Data & Knowledge Engineering 9 (1992/93) 1–18
North-Holland

Combining paradigms in knowledge engineering

Brian R. Gaines[a], Alain T. Rappaport[b] and Mildred L.G. Shaw[c]
[a]*Knowledge Science Institute, University of Calgary, Calgary, Alberta, Canada T2N 1N4*
[b]*Neuron Data, 156 University Avenue, Palo Alto, CA94301, USA*
[c]*Knowledge Science Institute, University of Calgary, Calgary, Alberta, Canada T2N 1N4*

Abstract

Gaines, B.R., A.T. Rappaport, and M.L.G. Shaw, Combining paradigms in knowledge engineering, Data & Knowledge Engineering 9 (1992/93) 1–18.

A system supporting knowledge engineering 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.

Keywords. Knowledge engineering; expert systems; knowledge acquisition; hypermedia.

1. Introduction

Ever since Feigenbaum [11] identified the knowledge engineering bottleneck impeding the effective application of knowledge-based systems, knowledge acquisition has been a topic attracting major research effort and resulting in the development of a wide range of methodologies, techniques and tools [3–6, 12, 20, 27]. In terms of the direct knowledge engineering process we now have a rich repertoire of approaches, support systems and experience on which to base further research and practical system development. The tools may be divided into four major categories: *graphic knowledge editors* for overt knowledge represented in semantic networks allowing experts and knowledge engineers to program expert system shells in a visual language; *repertory grid tools* for the development of knowledge structures through the elicitation of critical cases from experts; *inductive modeling tools* for the development of knowledge structures from large databases of cases with irrelevant attributes and erroneous data; *hypermedia-based tools* for informal data collection such as interviews and protocols, and their subsequent structuring. The classes of tools are listed in order of declining access to overt knowledge: direct knowledge entry is prefered if the knowledge is overt and available; repertory grids allow knowledge to be modeled rapidly from small sets of critical cases provided by an expert; inductive modeling allows noise and irrelevant attributes to be filtered from databases when expert selection of critical cases is not available; hypermedia tools allow the diverse data gathered through interviewing and

Correspondence to: B.R. Gaines, Knowledge Science Institute, University of Calgary, Calgary, Alberta, Canada T2N 1N4.

0169-023X/92/$05.00 © 1992 – Elsevier Science Publishers B.V. All rights reserved

protocol analysis to be managed and organized when overt knowledge is not available through computational analysis.

Graphic editors providing direct access to semantic network representations allowing knowledge to be encoded in frames and rules provide the most common development environment for knowledge-based systems. They are part of the application programming support environment of most expert system shells, and the widespread availability of modern graphic workstations has made it possible to provide excellent knowledge visualization environments. A wide range of knowledge acquisition tools have been developed that structure and improve the graphic editing environment, often taking advantage of domain knowledge to provide a more specific, meaningful and familiar knowledge framework to the expert. Examples are MOLE [10], KNACK [25], SALT [26] and KEATS [29].

Repertory grids are a technique for knowledge elicitation when experts cannot directly enter a knowledge structure. They stand between direct editing and empirical induction, and prompt the expert for distinctions relevant to the problem domain and for critical cases that exhibit significant phenomena in the domain. The prompting is done through online analysis of the data being entered leading to feedback to the expert suggesting missing distinctions and cases. This highly focused feedback aids the expert in developing his or her mental model of the domain. It also reduces the inefficiencies of duplication and the mental blocks of psychological set, supporting rapid prototyping. A wide range of knowledge acquisition tools have been developed that incorporate repertory grid elicitation and analysis as their major interface to the expert. Examples are PLANET [39], ETS [1], AQUINAS [4], KRITON [9], KITTEN [41], and KSS0 [22].

When experts cannot directly enter a knowledge structure or critical cases, they may still be able to point the knowledge engineer towards case histories that incorporate that expertise in being described in terms of relevant attributes and correct decisions. Empirical induction techniques may then be used to derive knowledge structures underlying the decisions made in these cases [28]. The best known empirical induction methodology is that of Quinlan's ID3 which has been refined in many ways, particularly to tolerate noise (incorrect decisions), resulting in the latest implementation, C4.5 [31]. The original decision tree structure of ID3 based on a subsumption hierarchy of concepts with rules at the leaf nodes only is unnecessarily large in many situations, and extensions to ID3 have been developed that generate modular production rules directly, such as Cendrowska's Prism [7]. This also can be extended to deal with noisy data as in Induct [18].

The three techniques so far described focus on the elicitation, derivation and structuring of computational knowledge, i.e. knowledge that can be operationalized within the computer as an active emulation of expertise. However, much knowledge is informal yet still valuable in an advisory expert system – text and pictures can encode and impart usable expertise, supplementing computational knowledge. Thus, the parallel development of hypertext and hypermedia is having a substantial impact on expert system architectures and knowledge acquisition tools. Hypermedia tools such as Cognosys [44], CAMEO [24], and CARTES [32], designed specifically for knowledge acquisition also provide means for developing knowledge structures arising out of, and linked to, the informal data.

What has been missing in research to date is the development of integrative architectures combining these different techniques. This is a natural result of the need to focus on high-quality development of specific classes of tools. There are many problems involved in integration. For example, at one level it is not trivial to transfer the knowledge structures developed in a general knowledge acquisition tool into the representation schema of a specific expert system shell. At another level, it is not easy to provide a uniform user interface to diverse knowledge acquisition tools developed by different groups for different purposes, but neither is it realistic for one research group to develop an integrated tool that

is of top quality in all the different approaches. One major problem with lack of integration, however, is that certain significant knowledge acquisition cycles that involve using tools in succession with feedback between stages are not being supported and investigated; for example, the feedback to the knowledge acquisition tools of cases in which errors occur when an expert system is in operation.

This paper presents an experiment in the integration of repertory grid, induction and hypermedia knowledge acquisition tools with database and expert system shell tools. The objective is to provide a system supporting knowledge engineering, and 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.

2. Design principles

Figure 1 summarizes the design objectives for a comprehensive knowledge support system in terms of the types of knowledge representation and tools already discussed. The starting point is a 'knowledge base' 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

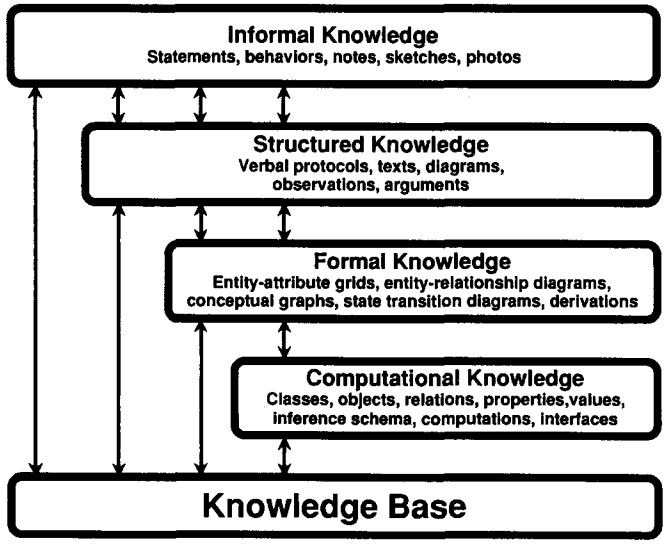


Fig. 1. Levels of representation in a knowledge base.

ontologies. Part of this formalization may be made operational by encoding it in computational form. 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.

As already noted, this study was undertaken as one of implementing the architecture of *Fig. 1* through heterogeneous integration of existing tools. When combining tools in an integrated system it is important to have a clear systems architecture in terms of required functionality and how it is to be allocated among the tools. It is necessary to exclude some facilities provided by some tools because it is duplicated in others or inappropriate in the overall system. If this is not done, and the combined system does not project a clear model of its intended use, then users can become confused by excessive features. A functional model for the integrated system has been built at a high level of abstraction in terms of four dimensions of logical validation of a knowledge base, corresponding to similar dimensions in evaluating the truth of scientific theories [36]:

- *Coherence* – the coherence of internal relationships between knowledge structures,
- *Consistency* – the lack of logical contradiction between knowledge structures,
- *Correctness* – the correctness of deductions from the knowledge structures checked against external data,
- *Completeness* – the adequate coverage of an intended scope for deductions from the knowledge structures.

The system implementation brings together four different existing application packages and integrated them through shared file structures and inter-application communication protocols:

- *HyperCard* [23] 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 [13, 22], is used as a knowledge acquisition system taking the structured knowledge to formal knowledge and encoding it as computational knowledge,
- *Oracle* [37] is used as a database holding and processing some parts of the computational knowledge,
- *NEXPERT* [33, 34] 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 2 shows how the functionality is divided among the packages in terms of the four dimensions of logical validation. 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.

One can now analyze the multiple paths for knowledge elicitation in terms of their impact on the form validation of the knowledge elicited:

- The visualization analyses in the acquisition tool allow the conceptual structures elicited

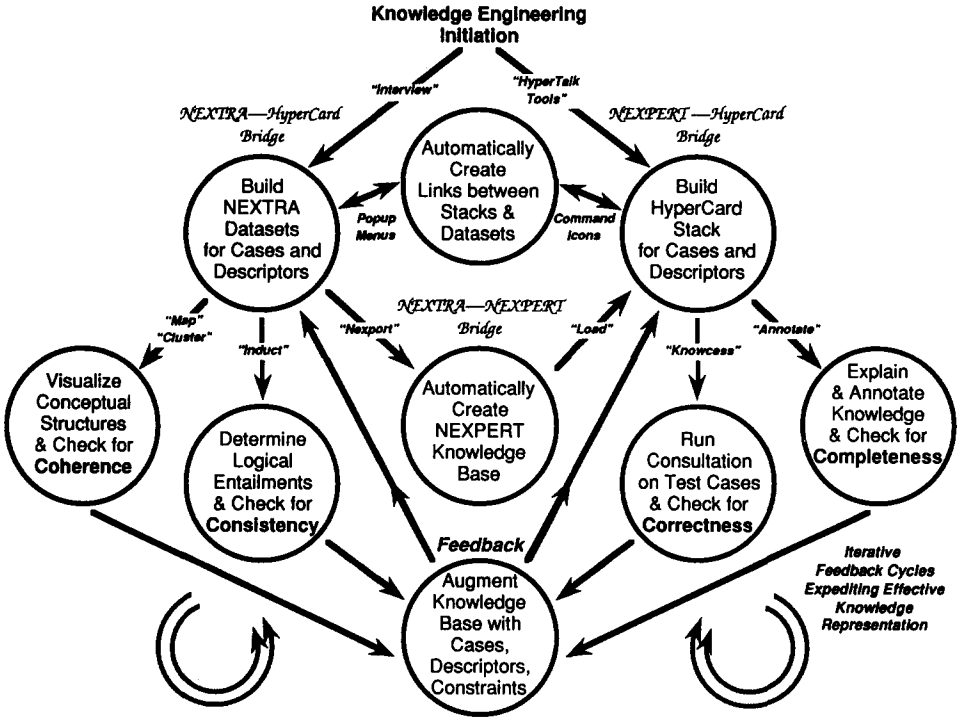


Fig. 2. Modes of knowledge acquisition available.

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 cannot 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.

This is a useful design framework within which to ascribe functionality to the different tools, and to explain the system to users. However, the knowledge elicitation process cannot be completely 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. There are also elicitation and validation processes available within the system that are not represented in *Fig. 2*. The inter-expert comparison facilities in the acquisition tool [42] 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. 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.

Figure 3 summarizes the data structures and flows of the knowledge support system integrating the four application packages. At the bottom of this diagram is the community of users operating in the material, mental and logical worlds [17]. 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 [15]. 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.

The primary technical problem in implementing the system of *Figs 1* through *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 [35]. 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. 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. Since even the names of structures can be

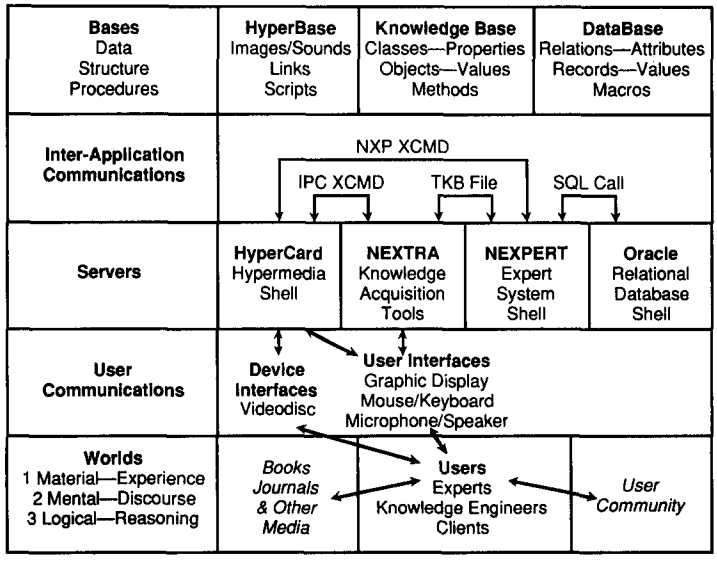


Fig. 3. Data structures and flows in the knowledge support system.

edited, unique accession numbers which 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.

3. Hypermedia data structures

The starting point 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 4* 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 imbellishment 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 atfributes 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.

The 'extensions' background and cards are freely created by the user for purposes such as

<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. 4. Stack structure.

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.

4. System operation

Figure 5 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 than a separate application with its own user interface.

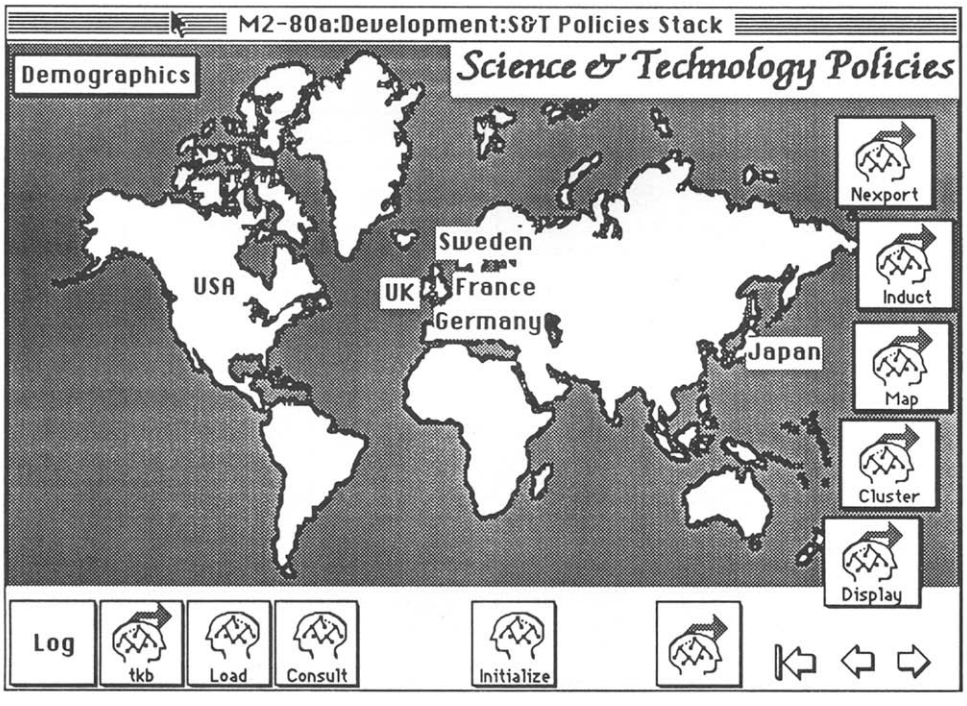


Fig. 5. Initial control card of 'Science and technology policies' stack.

However, HyperCard currently does not provide the variable size colored windows necessary to the full functionality of the acquisition application.

Figure 6 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 7 shows how the data card of *Fig. 6* 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 user presses the mouse button the pop-up menu appears as shown in *Figure 7*, offering a range of actions, one of which is transferred 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.

The pop-up menus are consistently available in all contexts. *Figure 8* 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.

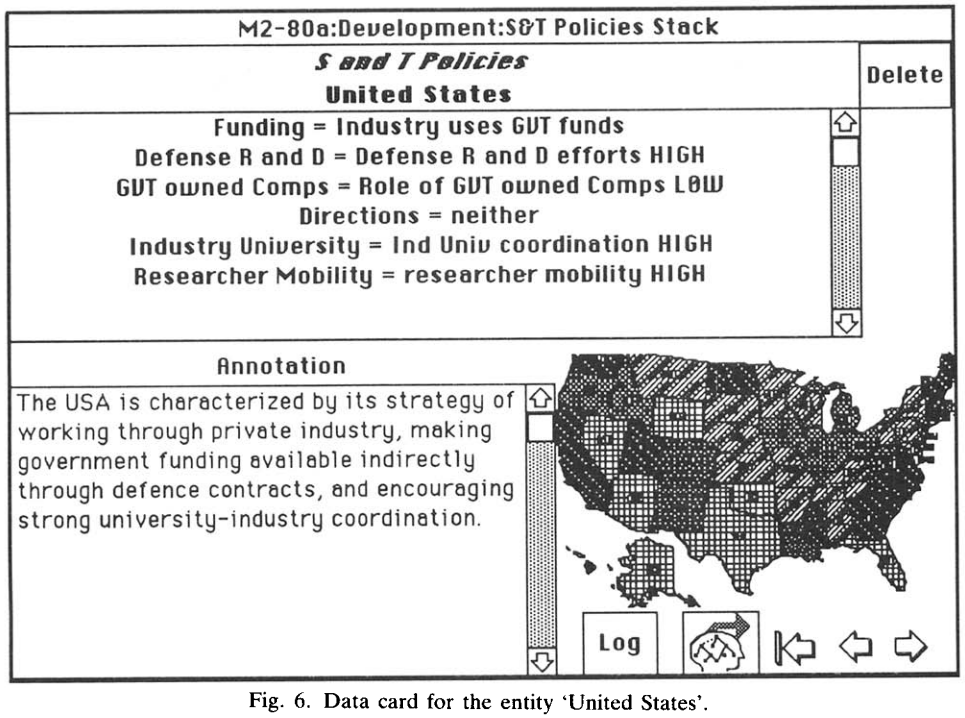


Fig. 6. Data card for the entity 'United States'.

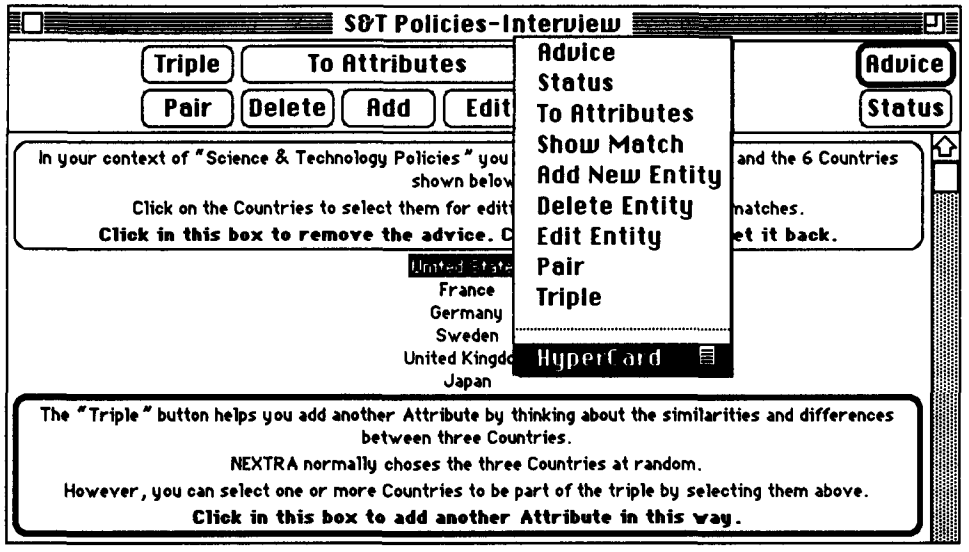


Fig. 7. Context-sensitive pop-up menu in the knowledge acquisition tool.

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 *Fig. 5*. 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.

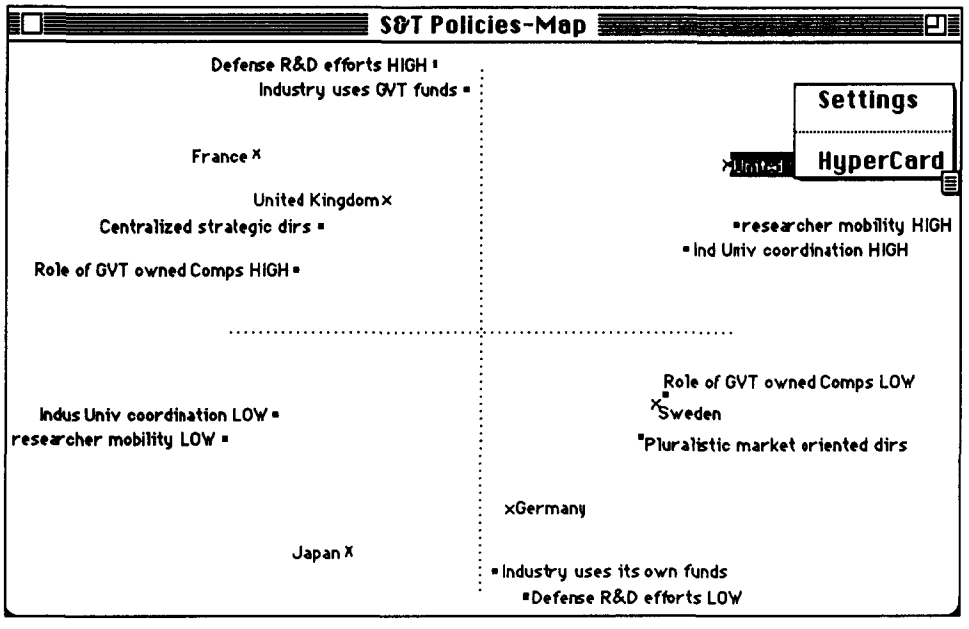


Fig. 8. Context-sensitive pop-up menu in conceptual map.

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 *Fig. 9*. 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 *Fig. 9*, 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 *Fig. 10*. 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. 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 *Fig. 10* which results in transfer to a query card again using the appropriate attribute data card. Clicking on the required value in the query card transfers control back to the card of *Fig. 10* with the new value entered.

When the editing is complete a new entity has been created as shown in *Fig. 11*. 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

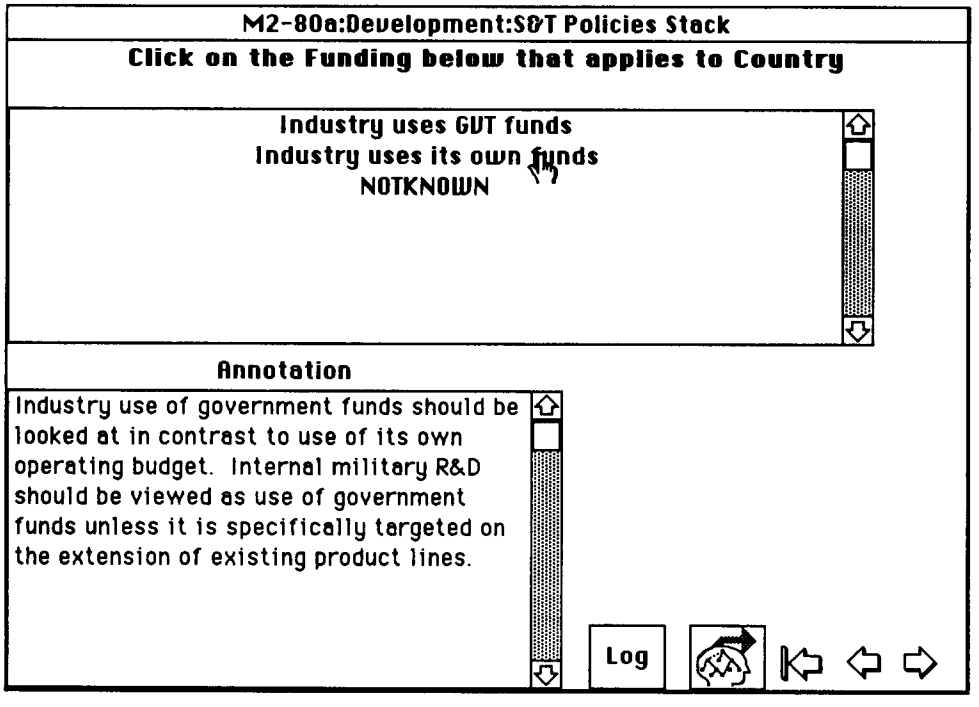


Fig. 9. User query through data card in a consultation.

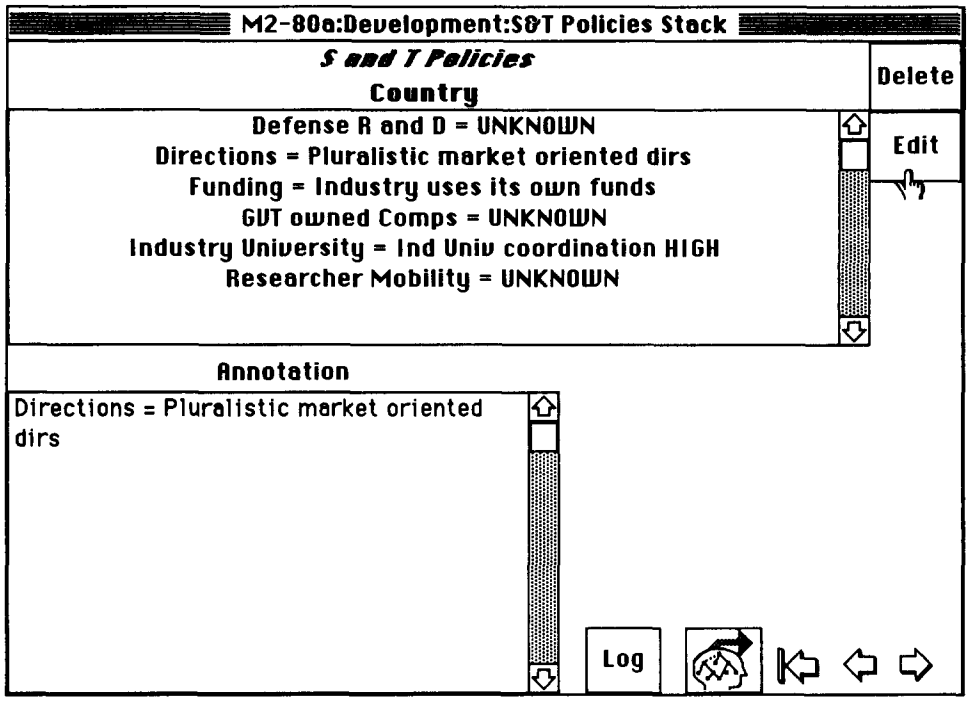


Fig. 10. Results of a consultation.

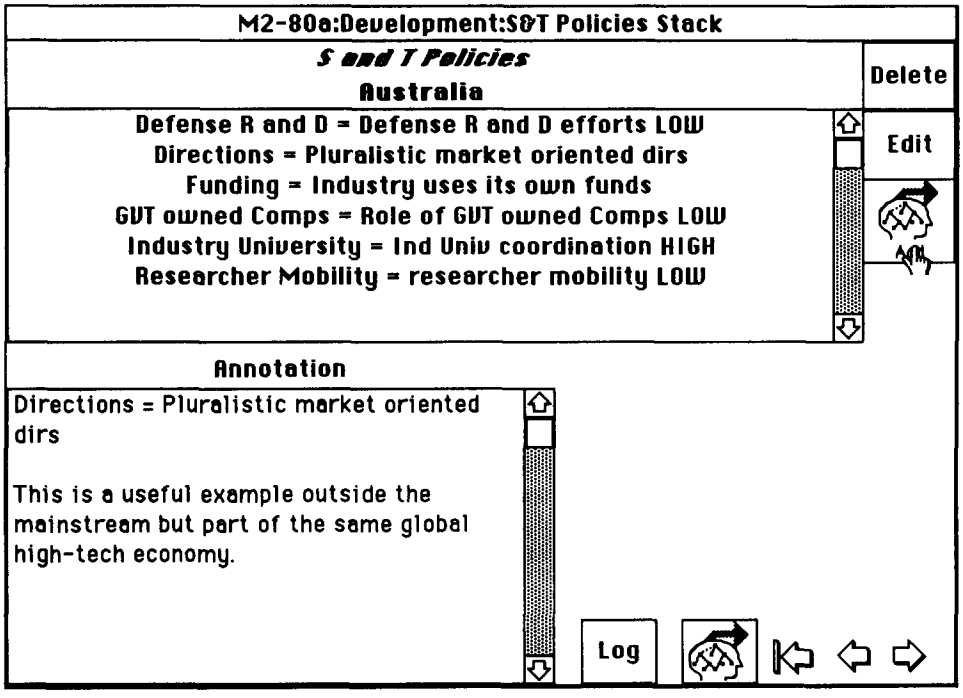


Fig. 11. Adding edited results as new entity in the dataset.

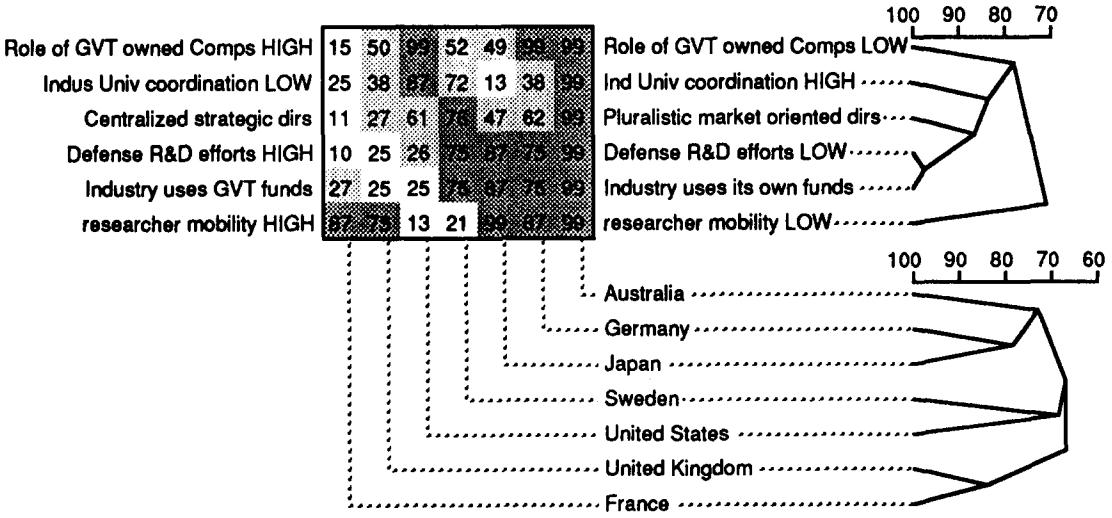


Fig. 12. Concept cluster showing new entity added.

shell. The effect of the new entity on the conceptual structure may be investigated immediately within the knowledge acquisition tool – for example, *Fig. 12* shows the resulting conceptual cluster including the new entity.

In terms of *Fig. 2*, the cluster analyses of *Figs 8* and *12* support the expert and knowledge engineer in evaluating the coherence of knowledge structures they have entered. The hypertext annotation structures of *Figs 6* and *11*, an the capability to move back and forth between these and the knowledge structures using pop-up menus support the expert and knowledge engineer in evaluating the completeness of the knowledge structures. The capability to undertake a consultation as part of knowledge acquisition as shown in *Figs 9* and *10*, and the capability to post back erroneous cases with corrections as shown in *Fig. 11*, support the evaluation of correctness of the knowledge structures. The evaluation of consistency has not been shown explicitly but is one of the features of the validation module in the rule induction subsystem.

5. 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 in 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 [5, 30], 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 [44] 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. Repertory grid tools [2, 38, 39] 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 had 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.

6. 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. At the communication level the integration is supported through inter-application communication protocols providing loose coupling of four programs designed independently for stand-alone use. At the knowledge level 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 *Figs 1* and *2*. The heterogeneous integration approach has proved an effective way of rapidly developing systems with new functionality. In another study, it has also been used to link knowledge acquisition tools to the German knowledge-based system shell, BABYLON [8, 21], and is the basis of a knowledge representation server-based knowledge acquisition architecture [19].

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 [43] and Gaines [14] 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 [16]. To participate fully in the knowledge processes shown in *Figs 1* and *15*, 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 to 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 [40]:

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 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 require 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 [3] 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 worldwide.

References

[1] J.H. Boose, Personal construct theory and the transfer of human expertise, *Proc. AAAI-84* (American Association for Artificial Intelligence, Palo Alto, CA, 1984) 27–33.

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

[3] J.H. Boose, A survey of knowledge acquisition techniques and tools, *Knowledge Acquisition* 1(1) (Mar. 1989) 39–58.

[4] J.H. Boose and J.M. Bradshaw, Expertise transfer and complex problems: Using AQUINAS as a knowledge acquisition workbench for knowledge-based systems, *Internat. J. Man-Machine Stud.* 26 (1987) 3–28.

[5] J.H. Boose and B.R. Gaines, *Knowledge Acquisition Tools for Expert Systems* (Academic Press, London, 1988).

[6] J.H. Boose and B.R. Gaines, *The Foundations of Knowledge Acquisition* (Academic Press, London, 1990).

[7] J. Cendrowska, An algorithm for inducing modular rules, *Internat. J. Man-Machine Stud.* 27(4) (Oct. 1987) 349–370.

[8] T. Christaller, F. di Primio and A. Voß, Die KI-Werkbank BABYLON (Addison Wesley, Bonn, 1989).

[9] J. Diederich, I. Ruhmann and M. May, KRITON: A knowledge acquisition tool for expert systems, *Internat. J. Man-Machine Stud.* 26(1) (1987) 29–40.

[10] L. Eshelman, D. Ehret, J. McDermott and M. Tan, MOLE: A tenacious knowledge acquisition tool, *Internat. J. Man-Machine Stud.* 26(1) (1987) 41–54.

[11] E.A. Feigenbaum, Knowledge engineering: The applied side of Artificial Intelligence, Report STAN-CS-80-812, Department of Computer Science, Stanford University, 1980.

[12] B.R. Gaines, An overview of knowledge acquisition and transfer, *Internat. J. Man-Machine Stud.* 26(4) (Apr. 1987) 453–472.

[13] B.R. Gaines, Rapid prototyping for expert systems, in M.D. Oliff, ed., *Intelligent Manufacturing: Proc. First Internat. Conf. on Expert Systems and the Leading Edge in Production Planning and Control* (Benjamin Cummins, Menlo Park, CA, 1987) 45–73.

[14] B.R. Gaines, Second generation knowledge acquisition systems, in *Proc. Second European Workshop on Knowledge Acquisition for Knowledge-Based Systems (EKAW'88)*, GMD-Studien Nr. 143 (Gellsellschaft für Mathematik und Datenverarbeitung, Bonn, Germany, Jun. 1988) 17-1–17-14.

[15] B.R. Gaines, Organizational integration: modeling technology and management, in P. Zunde and D. Hocking, eds., *Empirical Foundations of Information and Software Sciences II* (Plenum Press, New York, 1990) 51–64.

[16] B.R. Gaines, Positive feedback processes underlying the formation of expertise, *IEEE Trans. Systems Man Cybernet.* SMC-18(6) (Nov. 1988) 1016–1020.

[17] B.R. Gaines, Social and cognitive processes in knowledge acquisition, *Knowledge Acquisition* 1(1) (Mar. 1989) 39–58.

[18] B.R. Gaines, 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, in *Proc. 6th Internat. Workshop on Machine Learning* (Morgan Kaufmann, San Mateo, CA, 1989) 156–159.

[19] B.R. Gaines, Empirical investigation of knowledge representation servers: Design issues and applications experience with KRS, *SIGART Bull.* 2(3) (Jun. 1991) 45–56.

[20] B.R. Gaines and J.H. Boose, eds., *Knowledge Acquisition for Knowledge-Based Systems* (Academic Press, London, 1988).

[21] B.R. Gaines and M. Linster, Integrating a knowledge acquisition tool, an expert system shell and a hypermedia system. *Internat. J. Expert Systems Res. Appl.* 3(2) (1990) 105–129.

[22] B.R. Gaines and M.L.G. Shaw, Knowledge support systems, in *ACM MCC-University Res. Symp.* (MCC, Austin, TX, 1987) 47–66.

[23] D. Goodman, *The Complete HyperCard Handbook* (Bantam, New York, 1987).

[24] W.P. Jones, Bringing corporate knowledge into focus with CAMEO, *Knowledge Acquisition* 2(3) (Sept. 1990) 207–239.

[25] G. Klinker, J. Bentolila, S. Genetet, M. Grimes, and J. McDermott, KNACK – report-driven knowledge acquisition, *Internat. J. Man-Machine Stud.* 26(1) (1987) 65–79.

[26] S. Marcus, Taking backtracking with a grain of SALT, *Internat. J. Man-Machine Stud.* 26(4) (1987) 383–398.

[27] S. Marcus, ed., *Automating Knowledge Acquisition for Expert Systems* (Kluwer, New York, 1988).

[28] R.S. Michalski and R.L. Chilausky, Knowledge acquisition by encoding expert rules versus computer induction from examples – A case study involving soyabean pathology, *Internat. J. Man-Machine Stud.* 12 (1980) 63–87.

[29] E. Motta, M. Eisenstadt, K. Pitman and M. West, Support for knowledge acquisition in the knowledge engineer's assistant (KEATS), *Expert Syst.* 5(1) (1988) 6–28.

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

[31] J.R. Quinlan, Simplifying decision trees, *Internat. J. Man-Machine Stud.* 27(3) (Sept. 1987) 221–234.

[32] J.A. Rantanen, Hypermedia in knowledge acquisition and specification of user interface for KBS: an approach and a case study, *Knowledge Acquisition* 2(3) (Sept. 1990) 259–278.

[33] A. Rappaport, Multiple-problem subspaces in the knowledge design process. *Internat. J. Man-Machine Stud.* 26(4) (Apr. 1987) 435–452.

[34] A. Rappaport, Cognitive primitives, in J.H. Boose and B.R. Gaines, eds., *Proc. Second AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop* (Oct. 1987) 15-0–15-13.

[35] A. Rappaport and B.R. Gaines, Integrated knowledge base building environments, *Knowledge Acquisition* 2(1) (Mar. 1988) 51–71.

[36] N. Rescher, *Cognitive Systematizatio* (Basil Blackwell, Oxford, 1979).

[37] D. Shafer, *Using Oracle with HyperCard* (Hayden, Carmel, IN, 1990).

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

[39] M.L.G. Shaw and B.R. Gaines, A computer aid to knowledge engineering, in *Proc. British Comput. Soc. Conf. on Expert Systems* (Dec. 1983) 263–271.

[40] M.L.G. Shaw and B.R. Gaines, Advances in interactive knowledge engineering, in M.A. Bramer, ed., *Research and Development in Expert Systems III* (University Press, Cambridge, 1986) 111–122.

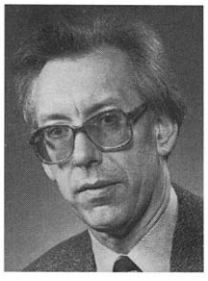
[41] M.L.G. Shaw and B.R. Gaines, KITTEN: Knowledge Initiation & Transfer Tools for Experts & Novices, *Internat. J. Man-Machine Stud.* 27(3) (Sept. 1987) 251–280.

[42] M.L.G. Shaw and B.R. Gaines, A methodology for recognizing conflict, correspondence, consensus and contrast in a knowledge acquisition

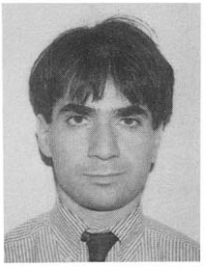
system, *Knowledge Acquisition* 1(4) (Dec. 1989) 341–363.

[43] L. Steels, Second generation expert systems, in M.A. Bramer, ed., Research and Development in Expert Systems III (University Press, Cambridge, 1986) 175–183.

[44] B. Woodward, Knowledge engineering at the fron-end: defining the domain, *Knowledge Acquisition* 2(1) (Mar. 1990) 73–94.



Dr. Brian R. Gaines is Killam Memorial Research Professor and Director of the Knowledge Science Institute at the University of Calgary. His previous positions include Professor of Industrial Engineering at the University of Toronto, Technical Director and Deputy Chairman of the Monotype Corporation, and Chairman of the Department of Electrical Engineering Science at the University of Essex. He received his B. A, M. A. and Ph. D. from Trinity College, Cambridge, and is a Chartered Engineer, and a Fellow of the Institution of Electrical Engineers, the British Computer Society and the British Psychological Society. He is editor of the *International Journal of Man-Machine Studies* and *Knowledge Acquisition*, and of the *Computers and People* and *Knowledge-Based Systems* book series. He has authored over 300 papers and authored or edited 10 books on a wide variety of aspects of computer and human systems. His research interests include: the socio-economic dynamics of science and technology; the nature, acquisition and transfer of knowledge; software engineering for heterogeneous systems; and expert system applications in manufacturing, the professions, sciences and humanities.



Alain T. Rappaport received an M. D. degree from the Université René Descartes, Necker Medical School (Paris VI) in 1984 and a Ph. D. in Molecular Pharmacology from the Université Pierre et Marie Curie (Paris VI) in 1984. Researcher in machine learning at Carnegie-Mellon University in 1984–85, prior to co-founding Neuron Data. Currently President and Chief Scientist of Neuron Data, Palo Alto, and adjunct research scientist, the Robotics Institute, School of Computer Science, Carnegie-Mellon University. Research interests include knowledge acquisition, reasoning architectures, cognitive foundations of AI and the relation of new generation software.

Dr Mildred L.G. Shaw is Professor of Computer Science at the University of Calgary. She received her B. Sc. and M. Sc. from the University of London, and her Ph. D. from Brunel University. She is a Fellow of the Institute of Mathematics and its Applications and the British Computer Society and an Associate Fellow of the British Psychological Society. Dr. Shaw is a member of the editorial boards of the *International Journal of Man-Machine Studies* and *Knowledge Acquisition*, and managing editor of *Future Computing Systems*. She has authored over 100 papers and authored or edited 5 books on a wide variety of aspects of computer and human systems. Her research interests include: human-computer interaction; the acquisition and transfer of knowledge; software engineering; and expert system applications.