Mediating Representations for Knowledge Acquisition

Jeffrey M. Bradshaw & John H. Boose
Computer Science Organization, Boeing Computer Services
P.O. Box 24346, M/S 7L-64, Seattle, Washington 98124 USA (206) 865-3422;
jbrad@atc.boeing.com

Abstract

Knowledge acquisition is a constructive modeling process, not simply a matter of "expertise transfer." Consistent with this perspective, we advocate knowledge acquisition practices and tools that facilitate active collaboration between expert and knowledge engineer and that support knowledge-based system development from a life cycle perspective. Serious problems of modeling can often be traced directly to the inadequacies of the particular knowledge representations used at a given stage of development. To counter these problems, we propose an approach to knowledge representation for knowledge modeling that distinguishes between external, conceptual, and internal schemata. As an implementation of the external schemata, we emphasize the use of *mediating representations* which serve as a means of communication between expert and knowledge engineer. *Intermediate representations* implement the conceptual schema, and help bridge the gap between the mediating representations and a particular implementation formalism. DDUCKS is described as an example of an "open architecture" constructivist knowledge modeling environment embracing the modeling perspective and built around collections of mediating and intermediate representations. The architecture of DDUCKS facilitates reuse and tailoring of models and tools. We conclude with a discussion of the issues in design and evaluation of mediating representations.

1. Introduction: Model-based Knowledge Acquisition

Recent work in knowledge acquisition has emphasized that the creation of knowledge bases is a constructive modeling process, and not simply a matter of "expertise transfer" or "design knowledge capture" (e.g., Bradshaw & Boose, 1990; Clancey, 1990; Cox, 1991; Ford, Bradshaw & Adams-Webber, 1992; Gruber, 1990; Musen, 1989; Shaw, Woodward & Gaines, 1990; Wielinga, Akkermans, Schreiber & Balder, 1989). For this reason, use of the term *knowledge modeling* is beginning to replace the term *knowledge acquisition* to describe activities in this field.

Modeling is purposive, that is, to be involved in modeling is necessarily to be engaged in using the model in some particular setting for particular reasons that determine what should be modeled, how to model it, and what can be ignored (Thimbleby, 1990; Winograd & Flores, 1986). Together, the criteria of purpose and cost-effectiveness determine how additional pragmatic issues should be resolved such as who the users of the model are, how it ought to be presented in order to be both usable and useful, and how it will be maintained over its projected lifetime (Rothenberg, 1989). This modeling perspective implies the need for active participation by both experts and knowledge engineers in the creation of knowledge bases.

Submitted to the 1992 AAAI Spring Symposium on Reasoning with Diagrammatic Representations,
Stanford University, March 24-27, 1992.

In a model-based approach to knowledge acquisition, experts and knowledge engineers *cooperatively* build models comprising explicit representations of the processes and concepts in the domain. We are not implying that the model will necessarily be the basis of the performance system (as would be the case in a model-based reasoning system), but rather that it can constitute a rich description of a problem domain that is initially *independent* of a particular implementation formalism.

The view of knowledge acquisition as a modeling activity runs counter to the idea that the process consists of "mining those jewels of knowledge out of [the experts'] heads one by one" (Feigenbaum & McCorduck, 1983, p.2). Such a perspective erroneously assumes that there exists some 'gold standard' of knowledge and that a domain expert is one who has captured a discrete (presumably large) part of the 'reality' underlying observed events in the relevant domain. This 'mining analogy' is both fundamentally at odds with cognitive science theory (see e.g., Agnew & Brown, 1989a, b), and dangerously misleading as a metaphor for the guidance of knowledge acquisition practice. Expertise is not a natural resource that can be extracted, harvested, transferred, or captured. Experts involved in knowledge acquisition are not restating a coherent body of knowledge that already exists in their minds; rather they are engaged in a constructive modeling process, in the context of which formal representations are newly created and shaped (Clancey, 1989; Ford, Bradshaw & Adams-Webber, 1992).

From a constructivist perspective, a model is not a 'picture' of the problem, but rather a device for the attainment or formulation of knowledge about it (Kaplan, 1963). Often, the most important outcome of a knowledge acquisition project is not the resulting knowledge-based system, but rather the insights gained through the process of articulating, structuring, and critically evaluating the underlying model (Moore & Agogino, 1987). From this, we infer that the value of the knowledge acquisition effort may derive not simply from a final 'correct' representation of the problem, but additionally from our success in framing the activity as a self-correcting enterprise that can subject any part of the model to critical scrutiny, including our background assumptions. From this standpoint, the crucial question for knowledge engineers is not "How do we know the model is correct?" (every model is an incorrect oversimplification); but rather "How useful is the model (and the modeling process) in facilitating our understanding of the domain?"

Our understanding of models and the modeling process entails a life cycle perspective on knowledge acquisition. Knowledge modeling does not culminate at some arbitrary point in development, but rather extends throughout the life of the system. It follows that knowledge modeling tools must support the gradual evolution of the model through numerous cycles of refinement.

Each phase of development activity imposes its own requirements and difficulties. Serious problems of knowledge modeling can often be traced directly to the inadequacies of the particular knowledge representations used at a given stage of development. Many tools are limited in both their repertoire of modeling representations and their support for evolution and transformation of representations. The ideal knowledge modeling tool would support a smooth transition of the model from an easily communicated, relatively unconstrained statement of the problem to an unambiguous specification of design. A number of changes in representation may be required to accompany successive stages in model construction: from mental models to increasingly refined conceptual models via elicitation and analysis techniques, and eventually, from these highly elaborated models to an operational knowledge base via formalization and implementation procedures (Shaw & Woodward, 1989; Shaw, Woodward & Gaines, 1990; Figure 1).

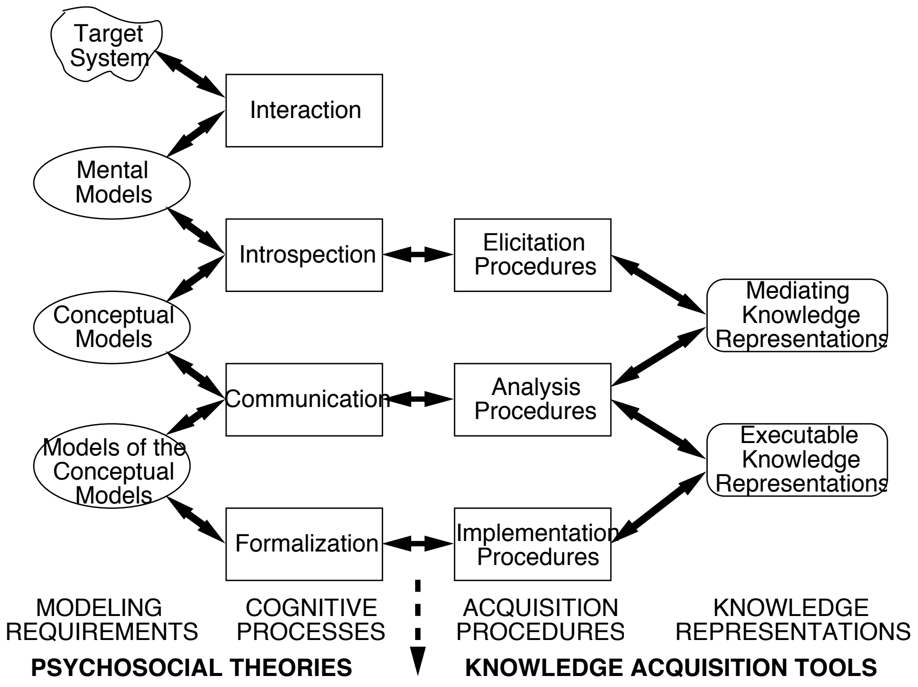

Figure 1. The ideal knowledge acquisition tool would support a smooth transition of the model through different phases of the knowledge engineering process (Figure adapted from Shaw & Woodward, 1989).

Unfortunately, the emphasis given to rapid prototyping in traditional accounts of knowledge acquisition, along with the faulty notion that 'the production of working code is the most important result of work done', often leads to the premature encoding of knowledge in an implementation formalism associated with a specific performance environment (Bradshaw & Boose, 1989; Figure 2). The unfortunate result is that no independent description of the model will exist other than the rule base itself and possibly some glossaries in the help information of the system (Johnson, 1989).

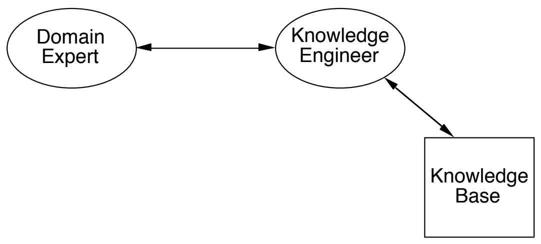

Figure 2. Depiction of traditional view of knowledge acquisition.

The problems of premature encoding of knowledge in implementation-driven representations have spurred efforts to develop other representations that more adequately support the early stages of conceptual modeling. We call these *mediating representations*.

2. The Role of Mediating Representations

Mediating representations (e.g., repertory grids, network diagrams) are designed to reduce the problem of representation mismatch, the disparity between a person's natural description of the problem and its representation in some computable medium (Gruber, 1989). They provide a bridge between verbal data and typical knowledge representation schemes such as production rules (Johnson, 1989; Bradshaw, Ford, & Adams-Webber, 1992). Work on mediating representations for knowledge modeling parallels work on visual programming languages for software engineers (e.g., Chang, 1990; Ichikawa & Chang, 1990).

The term *mediating representation* has various interpretations in the literature, however we take it to "convey the sense of… coming to understand through the representation" (Johnson, 1989, p. 184). A crucial feature is that mediating representations should be "easily readable by those who were not involved in the original development programme…" (Diaper, 1989, p. 34). This is essential, since executable knowledge bases are seldom organized for direct use by humans, but instead for the convenience of the reasoning mechanisms of the performance environment. The design of a mediating representation, on the other hand, should be optimized for human understanding rather than machine efficiency.

Effective mediating representations make important things explicit and hide unnecessary detail. They expose natural constraints, facilitate computation, and are complete and concise (Winston, 1984). The choice of representation can have an enormous effect on human problem solving performance (e.g., Larkin & Simon, 1987). As an example, consider that we can represent numbers as Arabic numerals, Roman numerals, or as bits in computer memory. While each of these forms are logically equivalent, they are not the same in a practical sense. It is much more efficient for a computer to multiply numbers represented as bits than as numeric symbols. Similarly, from a human perspective, it is easier to do multiplication with Arabic numerals than with Roman numerals or binary numbers. Narayanan, Chandrasekaran, Iwasaki and Simon (1991) state that: "The power of diagrammatic representations stems from the property that they allow the explicit representation and direct retrieval of information that can be represented only implicitly in other types of representations and then has to be computed, sometimes at great cost, to make it explicit for use."

Wielinga, Akkermans, Schreiber & Balder (1989) present a number of compelling arguments for making a clear distinction between knowledge-level conceptual models and implementation-focused design models in the knowledge-based system development process (see also Schreiber, Akkermans & Wielinga, 1990). Sowa (1987) has likewise argued that considerations of human efficiency far outweigh considerations of machine efficiency for complex modeling problems.

He observes that, since the time used for transformation from one knowledge representation to another is linearly proportional to the length of the formula, a change of notation increases efficiency by constant factor:

> “But if a problem is truly NP hard, a change of notation will not make it solvable in polynomial time. Yet, because one notation for logic can always be translated into another, it is possible to do the input and output in a form that has the best human factors, and than translate it to the form that has the fastest theorem prover. Although the issues of machine efficiency are not irrelevant, for knowledge representation and acquisition, they are less important than naturalness and ease of use…” (see also Sowa, 1991)

Work on mediating representations aims to improve the knowledge modeling process by developing and improving representational devices available to the expert and knowledge engineer (Ford, Bradshaw & Adams-Webber, 1991). A mediating representation provides an explicit external medium in which experts can build a model (Figure 3). Whatever structure is inherent in the form of representation provides grounding constraints that both facilitate expression within the set of conventions adopted, and enforce comprehensibility and consistency by preempting certain possibilities. The mutual development of permanent “cognitive artifact” supplementing the exchange of information between participants promotes and enriches communication, leading gradually to a shared understanding of the emerging conceptual model of the domain (Norman, 1988, 1991). In this way, mediating representations enable domain experts and knowledge engineers to cooperatively build problem solving models. In the later stages of system development, mediating representations may also facilitate maintenance and explanation by enabling both knowledge engineers and the people eventually using the system to explore the conceptual domain model without resorting to low-level representations (e.g., C code, Lisp, rules).

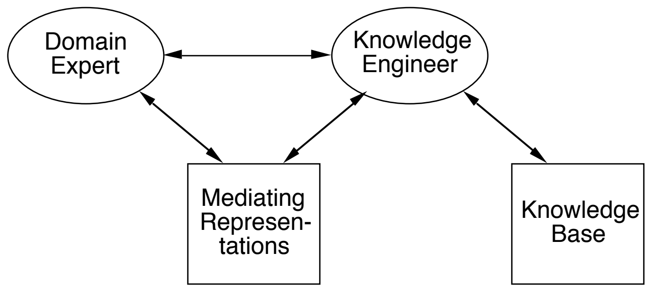

Figure 3. Mediating representations facilitate communication between domain expert and knowledge engineer.

A number of automated knowledge modeling tools have incorporated effective mediating representations. These tools tend to adopt one of two approaches. Either they contain interfaces that bear a close resemblance in appearance and procedure to the original manual task—for example, cancer-therapy protocol forms in OPAL (Musen, 1989) and engineering notebooks in vmacs (Sivard, Zweben, Cannon, Lakin & Leifer, 1989), or they rely on some easily-learned, generic knowledge representation form—for example, repertory grids and directed graphs (Boose & Bradshaw, 1987; Eisenstadt, Domingue, Rajan & Motta, 1990; Ford, Stahl, Adams-Webber, Cañas, Novak & Jones, 1991; Gaines & Shaw, 1986a)

3. The Role of Intermediate Representations

Since modeling is a process of iterative refinement, we would like to be able to map back and forth from the kinds of representations used in performance environments to mediating representations that are more useful for communication purposes. For example, initial approaches to knowledge acquisition in ETS (Boose, 1984, 1986a), KSS0 (Gaines & Shaw, 1986b), and *Nicod* (Ford, 1987; Ford, Petry, Adams-Webber & Chang, 1991) embodied procedures for transformation from repertory grids to rules. This was found to be a useful and productive step for knowledge engineers, particularly in the early prototyping phases of a project. Some kinds of information, however, could not be conveniently represented in simple repertory grids. Furthermore, this was essentially a one-way procedure—while the kinds of knowledge available in repertory grids could be transformed to rule sets, in most cases there was no natural mapping from rules back to grids.

Over time the semantic gap between modeling systems and performance systems has widened dramatically. A distinguishing characteristic of some of the newer knowledge modeling tools is the degree to which they promote the use of multiple perspectives on the same information. They also exemplify the push toward informal textual, graphical, and multimedia forms of knowledge representation (Boy, 1991; Eisenstadt, Domingue, Rajan & Motta, 1990; Gaines & Boose, 1991). As new mediating representations have increased the richness, complexity, and subtlety of the knowledge elicited by automated knowledge modeling tools, a requirement has emerged for *intermediate representations*. Intermediate representations can integrate the diverse perspectives presented by the mediating representations. They help bridge the gulf between human participants and the implementation formalism required by the performance environment. In addition, intermediate representations facilitate the integration of knowledge modeling and performance systems, allowing rapid feedback throughout the process of system development (e.g., Shema & Boose, 1988; Linster & Gaines, 1990).

Figure 4 outlines a three-schemata architecture for knowledge modeling tools, with mediating representations as external schemata, the intermediate representation corresponding to a conceptual schema, and the knowledge base or database as an internal schema. The external schemata are optimized for communication, the conceptual schema for semantic completeness, and the internal schema for performance. Obvious similarities will be seen between our suggested architecture for knowledge modeling tools and the proposed ANSI-SPARC three-schema model for data management. The definitions for the three schemata given by van Griethuysen and King (1985) provide a good summary of this perspective:

> "The… *conceptual schema* controls what is described in the information base. The conceptual schema controls the semantic meaning of all representations, that is, defines the set of checking, generating, and deducing procedures of the information at the conceptual level in the information system.
>
> The *external schemata* describe how the users wish to have the information represented. The external processor interfaces directly with the users and coordinates their information exchange.

> The *internal schema* describes the internal physical representation of the information… The mapping between the external schemata and the internal schema must preserve meaning as defined by the conceptual schema."

This approach allows views containing mediating representations to be coupled to the underlying intermediate representation so that any changes made to one view may be immediately reflected in all related views. Knowledge analysis and performance tools may be similarly designed to exploit the integration of information at the intermediate level.

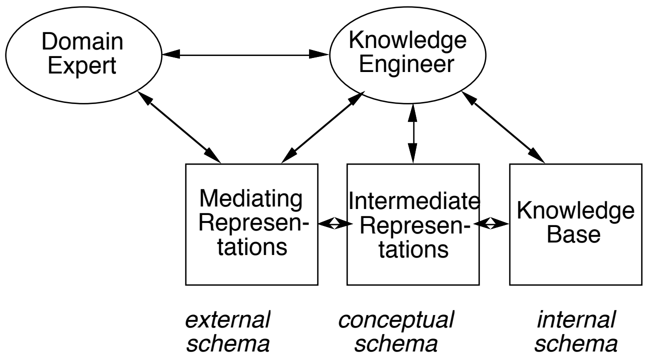

Figure 4. Three-schema architecture for knowledge acquisition tools

An intermediate knowledge representation is one "which only exists between flanking representations and is bound to them by clearly defined projection rules which map one representation to the next" (Johnson, 1989, p. 184). The issue of mapping between representations is a troublesome one. For one thing, it is obvious that much of what can be modeled in mediating representations cannot be directly incorporated into the current commercial performance systems. Furthermore, since every transformation of knowledge is a reconstruction of that knowledge, we know that, even if logical equivalence as part of representational mapping is assured, we cannot assume practical or even conceptual equivalence. For these and other reasons, automated mapping between representations will continue to be an issue, and some amount of manual mapping from one representation to another will remain common practice. Whether mapping is automatically assisted or manual, informal or formal, does not obviate the need for integrative, intermediate representations that are relatively independent of the constraints of the delivery environment.

4. DDUCKS: Decision and Design Utilities for Comprehensive Knowledge Support

Early knowledge-based tools generally made strong assumptions about their operating environment. At first, dedicated, stand-alone applications were the rule. Over time, as the value of adding "hooks" for access to external applications and data was realized, most knowledge-based tools still operated under the assumption that they were in ultimate control of the system as the highest level executive. Currently, the greatest potential for use of knowledge-based systems is in areas requiring close interaction with traditional software applications and data. An application that assumes it is in ultimate control will be ineffective in such environments.

Brodie (1989) has discussed the need for intelligent interoperability in information systems. He defines the term to mean intelligent cooperation among systems to optimally achieve specified goals. While there is little disagreement that future computing environments will consist of multiple heterogeneous software systems running on multiple heterogeneous machines, most knowledge-based systems are disjoint: they do not communicate.

To facilitate intelligent interoperability between knowledge acquisition tools and other software, we are in the process of implementing an "open architecture" integrating environment that allows for a high degree of connectivity among knowledge modeling tools and commercial software (Figure 5; Bradshaw, Covington, Russo & Boose, 1990, 1991). This environment is called DDUCKS (Decision and Design Utilities for Comprehensive Knowledge Support)[1].. Previously, individual components of DDUCKS have been applied to prototype a number of specific applications. As part of a Boeing project called Design of Information Systems or DIS (Benda, 1990), we are exploring how knowledge modeling and decision support tools can work cooperatively with one another and in conjunction with commercial applications such as spreadsheets, databases, or hypermedia software, within a computer-supported meeting environment (Boose, Bradshaw, Koszarek & Shema, 1992).

DART. DART (Design Alternatives Rationale Tradeoffs) is a repertory-grid-based knowledge acquisition tool. It was originally developed with funding from NASA as part of an effort to capture design knowledge for the Space Station Freedom program (Boose, Shema & Bradshaw, 1990). Similar tools and concepts have been under development at The Boeing Company for many years (Boose, 1984; Boose & Bradshaw, 1987; Boose, Bradshaw, Kitto & Shema, 1989; Boose, Shema & Bradshaw, 1989). DART contains a number of elicitation, analysis, representation, and inference methods derived from personal construct theory (Kelly, 1955).

Canard. *Canard* is a knowledge acquisition tool that can be used to generate and structure complex alternatives (Bradshaw, Boose, Covington & Russo, 1989; Shema, Bradshaw, Covington, and Boose, 1990; Bradshaw, Shema, & Boose, 1992). Links are maintained between the tables and underlying repertory grids. The possibility table representation is based on the manually developed strategy tables (McNamee & Celona, 1987) and morphological charts (Zwicky, 1969) that have been used by decision analysts and designers for many years. *Canard* automates this representation and extends its logic and structure to allow knowledge-based inference and the representation of more complex problems (e.g., hierarchical tables, explicit representation of constraints).

[1] Either the first or second *D* in *DDUCKS* is silent, depending on whether one is using it for decisions or design.

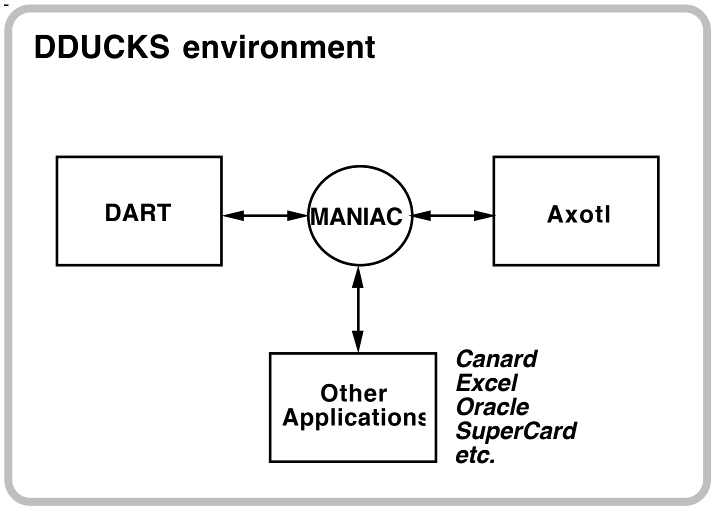

Figure 5. DDUCKS integrates components from *Axotl*, DART, and other applications.

Axotl. *Axotl* combines a influence-diagram-based decision analysis workbench with knowledge-based tools to assist individuals consulting with decisions involving high stakes, difficult tradeoffs, or critical uncertainties and risks (Bradshaw, Covington, Russo, & Boose, 1990, 1991). The knowledge-based tools in *Axotl* can be configured with application-independent knowledge (i.e., knowledge of decision analysis tools and methodology) and application-specific knowledge (i.e., knowledge about a particular domain) to provide guidance and help during a consultation. The relationship between personal construct theory and decision theory is discussed in Bradshaw & Boose (1990). An additional set of tools (which go by the name of *eQuality)* assist users in modeling and executing the decision making process, and in rigorously modeling and analyzing the entities and processes involved in the enterprise of which the decision is a part (Bradshaw, Holm, Kipersztok, Nguyen & Covington, 1992). *eQuality* is being implemented in a new version of the framework called *Axotl II*.

MANIAC. MANIAC (MANager for InterApplication Communication) supports asynchronous and synchronous communication between any number of multitasking applications. Plans for coordination among applications are modeled and executed using knowledge-based capabilities in *Axotl*, while MANIAC provides the infrastructure for the actual message passing. Originally implemented as a driver in the 6.x version of the Macintosh operating system, MANIAC is currently being updated to take advantage of new interapplication communication features in version 7.0. Future versions may support additional hardware and software platforms.

In the following section, we will explain how DDUCKS provides an architecture for *reuse* and *tailoring*. Next, we will discuss mediating and intermediate representations used in DDUCKS, and an organization tool called the *project notebook*. Then we will briefly describe *applications* of DDUCKS within The Boeing Company.

4.1. An Architecture for Reuse and Tailoring

Because building knowledge modeling tools is labor intensive, their development can usually be justified only if they can be easily applied to more than a single application. Knowledge modeling tool developers interested in deriving the most benefit from their tools may look for areas consisting of several problems that can each be characterized by a general task model (Boose, 1989; Klinker, 1989). Knowledge modeling tools can then be created that both fit the general task model and are tailorable to several specific problems.

Musen (1989) was one of the first to present an explicit, general approach to creating tailorable knowledge modeling tools. Knowledge modeling tools are tailored using a meta-level knowledge modeling tool to edit a domain-independent conceptual model (see also Puerta, Egar, Tu & Musen, 1991). The meta-level tool, PROTOGE, provides a system to generate knowledge editors tailored for various classes of treatment plans. Physician experts can then use the knowledge editors created by PROTEGE to develop knowledge bases (e.g., OPAL) that encode specific treatment plans in their medical specialty; the resulting systems (e.g., ONCOCIN) could then be used in turn by attending physicians to obtain therapy recommendations for a particular patient.

Besides the reuse of task models, a number of researchers have also emphasized the importance of a common ontology in achieving sharable, reusable knowledge bases (e.g., Gruber, 1991; Lenat & Guha, 1990; Neches, Fikes, Finin, Gruber, Patil, Senator & Swartout, 1991; Skuce, 1991a). Alexander, Freiling, Shulman, Rehfuss, and Messick (1988) introduced ontological analysis as a knowledge modeling technique for the preliminary analysis of a problem-solving domain (see also Wielinga, Schreiber & Breuker, 1991). As one of the first steps in knowledge modeling, we carry out a knowledge level conceptual analysis of the domain, which consists of building a rich model of static, dynamic, and epistemic knowledge. The initial conceptual model produced by this analysis can be extended by designers and users of the system and applied directly to problem-solving as described below.

We aim to maximize reusability and tailoring of models and tools by generalizing Musen's approach. It is useful to think of DDUCKS in terms of four layers of functionality: workbench, shell, application, and consultation (Figure 6).

The workbench consists of five major elements:

- methodology-independent problem-solving task models (e.g., heuristic classification, constraint satisfaction);
- generic interaction paradigms (see section 4.2 below; e.g., graph view, matrix view, various widgets);
- a methodology-independent ontology (a specification of the abstract schema; e.g., generic object types such as entity, relationship);
- application-configuration process models (i.e., model of how to configure the workbench for a particular application such as process management, decision support, or design);
- a standard library of inference types and functions (e.g., mathematical and logical mechanisms that implement problem-solving, analysis, or simulation procedures).

An instance of a shell, created by using the knowledge modeling facilities generated by the workbench, may contain:

- methodology-specific problem-solving task models (e.g., maximization of expected utility across decision alternatives, hierarchical constraint satisfaction using extended AND-OR graphs, process optimization through event-based simulation)
- methodology-specific mediating representations created out of the combination of generic interaction paradigms with a particular semantic and possibly computational interpretation of the elements (e.g., process views, influence diagrams, repertory grids);
- a methodology-specific ontology (a specification of the schema itself; e.g., activities, performers; decision and chance nodes; elements and constructs);
- methodology-specific model-building process models (i.e., knowledge about how to acquire application-specific knowledge within the context of a methodology);
- methodology-specific extensions to the inference and function library.

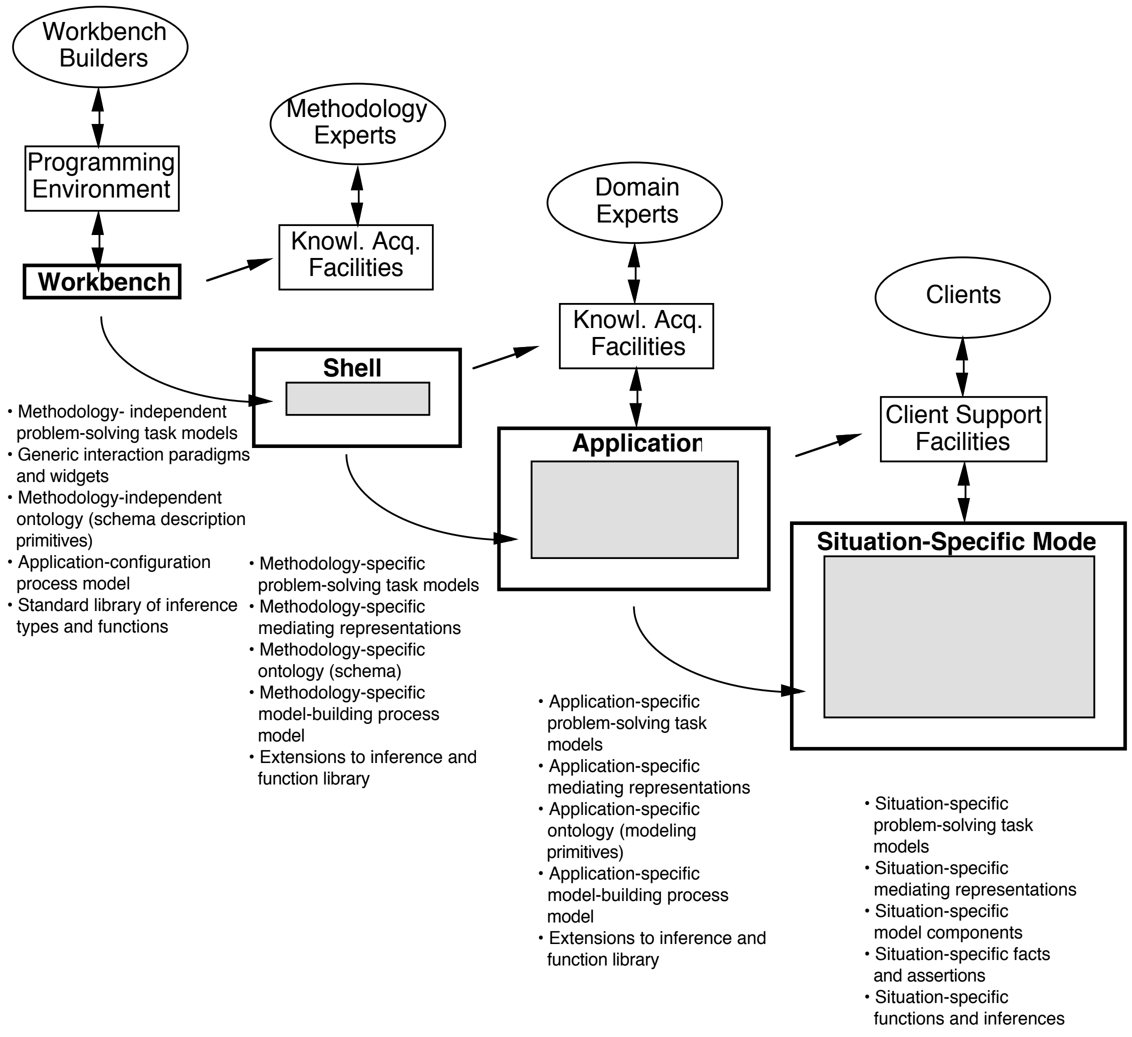
Workbench Builders
Programming Environment
Workbench
Methodology Experts
Knowl. Acq. Facilities
Shell
Domain Experts
Knowl. Acq. Facilities
Application
Clients
Client Support Facilities
Situation-Specific Mode
• Methodology- independent problem-solving task models
• Generic interaction paradigms and widgets
• Methodology-independent ontology (schema description primitives)
• Application-configuration process model
• Standard library of inference types and functions
• Methodology-specific problem-solving task models
• Methodology-specific mediating representations
• Methodology-specific ontology (schema)
• Methodology-specific model-building process model
• Extensions to inference and function library
• Application-specific problem-solving task models
• Application-specific mediating representations
• Application-specific ontology (modeling primitives)
• Application-specific model-building process model
• Extensions to inference and function library
• Situation-specific problem-solving task models
• Situation-specific mediating representations
• Situation-specific model components
• Situation-specific facts and assertions
• Situation-specific functions and inferences

Figure 6. Layers of functionality facilitate reusability (inspired by figure from Musen, 1989).

An instance of an application, created by using the knowledge modeling facilities generated by the shell, may contain:

- application-specific problem-solving task models;
- application-specific mediating representations (e.g., form-filling interfaces tailored to R&D investment decision makers, engineering process modelers, or space station designers that may be used in place of influence diagrams, generic process views, or grids);
- an application-specific ontology (extensions to the schema that become the modeling primitives for the application; e.g., go/no-go investment decision nodes, technical risk chance nodes; airplane design-build activities; alternatives and criteria);
- application-specific model-building process models (i.e., knowledge about how to conduct a consultation with clients such as R&D investment decision makers, airplane design-build process improvement team members, or space station designers);
- application-specific extensions to the inference and function library.

An instance of a consultation, created by using the consultation facilities generated by the application, may contain:

- situation-specific problem-solving task models (e.g., a model for a particular business, design, or decision-making process).
- situation-specific mediating representations (e.g., text and graphical annotation of views on the model);
- situation-specific model components (e.g., decision and chance nodes for a particular project decision model; activity and entity instances for a particular enterprise model; alternatives and criteria for a particular design decision);
- Situation-specific facts and assertions (e.g., particular information about a situation);
- situation-specific functions and inferences.

The complete situation-specific model represents the unique characteristics of a particular problem and comprises all the information mentioned above. This model is formulated, evaluated, and analyzed during the consultation to produce recommendations for action or for further model refinement.

4.2. Interaction Paradigms Related to Mediating and Intermediate Representations

Figure 7 is a view of knowledge representation in DDUCKS . The intermediate representation (i.e., enterprise model) consists of entities, relationships, and situations as the primary concepts, and domains, properties, and constraints as secondary concepts. We are evaluating a version of *Axotl II* that uses CODE version 4 as the underlying semantic representation language (Skuce, 1991b, c; Lethbridge, 1991). We have derived our general taxonomy for conceptual modeling from Tauzovich and Skuce (1990), with extensions for dynamic and epistemic aspects of the model. CODE provides a rich, paradigm for the definition of knowledge level concepts. These concepts are arranged in an inheritance network using a flexible inheritance mechanism. The emphasis in CODE is on providing tools to support the important and frequently overlooked aspects of conceptual, ontological, and terminological analysis. Several associated semantic subsystems, such as a first order logic system and a simple natural language system, allow various types of syntactic and semantic checks to be performed, if desired. A comprehensive lexicon allows references to concepts to be automatically maintained and quickly accessed. We have found the rich semantic representation of terms and concepts to be of great importance throughout the life of a project.

User-interface management systems (UIMS) are becoming an essential part of interactive tool development and end-user tailoring (Hix, 1990). We are extending the capabilities of a Smalltalk-80-based direct-manipulation user-interface builder to form the building blocks for the views in DDUCKS (Laland, Novotny, Enzer & Bortz, 1991). The tools in DDUCKS rely on the Smalltalk-80 MVC (model-view-controller) concept for managing consistency among views (Goldberg, 1990; Krasner & Pope, 1988; Adams, 1988b). The MVC approach provides a way to effectively factor out the data in an underlying model from the data in dependent views, so that changes to the model in one view are immediately reflected in all related views. Class hierarchy mechanisms in Smalltalk-80 allow generic views of a certain sort to be easily specialized for different purposes. This, in conjunction with the DDUCKS UIMS, has allowed us to define many different views on similar aspects of the model, as well as several similar views on different aspects of the model.

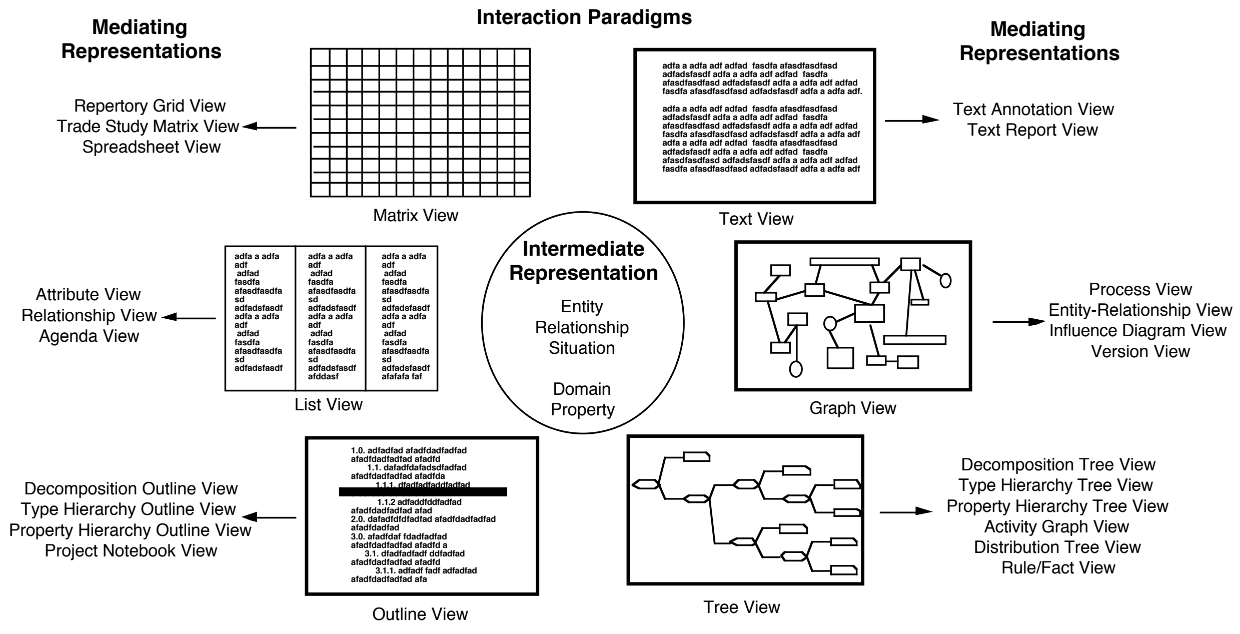

Figure 7. The intermediate representation in DDUCKS, surrounded by examples of generic interaction paradigms, and mediating representations.

The six views surrounding the intermediate representation correspond to the generic user-interface interaction paradigms that are implemented as abstract “pluggable” view classes (Krasner & Pope, 1988; Adams, 1988a, b). These views are generic in the sense that they define the graphical form for the representation, but the form has no underlying semantics. Within DDUCKS, various configurations of these interaction paradigms can be called up as *sketchpad views* to record free-form graphical and textual information. For example, individuals and groups can capture back-of-the-envelope drawings, agendas, issues, action items, requirements, and other information pertinent to their task. While not part of the formal model, users can link elements within sketchpad views to elements in other views in hypertext fashion.

By combining one or more of these generic interaction paradigms with a semantics defined in the intermediate representation and (for some representations) the problem-solving method, methodology-specific or application-specific mediating representations are defined. Mappings are defined between graphical actions in the model views and operations on logical entities, relationships, and properties in the enterprise model. For example, influence diagrams combine a graph view with the concepts of decision, chance, and value nodes and the problem-solving method of maximization of expected utility across decision alternatives. Trade study matrices (a methodology-specific kind of repertory grid) are built out of a matrix view, the concepts of alternatives, criteria, and ratings, and a heuristic classification problem-solving method. Process views combine a graph view with the a formal definition of activities and relationships between them. Type definition views allow the users to extend the ontology. We call all these representations *model views,* because they portray different perspectives on the formal enterprise model. By virtue of the Smalltalk-80 model-view-controller paradigm, consistency is continuously maintained for all model views portraying the same version of the enterprise model.

The volume and diversity of information that can be represented in DDUCKS drives a requirement for ways to manage, organize, and link that information. A *virtual project notebook* helps team members collect and organize the diverse materials associated with a particular knowledge acquisition project. It also helps manage changes between different versions and views of the model as it evolves. The project notebook can assist in planning and modeling activities throughout the life of the project. Using project notebook *templates,* groups can tailor the contents of the boiler plate project notebook to be consistent with their own preferences for accessing, viewing, and using the information. For example, a process improvement team's blank notebook can come pre-configured with information about organizational standards (e.g., required entity and icon types, reporting forms) and procedures (e.g., required steps in a project plan), just as a real notebook could be pre-loaded with labeled dividers and forms. In addition to its obvious use in managing information about the model, the project notebook supports the team as a simple computer-supported meeting facilitation tool and as a form of group memory.

4.3. Applications

Various combinations of DDUCKS components have been evaluated with respect to several different kinds of problems. These have included:

- rapid prototyping for expert systems (Boose, 1984; 1986a);
- analysis and conflict resolution of knowledge from multiple experts (Boose, 1986b);
- knowledge acquisition and delivery of knowledge-based systems for heuristic classification problems (Boose & Bradshaw, 1987; Boose, Shema & Bradshaw, 1989);

- acquiring and verifying control knowledge for a blackboard system (Baum, Shema, Boose & Bradshaw, 1989);
- evaluation of training course effectiveness (Schuler, Russo, Boose, & Bradshaw, 1990);
- assisting R&D managers to make project investment decisions involving substantial uncertainty, risk, and complex tradeoffs (Bradshaw, Covington, Russo & Boose, 1990, 1991);
- design knowledge capture for a Corporate Memory Facility for NASA's Space Station Freedom (Boose, Shema & Bradshaw, 1990; Bradshaw, Boose & Shema, 1992);
- alternative generation and constraint management for synthesis and design problems (Bradshaw, Boose, Covington & Russo, 1989; Shema, Bradshaw, Covington & Boose, 1990; Bradshaw, Shema & Boose, 1992); and
- documentation and streamlining of business processes (Bradshaw, Holm, Kipersztok, Nguyen & Covington, 1992).

As part of the DIS project, we are implementing enhancements to these individual tools, and developing and evaluating a methodology for their joint use in an electronic meeting room setting (Boose, Bradshaw, Koszarek & Shema, 1992).

5. Discussion: Design and Evaluation of Mediating Representations

Some of the most important unresolved issues about mediating representations concern how they should be designed and evaluated (Tortora, 1990). Most of the past work in this vein has been guided by intuition rather than principle, and evaluated by anecdote rather than empirical analysis. While some amount of this is unavoidable (and in fact desirable), we must do more to develop a theory-based mediating representation design methodology (see e.g., Carroll, Kellogg & Rosson's (1991) discussion of the *task-artifact cycle).* Casner and Larkin (1989), for example, have begun to apply recent research in how representation affects problem solving to guide the definition of a principled methodology for designing effective perceptual codes and interpretations that support particular kinds of tasks. Criteria, derived from Johnson (1989, p. 185) and Winston (1984) also suggest general rules-of-thumb for evaluating the effectiveness of a representation:

- Is the formalism sufficiently expressive?
- Does the formalism aid communication between the members of the development team?
- Does the formalism actually guide knowledge analysis in a significant way?
- Does it make the important things explicit, suppressing detail and keeping rarely used information out of sight, but still available when necessary?
- Does it expose natural constraints?
- Is it complete and concise, efficiently saying all that needs to be said?

If we could design a representation performed well with respect to these 'acquirability' criteria above that was also endowed with with Turing-equivalent 'computational expressiveness', we would have achieved our ideal. Unfortunately, there appears to be an inevitable tradeoff between the acquirability and computational expressiveness of knowledge representations (Gruber, 1989;

Webster, 1988). Furthermore, the relative effectiveness of a mediating representation depends on both features of the problem and the roles and experience of participants. We discuss these issues below.

5.1. The Tradeoff Between Acquirability and Computational Expressiveness

Figure 8 depicts the tradeoff between acquirability and computational expressiveness. On one hand, programming languages are the epitome of computational expressiveness, but are not usable by those lacking special training. On the other hand, form-filling interfaces that may resemble the way a user normally enters information on paper are easy to learn , but they tend to be rigid, and thus limited in their range of applicability to specific problems that the system designers have foreseen.

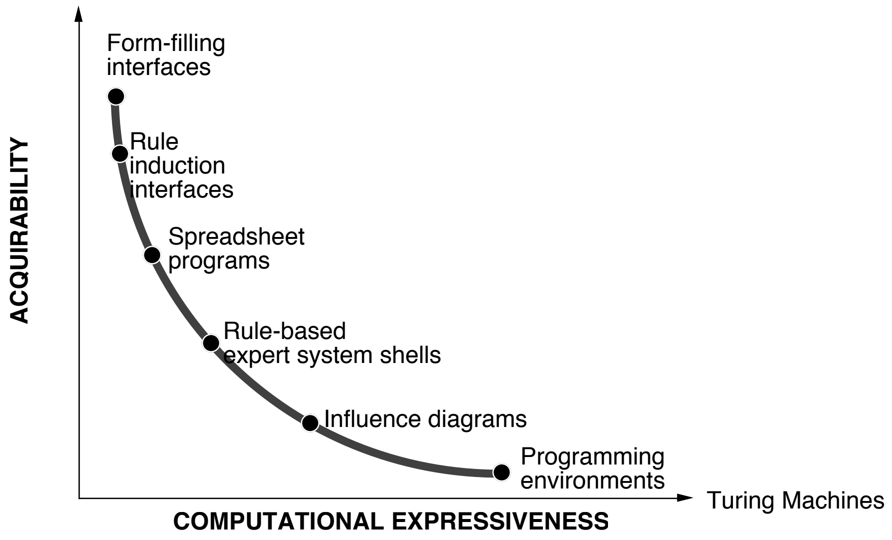

Figure 8. The tradeoff between acquirability and computational expressiveness (figure adapted from Gruber, 1989; Webster, 1988).

Knowledge acquisition tools do not eliminate the competition between acquirability and expressive power, but they can act as a kind of magnet to help pull the curve out (Figure 9). Applying such automated techniques can make acquirable representations more powerful and powerful representations more easy to learn and use.

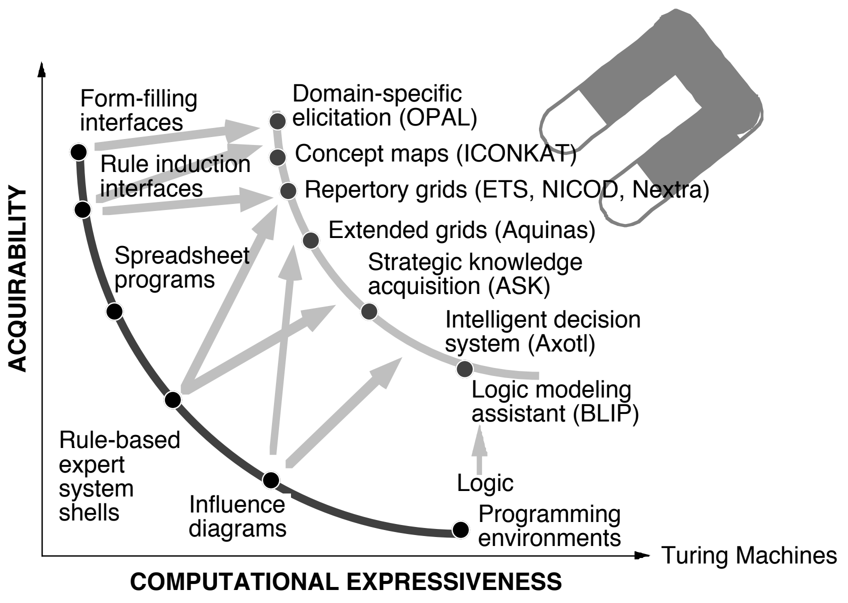
ACQUIRABILITY
Form-filling interfaces
Rule induction interfaces
Spreadsheet programs
Rule-based expert system shells
Influence diagrams
Programming environments
Domain-specific elicitation (OPAL)
Concept maps (ICONKAT)
Repertory grids (ETS, NICOD, Nextra)
Extended grids (Aquinas)
Strategic knowledge acquisition (ASK)
Intelligent decision system (Axotl)
Logic modeling assistant (BLIP)
Logic
Turing Machines
COMPUTATIONAL EXPRESSIVENESS

Figure 9. Knowledge acquisition tools can make acquirable representations more powerful and powerful representations more easy to learn and use (Figure adapted from Gruber, 1989).

The dotted arrows in Figure 10 illustrate the knowledge acquisition tool designer's dilemma of trying to create an ‘ideal’ representation that combines the naturalness of form-filling interfaces with the power and flexibility of a Turing machine. Research has generally attempted either to improve the computational expressiveness of human-efficient representations (horizontal vector; e.g., repertory grids and concept maps) or to improve the learnability of computationally powerful ones (vertical vector; e.g., McDermott, Dallemagne, Klinker, Marques & Tung, 1990). These two stereotypical flavors of knowledge acquisition research programmes are shown as horizontal and vertical vectors.It seems that the more computationally powerful the representation, the more difficult it is to maintain a high level of acquirability. This dilemma is identical to the one faced by software engineering researchers in their attempts to achieve the goal of automatic programming (Rich & Waters, 1987).

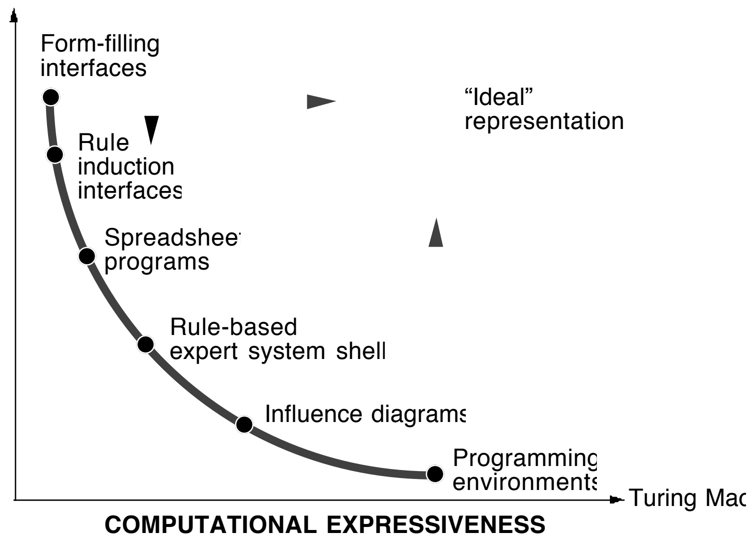

Figure 10. The knowledge acquisition tool designer's dilemma.

While not solving the dilemma, the use of a three-schemata architecture for knowledge representation can sometimes reduce the need for an "ideal" representation by providing formal and informal mappings between different problem representations.

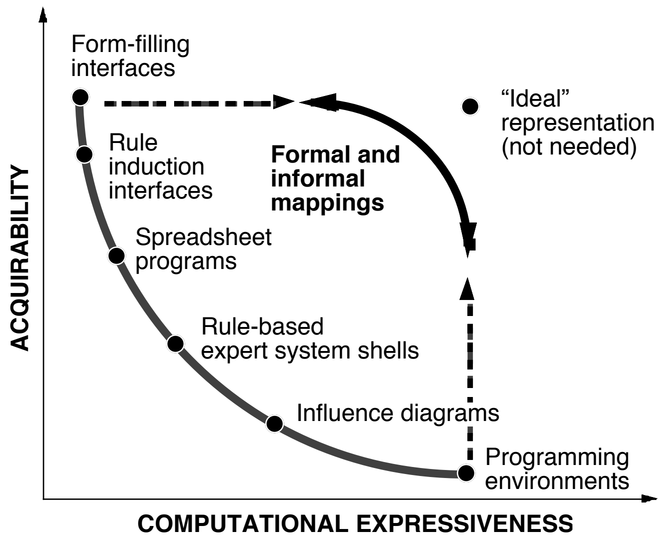

Figure 11. A three-schemata architecture for knowledge representation reduces the need for an "ideal" representation.

5.2. Relative Effectiveness of Mediating Representations for Different Kinds of Problems

We expect that graphical, diagrammatic representations will prove superior to prose for tasks involving complex spatial or conceptual relationships (Larkin & Simon, 1987). As Checkland (1979) has noted:

> "a diagram is an improvement on linear prose as a means of describing connections and relationships. Looking at a map, for example, we can take it as a whole. Our minds can process different parts of it simultaneously, in parallel, whereas prose has to be processed serially, putting a much greater burden on memory if our concern is with relationships. In addition, and presumably because of this possibility of 'parallel processing,' diagrams are automatically summaries. Imagine the amount of prose needed to convey all the information contained in a 1:50,000 Ordnance Map."

Various kinds of graphical representations will differ in their effectiveness for a given situation. For example, we find both repertory grid and directed graph representations to be useful as visual problem clarifiers. The grid presentation allows the person to see patterns of similarity and difference that would otherwise be difficult to grasp. However, dependencies, abstraction or subsumption relationships cannot readily be shown within a grid, and are most naturally displayed in directed graphs (Pearl & Verma, 1987; Pearl, Geiger, & Verma, 1988). Our experience in using both grids and directed graphs as mediating representations for knowledge convince us that each has advantages, depending on the context. As Jones (1981) states:

> "Matrices and nets are complementary ways of expressing a single set of relationships. The matrix enables a pattern that is too complex for the brain to generate all at once to be built up piece-by-piece outside the brain. A net of the same connections permits the assimilation of this pattern, once it has been completed and checked, back into the brain from whence came its constituent parts. Thus the brain can use an external aid to discover patterns among pieces of information that were originally understood only in isolation."

5.3. Relative Effectiveness of Mediating Representations for Different Kinds of Participants

Beyond these general considerations, there is the fact that a mediating representation that is expressive and understandable for one set of participants may not be useful for another, either because the content is foreign to their concerns or the form in which the information is presented is unfamiliar. Zachman's work (1987), for example, has emphasized the importance of selecting or creating the appropriate representations to support different kinds of participants in information system development. He uses the definition of classical architecture deliverables and adapts them for information system deliverables. These deliverables become the cognitive artifacts to support each kind of participant in information system development. Hence, in the process of constructing a building the deliverables are as follows:

- first, the architect and the owner derive a mutual understanding of the basic concepts of the building through use of the bubble charts;
- then the architect and owner agree on the form of the building as rendered in the architect's drawing;
- next, the architect's detailed drawings establish a basis for negotiation with the general contractor;
- following this, the contractor's plans are formulated to describe the final building as seen by the builder;
- finally, the shop plans provide a subcontractor's specifications and instructions for construction of a part or section of the building.

Similar levels of deliverables characterize the idealized development of information systems (Rich and Waters, 1987):

- At the top of the hierarchy are corporate executives who formulate vision, mission, goals, objectives, plans, and product deliverables for the company in relatively brief statements of policy.
- These statements of policy are used by strategic planners and experts on specific aspects of the business to arrive at detailed enterprise-wide descriptions of business entities and processes as they are and as they should be. These descriptions are couched in the vocabulary of the business and may serve to generate requirements not only for information systems, but also for other changes such as organizational restructuring.

- Systems analysts and designers develop information system models as the basis for design of the architecture of the program, translating high-level requirements into a detailed specification. A key feature of this specification is that it is couched in the vocabulary of programming rather than business.
- Finally, the programmer creates code in a programming language based on the detailed specification.

The important principle to draw from this illustration is that each "representation" of the building or information system has its own sphere of usefulness as an explicit basis of communication and agreement between adjacent participants in the process, and no single representation serves the needs of all. The goal is to support the most natural kind of formalization possible for a given set of participants, making the representation an effective vehicle for debugging the model.

6. Conclusion

We conclude with the words of David Parnas on traditional software specification, which apply equally well to knowledge acquisition:

> "The word 'formal' has been commandeered by a bunch of people who feel that it isn't formal if human beings can read it… I have fallen into the same trap. I could write something and I could read it but my students couldn't. And they could write something and they could read it but I couldn't. And, not only that, but neither of us *wanted* to read it. … Therefore I have worked on new ways to write specifications so that people could read it… You can't imagine how overjoyed I was when a pilot told me we had made a mistake with the A7 [avionics software specified in an earlier project] — not because we made a mistake but because the pilot could tell us." (Parnas, 1991; see also Brooks, 1987; Kapor, 1991).

It is our hope that a continued discussion and work on the design and use of mediating representations will increase our ability to design effective mechanisms for communication and shared understanding between participants in knowledge-based system development.

Acknowledgements

We express our appreciation to Miroslav Benda, Kathleen Bradshaw, Beverly Clark, Stan Covington, Jim Fulton, Pete and Cindy Holm, Earl Hunt, William Jones, Oscar and Sharon Kipersztok, Cathy Kitto, Joe Koszarek, Tim Lethbridge, Allen Matsumoto, Thom Nguyen, Steve Poltrock, Bob Schneble, Doug Schuler, Kish Sharma, Dave Shema, Bruce Wilson, and Debra Zarley for their contributions and support. This work has benefitted from discussions with Guy Boy, Ken Ford, Brian Gaines, Mildred Shaw, and Doug Skuce, as well as numerous colleagues in the knowledge acquisition community.*Axotl*, *Canard,* DART, *eQuality*, and DDUCKS have been developed within the Boeing Computer Services Computer Science organization.

References

Adams, S.S. (1988b). MetaMethods: Active values. *HOOPLA!,* 1(1),.3-6.

Adams, S.S. (1988b). MetaMethods: The MVC paradigm. *HOOPLA!,* 1(4), July, 5-6, 13-21.

Agnew, N. M., & Brown, J. L. (1989a). Foundations for a theory of knowing: I. Construing reality. *Canadian Psychology,* 30, 152-167.

Agnew, N. M., & Brown, J. L. (1989b). Foundations for a theory of knowing: II. Fallible but functional knowledge. *Canadian Psychology,* 30, 168-183.

Alexander, J.H., Freiling, M.J., Shulman, S.J., Rehfuss, S. & Messick, S.L. (1988). Ontological analysis: An ongoing experiment. In J.H. Boose & B.R. Gaines (Eds.), *Knowledge Acquisition Tools for Expert Systems*. London: Academic Press.

Baum, L.S., Shema, D.B., Boose, J.H., Bradshaw, J.M. (1989). Acquiring and Verifying Control Knowledge for a Blackboard System, *Proceedings of the Fourth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, October, 1989

Benda, M. (1990). Design of information systems: Towards an engineering discipline. Boeing Computer Services Technical Report. Seattle, WA: Boeing Computer Services, Computer Science Organization.

Boose, J.H. (1984). Personal construct theory and the transfer of human expertise. *Proceedings of the National Conference on Artificial Intelligence*, Austin, TX.

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

Boose, J.H. (1986b). Rapid acquisition and combination of knowledge from multiple experts in the same domain. *Future Computing Systems Journal,* 1, 191-216.

Boose, J.H. (1989). A survey of knowledge acquisition techniques and tools. *Knowledge Acquisition Journal*, 1, 3-37.

Boose, J.H. & Bradshaw, J.M. (1987). Expertise transfer and complex problems: Using *Aquinas* as a knowledge-acquisition workbench for knowledge-based systems. *International Journal of Man-Machine Studies*, 26, 3-28. Also in J. Boose & B. Gaines (Eds.), *Knowledge Acquisition Tools for Expert Systems*. London: Academic Press, pp. 39-64.

Boose, J. H., Bradshaw, J. M., Kitto, C. M., Shema, D. B. (1989). From ETS to *Aquinas*: Six years of knowledge acquisition tool development. *Proceedings of the Fourth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, October, pp. 5.1-17.

Boose, J.H., Bradshaw, J.M., Koszarek, J.L. & Shema, D.B. (1992). Better group decisions: Using knowledge acquisition techniques to build richer decision models. *Proceedings of the 1992 Hawaii International Conference on Systems Sciences*, January.

Boose, J.H., Bradshaw, J.M., Shema, D.B., Covington, S.P. (1989). Design knowledge capture for a corporate memory facility, *IJCAI-89 Workshop on Knowledge Acquisition: Practical Tools and Techniques*, Detroit, Aug., pp. 5-6.

Boose, J.H., Shema, D.S., & Bradshaw, J.M. (1989). Recent progress in *Aquinas*: A knowledge acquisition workbench. *Knowledge Acquisition Journal*, 1, 185-214.

Boose, J.H., Shema, D.S., & Bradshaw, J.M. (1990). Capturing design knowledge for engineering trade studies. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber, & M. Van Someren (Eds.), *Current Trends in Knowledge Acquisition*. Amsterdam: IOS Press.

Boy, G.A. (1991). Indexing hypertext documents in context. *Proceedings of the Third ACM Conference on Hypertext,* San Antonio, TX, December 15-18.

Bradshaw, J. M. & Boose, J.H. (1989). Knowledge acquisition as CASE for knowledge-based systems. Presentation at the *Third International Workshop on Computer-Aided Software Engineering* (CASE-89), London, July.

Bradshaw, J. M. & Boose, J.H. (1990). Decision analysis techniques for knowledge acquisition: Combining information and preferences using *Aquinas* and *Axotl*. *International Journal of Man-Machine Studies*, 32, 121-186. Also in J.H. Boose & B.R. Gaines (Eds.), *Progress in Knowledge Acquisition for Knowledge-Based Systems*. London: Academic Press.

Bradshaw, J.M., Boose, J.H., Covington, S.P., & Russo, P.J. (1989). How to do with grids what people say you can't: The application of decision analysis methods in *Axotl* and personal construct methods in *Aquinas* to design problems. *Proceedings of the Third AAAI Knowledge Acquisition for Knowledge-based Systems Workshop*, Banff, Canada, Nov., 1988.

Bradshaw, J.M., Boose, J.H. & Shema, D.B. (1992). A knowledge acquisition approach to design rationale. In J. Carroll and T. Moran (Eds.), *Design Rationale,* in preparation.

Bradshaw, J.M., Covington, S., Russo, P., & Boose, J.H. (1990). Knowledge acquisition techniques for intelligent decision systems: Integrating *Axotl* and *Aquinas* in DDUCKS. In M. Henrion, R.D. Shachter, L.N. Kanal, & J.F. Lemmer (Eds.), *Uncertainty in Artificial Intelligence 5*. North-Holland: Elsevier.

Bradshaw, J.M., Covington, S.P., Russo, P.J. & Boose, J.H. (1991). Knowledge acquisition techniques for decision analysis using *Axotl* and *Aquinas*. *Knowledge Acquisition Journal*, 3(1), 49-77.

Bradshaw, J.M., Ford, K.M., Adams-Webber, J.R. & Boose, J.H. (1992). A constructivist approach to knowledge acquisition tool development. In K. Ford & J. Bradshaw (Eds.), special knowledge acquisition issue of the *International Journal of Intelligent Systems,* in preparation. Also to appear in K. Ford & J.M. Bradshaw (Eds.), *Knowledge Acquisition as a Modeling Activity*. New York: John Wiley, volume in preparation.

Bradshaw, J.M., Holm, P., Kipersztok, O., Nguyen, T. & Covington, S. (1992). *eQuality:* A knowledge acquisition approach to process management and decision support tools. *Proceedings of the 1992 Hawaii International Conference on Systems Sciences*, January.

Bradshaw, J.M., Shema, D., Boose, J.H. & Koszarek, J.L. (1992). *Canard:* An alternative generation tool based on possibility tables. In S. Kim (Ed.), *Creativity: Models, Methods, and Tools,* AAAI Press, in press.

Brodie, M.L. (1989). Future intelligent information systems: AI and database technologies working together. In J. Mylopoulos & M.L. Brodie (Eds.) *Readings in Artificial Intelligence and Databases*. San Mateo, CA: Morgan Kaufmann.

Brooks, F. P., Jr. (1987). No silver bullet: Essence and accidents of software engineering. *Computer,* April, 10-18.

Carroll, J.M., Kellogg, W.A. & Rosson, M.B. (1991). The task-artifact cycle. In J.M. Carroll (Ed.), *Designing Interaction: Psychology at the Human-Computer Interface*. New York: Cambridge University Press.

Casner, S.M. & Larkin, J.H. (1989). Cognitive efficiency considerations for good graphic design. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*. August, 275-282.

Chang, S. K (Ed.) (1990).*Visual Languages and Visual Programming*. New York: Plenum Press.

Checkland, P.B. (1979). Techniques in "Soft" systems practice: Part I. Systems diagrams—some tentative guidelines. *Journal of Applied Systems Analysis,* 6, 33-40.

Clancey, W.J. (1989). The frame of reference problem in cognitive modeling. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*. Ann Arbor, Michigan, August 16-19.

Clancey, W.J. (1990). Implications of the system-model-operator metaphor for knowledge acquisition. In H. Motoda, R. Mizoguchi, J. Boose, & B. Gaines (Eds.) *Knowledge Acquisition for Knowledge Based Systems*. Amsterdam: IOS Press.

Cox, L.A. (1991). Knowledge acquisition for model building. In K.M. Ford & J.M. Bradshaw (Eds.), special knowledge acquisition issue of the *International Journal of Intelligent Systems*, in press.

Diaper, D. (1989). Designing expert systems—from Dan to Beersheba. In D. Diaper (Ed.) *Knowledge Elicitation: Principles, Techniques and Applications*. New York: John Wiley.

Eisenstadt, M., Domingue, J., Rajan, T. & Motta, E. (1990). Visual knowledge engineering. *IEEE Transactions on Software Engineering,* 16(10), October, 1164-1177.

Feigenbaum, E.A. & McCorduck, P. (1983). *The Fifth Generation*. New York: Addison-Wesley.

Ford, K.M. (1987). *An Approach to the Automated Acquisition of Production Rules from Repertory Grid Data*. Ph.D. dissertation, Tulane University.

Ford, K., Bradshaw, J.M. & Adams-Webber, J.R. (1992). Knowledge acquisition as a modeling activity. In K. Ford & J. Bradshaw (Eds.), special knowledge acquisition issue of the *International Journal of Intelligent Systems,* in preparation. Also to appear in K. Ford & J.M. Bradshaw (Eds.), *Knowledge Acquisition as a Modeling Activity*. New York: John Wiley, volume in preparation.

Ford, K.M., Petry, F.E., Adams-Webber, J.R., & Chang, P.J. (1991). An approach to knowledge acquisition based on the structure of personal construct systems. *IEEE Transactions on Knowledge and Data Engineering*, 3, 1-11.

Ford, K.M., Stahl, H., Adams-Webber, J.R., Cañas, A.J., Novak, J. & Jones, J.C. (1991). ICONKAT: An integrated constructivist knowledge acquisition tool. *Knowledge Acquisition Journal*, 3(2), 215-236.

Gaines, B.R. & Boose, J.H. (1991). Standards requirements, sources, and feasibility in knowledge acquisition.*Working notes of the AAAI Workshop on Standards in Expert Systems*. Anaheim, CA: July 14.

Gaines, B.R. & Shaw, M.L.G. (1986a). Interactive elicitation of knowledge from experts. *Future Computing Systems*, 1(2).

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

Goldberg, A. (1990). Information models, views, and controllers. *Dr. Dobb's Journal,* July, 1-4.

Gruber, T.R. (1989). *The Acquisition of Strategic Knowledge*. New York: Academic Press.

Gruber, T.R. (1990). Justification-based knowledge acquisition. In H. Motoda, R. Mizoguchi, J. Boose, & B. Gaines (Eds.) *Knowledge Acquisition for Knowledge Based System*s. Amsterdam: IOS Press.

Gruber, T.R. (1991). The role of common ontology in achieving sharable, reusable knowledge bases. Stanford Knowledge Systems Laboratory Report No. KSL 91-10, February. To appear in J.A. Allen, R. Fikes, and E. Sandewall (Eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*. San Mateo, CA: Morgan Kaufmann.

Hix, D. (1990). Generations of user-interface management systems. *IEEE Software,* September, 77-87.

Ichikawa, T. & Chang, S.K. (Eds.) (1990). Special issue on visual programming. *IEEE Transactions on Software Engineering,* 16(10).

Johnson, N.E. (1989). Mediating representations in knowledge elicitation. In D. Diaper (Ed.) *Knowledge Elicitation: Principles, Techniques and Applications*. New York: John Wiley.

Jones, J.C. (1981). *Design Methods*. New York: John Wiley & Sons.

Kaplan, A. (1963). *The Conduct of Inquiry*. New York: Harper and Row.

Kapor, M. (1991). A software design manifesto. *Dr. Dobbs Journal,* January, 62-67.

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

Krasner, G.E. & Pope, S.T. (1988). A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80. *Journal of Object-Oriented Programming,* August-September, 26-49.

Laland, A., Novotny, R., Enzer, S. & Bortz, J. (1991). *The TIGRE Programming Environment*. Santa Cruz, CA: TIGRE Object Systems.

Larkin, J.H. & Simon, H.A. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11, 65-99.

Lenat, D.B. & Guha, R.V. (1990). *Building Large Knowledge-based Systems*. Reading, MA: Addison-Wesley.

Lethbridge, T.C. (1991). Creative knowledge acquisition: An analysis. *Proceedings of the 1991 Banff Knowledge Acquisition for Knowledge-Based Systems Workshop,* Banff, Canada, October.

Linster, M. & Gaines, B.R. (1990). Supporting acquisition and performance in a hypermedia environment. Presentation at *Terminology and Knowledge Engineering Workshop*, Oct.

McDermott, J., Dallemagne, G., Klinker, G., Marques, D. & Tung, D. (1990). Explorations in how to make application programming easier. In H. Motoda, R. Mizoguchi, J. Boose, & B. Gaines (Eds.) *Knowledge Acquisition for Knowledge Based Systems*. Based on Proceedings of JKAW '90, Tokyo, Japan, Oct. Amsterdam: IOS Press.

McNamee, P. & Celona, J. (1987). *Decision Analysis for the Professional—With Supertree*. Redwood City, CA: Scientific Press.

Moore, E.A. & Agogino, A.M. (1987). INFORM: An architecture for expert-directed knowledge acquisition. *International Journal of Man-Machine Studies*, 26, 213-230.

Musen, M. A. (1989). *Automated Generation of Model-Based Knowledge-Acquisition Tools*. San Mateo, CA: Morgan Kaufmann.

Narayanan, H., Chandrasekaran, B., Iwasaki, Y. & Simon, H. (1991). Announcement of the AAAI Spring Symposium on Reasoning with Diagrammatic Representations, March 25-27, 1992, Stanford University, Stanford, CA).

Neches, R. , Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T. & Swartout, W.R. (1991). Enabling technology for knowledge sharing.*AI Magazine,* Fall, 36-55.

Norman, D.A. (1988). *The Psychology of Everyday Things*. New York: Basic Books.

Norman, D.A. (1991). Cognitive artifacts. In J.M. Carroll (Ed.), *Designing Interaction: Psychology at the Human-Computer Interface,* New York: Cambridge University Press.

Parnas, D. (1991). The use of formal methods for computer system documentation. Quoted in *Software Maintenance News,* 9(5), May, 29.

Pearl, J., Geiger, P. & Verma, T. (1988). The logic of influence diagrams. *Proceedings of the Conference on Influence Diagrams for Decision Analysis, Inference, and Prediction*. May 9-11, 1988, Berkeley California, University of California at Berkeley.

Pearl, J. & Verma, T. (1987). The logic of representing dependencies by directed graphs. *Proceedings of the National Conference on Artificial Intelligence,* Seattle.

Puerta, A., Egar, J., Tu, S. & Musen, M. (1991). A multiple-method knowledge-acquisition shell for the automatic generation of knowledge-acquisition tools. Stanford Knowledge Systems Laboratory Report KSL-91-24. *Proceedings of the Sixth Banff Knowledge Acquisition for Knowledge-Based Systems Workshop,* Banff, Canada, Oct. 6-11.

Rich, E. (1983) *Artificial Intelligence*. New York: McGraw-Hill

Rich, C. & Waters, R.C. (1987). Artificial intelligence and software engineering. In W. E. L. Grimson and R.S. Patil (Eds.) *AI in the 1980s and Beyond: An MIT Survey*. Cambridge, MA: The MIT Press.

Rothenberg, J. (1989). The nature of modeling. In L.E. Widman, K.A. Loparo & N. R. Nielsen (Eds.) *Artificial Intelligence, Simulation, and Modeling*. New York: John Wiley.

Schreiber, G., Akkermans, H. & Wielinga, B. (1990). On problems with the knowledge level perspective. *Proceedings of the Fifth Banff Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Banff, Canada, November.

Schuler, D., Russo, P., Boose, J., & Bradshaw, J. (1990). Using personal construct techniques for collaborative evaluation. *International Journal of Man-Machine Studies,* 33, 521-536.

Shaw, M.L.G. & Woodward, J.B. (1989). Mental models in the knowledge acquisition process. *Proceedings of the Fourth Knowledge Acquisition Workshop for Knowledge Based Systems.* (KAW-89). Banff, Canada, Oct.

Shaw, M.L.G., Woodward, J.B. & Gaines, B.R. (1990). A cognitive framework for knowledge acquisition methodologies and tools. Department of Computer Science, University of Calgary, Internal Report. Calgary, Alberta, Canada.

Shema, D.B. & Boose, J.H. (1988). Refining problem-solving knowledge in repertory grids using a consultation mechanism. *International Journal of Man-Machine Studies*, 29, 447-460.

Shema, D.B., Bradshaw, J.M., Covington, S.P. & Boose, J.H. (1990). Design knowledge capture and alternative generation using possibility tables in *Canard*. *Knowledge Acquisition Journal*, 2, 345-363.

Sivard, C., Zweben, M., Cannon, D., Lakin, F. & Leifer, L. (1989). Conservation of design knowledge. *Proceedings of the 27th Aerospace Sciences Meeting of the American Institute of Aeronautics and Astronautics*, Reno, Nevada, January 9-12.

Skuce, D. (1991a). A review of 'Building large knowledge based systems' by D. Lenat and R. Guha. *Artificial Intelligence,* in press.

Skuce, D. (1991b). A frame-like knowledge acquisition integrating abstract data types and logic. In J. Sowa (Ed.), *Principles of Semantic Networks,*. San Mateo, CA: Morgan Kaufmann.

Skuce, D. (1991c). A wide spectrum knowledge management system. *Knowledge Acquisition Journal,* in press.

Sowa, J.F. (1987). There's more to logic than predicate calculus. In J. Carbonell and K. Fuchi (Eds.) *Proceedings of the U.S.-Japan AI Symposium,* Tokyo, Japan, December.

Sowa, J.F. (1991). Toward the expressive power of natural language. In J. Sowa (Ed.), *Principles of Semantic Networks,*. San Mateo, CA: Morgan Kaufmann.

Tauzovich, B. & Skuce, D. (1990). A general taxonomy for conceptual data modeling. Ottawa, Canada: Cognos, Inc.

Thimbleby, H. (1990). *User Interface Design*. Reading, Mass.: Addison-Wesley.

Tortora, G. (1990). Structure and interpretation of visual languages. In Shi-Kuo Chang (Ed.), *Visual Languages and Visual Programming*. New York: Plenum Press.

van Griethusen, J.J. & King, M.H. (Eds.) (1985). Assessment guidelines for conceptual schema language proposals (ISO TC97/SC21/WG5-3), August.

Webster, D.E. (1988). Mapping the design information representation terrain. *Computer*, Dec., 8-23.

Wielinga, B.J., Akkermans, H., Schreiber, A.TH. & Balder, J.R. (1989). A knowledge acquisition perspective on knowledge-level models. In J.H. Boose & B.R. Gaines (Eds.), *Proceedings of the 4th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*. Canada, Nov.

Wielinga, B.J., Schreiber, A.Th. & Breuker, J.A. (1991). KADS: A modeling approach to knowledge engineering. *Knowledge Acquisition* , in press.

Winograd, T. & Flores, F. (1986). *Understanding Computers and Cognition*. Norwood, N.J.: Ablex.

Winston, P. H. *Artificial Intelligence*. 2nd Edition. Reading, Mass.: Addison-Wesley, 1984.

Zachman, J.A. (1987). A framework for information systems architecture. *IBM Systems Journal*, 26(3), 276-292.

Zwicky. F. (1969). *Discovery, Invention, Research through the Morphological Approach*. New York: Macmillan.

Note that the same information can be represented in more than one way. For example decomposition: tree (depth), outline (breadth), dependent list (fixed number, each level different meaning).

Multiple types of information in same form: Graph view: ID, E-R, process views

Transformation: Process view -> activity graph view -> agenda. Grids to influence diagrams

Composition: Embedding grids within possibility tables

Organization views vs. sketchpad views, vs.model views: Project notebook