eQuality: An Application of DDUCKS to Process Management

Jeffrey M. Bradshaw, Peter Holm, Oscar Kipersztok & Thomas Nguyen

Computer Science, Research and Technology, Boeing Computer Services
P.O. Box 24346, M/S 7L-64, Seattle, Washington 98124 USA (206) 865-3422;
jbrad@atc.boeing.com

Abstract. Process management is a method for improving Boeing's business processes, however many aspects have been difficult to implement. *eQuality* is a software system based on a framework called DDUCKS that is being designed to support the process management life cycle. We take a knowledge acquisition approach to the development of the tool, emphasizing the importance of mediating and intermediate knowledge representations. Sharing and reuse of tools, models, and representations is facilitated through a layered architecture. *eQuality's* process documentation capability includes a number of views, that can be used either in *sketchpad* or *model* mode. Using the views, an integrated business enterprise model may be developed. Analysis and simulation tools supporting process improvement are implemented with attribute, function, and task editors that make use of a user scripting language and extensible function library. A *virtual project notebook* is used to organize project information and help facilitate group meetings.

1 Process Management at The Boeing Company

The Boeing Company is undergoing fundamental changes in the way it manages its business processes. There are many catalysts for these changes, springing from both internal and external sources — for example, the Boeing Process and System Strategy, the need for concurrent product definition on the new 777 plane, cost management initiatives, CALS, and customer demand for low cost and high quality. Boeing CEO Frank Shrontz [72] has made continuous quality improvement the company's number one objective, affirming that it constitutes "the cornerstone of our business strategy to be the world's leading aerospace company."

In 1988, the company undertook a study of traditional aviation design and manufacturing processes. As a result of the study, the Corporate Computing Board developed a new process requiring *concurrent* design, build, and support activities. While concurrent design, build, and support efforts require significant advances in technology (e.g., 100% 3D CAD digital product definition and preassembly), an equally important challenge is to make the necessary cultural and organizational changes. In the past, processes and organizations remained unchanged when automated support tools were developed. However, new computing applications could do little of themselves to reduce problems of error and rework. Now we are required to document and streamline business, engineering, and

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, Germany: Springer-Verlag, 1992.

manufacturing processes before we consider automating them. This is the only way to avoid automating wasteful practices or implementing obsolete design requirements [47].

Process management is a rubric that encompasses the several methodologies adopted by Boeing for improving business, engineering, and manufacturing processes. To implement process management, the company has formed many process improvement teams, each charged with understanding and streamlining a particular aspect of the business. The teams typically go through the following steps:

- Identify and *document* existing key cross-departmental processes using integrated models of the activities, the items flowing through the activities, and related entities such as organizations and resources.
- Establish points of measurement, then determine how to *improve* the process by minimizing defects, reducing cycle time, and eliminating unnecessary activities.
- Support the *execution* of processes, and monitor performance as part of continuous improvement.

We distinguish *process management* from *process implementation* methodologies and tools. Process management is targeted toward planned, repeatable, but modifiable business processes, regardless of whether automation is being considered. Process implementation methodologies, on the other hand, focus on solving a particular instance of a problem (e.g., creating a specific piece of software, ordering a part, manufacturing a given number of widgets before a particular deadline). They are geared toward successful completion of a unique, one-shot process. Our effort is currently oriented toward supporting process improvement teams; links to implementation methodologies and tools may be addressed in future stages of the project.

In 1989, we surveyed several process improvement teams to determine their current practices and needs. Our findings are summarized in Figure 1, which depicts process management as it is typically implemented. Most teams rely on sticky notes for the early stages of process documentation. Teams track issues and comments manually using large flip charts attached to the walls of the meeting room. Once there is consensus on the a description of the current process, a person who is expert in the use of drawing or CASE software creates a diagram of it. Relatively few teams make it past the process documentation phase. When they do try to measure the process, they use separate analysis and charting programs that are not integrated with the process diagramming tools. Sometimes they must key in information more than once in order to exchange data between different programs.

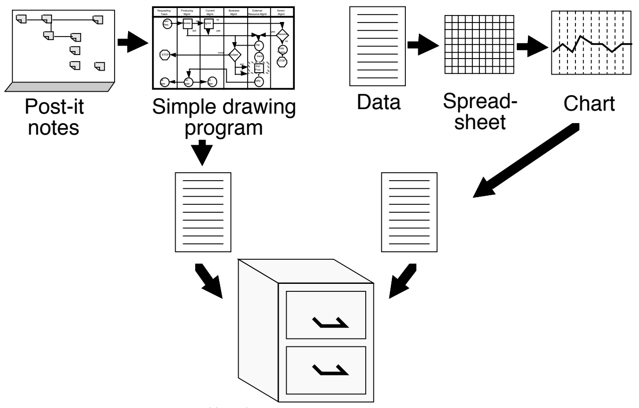

Figure 1. Process management as usually done.

To address these problems, we needed to integrate the functions of an automated process management system (Figure 2). To manage the complexity of enterprise-wide business processes, we need more than thorough documentation—we must have process improvement tools to help us discover how our work can be simplified and streamlined; we must have work-flow execution tools operating on 'live' process models to support our performance of tasks and to facilitate measurement as part of continuous improvement.

To access, share, and reuse models within different tools or for different applications, we need means to translate between them without loss of meaning. A number of standard languages, protocols, and interchange formats are emerging [e.g., 17, 66]. The Semantic Unification Meta-Model (SUMM) is an effort being undertaken by the PDES Dictionary / Methodology Committee [29] to define a formal semantics for such modeling languages. An interface between process management tools and model unification capability based on standards such as the SUMM specification will provide means for data exchange with commercial software (e.g., Excelerator™, IDEF-based tools), internal Boeing tools (e.g., Boeing Flow), and repository management systems. The availability of automated interchange capability will also reduce barriers to active collaboration and sharing between research groups, in the spirit of previous manual efforts such as [60].

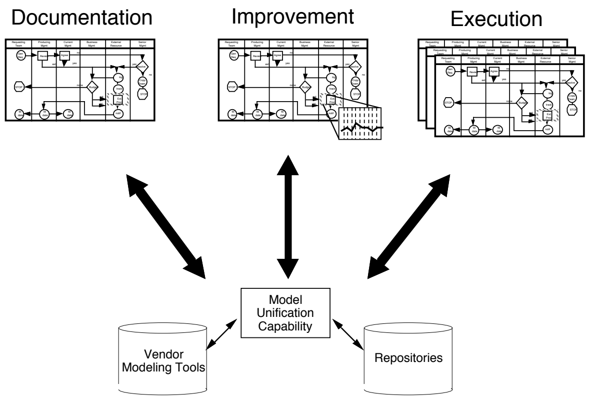

Figure 2. Automated process management support.

Section two will present in general terms how a knowledge acquisition approach can be applied to the development of automated tools for process management. We will discuss the role and importance of mediating representations, a modeling framework for process management, and the architecture of the DDUCKS environment. Section three specifically describes *eQuality*, an application of DDUCKS to problems in process documentation, process improvement, and process execution. Section four presents our conclusions.

2 A Knowledge Acquisition Approach for Process Management

Our approach to process management support systems springs from our many years of work in knowledge acquisition for knowledge-based systems. Over the years, many of our views on knowledge acquisition have changed. We used to think of knowledge acquisition as something that occurred mainly in the early stages of system development. Now we have come to realize that knowledge acquisition tools can assist in formulation, validation, verification, and maintenance throughout the *lifetime* of a knowledge-based system. Thus, it might be said that researchers are attempting to do for knowledge engineering what CASE is attempting to do for traditional software engineering [10, 30, 63]. Indeed, as the scope of application of knowledge acquisition work has broadened, lessons learned from the development of traditional knowledge-based systems have been applied to hybrid systems that combine conventional and knowledge-based components [e.g., 8. 14, 15, 32l. Gaines [31] has suggested the term *knowledge support systems* for knowledge acquisition tools capable of targeting wider applications such as information retrieval, education, personal development, group decision support, and design rationale support. We think that the knowledge acquisition perspective has much to offer for many kinds of problems.

In sections 2.1 and 2.2, we describe some aspects of the knowledge acquisition perspective that have had an influence on *eQuality*. In particular we discuss knowledge acquisition as a modeling activity, and examine the role of mediating representations in the process of model formulation and refinement. Section 2.3 presents the system architecture and explains how it provides for reusability of tools, models, and representations.

2.1 Knowledge Acquisition as a Modeling Activity

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 “knowledge capture” [26]. For this reason, use of the term *conceptual modeling* has begun to replace the term *knowledge acquisition* to describe many of the activities in this field.

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 [27, 50]. 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 [64]. 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 [79]. 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. Modeling does not culminate at some arbitrary point in development, but rather extends throughout the life of the system. It follows that 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 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 conceptual 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 [38].

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 [10]. 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 [49].

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*.

2.2 Mediating and Intermediate 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 [43]. They provide a bridge between verbal data and typical knowledge representation schemes such as production rules [12, 49]. Work on mediating representations for conceptual modeling parallels work on visual programming languages for software engineers [e.g., 40].

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” [49, p. 184]. A crucial feature is that mediating representations should be “easily readable by those who were not involved in the original development programme…” (21, 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.

Work on mediating representations aims to improve the modeling process by developing and improving representational devices available to the expert and knowledge engineer. Several automated conceptual modeling tools have incorporated effective mediating representations [13]. These tools 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 [65] and engineering notebooks in vmacs [55], or they rely on some easily-learned, generic knowledge representation form—for example, repertory grids or directed graphs [7, 23, 28, 37, 52].

Over time the semantic gap between modeling systems and performance systems has widened dramatically. A distinguishing characteristic of some of the newer 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 [9, 23, 34, 35]. As new mediating representations have increased the richness, complexity, and subtlety of the knowledge elicited by automated conceptual 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 conceptual modeling and performance systems, allowing rapid feedback throughout the process of system development [e.g., 36, 71, 59].

Figure 3 depicts a three-schemata approach to knowledge representation [27]. *Mediating representations* serve as external schemata, the *intermediate representation* corresponds to the conceptual schema, and the knowledge base or database implements 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 conceptual 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 (77) provide a good summary of this perspective:

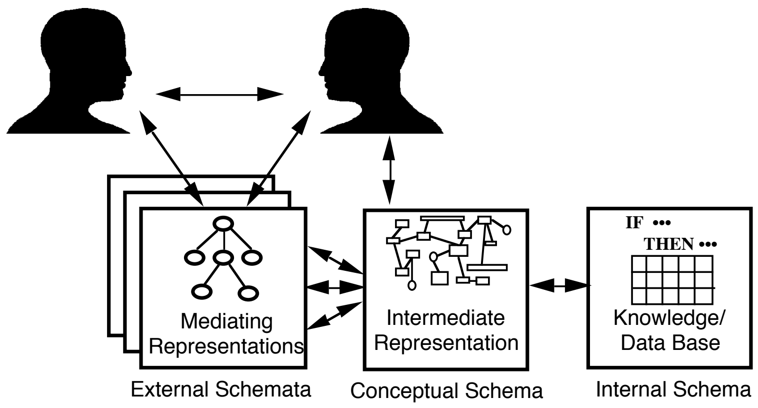

Figure 3. Three-schemata architecture.

> "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.

2.3 An Architecture for Reusability of Tools, Models, and Representations

Because building conceptual modeling tools is labor intensive, their development can usually be justified only if they can be easily applied to more than a single application. Conceptual 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 [6, 53]. Conceptual modeling tools can then be created that both fit the general task model and are tailorable to several specific problems.

Many conceptual modeling tools derive their power from relying on a well defined problem-solving model that establishes and controls the sequences of actions required to perform some task [6, 43, 51, 53]. For example, SALT [61] is based on a method for design called "propose-and-revise", while MOLE [25] uses a method of heuristic classification called “cover-and-differentiate”. More recently, researchers have developed approaches that allow the knowledge engineer to *configure* systems from one or more problem-solving mechanisms [13, 62, 66, 69]. The problem-solving mechanisms define the kinds of knowledge applicable within each step, thereby making explicit the different roles knowledge plays. Having defined these roles, developers can design modeling tools appropriate to each kind of knowledge.

Musen [65] was one of the first to present an explicit, general approach to creating tailorable conceptual modeling tools. Conceptual 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 to allow for alternate problem solving methods and interface styles [68, 69].

Besides the reuse of task models, a number of researchers have emphasized the importance of defining libraries of ontologies, with the goal of increasing knowledge base reusability [45, 57, 66, 73]. Alexander, Freiling, Shulman, Rehfuss, and Messick [4] introduced ontological analysis as a conceptual modeling technique for the preliminary analysis of a problem-solving domain (see also 3, 81]. This kind of analysis results in 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. Well-designed conceptual models can also be shared or reused by different tools and applications.

Our objective is to increase reusability by generalizing Musen’s approach. We have implemented an “open architecture” integrating environment that allows for a high degree of connectivity among hardware and software components. This environment is called DDUCKS (Decision and Design Utilities for Comprehensive Knowledge Support; 14, 15][1]. It is useful to think of DDUCKS in terms of four “layers” of functionality: workbench, shell, application, and consultation (Figure 4)[2] . 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.

The DDUCKS workbench consists of five major elements:

---

[1] Either the first or second *D* in *DDUCKS* is silent, depending on whether one using the tool in a decision or design context.

[2] 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”.

- methodology-independent problem-solving task models (e.g., heuristic classification, constraint satisfaction);
- generic interaction paradigms (see section 3.1 below; e.g., graph view, matrix view, various widgets);
- a methodology-independent ontology (a specification of the abstract conceptual 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 (e.g., *Axotl II)*, created by using the conceptual 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.

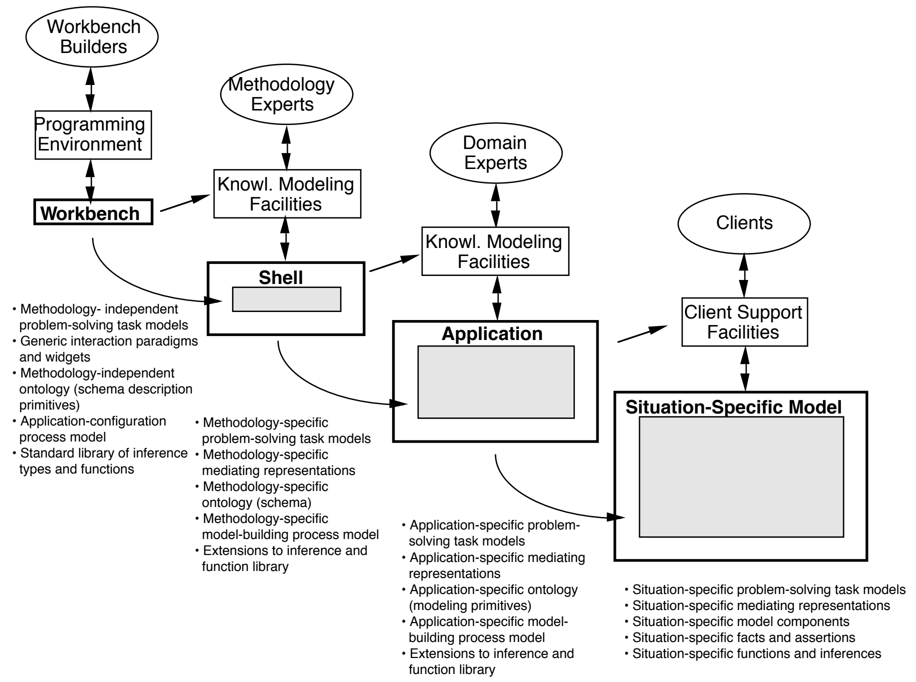
Workbench Builders
Programming Environment
Workbench
Methodology Experts
Knowl. Modeling Facilities
Shell
• Methodology- independent problem-solving task models
• Generic interaction paradigms and widgets
• Methodology-independent ontology (schema description primitives)
• Application-configuration process model
• Standard library of inference types and functions
Domain Experts
Knowl. Modeling Facilities
Application
• 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
Clients
Client Support Facilities
Situation-Specific Model
• 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 4. Four layers of functionality facilitate reusability (inspired by figure from Musen, 1989).

An instance of an application (e.g., *eQuality)*, created by using the conceptual 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.

3 *eQuality:* An Application of DDUCKS to Process Management

eQuality (*e*nhanced *Quality*) is an application ofDDUCKS designed to support the enterprise integration and process improvement through the application of advanced modeling, analysis, and simulation tools. Process management methodologies provide a way to specify design activities and products as part of an *enterprise model*. The enterprise model captures the activities, resources, and organizational context of design from the process owner's point of view. It can also represent models of the structure of the products of design, for analysis and simulation purposes. Figure 5 depicts the components of *eQuality* as a set of project organization and meeting tools and six functional modules.

In the following three sections, we will describe the process documentation, process improvement, and project organization capabilities of *eQuality*.

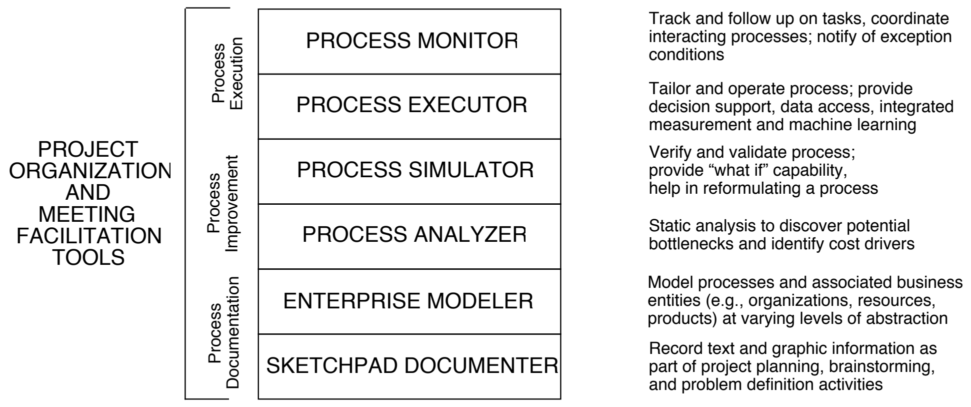
PROJECT ORGANIZATION AND MEETING FACILITATION TOOLS
Process Execution
Process Improvement
Process Documentation
PROCESS MONITOR
PROCESS EXECUTOR
PROCESS SIMULATOR
PROCESS ANALYZER
ENTERPRISE MODELER
SKETCHPAD DOCUMENTER
Track and follow up on tasks, coordinate interacting processes; notify of exception conditions
Tailor and operate process; provide decision support, data access, integrated measurement and machine learning
Verify and validate process; provide "what if" capability, help in reformulating a process
Static analysis to discover potential bottlenecks and identify cost drivers
Model processes and associated business entities (e.g., organizations, resources, products) at varying levels of abstraction
Record text and graphic information as part of project planning, brainstorming, and problem definition activities

Figure 5. Six functional modules and a set of project organization and meeting tools support documentation, improvement, and execution phases of the life cycle.

3.1 Process Documentation.

Figure 6 is a view of knowledge representation in DDUCKS . The intermediate representation in DDUCKS (i.e., the conceptual model) consists of entities, relationships, and situations as the primary concepts, and domains, properties, and constraints as secondary concepts. We are using an enhanced version of CODE version 4 as the underlying semantic representation language [58, 74, 75]. We have derived our general taxonomy for conceptual modeling from Tauzovich and Skuce, with extensions supporting inferencing, analysis, and simulation.

CODE provides a rich, paradigm for the definition of knowledge level concepts. A collection of integrated tools support the important and frequently overlooked aspects of conceptual, ontological, and terminological analysis. Our extensions to the representation allow the system to share several features of Sowa's [76] conceptual graphs, and Gaines' [33] KRS, which interpret taxonomic and entity-relationship structures in terms of typed formal logics. 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 Default facilities for analysis and inferencing over conceptual structures can be augmented by users by means of an integrated scripting and query language.

User-interface management systems (UIMS) are becoming an essential part of interactive tool development and end-user tailoring [48]. We are extending the capabilities of a Smalltalk-80-based direct-manipulation user-interface builder to build a DDUCKS UIMS, called *Geoducks*[3] [56]*Geoducks* relies on the Smalltalk-80 MVC (model-view-controller) concept for managing different perspectives on data ([2, 42, 54]. The MVC approach provides an effective way to factor out the data in an underlying model from the data in dependent views, so that new views can easily be added to an existing model. A sophisticated dependency mechanism assures that changes to the model made within 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 additional capability in *Geoducks,* 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.

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 *eQuality*, various configurations of these interaction paradigms can be called up in *sketchpad mode* 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 created in sketchpad mode to elements in other views in hypertext fashion.

---

[3] Pronounced “gooey-ducks”.

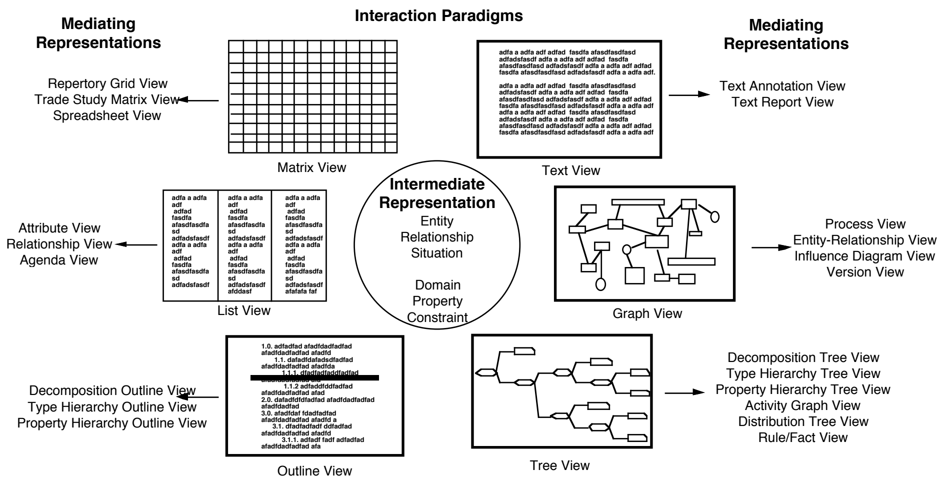

Figure 6. The intermediate representation in DDUCKS, surrounded by examples of generic interaction paradigms, and mediating representations.

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 intermediate representation. 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 built-in ontology. Configured with semantic information, these mediating representations can operate in*model mode,* portraying different perspectives on the formal conceptual model in the intermediate representation. By virtue of the Smalltalk-80 model-view-controller paradigm, consistency is continuously maintained for all model views portraying the same version of the conceptual model.

3.2 Process Improvement

In parallel with development of process documentation tools, we are building analysis and simulation capability supporting process improvement. Simple drawing tools typically available to process improvement teams provide no support for analysis and simulation. Traditional analysis and simulation tools support alternatives analysis and richer models, but require a significant amount of training and data entry to achieve realistic results. *eQuality* is unique in that it addresses the needs of individuals who know a lot about their domain, but do not know very much about formal modeling. People do not have to worry about the simulation when they are creating various diagrams. However when they are ready, the system can use the information contained in the diagrams to support analysis and simulation.

Analysis tools within *eQuality* support the identification of bottlenecks, cost drivers, and the restructuring of processes to exploit concurrency. In addition to formal analysis, built-in knowledge-based system tools can provide support for heuristic analysis. Users can implement analysis metrics such as cycle time, defects per unit of output, and financial parameters using attribute and function editors that make use of a simple scripting language and extensible function library. Using MANIAC, we have developed an initial 'hot link' capability with Microsoft Excel™ that will increase the power and flexibility of the analysis tools.

Discrete-event simulation tools build on the analysis capabilities to provide insight on the dynamic behavior of the enterprise. Users can define active monitors during a running simulation to display results. The monitors selectively respond to changes in the model and dynamically display the results in an appropriate way. For example, a textual event monitor would print out a textual message that described a simulation event, while a graphical monitor such as a histogram or bar gauge might plot the number of occurrences of an event or the value of a parameter.

3.3 Process Execution

An eventual goal is to couple the streamlined enterprise models to the enterprise itself, supplying the semantic transformations that map the models to the enterprise and incorporating feedback from the enterprise concerning the actual execution of the models. We envision integrated process management technology that will someday enable us to move from the current situation where process documentation, if it exists at all, is represented on paper in three-ring binders and control room wall charts; to the near term where models of important processes can be available online in a form amenable to analysis and simulation; to the vision where 'live' process models are woven into the fabric of the way we perform out business. Enterprise models will never be kept up to date properly when they can only be maintained by modeling experts. Enterprise models will never be consistent with the way processes are actually performed until the model actually becomes executable.

We are currently prototyping future possibilities for process execution. To support this, a future release of *eQuality* could produce a form of the enterprise model that can be fed into planning, scheduling, and project management software and linked to relevant data and applications. Process instances could be created each time a process is executed, with status maintained in a repository. Process participants could receive knowledge-based help in carrying out their tasks as the process is executed. An intelligent agent could monitor the activity of the process, notifying process participants of exception conditions and helping to route data associated with the task. Decision analysis capability could help process participants deal with decisions involving high stakes, difficult tradeoffs, or critical uncertainties or risks. Data collected by monitors operating during the execution of the process could be fed back into *eQuality* and used as the basis for further process improvement.

3.4 Project Organization and Meeting Facilitation Tools

Creating a description of an enterprise typically involves the collection, organization, and refinement of a large body of documentation that may include reports, transcripts, glossaries, photographs, diagrams and various representations of formal models. Process improvement team members draw from this evolving corpus as they construct an enterprise model. Effective documentation is more useful during operation of the process before than during the process improvement phase [9]. If they are effectively designed and kept up-to-date, the sketchpad documentation and the enterprise model may later be reused for operations, diagnosis, maintenance and as the basis for improving similar processes in the future. However, the documentation currently produced by process improvement teams is often shallow, scattered, obsolete, incomplete, contradictory, or unintelligible, making maintenance and reuse of the knowledge difficult.

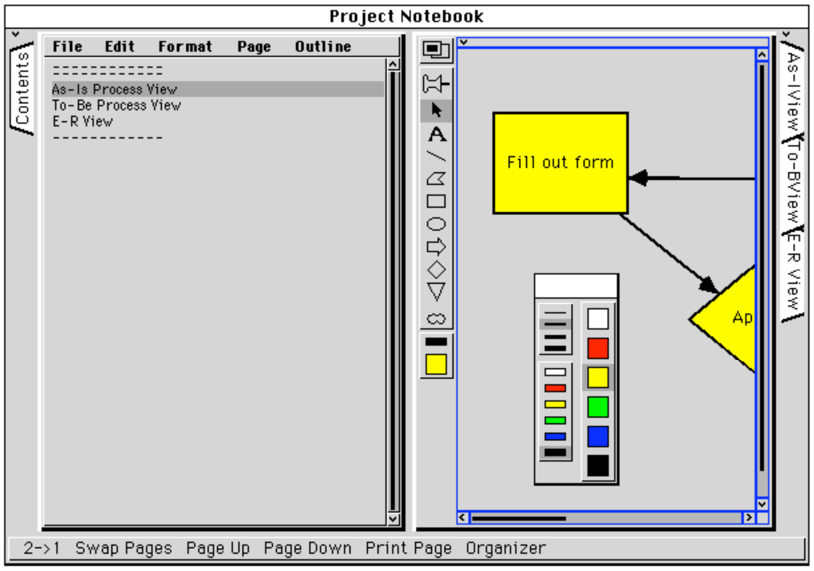

Figure 7. Screen snapshot of the project notebook facility.

The volume and diversity of information that can be represented in *eQuality* drives a requirement for ways to manage, organize, and link that information. A *project notebook* facility helps team members collect and organize the diverse materials associated with a particular process improvement project (Figure 7). 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., default set of concept types and relationships, standard icons and terms for concepts, 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 enterprise model and views, the project notebook supports the team as a simple computer-supported meeting facilitation tool and as a form of group memory.

3.5 Project Status

eQuality was originally implemented within a version of a Boeing-produced shell called *Axotl* [11, 14, 15] *Axotl* was developed on the Apple Macintosh and runs on all platforms that support ParcPlace Smalltalk-80 (e.g., Sun3 and SPARCstations, Apollo workstations, Hewlett-Packard series 300 and 400 systems, IBM '386 compatibles, IBM RS/6000, DECstations). From March to December 1991, a version of *eQuality*, containing sketchpad tools, project organization tools, limited enterprise modeling capability, and a set of prototypical analysis and simulation tools was evaluated at several sites within Boeing. Applications included finance, concurrent engineering, manufacturing, corporate internal audit, continuous quality improvement, and information system requirements analysis. Customers at these evaluation sites have used the software in each of their unique settings, and have provided valuable comments to guide future development directions. Based on results of the evaluation, we designed and developed a completely new version of the *Axotl II* shell as a host for general release of *eQuality* within The Boeing Company. The first general Boeing release of the documentation capability was made in April 1992. Development and evaluation of analysis and simulation capability will follow.

As part of a Boeing project called DIS (Design of Information Systems; Benda, 1991), we explored how knowledge acquisition and decision support tools can work cooperatively with one another and with commercial applications such as spreadsheets, databases, or hypermedia software. We described how such integrated tools could be used for applications such as group decision support in a computer-supported meeting environment [8]. We have developed a facility called MANIAC (MANager for InterApplication Communication) that supports intelligent communication and cooperation between applications. Plans for coordination among applications are modeled and executed using integrated planner 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 has been updated to take advantage of new interapplication communication protocols in version 7.0 (Apple events or Mac DDE for Microsoft applications). An interface to TCP/IP has been built so that we will eventually be able to transparent support for heterogeneous platforms in a networked environment.

4 Conclusions

We attribute much of the initial success of *eQuality* to the knowledge acquisition outlook. In focusing on process management rather than the development of a traditional knowledge-based system, we have seen even more acutely the need for modeling tool developers to attend to the 'acquirability' and reusability aspects of design. 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." [67]

It is our hope that a continued discussion and work on extending knowledge acquisition concepts and tools to additional areas of application will contribute to better communication and shared understanding between participants in system development.

Acknowledgements

We express our appreciation to Mike Anderson, Miroslav Benda, Kathleen Bradshaw, Beverly Clark, Jim Fulton, Cindy Holm, Earl Hunt, William Jones, Sharon Kipersztok, Cathy Kitto, Joe Koszarek, Tim Lethbridge, Allen Matsumoto, Steve Poltrock, Peter Russo, Bob Schneble, Doug Schuler, Kish Sharma, Dave Shema, Bruce Wilson, and Jack Woolley for their contributions and support. We also thank members of Boeing organizations who provided extensive feedback and suggestions on *eQuality* prototypes. This work has benefitted from discussions with John Boose, Guy Boy, Ken Ford, Brian Gaines, Mildred Shaw, and Doug Skuce, as well as numerous colleagues in the knowledge acquisition community. Stan Covington developed and helped conceptualize the first *eQuality* prototype. *Axotl II, eQuality*, and DDUCKS have been developed at the Boeing Computer Services Computer Science organization.

References

1. Adams, S.S. (1988b). MetaMethods: Active values. *HOOPLA!,* 1(1),.3-6.
2. Adams, S.S. (1988b). MetaMethods: The MVC paradigm. *HOOPLA!,* 1(4), July, 5-6, 13-21.
3. Akkermans, H., Van Harmelen, F., Schreiber, G. & Wielinga, B. (1992). A formalization of knowledge-level models for knowledge acquisition. In K. Ford & J. Bradshaw (Eds.), special knowledge acquisition issue of the *International Journal of Intelligent Systems*, in press. Also to appear in K. Ford & J.M. Bradshaw (Eds.), *Knowledge Acquisition as a Modeling Activity*. New York: John Wiley, volume in preparation.
4. 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.
5. Benda, M. (1991). Design of information systems: Towards an engineering discipline. Boeing Computer Services Technical Report. Seattle, WA: Boeing Computer Services, Computer Science Organization.
6. Boose, J.H. (1989). A survey of knowledge acquisition techniques and tools. *Knowledge Acquisition Journal*, 1, 3-37.
7. 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 and B. Gaines (Eds.), *Knowledge Acquisition Tools for Expert Systems*. London: Academic Press, pp. 39-64.
8. 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.

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

10. 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.

11. 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(2): 121-186. Also in J.H. Boose and B.R. Gaines (Eds.), *Progress in Knowledge Acquisition for Knowledge-Based Systems*. London: Academic Press.

12. Bradshaw, J.M. & Boose, J.H. (1992). Mediating representations for knowledge acquisition. Manuscript submitted to the *AAAI 1992 Reasoning with Diagrammatic Representations Session of the Spring Symposium,* Stanford, CA, March.

13. Bradshaw, J.M., Ford, K.M., Adams-Webber, J.R. & Boose, J.H. (1992). Beyond the repertory grid: New approaches to constructivist 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.

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

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

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

17. Burkhart, R., Dickson, S., Hanna, J., Perez, S., Sarris, T., Singh, M., Sowa, J. & Sundberg, C. (1991). IRDS Conceptual Schema Working Paper, October 18.

18. 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.

19. Covington, S.P. & Bradshaw, J.M. (1989). *eQuality* Needs and Alternatives Survey. Seattle, WA: Boeing Computer Services, Computer Science Organization.

20. Daniels, R.M., Dennis, A.R., Hayes, G., Nunamaker, J.F. & Valacich, J. (1991). Enterprise Analyzer: Electronic support for group requirements elicitation. *IEEE* , 43-52.

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

22. Edwards, J. (1991). *Ptech Overview Presentation*. Westborough, MA: Associative Design Technology.

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

24. Eliot, L.B. (1991). Playing the top 20. *AI Expert,* 6(7), 11-12.

25. Eshelman, L. (1988). MOLE: A knowledge acquisition tool for cover-and-differentiate systems. In S. Marcus (ed.), *Automating Knowledge Acquisition for Expert Systems*. Boston, Massachusetts: Kluwer Academic Publishers.
26. Ford, K. & Bradshaw, J.M. (Eds.), *Knowledge Acquisition as a Modeling Activity*. New York: John Wiley, volume in preparation.
27. Ford, K., Bradshaw, J.M. , Adams-Webber, J.R. & Agnew, N. (1992). Knowledge acquisition as a constructivist 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.
28. 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.
29. 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. Report of the Dictionary/Methodology Committee of the IGES/PDES Organization, ISO TC184/SC4. Draft 0.4, September 25, 1991.
30. Gaines, B.R. (1988). Software engineering for knowledge-based systems. *Proceedings of the Second International Workshop on Computer-Aided Software Engineering*.
31. Gaines, B. R. (1989). Design requirements for knowledge support systems, *Proceedings of the Fourth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, October, pp. 12.1-20.
32. Gaines, B. R. (1990a). Knowledge acquisition based on an open-architecture knowledge representation server, *Proceedings of the AAAI-90 Workshop on Knowledge Acquisition: Practical Tools and Techniques*, Boston, July.
33. Gaines, B.R. (1991) Empirical investigation of knowledge representation servers: Design issues and applications experience with KRS. AAAI Spring Symposium: Implemented Knowledge Representation and Reasoning Systems. pp. 87-101. Stanford (March)—also SIGART Bulletin 2(3) 45-56.
34. Gaines, B.R. (1990b). An architecture for integrated knowledge acquisition systems. *Proceedings of the Fifth AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, November.
35. 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.
36. Gaines, B.R. & Rappaport, A.T. (1989). The automatic generation of classes, objects and rules at the interface between knowledge acquisition tools and expert system shells. IJCAI-89 Workshop on Knowledge Acquisition: Practical Tools and Techniques, Detroit, Michigan, August 1989.
37. Gaines, B.R. & Shaw, M.L.G. (1986). Interactive elicitation of knowledge from experts. Future Computing Systems, 1(2).
38. Gaines, B.R., Shaw, M.L.G. & Woodward, J.B. (1992). Modeling as a framework for knowledge acquisition methodologies and tools. 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.

39. Genesereth, M. R. & Fikes, R. (1991). Knowledge Interchange Format Version 2.2 Reference Manual. Logic Group Report, Logic-90-4. Stanford, CA: Stanford University Department of Computer Science, March.
40. Gilnert, E.P. (1990). (Ed.) *Visual Programming Environments: Paradigms and Systems*. Los Alamitos, California: IEEE Computer Society Press.
41. Goldberg, A.T. (1986). Knowledge-based programming: A survey of program design and construction techniques. *IEEE Trans. Software Eng.,* 12, 752-768.
42. Goldberg, A. (1990). Information models, views, and controllers. *Dr. Dobb's Journal,* July, 1-4.
43. Gruber, T.R. (1989). *The Acquisition of Strategic Knowledge*. New York: Academic Press.
44. 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.
45. Gruber, T.R. (1991a). 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.
46. Gruber, T. (1991b). Ontolingua: A mechanism to support portable ontologies. Stanford Knowledge Systems Laboratory Technical Report KSL 91-66. Stanford, CA: Stanford University Department of Computer Science.
47. Hammer, M. (1990). Reengineering work: Don't automate, obliterate. *Harvard Business Review,* July-August, 104-112.
48. Hix, D. (1990). Generations of user-interface management systems. *IEEE Software,* September, 77-87.
49. Johnson, N. E. (1989). Mediating representations in knowledge elicitation. In D. Diaper (Ed.) *Knowledge Elicitation: Principles, Techniques and Applications*. New York: John Wiley.
50. Kaplan, A. (1963). *The Conduct of Inquiry*. New York: Harper and Row.
51. Karbach, W., Linster, M. & Voss, A. (1990). A confrontation of models of problem solving. *Knowledge Acquisition Journal,* in press.
52. Kelly, G. A. (1955). *The Psychology of Personal Constructs*. 2 volumes. New York: Norton.
53. Klinker, G. (1989). A framework for knowledge acquisition. *Proceedings of the Third Annual European Knowledge Acquisition Workshop,* Paris, France, July.
54. 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.
55. Lakin, F. (1990). Visual languages for cooperation: A performing medium approach to systems for cooperative work. In J. Galegher, R.E. Kraut, & C. Egido (Eds.), *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*. Hillsdale, N.J.: L. Erlbaum.
56. Laland, A., Novotny, R., Enzer, S. & Bortz, J. (1991). *The TIGRE Programming Environment*. Santa Cruz, CA: TIGRE Object Systems.
57. Lenat, D.B. & Guha, R.V. (1990). *Building Large Knowledge-based Systems*. Reading, MA: Addison-Wesley.

58. Lethbridge, T.C. (1991). Creative knowledge acquisition: An analysis. *Proceedings of the 1991 Banff Knowledge Acquisition for Knowledge-Based Systems Workshop,* Banff, Canada, October.
59. Linster, M. & Gaines, B.R. (1990). Supporting acquisition and performance in a hypermedia environment. Presentation at *Terminology and Knowledge Engineering Workshop*, Oct.
60. Linster, M. & Musen, M. (1991). Use of KADS to create a conceptual model of the ONCOCIN task. *Proceedings of the Sixth AAAI Knowledge Acquisition for Knowledge-based Systems Workshop*, Banff, Canada, October.
61. Marcus, S. (1988). SALT: A knowledge acquisition language for propose-and-revise systems. In S. Marcus (ed.), *Automating Knowledge Acquisition for Expert Systems*. Boston, Massachusetts: Kluwer Academic Publishers.
62. 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*. Banff, Canada, October 6-11.
63. 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*. Amsterdam: IOS Press.
64. Moore, E.A. & Agogino, A.M. (1987). INFORM: An architecture for expert-directed knowledge acquisition. *International Journal of Man-Machine Studies*, 26, 213-230.
65. Musen, M. A. (1989). *Automated Generation of Model-Based Knowledge-Acquisition Tools*. San Mateo, CA: Morgan Kaufmann.
66. 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.
67. Parnas, D. (1991). The use of formal methods for computer system documentation. Quoted in *Software Maintenance News,* 9(5), May, 29.
68. 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, 6-11.
69. Puerta, A., Egar, J., Tu, S. & Musen, M. (1992). Modeling tasks with mechanisms. 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.
70. 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.
71. 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.
72. Shrontz, F. (1990). Continuous quality improvement. *Manager: A Publication of the Boeing Management Association,* 9(2), March-April, 4-5.
73. Skuce, D. (1991a). A review of ‘Building large knowledge based systems’ by D. Lenat and R. Guha. *Artificial Intelligence,* in press.

74. 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.
75. Skuce, D. (1991c). A wide spectrum knowledge management system. *Knowledge Acquisition Journal,* in press.
76. Sowa, J.F. (1991). Toward the expressive power of natural language. In J. Sowa (Ed.), Principles of Semantic Networks. San Mateo, CA: Morgan Kaufmann.
77. van Griethusen, J.J. & King, M.H. (Eds.) (1985). Assessment guidelines for conceptual schema language proposals (ISO TC97/SC21/WG5-3), August.
78. Webster, D.E., (1988). Mapping the design information representation terrain. *IEEE Computer Magazine,* December, 8-23.
79. Weimer, W. (1979). *Notes on the Methodology of Scientific Research*. Hillsdale, New Jersey: Erlbaum.
80. Wiederhold, G., Finin, T. & Fritzson, R. (1991). KQML: Partial report on a proposed knowledge acquisition language for intelligent applications, September 6.
81. Wielinga, B.J., Schreiber, A.Th. & Breuker, J.A. (1991). KADS: A modeling approach to knowledge engineering. *Knowledge Acquisition* , in press.