Harnessing Detailed Assembly Process Knowledge

William J. McClay and John A. Thompson

Boeing Computer Services Electronics Support
P.O. Box 24346, MS 7L-45
Seattle, WA 98124-0346

Abstract

The Connector Assembly Specifications Expert is an AI system in use at Boeing which advises personnel in the proper assembly of electrical connectors. The information delivered is in strict accordance with the complex set of Boeing process specification documents. The system is written entirely in Prolog and has more than 25,000 rules contained in over 300 knowledge base files. A companion knowledge acquisition system written in Lisp and Knowledge Craft is in development, and a uniform programmatic query interface is being designed to make the knowledge bases accessible to any computer system in the company.

Introduction

The Connector Assembly Specifications Expert System, CASE (pronounced Casey), is an AI system which has been in use at Boeing in a production mode since October, 1986. It advises engineering, manufacturing, and field service personnel in the proper assembly of electrical connectors and other electrical terminations which require special tools for assembly.

Detailed assembly instructions, including graphics, are printed at the user's terminal at the end of a typical consultation; these provide just the information needed to build a particular device for a particular program (e.g. 747, B-52, Minuteman). The information delivered is in strict accordance with the Boeing process specifications, which are contained in the Boeing Corporate Standards, various program standards, and a large number of military standards, and whose combined volumes contain over one hundred thousand pages.

Although the number of pages of specification data used for electrical assembly is probably no more than twenty thousand, the process of finding the necessary information is far from simple. The specification search time for a person has been measured by industrial engineering at an average of forty two minutes.

The system of documents to be searched can be likened to the building codes which apply to a contractor. There are federal laws, state laws, county laws, and city laws which all have to be considered before the contractor starts construction. Certain state laws may take precedence over federal or local laws, and various aspects of construction will be found in different books. But all laws must be considered together, applying those which take precedence. From these, a set of guidelines must be assembled which will guide the construction.

The user of the standards documents faces essentially the same, complex situation. First, all the pertinent bits and pieces of information must be collected from all the various documents and sections which *might* apply to the assembly at hand, then the instructions and options which *do* apply must be sorted out and resolved. Finally, of all the tools and materials which could be used, the researcher must decide which tools are available to the assembler and which methods will be most cost effective for that shop.

It requires many years of experience to be familiar enough with the standards system to know where to look for all the relevant information and not be misled by what often appears to be conflicting statements in different sections. The ability to resolve such conflicts and provide clear and concise instructions to the shop is a rare and highly valued skill possessed by a small number of "shop experts."

One previous system which attempted to automate this task collected all the bits and pieces which might apply, but was not able to eliminate irrelevant information. It produced reports as long as 50 or 60 pages for a single connector. Another attempted system managed to reduce the amount of irrelevant information, but the knowledge about the organization of the specs was distributed throughout the Fortran code which accessed the database.

This presented a considerable maintainance problem, since specs change frequently and sometimes radically.

In late 1984, when this problem was once again brought up by Boeing Electronics factory management as lacking a good solution, it was suggested by our Manufacturing Research and Development Organization that Artificial Intelligence might be able to contribute something towards a better solution. It was clear that if AI were going to provide any help for this problem, it would be in the area of representing the standards knowledge in such a way that it would be very easy to capture, maintain, and reason about.

System Design

There are only a small number of basic questions which any particular connector assembly standard is intended to answer, for example:

"What types of connectors are covered in this document?"

"What is the value of attribute X for connector Y?"

"What contact part numbers can be used in connector X?"

"What wire sizes can be used with contact X?"

"What tools, materials, and assembly steps are required to assemble wire size X into contact Y?"

"What tools and procedures are required for inserting contact X into the connector shell?"

"What sealing is required for unused connector cavities?"

"Are there any special problems in assembling this connector?"

It seemed natural that these questions should serve as the interface between the basic consultation control mechanisms and the standards knowledge base. This would eliminate the need for the program using the knowledge to know anything about internal table structures, precedence relations between specs, or how to interpret any lower level data items, which had been the failings of the earlier database oriented attempts. These "interface questions" would also provide a basic framework of goals which would take certain input values and, through some reasoning process, provide direct answers without any "ifs" in them.

This type of organization seemed to suggest some type of intelligent database which could ask questions when necessary to eliminate the "ifs" and return a set of values for each solution found. Forward chaining systems did not seem to fit, since the user must have *all* solutions to a given question returned on demand. Exhaustive backward chaining seemed to be more appropriate for this purpose and did not seem frightening in terms of search time, since the standards themselves limit the depth and breadth of search by explicitly referencing supplemental documents when needed. Certainty factors were not needed, since process specifications are very definite about what is allowed and what is not.

System Implementation

The first prototype system was built using a PC based expert system building tool and proved to be very satisfactory with respect to the intelligent database idea. However, a number of problems surfaced during this exercise. The first was that processing time during a consultation was close to ten minutes, which was twice as long as expected. The next problem was that the system was required to retain certain information between consultations. Although there was a solution for this, it was anything but straightforward using this tool.

Another problem was that it seemed desirable to have a separate knowledge base for each standards document, so that each one could be maintained independently of the others, but this was not supported. Also, it was inevitable that this knowledge based system, if successful, would be widely used throughout the corporation and would have to run on some sort of mainframe computer system. It was evident that progressing beyond this early prototype would be painfully slow unless a better tool could be found.

An evaluation was made of virtually all the expert system shells and languages available at that time which supported the declarative approach. Maximum flexibility, computational efficiency, and transportablility were weighted heavily. Since Prolog was similar to the syntax of the PC based tool, it was tried as the new implementation language.

The effort to convert the prototype to Prolog took only a couple of days and the results were impressive. The performance was two orders of magnitude better, and it was now running on a VAX. There was no problem breaking up the knowledge base into separate chunks for each standard and consulting them when needed. Keeping certain consultation parameters around for more than one consultation was also not a problem.

On the down side, it took a couple of months to replace those nice user interface and consultation control features

provided by the expert system tool. However, this was viewed as an opportunity to provide a custom look and feel for the system which could be extended as needed. User interface utilities were developed to provide such features as auto completion for user inputs and automatic menu generation from a list of items. Though still lacking in sophistication, the new "shell" was functionally adequate for continuing the project.

The next big challenge was only beginning to be fully appreciated. The problem was that each knowledge base had to be an intelligent agent, able to interpret information being returned by other knowledge bases and selectively accept, reject, or replace any piece of information in the answer being returned. This required some way of labeling each piece of the answer at the proper granularity so that it could be easily reasoned about at some higher level.

This process of filtering knowledge was helped by the use of Prolog structures. For example, each text note can be represented by a note structure such as: note('BAC 5162-9', 'Strip insulation 11/16 inch and double back conductor on itself before crimping', [strip, double_back], bac5162_9fig8). This note structure identifies the text of the note as having come from document BAC 5162-9 and it labels the contents as discussing both the stripping and doubling back of the conductor before crimping. The note also references a figure 8 which illustrates the operation.

These structures are easily recognized and manipulated by the pattern matching capabilities within Prolog, which facilitated the writing of rules to deal with them. For instance, if in a higher precedence spec the use of a filler wire is specified instead of doubling the conductor back, the note covering the "doubling back" is easily identified in the instruction list and replaced with the "filler wire" note along with another note to cover the stripping requirements.

Compared to a typical database system, the freedom and expressiveness of a symbolic language such as Prolog was a welcome relief and most likely a critical factor in the success of the project. Connectors vary considerably, and representing the knowledge about how they are assembled requires a flexible and expressive knowledge modelling language. Extensive use was made of the built-in operations on lists and the very powerful pattern matching capabilities of special data structures like the note structures just mentioned.

As illustrated in Figure 1, the system architecture features a "knowledge base network" concept. The bold boxes and arrows represent the passing of information

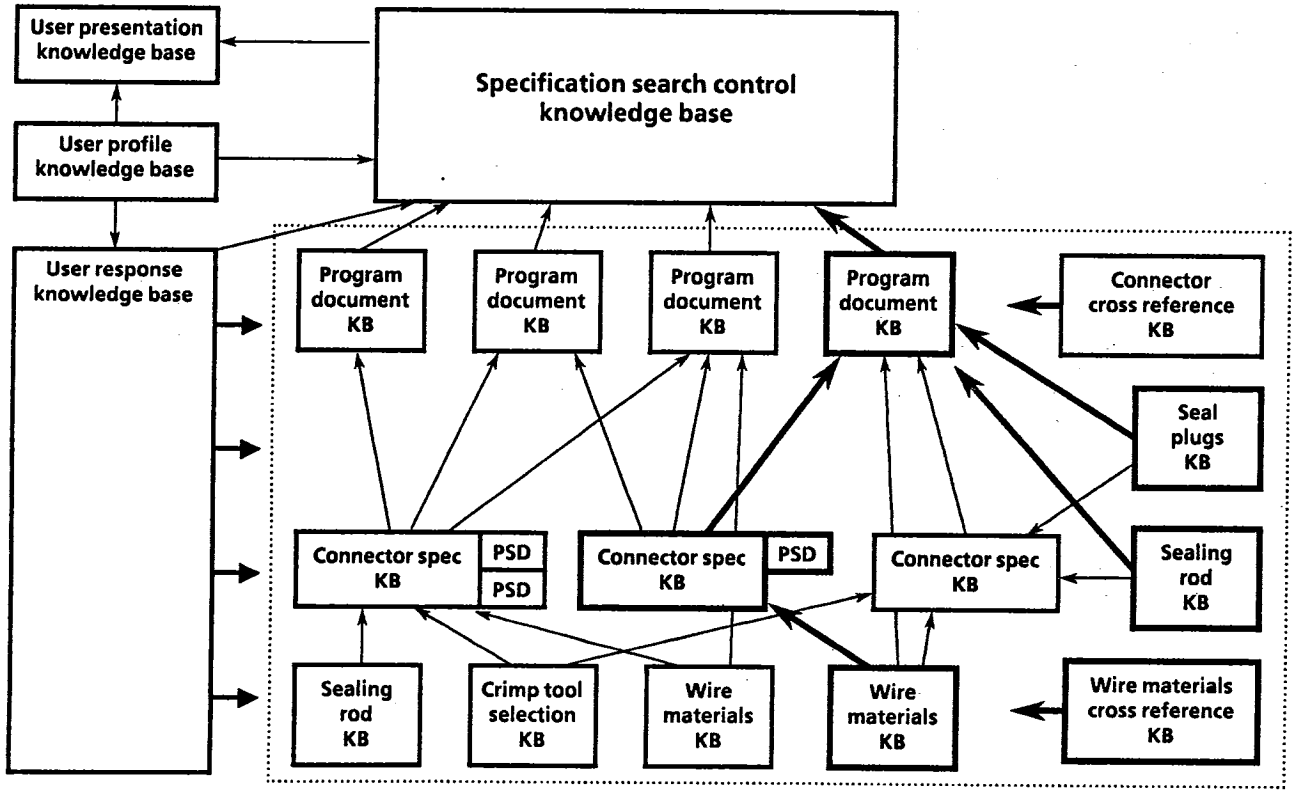

Figure 1 - CASE Knowledge Base Network

from knowledge base to knowledge base until the answer to the query is finally assembled and returned to the Search Control knowledge base. Once all of the possible assembly methods are collected, Search Control selects the optimal choice or choices using various parameters contained in the user's profile, and a report is printed on the user's local printer.

System Validation

An early prototype was installed in the shop in late 1985 and was well received as being responsive, friendly, and very easy to learn to use. Over the period of the next year a considerable amount of knowledge was captured in knowledge bases and enthusiasm for its capabilities and potential cost savings was widespread.

Despite this success, other obstacles had to be overcome before this technology could be widely deployed. When it was decided to have a formal acceptance test in October of 1986, a test team selected ten test cases and proceeded to do the research for each of them. There was no advance notice given about those tests, only that CASE had the required knowledge bases to complete the consultations.

In six of the test cases, both CASE and the test team came up with the same results. In two others CASE contained typographical errors which were obvious to those performing the test and would not have misled users. In the last two cases the test team missed some obscure Process Specification Departure (PSD) and indicated a tool and procedure which was not actually the latest information according to the standards.

This satisfied manufacturing that CASE could be trusted for building hardware, and this was the beginning of its regular use. However, CASE was not considered to be engineering authority and quality control would still have to use the paper system for their final check that everything was done correctly. All that could be claimed was that a CASE report was probably more reliable than a spec search by an experienced researcher.

Knowledge Acquisition

In order for CASE to have engineering authority, it would be necessary for those engineering organizations currently responsible for the process specs to actually produce and maintain the knowledge bases. No amount of testing would ever insure that the hand coded CASE knowledge

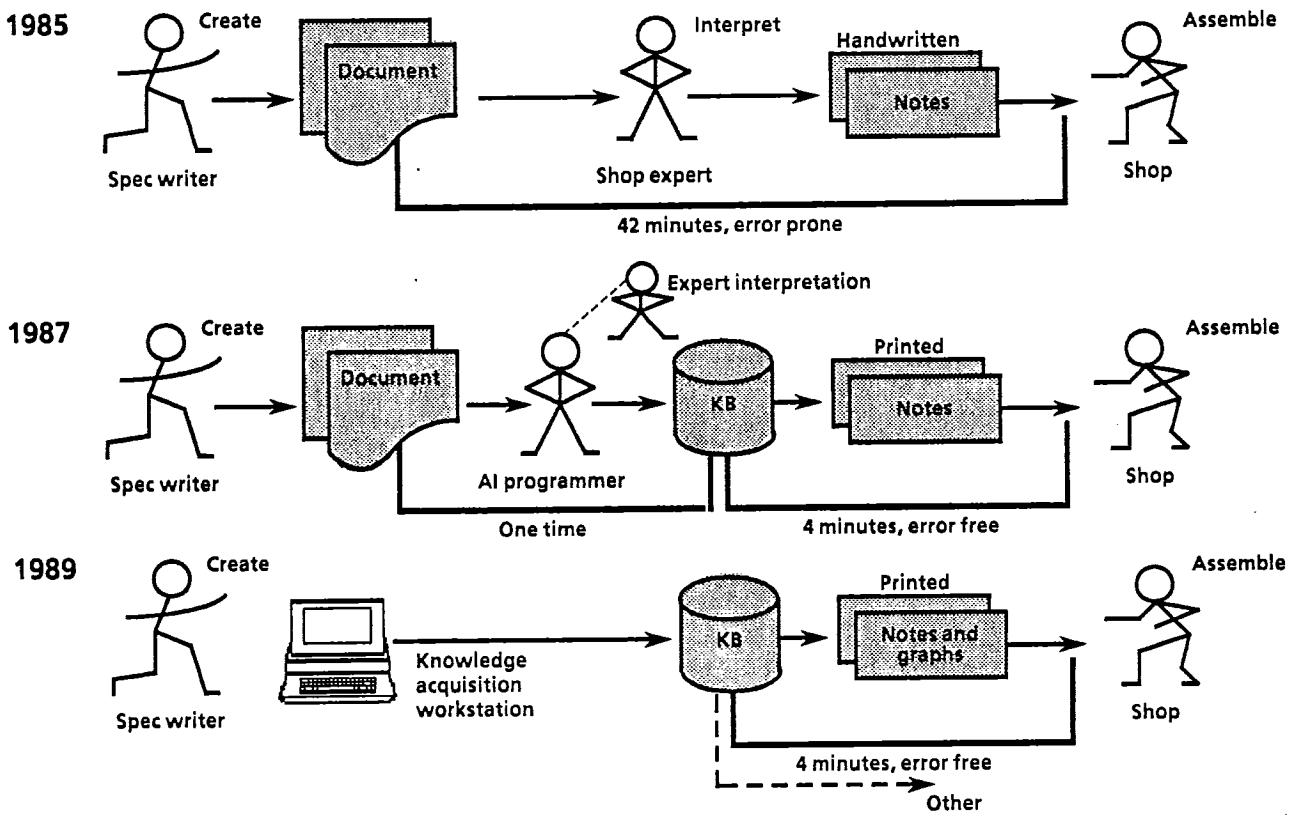

Figure 2 - Automation of Standards Usage

bases were absolutely correct. This gave rise to a knowledge acquisition project started in 1987 which would not only take care of the engineering authority issue, but would provide a significant productivity improvement for those engineers engaged in creating and maintaining standards documents.

The solution was a workstation which would include a model of a connector assembly specification and would engage the engineer in a dialog to "fill in the blanks" (see Figure 2). A Symbolics workstation running Knowledge Craft and Lisp was used for this prototype knowledge acquisition system, and after a considerable development effort during 1988, a more advanced prototype was demonstrated which solicits the necessary information from the engineer by asking for various tables to be filled in, graphics to be scanned in, and assembly instructions to be entered into various forms which capture their underlying meaning.

The information collected by the knowledge acquisition system is stored as a network of related objects. From this set of objects, which is called the neutral format, the system can produce a CASE knowledge base in Prolog or a human-readable document ready for publication in the paper system (see Figure 3). In order to achieve the document generation capability, Concordia, a Symbolics product which provides desktop publishing functionality, was integrated with the Knowledge Craft knowledge representation capabilities. Without this upcoming knowledge acquisition workstation, the CASE staff would be forever in the business of maintaining standards knowledge bases and CASE would forever be restricted to use only as a "shop aid."

Knowledge Accessibility

The most recent development project is to provide CASE knowledge directly to other software systems. Since the CASE knowledge bases can be queried much like a database, a special interface was constructed late in 1988 which provided assembly data to an IMS database application program running on an IBM mainframe. Although this is as yet only a proof of concept, work is proceeding on a "knowledge gateway" with a general purpose query language which can be used for knowledge base queries as well as database queries (see figure 4).

The primary difference in querying a knowledge base is that the system initiating the query must be able to answer questions from the knowledge base being queried. The reason for this is simply to satisfy the knowledge base's need for data as it seeks to find a solution for the stated goal. If the calling application does not happen to know the value of the attribute being asked, it can ask the knowledge base how to ask the user for the information. In this case the knowledge base returns the correct prompt string, the type of question (yes or no, single value, multiple value, etc.), a list of possible values, and a help message describing why the question is being asked and the consequences of choosing a particular answer.

It is hoped that this approach to putting knowledge "on-line" will help to solve the expert knowledge access problem and even allow applications to reason across several expert knowledge sources.

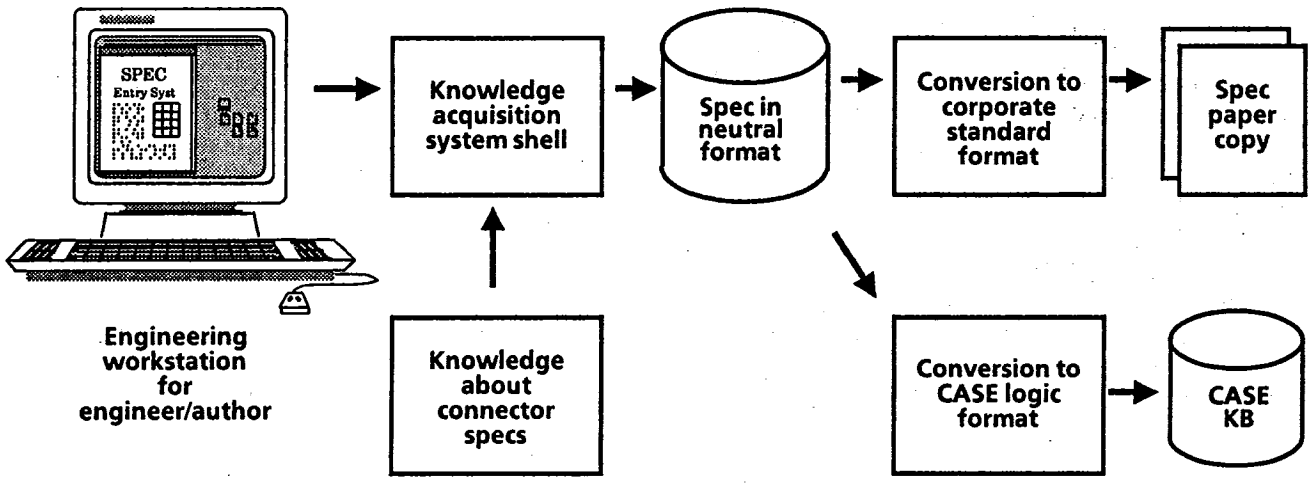

Figure 3 - Standards Knowledge Acquisition System

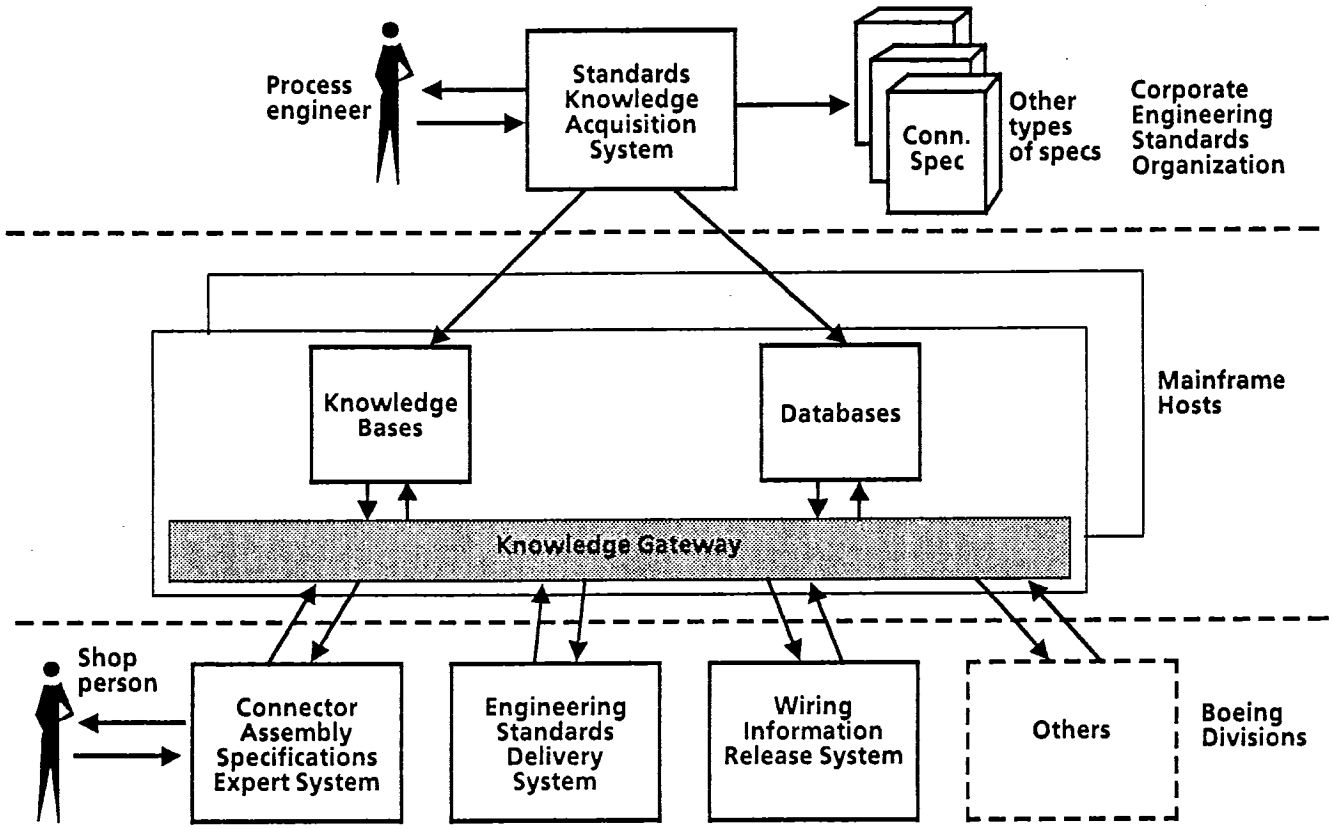

Figure 4 - Future Standards System Concept

Conclusions

CASE currently is running on a MicroVAX II; it is written entirely in Quintus Prolog and has more than 25,000 rules contained in over 300 knowledge base files. It has its own expert system shell which maintains the context of the consultation and deduces answers to questions issued by the knowledge bases from previously entered information or from the user's profile whenever possible. The shell is now being used for other expert system projects and is constantly being expanded in capability.

CASE represents more than six person years of effort over a four year period. It reduces the specification search time to as little as two or three minutes and reduces the report size to as little as one page. But most important, CASE greatly reduces the dependence on "shop experts" and provides information which is of the highest possible quality.

References

M. L. Brodie, J. Mylopoulos, *On Knowledge Base Management Systems*, Springer-Verlag, 1986.

B. Chandrasekaran, "Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design", *IEEE Expert*, Fall 1988.

L. Kerschberg (editor), *Expert Database Systems*, Proc. 1st International Conference on Expert Database Systems, 1986.

S. Marcus (editor), *Automating Knowledge Acquisition for Expert Systems*, Kluwer Academic Publishers, 1988.

W. J. McClay, P. J. MacVicar-Whelan, "AI Based Connector Assembly", Applications of AI III, SPIE, 1986.

W. J. McClay, P. J. MacVicar-Whelan, "A Knowledge Base Network for Connector Assembly Specifications", IEEE Computer Society Conference on Robotics and Automation, 1988.

J. D. Ullman, *Principles of Database and Knowledge-Base Systems*, Computer Science Press, 1988.