EXPERT SYSTEMS IN ARCHITECTURAL AND PLANNING EDUCATION

Dr Ian Cullen
Computer Unit
Bartlett School of Architecture and Planning
University College London

ABSTRACT

The paper discusses the problems and possibilities of a project initiated recently within the Bartlett in the general area of knowledge engineering. The aim is to assemble a set of knowledge bases which may be explored interactively by students. The system will differ from traditional CAL packages in that it will be both problem oriented - designed to extract from the students the information required to reach a specific decision - and capable of providing an explanation of its approach at any point.

July 1983

EXPERT SYSTEMS IN ARCHITECTURAL AND PLANNING EDUCATION

Dr Ian Cullen

Computer Unit

Bartlett School of Architecture and Planning

University College London

Introduction

The aim of this paper is to describe the early stages of a software development and implementation programme, designed both to reduce the workloads of architectural and planning members of teaching staff and to make Computer Assisted Learning (CAL) accessible to complete, and sometimes quite sizable, cohorts of both undergraduate and postgraduate students [1] within the context of some of the many intensive phases of studio and project teaching. The software developed so far is strictly "unintelligent" in the sense of that term most commonly accepted within the field of knowledge engineering, but its development is part of a strategy which will involve the production, at least of "expert" if not "intelligent" CAL packages [2].

Context

The context of the project to be described below is one which cannot be unfamiliar to schools of architecture and planning at the present time. On the one hand, a period of unrelenting pressure upon the resources made available for higher education in the UK has meant a significant reduction in the staff-student ratios which measure, albeit crudely,

[1] At the Bartlett School a typical undergraduate cohort contains between 45 and 55 students, almost all of whom are following an academic programme designed to lead to RIBA recognition. The various postgraduate programmes are each significantly smaller and not all directed towards professional qualifications.
[2] The classic, and still probably the simplest, definition of an intelligent machine is that of Turing who argued that the crucial criterion should be whether or not a human observer of a man-machine dialogue could tell which was which. Within this context, an expert system differs from a conventional computer program primarily in that it is capable of deductive (and sometimes even inductive inference) rather than being just passively responsive. In other words it is capable not only of responding to a given situation but also of reasoning about that situation.

our ability to teach the subject in a fashion which we have in the past found acceptable. This process of educational attrition has been experienced in a variety of ways, but two of the most painful have been the loss of up-to-the-minute specialist expertise and the erosion of labour intensive teaching modes. Much specialist support has in the past been provided by practitioners - civil engineers, psychologists, lawyers as well as architects and planners - who have been employed on a part time basis. Budgets for such staff are invariably amongst the most vulnerable in academic institutions and so they go first when the going gets rough. The costs of these losses are multiplied by the effect which they, and other enforced economies, have had upon one of the most central elements in the education of an architect or planner. The design studio and small group project modes of teaching are dependent, not only upon specialist inputs but also upon an expensive tutorial style of operation. Many institutions are now being forced to withdraw support from studio and project teaching simply because such teaching consumes a very large proportion of the available resources.

The ever increasing use of computers in professional offices has been interpreted as a response in the world of trade and commerce to market forces whose impacts there may be legitimately regarded as analogous to expenditure cuts in the public sector. one might, therefore, be forgiven for concluding that schools of architecture and planning have been adopting what is commonly labelled the "high technology" response to economic pressures for some years now, for there is certainly ample evidence that the vast majority of schools offer computer based techniques at some point to at least some of their students [3].

In fact it would not, in my view, be legitimate to draw the above conclusion. Clearly there has been an expanding interest in Computer Aided Architectural Design (CAAD) in the schools in recent years. Clearly this interest has mirrored, sometimes leading sometimes following, that of architectural practices. The paradox, however, is that techniques adopted as labour saving devices in the world of practice can have exactly the opposite effects in the world of education. This is because, no matter how closely related, the ultimate purposes of the two worlds are different ones. Thus as professional architects or planners adopt more and more new techniques - in many cases with a view to reducing the labour intensity of the design process - the schools are faced with the prospect of expanding their curricula in order to keep pace. The point about CAAD techniques is that they are directed towards the practical purposes of architects and planners, not the pedagogic ones of the teachers of the subjects. Thus, as with all new developments in the field of practice, they simply represent a broadening of the territory, all of which must be carefully patrolled by teachers of the subjects.

[3] A recent AJ survey (see Greig(1983)) indicated that no fewer than 39 UK schools of Architecture offered some form of computer experience to their students.

Moreover, whereas the introduction of a new building or design technique may mean extra work for somebody because this technique must be covered at some point within the course structure, the introduction of a new computer technique can have far more painful ramifications. First, the technique will in many cases affect the way in which non-computer courses are taught. The building technologist, the planning analyst or the structural engineer will have to get to grips with the way in which the use of the technique affects the application of his or her skill and thence the communication of that skill. Second, the computing staff (if there are any) will have to find ways of implementing the technique locally. This usually involves a choice of strategies between the two extremes of committing a small to moderate sum of money (for the purchase of the software) and a large amount of staff time (for bending it to local hardware circumstances), or committing a much larger sum of money (for the purchase of a hardware-software package, sometimes called a "turnkey system") and a much smaller amount of staff time (for working out and teaching other staff how to use it). Finally, any computer based courses which, amongst other things, exemplify the use of the machine, will have to be adjusted to accommodate yet another example.

Each of these sorts of adjustment are currently imposing very real burdens upon both the capital and staff time resources of architecture and planning schools at present. The computer is neither reducing the cost nor cutting down the effort of professional education in these spheres. It may be argued that at least the third of the problems quoted above is a transitional one and will disappear as the computer becomes fully integrated into all levels and modes of the teaching programme, but even if this is true that utopia is still some way off. For the foreseeable future there will continue to be separate CAAD courses which will need constant updating and a labour intensive mode of delivery.

The central point is that the computer is changing the practices of architect(ire and planning faster than it is changing the practice of higher education. The schools are therefore being forced to offer students experience of increasingly complex professional worlds whilst at the same time being denied the necessary didactic tools. Practitioners (and academics) are producing the requisite tools of practice, from computerised demographic and economic modelling techniques; through structural, energy, acoustic and lighting evaluation packages; to automated 2- and 3-dimensional graphics systems. But nobody is attempting to automate the process of teaching much above the primary level [4].

[41 The primary stage in the British education system goes up to the age of 11 years. However even at this level there is still an acute shortage of good CAL packages. Beyond this level the position gets progressively worse.

The problem is clearly reflected in the history of computer developments in the Bartlett School over the past decade. A Unit has existed ever since 1974 and has expanded more or less continuously ever since. It started with a 16k word PDP 11/10 (which is still used intensively today) and the spare time of the author. Now we operate a 16 line UNIX multi-access system, an autonomous graphics workstation (still, however, based on the old 11/10), and a range of desk top machines. There are now two full time members of staff and part time contributions from several others (as well as the author). Whether or not this represents a typical framework, the pattern of software development and acquisition must be similar, at least in general terms, to that of several other institutions. A few major programs have been developed from scratch (including a 3-D visualisation system and several planning analysis programs); a larger number of mainly technical programs have been acquired from elsewhere and adapted (including a drawing system and a variety of energy and lighting packages); and a few software development tools such as compilers and editors have been assembled. Increasing numbers of students have benefited from these developments and the course structures have been adapted more and more to accommodate them. Yet no-one, least of all those teachers involved in computer related initiatives, has experienced anything but increasing workloads.

The position in the Bartlett School, and I suspect the position in many other schools, is that students have "in principle" access to fairly sophisticated hardware and ,software, but "in practice" patterns of experience which fall far short of what they have a right to expect. This is partly because staffing levels are woefully inadequate, but also has a great deal to do with the way in which schools are deploying their effort in the areas of computer related research and development. The problem that we therefore face, and began to address just over a year ago now, is one of reorienting very limited resources so as to confront the problem of computer assisted education in architecture and planning directly.

The State of the Art

The most obvious place to start upon such a venture is with the software which is directed explicitly at the target of design or planning education. Unfortunately, and partly for, the reasons noted above, there appears to be very little of such software on the market. The only package we were able to find which covered topics close to our area of interest was a building management suite produced by Brian Fine(1981).

This allows students to work through a series of simulations each one designed to introduce techniques of financial appraisal and project scheduling. In many ways this a very interesting (and unique) set of programs. It has certainly been used to good effect by students on postgraduate building economics courses. However, it does have limitations. At the logistic end of the spectrum, it is tied very tightly to a particular hardware configuration (based upon the Commodore 8032). This severely limits its accessibility unless large numbers of these machines are acquired, so precluding all sorts of alternative applications which will not run on the PETs. At a more general level, the package suffers somewhat from the duality of its aims, for it is clearly intended both as an educational and an operational tool. Thus its use often requires considerable guidance (ergo its greater relevance to specialist postgraduate needs). moreover, since it often employs complex algorithms and accommodates large databases it can sometimes take a long time to run (with computation pauses of an hour or more being not uncommon).

Clearly this suite does not exhaust the range of relevant CAL programs. There are one or two others in the construction field as well as more generalised packages developed for teaching purposes in engineering, statistics and economics. The central difficulty, however, is that there is nothing available which can be used as an integral part of a design or planning project by all of the students taking that project.

Alternatives: (1) The Automated Brief Project

This project was the result of a collaborative effort involving both computer and teaching staff [5]. The aim was to produce a simple device which might be used at a variety of different stages of project work, and by a large number of students over a relatively short period of time. The role of the system, which comprises a relatively simple 'C' program and a series of scripts, is to provide the student with advice, commentary and (where appropriate) instruction as he or she moves through the major stages of a design project and in response to the decisions taken during preceding stages. Eventually, we hope to provide scripts which cover the following main stages:

(a) site selection and evaluation

(b) choice of building form and structural system

(c) technical and cost validation.

Up to now, however, we have tested the system only in the context of the early site selection phases of one major first: year architectural project [6].

The system is designed to operate in a crudely Socratic

[5] Apart from the author those most centrally involved were Marion Fuller, who actually wrote the software, and John Rae and Rob Hughes, respectively yearmaster and studio tutor in the first year programme at the Bartlett.
[6] This is a project which effectively runs over the last two terms of the year and involves detailed landscape evaluation followed by the design of a study or interpretation centre for Hampstead Heath. The exercise was devised by John Rae.

fashion. In other words, it asks the student a series of chained questions and offers its advice or instruction in response to his or her most recent replies. Each of these reveals some facet of the decisions taken by the student within the current phase of a project [7]. Thus the student might beasked about the access constraints and the machine might respond with reminders about emergency services, refuse disposal etc. Some of these comments might take the form of mandatory extensions to the project brief (ergo the system title) with the result that each student ends up working to a somewhat different brief, the differences being the products of early design decisions. Since the program is effectively doing no more (in its first phase at least) than controlling the presentation of a script, there are no limitations upon the form of the textual material presented. Thus inour first experimental exercise we structured the computer responses to look like a dialogue involving interested parties (planner, client, technologist etc). This extended the educational value of the exercise by demonstrating that a brief is not only a dynamically varying phenomenon but also the product and subject of controversy.

After this first phase of structured questioning and response, the program moves into a second phase of more open ended interrogation in which the student simply records his or her preliminary thoughts on the next stage of the design exercise, relating perhaps to building form, structural system and materials. Eventually this phase will serve the dual function of linking up the major stages of project work and assisting assessment. It will do this by keeping a record of all responses entered by each student, including these open ended 'first thoughts'. Then, when using the program for a later stage in the project, the student can be reminded of these initial ideas, and be invited to reconsider them in a more formal fashion (that is as part of another structured question and response session). Moreover, since the program creates a record file for each student who uses it and then appends details of all answers given, staff can monitor progress at any point in the project. The program in no sense "interprets" these verbatim responses.

The third phase of the program's operation, after all questions of both types have been asked and answered, involves a reconsideration of the replies collected in the first phase of formalised questioning. At this point the student is encouraged to question the overall technical or logical consistency of his or her current framework of design decisions. What the program does is to compare the profile of those decisions with a symmetrical matrix of potential conflicts. This is constructed by members

[7] The program as it stands is not "intelligent" in the sense described earlier in this paper. This lack of intelligence is reflected in its inability to interpret natural language. Thus students are restricted to selecting from multiple response questions those responses which most closely reflects their decisions and then recording these facts by typing codes (y or n, 1,2,3...etc).

of teaching staff as part of the script preparation, and simply relates each potential answer to every other, flagging all potential inconsistencies [8].

As already mentioned, the program does not itself rely upon or reflect any particular model of intelligence. This is because the inference system, such as it is, is built separately into each of the scripts, or knowledge bases, which the software interrogates. This software can therefore remain purely passive in its mode of interaction with the knowledge base, as is indicated by the design flow chart (Figure 1) below.

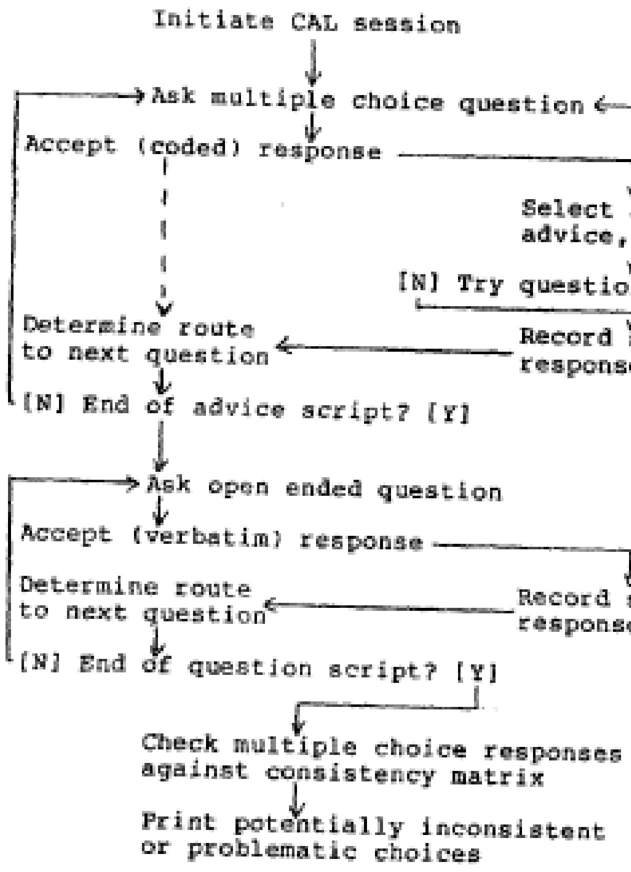
Initiate CAL session
Ask multiple choice question
Accept (coded) response
Select and print advice, comments etc
[N] Try question again? [Y]
Determine route to next question
Record student's response
[N] End of advice script? [Y]
Ask open ended question
Accept (verbatim) response
Determine route to next question
Record student's response
[N] End of question script? [Y]
Check multiple choice responses against consistency matrix
Print potentially inconsistent or problematic choices

Figure 1. Structure of the Automated Brief program

[8] These are not presented to sophisticated undergraduates as illogicalities but rather as relationships between decisions which might be usefully reconsidered.

The route selection part of the program's first phase is built into the question script since each permissible coded response is associated with a pointer to a section of the advice or instruction script.

The advantages of this approach do not end with the simplicity of the programming task. More important by far is the system's ability to handle a very wide variety of knowledge bases. Indeed, any teaching context in which it is appropriate to provide advice or information in a selectively sequential fashion might be handled in this way. There are, however, serious disadvantages, and these correspond very closely to some of the major differences between "intelligent" and "unintelligent" software.

These disadvantages all stem from the fact, mentioned above, that the approach depends an the assignment of a totally prespecified inference pattern to each unique knowledge base, rather than on the application of an independently designed inference technique which controls the exploration of the base. This means, first and most obviously, that the approach is incapable of accommodating an unpredicted profile of circumstances generated by the student. Thus if a student tries to skip, temporarily, one set of questions because he or she wishes to sort out some other facet of the problem first, the program cannot cope. The order of the dialogue is built into the question script. Second, the system is incapable of coming to a reasoned conclusion or of offering advice which is precisely specific to the student's own problem. Again the reason is essentially the same. The knowledge base contains an inference pattern which the purely passive software can only "unfold" to the student. The unfolding is selective but not actively applied to a problem because such an application would imply a means of drawing conclusions which was not wholly contained in a necessarily generalised knowledge base. Finally, the system cannot provide any really useful self-explanatory facility. If the user asked why a question was put the program could at best assume the wording was inadequate and offer amplification from yet another script. If, however, the student's problem was with the relevance of the question, the program could do no more than boringly repeat the preceding sequence since there is no more to its logic than this built in questioning route. It cannot explain its localised approach to and purpose of information assembly because neither exist to be explained.

Alternatives: (2) The Expert Systems Project

It was for just these sorts of reasons that we began, at about the same time, to explore more ambitious avenues, and more specifically those being opened up as a result of work in the field of artificial intelligence. Since the limitations of the above experiments are not unreasonably summarised in the phrase "lack of intelligence" this seemed an obvious place to start. However, the difficulties were always bound to be much greater, notwithstanding the special demands we imposed upon the venture, simply because of the complexity and indeed the controversial nature of the field. Despite recent government interest and support, the general aim of producing practically useful "intelligent" software across a broad spectrum of application areas is still some way off.

Given the difficult and uncertain status of the whole field of knowledge engineering, it may appear foolhardy to superimpose further exacting conditions upon software development exercises. However, there is no point in launching into a field of research just for its own sake, and so any major initiatives within the Bartlett must remain consistent with the overall context outlined above. In other words, programs must be simultaneously accessible to a reasonably large number of students. They must be applicable to central problems encountered in project work and must be capable of assisting in their solution. And they must present such assistance in a didactic fashion so that students actually learn from the help that they receive in solving their problems.

We know of no current studies attempting to apply artificial intelligence in this way within the professional fields of architecture and planning. There has been a fair amount of work on "intelligent" tutoring systems, perhaps the best known being Carbonell's (1970) SCHOLAR program which participates with a child in a mixed initiative dialogue on the subject of South American Geography. Within the areas of architecture and construction an interesting paper by Lansdown (1982) explores some of the potential of expert systems in these related professions, but does not treat their educational value. Thus he concentrates exclusively upon the opportunities for using expert systems in architect's offices in tasks such as diagnosing airconditioning problems and interrogating an extensive tree database for landscape design.

Our interests, however, suggested different applications. We have not yet worked up a comprehensive list since the project is still at an early stage of development, but some which may prove susceptible to this approach include:

(a) the evaluation of site location choice;

(b) the analysis of development control decisions;

(c) the selection of a structural system and related materials; and

(d) the performance of a preliminary cost and resources estimate.

The choice of a software development strategy proved to be a highly constrained exercise since we did not have the resources for the design and production of our own purpose specific artificial intelligence systems. We were therefore faced with the very limited range of general purpose expert system generators currently available. Our eventual choice was a program called Micro Expert (see Cox (1982)). This is a small advice language system intended to facilitate the design and use of expert models in a variety of fields. It first of all helps the domain expert to formalise his or her understanding in a way which is relevant to the solution of specific decision problems, and then interrogates the final user about the characteristics of any such problem on the basis of the inference system and knowledge base acquired from the expert. The knowledge base which is assembled is one which takes the overall structure of a production system (see Winston (1977) for a simple discussion of the various main techniques of knowledge engineering). In other words, knowledge is expressed generally as a set of rules in each of which there is a situation recognising part and an action part. The action is triggered when the situation is encountered. Constructing the model is thus tantamount to describing the tree-like structure which links a large number of production rules together in such a way that their resolution eventually and coherently leads to a decision on the central issue. This inference system is constructed in detail in much the same way as in the task-specific program PROSPECTOR (see Duda (1980)). Production rules are thus generally linked in a probabilistic fashion and the process of reaching a decision becomes that of establishing the plausibility of a central hypothesis rather than that of achieving absolute certainty upon the matter.

This program was chosen partly for logistic reasons and partly because of the pertinence of its design to our particular needs. At the level of logistics its advantages have to do with its small size and portability. Written in UCSD Pascal, it will run on a variety of modest micros and should not be too difficult to recompile under UNIX [9]. Eventually we hope that the adoption of this program will enable us to make a variety of expert models accessible to significant numbers of students within the sorts of tightly timetabled studio and project contexts alluded to above.

The first of the requirements listed at the beginning of this section should, therefore, be satisfied by applications of Micro Expert. The second and third, however, depend critically upon the design of the program itself, for they relate to its capacity to provide problem solving assistance during the course of project work and to do so in a way that is educative as well as technically of an expert standard.

Though the program has not yet been tested in studio exercises, we are optimistic that it will satisfy these further requirements. Its chief virtue seems to be that it is does provide a powerful set of tools for the construction of efficient hypothesis testing models. Its basis in Bayesian statistics means that it will cope with the large set of problems in which it is wholly unrealistic to talk in terms of complete certainty, but not unreasonable to expect domain experts to be able to assign probabilities to the

[9] In the current development phase we have chosen to avoid tampering with the idiosyncrasies of the program code and run it by implementing a UCSD interpreter which sits on top of the UNIX system. This is an inefficient operating environment but has enabled us to experiment with the program earlier than might otherwise have been possible.

relationships between information and hypotheses or decisions. Thus an experienced developer may be willing and able to say roughly how important public transport facilities are in the assessment of a potential site for a particular purpose but not that there is any precise relationship of logical necessity. The Bayesian features of the program allow for an element of logical fuzziness in the formalisation of domain expertise and for inexact sciences such as architecture and planning this has to be an important advantage. The program is nonetheless efficient in that it can accommodate a large variety of logical relationships between production rules (that is apart from fuzzy ones), and it adopts an effective and clear backward chaining system of inference in applying a model to a particular case. In other words, it seeks only the information it actually needs to establish the probability of a target hypothesis.

Perhaps of even greater importance in our circumstances, however. is the didactic potential of the system. For once a model has been constructed and tested, it may be applied by students in ways which should prove to be most illuminating. For a start, the system lends itself to the project mode in that it is intrinsically problem or decision oriented. Thus it is applicable in what is still the central teaching context for architecture and planning. Furthermore, it demands of the domain expert (and thus in our case the teacher) a great deal of rigorous preliminary thought. However, the educational payoff derives from the fact that that work thenceforward becomes accessible equally to all who use the model. The system does not become tired. There is no expert system analogue of the Friday afternoon tutorial. Moreover, the efficiency of the system should mean that there are not too many Friday afternoon tutees. Use of the system should remain interesting since it should always appear pertinent to the student's own problem. And finally, when its pertinence does start to seem obscure, it can always explain itself. Just as we would expect a design tutor or domain expert to he able to justify his or her rea soning, so Micro Expert is capable at any stage of explaining why it is seeking information by showing how that information will contribute to reaching a decision or verifying a hypotheses.

As noted above, the program has not yet been tested in project teaching. However, a very simple demonstration model has been constructed, and it may be helpful to outline the structure and operation of this in order to indicate something of the approach. The model relates to the first of the application areas mentioned earlier. In namely that of location choice. Specifically, it is intended to assess the suitability of a for commercial development [10] The best way to explain the model is through a tree diagram, since this is effectively what is reconstructed by the program

[10] The model was designed purely to demonstrate the system. It is not intended as a genuinely expert or comprehensive representation of the site selection process.

from the production rules provide by the domain expert (in this case, perhaps, a developer, surveyor or estate agent). Each branch of the tree represents a condition of logical or probabilistic dependence and the end of each branch represents an information requirement, to be satisfied either by questioning the user or through further, as yet unspecified, backward chaining until direct interrogation does become necessary. In other words the model described in Figure 2 is extendible along any branch as far as is consistent with an expert understanding of the problem area.

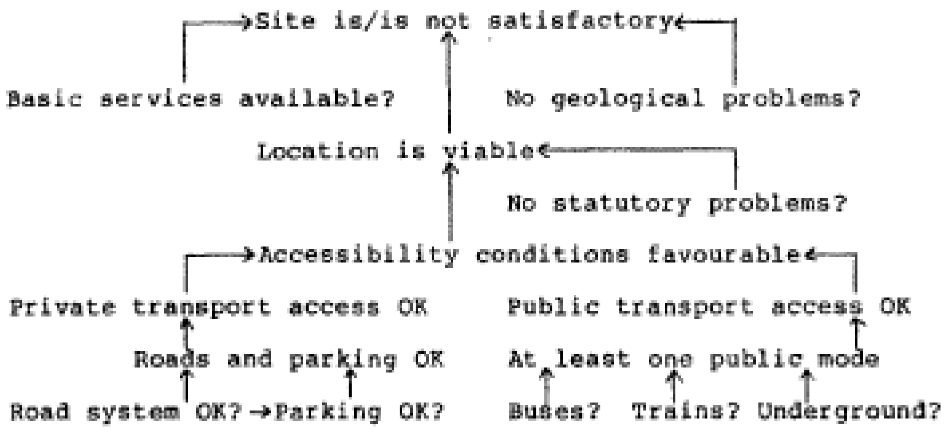

Figure 2. A demonstration model for expert site selection.

The educational potential of a sophisticated version of a model of this sort is clear. Not only will it improve the quality of students' design decisions (and thus probably the quality of their designs as well). It should also, through its logical pattern of questioning and self-explanation, teach them a great deal about what lies beneath and behind such decisions. However, though we have yet to discover the problems and limitations of this approach through experience of its use in the studio, it is already clear that there are likely to be difficulties. The manner of its implementation has already been mentioned, but it is worth pointing out that 6 or 7 simultaneous users of a largish model (a target already achieved with the automated brief program) would almost certainly bring our 11/44 to its knees. Implementation problems are, however, generally soluble. Design ones are likely to prove less tractable. The most serious of these that we have so far discovered is what may be best thought of as the syntactic gap between the approaches of the program and those of the human expert, at least in the fields of architecture and planning. Domain specific models of any significance are in fact very difficult to construct because all of the intangible wisdom, technique and knowledge of the expert has to be converted into the sorts of Bayesian production rules described above. Though the probabilistic elements in one sense extend the applicability of the system by allowing for all shades of uncertainty, they also increase the complexity of the modelling task by increasing the variety of inter-rule relationships which has to be handled. It is not inaccurate to describe this approach to expert systems as a synthesis of a branch of symbolic logic (the predicate calculus) with a branch of inferential statistics (Bayes' Theorem). Though this is a no doubt efficient way of solving a class of problems, it cannot be considered a mirror image of the approach adopted by an architect or a planner. So if the system is to be useful in education (or, for that matter, in practice) there will for many years be a central role for an interpreter, known in the jargon as a "knowledge engineer", whose function is to convert the domain experts skills into the unnatural syntactical form of a probabilistic production system.

Conclusions

This requirement may not be a wholly bad thing. One of the complaints most frequently levelled at architects and planners is that their approaches are unnecessarily vague. if a handful of knowledge engineers, armed with unwieldy but brutally logical and efficient expert systems generators, were to be sprinkled around the architecture and planning schools of the world, their catalytic effect upon the precision of design thinking might be considerable. More seriously, and particularly in an educational context, it must be beneficial for students to learn from the example of a rigorous approach to problem analysis and solution. It may be that one characteristic of a human expert is that he or she has developed the skill, over many years, of intuitively short circuiting laborious chains of reasoning. This does not mean that they do not remain implicit in any expert's solution, nor that an inexpert student can avoid the experience of learning from observing the implicit rigour.

However, it is equally clear that much of what we value in the processes of applying architectural and planning expertise is missed by a system such as that discussed above. This must limit its accessibility and effectiveness as a decision tool and thus its acceptability in a teaching environment. The answer is probably twofold. First, as with any new technology and particularly one with the potential breadth of appeal of artificial intelligence, it should not be oversold at the outset. It should be introduced into professional higher education, as I believe we are attempting to introduce it in the Bartlett, one cautious and well tested step at a time. But this should not slow down the pace of research which will eventually broaden its applicability. One example of work which will very soon hit the market place of packaged software is that of Quinlan (1982) in Australia. This has produced a technique for assembling and structuring a production system type of knowledge base as it were inductively. In other words all it demands of the human expert is a series of examples of "expert" decisions along with anything else that is thought by the expert to be in some way (not prespecified) relevant to each decision. The system then extracts a parsimonious set of rules to account for as big a proportion of the case decisions as possible, so neatly overcoming one of the major problems of the sort of system discussed in the preceding section. Many further problems, of course, still remain to be solved, but then this is only one example of potentially applicable research.

Acknowledgments

I am most grateful to colleagues in the Bartlett School whose interest and effort have greatly facilitated the preparation of this paper. Thanks are due to John Rae and Rob Hughes for their invaluable contributions to the automated brief project, and most of all to Marion Fuller for the willing contribution of her skills as a programmer and an advisor. Responsibility for the views expressed remains of course mine alone.

References

CARBONELL, J. R., "AI in CAI: an artificial intelligence approach to computer-assisted instruction," IEEE transactions on man-machine systems Vol. 11, pp.190-202 (1970).

COX, PHIL, "Micro Expert: an advice language for expert systems," SWURCC Microprocessor Software Quarterly Vol. 6, pp.19-29 (1982).

DUDA, R.O., "The PROSPECTOR system for mineral exploration," SRI Rept 8172, Artificial Intelligence Center, SRI International, Menlo Park (D).

FINE, BRIAN, Construction Management Laboratory, Fine, Curtis and Gross, London (1981).

GREIG, JONATHAN, "Computing in the schools," Architects Journal, pp.90-91.

LANSDOWN, J., Expert systems: their impact on the construction industry, RIBA conference fund, London (1982).

QUINLAN, J. R., "Semi-autonomous acquisition of pattern based knowledge," pp. 159-172 in Machine Intelligence 10, ed. J. E. Hayes, D. Michie and Y-H Rao, Ellis Horwood, Chichester.

WINSTON, P. H., Artificial Intelligence, Addison-Wesley, Reading (1977).