ENHANCING MANUFACTURING PRODUCTIVITY
THROUGH THE DESIGN AND DEVELOPMENT
OF EXPERT SYSTEMS

A Thesis Presented to
The Faculty of the College of Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
Donald Allan Roth

March, 1992
Acknowledgements

The author wishes to thank Dr. Constantinos Vassiliadis for his support, concern, understanding and advice. Thanks also go to all professors involved in my graduate studies, who taught me the true meaning of "in depth".

Appreciation is expressed to Cooper Industries for the Fellowship awarded through Grant # 2-07501-0630-171, enabling the hands-on research and experience in two of their manufacturing plants.

The author also wishes to thank Dr. Jeffery Giesey and Dr. Dennis Irwin for their support and guidance, along with Dr. Herman Hill for his administrative help.

Finally, I wish to thank my wife and son for their forgiveness and support throughout my education. Without their tolerance, none of this would have been possible.
Table of Contents

Abstract ............................................. i
Acknowledgements .............................. ii
Table of Contents ............................... iii
List of Figures ................................. vii

Chapter 1  Introduction ......................... 1
  1.1 Benefits Of Using Expert Systems ....... 1
  1.2 Organization of work .................. 2

Chapter 2  Expert Systems as Artificial Intelligence 3
  2.1 Artificial Intelligence .................. 3
    2.1.1 Expert Systems ................... 7
    2.1.2 Computer Vision ................. 10
    2.1.3 Natural Language Interface ...... 11
  2.2 Anatomy of an Expert System .......... 12
    2.2.1 Knowledge Base .................. 13
    2.2.2 Data Bases ..................... 13
    2.2.3 Inference Engines ............... 14
      a. Forward Chaining ............... 14
      b. Backward Chaining ............. 15
    2.2.4 User Interface .................. 16
    2.2.5 Other Considerations .......... 16
2.3 Knowledge Engineering .................. 18
  2.3.1 Knowledge Sources .................. 19

Chapter 3

Choosing Development Tools and Applications .... 22
  3.1 Overview of Software .................. 22
  3.2 Choosing Applications .................. 25
  3.3 Cooper Industries ...................... 26
     3.3.1 Apex Operations ................... 26
        a. Application Heat Treat .......... 26
        b. Application Socket Feasibility .. 33
  3.3.2 Gardner-Denver Mining and Construction .... 36
        a. Application Order Entry System .. 36
  3.4 Software Evaluation .................... 39

Chapter 4  Development and Testing ................ 44
  4.1 Development and Testing of
     Order Entry Program .................... 45
     4.1.1 Problem Definition ............... 45
     4.1.2 Suitability of Task ............... 46
     4.1.3 Alternatives ................. 46
     4.1.4 Return on Investment ............ 46
     4.1.5 Development Tool ............... 47
     4.1.6 Knowledge Engineering .......... 48
4.1.7 Design of System .............. 49
4.1.8 Development of System ........... 55
4.1.9 Test and Debug System ........... 57
4.1.10 Maintain the system ............ 59

Chapter 5 Summary and Recommendations ........ 60
  5.1 Results .................................. 61
  5.2 Conclusions ............................. 63
  5.5 Recommendations ........................ 66

Bibliography .................................. 69
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.1</td>
<td>Production Rule Outline</td>
<td>7</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Expert System Structure</td>
<td>11</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Heat Treat Process Card</td>
<td>25</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Heat Treat Sequence</td>
<td>32</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Input Dimension Screen</td>
<td>30</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Operation Choice Screen</td>
<td>31</td>
</tr>
<tr>
<td>3.3.5</td>
<td>ART-IM Rule Definition</td>
<td>40</td>
</tr>
<tr>
<td>3.3.6</td>
<td>ART-IM Schema Definition</td>
<td>40</td>
</tr>
<tr>
<td>4.1</td>
<td>Ten Point Plan for</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Expert System Development</td>
<td>44</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Base Model Choice Screen</td>
<td>49</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Submodel Choice Screen</td>
<td>52</td>
</tr>
<tr>
<td>4.1.3</td>
<td>User Interface Menu Screen</td>
<td>52</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Binary Representation of Choice</td>
<td>53</td>
</tr>
<tr>
<td>4.1.5</td>
<td>Constructed Codes</td>
<td>53</td>
</tr>
<tr>
<td>4.1.6</td>
<td>Order Entry Example</td>
<td>54</td>
</tr>
<tr>
<td>4.1.7</td>
<td>Order Entry Sequence</td>
<td>56</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Benefits of Using Expert Systems

With today's competitive manufacturing market, any opportunities for decreasing costs must be explored. Computer Integrated Manufacturing (CIM) combines the design and manufacturing processes into a cooperative operation. Most manufacturing processes are divided into smaller work stations, which are managed by engineering, on-site services and production control. Without the cooperative effort of all controlling interests, the station is not operating to its fullest capacity.

For such systems to operate with an assembly line, all elements of the process must be synchronized. If a person is not present, his or her absence could conceivably bring production to a halt. With the use of Expert Systems, an expert need not be present and the expertise is available on a 24-hour basis.

Expert Systems function better when the desired goal is a specific decision or recommendation and the knowledge required to arrive at the goal is limited.

Unlike traditional software programs that manipulate
large amounts of data, Expert Systems contain heuristics, or rules of thumb in their code. These rules reduce the amount of data required to reach a conclusion. Expert Systems are being used to facilitate machine maintenance, engineering decisions and optimal operation. Expert Systems are not designed to eliminate personnel, but rather to enhance the workplace and free experts to perform more challenging tasks. These systems are designed to work in conjunction with people, to help them, not to replace them.

1.2 Organization of Work

This thesis illustrates Expert Systems as an enhancement to productivity. Chapter one serves as an introduction. Chapter 2 briefly describes a few AI technologies and details the components and functions of an Expert System. Examples of some knowledge representation methods are also presented. Chapter 3 describes the process of choosing Expert System development packages and selection of realistic applications. Basic details of two applications developed for tool manufacturing are covered. Chapter 4 outlines the steps taken to design and develop an application, with emphasis on an actual application. Chapter 5 contains results and conclusions, including user acceptance and future plans.
Chapter 2

Expert Systems as an Artificial Intelligence Technique

2.1 Artificial Intelligence

Artificial Intelligence (AI) is one of today's most controversial discussion topics. Webster's defines intelligence as "the ability to acquire knowledge and apply it". Knowledge is the formulation of information in a usable fashion through experience and training. AI technology simulates some of the human's thought processes and incorporates this simulation into computer programs. Data storage is one of the computer's forte, but organizing and grouping data in a useful, retrievable form is best performed by humans. AI technologies simulate these human functions into computer programs.

Artificial Intelligence technology incorporates various methodologies. Those making considerable progress include Expert Systems, Natural Language Interfaces and Computer Vision. These programs take a process performed by humans and replicate the process.

The actual implementation of these simulations is realized using two separate methods, software and hardware. The first method is a strictly software approach which mimics classification and information storage through
programming. Large amounts of RAM and hard disk space are required to hold the information necessary to reach a valid decision. In order to achieve the desired goals, the programs must represent the knowledge in a retrievable format.

Knowledge representation methods encompass traditional procedural programming as well as Object Oriented Programming (OOP). Given the proven strengths of representing knowledge in a list or tree structure and the inheritance schemes, very precise descriptions of a particular object can be obtained with small amounts of memory. Once the initial objects are defined, variations on these objects are represented as sub-classes.

The OOP approach uses segments of code to describe not only an object (variable), but also this object's relationship(s) with other objects. Inheritance is the ability to pass object descriptions to child objects. Some programming languages use the terms objects and classes interchangeably. Polymorphism is the re-using of code on another slightly different object. Encapsulation refers to describing an object not only by what it is, but also by what it can do.

Generic code segments known as methods are specific action sequences that are sent to an object. This action is
similar to a subroutine call, but the mechanism is sent to
the object, unlike the subroutine which carries the object
to a separate manipulation segment. Portability of code is
a prime concern with all developers, and the OOP approach
employing the C++ language is as portable as possible.
Using code segments that have already been proven reliable
and robust lessens the amount of work a programmer must
complete and increases the consistency of the program. True
OOP languages adhere to the three basic premises that all
code is encapsulated, polymorphic, and is capable of
inheritance [13].

The second method involves a hardware related approach.
Neural networks simulate the human brain in both function
and physical orientation. As an example of the advances in
microelectronics, presently 256 simulations of neurons and
their programmable interconnections are packaged in a single
integrated circuit [2].

Artificial Neural Networks (ANN) are a primitive model
of the human brain’s neurons. One of the at least 10
billion [3] neurons found in the brain, has been found to
have over 100 factors influencing their behavior. The
majority of these relationships involve chemical
concentrations between adjacent receptor cells. These
relationships have been isolated in lower evolved mammals. A
popular simulation of a neuron's behavior employ a weighted summation of all inputs which is then quantized, using a simple threshold device. This simulates a neuron's ability to fire when conditions are appropriate.

Adjusting the weights of the inputs constitutes the learning process. Once properly trained, these ANN systems are used in pattern recognition systems with very good accuracy.

The interconnection of neurons present a basic understanding of how the brain distributes its knowledge associatively. If part of the brain is damaged, the knowledge contained in that particular area is not lost, it can be reconstructed from relationships stored in neurons and interconnections from another area of the brain.

Storage of knowledge is an ongoing area of research as well as how we use this information. Humans do not communicate in a strictly quantitative language. The transfer of information is more analog than digital. Incomplete or unreliable data has been dealt with using so-called "fuzzy logic", which deals with uncertainty using a combination of both statistical and probabilistic methods. This concept was conceived by mathematician Lofti Zadeh. A value between zero and one is assigned to classify
how well an object fits a category. Using this approach allows definitions such as "almost perfect" or "somewhat cold".

2.1.1 **Expert Systems**

Expert Systems are perhaps the most familiar form of A.I. Because these systems put an expert’s knowledge at anyone’s disposal, the expertise is not lost or confined. These programs are different from procedural programs most people are accustomed to. Prediction, analysis, recommendations and identification are tasks that have been successfully implemented in computer programs.

Two famous Expert Systems developed at Stanford University are MYCIN and DENDRAL. MYCIN was designed to help a physician identify infectious diseases. The input to the program includes the patient’s medical history, recent laboratory test results and a few questions presented at runtime. The system attempts to identify the disease and prescribe a treatment. DENDRAL is a program that identifies unknown molecular structures using Nuclear Magnetic Resonance and Mass Spectrometer information. Chemical and molecular structures are stored in the system’s knowledge base and the program generates all possible structures according to the constraints of the input data. These generated configurations are compared to known substances
and further sorted until the best prediction of what the substance is has been inferred.

An Expert System is programmed to reach an objective decision, rather than a strictly numerical answer or completion of an algorithm. These decision-making functions usually take the form of a production rule approach. Figure 2.1.1 illustrates the format of a production rule.

**PRODUCTION RULE FORMAT**

IF (condition 1) and/or (condition 2) and/or ... (condition n)

THEN (conclusion 1) and (conclusion 2) and .. (conclusion n)

<optional>
ELSE (conclusion 1) and ... (conclusion n)

FIG 2.1.1

Normal IF-THEN-ELSE rule syntax applies, the conditions listed in the IF part are known as a premise and the conclusions in THEN part the conclusion.

Any programming language such as BASIC or C or Pascal can be used to form the programming shell. LISP and Prolog are specialized development languages that are popular and recently some of the object oriented languages, Smalltalk,
Object Vision (BASIC) and C++ have been widely used.

The large amount of memory and processing time required have confined most stand-alone Expert Systems to a mainframe or at least a powerful workstation. Many modern PC development systems today require only 512 kilobytes of RAM and an IBM Personal Computer. Implementation on the DOS, OS/2, and MacIntosh operating systems allow the integration of popular business software, for example LOTUS and Dbase.
2.1.2 Computer Vision

Another task better performed by humans is pattern recognition and identification using incomplete data. A famous example is picking your grandmother out of a crowd. Image acquisition, image processing, image analysis and image understanding comprise a computer vision system. Image acquisition is most commonly realized using two popular methods, video cameras and charge coupled arrays \(^4\). Once the binary image is present, processing the information yields a usable data format for analysis.

Traditional video cameras require a vidicon tube to produce an analog output signal, which is then digitized. Charge coupled devices are widely used for the acquisition process. These small semiconductor devices are relatively inexpensive, afford greater sensitivity and require less power than traditional methods to obtain a silhouette of an object. Data compression techniques, similar to those used in FAX and modem transmissions, reduce the numerical calculation time and memory required for storage with little loss of resolution.

Analytical edge detection, pattern matching or symbolic shaping allow the computer to define or analyze objects or conditions. Visioning systems can operate in hostile
environments, work with incomplete or corrupted data, and do not require a vacation or sick leave, unlike a human completing the same task.

2.1.3 Natural Language Processing

Communication with a computer employing Natural Language Processing (NLP) allows the user to operate the computer without specific knowledge of particular software commands. This accomplishment requires both representation of language and use of an analysis tool. Language is the medium we use for expressing and organizing what we know [1]. A successful NLP system contains a knowledge base which includes syntax and grammar rules, and an inference engine. Several schemes are known for representing sentences, one of the more popular is the tree structure. Templates of these structures are stored in the knowledge base. The inference engine devises a sort and look-up methodologies to attempt identification. The goal of a NLP front end is to isolate the user from actual program-specific syntax.

These three technologies are all examples of how the computer can simulate processes normally performed by humans. Details on the Expert System technology are presented to illustrate how the technology facilitates everyday activities.
2.2 Anatomy of Expert Systems

Expert Systems are composed of four basic components. Figure 2.2.1 depicts the four components (user interface, inference engine, knowledge base and data base) and the communication links required. The realization of these elements is dependant upon the software being used.
2.2.1 Knowledge Base

In a rule-based program, the Knowledge Base (KB) of the system contains the production rules. It may also contain objects and their relationship to each other in an Object Oriented case. In either configuration, the actual relationships between items are defined in this section’s code. The knowledge base may be used by another Expert System using the same knowledge format. Changes to the knowledge base can be made without affecting other code segments. Updating and refining the knowledge base keep the Expert System robust.

2.2.2 Data Base

Any information required to reach a conclusion or recommendation is drawn from this area. The connections between the data base and the other Expert System components show the communication channels. For compatibility, most development packages use ASCII delineated flat files and support both types of files used in database (.dbf) and spreadsheet (.wk?) software programs. Included in this section is any data that is supplied by the user. Conclusions and inferred items from successful rule completion are added to the data base and used for further rule completion.
2.2.3 **Inference Engine**

The Inference Engine is the driving force behind any Expert System. The software that defines the Inference Engine examines the rules in the knowledge base and attempts to match the rules with the facts in the data base. Depending upon how the rule base and inference engine are configured, the matching mechanisms differ, though most employ some type of pattern-matching. Program execution methods are contained in the inference engine.

Two approaches are used to validate a rule, either forward chaining or backward chaining. Each method employs a lookup and match method. Forward chaining takes a piece of information from the data base and attempts to match it with a premise of one of the rules in the knowledge base. If a match is found, the conclusion of the matching rule is added to the data base as a new fact. Backward chaining picks an item from the knowledge base and attempts to find a match with a rule's conclusion. If a match is found, the premises may be considered valid.

a. Forward Chaining

Forward chaining looks at all current data and reaches a conclusion by application of this data to the appropriate
rules. This is often referred to as Goal-Oriented. The rules are "looked at" in a forward direction, satisfying the premise using the input and known data. Upon successful completion of the premise, the conclusion is validated. In the process of examining the rules, intermediate and final information is inferred, which is added to the data base.

b. Backward Chaining

The classical method of rule execution is known as backward chaining. This approach assumes a conclusion and examines the data base searching for a matching premise. Backward chaining is also known as "Data Driven" method. Forward chaining is simulated by including goals as a premise, and consequently looks for supporting data. This adds substantially to the complexity of the Knowledge Base. Many systems allow the mixing of both types of rules for search mechanisms.
2.2.4. **User Interface**

The interface is a means of getting data to the inference engine. The interface can be as simple as a DOS command line prompt with a series of yes/no questions. Modern development packages use a Graphical User Interface (GUI), complete with icons, mouse support and pull-down menus. The advantage of the GUI includes ease of use and consistent appearance. Clicking on a pictorial representation of an action is easier than entering the command line expression. The most important feature of the user interface is to make it as easy to use as possible. As GUIs become more common and standardized, applications presented in this format are more readily accepted than line item monochrome applications.

2.2.5 **Other Considerations**

All software is ultimately judged by how well it performs. Expert System shells should do more than represent knowledge symbolically and present menus to the user. Most development shells include some sort of explanation facility. Usually a hot key away, it produces an explanation as to why something is being asked, or why a fact has been inferred. The system also must be easily
modified allowing changes in either the knowledge base or
data base without recompiling the entire program. A runtime
version of the application is definitely desirable,
permitting the end user to access the application without
the added complexity and expense of developing the software.

The ability to transfer the program to different
operating platforms must also be considered. The
development shells available today have platform-specific
versions of the software and the ability to port an
application to another operating system. The more platforms
an application can run on, the better exposure and more use
the system will receive.

Enthusiasm for the project should be kept high. The
best approach is to involve all persons related to the
application. Not only the managers and experts but also end
users and supervisors should be aware of the program’s
progress and current state. Many programs get no further
than the prototype stage due to lack of support. Programs
that only solve a small problem in a limited area may not be
perceived as important and funding cut.
2.3 **Knowledge Engineering**

From the initial problem definition session, the developer must keep in mind what the problem is, what the particular input will be, and the ideal outcome. Collecting and organizing pertinent information into a usable fashion is the responsibility of a knowledge engineer. This individual does not need to be an expert in the problem domain, but should understand the techniques used by the expert to solve the problem. The data collected in a problem domain can come from various sources. Human experts, maintenance manuals, data books, repair flowcharts and corporate forms are all knowledge sources.

Knowledge engineering entails more than rule formation. The individual must be a problem solver and possess excellent interviewing skills. It is his/her responsibility to extract the knowledge obtained from documentation, interviews with experts and general problem definitions and code the knowledge into a usable format for the inference engine. The developer applies the needed information to the construction of the knowledge base. The most common format is a production rule system, although the OOP approach has shown considerable merit. The reusability of OOP code for similar problem solutions saves development time and human effort.
2.3.1 **Knowledge Sources**

Knowledge sources give the program intelligence. Much or all of the knowledge required to develop the Expert System can be in the form of documentation. These sources include textbooks, training manuals, repair flowcharts and service guides. The knowledge engineer collects as many of these as possible and reviews them. This accomplishes two directives, first it gives the developer some background material and second, it reveals what the final solution will be. Interoffice memos and corporate proceedings also contain valuable information. Database format and corporate standards are already built into application forms. The Expert System must be able to work within these confines.

Many times, the available knowledge is obtainable only through a local expert. This person is often the senior engineer or manager, the person contacted when an emergency arises. This individual is usually very busy and the organization can least afford his/her loss. A successful knowledge engineer will have excellent interpersonal skills and develop a clear communication channel with the expert in the exchange of information.

The knowledge engineer asks someone who has worked long and hard to obtain their current status to tell him/her what
makes them the expert. Some experts are reluctant to reveal the information that make them experts because their expertise is the result of education and experience.

The knowledge engineer's goal is to identify how the expert reaches a decision and apply that same procedure to the program. Tape recordings and written notes aide the knowledge engineer in recalling specific details of interviews. Once all documentation has been gathered and interviewing has taken place, the next step is building a knowledge representation. Traditionally, this is accomplished through the construction of production rules. If OOP definitions are required, class definitions and methods must be defined.

If the expert system developer is fortunate enough to have multiple experts, picking the most responsive and cooperative is the best approach. Multiple experts will yield multiple problem solving mechanisms, giving the developer additional insight into particular decision making processes. Any system created using an uncooperative expert as a source will not function to expectations. Enthusiasm for the entire project is also necessary. All managers of the company should be notified of the project, and asked for their comments and criticisms. Many systems get no further than the prototype stage because the development team is
confined to one problem area and the application is not perceived as important. Momentum is a prime factor in the success and acceptance of Expert Systems.
Chapter 3

Choosing Development Tools and Applications

3.1 Overview of Software

One of the research activities involved the evaluation of existing PC-based software development packages. Three criteria for evaluation were defined: the operating system requirements, ease of use, and the ability to port the code from the PC to a mainframe environment. The project team was composed of the managers involved with design and manufacturing and the research team from Ohio University. One requirement is that the development be completed on a PC. Development and maintenance will be handled by in-house programmers, therefore the package must be easily learned and modified. The completed program needs to reside in a networked environment, in this case a UNIX operating system.

Three companies arranged demonstrations and availability of evaluation copies of their software. The three companies involved were Inference (ART-IM), Intellieorp (KEE and KAPPA) and Aion Corporation (ADS), all from California. The KEE (Knowledge Engineering Environment) system required a UNIX platform to operate, and the KAPPA system relied on Windows 3.0 as an operating environment. Although the KAPPA and KEE systems were impressive in appearance and performance, the hardware
requirements eliminated these two packages from consideration.

The other two companies, Aion and Inference, provided their IBM PC versions. The time between signing an evaluation agreement and delivery of the software ranged from one week to one month.

For prototyping possible applications, software currently available at Ohio University was used, an Expert System shell known as VP-Expert. The software is limited in execution speed and in the size of data bases that it can access. The VP-Expert package is ideal for fast development of a prototype application.

The ART-IM system is a LISP-based development package, whereas the ADS package is Pascal-based. The acronym of LISP comes from LISt Processing. LISP is one of the initial Artificial Intelligence languages, quite powerful, but highly structure oriented. Because of its hierarchical parentheses format, the language is very good at processing structured lists for high speed searches. The language is powerful once the syntax of functions and subroutines is learned. The Inference package included an integrated EMACS text editor for code development.

The Pascal-based Aion package incorporates several
editors, one for each type of variable required. There are separate rule, display, parameter, function, process and report editors. The linking of these code segments is handled by the software.

The packages from Aion and Inference require at least an 80286 (AT) class of machine with a minimum of 2 megabytes of memory. Ram upgrades were required for both development machines used. Pricing and availability were two concerns since both machines required the purchase of proprietary memory expansion boards. Average cost after the expansion boards worked out to $100.00 per Megabyte. The extra RAM was required because both packages use DOS extenders to enable more than the 640 kilobyte address limit imposed by the 16 bit DOS. The DOS extenders shift the 286/386 family of microprocessors into their protected mode, enabling addressing of the available extended memory (above 1024 bytes) present in each machine \(^{[10]}\). This same trick has been successfully employed in the graphical interface, Windows 3.0 by Microsoft.

With hardware requirements satisfied, choosing applications and actual development takes place next. Care is taken to provide a fair proving ground for the two development shells.
3.2 Choosing Applications

For Expert Systems to function as a useful addition, the problem presented should have definite solutions and the knowledge needed to reach these outcomes should be limited in domain. Three applications, a Heat Treating Process System, a Feasibility Test for socket design and an Order Entry and Pricing system were developed. These applications were selected due to their problem definition and availability of the required knowledge. Apex Operations was the setting for the first two applications and Gardner-Denver Mining and Construction the third.

General descriptions are supplied for the Apex Operation applications as the information contained in the knowledge base is proprietary. Wherever possible, actual values and functions will be described. A case study approach to the Order Entry application illustrates the actual steps taken and coding performed.
3.3 Cooper Industries

Cooper Industries has allocated grant money to explore the benefits of modern technologies in the manufacturing setting. The company itself is quite diversified. Cooper's holdings include electrical products, electrical power equipment, hardware, automotive products and petroleum and industrial equipment. It is the goal of Ohio University and Cooper Industries to increase the efficiency of the companies through the use of computer-aided techniques. The grant provided funding for research at two local plants, Apex Operations, in Dayton, Ohio and Gardner-Denver Mining and Construction, in Roanoke, Virginia.

3.3.1 Apex Operations

Apex Operations is a specialty tool manufacturing company. The tools must withstand torques and operating conditions that would cause commercial tools to fail. The majority of the tools produced are used in manufacturing setting, specifically in pneumatic assembly processes. Because the tools are subjected to continual, harsh use, the normal characteristics of the steel must be altered. Heat treating the steel allows characteristics to be changed.

a. Application Heat Treat

Heat treating alters the molecular structure of the
steel, increasing or decreasing the hardness. Hardness is measured using a Rockwell Hardness scale that defines how much deformation the metal produces for a given force. The items the process card lists deal with specific type(s) of heat treating, time and temperatures for these heat treats and operating parameters. Neutral hardening affects the entire piece. Tempering and annealing operations usually deal with only part of the tool. Tempering is a typical operation for screwdriver blades, making the blade a harder steel than the rest of the tool. Annealing is a softening procedure, usually performed on threads of a tool. The operating parameters define how the item is transported (fixture) and which operation is performed. Figure 3.3.1 shows an abridged process card.

<table>
<thead>
<tr>
<th>PROCESS CARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>PART: I11</td>
</tr>
<tr>
<td>01-DESC:</td>
</tr>
<tr>
<td>2-TYPE STEEL:</td>
</tr>
<tr>
<td>03-HEAT TREAT CODE:xxxxxxx</td>
</tr>
<tr>
<td>04-REMARK:</td>
</tr>
<tr>
<td>05-Parameter:</td>
</tr>
<tr>
<td>06-Parameter:</td>
</tr>
<tr>
<td>07-USE:xxxx</td>
</tr>
<tr>
<td>08-REMARK:</td>
</tr>
<tr>
<td>09-Parameter:</td>
</tr>
<tr>
<td>10-TIME:</td>
</tr>
<tr>
<td>11-Operation:</td>
</tr>
<tr>
<td>12-TIME:</td>
</tr>
<tr>
<td>13-Operation:</td>
</tr>
<tr>
<td>14-TIME:</td>
</tr>
<tr>
<td>15-Operation:</td>
</tr>
<tr>
<td>16-TIME:x.xx</td>
</tr>
<tr>
<td>17-Operation:</td>
</tr>
<tr>
<td>18-Operation:</td>
</tr>
<tr>
<td>19-TIME:x.xx</td>
</tr>
<tr>
<td>20-Operation :</td>
</tr>
<tr>
<td>21-PRESS:</td>
</tr>
<tr>
<td>23-Parameter :</td>
</tr>
<tr>
<td>24-DRAW:</td>
</tr>
<tr>
<td>25-TIME:x.xx</td>
</tr>
<tr>
<td>26-Operation :</td>
</tr>
<tr>
<td>28-SPEED :0</td>
</tr>
<tr>
<td>30-MISC:</td>
</tr>
</tbody>
</table>

Fig. 3.3.1 Heat Treat Process Card
There are essentially only a few people that possess the adequate expertise, making this process a prime candidate for Expert System technology. The problem definition is to produce the process card. The knowledge required to complete the card has been coded into production rules and database files. Because the process is a small part of a sequence of events, the initial risk is small, but useful for demonstrating Expert System strengths.

The present method of completing this process involves a large (10 Meg) database file. Part numbers are entered and a retrieval of a matching record produces the values for the process card. When a new part needs a process card, the operation involves entering approximately 25 parameters, many available through a simple look-up procedure. References to charts for specified hardness yield time and temperatures for the hardening operation. Internal Cooper Industries guidelines define the other values listed on the card.

The goal of the Expert System is to replace the repetitious work involved with cross-referencing databooks and computing time and temperatures for the process card. The main advantage of using this approach is the consistent output and therefore a quality improvement. The execution
time is under 1 minute, compared to as much as half an hour for a human operator to complete the same task. The full time availability of this knowledge allows a non expert to complete the operation.

Knowledge sources for this application include a set of corporate guidelines for completing the process card, supplied by Cooper (ESD-31)⁶, a set of National Standards⁸ and the database file currently used. Standardized rules for time and temperature of neutral hardening have been composed by Cooper Industries (ESD-32)⁷ and are held as internal proprietary information. The expert involved is the metallurgical department manager who has compiled tables for the specialty steels used at Apex.

The guidelines were translated into production rules which present menus for input parameters. Production rules directly determine 8 of the 19 computed parameters required for a completed card. The additional parameters are determined using a combination of database searches, basic calculations and operation specific entries.
The length, diameter and weight of the tool are the first menus presented to the user. These values determine many of the parameters therefore inclusion of them in the database determines the execution path. Steel type and the heat treat code are listed in the legend of the blueprint. The heat treat code is an internal coding scheme that specifies distinct hardness, quench medium type of heat treat. The second menu asks the user to select a steel type. With this information, the program calls a database function to retrieve applicable heat treat codes. This operation is the first of several dynamic menus. The third menu asks the user to choose the type of operation. The final menu asks the user to enter a section size value. This measurement defines the maximum dimension of solid metal. Figures 3.3.3 and 3.3.4 show the dimension and operation choice user interface screens.

Please supply the following dimensions. Length and diameter are in inches, weight in pounds, pressing enter after each entry.

<table>
<thead>
<tr>
<th>LENGTH (in inches)</th>
<th>DIAMETER (in inches)</th>
<th>WEIGHT (in pounds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIG. 3.3.3 Dimension Screen
The program now has enough information to complete the process card. A final menu presents space for special remarks and the destination of the tool. An option to print the card and then a re-run option complete the sequence.

Historical data revealed physical limitations of the materials themselves. Very large or very small pieces require special consideration for transporting. These differences were discovered by analyzing the database file which contains previous entries. A systematic search using one particular fixture as a filter, i.e. "SORT ON" revealed
the exceptions to general fixture rules. By examining all pieces to use this fixture, heuristics were discovered and included in the knowledge base. Figure 3.3.2 outlines the program sequence.

FIG. 3.3.2 Heat Treat Sequence
b. Application Socket Design Feasibility

The majority of marketing activity at Apex involves telephone conversations. The second application is a program to check a particular socket configuration before processing the order. This system is designed to mimic the order form that a marketing representative fills out in the process of taking an order. Presently, an order form is completed during a telephone conversation and then passed to engineering for design. If a design is found to be improper, the design engineer contacts the marketing representative and informs the individual why. The marketing representative then calls the customer back and asks if some design specification can be changed. The process is both embarrassing and time consuming.

The desired outcome is to either verify the design as acceptable or flag the design as improper. This will not only free design engineers from dead-end routines but also enable the marketing staff to tell the customer during the initial conversation if the tool can be manufactured.
The program presents the user with a series of menus asking type of tool, length, overall outside diameter and desired drive and broach size. These values are included in the application’s database. Additional menus present specific choices based upon previous selections. In this manner, the user is lead through all entries and presented with a summary screen. The final screen either passes the design, or denies it for failure to pass any of five tests. If a test fails, suggestions for revising the design are presented.

Knowledge sources for this application are primarily tools that have already been manufactured. From existing tools listed in the product literature, values for quantities concerning the physical limitations are extracted. Departmental managers also provided most of the background calculations for determining the internal forces. In socket design, several reactions must be considered. The amount of torque the socket will withstand has a maximum limit, beyond which Apex will not warranty the tool. This value increases as the drive size increases and decreases as the broach opening decreases. Limits on the ratio between drive torque and driven torque have been established. Many of the sockets are also subjected to downward forces because of the application. Limits on the wall thickness have been set so a socket used in this manner does not collapse. All
of these values change with the size of both drive and broach end.

The product literature lists all sockets available, grouped by drive size. Charts of allowable torque versus broach opening for specific drive sizes allow this relationship to be expressed as a slope. A quick check on allowable torque limit is completed by a simple multiplication rather than a database search.

This application is implemented in an Object Oriented sense by defining the desired tool as an object with slots for dimensions and heat treat parameters. The required dimensions are supplied by the user and the rules determine the remaining values. The socket is abstractly constructed and its characteristics compared to the corporate standards.

Five tests are performed on the desired configuration. A minimum outside diameter test assures the socket can withstand a downward force. A torque test checks wall thickness for the particular broach size. An overall length test checks the sum of the drive end depth plus the broach end depth against the overall length desired. A material check makes sure the steel type required is available in the desired diameter, and the final test checks the ratio between drive end torque and broach end torque.
3.3.2 Gardner-Denver Mining and Construction

This Roanoke Virginia based company has been manufacturing mining and construction equipment for since 1859. The company's product lines range from small pneumatic drills to two-story tall mining machines. The goal of Cooper is to streamline the assembly process and reduce turn around time on orders. Gardner-Denver produces five different sizes of compressors each having up to 50 separate options and at least two different transporting mechanisms. Size consideration is based on the cubic feet per minute of air (CFM) output required. Each compressor can use different engines, type of brakes, tires, gauges and accessories.

a. Application Portable Compressor Order Entry

During a preliminary meeting with Gardner-Denver managers, the two applications developed for Apex were demonstrated. An immediate response to the socket design program sparked an interest to automating some of the work done at Gardner-Denver. One suggestion was for an order entry system. The marketing staff completes orders via telephone conversations then passes the paperwork on to one of two individuals responsible for quoting price and
availability of machinery. In particular, the portable compressor line required a large amount of paper shuffling. The idea of automating this process seemed to be an ideal Expert System application.

One of the most serious problems of the present method of manufacturing portable compressors was the lack of any standard machine serial numbering scheme. The components all have specific part numbers and costs, but the final product was numbered sequentially. This posed many problems and frustrations. To access a particular billing, the customer’s name, account number and the production manager’s recollection of how the compressor was configured are required. The present mainframe software database structure has a limitation of alphanumeric space for the serial number. There were simply too many different configurations to include a descriptive number using conventional coding schemes.

It was agreed to develop an order entry system covering only compressors. Prime consideration was to standardize the serial numbering scheme to include all pertinent data. The ability to quote a price and delivery date in the initial phone call would definitely give Gardner-Denver a competitive edge. Limiting the program to cover only compressors allows realistic goals for the three month time
frame and gives the managers a sample of the advantages of Expert Systems.

The application encompasses many desirable features of an Expert System. The domain is relatively narrow and the knowledge required was in manuals and available through the managers. The final version was completed just before the contract with Ohio University expired, and as with all software development, several revisions were incorporated. Ideas seemed to multiply as to how the program could be more "user friendly". All of this feedback is necessary to maintain a workable, useful program. Time constraints prohibited all ideas from being incorporated, but the majority were included.

The program allows the user to place an order for a particular configuration of compressor, and compute the cost within a few minutes. Extensive dynamic menus are presented, many require only a yes/no answer. Specific details of program development are covered in Chapter 4.
3.3 Software Evaluation

The development shells are tested for their ability to access database files as well as present several menus. The VP-Expert shell was used to prototype the heat treat program for demonstration purposes. This showed the departmental managers the technology did complete the process and gave the development team a better understanding of the problem. The heat treat application was used to compare the two PC based software packages because it is a typical, database intensive program that Cooper intends to computerize.

Development using the ART-IM system was completed first. Training materials for the ART-IM package were not available, therefore sample programs were analyzed to shed some insight into LISP programming techniques. A manual for operating the EMACS text editor was included. Constructing the menus required hard coding of location, size, color and contents of each menu. Program execution relied on backward chaining operation. The process card itself was treated as a goal, which had unknowns as values to be determined. Because the values requested are unknown, the program looks at the rules that have these values as a conclusion, then attempts to find supporting premises in the database. Figures 3.3.5 and 3.3.6 are examples of the LISP approach of ART-IM.
I DEFINITION FOR FIXTURE RULE ---

(defrule fixture
"Determines fixture by weight and dimensions"
(loadweight ?lw)
(schema dimensions
(pieceweight ?pw)
(length ?ln)
(maxdiameter ?md))
=>
(if(> ?ln 6)then(assert(schema fixture
(ppw 0)
(use rack)))
(assert(fact4 true))else
(if(and(< ?ln 4)(< ?pw 1))then(assert(schema fixture
(ppw 0)
(use basketcase)))
(assert (fact4 false)))
else(assert(schema fixture
(ppw find)
(use spider)))))

Fig 3.3.5 ART-IM Rule Definition Example

---- DEFINITION FOR SCHEMA VALUES ------

(defschema steel
(steel-type)
(heat-code)
(tempering)
(rock)
(ddraw)
(temp-range))

Fig. 3.3.6 ART-IM Schema Definition Example
The schema definition sets the class name, steel, and its slots; steel-type, heat-code, tempering, draw (time) and temp-range. These slots all require values for the final process card. The rule definition is in Reverse Polish Notation, the operation is listed, then the two operands. Parentheses add one more level of hierarchy. The => symbol translates to the THEN in normal rules. The variables preceded with a question mark represent inputted or asserted values. The first set of conditions, those within the first set of parentheses must be satisfied before anything following the => is examined.

Although the ART-IM shell required learning a new language, the mouse support and use of colors eased the debugging process. If a schema is viewed, the slots associated with the schema are one of three colors. A green appearance means the slot value is valid and asserts another value. A yellow color designates that the slot value is presently unknown and red signifies the slot has an untrue value, it is working against asserting the schema as valid.

Debugging in the ADS shell requires running the program until a breakpoint is reached, then examining a log of internal activities. All logical connections between objects are retained in the editor for the specific item.
In this manner, the developer can trace connections while editing the program.

Execution time for the heat treat application was much longer in the ART-IM shell than the ADS shell. Both shells performed database calls with some revision to the data files. The ART-IM shell compiles the entire code segment in the process of making a runtime version. The ADS shell relies on an execution program to read the development code for a runtime version.

After constructing the same program using both packages, an evaluation was performed. Primary concerns were the ease of use, database communication facilities and upgradability. The ADS package was much easier due to the separate object editors and the familiarity of the Pascal language. Both shells claimed to be portable to other platforms.

Customer support was slightly better with the Aion (ADS) product. Three telephone calls were placed to each vendor for comparison purposes. The ADS support personnel were very helpful and friendly. The software engineer and sales representative visited Apex on two occasions and always stopped to check on development. Aion also supplied reference manuals and training material. The ART-IM shell
had minimal documentation and required a vendor-supplied function to communicate with data files. This was supplied by the vendor after two telephone conversations.

In conclusion, both packages performed satisfactorily. ART-IM has the mouse support and menus, superior for debugging purposes, but the limited output facilities and lack of pre-defined functions for data base manipulation detracted from overall ease of use. Because of the dependance on LISP, knowledge representation is confined to one format.

The ADS shell is not without weaknesses. To find an object the developer often needs to look four to eight layers deep in code. The program cannot be printed in a start-to-finish fashion, due to the multiple editors and object inter-dependance. The familiarity with Pascal makes developing an application less tedious, and the predefined display functions make creating menus much easier. Applications can be developed in either an Object Oriented manner or in the traditional production rule format. These factors and the ability to describe knowledge in many formats makes this the package of choice.
For an Expert System to be successful, the program must meet the expectations of the end users. Demonstrating the program in stages of development insures that the system is functioning as envisioned. The order entry program developed for Gardner-Denver Mining and Construction will serve as a case study. The two applications developed for Apex will not be included in this section.

Guidelines for successful development include the ten points listed below in Fig. 4.1 [5].

---

**Ten Point Plan**

1. Identify problem and need.
2. Determine the suitability of task.
3. Consider alternatives.
5. Select development tool.
6. Perform the Knowledge Engineering.
7. Design the system.
8. Complete the development.
9. Test and debug the system.
10. Maintain the system.

---

Fig. 4.1 Ten Point Plan for a Successful Development of Expert Systems
4.1 Application Order Entry System

Portable Compressors constitute a substantial part of Gardner-Denver's product line. Orders are often placed over the telephone. Most customers decide which product to purchase based on availability and price. Only a few people have the knowledge to quote delivery prices and delivery date. This application will replicate their expertise and give the marketing staff an added advantage.

4.1.1 Define the Problem

Presently only a few persons are authorized to quote a price and delivery date for a compressor. The price is a function of configuration, and the delivery date is a function of materials in inventory and the manufacturing schedule. The present serial numbering system is not acceptable. The numbers have been assigned sequentially with no regard as to the particular configuration. The paperwork required and limited personnel capable of performing the make this a feasible Expert System application.
4.1.2 Determine Suitability of Task

The knowledge required to complete this task is readily available. Documentation provides most of the information required and cooperative experts are involved. The process of filling an order has been standardized in format and contents, but requires a lengthy path of paperwork processing. If the people responsible for quoting delivery date and final cost are not present, the process takes even longer. The risk involved is small because this is not a critical operation, but the returns are numerous.

4.1.3 Consider Alternatives

The serial number problem has existed from the onset. The alternative is to continue to operate under present system and/or construct a DBMS front-end to automate the process. The end user is then tied to the mainframe. The constraints of the DBMS software prohibit incorporating the serial numbering idea.

4.1.4 Compute Return on Investment

Consistency and the assignment of a meaningful serial number are the most tangible benefits. Quicker quotes increase competitiveness and consistency improves overall
quality. When the serial numbering scheme is implemented, configuration information can be retrieved using only the serial number, improving customer support. This application will be installed on notebook computers for field sales and service. The cost of developing and maintaining the application is not negligible and must be considered.

4.1.5 Select Development Tool

The ADS shell was used for development of the order entry program. The user interface is composed of several menus and the ADS shell has provisions for quickly creating and modifying these menus. A modular coding scheme was used for development. The development team divided larger tasks into smaller pieces. Because the ADS system uses separate editors for each object, this presented no problems. Each member was responsible for specific code revisions which were included in the application every evening. Small database files are accessed and the developers were familiar with the process from earlier applications.
Sources of knowledge consisted of documentation and local experts. The program entailed automating order form completion and the serial number coding. Brochures provided the options available for each of 5 sizes of portable compressors. The specification sheets contained the actual part numbers and price for each option. The standardized order form served as a basis for the user interface. The output was similar to the present final work order.

Some of the information is already in the mainframe database, but its arrangement is not optimal. Mainframe data is retrievable using an ASCII file format. The primary data items required are the customer name and account number. The account number allows access to the customer's financial record which includes delivered machines. The present system has no direct link between a customer and the Gardner-Denver machines they own.

The available option list changes with each different model. These lists are converted into DBMS type files. Each record of a file contains the script as the option appears in the menu, its part number and a brief description of the option. All pricing is performed by the program.
Database files referenced by part number were constructed for base model prices and option prices.

There are currently 45 portable compressor options. The development team decided to include 5 future options in the coding scheme for descriptive serial numbers. Standard codes for compressor base model and submodels have been established. Both of these items and a description of options on the compressor must reside in a fifteen space block in the mainframe database.

4.3.7 Design the system

The initial task was to design an interface that allowed the user to choose the desired options. Following the order forms approach, preliminary questions select base model and submodel. After a base model is selected, a production rule imports the correct list of submodels and inserts them in a choice menu. The user interface was rewritten numerous times, mainly to simplify the operation. The options were grouped into 10 groups of 5 options each. Care was taken to group related items together for a logical flow. Because the option list changes for each different base model, the contents of the menu was assigned a variable. Another production rule assigns the menu contents according to base model selection. Figure 4.1.1 shows the base model choice screen and Figure 4.1.2 shows the submodel.
Fig. 4.1.1 Base Model Screen

Fig. 4.1.2 Submodel Choice Screen
Each group of five options is assigned one place in a ten symbol string. By tracking the yes or no choices on each option, a five-bit binary representation of the choices is constructed, 0 representing no and 1 representing yes. The string is converted to an alphanumeric by the function CONVERT. The function imports the string of symbols and using a matching technique, the alphanumeric representation is assigned. The function uses two string-lists, one containing the binary values in ascending order, the other contains the numerics 0-9 and alphabetic A-X (excluding I and O). The imported string is examined for a match and when it is found a pointer value determines which "super-hex" symbol is returned. The other 9 groups of options undergo the same treatment. The 10 "super-hex" values are concatenated to form the bill of materials (BOM) number. The scheme is similar to a Vehicle Identification Number (VIN). Using production rules, options that are either standard equipment ("STD") or not available ("N/A") are taken into account. The assignment of these items relies on base model and submodel choices. Figure 4.1.3 illustrates a typical menu screen and Figure 4.3.4 illustrates the option summary for a completed run.
Fig. 4.1.3  Typical Option Choice Screen

Fig. 4.1.4  Binary Representation of Choices
Once the choices had been coded, the completed BOM and model/submodel codes are displayed. Figure 4.1.5 depicts these codes.

Output of the program was defined early in the development stage. The configuration listing is very similar to the form currently used. The ADS shell has several methods of printing data, but since the length of the form will vary from order to order an adaptive approach was taken. Three variable length string-list parameters were created to complete the output file, price_list, part_num_list and quantity_list. As options are selected, the corresponding quantities, part numbers and prices are substituted into the respective lists.
These lists are passed to an output function. The actual printing is looped on one of the synchronized lists. The pricing of an order requires the operator to enter discount amounts for both base model and options. Figure 4.1.6 shows a typical order entry form.

| GARDNER-DENVER COMPLETE MACHINE Page ___ |
|-----|---|-----|---|---|
| of____ | of____ | of____ | of____ | of____ |
| MINING and CONSTRUCTION SALES ORDER DATE |
| 12-Dec-91 |

----SOLD TO-----
ANY COMPANY
217 ROUTE 00
SADDLE BROOK NJ USA

----SHIP TO-----
ANY COMPANY
217 ROUTE 00
SADDLE BROOK NJ USA

10. QUANTITY OF COMPRESSORS :  1
11. COMRESSOR MODEL : D0190J
12. BILL OF MATERIALS #: P19J-0-X14G180000
13. COMPRESSOR S/N : 0190.....

-----QTY---PART#---OPTION DESCRIPTION-----LIST
PRICE---NET PRICE
1 xxxxxxx AIR FILTER DP xx.00
xx.00
1 xxxxxxx SEPARATOR DP xx.00
xx.00
1 xxxxxxx AMMETER xx.00
xx.00
1 xxxxxxx xx TACHOMETER xxx.00
xxx.00
1 xxxxxxx* KEYED IGNITION xx.00
xx.00
1 xxxxxxx* GEARED J-STAND xxx.00
xxx.00
1 C\E STD LIGHTS 7-POLE x00.00

Fig. 4.1.6 Example of Order Entry Output
At the third or fourth demonstration of the system, the managers requested customer information to be included in the program. The mainframe database file was trimmed to include only the name, address and customer number. Another desired addition was for serial number manipulation to be included in the program. An option to look up configuration information was also added. In total, 5 different action sequences are covered. These actions deal with ordering a compressor with or without options, ordering options only, bill of materials number decoding for configuration information and serial number manipulation.

The additional items required some new coding, but the groundwork for the decoding sequence was in place. The approach to decoding was almost the exact opposite of the coding scheme. The ten-bit code is examined one symbol at a time and decoded by the function RECONVERT. The function exports a five-bit binary code which is ANDed with the appropriate list of 5 options. The options corresponding to a 1 are added to a string-list for output. Including the customer's information meant creating a few new parameters and menus. The serial number manipulation was developed separately. Figure 4.1.7 shows the final program flow.
ORDER ENTRY SEQUENCE

Fig. 4.1.7
4.1.9 Testing and Debugging

Testing the application involved comparing previously filled orders to the output of the program. The manager supplied the research team with approximately 20 orders. The testing revealed only two differences. Total price differed on several of the orders. This was not a programming error, the price of certain items had increased, so the total cost from the program was higher than the historical run. Another case revealed that the person who completed the order had miscalculated one price.

The second inconsistency surfaced due to a transporting mechanism that the knowledge engineer did not consider. The larger compressors are occasionally mounted on a wooden skid instead of a trailer. When this occurs, the options pertaining to wheels, lights, brakes and towing accessories are not applicable. A rule to cover this situation was inserted along with a simple menu asking the user if the compressor requires a wood skid.

With all of the desired features incorporated in the program, an average iteration is completed in under 5 minutes, including output.

Execution bottlenecks occurred at two places. When the
Expert System imports the database file containing the customer information, a 10-20 second delay takes place. The delay is dependent on the access time and operating speed of the PC. A "quick-fix" for this action called for dividing the customer database file into domestic and foreign customers. All that was required was the addition of another menu and a production rule for import file designation. The second delay takes place when pricing takes place. The format of the string-lists confines the program to another search and match technique for extracting current prices.

A final demonstration to the marketing staff discovered the need for additional lines for special shipping and billing instructions. Because the program executes procedurally, the addition of another menu was not a problem.
4.1.10 Maintain the System

Revisions to the application seemed to occur every day. Ideas and suggestions from the managers and experts came one after another. Due to time constraints, the research team had to set a date after which no more revisions would be considered. The main reason for this action was that the team needed to write some documentation for the program. General instructions and a section on revising the database files were included.

Initially the maintenance manuals were included in the order, those have since been eliminated. Some fine tuning of the execution path has also taken place. For a system to be fully functional, all improvements must be considered. This feedback helps the end user get the program they want and helps the developer recognize future applications. By incorporating the improvements in the system, it becomes more useful and more reliable.
Chapter 5

Results, Conclusions and Recommendations

For any software to be considered a viable product, it must fulfill a need, perform some type of service. In Expert System technology, the desired output is a rational decision or course of action, one based on the knowledge and insight of some defined area of expertise. This decision-making process is the exact behavior that is encapsulated in a program that a non-expert can use. The desired outcome in each program developed is a specific item or items. In the Heat Treat Program, the end goal is the process card, an orderly list of operations and the specific parameters required for these operations.

A simple go-no go answer and applicable list of dimension values is the product of the Socket Design Feasibility Tester program. For the Order Entry Program, the output can take on several forms, but basically consists of the unique Bill of Materials number and completed order form.

Since all of these procedures were previously completed manually by specific persons, the risk involved by using these programs presented no loss of productivity or danger. The knowledge required to complete these tasks is from a
limited domain. For these reasons, all three applications fit the "classic" Expert System problem definition.

5.1 Results

One definition of a successful program submits that a successful program performs as designed. The applications developed for Apex Operations performed admirably. The heat treat program reproduces the same results as the DBMS program without relying on the large database file. Completing a process card no longer involves many reference manuals and calculations. The output is consistent and quick. This is as expected, for the knowledge used in the knowledge base came from this file. The advantages of the Expert System are that anyone can produce the process card without any real expertise in the metallurgical area and the expertise is available 24 hours a day. One test of the system was inviting a machine operator to run the program, who did so with very little prompting. This action is an indirect measure of "user friendliness".

The socket design checker also performed as expected. The system either accepts or rejects a configuration almost immediately without a design engineer's input. This particular program has future plans to incorporate more than
the socket line of products.

The order entry program was the most significant program developed. Five different action sequences are possible, and the program performs each as expected. There is a concern with the execution bottlenecks when accessing the large customer information file. This is an annoyance, but unavoidable if the user demands this information be present. The construction and decoding of the Bill of Material number operates quickly even though there is considerable string-list manipulation.

All three applications demonstrated the advantages of using Expert Systems for some procedures. The use allows a non-expert access to the decision-making process the expert uses. The full time availability of this expertise eliminates downtime when the expert is unavailable. The systems also allow the expert to perform additional tasks rather than repetitious, routine work. Customer support benefits directly from two of the applications and the consistency of output increases overall quality.

The experts involved tested the applications for validity. In the case of the heat treat program, the metallurgical manager was asked to try to fool the program.
by entering an obscure piece of metal. The results impressed the manager because it handled this task quite well. The socket design program was validated by a design engineer who asked the program to check a design known to be improper. The system not only failed the design but also suggested revising the proper parameters. The order entry program completes its task as expected with the added benefit of speed and consistent results.

5.2 Conclusions

Increasing efficiency was the main goal of the Expert System research. Efficiency is measured as a ratio of work required for a given output. In this regard, all of the applications increased efficiency. The quick solution to a given task definitely improves customer support. The effort required to complete a heat treat process card was not unrealistic, but time consuming and routine. Automating this process frees the expert to perform other more meaningful tasks.

The user interfaces were constructed to make operating the system as simple as possible. Revisions on all three applications improved the ease-of-use factor. The users were more comfortable using the simple menus.
The order entry program in particular required rewriting the interface many times. Initially, each group of five options were presented in a single menu and the user would choose the desired items with a combination of cursor and function keys. This proved confusing for those with little computer experience. The final version of the system has simple yes/no choices. Items not applicable are marked as "N/A" and choice of these is automatically bypassed. Although this required creating a large number of new variables and menus, the end result is an interface that is almost fool-proof.

Cooperative experts and available knowledge sources are a must for proper development. The people involved at both manufacturing sites were quite accommodating in this regard. The actual data used on the Apex programs is considered proprietary and apologizes are made for the lack of detail. As pilot programs the applications demonstrated the advantages of using Expert Systems and the procedures required in development. Interest and enthusiasm was evident from the positive feedback received at demonstration time.

Manufacturing operations rely on sequential completion of tasks. Expert Systems fill the gap created in the process if the individual responsible for a task is absent.
Integrating the program into existing software and data structures is necessary for the system to be useful. Expert Systems usually work in conjunction with existing methods rather than perform the entire process.
5.3 **Recommendations**

The most important result of the research is the demonstration that a specific task can be completed by a computer program. Capturing an expert's knowledge and coding it into a knowledge representation was probably the biggest undertaking for all applications. Many end users are computer novices therefore the user interface appearance may make or break an application. Working through the three programs, it was discovered that if a menu is presented with only distinct choices the user is more comfortable than if a line is presented asking for an input. By limiting the choices, program flow can be controlled by the programmer.

Demonstrations at different stages of development are mandatory for a workable system. The feedback from the demonstrations is valuable information. The developer finds which areas are well received and understood as well as where in the program the user is uncomfortable.

Adapting to existing data structures can be a major area of work. Mainframe computers are almost exclusively used for recording and manipulating inventory. The Expert system must be able to access this information. Often applications are integrated with other operating protocols so the software must communicate with other programs.
The goals of the system should be discussed and finalized early in the development cycle. Think big and start small is a popular phrase used to describe Expert System goals. By illustrating the technology using a smaller task, end users and management can appreciate what the technology can accomplish without a lot of risk. The system must be perceived as useful and trusted, not frightening. The order entry program is a good example of trying to do too much in a limited time frame. Initially the program was to handle configuring a portable compressor. The program grew in complexity with every demonstration. Including the customer information (name, address, account number) was a nice enhancement but definitely slows execution speed.

Future refinements are planned for all applications. Apex is planning to integrate the design checker into their CAD programs. Gardner-Denver plans to use a similar approach of the order entry program for their entire product line. Eventual links with the mainframes will allow not only configuration and pricing of a machine, but also inventory adjustment and manufacturing scheduling.

The work involved in maintaining Expert Systems is continual. The problem definition may change and alternate
solutions may also be incorporated in the program. With the support of all involved in developing applications, the systems have proven effective and accurate.

Today's development shells provide methods for quickly forming a workable prototype. This prototype serves as a demonstration tool and a basis for the complete application. Expert System technology is not black magic or a human replacement, rather a productive tool to aid people in decision making and complex tasks.

Artificial Intelligence technologies are used in ways most people do not realize. Expert Systems are a proven technique that should be treated like any other programming tool. As the technology matures and acceptance and use increase, these systems will be performing tasks that previously required specialists.

Computers are still dumb machines. The machine will not perform anything it is not told to do. Expert System technology gives the computer program the instructions necessary to arrive at decisions based on an expert's knowledge in the subject area. AI technology simulates a human brain in operation for completing tasks. As the mechanisms of human thought processes are better understood, the simulations of these processes will also improve.
Bibliography


[7] ESD-32 "General Procedure, Apex Heat Treat Department" Cooper industries, Houston, TX.


[14] Apex Fastener Tools Catalog C-100, Cooper Industries, Houston, TX.
