Expert Systems and the Leading Edge in Production Planning and Control.** pp.45-73. Menlo Park, California, Benjamin Cummins.

Gaines, B.R. (1988a). Second generation knowledge acquisition systems. **Proceedings of the Second European Workshop on Knowledge Acquisition for Knowledge-Based Systems (EKAW'88)**. GMD-Studien Nr.143. pp.17-1-17-14. Bonn, Germany: Gellsellschaft für Mathematik und Datenverarbeitung (June).

Gaines, B.R. (1988b). Organizational integration: modeling technology and management. Zunde, P. (Ed.) **Empirical Foundations of Information and Software Sciences II.** New York: Plenum Press.

Gaines, B.R. (1988c). Positive feedback processes underlying the formation of expertise. **IEEE Transactions on Systems, Man & Cybernetics**, SMC-18(6), 1016-1020 (November).

Gaines, B.R. (1989a) Social and cognitive processes in knowledge acquisition. **Knowledge Acquisition 1** (1), 39-58 (March).

Gaines, B.R. (1989b) An Ounce of Knowledge is Worth a Ton of Data: Quantitative Studies of the Trade-Off between Expertise and Data based on Statistically Well-Founded Empirical Induction. **Proceedings of 6th International Workshop on Machine Learning**, pp.156-159. San Mateo, California: Morgan Kaufmann (June).

Gaines, B.R. & Boose, J.H., Eds. (1988). **Knowledge Acquisition for Knowledge-Based Systems**. London: Academic Press.

Gaines, B.R. & Shaw, M.L.G. (1980). New directions in the analysis and interactive elicitation of personal construct systems. **International Journal of Man-Machine Studies 13**(1) 81-116 (July).

Gaines, B.R. & Shaw, M.L.G. (1987). Knowledge support systems. **ACM MCC-University Research Symposium.** Austin, Texas: MCC. pp.47-66.

Goodman, D. (1987). **The Complete HyperCard Handbook.** New York: Bantam.

Musen, M. (1989). **Automated Generation of Model-Based Knowledge Acquisition Tools.** London: Pitman.

Rappaport, A. (1987a) Multiple-problem subspaces in the knowledge design process. **International Journal of Man-Machine Studies 26**(4), 435-452 (April).

Rappaport, A. (1987b) Cognitive primitives. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the Second AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.15-0-15-13. Banff (October).

Rappaport, A. & Gaines, B.R. (1988). Integration of acquisition and performance. **Proceedings of the Third AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.25-1-25-20, Banff (November).

Sharp, M. & Gaines, B.R. (1989). Knowledge acquisition for user support in heterogeneous networks. **CASE'89: Proceedings of the Third International Workshop on Coomputer-Aided Software Engineering.** pp.229-239. London (July).

Shaw, M.L.G. (1980). **On Becoming a Personal Scientist.** London: Academic Press.

Shaw, M.L.G. & Gaines, B.R. (1983). A computer aid to knowledge engineering. **Proceedings of British Computer Society Conference on Expert Systems,** 263-271 (December). Cambridge.

Shaw, M.L.G. & Gaines, B.R. (1986). Advances in interactive knowledge engineering. Bramer, M.A., Ed. **Research and Development in Expert Systems III.** pp.111-122. Cambridge: University Press.

Shaw, M.L.G. & Gaines, B.R. (1987) KITTEN: Knowledge Initiation & Transfer Tools for Experts & Novices. **International Journal of Man-Machine Studies 27**(3), 251-280 (September).

Shaw, M.L.G. & Gaines, B.R. (1988). A methodology for recognizing consensus, correspondence, conflict and contrast in a knowledge acquisition system. **Proceedings of the Third AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.30-1-30-19, Banff (November).

Shaw, M.L.G. & Woodward, B. (1989). Mental models in the knowledge acquisition process. **Proceedings of the Fourth AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** Banff (October).

Steels, L. (1986). Second generation expert systems. Bramer, M.A. (Ed). **Research and Development in Expert Systems III.** pp.175-183. Cambridge: University Press.

Woodward, B. (1988) Knowledge engineering at the front-end: defining the domain. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the Third AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.37-1-37-25. Banff (November).