An Experiment in Direct Knowledge Acquisition

Peter W. Mullarkey
Schlumberger Laboratory for Computer Science
P.O. Box 200015, Austin, TX 78720-0015, U.S.A.
mullarkey@slcs.slb.com

Abstract

LQMS is a knowledge-based system that identifies and explains anomalies in data acquired from multiple sensors. The knowledge base was built by a sequence of domain experts. Its prototype performed with a high level of accuracy and that performance has been incrementally and significantly improved during development and field testing. Several points are developed in this paper. (1) The combination of an intuitive model (sufficient for the task) and powerful, graphical development tools allowed the domain experts to build a large, high performance system. (2) The Observation-Situation-Relation representation illustrates an intermediate point on the simplicity-expressiveness spectrum, which is understandable to the domain experts, while being expressive enough for the diagnostic task. (3) The system was designed as a workbench for the domain experts. This enticed them to become more directly involved, and, resulted in a better system. (4) The use of an integrated knowledge base edit-tracking system was important to the project in several ways: it reassured computer-naive experts that they could not damage the overall system, which increased their productivity; and, it also allowed experts located in various places around the world to compare, contrast, and integrate changes in a structured way.

Introduction

LQMS is a knowledge-based system that identifies and explains anomalies in data acquired from multiple sensors [O'Neill and Mullarkey, 1989]. This paper describes its knowledge representation and graphical development tools, which allowed direct involvement of several domain experts in the construction of its knowledge base.

Task Description

The task is to enhance the performance of well-trained field engineers in a demanding environment (oil field exploration). Oil-well logs are made by lowering tools into the borehole and recording measurements made by sensors in the tools as they are raised to the surface. The resulting logs are sequences of values indexed by depth. Logging tools measure a variety of petrophysical properties. The field engineer's task involves data acquisition and interpretation, and is characterized by high data rates, with noisy and uncertain data. Earlier work on some aspects of the interpretation problem is discussed in [Smith, 1984]. In this paper, we concentrate on data acquisition. The quality of the data may be affected by many sources: problems with the tools, problems with the downhole environment (*e.g.*, unusual borehole geometry and fluids, extreme formation conditions), and untoward interactions between the tools and the downhole environment. In normal circumstances, it takes three to five years for a field engineer (with an engineering degree) to become truly competent in this task.

The goal of LQMS is to assure that the field engineers leave the field location with full knowledge of the quality of the data collected. Given the task complexity and the level of training required, it was clear from the outset that the knowledge-based system would need to encompass a large amount of knowledge in order to achieve high performance. This makes the acquisition of domain knowledge critical.

Overview of LQMS

LQMS has two main components: a signal-to-symbol module that identifies anomalous behavior in the signal data and an analysis module that applies explicit domain models to arrive at causal explanations for the anomalies. These two modules are implemented in an object-oriented paradigm, and communicate via asynchronous message-passing. The system was implemented in the HyperClass environment [Smith *et al.*, 1987, Schoen *et al.*, 1988] and Common Lisp and runs on both Sun and VAX workstations.

The overall process can be envisioned as a progression of transformations from the raw data, through a signal-to-symbol module, to the analysis module (which can direct the signal-to-symbol module to search for further evidence about features of interest). Our basic design philosophy is to have small, efficient computational agents (the signal-to-symbol module) whose responsibility is only to find anomalous seg-

ments of data, and then use more extensive domain knowledge in the analysis module to discern which anomalies are problems requiring action, and which are explainable in other ways. This decomposition allows the signal-to-symbol module to run real-time, while the analysis module works in background.

A research prototype was deployed over a two-year field test program in several locations (Houston, TX; New Orleans, LA; Cairo, Egypt; Midland, TX; and, Paris, France). It has done an effective job of dealing with the task and providing a model that the users feel is intuitive.

Knowledge Representation Model

The analysis module consists of an inference engine that operates on domain knowledge represented as networks. These networks are composed of Situations that can be used to explain groups of Observations connected by Relations; they are referred to as OSR networks. We will explain via the example shown in Figure 1.

In the example network, Situations are boxed nodes (*e.g.,* LDT-PROBLEM), Observations are unboxed leaf nodes (*e.g.,* RHOB-UNUSUALLY-LOW). Relations are unboxed interior nodes, drawn from the set AND, OR, COMB (a weighted average), NOT, and ANY (essentially an OR that uses breadth-first search).

The example network can be understood (and constructed) from either of two viewpoints.

1. From a symptomatic view, the network has possible explanations for the anomaly RHOB-UNUSUALLY-LOW (highlighted in the middle right section of the network). This observation means that the sensor is measuring a bulk density of the surrounding material that is lower than normally expected, but not so low as to be out of the physical measurement range of the sensor. This anomaly can be explained by a misapplication of the sensor pad to the surrounding rock (LDT-PAD-STANDOFF), allowing lower-density material (*e.g.,* drilling fluid) to exist between the sensor and the material of interest, and/or a problem with the tool that houses the sensor (LDT-PROBLEM).
2. From a "failure model" point of view, there are two subnetworks shown in Figure 1. One describes the observations that would be associated with an LDT sensor pad being out of contact with the borehole wall. The other describes the observations that would be associated with an LDT tool problem.

The various domain experts have found that a choice of different ways of visualizing (and building) the networks allows them to think about their knowledge in the way that is most natural for them. One domain expert prefers the symptomatic perspective, while another prefers the failure model view, and a third seems to be comfortable switching between these views.

The basic components of the knowledge representation scheme were derived by direct observation of the domain experts at work. This recognition that the behavior of the experts is a real expression of the knowledge is discussed in [Musen, 1989] in regard to separating the *knowledge level* from the *symbol level.* The domain experts visually recognize an anomalous data pattern, and attempt to construct a set of conditions that could explain that anomaly. Their "heuristic associations" between anomalies and multiple possible explanations are not through explicit relations, but such relations are implicit in the context. For example, to support an LDT-PAD-STANDOFF there needs to be evidence of both borehole wall roughness (described by the upper subnetwork to the right of the AND in Figure 1) AND an indication of abnormal density borehole fluid (described by the lower subnetwork to the right of the AND in Figure 1). In describing this situation, the AND relation would be obvious to a domain expert from the context. The OSR environment encourages the domain expert to make these implicit relations explicit, both for obvious computability and to make the networks less ambiguous for the next domain expert.

The OSR framework was developed to organize and unify the various concepts that seemed intuitive to the domain experts. These experts clearly had more than a jumble of loosely-connected heuristics, but did not use a complete structural and behavioral model. Our early experience with a rule-based prototype demonstrated that the domain experts had difficulty encoding their knowledge as rules. Furthermore, the prototype did not clearly support the domain experts' model. This was also noted in [Davis and Hamscher, 1988].

We were also led to believe that a "deep model" approach was also not appropriate because:

- No complete model exists for the domain. The lack of a strong domain model is a good reason for using a different approach [Hamscher, 1988, page 19].
- The problem solving must be done in a real-time environment and "deep models" are typically computationally intensive.
- The users do not reason with a deep model; hence, a system using such a model would be less understandable to them.

The inherent simplicity of the OSR framework is a strong asset, since (after three years of successful use) it appears to be expressive enough to cover the domain of interest, while being intuitively comfortable to a set of five geographically distributed domain experts with different backgrounds. Thus, it illustrates an intermediate point on the simplicity-expressiveness spectrum.

Inference

LQMS uses its knowledge base of OSR networks in a way similar to the "cover and differentiate" problem-solving method used in MOLE [Eshelman, 1988]. The signal-to-symbol module notices an anomaly. The

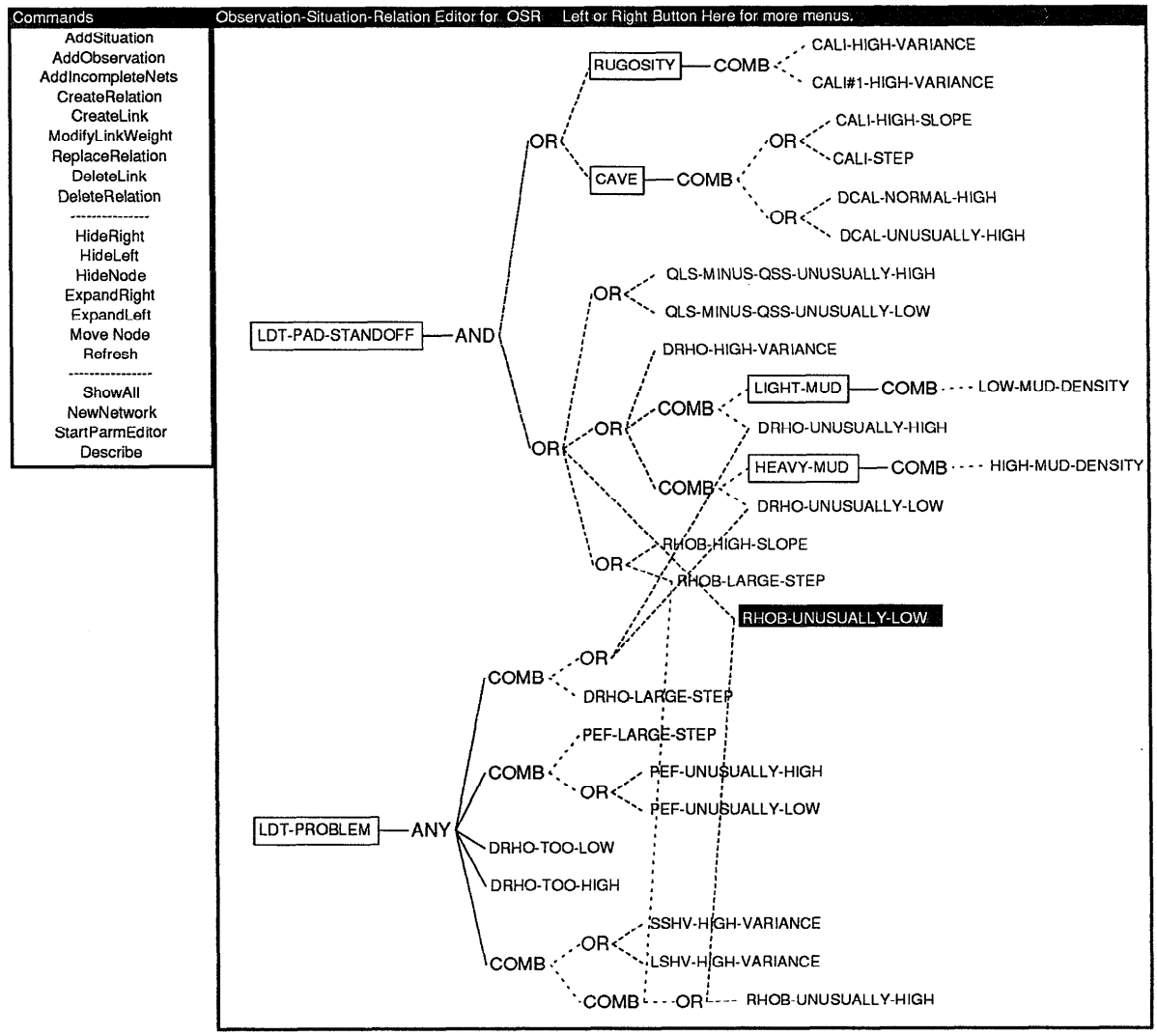

Figure 1: The OSR Network Editor (see text for legend)

analysis module retrieves all situations that can explain the anomaly. It gathers appropriate evidence for discrimination between the possible explanations by traversing the networks. It develops the best explanation(s) for the anomaly from the situations that have the highest belief, based on available evidence. LQMS combines evidence in the OSR networks using an algebra that combines beliefs for the relation types in a conventional manner [Reboh, 1981, Buchanan and Shortliffe, 1984]. Beliefs are represented as a five-valued range with an explicit representation for unknown values. The particular combination rule used is determined by the Relation type. A small set of relations has been found to work well in the LQMS domain.

LQMS can show its reasoning with an animated display of the OSR networks it is processing (highlighting nodes currently being processed, and changing fonts to indicate current belief). The user can also direct the system to explain its conclusions after a session using a mouse-sensitive debriefing report to access the appropriate OSR networks and signal-to-symbol agents. The system's explanation of its results is clear and meaningful to our domain experts. Over the three years of construction, refinement and use, the main debugging tool used has been animated OSR networks.

In addition to the basic framework, the OSR environment offers the domain expert several ways of augmenting the networks with further experiential knowledge.

- `search first` — The domain expert can specify a partial order of situations that are most likely to

explain a particular anomaly.

- `relative link weights` — The links between a relation and its operands can be assigned different weights, thus their evidence will be combined based on the specified relative importance.
- `tie breaking` — The domain expert can designate that some situations be preferred. Then if a "preferred" situation is among the equally believed explanations for an anomaly, it is selected as the best explanation.

Knowledge-Acquisition Tools

LQMS has a rich knowledge engineering environment (based on the HyperClass system). These tools have made it possible for domain experts to enter, test, and refine the OSR network knowledge base (which contains 1880 classes) with minimal assistance from the system developers. Like the knowledge editors in ROGET [Bennett, 1985] and MOLE [Eshelman, 1988], the OSR Network Editor takes advantage of knowledge of the problem-solving method used in LQMS to assist the user in entering knowledge. This approach has been referred to as using a *method-oriented conceptual model* to support knowledge acquisition [Musen, 1989].

The "boxes and links" OSR Network Editor (Figure 1) is the main tool for constructing, maintaining, and refining OSR networks. It has an integrated, object-oriented, edit-tracking system to notice, save (and restore) incremental changes to the knowledge base. This allows multiple domain experts to work independently on the knowledge base, making incremental changes. It supports review, comparison, and conflict detection of various changes proposed for the core knowledge base. Aside from indicating when they start and end editing sessions, the domain experts do not notice the presence of edit-tracking in their normal style of development. The edit-tracking capabilities enhanced both speed and consistency of development and inter-developer communication (discussed in the subsequent section, Support for Distributed Development).

Experiences

This section describes some of our experiences during the construction of LQMS. We begin with a timeline that summarizes the main knowledge-acquisition phase of its development. We then describe two episodes from late in the project that illustrate the ease with our domain experts could understand and modify the knowledge base.

Development

The initial domain expert (who will be referred to as DE_1), had an engineering degree, over 15 years of varied field experience, and much exposure to programming. He worked on the project part-time, helped with the analysis of the domain, and prompted the development of the OSR idea. He was followed, with overlap, by a second domain expert (DE_2), working full-time on the project. DE_2 had an engineering degree, over ten years of varied field experience, and much exposure to programming. DE_2, using the early knowledge-acquisition tools, built the first 80 networks, which represented behavior models for four basic tool types. The knowledge base by then included about 800 total classes. DE_2 transferred back to the field after over a year on the project, and there was a 4 month gap before a third domain expert (DE_3) was transferred in from the field to work on the project. During this time, we developed the OSR Network editor, based on observing DE_2 and seeing that while he was able to construct the networks using the object/slot/value linkage editors, he always kept a current graphic display that would show him the developing structure. The new OSR Network editor allowed these activities to be unified, thus increasing the productivity of future network development. This process of knowledge-acquisition tools being developed contemporaneously with the system is discussed in [Smith, 1984]. DE_3 had an engineering degree, over five years of field experience, and very little exposure to programming or workstations. DE_3 reviewed the state of the system, revised fewer than 10% of the networks, and began adding models of three additional tools. These new tools were quite complex, and involved the addition of 200 networks (and tie-ins to many of the existing networks), over one and a half years. The system used during the final field tests and transferred from research to engineering had models of nine tools, with over 280 networks, including 480 Observations, 280 Situations, and 680 Relations.

Refinement

During the later development phase of the prototype, the system was field tested in five locations worldwide. Each of these field tests involved installing the system in an office environment where engineers would use it to replay previously run commercial data acquisition jobs and provide feedback on the utility, ease of use, and performance of the system. The office environment was selected since the commercial logging acquisition trucks work in a very high-cost, time-critical environment while the drilling rig is configured to allow logging; while in the office, the systems and data are still available after normal work hours. Some of the field test sites provided excellent feedback on how naive users might benefit from the system's analyses, while others provided feedback on how a complex, knowledge-based system would react to local modifications. One of these latter (more technically demanding) locations was Cairo, Egypt, where by that time DE_2 was working. During the field test, in addition to the normal use and testing of the system, DE_2 recognized a weakness in the system's coverage, understood the part of the knowledge base that required refinement, made several changes, and tested and validated

his new (better) results. All this activity transpired without the support of the developers.

Support for Distributed Development

The next episode illustrates how the edit-tracking system acted as a communication vehicle. When LQMS was transferred to engineering, two more domain experts (DE_4 and later DE_5) became part of the team. After a period of electronic communication, DE_3 met with DE_4. They worked on understanding the current state of the system, and discussed refinements that might be made. After DE_3 returned to his home location, he made several hundred changes (of various sizes) since he was still the principal domain expert. DE_4 then asked if he could find out what had been changed. Since all the changes DE_3 had made were recorded by the edit-tracking system, it was straightforward to generate a transcript of those changes. A single change summary is shown that captures adding a link from a situation (`MULTIPLEXER#1-FAILURE`) to an AND relation (`AND-99`). Although the transcript is very low level, DE_4 found it quite helpful.

```
Updating Object MULTIPLEXER#1-FAILURE
Slot ACCOMPANIEDBY to ADD AND-99
```

These transcripts were also used as a chronicle of knowledge base development. This automatic construction of a record of the construction and refinement of the knowledge base deals with one of the inherent problems in knowledge engineering: "interviewing without a record"[Forsythe and Buchanan, 1989].

Performance

The size of a knowledge base and ease of its construction are interesting only when the system performs well. In this section, we summarize several studies of LQMS's performance.

DE_4 performed a set of case studies on the system. The first set was done in 1988 before he had any knowledge of the internals of the system, while the second set was done in 1989 after DE_4 had become familiar with the system. He was not responsible for the knowledge base development until after these studies were completed. In the earlier studies he found that correct diagnoses of causes of anomalies had been far from perfect (60% correct) but with the testing and refinement activities (done by DE_3 primarily), accuracy steadily improved.

In June, 1989, DE_4 ran eight jobs and compiled the following estimated results. He judged approximately 85% of the explanations to be reasonable. In fact, the system found and explained some problems likely to be missed by a normal field engineer. He believed 15% were wrong, since the given explanation could easily be shown to be inappropriate by anyone with a reasonable level of experience in log interpretation.

All of the grading was done on the basis of information that an engineer would normally have available post-hoc, in the office environment. This level of information is not always available to the field engineer, on location, so these results are very encouraging. Explanations classified as wrong were typically those that reasonably fit with a subset of the observations, but conflicted with other available information. This additional information was missed for any of several reasons. It may have been omitted (or improperly included) in the OSR network. It was sometimes inaccessible because no Observers had been designed to monitor that particular data. We believe that most of these wrong explanations can be corrected through further completion of the knowledge base.

LQMS was estimated by DE_4 to be performing at 85% of the level of an experienced field engineer, and had shown that its performance can be improved incrementally and substantially.

Current Status

The prototype LQMS was transferred to engineering in 1989 after three years of development and field testing. The system is being ported into the commercial environment of the data acquisition system.

Related Work

The idea of using networks to structure the association of observations and situations has some of its roots in Hendrix's partitioned semantic networks [Hendrix, 1979].

The basic architectural approach of abstraction of data by low-level modules with higher-level processing (sometimes directing the low-level modules) is very much like Hearsay-II [Erman *et al.*, 1980]. The MOLE [Eshelman, 1988] system has many similarities to the OSR framework used in LQMS. MOLE has more powerful support for refinement of the knowledge base, although the OSR Network Editor has the capability to discover and display (partial) networks it considers incomplete (Figure 1—*AddIncompleteNets* command). The domain expert has played a more active role in refining the knowledge base in LQMS (with good results). Based on the examples shown in [Eshelman, 1988], MOLE has a less expressive knowledge representation, in that there is only one "relation" called "*covering (explanatory)*", which is visually encoded as a link. Additionally, there is a presumption that *initial symptoms* are different in a significant way from *other symptoms* and that only initial symptoms can be used to prompt a diagnosis. In contrast, OSR has the concept of a *trigger observation*, but this is just an additional characteristic that any observation can have, although intuitively, some observations (*e.g.*, value-out-of-range) are much more likely to be interesting triggers in a diagnosis task. We designed this uniformity into the observations since we believe that if a knowledge base describes a domain, there may be several uses for that knowledge (*e.g.*, data quality diagnosis, pre-interpretation data analysis, training), and

the trigger observations for the different uses may not be the same.

There is a problem inherent with the development of ambitious systems: keeping the pace of development fast enough to maintain the domain experts' interest and contribution [Buchanan *et al.*, 1983]. One approach is to use a team of developers, suggested by Reboh [Reboh, 1981, p. 94], one to interview the domain expert and another to implement the knowledge in a prototype. The approach used in LQMS brings the domain expert directly to the center of the action and responsibility. This approach has the additional feature of making the domain expert a real part-owner of the system, with all the positive aspects that "ownership" brings.

Conclusions

LQMS is a system based primarily on knowledge encoded directly by domain experts. Its prototype performed with a high level of accuracy and that performance has been incrementally and significantly improved during development and field testing. Several points have been developed in this paper.

- The combination of an intuitive model (sufficient for the task) and powerful, graphical development tools allowed the domain experts to build a large, high performance system. There was a significant transition from the prototype system based on rules (no direct interaction between domain expert and knowledge base) to the system based on OSR networks built with direct "boxes and links" editing (very productive interaction).
- The OSR representation illustrates an intermediate point on the simplicity-expressiveness spectrum, which is understandable to non-developers, while being expressive enough for the domain.
- The system was a natural workbench for the domain experts, which enticed them to become more directly involved, resulting in a better system.
- The edit-tracking system served many useful purposes. It reassured computer-naive experts that they could not damage the overall system, which increased their productivity. It also allowed experts located in various places around the world to compare, contrast, and integrate changes in a structured way.

Acknowledgments

There have been many people associated with the successful history of the LQMS project without whom these lessons would not have been possible. With respect to the knowledge-acquisition aspects, I would like to acknowledge the efforts of our excellent domain experts: Alistair Cox, Ace Dumestre, Laurent Moinard, and Alan Sibbit. The core development team included Dennis O'Neill, Paul Gingrich, and the user interface work of Ruven Brooks. Although this paper primarily describes work on the research prototype, the engineering team at Schlumberger Austin Systems Center, involved in the technology transfer, provided valuable interactions. Bob Young was a critical resource during the original development of the OSR ideas, provided the spark of motivation to produce this paper, and excellent feedback to make it right. Eric Schoen and Reid Smith supplied insightful comments and helpful pointers. Stan Vestal and SLCS allowed this retrospective activity to occur.

References

[Bennett, 1985] James S. Bennett. ROGET: A knowledge-based consultant for acquiring the conceptual structure of a diagnostic expert system. *Journal of Automated Reasoning*, 1:49–74, 1985.

[Buchanan and Shortliffe, 1984] B. G. Buchanan and E.H. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, Mass., 1984.

[Buchanan *et al.*, 1983] Bruce G. Buchanan, David Barstow, Robert Bechtal, James Bennett, William Clancey, Casimir Kulikowski, Tom Mitchell, and Donald A. Waterman. Constructing an expert system. In Frederick Hayes-Roth, Donald A. Waterman, and Douglas B. Lenat, editors, *Building Expert Systems*, chapter 5, pages 127–168. Addison-Wesley, Reading, Mass., 1983.

[Davis and Hamscher, 1988] Randall Davis and Walter Hamscher. Model-based reasoning: Troubleshooting. In H.E. Shrobe and AAAI, editors, *Exploring Artificial Intelligence*, pages 347–410. Morgan Kaufmann, 1988.

[Erman *et al.*, 1980] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213–253, June 1980.

[Eshelman, 1988] Larry Eshelman. MOLE: A knowledge-acquisition tool for cover-and-differentiate systems. In Sandra Marcus, editor, *Automating Knowledge Acquisition for Expert Systems*, chapter 3, pages 37–80. Kluwer Academic Publishers, Boston, Mass., 1988.

[Forsythe and Buchanan, 1989] Diana E. Forsythe and Bruce G. Buchanan. Knowledge acquisition for expert systems: Some pitfalls and problems. *IEEE Transactions on Systems, Man and Cybernetics*, 19(3):435–442, May/June 1989. Special issue on perspectives in knowledge engineering.

[Hamscher, 1988] Walter C. Hamscher. *Model-based Troubleshooting of Digital Systems.* PhD thesis, MIT AI Lab, 1988.

[Hendrix, 1979] G. G. Hendrix. Encoding knowledge in partitioned networks. In *Associative Networks—The Representation and Use of Knowledge in Computers*, pages 51–92. Academic Press, New York, NY, 1979.

[Musen, 1989] Mark A. Musen. *Automated Generation of Model-Based Knowledge-Acquisition Tools.* Research Notes in Artificial Intelligence. Pitman Publishing, London, 1989. Revision of Stanford University PhD dissertation (STAN-CS-88-1194).

[O'Neill and Mullarkey, 1989] D. M. O'Neill and P. W. Mullarkey. A knowledge-based approach to real time signal monitoring. In *Proceedings of the Fifth Conference on Artificial Intelligence Applications*, pages 133–140, March 1989.

[Reboh, 1981] R. Reboh. Knowledge engineering techniques and tools in the Prospector environment. Technical Report 243, SRI International, Menlo Park, Calif., June 1981.

[Schoen *et al.*, 1988] Eric Schoen, Reid G. Smith, and Bruce G. Buchanan. Design of Knowledge-Based Systems with a Knowledge-Based Assistant. *IEEE Transactions on Software Engineering*, 14(12):1771–1791, December 1988.

[Smith *et al.*, 1987] R. G. Smith, P. S. Barth, and R. L. Young. A substrate for object-oriented interface design. In *Research Directions in Object-Oriented Programming.* MIT Press, Cambridge, MA., 1987.

[Smith, 1984] R. G. Smith. On the Development of Commercial Expert Systems. *AI Magazine*, 5(3):61–73, Fall 1984.