3rd AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop
Banff, Canada, November 1988

INTEGRATION OF ACQUISITION AND PERFORMANCE

Alain T Rappaport and Brian R Gaines

Neuron Data, 444 High St, Palo Alto, CA 94301
and
Knowledge Science Institute, University of Calgary
Calgary, Alberta, Canada T2N 1N4

ABSTRACT

Knowledge base building environments must progress in two important directions: (i) increased participation of domain experts in the knowledge design process through new computational models and effective man-machine interfaces and (ii) automated knowledge acquisition tools to facilitate the overt expression of knowledge. This paper presents the integration of a knowledge acquisition methodology with a performance system. The resulting architecture represents a combination of techniques from psychology, cognitive sciences and artificial intelligence. New dimensions emerge from this implementation and integration both at the theoretical and practical levels. The overall system is not linked to a particular control structure and is not task dependent. We discuss the value of intermediate representations in this context and the role of different approaches to the induction process. Topological induction is particularly efficient in the elicitation process and stresses the importance of interactive inductive techniques with participation from the experts. While the knowledge acquisition tool provides an analysis and structuring of the domain knowledge, the control is implemented using the performance system's interface. Therefore, both modules participate in the overall knowledge acquisition process. Beyond the integration of these knowledge acquisition and performance systems, the architecture can also be integrated with databases, text analysis techniques, and hypermedia systems.

INTRODUCTION

The integration of knowledge acquisition tools and performance systems is a highly significant issue in knowledge-based system design. Achieving this integration raises problems of representational shift, domain and the task-dependence, knowledge validation and maintenance, and others. It brings to light new issues for AI research. While research continues on new representations and inference mechanisms, achieving such integration will provide new insights on how the techniques from the knowledge acquisition and machine learning communities relate to those involved in reasoning architectures. To make progress in integration, it is necessary to provide the techniques in a highly useable form which can be successfully implemented in real world environments. In the present paper, we explore some of these issues through the integration of methodologies based on the NEXPERT *OBJECT* and NEXTRA tools.

Figure 1 illustrates the knowledge acquisition process for current systems. The system developer draws upon three major knowledge sources: knowledge encoded in *relevant media*, such as books, journals and videotapes; knowledge available by observing and modeling the *relevant domain*; and knowledge available from discourse with, and observation of, the *relevant community*. It is this last source that is most associated with expert systems as such, although all three sources play significant roles in the development of any practical system.

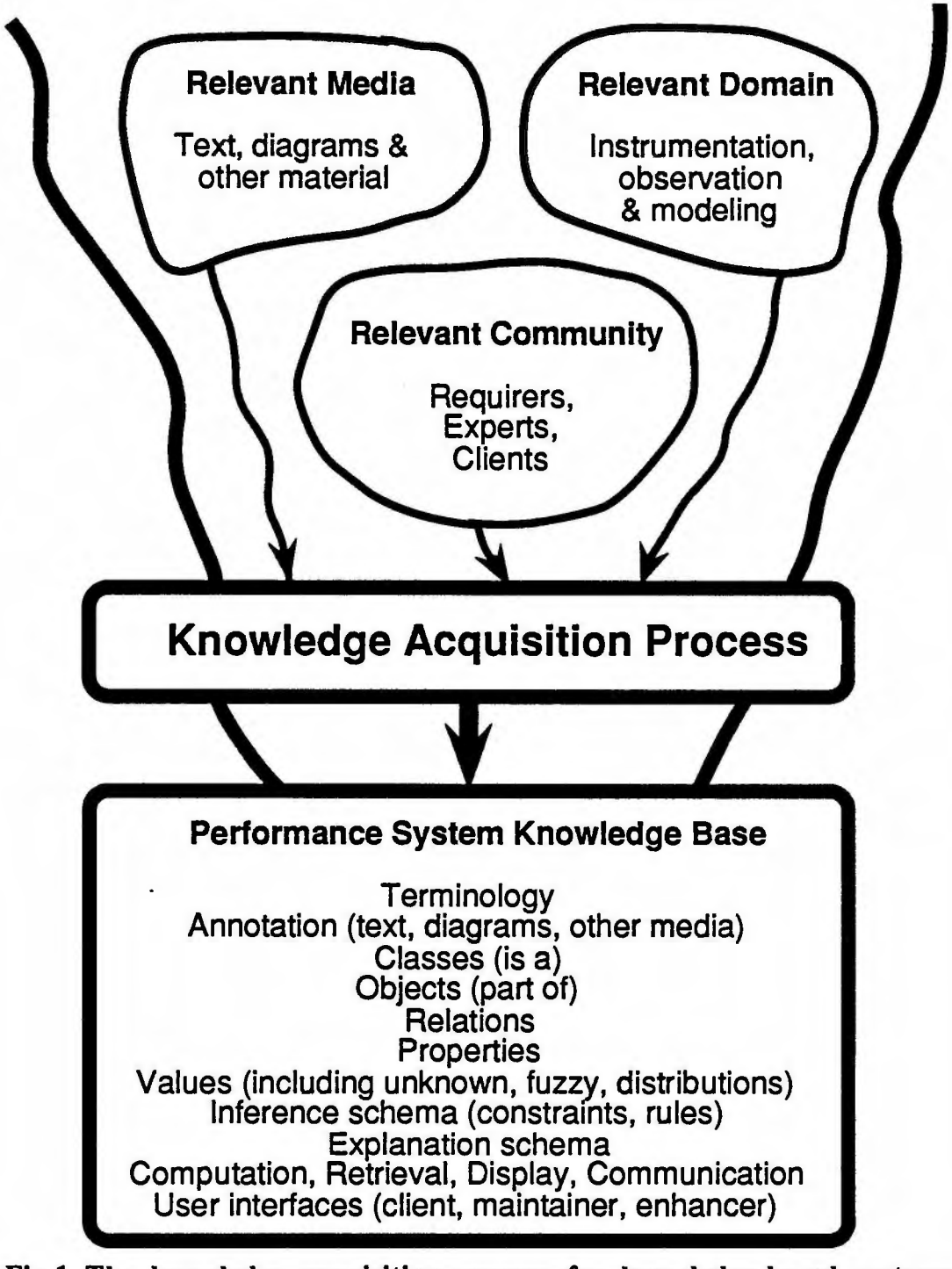

Fig.1 The knowledge acquisition process for knowledge-based systems

Note that the *relevant community* has been split into requirers, experts and clients. Domain experts are the expected source of knowledge for an expert system, but detailed requirements specifications and involvement of end-users play as a significant role as they do in other computing system design. It is useful to think of those with these different roles as being all 'experts' in some aspect of the system functioning, since the knowledge acquisition techniques used with domain experts are also well-suited to establishing requirements specifications and end-user perspectives.

The content of the *performance system knowledge base* in Figure 1 illustrates the range of technical features which an adequate knowledge-based system shell may be expected to support:

- *Terminology* is often neglected as computationally significant. However, it is particularly important in knowledge-based systems where it is critically important that the terms used in questions, recommendations and explanations are understood by clients in the same way as they are by experts. Obtaining the terminology direct from experts without distortion is an important design consideration for knowledge acquisition systems.
- *Annotation* is used here as a term for non-computational knowledge such as text, diagrams, photographs, videos, and other media, that may be used to impart knowledge to people. Some of it may not be suitable for computer storage but not for inferential processing. Providing appropriate access to such knowledge is important in many knowledge-based systems, and collecting and organizing such knowledge is part of the knowledge acquisition process.
- *Classes, objects, relations, properties* and *values* form the abstract infrastructure of computational knowledge representation currently. Object-orientated knowledge-bases provide a formal conceptual and computational schema which subsumes and supports a wide variety of knowledge representation schema (Shriver & Wegner 1987). These primitives also correspond to natural cognitive operations of classifying, instantiating, relating, attributing and measuring. An important aspect of knowledge acquisition is to determine these cognitive primitives in a domain (Rappaport 1987b).

  Where knowledge representation goes beyond conventional data representation is largely in the variety of types of value that must be supported. In a data base, a numeric field may be expected to have a clearly defined value. In a knowledge base, because knowledge is 'less than truth', it may be necessary to allow for the value being unknown, constrained by a fuzzy hedge, a range, a distribution, or some combination of these. There may also be mutual constraints between such ill-defined values.
- *Inference schema* are systems of constraints that express the relations between values in a knowledge base. These constraints may be thought of as rules expressing the relations and many of them may necessarily be expressed by explicit rules in the knowledge-base. However, the effective organization of the knowledge base in terms of generic classes and relationships should allow the majority of the constraints to be expressed implicitly in a natural way. This is the *framing problem* in knowledge representation for a domain, and an important aspect of knowledge acquisition is to determine a representation schema that minimizes the number of low level rules required.
- *Explanation schema* are forms of argument that generate satisfactory explanations of the inferences made from a knowledge base. Early expert systems generated explanations by displaying the chain of reasoning that was used from premises to conclusions. This approach is useful but does not necessarily 'explain' the conclusions in a way that an expert would, or in a way satisfying a client's need for knowledge. The acquisition of knowledge necessary for adequate explanation goes beyond that adequate for performance.
- *Computation, retrieval, display, communication* and other data processing activities play major roles in most knowledge-based systems. Modern expert system shells make provision for integration with data-processing, simulation, database, graphics and communications programs. The systems analysis for these needs to be integrated with the knowledge acquisition process.
- User interfaces are very important in knowledge-based systems as they are generally highly interactive and involve clients with little or no computer experience. The design of the interface in consultation with experts and clients is an important part of knowledge acquisition. Since most knowledge is fairly volatile, knowledge acquisition and updating the knowledge base are important aspects of system operation, and the interface to system maintainers and enhancers also needs careful attention.

NEXPERT AND NEXTRA

In our integrated knowledge support system the knowledge acquisition process is carried out by NEXTRA which creates a knowledge base for use by the performance system NEXPERT. One of our objectives is to create even closer integration by having the arrow from acquisition to performance be reversible, so that the knowledge base itself is always open to further development using the acquisition tools, even when it has been edited in the support environment of the performance tool. However, Figure 1 accurately represents the current state of the art in which acquisition takes place as a separate step prior to performance.

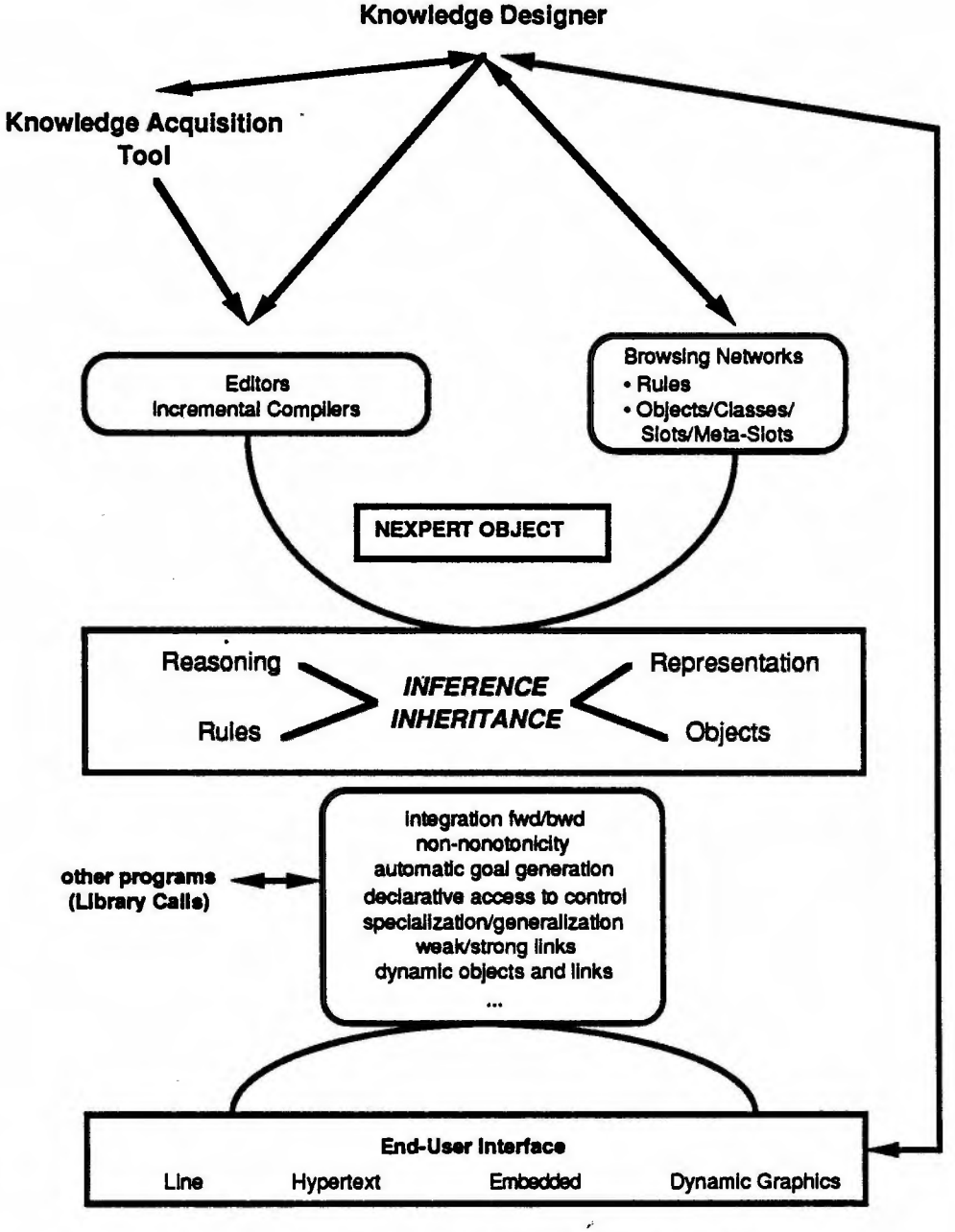

Fig.2 Architecture of NEXPERT *OBJECT*

NEXPERT *OBJECT* is a general-purpose development environment for the building of knowledge based systems with the architecture shown in Figure 2. Applications cover a wide range of tasks, from analytic ones such as interpretation and classification to synthetic tasks such as scheduling and planning. The generality of the tool results from an approach based on "cognitive primitives" (Rappaport, 1987b). These allow a great flexibility by permitting the developer or domain expert to build the control structure in terms of comprehensible primitives. This approach contrasts with earlier approaches based on the separation of control from knowledge. This would be ideal is the control structures were ideal. In most cases, however, when a system follows this separation, the restrictions imposed by the control structure make it task-dependent. Another fundamental aspect of the NEXPERT performance system is the highly graphic interface to visualize the reasoning and representation dimensions of the knowledge bases (Rappaport, 1987a).

NEXTRA is a knowledge acquisition toolbox based on an extensible set of techniques developed for the knowledge support system KSS0 (Gaines 1987a). The NEXTRA technology encompasses elicitation tools, visual analysis and display tools, group comparison tools, inductive tools, and knowledge base generative tools as shown in Figure 3.

The interviewing tools have their foundations in the personal construct psychology of Kelly (1955), subsequent work on the computer elicitation of conceptual structures through repertory grids (Shaw 1980), and applications of these techniques to knowledge elicitation (Boose 1984, 1986, Boose & Bradshaw 1986, Shaw & Gaines 1983, 1986, 1987a,b). The objective is to gather domain knowledge from prototypical entities represented along certain dimensions:

- **Interview** accepts specifications of entities within a sub-domain and provides an interactive graphical elicitation environment within which the experts can distinguish entities to derive their attributes. The resultant class is continuously analyzed to provide feedback prompting the expert to enter further entities and attributes.

The visual analysis tools consist of an interactive interface to represent the abstractions derived from those entities in terms of hierarchical clusters using Shaw's (1980) FOCUS algorithm, and relational diagrams such as a non-hierarchical conceptual maps derived through principal components analysis (Slater 1976). The objectives are to validate the raw domain knowledge and suggest further structure at a higher level through interactive topological induction:

- **Cluster** hierarchically clusters entities and attributes within a sub-domain prompting the experts to add higher-level entities structuring the domain.
- **Map** spatially clusters entities and attributes within a sub-domain prompting the experts to add higher-level entities structuring the domain.

The group comparison tools consist of an interactive interface to represent the relations between the terminologies and conceptual systems of different experts, or experts and clients. The objectives are to determine the consensus, conflict, correspondence and contrast between different conceptual systems (Shaw & Gaines 1988):

- **Compare** compares the structures for the same sub-domain generated by different experts, or the same expert at different times or from varying perspectives.

The inductive part consists in the derivation of constraints within the conceptual structures through logical entailment analyses (Gaines and Shaw 1985, Quinlan 1987, Cendrowska 1987). The objective is to suggest further structure at a higher level that translates into class inclusions or rules in NEXPERT:

- **Entail and Induct** induce logical entailments enabling the attributes of an entity, or the evaluations of a decision-making situation in a domain, to be derived from other attributes.

The generative part consists in the transformation of the knowledge analysis made by the previous tools into a formalism understandable by the NEXPERT inference mechanisms:

- **Nexport** formats the specifications of sub-domains as classes, of entities as objects, of attributes as properties, and of entailments as methods, and transfers them to the performance tool.

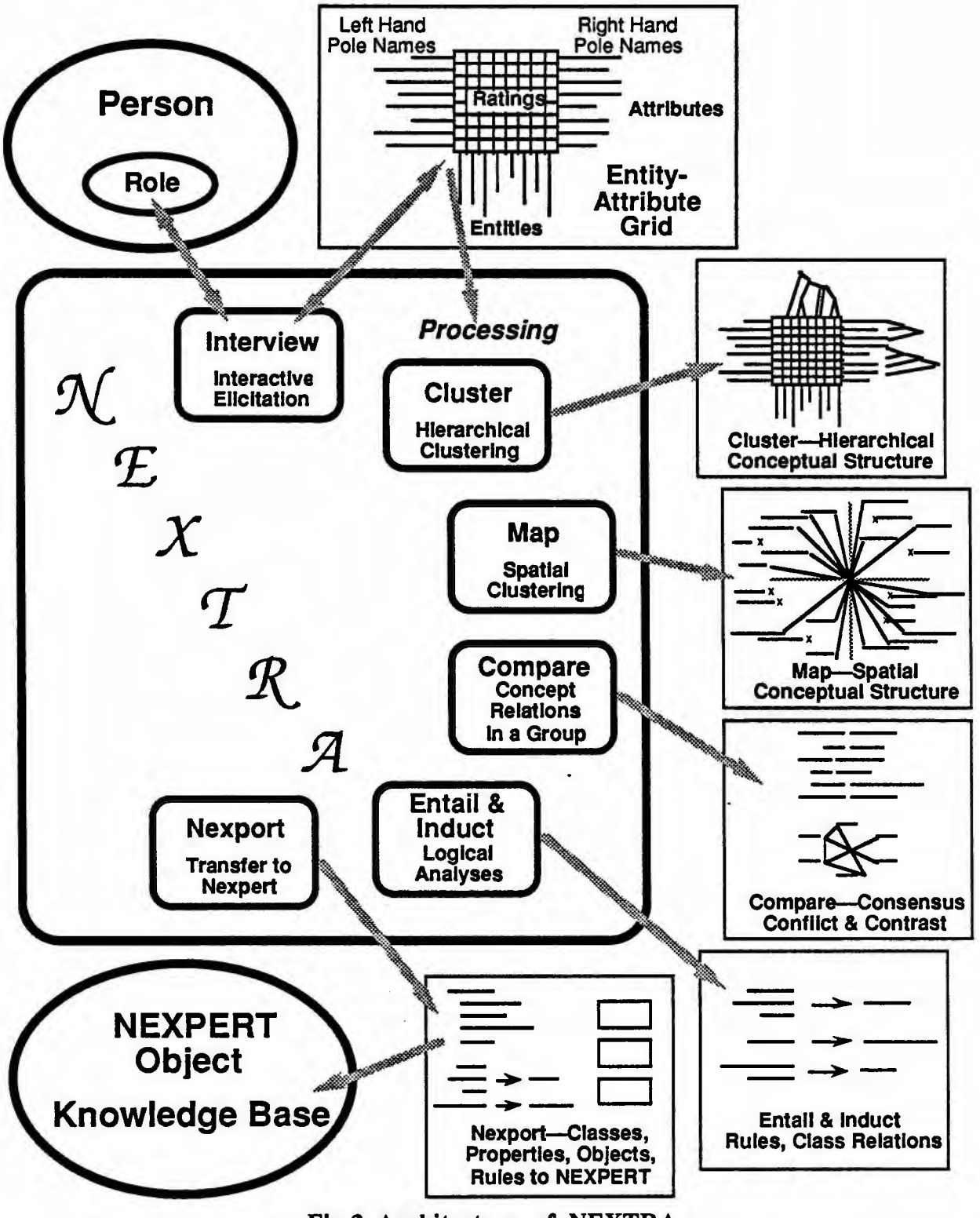

Fig.3 Architecture of NEXTRA

KNOWLEDGE CREATION, EDITING AND TRANSFER

A major aspect of the integration of NEXTRA and NEXPERT lies in the variety of debugging cycles due to the multiplicity of representations available. Whether the knowledge entry comes from elicitation, interactive topological induction, or an inductive algorithm, the immediate transfer into the performance systems allows the validation of knowledge at any stage of development. In fact, as shown in Figure 4, two general ways of editing or creating knowledge are now available: (i) through the knowledge acquisition tools and (ii) through the performance system development.

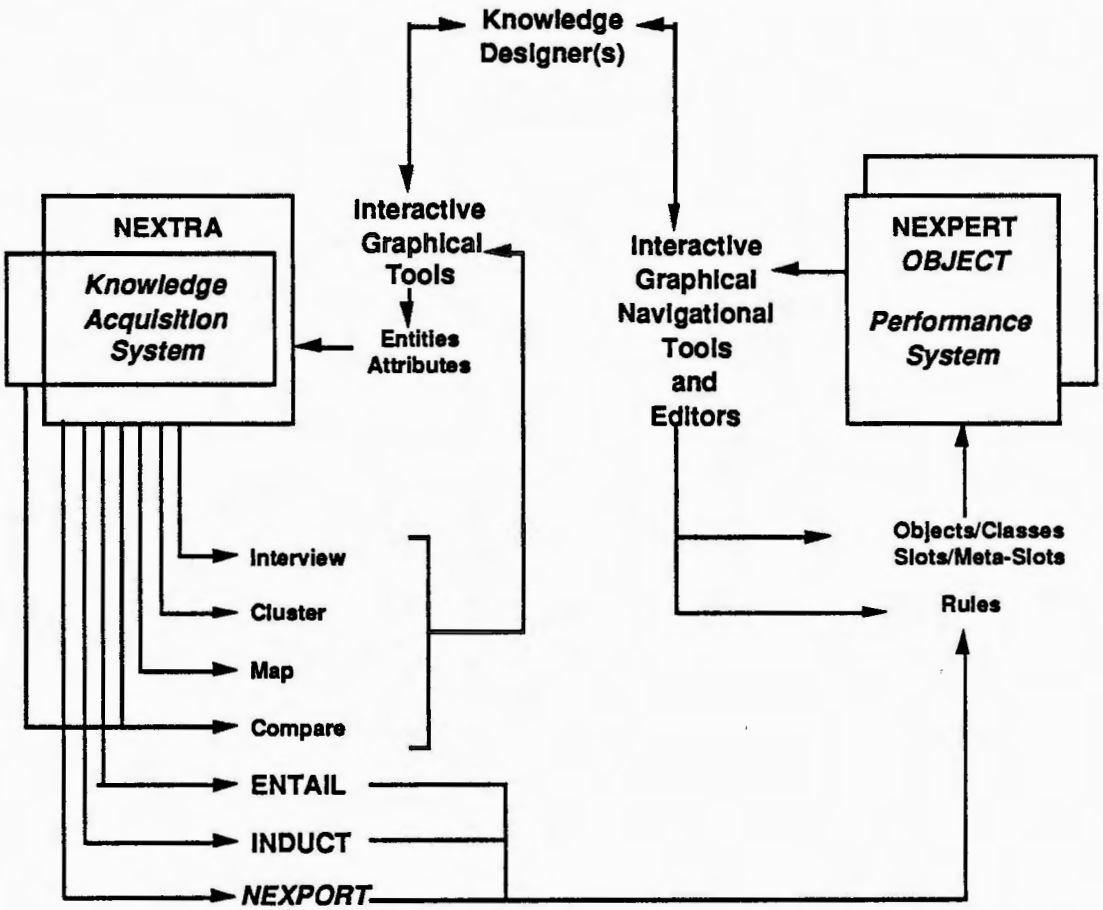

Fig.4 Integration of two knowledge creation and editing systems

The **Interview** tool allows entities and attributes to be specified through an interactive click and drag dialog as shown in Figure 5. The **Cluster** tool sorts the entities and attributes to produce a hierarchical cluster as shown in Figure 6. The **Map** tool arranges the entities and attributes to produce a spatial cluster as shown in Figure 7. The **Compare** tool matches entities and attributes between grids to produce correspondences as shown in Figure 8. The **Induct** tool derives entailments between attributes as shown in Figure 9. The **Nexport** tool converts attribute sets to class definitions, entities to objects and entailments to rules and formats them as a NEXPERT knowledge base as shown in Figure 10. The classes, objects and rules can then be edited in the NEXPERT graphic editing environment as shown in Figure 11. A typical expert system development will involve a number of elicitations with NEXTRA to develop the conceptual structures for the sub-domains followed by the use of NEXPERT editing tools to link the sub-domains through appropriate control structures.

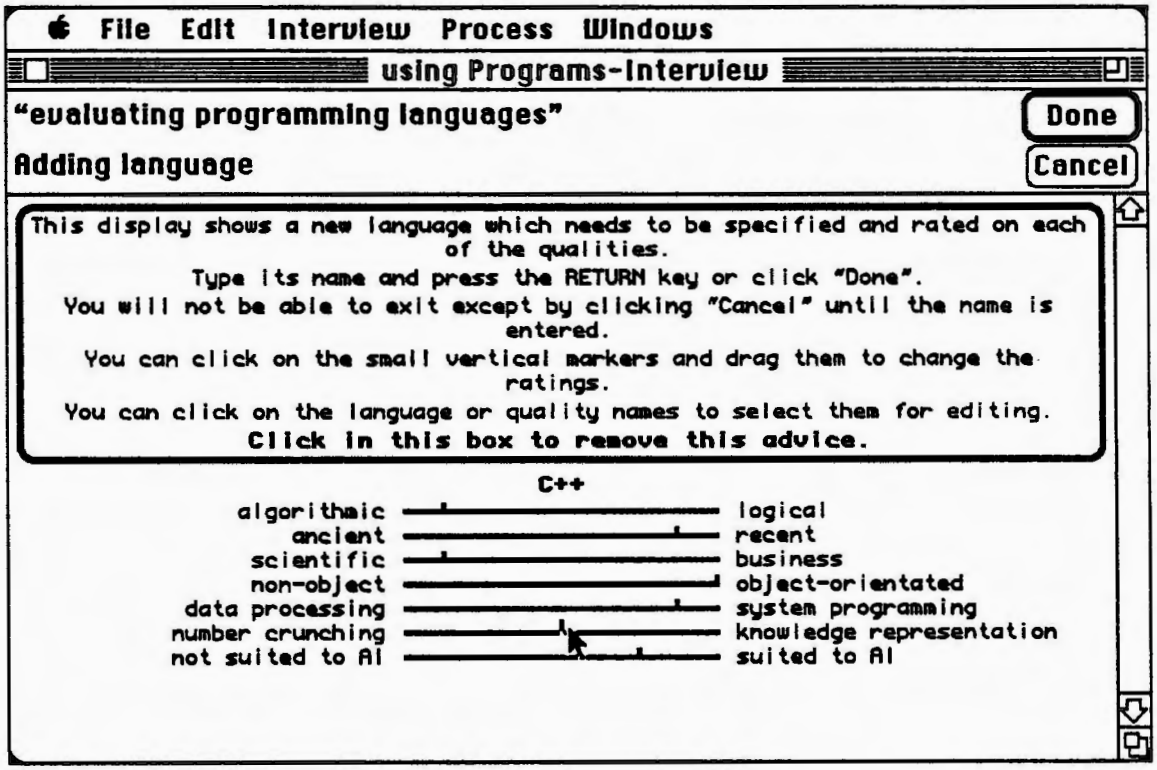

Fig.5 Interview entity rating screen

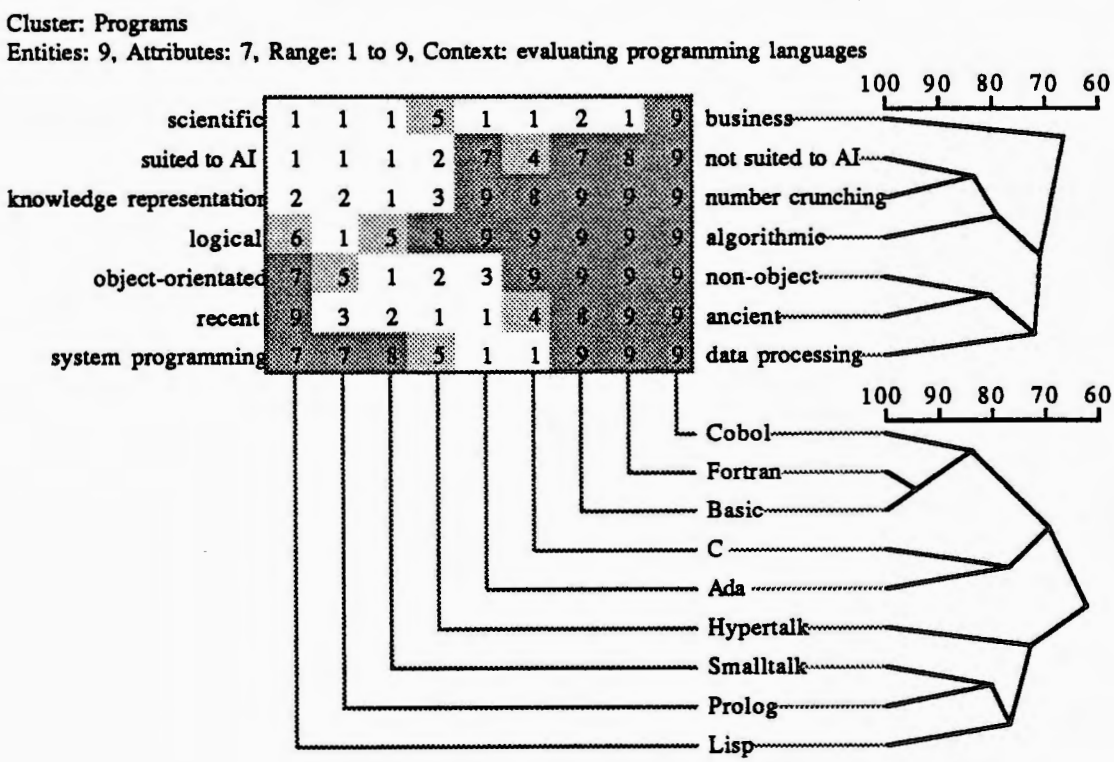

Fig.6 Cluster hierarchical display

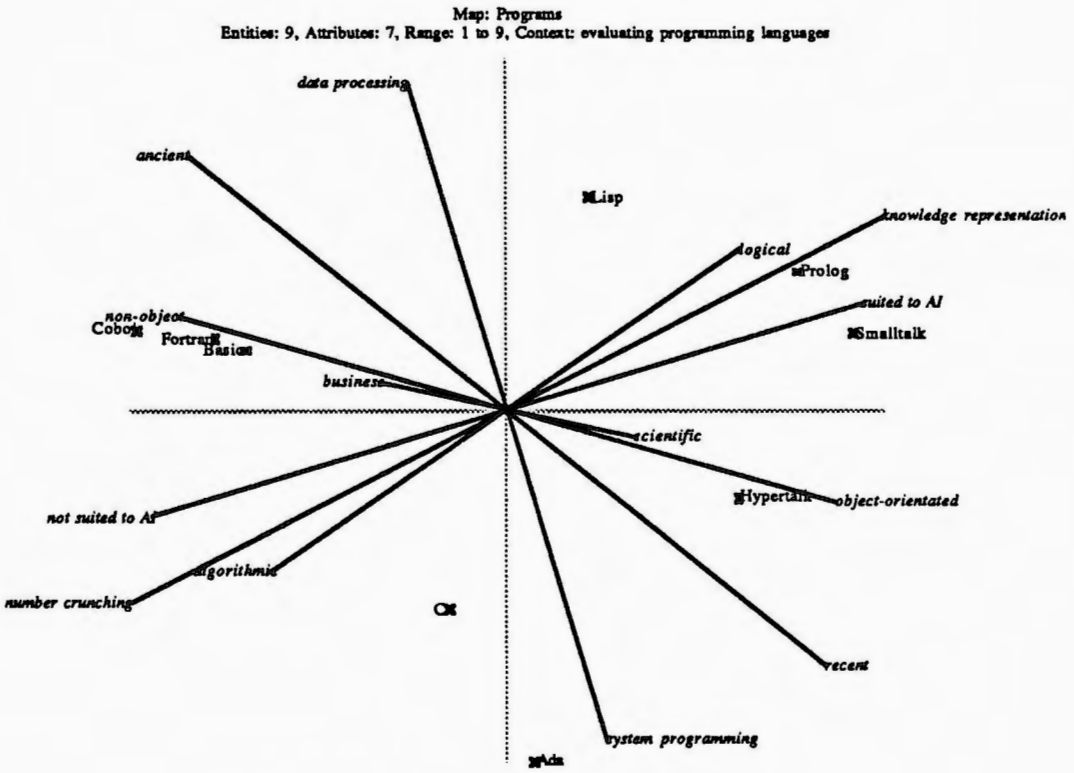

Fig.7 Map spatial display

```
1:   6.2% > 88.5    G1A2: Local - Global
                    G2A2: local - global
2:  12.5% > 87.5    G1A3: Low level data - High level data
                    G2A8:   nominal data - interval or ratio data
3:  18.8% > 86.5    G1A1: Does not honour data points - Honours data points
                    G2A4:  doesn't honour data points - honours data points
4:  25.0% > 86.5    G1A7: Short distance autocorrelation - Long distance autocorrelation
                    G2A2:                          local - global
5:  31.2% > 86.5    G1A9: New geographical technique - Old geographical technique
                   G2A18:            not widely used - widely used
6:  37.5% > 85.4   G1A16: Not widely used - Widely used
                   G2A18: not widely used - widely used
7:  43.8% > 83.3    G1A5: Discontinuous - Continuous
                    G2A2:         local - global
8:  50.0% > 82.3    G1A4: Mathematically complex - Mathematically simple
                   G2A11:   heavy computing load - no computing load
9:  56.2% > 81.2   G1A10:   Hard to adapt to multivariate - Easy to adapt to multivariate
                    G2A5: usually one variable considered - multiple variables considered
10: 62.5% > 81.2   G1A12:   Does not require spatial search - Requires spatial search
                   G2A13: estimates susceptible to clusters - not as susceptible to clusters
11: 68.8% > 79.2    G1A6: Does not require a priori model - Requires a priori model
                    G2A2:                          local - global
12: 75.0% > 79.2   G1A11:      Few points - Many points
                   G2A18: not widely used - widely used
13: 81.2% > 76.0   G1A13:          Does not use polynomial - Uses polynomial
                    G2A6: doesn't fit a mathematical curve - mathematical curve fitting
14: 87.5% > 76.0   G1A15: Not very effective - Very effective
                   G2A17: not very effective - very effective
15: 93.8% > 72.9   G1A14: Not very important - Very important
                   G2A18:    not widely used - widely used
```

Fig.8 Compare attribute correspondence display

```
recent≥3 & non-object≥2 -> logical≥3 (1/9:1/1 88.89)
non-object≥3 -> ancient≥2 (5/9:5/5 94.71)
non-object≥3 & data processing≥2 -> ancient≥3 (4/9:4/4 96.10)
object-orientated≥2 -> recent≥2 (5/9:4/4 90.47)
object-orientated≥2 -> recent≥3 (4/9:4/4 96.10)
ancient≥2 -> non-object≥3 (5/9:5/5 94.71)
recent≥3 -> object-orientated≥2 (4/9:4/4 96.10)
algorithmic≥2 & recent≥3 -> object-orientated≥3 (3/9:3/3 96.30)
algorithmic≥3 & recent≥2 -> system programming≥2 (3/9:3/3 96.30)
recent≥2 & number crunching≥2 -> system programming≥3 (2/9:2/2 95.06)
not suited to AI≥2 -> number crunching≥3 (5/9:5/5 94.71)
suited to AI≥3 -> knowledge representation≥3 (4/9:4/4 96.10)
number crunching≥2 -> not suited to AI≥2 (5/9:5/5 94.71)
ancient≥3 & number crunching≥2 -> not suited to AI≥3 (4/9:3/3 91.22)
algorithmic≥3 & ancient≥3 -> not suited to AI≥3 (4/9:3/3 91.22)
algorithmic≥3 & data processing≥3 -> not suited to AI≥3 (4/9:3/3 91.22)
knowledge representation≥2 -> suited to AI≥3 (4/9:4/4 96.10)
```

Fig.9 Induct entailments between attributes

```
(@PROPERTY=       algorithmic    @TYPE=Boolean;)
(@PROPERTY=       logical        @TYPE=Boolean;)
(@PROPERTY=       ancient        @TYPE=Boolean;)
(@PROPERTY=       recent @TYPE=Boolean;)
(@PROPERTY=       non_object     @TYPE=Boolean;)
(@PROPERTY=       object_orientated     @TYPE=Boolean;)
(@PROPERTY=       data_processing       @TYPE=Boolean;)
(@PROPERTY=       system_programming    @TYPE=Boolean;)
(@PROPERTY=       number_crunching      @TYPE=Boolean;)
(@PROPERTY=       knowledge_representation     @TYPE=Boolean;)
(@PROPERTY=       not_suited_to_AI      @TYPE=Boolean;)
(@PROPERTY=       suited_to_AI   @TYPE=Boolean;)

(@CLASS=   languages
    (@PROPERTIES=
           algorithmic
           logical
           ancient
           recent
           non_object
           object_orientated
           data_processing
           system_programming
           number_crunching
           knowledge_representation
           not_suited_to_AI
           suited_to_AI
    )
)

(@RULE=    Ind3
    (@LHS=
           (Yes    (<languages>.algorithmic))
           (Yes    (<languages>.data_processing))
    )
    (@HYPO=       not_suited_to_AI)
    (@RHS=
           (Let    (<languages>.not_suited_to_AI)      (True))
    )
)
```

Fig.10 Nexport knowledge base

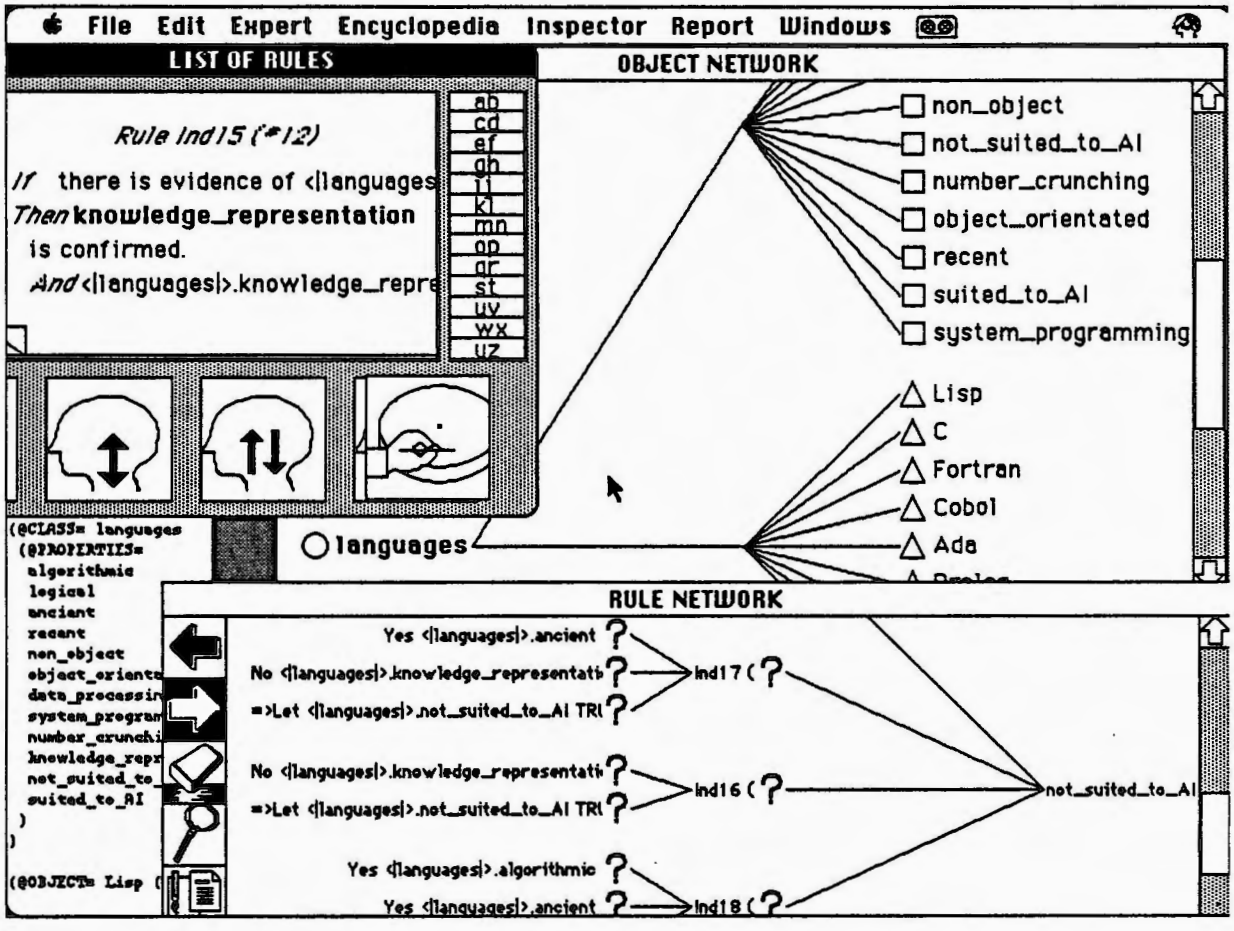

Fig.11 Graphic editing tools after knowledge transfer to NEXPERT

INTERMEDIATE REPRESENTATIONS

In terms of information, the transfer of knowledge into any given formalism is an irreversible task: there is a loss of information due to the limits of the artificial representation. The more steps between the original expression of the knowledge and the performance system which applies this knowledge, the more this effect applies. However, there may be also the emergence of information of a new kind, the derivation of higher-level structures corresponding to the development of the upper levels of the inductive modeling hierarchy (Gaines 1987b). The double effect of loss and gain of information is fundamental in the knowledge acquisition process. It means that while some information may be lost, a key role of the system is also to criticize the knowledge and derive new information. Knowledge acquisition must be as much a deconstructive as a reconstructive process. The technology becomes not only a medium to carry the original knowledge, but also a generative environment to enhance this knowledge.

Decoupling the knowledge acquisition process and the performance system adds risk to the overall process. In return, the pre-processing of the knowledge and its refinement before formalization provide the designer with greater understanding and fewer limitations. Gaining information in the knowledge acquisition phase can be achieved by two means: (1) automated systems that learn and (2) presenting the new information to the human in an understandable fashion. In either case, it is important to ensure that the information gained is understandable and useable.

In the NEXTRA approach, we have incorporated both perspectives. Their association leads to important results for the design of such systems. Since humans perceive the information with their sensory means, the first natural way to make this new information available is to use those means, which translates into graphic, interactive interfaces. For example, in the **Map** tool in NEXTRA: first, the domain expert enters the information using the interviewing scales; then the components analysis is performed by the system and new information is generated, the map of concepts in the space of selected dimensions; then the system creates a bi-dimensional representation of the results which the domain expert can understand and the new information is made accessible.

Figure 12 shows various levels of abstractions for the data structures of NEXTRA and NEXPERT. The initial data structure is a rating along a dimension, a low-level expression of partial knowledge. The output of the **Cluster** tool generates a more global vision which is made very perceptible using the **Map** tool. The generation of rules using algorithms such as Induct generates relations at a level which may be considered a decrease in abstraction. However, when the performance system is running, it provides the user with yet another type of knowledge.

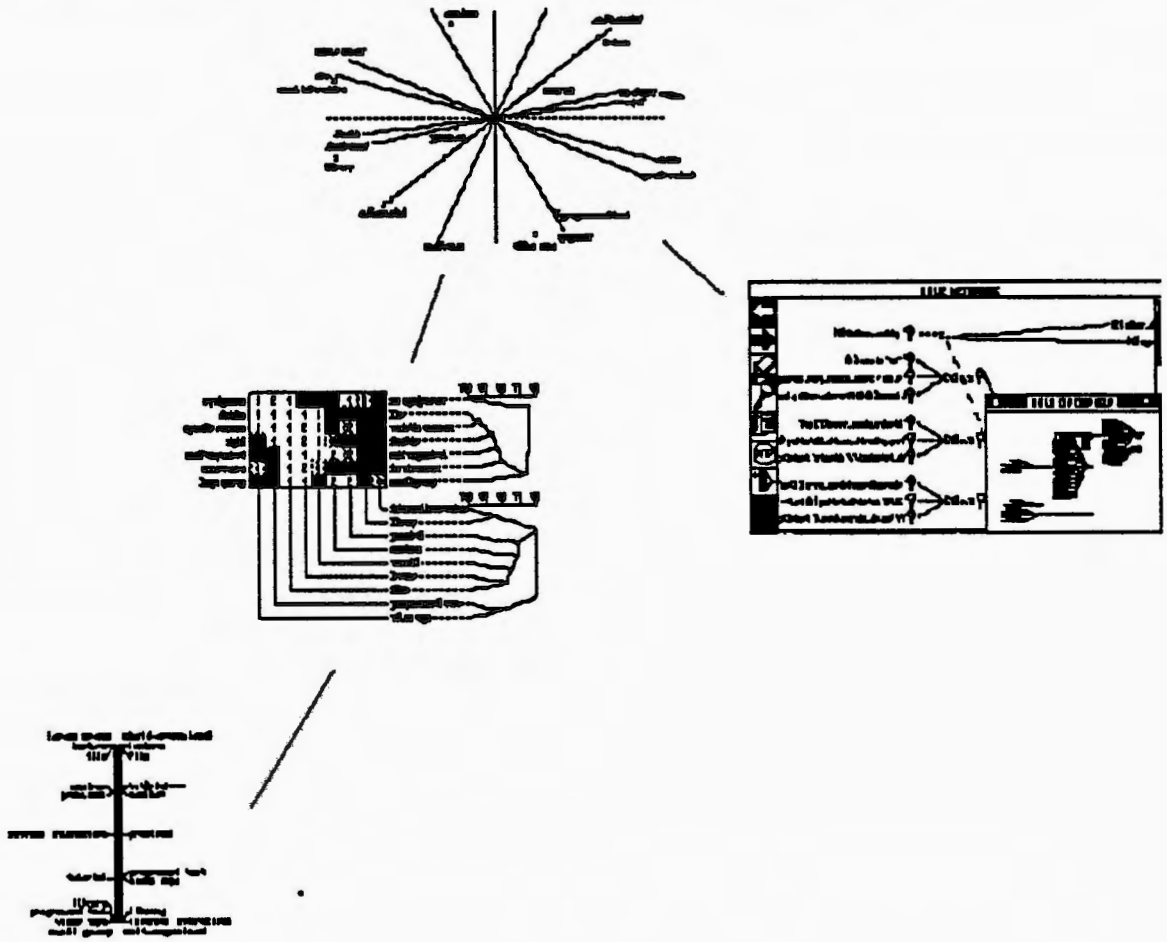

Fig.12 Levels of abstraction in NEXTRA and NEXPERT

TASK DEPENDENCIES

Knowledge formalization and transfer is difficult because many parameters are involved: the nature of the representation, the sources of knowledge, the nature of the task to be modeled and the specific aspects of the domain theory. If, however, the scope of the performance is well-circumscribed to a particular task and that some task-related theory is available, then it becomes easier to generate knowledge for such a system. Amongst the different tasks, some have proven harder to capture than other. Classically, analytic tasks are easier to model while knowledge acquisition for synthetic tasks is more difficult. The simplicity of some classification tasks lends itself to task-level editors and structure or constraint monitoring as well as a well defined debugging cycle. By contrast, a scheduling application typically involves more depth of representation and iterative or non-monotonic behaviors where the control structure is more predominant.

In general, when the reasoning paradigm and representation models are either rather simple or well defined, the knowledge acquisition tools will mainly help reduce the possible mismatches between the actual problem and the performance system. Because of the constraints of the final representation, having intermediate representation might be less useful for whatever is discovered might not be formalizable.

We can now relate the notion of task-dependency to that of intermediate representation. Intermediate representations are less useful when the control mechanisms are fixed. When, the control structure is to be designed, however, an intermediate representation addressing mostly the nature of the chunks of knowledge involved is necessary. Hence, the control structure is built on top of those chunks using either some other knowledge acquisition functions or the computational toolbox of the performance system. In the case of NEXTRA->NEXPERT, the NEXTRA techniques are to be used to mainly elicit the sub-domain conceptual structures, formulating the chunks of knowledge or *knowledge islands* relevant to the domain as shown in Figure 13.

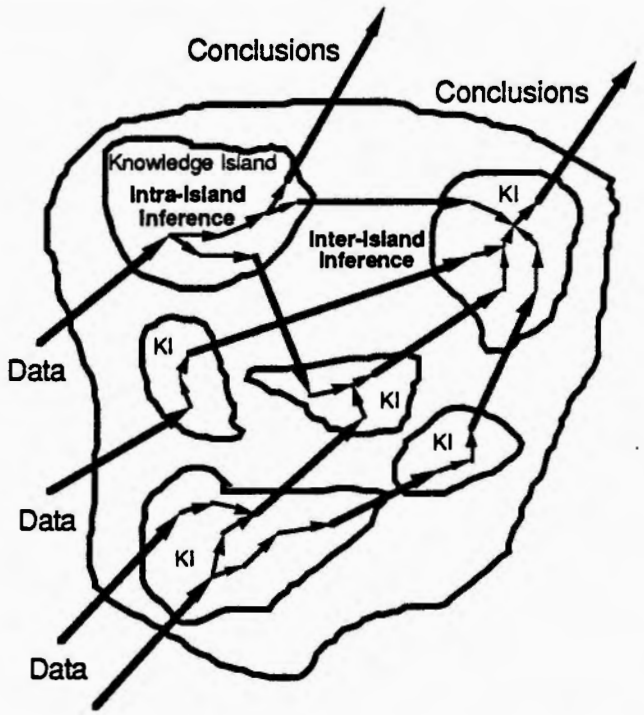

Fig.13 Structuring the domain through knowledge islands

Since the performance system allows one to design a wide variety of tasks, the control structure for transferring the focus of attention between knowledge islands is defined using the performance system's interface . The interface of the performance system itself allows not only to visualize but also to understand the mechanics of the generated knowledge base. From there, the knowledge can undergo multiple refinement and debugging cycles. Thus, in terms of Figure 13, NEXTRA's tools are used within knowledge islands, and NEXPERT's tools are used between them—note that, in these terms, the performance system is also part of the knowledge acquisition environment.

In future developments we intend to achieve even closer integration between acquisition and performance systems, treating the different knowledge representations and different related sub-domains as knowledge bases in their own right. Our overall problem can then be seen as one of cross-deriving, cross-relating and editing these knowledge bases to maintain their mutual consistency. Editing is often regarded as a mundane requirement. However, the variety of related knowledge representations that will usually be present in a knowledge-based system, particularly in the acquisition phase, making the editing of knowledge bases very difficult to support. If representation A, for example text, is processed to give representation B, for example entities, attributes and values, then the editing of the material in representation A should result in changes to that in representation B, and vice versa. However, the relation between representations is usually not functional but merely constrained, that is knowledge in representation A does not determine that in representation B, and vice versa, but only constrains it in some way as shown in Figure 14. Thus, the result of editing knowledge in one representation does not necessarily result in well-defined changes in other representations, only the detection of constraint violations. The editing system should mininimally highlight the violations, but usually it can do more, for example it can adumbrate "likely" changes in other represenations that will remove the violation.

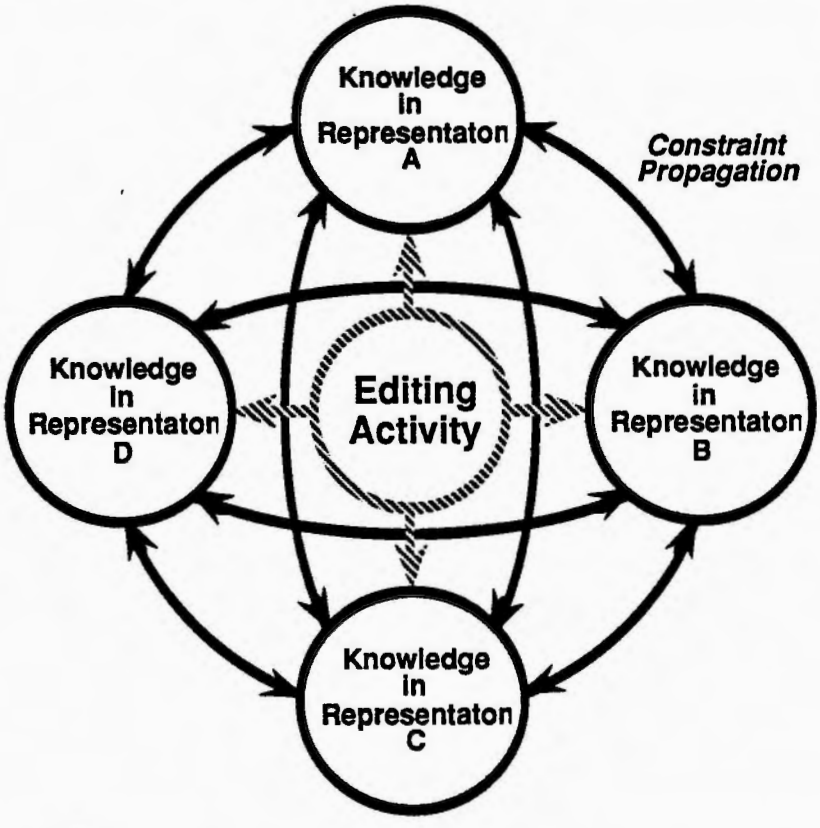

Fig.14 Mutual constraints between knowledge structures in editing

INTERACTIVE TOPOLOGICAL INDUCTION

Induction is a process of knowledge creation based on experience. However, it takes may forms. One can acquire knowledge from a single instance or from a large population of examples, but one also revises this knowledge, based again on any type of experience. One may also simply enrich or refine based on those experiences. In an other case, one compares, one tries and recognize the problem and/or its solution from past experiences. None of these processes are mutually exclusive in human information processing.

Many inductive methods have been described in AI, not including the general literature on the possible theories of induction. In general, the same constraints seem to apply to induction as to problem-solving itself. The main question is: what is the relevant information. As with some AI methodologies which are knowledge-intensive, induction algorithms which generate a sufficient amount of irrelevant knowledge are bound to fail. The mathematical consistency of the methods may provide a framework of validity, but the real framework for validation is whether the generated knowledge is in itself interesting, and not diluted in useless information as well.

Certainly, the algorithms must be mathematically sound, but the mathematical completeness is not necessary. Consider the human operations involved or believed to be involved in the induction process such as similarity, analogy, generalization. They involve the comparisons of abstract structures as well as the definition of operations allowing to transform them or relate them together. In these manipulation is a reflection of the intentions. Inductive algorithms lack the latter.

We believe that two different approaches to the induction problem must be used: (i) providing algorithms which will generate appropriate data structures upon analysis of a data set and (ii) providing an interactive environment where the results of computations are expressed in a graphic form, where the domain expert can discover similarities, differences and relationships.

Figure 15 shows an example of interaction which may provide important information using the **Map** tool. For any given situation (defined as a set of ratings for the constructs along a series of dimensions), there exists one mapping (with specific settings). Any operation such as modifying a rating, as shown on the figure, will yield a different result by principal components analysis and therefore a different map of the concepts. The clusters can be mathematically determined, according to the depth chose in the hierarchical representation (notion of threshold, which can be determined manually or according to some domain knowledge).

Thus, in the mind of the observer we have the following statement: if this event takes place then the overall map (or a subpart of it) is modified in such a way. This formulation, which can naturally be more complex, is then translated into a more domain-dependent syntax and constitutes the body of a new chunk of knowledge. The central argument lies in the power of the graphic representation in eliciting not only the operation but also the ability to formalize its result or impact. At all times, the system makes full use of the perception of the user. In the NEXTRA architecture, we are bringing all aspects of the interaction at the same level of visual abstraction. As a result, the rules will be edited directly from the mapping, but the mapping remains an important intermediate representation for the overall elicitation process.

Another example would be a single level operation whereby the existence and graphic representation of the clusters of concepts elicits the construction of relations between them. This is illustrated in the following Figure 16.

The **Cluster** and **Map** tools are based on mathematical distance-based analyses, but their output is an image as opposed to expert system code. They require another level of interactive analysis before generating the code in question.

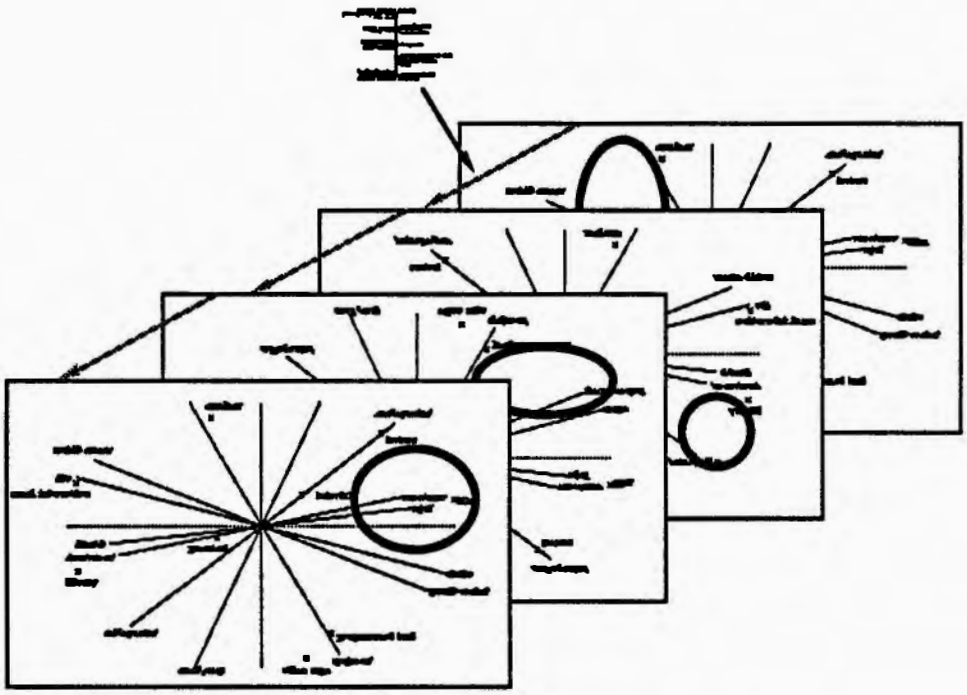

Fig.15 Knowledge analysis and creation in the graphic domain

By introducing the developer in the inductive process which makes the knowledge acquisition incremental, we reduce the amount of irrelevant information which may be generated by pure inductive techniques. What we may introduce though is "visual noise", or noise coming from the inherent lack of completeness of the human processes. We believe, however, that this is a more powerful approach, driven by intention and relations. Brittleness of the overall system will also come from the generated rules and the validity of the generalizations made by the system or elicited from the expert. Again, different means can be used. The more computational approach will try to keep degrees of certainty or decreases in information underlying any assumption. The latter should then be reflected in the rules and taken into account either by the inference engine or by some other technique interacting with the engine. At another level, closer to the graphic approach to the bounded rationality, we envision direct access to hypertext support, from either the concepts mapped in NEXTRA, or as is the case currently, at the level of the data represented in the rules.

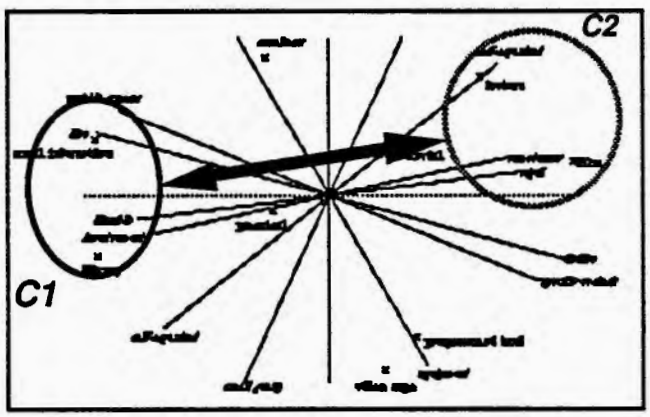

Fig.16 Relations in the graphic domain

FROM STATES TO TRAJECTORIES

Each operation performed in NEXTRA such as the modification of the rating of an attribute can be represented as a modification of the **Map** generated graph. As suggested above, the comparison between the two states may elicit a particular chunk of knowledge from the domain expert. Mathematical methods can be designed which will also formalize this transformation. However, comparing final and initial states can be a tedious approach if the final state is not well known in advance, or is too different from the initial state. When for instance many variables are involved which evolution follows different functions, it becomes necessary to be able to draw the trajectory followed by a particular entity or cluster. In order to obtain such a representation, the operations performed by the inductive tools must be made highly incremental. In order to achieve this, analytic models of the transformations are required. We are currently working in this direction.

Figure 17 illustrates the relative positioning of an entity in terms of the cluster to which it belongs, as a function of the evolution of one of its attributes A1. The initial state is described on top of the diagram, and the final state at the bottom. The final state is reached earlier than this (intermediate square). The shape of the curve may thus be indicative of the "fragility" of the initial description. In other terms, it is possible to observe the stability of the system's state with respect to one or more variables. The objective is to represent the nature of the state of the system and its evolution in a graphic form so as to elicit abstract knowledge, which would otherwise most likely remain buried in the expert mind. If such techniques allow the uncovering of expert knowledge, they certainly provoke the reflection on whatever knowledge is available and may also lead to the discovery of new knowledge.

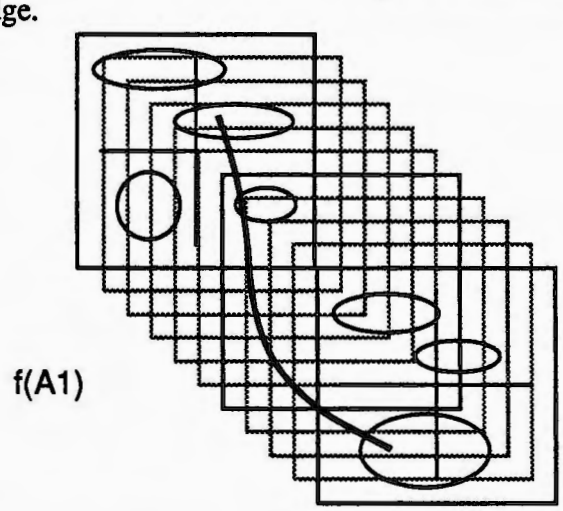

Fig.17 Trajectory of evolving knowledge in the graphic domain

INTERACTION, VISUALIZATION AND VALIDATION

We have emphasized in this paper the significance of user interaction with knowledge-based systems and the provision of facilities for users to visualize the knowledge base from different perspectives. This emphasis recognizes the role of knowledge-based technologies as supporting knowledge processes in human society, substituting for them only when appropriate, and even then giving easy access to the process if required. Hawkin's (1983) studies of the behavior of experts and their clients, and the management literature on professional development (Schön 1983), suggest that expert-client interaction is a process of negotiation, modeling, and continuing acquisition, rather than authoritarian decision-making based on a static knowledge-base. It is this dynamic process of knowledge exploration which is our target for an integrated system.

This emphasis on visualization and exploration is not in conflict with the use of the system to make 'quick and dirty' decisions with no exploration and no explanation. Quite the contrary—it is the potential for full exploratory visualization which allows users to trust in more routine applications of the system, knowing that a full audit trail of all the knowledge processes is always open to them. In particular, meaningful visualization is a powerful technique for highlighting situations with which the system is not designed to cope. Such anomaly detection is important not only in making available to clients meta-information about the suitability of their problem for solution by the system, but also in closing the inductive knowledge acquisition loop back to the expert enabling him or her to examine novel situations that might otherwise go undetected (Gaines 1988).

The central core about which our thinking revolves may best be characterized as one of *validation.* Not just the validation of the performance system in the quality of its ultimate operation, but the ongoing validation of the knowledge base at all stages and in all forms. The internal cross-consistency checks of Figure 14 are a form of validation. The visual relationships in Figures 5 through 7, 11, and 15 through 17, all support direct human validation of the knowledge base as simply and naturally as possible. Our requirement to be able to move the knowledge base back and forth freely between the acquisition and performance tools is precisely to support a continuous knowledge validation process. Knowledge base management requires the careful methodologies developed for data base management supplemented by techniques that recognize and support the volatility, fallibility, vagueness and inconsistencies of knowledge.

Figure 18 shows what we are trying to achieve at the next stage in the forms and relations of knowledge that will be supported in the combined tool. All levels and forms of knowledge exist in some form in NEXTRA and NEXPERT systems currently. The task is to achieve as seamless an integration as possible with the only discontinuities being those intrinsic to knowledge, not those introduced by the tools.

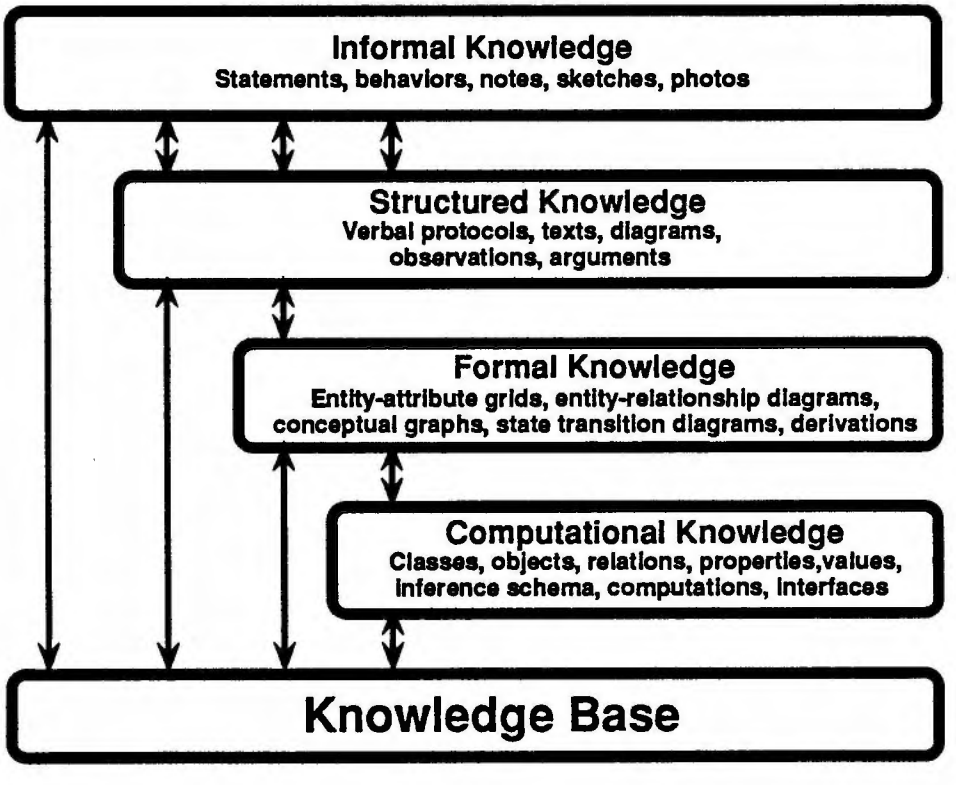

Fig.17 Trajectory of evolving knowledge in the graphic domain

CONCLUSIONS—TOWARDS A KNOWLEDGE SUPPORT TOOLBOX

We see the integration the techniques described in this paper not only as powerful new tools for the acquisition of knowledge but also as the initial approach to building of a toolbox of technologies including relational and other databases, hypertext, knowledge structuring tools, CD-I based and on-line information systems, all managed by the AI performance system, and incorporating integral knowledge acquisition.

This integration of technologies is both a natural direction of development in knowledge-based systems and a necessary step towards supporting the full range of knowledge processes in human society.

Neither the computational nor the cognitive foundations of knowledge-based systems are sufficiently advanced for it to be possible to give strong normative guidelines for the design of the type of integrated system discussed in this paper. The only approach possible is an empirical one, founded on the available knowledge in AI, guided by intuition, and leading to the production of tools that can be tested in real-world applications. NEXTRA and NEXPERT *OBJECT* are now being used by a very diverse community, and it is the experience of these users that will determine whether we are succeeding in providing an effective toolbox supporting the development and application of knowledge-based systems.

ACKNOWLEDGEMENTS

We are grateful to many colleagues for discussions, particularly at the Knowledge Acquisition Workshops, that have influenced this paper. In particular we would like to thank John Boose, Jeff Bradshaw, and Mildred Shaw, for access to their own research and many stimulating discussions.

REFERENCES

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

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** (1), 3-28 (January).

Cendrowska, J. (1987) An algorithm for inducing modular rules. **International Journal of Man-Machine Studies 27** (4), 349-370 (October).

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

Gaines, B.R. (1988). Positive feedback processes underlying the formation of expertise. **IEEE Transactions on Systems, Man & Cybernetics,** to appear.

Gaines, B.R. & Shaw, M.L.G. (1986). Induction of inference rules for expert systems. **Fuzzy Sets and Systems,** 8(3), 315-328 (April).

Hawkins, D. (1983). An analysis of expert thinking. **International Journal of Man-Machine Studies, 18**(1), 1-47 (January).

Kelly, G.A. (1955). **The Psychology of Personal Constructs.** New York: Norton.

Quinlan, J.R. (1987) Simplifying decision trees. **International Journal of Man-Machine Studies 27** (3), 221-234 (September).

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).

Schön, D.A. (1983). **The Reflective Practitioner.** New York: Basic Books.

Shaw, M.L.G. (1980). **On Becoming a Personal Scientist: Interactive Computer Elicitation of Personal Models of the World.** 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). Interactive elicitation of knowledge from experts. **Future Computing Systems, 1**(2), 151-190.

Shaw, M.L.G. & Gaines, B.R. (1987a). An interactive knowledge elicitation technique using personal construct technology. Kidd, A., Ed. **Knowledge Eliciation for Expert Systems: A Practical Handbook.** pp.109-136. Plenum Press.

Shaw, M.L.G. & Gaines, B.R. (1987b) 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.** Banff (November).

Shriver, B. & Wegner, P. (1987). **Research Directions in Object-Oriented Programming.** Cambridge, Masachusetts: MIT Press.

Slater, P., Ed. (1976). **Dimensions of Intrapersonal Space: Volume 1.** London: John Wiley.