Sharable Ontologies as a Basis for Communication and Collaboration in Conceptual Modeling

Jeffrey M. Bradshaw, Peter D. Holm, John H. Boose
Research & Technology, Boeing Computer Services
P.O. Box 24346, M/S 7L-64, Seattle, WA 98124; (206) 865-3422; jbrad@atc.boeing.com

Douglas Skuce, Timothy C. Lethbridge
Department of Computer Science, University of Ottawa
Ottawa, Ontario, Canada K1N 6N5; (613) 564-4518; doug@csi.uottawa.ca

ABSTRACT

We are interested in improving collaboration among researchers, domain experts, and knowledge engineers through better knowledge sharing. Building models is not only a way to *formulate* domain knowledge, but serves more importantly as a means to *communicate* and *come to understand* the evolving problem space. An important prerequisite to effective communication is mutual agreement on important terms and concepts, the *ontology* of the domain. We give an overview of the CODE4 conceptual modeling representation and the DDUCKS modeling environment. We focus on how different classes of ontologies are developed within the tools: top level ontologies with very general concepts, ontologies with modeling concepts, and ontologies for particular domains. We are evaluating theories and contexts as mechanisms for partitioning and managing relationships between groups of concepts. We describe some of the mechanisms we are exploring for exchanging these ontologies with other research groups, and conclude by reviewing important research issues for future work.

1. INTRODUCTION: MODELING AS COMMUNICATION

Knowledge acquisition is largely a matter of communication. It begins when experts in some domain determine that they have valuable information to share. It evolves as participants collaborate to define models that represent a common understanding of certain aspects of the domain. It succeeds when participants can use these models effectively to promote and enrich communication.

This view of knowledge acquisition as modeling is at odds with early knowledge acquisition metaphors that stressed notions of *expertise transfer* and *knowledge acquisition bottlenecks* (Ford & Bradshaw, in press) . Clancey (Clancey, in press) has explained why the bottleneck metaphor is so misleading. He shows that knowledge acquisition is not a matter of squeezing pre-defined chunks of knowledge through a narrow communication channel, but “usually involves inventing new languages for modeling previously unarticulated experience." Clancey's arguments parallel those of Michael Reddy (Reddy, 1979) who criticizes the *conduit metaphor* as a characterization of language and meaning:

> "When one person talks to another, words do not come out of the speaker's mouth as packages full of direct report on facts, which the hearer then opens up to discover the fully packed meaning inside. On the contrary, the speaker forces air out of the lungs and shapes the molecules with the vocal apparatus; these molecules vibrate on the hearer's eardrums, sending neurochemical impulses to the brain. The hearer then *constructs a model* of what the speaker is thought to have said. People understand each other and communicate not by direct conveyance at all, but by broadcasting signals, which themselves evoke mental models." (Frawley, 1992)

Modeling is a *contextual* and *purposive* activity—that is, to be involved in modeling is necessarily to be engaged in using the model in some particular setting for particular reasons that together 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) .

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) . Indeed, sometimes the most important outcome of the modeling process may not be the model itself, but rather the insight we gain as we struggle to articulate, structure, critically evaluate, and rely on it (Moore & Agogino, 1987) . Thus, the value of such an effort derives 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. We must ask ourselves not only, How do we know the model is correct? (every model is an incorrect oversimplification); but also, How useful is the model (and the modeling process) in facilitating our understanding of the domain?

Hence we see that much of the power of models lies in their ability to function as tools for thought, or *cognitive artifacts* (Norman, 1992) . Cognitive artifacts can enhance human representational functions through their ability to maintain, display and operate upon information. They can change the way tasks get done by changing the actions required, and distributing actions across time (precomputation) and people (distributed cognition). Because of the major impact that particular representations chosen can have on problem solving, we ought to design the internal and external form of our models with great care and attention (Chandrasekaran, Narayanan, & Iwasaki, in press; Larkin & Simon, 1987) . We also must attend to the transformations of representational form that ought to occur to support different aspects of the modeling process (Gaines, Shaw, & Woodward, in press) .

Attention to the use of models as cognitive artifacts enhancing communication and cognition is reflected in the research in the field of conceptual modeling:

> *"Conceptual modeling* is the activity of *formally* describing some aspects of the physical and social world around us *for purposes of understanding and communication.* Such

> descriptions, often referred to as *conceptual schemata,* require the adoption of a formal notation, a *conceptual model* in our terminology. Conceptual schemata capture relevant aspects of some world, say an office environment and the activities that take place there, and can serve as points of agreement among members of a group, for example the workers in that office, who need to have a common understanding of that world. Conceptual schemata can also be used to communicate that common view to newcomers, through a variety of graphic and linguistic interfaces. Conceptual modeling has an advantage over natural language or diagrammatic notations in that it is based on a formal notation which allows one to 'capture the semantics of the application'. It also has an advantage over mathematical or other formal notations developed in computer science because unlike them, conceptual modeling supports structuring and inferential facilities that are psychologically grounded. After all, the descriptions that arise from conceptual modeling activities are intended to be used by *humans*, not machines." (Mylopoulos, 1991)

Unfortunately, until recently developers of modeling and knowledge representation tools have paid scant attention to the needs of the kind of users who are building a model with the intent of coming to understand some domain. For example, traditional data modeling tools introduce assumptions about the way conceptual schemata will be realized on a physical machine or database implementation, which constrain and confound domain experts preferring to leave these considerations to others. Furthermore, the need of such users to model the "real world" drives a requirement for much richer representations than modeling tools typically afford. Knowledge representation tools, on the other hand, almost always introduce the assumption that the resultant knowledge bases will be directly usable for some computational task. Considerations of efficiency and performance inevitably lead to limitations in the richness and flexibility of the knowledge representation. No less important, there are few modeling tools that have been optimized for *usability*. Tools are cumbersome, unnatural, and difficult to use without specialized training and extensive experience. The comprehensibility and succinctness of textual and graphical notations chosen is rarely considered (Bradshaw & Boose, 1992; Chandrasekaran, Narayanan, & Iwasaki, in press) . As a result, many potential users abandon modeling tools altogether and use simple graphical drawing tools instead.

Elsewhere we have discussed our ideas about how the application of certain general principles of knowledge representation and user-interface design can make knowledge acquisition tools more usable and extensible by non-experts (Bradshaw, Ford, & Adams-Webber, 1991; Ford, Bradshaw, Adams-Webber, & Agnew, in press; Skuce, in pressb) . Section 2 provides an overview of selected aspects of the CODE4 conceptual modeling representation and the DDUCKS modeling environment. We discuss the meaning triangle (Ogden & Richards, 1923) as a way to help us understand how objects in the "real world" relate to objects in the conceptual model and their expression in a particular syntax. We examine how tools and representations for conceptual modeling can satisfy requirements for richness, flexibility, and tailorability.

In Section 3, we discuss preliminary results in our efforts to create sharable specifications for the concepts upon which our applications depend. In this we join with a number of researchers who have argued the benefits for making conceptual commitments explicit in the form of *ontologies*(Gruber, 1991; Gruber, 1992a; Gruber, 1992b; Lenat & Guha, 1990; Neches, Fikes, Finin, Gruber, Patil, Senator, & Swartout, 1991; Neches, Foley, Szekely, Sukaviriya, Luo, Kovacevic, & Hudson, 1992; Skuce & Monarch, 1990) :

> “Consider a planning system based on a theory in which plans are composed of ‘steps’ which form ‘sequences’ with specific kinds of ‘resource dependencies’, and that the search for plans is guided by ‘ordering heuristics’ and ‘optimization criteria.’ If one wished to use this planning system, one would need to understand what these words mean, and build a knowledge base in which domain-specific knowledge was formulated in terms that the planning *program* also understands.” (Gruber, 1992a)

The term *ontology* is borrowed from the philosophical literature where it describes a theory of what exists. Such an account would typically include terms and definitions only for the very basic necessary categories of existence. However, the common usage of ontology in the knowledge sharing and reuse community is as a vocabulary of representational terms and their definitions at any level of generality. A knowledge-based system’s “ontology” defines what exists for the program—in other words, what can be represented by it (Gruber, 1992b) .

We focus on how different classes of ontologies are developed within the tools: top level ontologies with very general concepts, ontologies with modeling concepts, and ontologies for particular domains. We discuss theories and contexts as mechanisms for partitioning and managing relationships between groups of concepts. We describe some of the mechanisms we are exploring for exchanging these ontologies with other research groups, and conclude by reviewing important research issues for future work.

2. AN OVERVIEW OF CODE4 AND DDUCKS

2.1. Three-Schema Architectures, Predication, and the Construction of Meaning

There are two general approaches to meaning: one which posits a *direct relation* between symbols and their referents (Haiman, 1985; Wittgenstein, 1974/1918) , and another (now dominant) approach which posits an *indirect relation*(Frawley, 1992; Jackendoff, 1983; Lakoff, 1987; Perez, 1992; Sowa, 1984) . The roots of the indirect view are usually traced from Aristotle through the more recent *conceptualist position* proposed by Ogden and Richards (Ogden & Richards, 1923) . They characterized meaning as a *semiotic triangle* (or *meaning triangle),* a relation between a symbol (term, icon) and a referent (object, extension), mediated by thought (concept, intension, idea, sense) (see Figure 1).[1]

[1] As Sowa (Sowa, 1984) observes, the aspects of meaning described by Ogden and Richards reflect the distinction between *semantic* and *episodic* memory (Tulving, 1972). Semantic memory, which stores universal principles corresponding to dictionary definitions, is related to intensional meaning. Episodic memory, which stores facts about individual things and events corresponding to history and biography, is related to extensional meaning.

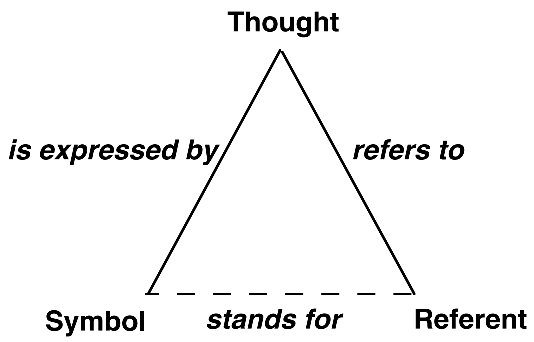

Figure 1. The meaning triangle (Ogden & Richards, 1923) .

The meaning triangle is useful for understanding how objects in the "real world" relate to objects in the conceptual model and their expression in a particular syntax (Fulton, 1992) . Figure 2 shows a set of interlocking meaning triangles illustrating relationships between employees. We have inverted the triangle so that referents *("real world" objects)* appear near the bottom of the figure, symbols *(the abstract syntax)* in the upper left, and mediating concepts *(the abstract objects constituting an abstract set-theoretic ontology)* in the upper right. Each term in the model *(Tom, Betty)* is linked to a corresponding object in the domain of discourse and a referent in the world. The predicate *works for* expresses the set of all *occurrences* and refers to the three situations in the world being described. The sentence *Tom works for Betty,* on the other hand, is related to a single occurrence of the predicate. An abstract set-theoretic *semantics* is defined by the mapping between the abstract objects and the abstract syntax.

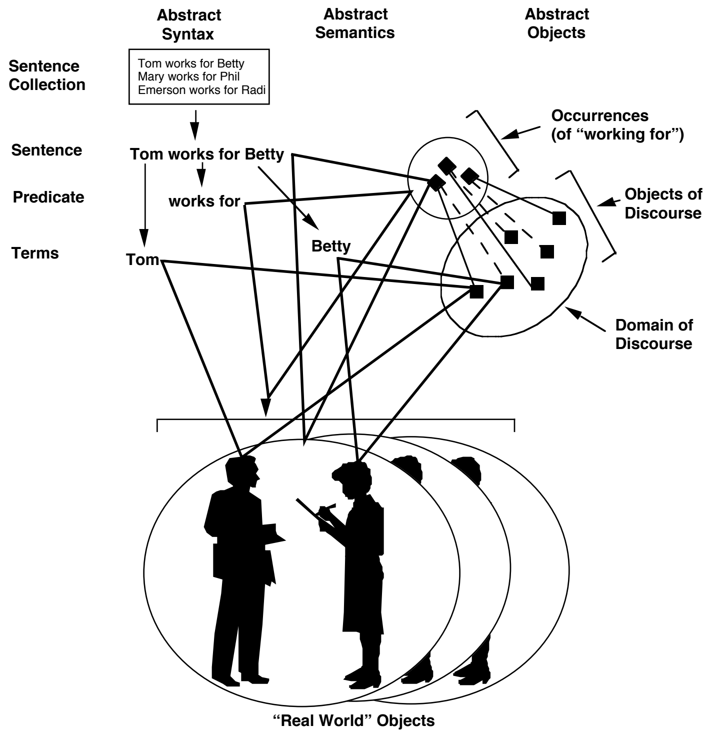

Figure 2. Interlocking meaning triangles show how objects in the "real world" relate to objects in the conceptual model and their expression in a particular syntax (Fulton, 1992) .

While a relatively neutral, purely descriptive model of the situation like the one above may be useful for some purposes, it leaves many potential questions unanswered. For example, we may want to get judgmental, *presciptive* advice from a model to help us determine whether Tom *should* be working for Betty or whether we have the optimal number of managers in the organization. To answer these questions, we must construct an interpretation of the situation within a more specialized modeling or problem-solving framework.

Let's say that we are trying to make a decision between Betty, Phil, and Radi as supervisor for Tom. Thus our three options are *Tom works for Betty (d1), Tom works for Phil (d2),* and *Tom works for Radi (d3).* We can represent these options in a variety of ways: for example, as *elements* in a repertory grid, as *alternatives* in an influence diagram or decision tree, or as *possibilities* in a possibility table (Figure 3). While we have argued previously that none of these representations are fundamentally incompatible (Bradshaw & Boose, 1990; Bradshaw, Covington, Russo, & Boose, 1991) , they all require different assumptions, vocabulary, and modes of interaction. Within our knowledge acquisition tool, we do not want to commit ourselves to any one of them to the exclusion of the others, because each assists in complementary ways with different aspects of the decision-making process. Furthermore, within The Boeing Company, heterogeneity of tools and notations, and differences in terminology are a fact of life. We have frequently observed the same domain information being used in several ways within different modeling frameworks as data is passed from organization to organization. We need a way to translate between these frameworks without loss of meaning.

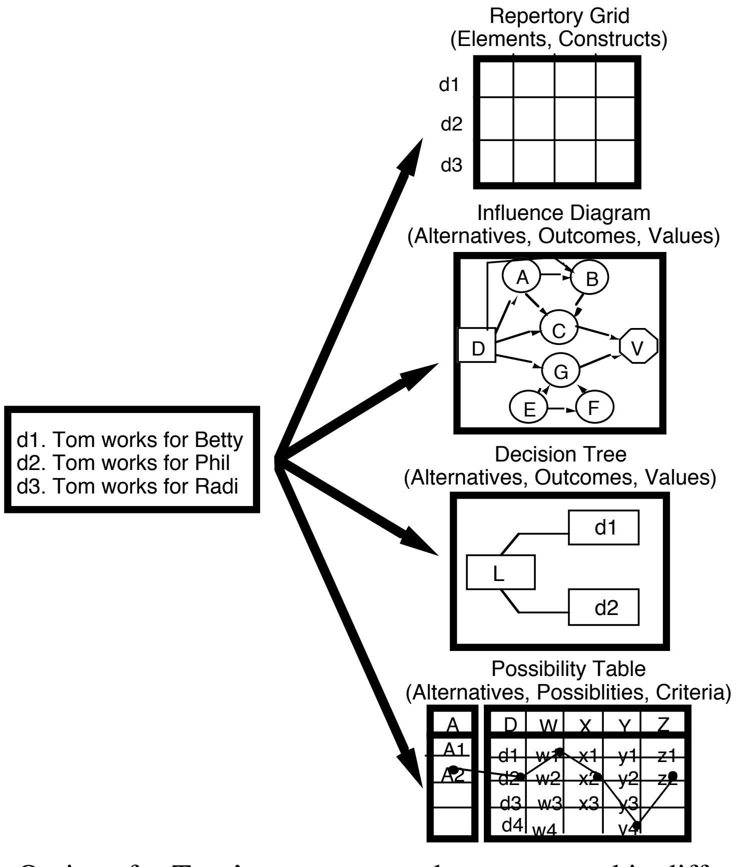

Figure 3. Options for Tom's manager can be represented in different ways.

The problem is one of *semantic unification,* a challenging research topic that is currently being tackled by several research groups (Fulton, Zimmerman, Eirich, Burkhart, Lake, Law, Speyer, & Tyler, 1991; Genesereth & Fikes, 1992; Gruber, 1992a; Gruber, 1992b; Perez, 1992; Sowa & Zachman, 1992) . While we do not presume to have an answer to the general problem of semantic unification, we are exploring some relevant issues that bear on the development of our knowledge acquisition tools.

Figure 4 shows a pair of cascading meaning triangles illustrating processes of *abstraction, interpretation,* and *expression* in modeling. In order to facilitate reuse, a relatively neutral domain model is constructed by abstraction based on a mental projection of the real world. The domain model may in turn be interpreted within a more specialized modeling or problem-solving framework, and expressed in terms of a particular visual or textual syntax. The same domain model may be interpreted in multiple ways (e.g., in terms of personal construct theory vs. decision theory); the same interpretation theory may be expressed differently (e.g., as decision trees vs. influence diagrams); and the interpretation model may itself be interpreted within another framework (e.g., a decision-theoretic model interpreted by a domain-specific explanation framework).[2]

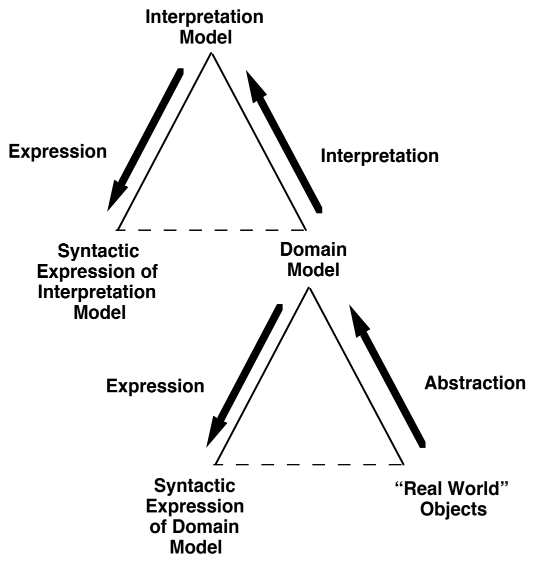

Figure 4. Cascading meaning triangles illustrating processes of *abstraction, interpretation,* and *expression* in modeling.

We discuss now two central problems in building tools to support conceptual modeling:

1. Defining a conceptual modeling representation that is sufficiently rich and flexible (Section 2.2).

[2] One way to think about the differentiation between the relatively “neutral” domain model and the interpretation models specialized to different modeling and problem-solving strategies is that it provides a way to address some of the requirements that led (Guha & Lenat, 1990; Lenat & Guha, 1990) to define separate *epistemological* and *heuristic* levels in Cyc.

2. Defining a capability for users to create mappings between different interpretations and between different expressions of the model (Section 2.3).

2.2. The CODE4 Conceptual Modeling Representation

Ideally, we want a representation that approaches natural language in its ability to describe the subtle details and fluid nature of experience, yet can also be translated straightforwardly into formal, computer-based representations. While we are still so far from the ideal that we do not know even if it will ever be possible to approximate it, a number of significant strides in knowledge representation research have been made over the past few years (Borgida, Brachman, McGuinness, & Resnick, 1989; Gaines, 1991; Genesereth & Fikes, 1992; Neches, Fikes, Finin, Gruber, Patil, Senator, & Swartout, 1991; Perez, 1992; Sowa, 1984; Sowa, 1991) . In this section, we briefly describe some of the basic features of the CODE4 conceptual modeling representation, which forms the heart of our approach.

CODE4 is a conceptual modeling tool that has been developed at the University of Ottawa (Lethbridge, 1991; Lethbridge & Skuce, 1992a; Lethbridge & Skuce, 1992b; Skuce, 1991; Skuce, in pressb; Skuce & Lethbridge, submitted for acceptance; Skuce & Monarch, 1990) . A collection of integrated tools support the important and frequently overlooked aspects of conceptual, ontological, and terminological analysis. All these tools are available through a rich, graphical interface.

CODE4 has been used in a number of university and industrial settings as a stand-alone tool. At The Boeing Company, however, we are using the tool in a somewhat different manner. We have developed our own user-interface in the DDUCKS modeling environment, while relying on CODE4 as a back end implementing the conceptual modeling representation. The Boeing group is also developing extensions to the representation to allow the system to make use of additional inferencing and representation facilities similar to those found in Sowa's (Sowa, 1991) conceptual graphs and Gaines' (Gaines, 1991) KRS, which interpret taxonomic and entity-relationship structures in terms of intensional and extensional logics. Currently, any CODE4 representation can be mapped to a conceptual graph, for which semantics exist (Skuce & Lethbridge, submitted for acceptance) .

Below, we present a brief summary of some of the basic concepts of the language, as extended in DDUCKS,. See(Lethbridge & Skuce, 1992a; Skuce & Lethbridge, submitted for acceptance) for a more complete discussion of knowledge representation in CODE4.

Things, concepts., types and individuals. Concepts in a CODE4 knowledge base are used to represent things in the world. Things are either types or individuals. Both types and individuals belong to one or more types (the type of “type” is *type*).

Names and unique ID's. Types and individuals have properties and are referred to by terms. The concept's main term is always its name. Concepts always have a term corresponding to a unique ID. The unique ID allows two concepts with the same name to be distinguishable.

Concept hierarchies. Types and individuals are arranged a *kind-of* hierarchy or taxonomy from more general to more specific concepts. Figure 5 shows the hierarchy of basic primitives available in CODE4. Types may have either individuals or more specific types as their subconcepts. Individuals may not have subconcepts.

[redacted]

Figure 5. The hierarchy of basic primitives available in CODE4

Inheritance, multiple inheritance and disjointness. Subconcepts typically inherit the properties of their superconcepts. However users have extensive control over the specific aspects of inheritance, if desired. Concepts may inherit from more than one superconcept *(multiple inheritance)*. Sometimes it is impossible for a concept to be a subconcept of a given pair of superconcepts at the same time (e.g., Dick cannot be both a Republican and a Democrat). If this is the case, we specify that the incompatible superconcepts are to be *disjoint*.

Properties and statements. Properties are characteristics that we ascribe to things. We describe things by making statements about them, e.g., "roses are red, violets are blue…"[3]. Statements have at least two essential parts:

• the subject which refers to a thing (roses, violets)

• the predicate which refers to a property of that thing (are red, are blue).

Property hierarchies. Predicates are arranged in a separate hierarchy from things. If a concept has a property, it implies that it also has the superproperty (e.g., *has wheels* implies *has parts*). Every predicate has two parts:

• its most general subject (the most general concept in the concept hierarchy it applies to)

• its implicants (the more general predicates that it implies).

In DDUCKS, the value of a property may be directly asserted by the user or computed by a script.

Attributes and related things. Some important specializations of the property hierarchy are attributes, related things, parts, subsets, state and event predicates, and roles.

Attributes are particular kinds of predicates. User-created attributes will have one of two attribute types as a superproperty: attributes with value or Boolean attributes. Attributes with value take simple values like text, symbols, numbers, bowlines, dates, times, lists, or dictionaries as their values. The system automatically interprets user entered values as the appropriate data type (e.g., *22 June 1990* is interpreted as a Smalltalk object of class *Date*). Values can be specified for a type or an individual. The values of the type are inherited to the individual, which may override them. Boolean attributes are useful when the predicate is naturally unary (e.g., the verb is intransitive (e.g., walks, moves) or the predicate is derived from an adjective (e.g., alive)). Statements about related things take other concepts as their value.

[3] *Statements* are roughly equivalent to Fulton's notion of *occurrences*.

Facets. Facets provide additional descriptive information about statements. Built-in user-editable facets include: status, statement comment, knowledge reference, cardinality, script, legal values, and access parameters. Users can add their own facets.

Metaconcepts and their properties. The system distinguishes between properties ascribed to the thing in the world and the properties of the concept representing the thing. For example:

Concept of Boeing 747 (metaconcept)

date created: 5 March 1992

unique ID: @34355

Boeing 747 (concept)

date created: 1 January 1990

ID: SN-59874-900

Both types and individuals have metaconcepts. The following metaconcept properties are predefined: last modifier, date last modified, creator, date created, description, definition, concept status, unique ID. Most of these are maintained automatically by the system. Additional metaconcept properties can be defined by the user.

Terms. One of the unique strengths of CODE4 is its treatment of linguistic knowledge, especially terminology, i.e., the association between concepts and the phrases that denote them. CODE4 provides a high level of integrated support for lexical functions such as: defining a term, comparing meanings of closely related terms, using a term in several senses, or using synonyms, checking that a term is used consistently and correctly, changing a term throughout a knowledge base, relating verbs to associated nouns (for example, *extract, extractor*), translating a term into another language, and critiquing and assisting in choice of terms. 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 stress the importance of comprehensive lexical support so that terminology can be carefully chosen and subsequently controlled.

2.3. The DDUCKS Modeling Environment

A rich conceptual model can only be managed successfully within a modeling environment that allows users to view and modify the model in forms that are convenient and natural for them (Norman, 1992). The difficulty is that people are language *inventors* more than mere language users. Because the adequacy of a representation in real-world settings is a function of the individuals involved, the situation, the kind of data available, and, most importantly, the questions being asked, people have become adept at back-of-the-envelope innovation, developing new notations on the fly that suit the task at hand. To be effective, conceptual modeling environments must contain a rich set of visual and problem-solving primitives that can be easily adapted and mapped to the conceptual model for different individuals and situations.

Several research groups are working on approaches that allow experts and knowledge engineers to *configure* systems from a set of representational and problem-solving components (Bradshaw, Ford, Adams-Webber, & Boose, in press; Gruber, in press; Marques, Dallemagne, Klinker, McDermott, & Tung, 1992; Marques, Klinker, Dallemagne, Gautier, McDermott, & Tung, 1991; Neches, Fikes, Finin, Gruber, Patil, Senator, & Swartout, 1991; Puerta, Tu, & Musen, in press; Rappaport, 1991) . The goal is to create sophisticated object-management and end-user-oriented configuration environments that use increasingly more modular and finely-grained components, thus facilitating radical tailorability, embeddability, and principled reuse. In this section, we provide an overview of elements of the DDUCKS architecture that aim to support these objectives.

At The Boeing Company, we are currently developing problem-solving approaches and ontologies that contain concepts useful in formally describing their work on applications in enterprise modeling and integration (Bradshaw, Holm, Kipersztok, & Nguyen, 1992) , design rationale (Bradshaw, Boose, & Shema, in preparation) , and group decision support (Boose, Bradshaw, Koszarek, & Shema, 1992) . Additionally, Bradshaw and his colleagues are formulating an approach to bone-marrow transplant patient support at the Fred Hutchinson Cancer Research Center (Bradshaw, Chapman, & Sullivan, 1992; Bradshaw, Chapman, Sullivan, Almond, Madigan, Zarley, Gavrin, Nims, & Bush, 1992) . The ontologies are being developed within a modeling environment called DDUCKS (Decision and Design Utilities for Comprehensive Knowledge Support) (Bradshaw, Ford, Adams-Webber, & Boose, in press)[4]. CODE4 has been integrated as a back-end implementation of a conceptual modeling representation for DDUCKS. DDUCKS is based on a three-schema architecture and the client-server model (Bradshaw, Ford, & Adams-Webber, 1991; Ford, Bradshaw, Adams-Webber, & Agnew, in press; van Griethusen & King, 1985) . In the future, some portions of the functionality provided by CODE4 will optionally be provided by commercial database products (Figure 6).

[4] Pronounced “ducks”

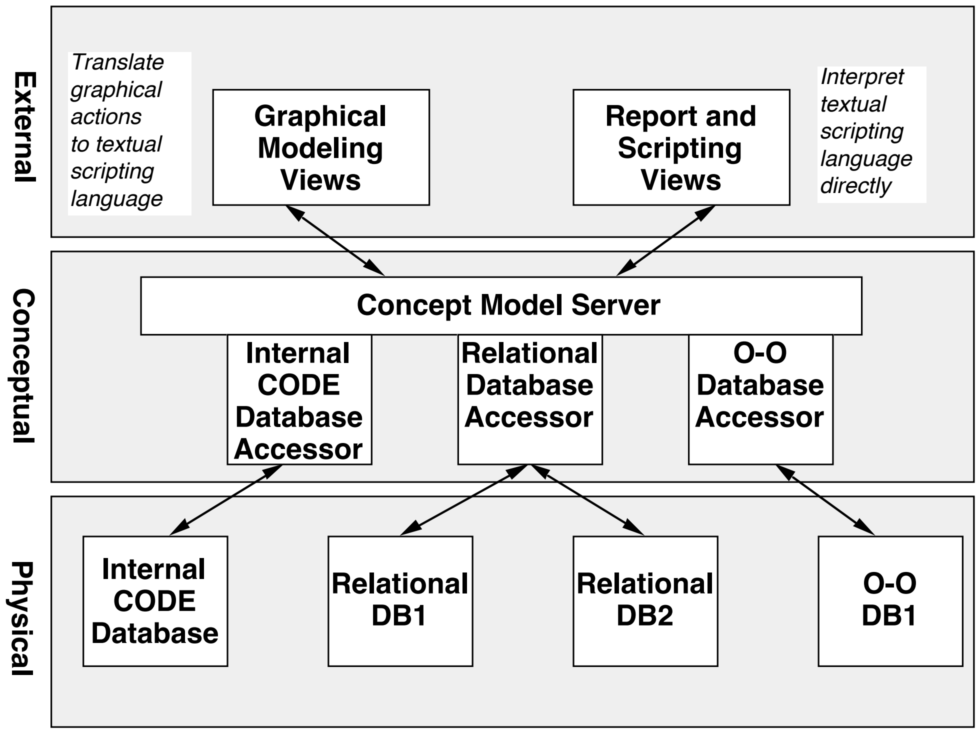
External
Conceptual
Physical
Translate graphical actions to textual scripting language
Graphical Modeling Views
Report and Scripting Views
Interpret textual scripting language directly
Concept Model Server
Internal CODE Database Accessor
Relational Database Accessor
O-O Database Accessor
Internal CODE Database
Relational DB1
Relational DB2
O-O DB1

Figure 6. DDUCKS is based on a three-schema architecture and the client-server model.

We describe some of the features that allow DDUCKS to be easily adapted and mapped to the conceptual model for different groups and situations.

The virtual notebook. Individual views containing particular expressions of a model are laid out onto a *page*. Pages are organized within a *virtual notebook,* which helps individuals 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. A new notebook is typically opened in double-page mode, displaying a page on the right and one on the left as in a paper notebook. The left page typically contains a table of contents view listing the set of pages available in the notebook. The right page might contain a representation for some portion of the knowledge base. Users move from page to page by selecting a “tab” on the side of the notebook or selecting an item in the table of contents view. Alternatively, the user can query the notebook to bring up pages meeting user-defined criteria. Figure 7 shows a virtual notebook in single-page mode.

Figure 7. A virtual notebook showing a medical decision-making framing template.

The concept model., scripts, and filters The CODE4 knowledge base in DDUCKS is called the *concept model*. The concept model provides a means semantic unification for information that may be simultaneously portrayed from a number of perspectives in different views (e.g., grids, concept maps, possibility tables, influence diagrams). In DDUCKS, we have defined an integrated scripting and query language so that the environment can be customized by users. Scripts are used for four things: making queries, modifying the structure and behavior of the concept model and views programmatically, defining computed properties, and defining the behavior of complex filters (explained below).

People observe and operate on the concept model *indirectly* by means of syntactic structures implemented in the views. This indirectness can often lead to situations where we are misled in our expectations about the state of the model (Figure 8)**:**

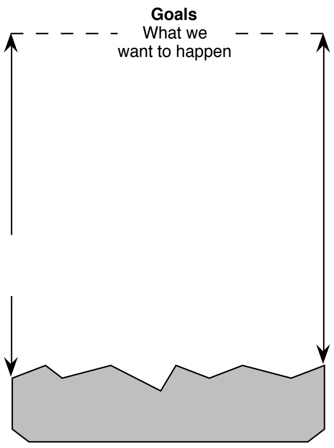

Figure 8. The gulfs of execution and evaluation **(Norman, 1992) .**

> "The gulfs of execution and evaluation refer to the mismatch between our internal goals and expectations and the availability and representation of information about the state of the world [or, in our case, the model] and how it might be changed. The gulf of execution refers to the difficulty of acting upon the [modeling] environment (and how well the artifact supports those actions). The gulf of evaluation refers to the difficulty of assessing the state of the [modeling] environment (and how well the artifact supports the detection and interpretation of that state)…
>
> We can conceptualize the artifact and its interface in this way. A person is a system with an active, internal representation. For an artifact to be usable, the surface representation must correspond to something that is interpretable by the person, and the operations required to modify the information within the artifact must be performable by the user. The interface serves to transform the properties of the artifact's representational system to those that match the properties of the person." (Norman, 1992)

There are two obvious ways to minimize mismatch between the user and the representations embodied in the interface: one is by designing the representations so that they match the user's way of thinking about the model; the other is through mental effort and training on the part of the user to understand the problem in terms of the available representations. Because it is impossible in our setting to assume that users will be willing or able to conform to our representational standard, only the former alternative is really an option.

We attempt to bridge the gulfs of execution and evaluation by providing tools to help users design modeling frameworks and views that fit their personal and organizational requirements. To do this, they must define an explicit specification of how concept model elements, user gestures, and views are mapped to one another (Neches, Foley, Szekely, Sukaviriya, Luo, Kovacevic, & Hudson, 1992) . Our goal is to minimize the effort typically required to write procedural code to tailor representations to user needs—we would like creating a new interpretation model and view to require no greater skill in programming than what a user now needs to know to create a macro or chart within a spreadsheet. Our intent is to provide straightforward means to extend a set of generic models declaratively.[5] Since interpretation models and views are represented explicitly as concepts in the concept model, default parameters can be filled-in automatically by inheritance unless a more specialized value is given.

Our general approach in mapping among interpretations and among expressions of the model is similar in many respects to the tool abstraction approach advocated by (Garlan, Kaiser, & Notkin, 1992) . The goal is to allow functionality to be enhanced incrementally and modifications to be developed independently in situations where traditional data abstraction approaches are insufficient. Spreadsheets and production systems approximate these objectives: spreadsheets can be enhanced incrementally by adding new equations that operate on existing data; in principle, rules in logic-based production systems can also be developed independently and added to the knowledge base. The essential characteristic of systems that support tool

[5] Some knowledge of scripting may be still be needed to create more complex interpretation models and views.

abstraction is that they contain a collection of cooperating *toolies* that provide for mutual notification of changes when shared data structures are modified.

For interpretation model type and each view type, we have defined a set of abstract tools called *filters*. Changes made on a page pass through filters to modify the concept model (Figure 9). Other affected pages are updated through their associated filters. Filter behavior is determined by user-definable scripts, which are executed in response to a change or update request.

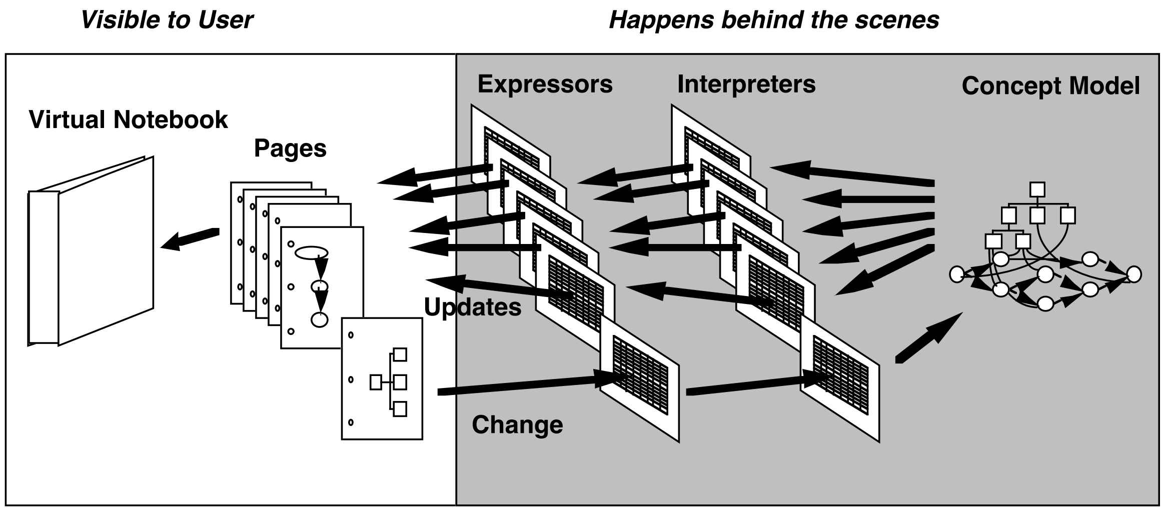

Figure 9. Filters determine how the concept model will be interpreted and expressed in a view. They also determine how actions in a view will be mapped to the concept model.

There are two basic types of filters. *Interpreters* determine which domain concepts are relevant to a view and how those domain concepts (e.g., employee, factory, inspection) will be mapped to interpretation-model-specific concepts (e.g., entity, alternative, activity). *Expressors* determine how and where interpretation model elements are to be expressed as symbolic, syntactical elements within a particular view. Similar kinds of filters determine how actions in the view affect the concept model. New kinds of views can be created on the fly by adding new filters or modifying scripts of existing ones.

3. PRELIMINARY RESULTS IN CREATING SHARABLE ONTOLOGIES

(Alexander, Freiling, Shulman, Rehfuss, & Messick, 1988) introduced ontological analysis as a knowledge modeling technique for the preliminary analysis of a problem-solving domain. Since that time, several groups of researchers have carried the theme onward, emphasizing the central role of ontologies in facilitating knowledge base reusability (Akkermans, van Harmelen, Schreiber, & Wielinga, in press; Gruber, 1992a; Gruber, 1992b; Lenat & Guha, 1990; Neches, Fikes, Finin, Gruber, Patil, Senator, & Swartout, 1991; Skuce & Monarch, 1990) . Well-designed conceptual models can be shared or reused by different tools and applications. For many well-established domains, there is enough similarity between terms and concepts used by various experts that common libraries of ontologies will no doubt flourish. On the other hand, for domains where expert knowledge is highly idiosyncratic, explicit specification of conceptual commitments can promote mutual understanding, even when consensus is neither possible nor desirable.[6]

Besides making knowledge sharing easier, careful design of the conceptual terms and definitions helps developers and users of the system. The result of ontological analysis is a rich conceptual model of static, dynamic, and epistemic aspects of the problem. The model can be extended by designers and users of the system and applied to problem-solving. Our experience confirms that terminological confusion breeds conceptual confusion (and vice versa). Such confusion tends to become more probable the higher one goes in a concept hierarchy. For example, it is usually easier to reach consensus on the meaning of the more specialized terms like *pencil* than on general terms like *object*, or *entity*. Hence the most general concepts are the ones we most need to share, yet also the most difficult to agree on.

We anticipate that research on very general ontologies will eventually result in a consensus, although this may be many years away. More specific ontologies use knowledge from more general ontologies, therefore the closer to consensus the higher-level ontologies can be brought, the more effective will be the sharing of lower level ontologies.

At present, the research community is far from agreement as to what the concepts *are*, let alone what to name them. Our approach is therefore three-fold:

1. Develop and evaluate concept specifications for specific domains of interest and problem-solving approaches;
2. Separately, try to develop a set of top-level, general ontological concepts and terminology for future agreement;
3. Evaluate mechanisms to promote knowledge sharing among similarly-minded researchers.

Different classes of ontologies are likely to be maintained by different people. The developers of a tool are likely to be the ones maintaining the specifications for the essential primitives in the conceptual model; modeling and problem-solving methodology experts will maintain the concepts and algorithms that concern them; user groups will define standards for domains and applications to enhance communication among them; and specific users will maintain concepts in the model that describes their particular situation. A layered approach seems well-suited to this kind of tailoring within a modeling environment.

(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 tool to edit a domain-independent conceptual model. The meta-level tool, PROTEGE, 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. PROTEGE-II generalizes the PROTEGE architecture

[6] See (Ford & Adams-Webber, 1992) and (Ford & Agnew, 1992) for discussions of the dangers of co-mingling models of idiosyncratic experts within the same knowledge base.

to allow for alternate problem solving methods and interface styles (Puerta, Tu, & Musen, in press) .

Our objective in DDUCKS is to increase reusability 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 10)[7]. Starting with any layer in the system, a user can produce a set of tools, models, ontologies, and representations that can be used to assist in configuration of a more specialized system at the layer below. See (Bradshaw, Chapman, Sullivan, Almond, Madigan, Zarley, Gavrin, Nims, & Bush, 1992; Bradshaw, Holm, Kipersztok, & Nguyen, 1992) for specific descriptions of the kinds of information besides ontologies that might exist in a given layer for particular applications.

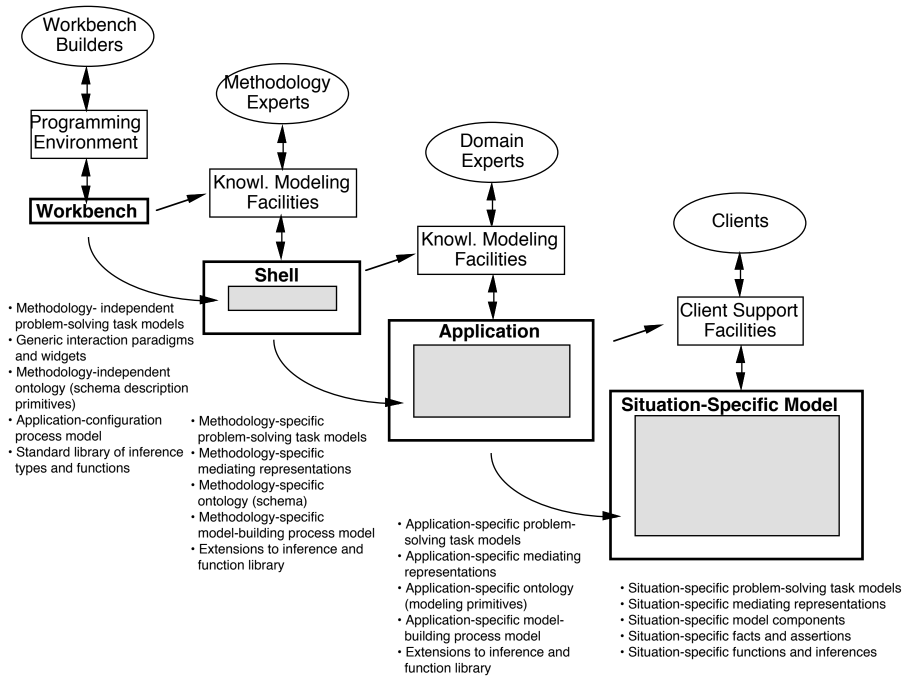
Workbench Builders
Programming Environment
Workbench
Methodology Experts
Knowl. Modeling Facilities
Shell
Domain Experts
Knowl. Modeling Facilities
Application
Clients
Client Support Facilities
Situation-Specific Model
• Methodology- independent problem-solving task models
• Generic interaction paradigms 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 10. Four layers of functionality facilitate reusability in DDUCKS (inspired by figure from (Musen, 1989)).

In the following subsection, we discuss how different classes of ontologies are developed within the tools: top level ontologies with very general concepts, ontologies with modeling concepts, and ontologies for particular domains of application (3.1). Then we explain how *theories, contexts,* and *templates* are used to partition and manage groups of concepts (3.2). Finally, we describe how Ontolingua (Gruber, 1992a; Gruber, 1992b) could be used as a mechanism to make ontology sharing possible (3.3).

3.1. Important Kinds of Ontologies in DDUCKS

---

[7] The four layers are simply a convenient abstraction that seem to apply to a number of applications. In reality, application configuration and tailoring is a continuous rather than discrete process which admits an unlimited number of "layers".

3.1.1. Very General Ontologies

An important area for of work concerns the most general categories that ontologies can have, notions like: thing, object, entity, property, attribute, event, process, state, situation, collection, and relation, to name a few favorites. At the moment, everyone uses these concepts and terms, but: a) they probably use them in very different ways, and b) they cannot tell anyone else what they mean by them. A well known example is the top of the Cyc ontology, which we find difficult to understand from the published descriptions. (Skuce, in pressa) .

We believe that such notions can best be clarified by studying linguistic and psychological data. An important ontology based on linguistic research is the Penman ontology (Bateman & Kasper, 1990) . This ontology, derived from Halliday's studies of English, is well-documented, with reasonable if minimal descriptions of each of some 200 categories. Other linguistically-based efforts include Miller’s WordNet ontology (Miller, 1990) , and the semantic analysis of William Frawley (Frawley, 1992) . Skuce has been working on an empirical approach to very general ontologies for a number of years. The goal is to prepare an initial proposal with approximately fifty categories of the kind listed in the previous paragraph. The current minimal set of conceptual primitives (e.g., concept, property, term) in CODE4 were described in Section 2.2.

3.1.2. Modeling Ontologies

While the domain ontologies are designed to be relatively neutral with respect to a particular modeling framework, the modeling ontologies are meant to capture a particular theoretical or practical point-of-view. In other words, the modeling ontologies characterize the *roles* that domain concepts play within a particular modeling framework. The knowledge acquisition community has a long and successful history in analyzing knowledge roles as they relate to various problem-solving methods and tasks (Marques, Klinker, Dallemagne, Gautier, McDermott, & Tung, 1991; Puerta, Tu, & Musen, in press) . We would like to the analysis of knowledge roles broadened to include not only the abstract function that model elements play computationally in deriving solutions to a problem, but also the roles played in the particular diagrammatic notations describing the model to facilitate communication and *mental* computation (Chandrasekaran, Narayanan, & Iwasaki, in press; Larkin & Simon, 1987) . After all, we have argued above that *insight* may well be the most important outcome of a knowledge acquisition project.

Sometimes a modeling methodology embodies both a solution algorithm and one or more particular graphical notations, as decision analysis typically requires either decision trees or influence diagrams (Howard & E., 1984) . Sometimes a modeling approach consists of a standard interaction paradigm with an application semantics that is largely user-supplied, as in a spreadsheet. Often at The Boeing Company, people use modeling methodologies that merely define a set of standard drawing conventions for informal diagrams, such as process flow charts.

We are performing an analysis of the knowledge acquisition, decision support, and design rationale tools we have developed over the past several years to understand the concepts and assumptions, implicit and explicit, they embody. We have found that the process of developing a formal modeling ontology for our tools is greatly increasing our understanding of them. As we increase our understanding of the ontological commitments of our own framework, we plan to develop and evaluate similar modeling ontologies derived from other perspectives.

In an application of DDUCKS to airplane design and manufacturing, we have developed an ontology appropriate to describing a large, complex business enterprise. The highest levels of the modeling ontology for conceptual modeling are based largely on the work of (Tauzovich & Skuce, 1990) . Within the ontology, a *model* is defined to be the most all-inclusive object of modeling at the highest conceptual level. All other model concepts are either somehow subordinate to the model or external to it. Subtypes of model are used to aggregate the particular types of concepts useful in creating that type of model (e.g., an *enterprise model* representing a company or university; or a *decision analysis model* comprised of information, preferences, and alternatives). Thus a model type is a definition of the kinds of facts of interest to a certain community for a given purpose, and from which particular descriptions can be built. The union of the description and all concepts external to it is called the *universe of discourse*. The model type determines the rules for determining how appropriate domain model components are to be interpreted before they are optionally expressed in a view.

The use of the term *model* to describe something conveys the idea of a dependency between the thing being modeled and the model of the thing. This dependency relation can be recursive, as when a model of something becomes itself a model for something else. Whether or not something is seen as a model also depends on a particular point of view: When we are viewing something as a model in relation to something else, our concern is how well the model characterizes of the things being modeled and the facts known about them. When this is the concern the language used to build the model is not important as long as each user of the model is able to understand the expressions of that language and, if desired, to translate it without loss of information to the language in which that user would prefer to work (Fulton, Zimmerman, Eirich, Burkhart, Lake, Law, Speyer, & Tyler, 1991) .

As a subtype of *concept*, we have defined the root of the hierarchy of primitives of which models are composed: *model concept* (Figure 11). We distinguish two subtypes of model concept: *major model concept* and *minor model concept*. Major model concepts are *model entities, model relationships,* and *model situations;* minor model concepts are *model domains, model properties,* and *model constraints*.

Figure 11. The top of a modeling concept ontology used for an airplane design and manufacturing application..

Model entities are described as follows:

> "An entity is a major concept that represents a class of individual things, persons, places, organizations, projects, etc. which are of relevance to the model. Instances of entities come into existence and cease to exist. An entity has at least one *identifier* (also termed *key property* or *identifying property*) by which instances of the entity are uniquely identified; it may have other properties, some of which may change over time. In natural languages, entities are nearly always named by nouns. The use of the singular for entity names is encouraged." (Tauzovich & Skuce, 1990)

Model entities are typically associated with each other by virtue of the specific roles they play in a common relationship. *Model relationships* are best described as a sort of associative entity:

> "An associative entity [model relationship] is an entity that is introduced to represent a connection between (or among) two or more entities… This connection is being treated as an entity rather than as a relationship, which is done typically to associate or aggregate entities, to eliminate many-to-many relationship, or to allow the connection to have other than solely pre-defined properties." (Tauzovich & Skuce, 1990)

Role properties are used to associate relationships with other concepts. For example, in a *sale* relationship, a role may be defined for a *buyer,* a *seller,* and a *thing being sold.* The particular buyer, seller, and thing sold for that relationship comprise the relationship's *participants.*

Model situations (states of affairs) are used to describe a finite configuration of some aspect of the world (Barwise & Perry, 1983; Sowa, 1984; Sowa, 1992) . A situation takes place or exists at some time and location (except for abstract situations, which have no time, location, or causes. Abstract situations are useful in formal subjects, e.g., the notion of environment in a programming language). Situations have a number of participants. Relationships like cause and purpose can be associated with them. They can usually be denoted by a sentence implying a time and a place. A situation may be relatively static, or it may include dynamic processes and events.

Properties describe an aspect of some other concept (see Section 2.2 above). *Domains* are minor concepts that are used to define a set of values for certain properties (e.g., integers from 1-100; {male, female}). *Constraints* are specifications of relations and values that must be maintained.

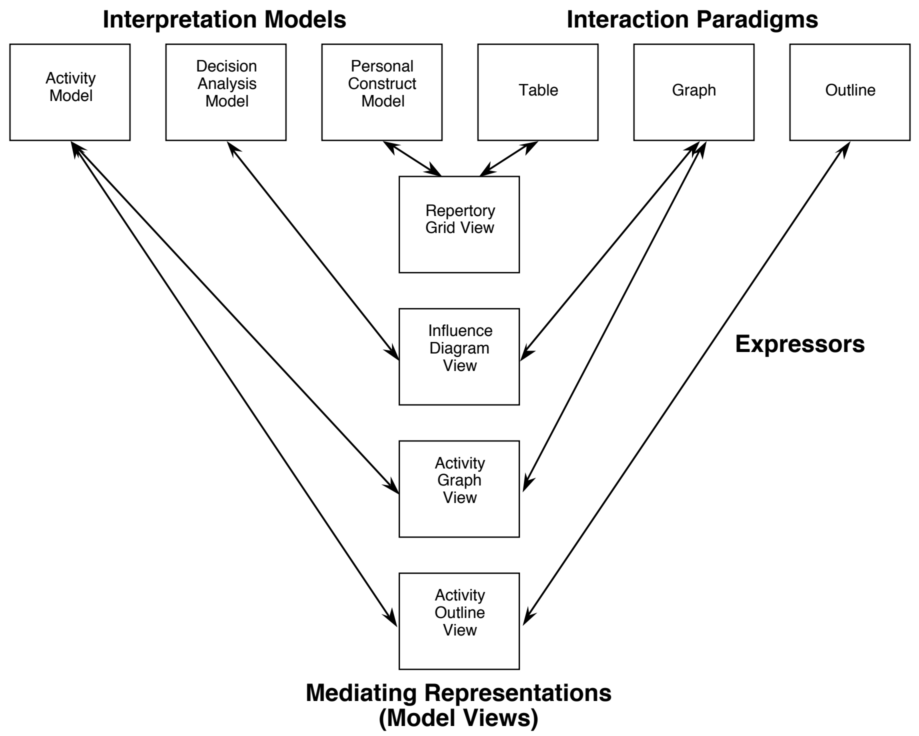

Figure 12. The semantics of a *mediating representation (model view)* are defined by *expressors* that map between a kind of *interpretation model* and particular *interaction paradigm*.

The semantics of a *mediating representation (model view)* are defined by *expressors* that map between a kind of *interpretation model* and particular *interaction paradigm*. At present, concepts that stand for six interaction paradigms have been defined in the ontology (tables, graphs, trees, outlines, lists, and text). The same model type may be expressed by in terms of different interaction paradigms (e.g., *activity models* as *graphs* vs. *outlines*; see Figure 12); conversely, the same interaction paradigm may be reused to create an expression for two different interpretation models (e.g., *graph interaction paradigm* applied to show *influence diagrams* or *activity graphs)*. Subtypes of model views defined in the ontology inherit by default the properties and behavior of their supertypes.

3.1.3. Domain Ontologies

Figure 13. Fragment of a DDUCKS concept model for the airplane design and manufacturing application.

In conjunction with other research groups interested in knowledge sharing issues, we are participating in an effort to develop ontologies for specific domains of interest. For example, at Boeing we have been developing ontological primitives in DDUCKS to represent knowledge of the airplane design and manufacturing enterprise (*eQuality;* see Figure 13) (Bradshaw, Holm, Kipersztok, & Nguyen, 1992) . At the Fred Hutchinson Cancer Research Center, we are interested in representing knowledge about bone-marrow transplant follow-up care (Bradshaw, Chapman, & Sullivan, 1992; Bradshaw, Chapman, Sullivan, Almond, Madigan, Zarley, Gavrin, Nims, & Bush, 1992) . Others have begun projects that will result in ontologies relevant to disciplines such as process planning, software engineering, circuit layout, and device modeling (Gruber, Tenenbaum, & Weber, 1992) .

Domain experts and knowledge engineers jointly define interpretors that map the domain model to an interpretation model. Domain-specific interpretation models and mediating representations may also be defined to fit the needs of specific user groups.

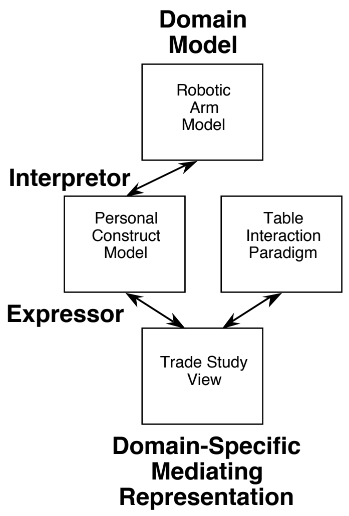

Figure 14. The *domain model* mapped to an interpretation model and expressed in a domain-specific view.

3.2. Theories and Contexts

Practical concerns about the engineering of large systems have led to the development of techniques for creating and operating on multiple partitions of the knowledge base (Gruber, 1992a; Guha, 1991) . Each partition, or *theory*, constitutes a set of axioms tailored for a particular domain or purpose. Each theory is associated with a *context* (McCarthy, 1987) that describes, among other things, the set of assumptions made by the theory.

Within DDUCKS, groups of concepts are organized into theories. Each concept must belong to one or more theories. Theories are semi-permeable partitions of the concept model. Theories themselves are concepts, which allows for relationships to be defined between a theory and other concepts. Individual theories are organized into hierarchies of supertheories and subtheories. By default, a subtheory imports all concepts of its supertheories. Theories can optionally “import” (i.e., “lift”) and “export” concepts selectively.

3.3. A Mechanism for Sharing

The rate of progress on ontological issues in the research community will belargely determined by how well knowledge can be shared. Ontological research is currently shared very little, and where sharing takes place, it is usually either a) in the form of paper reports that take time to distribute and get outdated rapidly, or b) among researchers using some specific piece of not-widely-distributed research software. To facilitate development of ontologies it will be necessary for determine how divers software tools can be facilitated to exchange their knowledge. It will also be necessary to determine *conceptual* formats, i.e. exactly what pieces of information are necessary to convey to accurately communicate ontological concepts among a community of researchers.

Gruber’s work on Ontolingua (Gruber, 1992a; Gruber, 1992b) currently provides the most promising mechanism for sharing ontologies between different tools and formalisms. Ontolingua extends the knowledge interchange format (KIF; (Genesereth & Fikes, 1992)) defined by the

DARPA knowledge sharing effort with standard primitives for defining classes and relationships, and organizing knowledge in object-centered hierarchies with inheritance. Ontolingua facilitates the translation of KIF-level sentences to and from forms that can be used by various knowledge representation systems (currently LOOM, Epikit, Algernon, and a canonical form of KIF). We are working with Gruber to define an Ontolingua interface for CKB, the CODE4 knowledge base file format (Lethbridge & Skuce, 1992a) .

The goal is to use the Ontolingua project as a basis for processing and distributing the ontologies. As good examples become available, the Sharing and Reuse of Knowledge Bases subgroup of the knowledge sharing effort can coordinate evaluations and experiments. In this way, we can pool our results with those of projects with similar goals.

4. CONCLUSION

In order to facilitate sharing of ontologies a number of key research issues must be addressed.

- Knowledge primitives: *Of what primitive elements is the knowledge composed?* We take the approach that all elements are to be considered as concepts· and consistent with the CODE4 system we categorize concepts. into types, instances, statements, properties, terms, metaconcepts, etc. Simply saying that the primitives shall be some set of "predicates" does not answer the difficult question of where we get this set from. This is where linguistic and terminological research becomes especially relevant.

- Standardization: *Should there be only one format or more than one?* The types of knowledge in ontologies developed within the research community may have special properties that differ significantly from knowledge exchanged in other communities. Some may consider these differences mandate different interchange standards. We believe that the differences are not great enough, and the benefits of a single standard are of over-riding importance.

- Inference primitives: *What are the basic semantics of the exchanged knowledge?* For example, in CKB format some basic rules of inheritance are assumed. If somebody attempted to process a CKB knowledge base but assumed different or no inheritance rules, the results of the processing would likely be invalid (or at the very least, substantial knowledge would be lost). We believe it is necessary to have an absolute minimum number of inference primitives otherwise each system will be forced to reimplement every other system. In order to allow the exchange of knowledge between systems that have differing inference primitives, we believe it is important to be able to exchange *informal* knowledge. Such knowledge is only manipulated by systems that recognize it. Informal knowledge can also facilitate human understanding by allowing such things as comments.

- Physical format: Obviously it will be necessary to develop a common syntax, but there are a number of meta-issues upon which to be agreed: Should compactness be a priority? What about human readability? Should the structure be hierarchical or flat? Should it look like predicate logic or some extension thereto such as conceptual graphs (Sowa, 1984)? How are cross-references represented: by unifiable symbols (English words), by indexes, or by pointers?

We have chosen our own positions on the above issues, and so have others. Now the issues have been raised we anticipate a lively debate. However, we hope existing formats for sharing knowledge will begin to accelerate the exchange of ontologies. (grounded in real exchange, real problems)

We hope that our initial efforts at developing ontologies and sharing mechanisms will spur other researchers to make similar efforts, with the goal of increasing knowledge sharing and reuse within the research community.

ACKNOWLEDGMENTS

We express our appreciation to Russell Almond, Miroslav Benda, Guy Boy, Kathleen Bradshaw, Jim Fulton, Cindy Holm, Earl Hunt, Oscar and Sharon Kipersztok, Cathy Kitto, David Madigan, 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 benefited from discussions with numerous colleagues in the knowledge acquisition community over the years, in particular Ken Ford, Brian Gaines, Tom Gruber, Mark Musen, Mildred Shaw, John Sowa, and Brian Woodward.

REFERENCES

Akkermans, H., van Harmelen, F., Schreiber, G., & Wielinga, B. (in press). A formalisation of knowledge-level models for knowledge acquisition. In K. M. Ford & J. M. Bradshaw (Eds.), Knowledge Acquisition as a Modeling Activity New York: John Wiley.

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 (pp. 25-37). London: Academic Press.

Barwise, J., & Perry, J. (1983). Situations and Attitudes, Cambridge, MA: MIT Press.

Bateman, J., & Kasper, R. (1990). A General Organization of Knowledge for Natural Language Processing: The Penman Upper Model, Marina del Rey, CA: USC/Information Sciences Institute.

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 Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW-92),* . Banff, Alberta, Canada:

Borgida, A., Brachman, R. J., McGuinness, D. L., & Resnick, L. A. (1989). CLASSIC: A structural data model for objects. J. Clifford, B. Lindsay, & D. Maier (Ed.), *1989 ACM SIGMOD International Conference on the Management of Data,* . ACM Press: New York.

Bradshaw, J. M., & Boose, J. H. (1990). Decision analysis techniques for knowledge acquisition: Combining information and preferences using *Aquinas* and *Axotl.* 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. (1992). Mediating representations for knowledge acquisition No. Seattle Washington:Boeing Computer Services.

Bradshaw, J. M., Boose, J. H., & Shema, D. B. (in preparation). A knowledge acquisition approach to design rationale. In J. Carroll & T. Moran (Eds.), Design Rationale Hillsdale, N.J.: Lawrence Erlbaum.

Bradshaw, J. M., Chapman, C. R., & Sullivan, K. M. (1992). An application of DDUCKS to bone-marrow transplant patient support. *Working Notes of the AAAI 1992 Artificial Intelligence in Medicine Session of the Spring Symposium,* . Stanford, California:

Bradshaw, J. M., Chapman, C. R., Sullivan, K. M., Almond, R. G., Madigan, D., Zarley, D., Gavrin, J., Nims, J., & Bush, N. (1992). KS-3000: An application of DDUCKS to bone-marrow transplant patient support. *Submitted to the Sixth Annual Florida AI Research Symposium (FLAIRS '93),* . Ft. Lauderdale, FL:

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*, 3(1), 49-77.

Bradshaw, J. M., Ford, K. M., & Adams-Webber, J. R. (1991). Knowledge representation for knowledge acquisition: A three-schemata approach. *Proceedings of the Sixth Knowledge Acquisition for Knowledge-Based Systems Workshop,* . Banff, Alberta, Canada:

Bradshaw, J. M., Ford, K. M., Adams-Webber, J. R., & Boose, J. H. (in press). Beyond the repertory grid: New approaches to constructivist knowledge acquisition tool development. In K. M. Ford & J. M. Bradshaw (Eds.), Knowledge Acquisition as a Modeling Activity New York: John Wiley.

Bradshaw, J. M., Holm, P., Kipersztok, O., & Nguyen, T. (1992). *eQuality:* An application of *Axotl II* to process management. In T. Wetter, K.-D. Althoff, J. H. Boose, B. R. Gaines, M. Linster, & F. Schmalhofer (Eds.), Current Developments in Knowledge Acquisition: EKAW-92 Berlin/Heidelberg: Springer-Verlag.

Chandrasekaran, B., Narayanan, N. H., & Iwasaki, Y. (in press). Reasoning with diagrammatic representations: A report on the AAAI Spring Symposium, March 25-27, 1992. *AI Magazine*.

Clancey, W. J. (in press). The knowledge level reinterpreted: Modeling socio-technical systems. In K. M. Ford & J. M. Bradshaw (Eds.), Knowledge Acquisition as a Modeling Activity New York: John Wiley.

Ford, K. M., & Bradshaw, J. M. (in press). Knowledge Acquisition as Modeling, New York: John Wiley.

Ford, K. M., Bradshaw, J. M., Adams-Webber, J. R., & Agnew, N. M. (in press). Knowledge acquisition as a constructive modeling activity. In K. M. Ford & J. M. Bradshaw (Eds.), Knowledge Acquisition as a Modeling Activity New York: John Wiley.

Frawley, W. (1992). Linguistic Semantics, Hillsdale, N.J.: Lawrence Erlbaum.

Fulton, J. A. (1992). Semantic Unification and the Meaning Triangle No. Boeing Computer Services.

Fulton, J. A., Zimmerman, J., Eirich, P., Burkhart, R., Lake, G. F., Law, M. H., Speyer, B., & Tyler, J. (1991). The Semantic Unification Meta-Model: Technical Approach No. Dictionary/Methodology Committee of the IGES/PDES Organization, ISO TC184/SC4.

Gaines, B. R. (1991). Empirical investigation of knowledge representation servers: Design issues and applications experience with KRS. *SIGART Bulletin*, 2(3), 45-56.

Gaines, B. R., Shaw, M. L. G., & Woodward, J. B. (in press). Modeling as a framework for knowledge acquisition methodologies and tools. In K. M. Ford & J. M. Bradshaw (Eds.), Knowledge Acquisition as a Modeling Activity New York: John Wiley.

Garlan, D., Kaiser, G. E., & Notkin, D. (1992). Using tool abstraction to compose systems. Computer, June, p. 30-38.

Genesereth, M. R., & Fikes, R. (1992). Knowledge Interchange Format Version 3.0 Reference Manual No. Logic Group Report, Logic-92-1). Stanford University Department of Computer Science.

Gruber, T. R. (1991). The role of common ontology in achieving sharable, reusable knowledge bases. In J. A. Allen, R. Fikes, & E. Sandewall (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference (pp. 601-602). San Mateo, CA: Morgan Kaufmann.

Gruber, T. R. (1992a). Ontolingua: A mechanism to support portable ontologies, Version 3.0 No. Stanford Knowledge Systems Laboratory Technical Report KSL 91-66). Stanford University Department of Computer Science.

Gruber, T. R. (1992b). A translation approach to portable ontology specifications. *Proceedings of the Seventh Knowledge Acquisition for Knowledge-Based Systems Workshop,* . Banff, Alberta, Canada:

Gruber, T. R. (in press). Model formulation as a problem solving task: Computer-assisted engineering modeling. In K. M. Ford & J. M. Bradshaw (Eds.), Knowledge Acquisition as a Modeling Activity New York: John Wiley.

Gruber, T. R., Tenenbaum, J. M., & Weber, J. C. (1992). Toward a knowledge medium for collaborative product development. J. S. Gero (Ed.), *Proceedings of the Second International Conference on Artificial Intelligence in Design,* . Pittsburgh, PA:

Guha, R. V. (1991). Contexts: A formalization and some applications No. Technical Report Number ACT-CYC-423-91). MCC.

Guha, R. V., & Lenat, D. B. (1990). Cyc: A midterm report. AI Magazine, p. 33-58.

Haiman, J. (1985). Natural Syntax, Cambridge: Cambridge University Press.

Howard, R. A., & E., M. J. (1984). Influence diagrams. In R. A. Howard & J. E. Matheson (Eds.), Readings on the Principles and Applications of Decision Analysis Menlo Park, California: Strategic Decisions Group.

Jackendoff, R. (1983). Semantics and Cognition, Cambridge, MA: MIT Press.

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

Lakoff, G. (1987). Women, Fire, and Dangerous Things: What Categories Reveal About the Mind, Chicago: University of Chicago Press.

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:

Lethbridge, T. C., & Skuce, D. (1992a). Informality in knowledge exchange. *Working Notes of the AAAI-92 Knowledge Representation Aspects of Knowledge Acquisition Workshop*, 92.

Lethbridge, T. C., & Skuce, D. (1992b). Integrating techniques for conceptual modeling. *Proceedings of the Seventh Annual Knowledge Acquisition for Knowledge-Based Systems Workshop,* . Banff, Alberta, Canada:

Marques, D., Dallemagne, G., Klinker, G., McDermott, J., & Tung, D. (1992). Easy programming: Empowering people to build their own applications. *IEEE Expert*, June, 16-29.

Marques, D., Klinker, G., Dallemagne, G., Gautier, P., McDermott, J., & Tung, D. (1991). More data on usable and reusable programming constructs. *Proceedings of the Sixth Banff Knowledge Acquisition for Knowledge-Based Systems Workshop,* (pp. 6-11). Banff, Canada:

McCarthy, J. (1987). Generality in artificial intelligence. *Communications of the ACM*, 30(12), 1030-1035.

Miller, G. (1990). WordNet: an on-line lexical database. *International Journal of Lexicography*, 3(4).

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.

Mylopoulos, J. (1991). Conceptual modeling and Telos No. Technical Report DKBS-TR-91-3). University of Toronto Department of Computer Science.

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

Neches, R., Foley, J., Szekely, P., Sukaviriya, P., Luo, P., Kovacevic, S., & Hudson, S. (1992). Knowledgeable development environments using shared design models No. USC/Information Sciences Institute and Georgia Institute of Technology.

Norman, D. A. (1992). Cognitive artifacts. In J. M. Carroll (Eds.), Designing Interaction: Psychology at the Human-Computer Interface (pp. 17-38). Cambridge: Cambridge University Press.

Ogden, C. K., & Richards, I. A. (1923). The Meaning of Meaning, New York: Harcourt, Brace, and Company.

Perez, S. (1992). Model unification for data repositories No. Technical Report Number X3H4.6/92-001, Version 0.2). American National Standards Institute (ANSI).

Puerta, A., Tu, S., & Musen, M. (in press). Modeling tasks with mechanisms. In K. Ford & J. M. Bradshaw (Eds.), Knowledge Acquisition as a Modeling Activity New York: John Wiley.

Rappaport, A. (1991). A theory of local and interstitial knowledge. *Proceedings of the Sixth Knowledge Acquisition for Knowledge-Based Systems Workshop,* . Banff, Alberta, Canada:

Reddy, M. (1979). The conduit metaphor: A case of frame conflict in our language about language. In A. Ortony (Eds.), Metaphor and Thought Cambridge: Cambridge University 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.

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

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

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

Skuce, D., & Lethbridge, T. C. (submitted for acceptance). A knowledge representation for interactive knowledge management. *Proceedings of the Third International Conference on the Principles of Knowledge Representation and Reasoning,* . Cambridge, MA:

Skuce, D., & Monarch, I. (1990). Ontological issues in knowledge base design: Some problems and suggestions. *Proceedings of the Fifth Knowledge Acquisition for Knowledge-Based Systems Workshop,* . Banff, Alberta, Canada:

Sowa, J. F. (1984). Conceptual Structures: Information Processing in Mind and Machine, Reading, MA: Addison-Wesley.

Sowa, J. F. (1991). Towards the expressive power of natural languages. In J. F. Sowa (Eds.), Principles of Semantic Networks (pp. 157-189). San Mateo, CA: Morgan Kaufmann.

Sowa, J. F. (1992). Representing and reasoning about contexts. In S. C. Shapiro, J. Barnden, D. Kumar, J. P. Martins, & J. F. Sowa (Ed.), Working Notes for the Propositional Knowledge

Representation Symposium of the AAAI Spring Symposium pp. 133-142). Stanford, CA: Stanford University.

Sowa, J. F., & Zachman, J. A. (1992). Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, 31(3), 590-616.

Tauzovich, B., & Skuce, D. (1990). A general taxonomy for conceptual data modeling No. Cognos.

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

van Griethusen, J. J., & King, M. H. (1985). Assessment guidelines for conceptual schema language proposals No. ISO-TC97/SC21/WG5-3). International Standards Organization.

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

Wittgenstein, L. (1974/1918). Tractatus logico-philosophicus, Atlantic Highlands, N.J.: Humanities Press.