LESSONS LEARNED WHILE IMPLEMENTING EXPERT SYSTEMS IN THE REAL WORLD OF CLINICAL TRIALS DATA ANALYSES:
THE POSCH AI PROJECT*


University of Minnesota
2829 University Avenue S.E., Suite 408
Minneapolis, Minnesota 55414

ABSTRACT

The POSCH AI Project has developed and previously reported two expert systems that allow us to automate certain elements of clinical judgement used in clinical trial data analyses. It is part of an ongoing effort by POSCH to use innovation in data processing and analyses of clinical trial data.

Once the systems have been developed and validated, many day-to-day problems must be resolved before procedures and programs are in place that allow us to process the thousands of individual cases found in a large database. In this paper we will document our experiences implementing expert systems in such an operational environment. This includes the problems encountered, procedures used and solutions found.

INTRODUCTION

Previous reports of the POSCH AI Project have described the successful development of expert systems that have allowed us to automate elements of clinical judgement used in clinical data analyses that were not previously possible[1,2]. It is part of an ongoing effort by POSCH to use innovation in data processing and analyses [5,6]. Simply building expert systems that can receive and analyze clinical data on individual POSCH patients does not fully accomplish the real world task. Once the systems have been developed and validated, many day-to-day problems must be resolved before procedures and programs are in place that allow the expert systems to be used efficiently to process thousands of individual POSCH cases. In this paper we will document our experiences implementing expert systems in an operational environment. This includes the problems encountered, procedures used and solutions found.

BACKGROUND

POSCH has successfully developed and validated two expert systems[1,2]. Each system compares a set of similar clinical data obtained on an individual patient at two different times, usually several years apart, and determines the progression or regression in a patient's performance or condition over time. The conventional evaluation process requires clinical judgement in order to consider the subtle nuances in the data and to adjust for missing or aberrant values. Algorithms are generally considered to be inadequate and statistical techniques used alone encounter problems as well[3].

The first of the two systems that we are implementing for production work compares a patient's change in performance on a pair of graded exercise ECG (electrocardiographic) tests taken one or more years apart[1,3]. This system is called ETA (Exercise Test Analyzer). The second system analyzes changes in the condition of the patient's coronary arteries as indicated by a pair of coronary arteriograms taken two or more years apart. This system is called ESCA (Evaluator of Serial Coronary Arteriograms)[2,4]. The problems and solutions discussed in this paper are based on our combined experiences implementing these systems in an operational environment; performing hundreds of comparisons for ESCA, and thousands of them in the case of ETA. To be more precise, we are reporting the lessons we have learned so far, since the work is continuing.

HARDWARE SELECTION

Our hardware related problems and experiences simply reconfirm the need to

*POSCH stands for Program on the Surgical Control of the Hyperlipidemias. It is funded by NHLBI under Grant HL15265. The work of Slagle, Wick and Irani is also partially supported by the Microelectronic and Information Science Center, University of Minnesota, and NSF. The POSCH AI Project is an effort to find more efficient and cost effective ways to process the POSCH Study data.

**See APPENDIX
stick to a pair of commonly known principles: use reasonably up-to-date and standard hardware and plan ahead.

The ETA system was developed on a PERQ Workstation and the ESCA system on a SUN Workstation. We believed neither of these systems were suitable for volume production work nor would they integrate well with our databases. For very practical reasons, namely, availability and very low marginal costs, we chose to implement both of the production systems on our in-house Data General (DG) MV/8000 minicomputer.

Conversion of the system developed on the PERQ Workstation (ETA) proved to be complex, even though we had a working Common Lisp interpreter/compiler on the DG. The lesson we relearned from this experience is that it pays to use workstations for development work that are standard enough that systems developed on them can be relatively painlessly transferred to the computers used for the production work. One main problem with the PERQ Workstation was its lack of communications utilities. This made the electronic transfer of files to the DG practically impossible.

ESCA transferred quite easily to the DG system because it was developed on the more advanced SUN Workstation. Another reason ESCA transferred more smoothly, besides being developed on a more advanced workstation, was that the knowledge engineer knew in advance that we planned to run the production system on the DG. Before he started his development work, he examined the DG and knew the conditions under which the production system would have to perform. This simple advanced planning step proved exceptionally valuable.

**LISP VERSUS FORTRAN**

Both of our systems were developed using Lisp. AGNESS, an evolving expert system shell developed and used by James Slagle and his graduate students, to support the knowledge engineering and development work [7]. We initially believed that Lisp would be too inefficient to be used in the production environment. However, DG Common Lisp, especially the compiled code version, proved efficient enough that it wasn't necessarily the case. Our experience with the issue of Lisp versus Fortran produced a split decision. One system was converted to Fortran and the other one is being operated in production work using Lisp code.

An important consideration in the Lisp versus Fortran issue is programmer availability. Lisp programmers are still rare and usually quite expensive. The circumstances in our particular case are somewhat different from that of a commercial enterprise. The developers of our expert systems, graduate students (MRW, ERI), were excellent Lisp programmers and codevelopers of AGNESS, the expert system shell. However, the graduate students' priorities are not in production work. We had to rely on a more conventional programming staff to build the production systems. We were fortunate to have excellent conventional programmers (FRW, PFC) who also had some interest in and knowledge of AI.

Be that as it may, the essential problems that we faced in deciding whether to use Lisp or Fortran in the production system will also be faced by almost anyone attempting to use an expert system in connection with a large database. The advantages of Lisp in the development of expert systems are generally known and accepted. Once developed in Lisp, however, the system may be too slow and inefficient and used to be rewritten for use in production work. Trade-offs must be evaluated in terms of Lisp inefficiency versus the cost of conversion to a more efficient programming language for the production system. In our case, the alternative was Fortran.

A related problem has to do with the fact that Lisp code sometimes is difficult, though not impossible, to recode in more conventional languages. Some languages, most versions of Fortran, for example, do not support techniques such as recursion. In many instances, the transformation from a recursive to an iterative procedure can be far from obvious and quite difficult. The strategy we used was to tackle the problem in logical steps. Operations such as data retrieval were done using Fortran code. The required information was then saved, in the proper format, to an intermediate file so that it could be used as the input to the Lisp coded expert system module. Similarly, the results of the Lisp program was stored in the proper format to be processed later by a program segment coded in Fortran.

The graded exercise ECG system, ETA, was converted to Fortran. There are several reasons why. As previously mentioned, the PERQ system is obsolete and difficult to use. Our programmer even had difficulty getting the well developed and documented system to run on the PERQ after a lapse of time. Conversion of the Lisp code to run on the DG would have required a major reworking of the system. The production system environment was so different that a total conversion of the system was required. Since our programming staff was more experienced in Fortran, and since a number of front end programs remain in Fortran, we decided to build the entire production system in Fortran.

Lisp was very useful in the development phase. It facilitated program modifications including debugging. This we would not want to change. However, once the expert system was developed in Lisp with
the rules and network structures clearly defined, recoding in Fortran can be quite straight forward. ETA was built using an hierarchical network structure that was easily recoded in Fortran. Had the expert system been more complex with loops folding back into the network, recoding in Fortran may have been more difficult.

Finally, since we will use ETA to run several thousand comparisons, program efficiency is a very important factor. Even relatively small improvements in run time are worthwhile in this case.

The arteriography system, ESCA, was not converted to Fortran. The fact that both the SUN and DG use Common Lisp proved very significant. ESCA runs quite efficiently on the DG using the compiled version of Common Lisp code. The compiled Lisp version of ESCA runs on the DG ten times faster than did the interpreted version. As previously mentioned, another significant factor allowing us to use Lisp in the ESCA production system was due to a simple action of the knowledge engineer who purposely built the system so that it would run efficiently on the DG. This consisted of examining the DG working environment and learning how the DG version of Common Lisp operates. In fact, the results proved so efficient that we believed the additional gain we might achieve by converting to Fortran was not great enough to offset the conversion cost.

Another consideration for the ESCA system is the mode of operation of the production system. Whereas the ETA production system processes thousands of cases drawn from the main medical records database to be analyzed serially in one operation, ESCA is used interactively by a technician who reads x-ray films and enters the case data interactively at a terminal one at a time. Once the measurements are complete and entered, ESCA also processes the case comparisons in a batch mode, but the number processed at one time are far less than for ETA. Speed is not as important a factor especially since ESCA runs very efficiently on the DG using the compiled Lisp code. In other words, run time is not a critical issue.

In summary, based on our limited experience, whether or not to convert expert systems developed in Lisp to a more efficient programming language for production systems depends on the individual circumstances. Elements that influence the decision must evaluate trade-offs involving:

- size and complexity of the expert system
- number of times the system is used
- ability of the responsible programmer to build and operate the production system
- efficiency of the Lisp code in the production system

- ability of system to shift between Fortran coded modules to Lisp coded modules.

OTHER SOFTWARE CONSIDERATIONS

Normally, the expert system is only one, although key, module in the production system. Other modules that are needed include such things as a program to find and retrieve relevant data from the database (data elements required as input for the expert system), a program to build an input file in such a way as to keep track of what, where and how cases are analyzed, a program to build a file to receive, identify and format cases as they are processed, etc. We built all of these using Fortran and existing features of the DG operating system. For these reasons, it seems relevant to point out that the developed and validated expert system is only one component of a production system. You then have to provide all of the other "housekeeping" modules to do file management, input and output, etc., that are common to such systems. In our experience, this increased development time by at least 50%.

STATISTICAL CONCERNS

Statistical procedures have been involved in two important but different ways in the development, implementation and operation of both ETA and ESCA. Statistics are first involved in the validation process, when the systems are developed. Once the production system is operational, statistical methods must be used to analyze the output of the expert system. That is, the evaluations performed by ETA and ESCA must still be analyzed using acceptable statistical methods.

Normally, one would expect that the statistical methods used to analyze the output of an expert system to be essentially the same as those used to analyze the results produced by human experts or whatever else the expert system imitates and replaces. This is the case with both the ETA and ESCA.

In both ESCA and ETA, the final conclusion of the comparison of the test pairs over time is recorded on a seven point symbolic scale as follows: much better, better, slightly better, no change, slightly worse, worse, much worse. In addition, ESCA but not ETA divided the "no change" category as either "leaning toward better" (coded 0+) or "leaning toward worse" (coded 0-). (For more details on these systems see references 1, 3, 5.)

The process involves tabulating the frequency of use of each of the seven (eight where "no change" is divided into two parts) symbolic categories ranging from "much better" to "much worse" as previously described. We then determine
the proportion of cases in each category, expressed as a percentage. The pattern of proportions falling in the various categories (sometimes truncated) for two groups being compared are then evaluated using the Kappa statistic.[8]

**SPECIAL PROBLEMS**

In both the ETA and ESCA systems, we encountered special problems unique to each application. These problems would not occur in just the same way in any other application, but we believe they are typical of the kinds of things one will encounter. For this reason, we believe it will be helpful to list some of them here.

**Validation and the Production System**

We consider the validation of an expert system to be an integral part of the development phase of the expert system. The process we used to validate ETA and ESCA is documented in earlier publications describing their development[2,3]. Essentially, it involves comparing the output of the expert system to that of human experts on the same set of cases. We used the same statistical procedure for validation that we use to compare, for example, the POSCH control group to the POSCH intervention group. In spite of all of this, our experience indicates that the validation issue may come up again in the production phase. We will describe how this happened with ETA.

One of the useful features of expert systems is their ability to handle situations with missing or incomplete data. This can also be a handicap. A bit of background is needed to explain how this proved to be the case. A physician administers each individual graded exercise ECG test and interprets it. ETA then uses as input data generated by each test along with the physician's interpretation. The interpretation is a key element. The attending physician can conclude that the test is negative (1 category) or positive (3 categories), all of which were easily handled by ETA even when one or both of the tests being compared have missing or aberrant data.

The physician can also classify a test as inconclusive, unsatisfactory or unknown. When we began to run the ETA system, we immediately encountered problems with these latter three categories of interpretation of the tests because, even though ETA will process and interpret pairs of tests where one or both of the tests are assigned one of these three categories, the results are suspect because ETA was not validated for tests with these interpretations.

To resolve this issue, we would have had to initiate a new expert system develop-ment phase of greater magnitude than the one used to built ETA in the first place. We had it decided if it was worth the effort. Could we go with what we had? Since about 6% of the individual tests were interpreted as inconclusive, 1% as unsatisfactory and 4% as unknown; there is some concern that, if we excluded all of these cases, the analyses of the remaining data would be suspect because of such a large group of missing cases in the subsequent statistical analyses. It is the old classical 80-20 problem. You spend 20% of your time to solve 80% of the problem and 80% to solve the remaining 20%.

Our solution was to do a "quick fix". After more consideration, it was determined that ETA would be valid in analyzing pairs of tests with one or both tests coded as "inconclusive". This category of interpretation of the tests is based, for the most part, on failure of the patient to achieve an arbitrarily set heart rate. Based on clinical considerations, it was also found that an evaluation by ETA would not be valid for tests where one or both tests in the pair being evaluated were coded as "unknown".

The largest and most troubling category, "unsatisfactory", remains unresolved. It appears that an intensive effort by the knowledge engineer and domain expert would allow us to expand ETA so that its evaluation of at least some of the cases coded unsatisfactory could be analysed. For now, we have abandoned the effort and decided not to evaluate pairs of tests where one or both of the cases are coded "unsatisfactory". Even so, ETA does at least as well as human analysts do, but we had hoped to do better than that. We may eventually come back to this problem.

The lesson learned from this experience is that one should expect to encounter many messy details that have to be resolved before the production system can be made operational, including some things thought to be resolved in the development phase.

**Interfacing the Components**

ESCA provides a global assessment of a pair of x-ray cines. This means that film has to be examined and evaluated, something ESCA cannot do. Specifically, someone must look at the film and determine the size and location of lesions (lipid deposits that can restrict blood flow) in the coronary arteries. These data provide the essential input for ESCA. Although some clinical judgement enters into determining the location and size of lesions, ESCA is built on the assumption that a technician can be trained to locate and measure the lesions if it was worth the clinical judgement is done by ESCA.

Thus, the development of the production
system for ESCA has taken a very different approach than that of ETA. The environment in which ESCA was used included a room with two special projectors where each of the pair of coronary arteriograms being evaluated can be viewed side by side and a terminal where the technician enters his findings interactively into the computer. The technician works in this setting and locates and measures the lesions and changes in lesion size that he sees and enters these data into the system.

The production system has a menu screen that prompts the technician to enter all of the data required by ESCA. It allows him to verify, backtrack, change and eventually enter the data. These data are then entered into a file. For efficiency reasons, ESCA is invoked to process these cases at some future appropriate time in a "batch" mode. The batches are in the tens and hundreds, however, instead of the thousands processed using ETA.

The lesson from this experience is that each production model needs to have some customized features.

Output Format

An interesting quirk came up in connection with the ESCA system. The protocol used by the panel of human experts is deliberately designed to force the human experts to decide which case is better, even when there is very little visible change. In theory, there is no such thing as "no change". We did not want the evaluators to take the easy way out by allowing them to simply declare "no change". It was feared that the code would be used too frequently. It is for this reason that the 0+ and 0- codes previously described were used. All of these precautions are not really necessary in ESCA. Such "psychological" precautions are not needed in an automated system; not yet anyway. Even so, ESCA can produce the two categories of "no change". It will be interesting to see what, if any, difference this feature will make in the final analysis.

AI Concerns

Artificial intelligence (AI) has provided the basis for our experimentation with new ways to analyze clinical research data. This created some conflicts for us in regards to our practical real world environment. What about trying still other interesting AI methods? Should we seek to advance AI technology by actually developing new methods? How far do we go in the pursuit of new and interesting ideas? After all, in this context, POSCH must operate in the real world of production work.

These questions are especially relevant in our situation, where graduate students in AI are a part of our development team. For this reason, we have not chosen an entirely consistent answer to the above questions. Our fundamental concern has been to focus on and give priority to the real world of data analysis, that is, to get systems that will allow us to actually analyze the POSCH clinical trial data.

In at least one instance we have found a new idea to be irresistible. We now have a project underway to see if we can build a neural network that can learn how to produce a global assessment of our serial coronary arteriography data as well as the knowledge engineered ESCA system and as well as our panel of human experts. We have also considered using the POSCH data to experiment with other types of learning systems. How well these turn out remains to be seen.

Conclusion

This paper is designed to provide the benefit of our experience to others who are considering using AI techniques. We have emphasized issues related to large databases: situations, problems and solutions that others are apt to encounter. We have also discussed some problems that, although unique to our given situation, are typical of the kinds of problems one will encounter elsewhere. We hope our discussion of these unique problems will give others who are starting similar projects a feel for what to expect.

Here are some of the lessons learned:

1. Even though the technology is new, the old standards for building good systems still hold. For example, in selecting the development tools, both hardware and software, keep in mind the ultimate working environment in which the expert system must operate.

2. Earlier assumptions were that Lisp is an excellent development environment but is not efficient enough to run in a working environment. Good planning and newer more efficient Lisp software now leaves this issue an open question.

3. The expert system itself will ultimately be only one part of the production system, albeit a key one. Building the other modules represents a significant additional expense in time and costs.

4. The real world is full of messy details that must be resolved before the production system can be made operational. Dealing with them can double development time.

5. Using new techniques such as expert systems can produce results that take on new and, perhaps, unconventional forms. This might require finding new ways to look at, interpret, and analyze the results.

6. New ideas often feed upon themselves. Experimenting with expert systems
will often suggest even more innovative ideas that can be investigated. If you live in a rigorous real world production oriented environment, you may need to resist this trend and focus on the production work goal. If you are fortunate, you can pursue some of them and, perhaps, add to the body of knowledge.

REFERENCES


APPENDIX

Principal Investigators
Henry Buchwald, M.D., Ph.D.
Richard L. Varco, M.D., Ph.D.

University of Minnesota Clinic
Arthur S. Leon, M.D.
Jean Rindal, R.N., M.A.
Marsha Eide, R.N.

University of Arkansas Clinic
Gilbert S. Campbell, M.D., Ph.D.
Malcolm Pearce, M.D.
Joseph K. Bissett, M.D.
Meredith Stuenkel, R.N., B.S.
Elizabeth Barger, R.N.

University of So. California Clinic
Albert Yellin, M.D.
Alan Edmiston, M.D.
Dorothy Fujii
Julie Hatch, R.N.

Lankenau Hospital
Robert D. Smink, Jr., M.D.
Henry S. Savin, Jr., M.D.
Helene Brooks, B.S.
Rebecca Carins, R.N., M.S.
Elizabeth McAndrews, R.N., M.S.

Central ECG Laboratory
Naip Tuna, M.D., Ph.D.
James N. Karnegis, M.D.
James Stevenson, M.D.
Regina Brykovsky
Mark Linssen

Central Lipid Laboratory
Jane Speech, M.S.

Central Arteriography/Radiology Laboratory
Kurt Amplatz, M.D.
Miguel Sanmarco, M.D.
Wilfredo Castaneda, M.D.
David Hunter, M.D.
Jeffrey A. Brown, B.S.

Coordinating Center
John M. Long, Ed.D.
John P. Matts, Ph.D.
Laurie L. Pitch, M.P.H.
LaVerne L. Christie, M.S.
James W. Johnson, M.S.
Christian T. Campos, M.D.

Administration
Bernard A. Ley, B.S.
Betty Hansen, R.N.
Data Monitoring Committee
Thomas C. Chalmers, M.D. (Chairman)
Jacob E. Bearman, Ph.D.
Gerald R. Cooper, M.D., Ph.D.
Samuel W. Greenhouse, Ph.D.
J. Ward Kennedy, M.D.
Paul Meier, Ph.D.
Curtis L. Meinert, Ph.D.
Jeffrey Probstfield, M.D.
Jeremiah Stamler, M.D.
D. Eugene Strandness, M.D.

Mortality Review Committee
Jesse E. Edwards, M.D. (Chairman)
Lawrence S.C. Griffith, M.D.
Arthur J. Moss, M.D.
David Spain, M.D.

Policy and Data Monitoring Board
Antonio M. Gotto, Jr., M.D., D.Phil.
(Chairman)
C. Morton Hawkins, Sc.D.
James J. Leonard, M.D.
Floyd D. Loop, M.D.
Elliot Rapaport, M.D.
David L. Sylwester, Ph.D.
Doris Tulcin, B.A.

Consultants
David H. Blankenhorn, M.D.
William L. Holmes, Ph.D.
Richard B. Moore, M.D.
Manford Morris, Ph.D.