A general purpose architecture for traditional CAI using an advanced authoring system

Dick J. Bierman & Volkert Balk
University of Amsterdam
Weesperplein 8
1018 XA, Amsterdam, The Netherlands

Keywords: 'Intelligent' features in orthodox systems
Architecture of computerized tools
Discovery learning and Structured teaching

Abstract
A shell is described which allows developers of traditional Courseware to build upon ideas and an approach normally only found in the research of 'Intelligent' Tutoring Systems. This approach is highly structured and allows for distinct representation of diagnostic and tutoring knowledge. A separate discussion is devoted to the incorporation of 'discovery' oriented features in the the otherwise highly structured 'intelligent tutoring' approach.

Introduction

Tutoring systems might be labeled 'Intelligent' if:

a. They are able to solve the problems which might be presented to the student and
b They can reproduce incorrect student's behaviour.
c. The architecture is such that educational knowledge of these systems is explicitly represented.

The general feeling in the ITS research community seems to be that at present for domains more complex than arithmetic (Brown & Burton, 1975) the reproduction of incorrect behaviour, for instance by trying several malrules from a bug-library, will result in combinatorial explosion and therefore should be controlled by domain-dependent meta-rules. This introduction of domain-dependent knowledge in the diagnostic component of the ITS does decrease the generality of ITS implementations. A part from the problems of diagnosing the wrong knowledge that the student uses some researchers do doubt that pointing out which 'malrule' has been used does produce better learning than just showing the correct solution (Sleeman, 1987, EARLI Tübingen).
Therefore it might be that traditional CAI systems upgraded with 'intelligent' features will prove to be more useful in the short term than ITS at least for practical purposes.
The third aspect of ITS, dealing with which knowledge has to be represented, in which way and how the different inference machines do cooperate is one of the 'intelligent' features that might be transferred to traditional CAI systems. In the

present paper a structured approach in which the knowledge necessary for different functions (like diagnosing and tutoring) is represented separately and explicitly within a traditional (non-'intelligent') authoring environment, is described. Such an authoring environment is not to be confused with an intelligent authoring system that either embodies knowledge about teaching (Merill, 1989) or a more stupid authoring system that creates intelligent tutoring systems (Bierman, 1988). The environment is an empty system (no domain knowledge is embedded) often called a 'shell'. The shell in some way reflects an underlying model of educational strategy which of course can not be applied in every case. The current shell can be applied in a wide range of domains. The shell functions also as a guide-line in the instructional design process.

Goal

The goal of the described architecture is to create an empty shell in CoA (Authorware Professional[1]) that will guide the developer in the design process and which results in educationally valid CAI programs that are easy to maintain and allow for student initiative. It should be emphasized that the shell nor the authoring environment with which it is created is in any way 'intelligent' in the sense as described above although it has some characteristics which by some authors are considered to be 'intelligent' (Allen & Szabo, 1990).

Assumptions

The following assumptions have been made:

a. The domain knowledge can be described in terms of topics and sub-topics.
b. According to an initial 'student-model' it is possible to generate an initial sequential teaching plan (called an 'agenda').
c. There exists a set of cognitive diagnoses which might result in a revision of the original teaching plan.

The initial 'student model' can be that of an ideal student. The real student generally will be less than ideal and consequentially diagnoses often will result in 'insertions' of tutorial transactions in the teaching plan. The initial model might on the other hand be that of an average student or even a very low ability student. In the latter case very often diagnoses might result in removal of planned tutorial transactions from the teaching plan.

Implementation

The major component in the shell is the instructional-event loop. This is a loop which considers the teaching-plan (called the 'agenda') and selects the next action to be performed. In fig. 1 an agenda is shown.

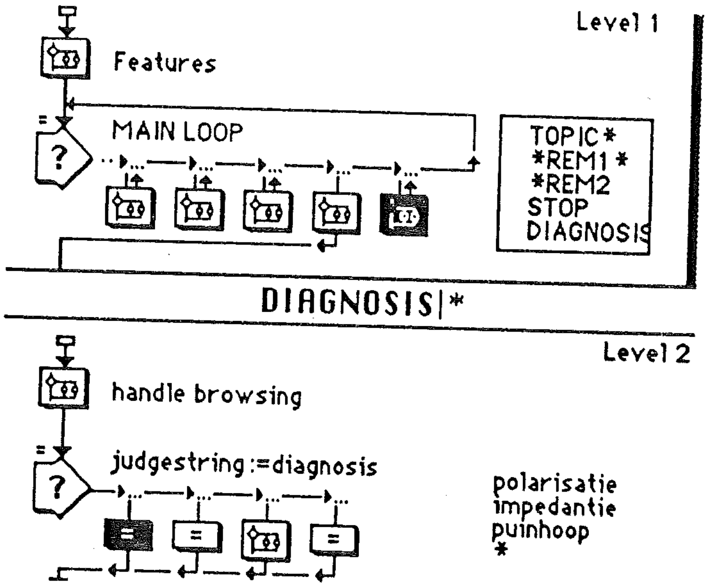
---

[1] Authorware Professional, (formerly called Course of Action)is a traditional visual (flow chart based) authoring environment.

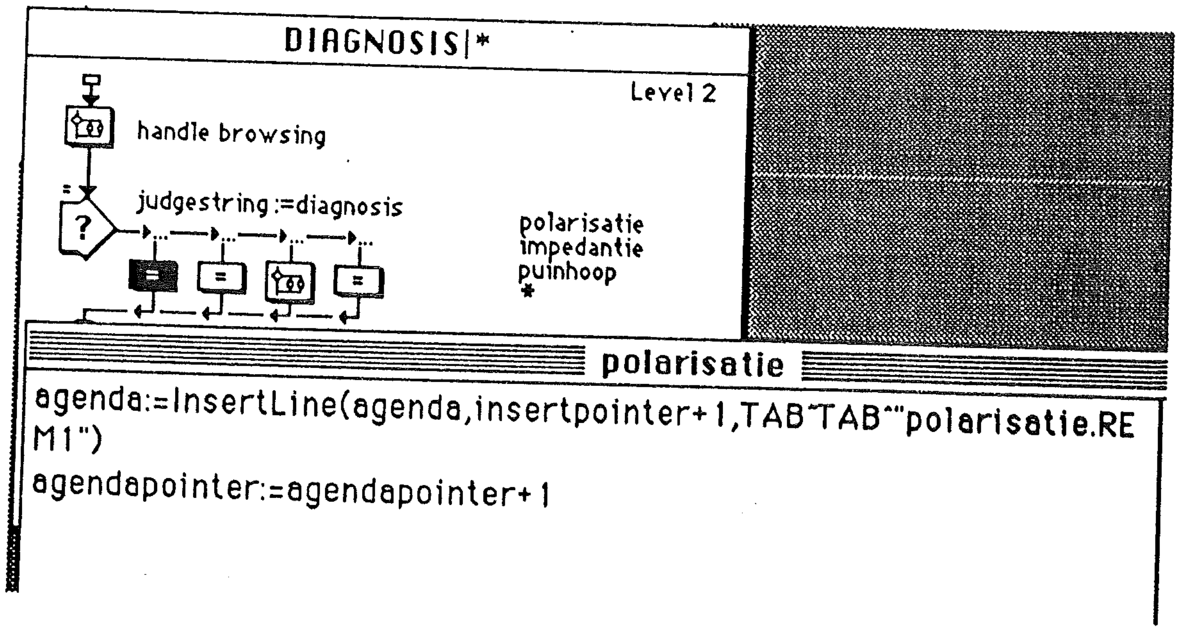
```
ACTIE:TOPIC.ELECTRODEN
________________________________________

AGENDA
TOPIC.intro
•TOPIC.ELECTRODEN
 TOPIC_ELECTRODEN_huidprep
 TOPIC_ELECTRODEN_onderhoud
TOPIC.FYS.VARS
 TOPIC.FYS.VARS_classificatie
 TOPIC.FYS.VARS_gsr_maten
 TOPIC.FYS.VARS_gsr_versterkersoort
 TOPIC.FYS.VARS_ademhaling
TOPIC.FYS.APP.
 TOPIC.FYS.APP_diff.verst
 TOPIC.FYS.APP_dc.verst
 TOPIC.FYS.APP_integrator
 TOPIC.FYS.APP_hartrate
 TOPIC.FYS.APP_versterking
 TOPIC.FYS.APP_impedantie
STOP
```

This is the agenda that the student will see if he/she asks for the current status. Each line of the agenda contains a teaching action. The agenda is constructed in the very beginning of the program. A dot-marker in the agenda indicates the current agenda-pointer. In fig. 2 it can be seen that on the uppermost level (1) there is a main loop

This is done by letting the Judgestring take the value of the line in the agenda where the agenda-pointer is pointing to. These lines contain words like "TOPIC_ELECTRODEN_onderhoud" (representing a topic_subtopic description). The Judgestring will be matched at this level against containing the word 'TOPIC' and will fall through that icon. In this way the action is decoded (in 3 subsequent levels) leading to the corresponding icon which actually performs the teaching-action. As a result of the teaching action the variable 'diagnosis' which started with a value 'OK' might change. As can been seen in the figure the icon 'TOPIC*' will be exited in such a way that in the same loop control will be passed to the 'DIAGNOSIS' icon (which matches on all values of Judgestring).



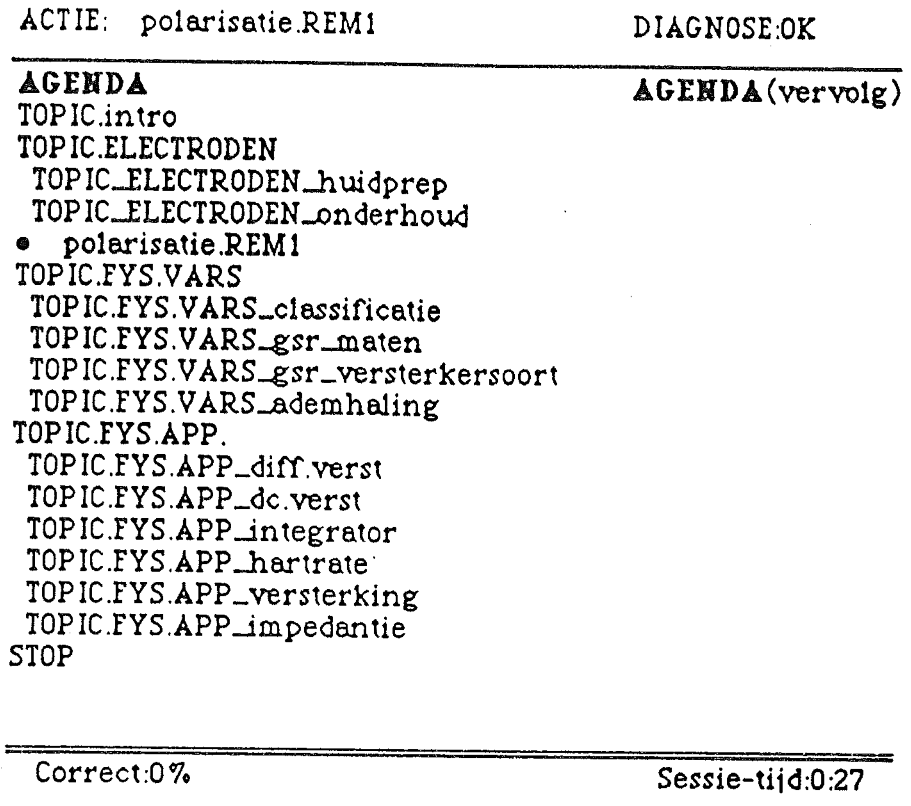
Within the diagnosis icon first potential 'browsing' by the student is handled. Then by a similar construct as used in the main loop the diagnosis is decoded. Each diagnosis (except 'OK') results in an adaptation of the agenda. For instance in fig. 3 the diagnosis 'polarisatie' results in an insertion of the action 'polarisatie.REM1'. The agendapointer is incremented so that this action will be performed right away. It is also possible to insert actions that will be postponed or that actions are inserted before the current action.
After the adjustment of the agenda the main event loop starts again. In this specific instance the control will pass through *REM1* at level 1 to the proper icon at level 3.



```
agenda:=InsertLine(agenda,insertpointer+1,TAB^TAB^"polarisatie.RE
M1")
agendapointer:=agendapointer+1
```

The diagnosis icon can be seen as a set of explicitly formulated production rules of the form
IF (some diagnosis) THEN (some agenda adjustment).

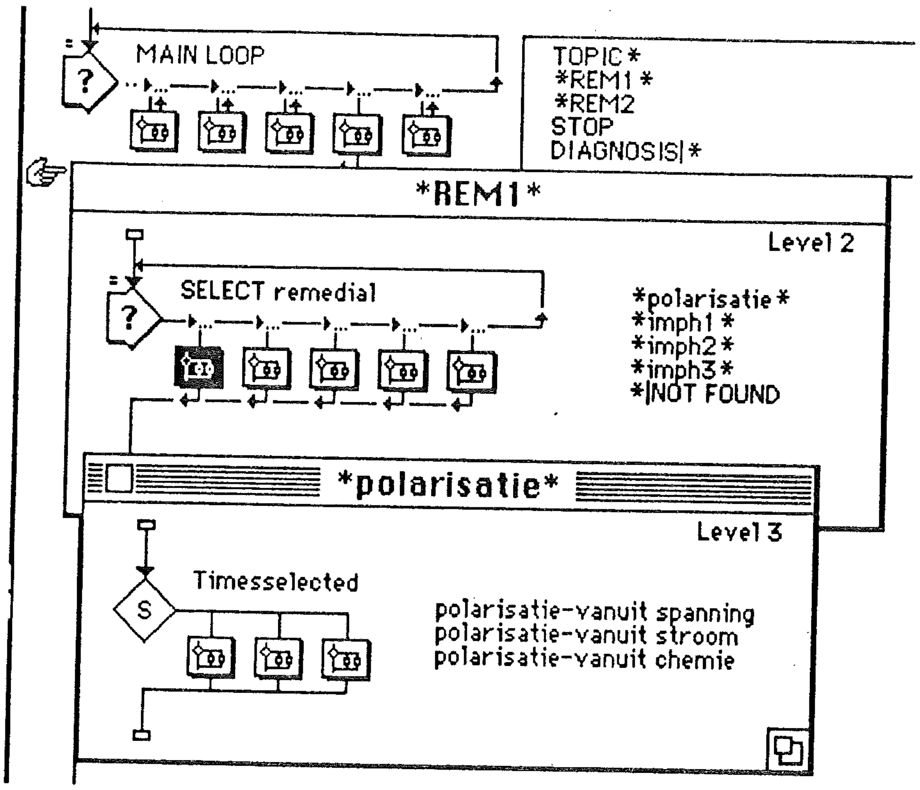
Thus all knowledge relating to adaptation of teaching-plans as a function of diagnosed misconceptions is stored locally in that icon. The resulting agenda can be seen in fig. 4



```
ACTIE:  polarisatie.REM1                          DIAGNOSE:OK

AGENDA                                            AGENDA(vervolg)
TOPIC.intro
TOPIC.ELECTRODEN
 TOPIC_ELECTRODEN_huidprep
 TOPIC_ELECTRODEN_onderhoud
•  polarisatie.REM1
TOPIC.FYS.VARS
 TOPIC.FYS.VARS_classificatie
 TOPIC.FYS.VARS_gsr_maten
 TOPIC.FYS.VARS_gsr_versterkersoort
 TOPIC.FYS.VARS_ademhaling
TOPIC.FYS.APP.
 TOPIC.FYS.APP_diff.verst
 TOPIC.FYS.APP_dc.verst
 TOPIC.FYS.APP_integrator
 TOPIC.FYS.APP_hartrate
 TOPIC.FYS.APP_versterking
 TOPIC.FYS.APP_impedantie
STOP

 Correct:0%                                       Sessie-tijd:0:27
```

Design consequences

An advantage of this approach is that the educational designer only has to worry about the topics and subtopics he wants to treat and on the possible misconceptions that he might diagnose. Thus the architecture structures the design process. In the current shell we follow the strategy that if misconceptions are diagnosed several times in one session the repair should use different perspectives subsequently. This is implemented by supplying the *REM* icons with multiple repairs arranged within a decision icon so that each time the next repair will be used. The designer does not have to bother with when a certain repair will be used exactly. In fig 5 this 'multiple perspective' remedial approach is sketched for the misconception 'polarisatie'.



Three different perspectives are to explain polarisation from the concept 'current', from the concept 'potential' and from the concept 'chemical'. Thus the architecture stimulates to explicitly represent the knowledge to remedy certain misconceptions.

Discovery oriented vs highly structured teaching systems

The amount of student initiative which is allowed by the educational systems is thought to be a relevant factor in learning. There are two rather opposite schools:

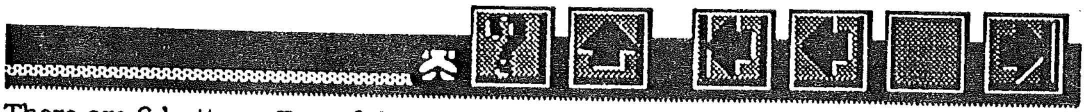
a. Those who believe that students can only learn important concepts and relations through discovery (Papert, 1980). This school advocates complete student freedom. The student is allowed to 'browse' the knowledge base at his or her own initiative.

b. Those who believe that learning should be based upon an already present corpse of student knowledge. This school advocates diagnosis-based structured teaching. Apparently the shell described here is very suitable to incorporate the latter approach.

However 'browsing' can and has be incorporated in the following way.

Browsing

In fig. 6 the lower half of the student presentation screen is shown.



There are 6 buttons. Two of them allow the student to 'jump' to the previous topic or subtopic. Two others allow to restart the current sub-topic or the jump to the next sub-topic or topic. These browsing functions are implemented by manipulating the agenda-pointer. When browsing a little frog appears and the handling of diagnosis

changes: the student has to decide whether his playing around will be taken seriously (and thus new remedials will be inserted and the point of origin will be forgotten) or that he/she is just fooling around to see feedbacks on other answers.
Left to these browsing buttons is a progress indicator. The progress is calculated on the basis of the length of the agenda and the value of the current agenda-pointer. Clicking on the progress-calculator results in display of the complete agenda.

Although both schools seem orthogonal there have been several approaches to reconcile both approaches (Eg. Bierman, 1986; Bierman & Kamsteeg, 1989 and Elsom-Cook, 1990).
It can therefore be argued that an authoring environment should be able to develop courses which allow also for free browsing of the knowledge base by the student. A problem when allowing for browsing is that browsing treats the knowledge base as intrinsically flat. However there are hardly domains that have no structure, where all items are of the same level of importance. Allowing for browsing (jumping through the knowledge base) therefore requires some important design decisions. Should the student be allowed to enter each item or should he/she get only access to a limited set of items. Related to this problem is the question about stepping back through the course. In a completely unstructured domain stepping back would be going through the knowledge base items just visited before in reverse order (going back in history). If there is any structure (Eg. a linear structure) stepping back could also mean logically stepping to the previous knowledge item in the domain structure (by decrementing the agenda-pointer).

Conclusions

It appears that certain aspects of 'Intelligent' tutoring systems, can be implemented if one uses traditional authoring environments. In the present work we focussed on the explicit representation of the educational knowledge. One disadvantage of the present approach is that contrary to most ITS the inference machine (the main control loop in our shell) and the educational (for instance the diagnostic) knowledge are not nicely separated.

The discussion on structured vs discovery oriented teaching systems shows that both approaches are difficult but not impossible to reconcile in the same architecture. At least the designer is forced to think about the constraints he wants to impose on the amount of student 'initiative'. For instance in the current implementation, which was geared towards highly structured domains, jumping forward is only allowed to the point where the student started browsing.

It should be stressed that in the present shell the way in which the agenda is structured (topics, subtopics and remedials) is not mandatory. This structuring was found to help the designer if he was working from an agenda which is based upon an average or good student. Starting from an agenda based upon a weak student might lead to another structure of the agenda. Also diagnosing will have a slightly different character and will generally result in deleting topics from the agenda rather than inserting them.

References.

Allen, M & Szabo, M (1990) *Dimensions of Authoring Aids Intelligence.* In G. Goos & J. Hartmanis (Eds) Lecture Notes in Computer Science: Computer Assisted Learning. Springer Verlag New York, pp.327-349.

Bierman, D.J. (1988) *Intelligent Authoring systems, towards better Courseware.* In: T.Reeves (Ed.) Proceedings of the ADCIS-Soviet Academy of Sciences conf., Urgench oct. 1988, ADCIS, Bellingham, 1988, p.31-40.

Bierman, D.J. (1986). 'Intelligent' Simulation Environments in Education. *Proc. of the 7th European Conference on Artificial Intelligence,* Part II- 84-90. Brighton, July 1986.

Brown, J.S. & Burton, R.R. (1975) Diagnostic models for procedural bugs in basic mathematical skills. Cognitive Science, 2, pp. 155-192.

Elsom-Cook (1990) Guided Discovery & ITS.

Kamsteeg, P.A. & Bierman, D.J. (1989). *Teacher thinking, student thinking: Think aloud protocol analysis in tutoring systems research.* In: Peter Goodyear (Ed.) Teaching Knowledge and Intelligent Tutoring. Ablex Corporation, Norwood, NJ. chapter 9.

Kamsteeg, P.A. (1989) Cognitive ATI Research: A simulated Laboratory Environment in (PCE-) Prolog. Presented at the 1989 AERA conference, San Francisco, March 27-31 1989.

Merill, M.D. (1987). Prescriptions for an Authoring System. J.o. Computer-Based Instruction, Winter '87, 14-1, pp.1-10.

Merill, M.D., Li, Z. (1989). An Instructional Duign Expert System. J.o. Computer-Based Instruction, Summer 1989, 16-3, pp. 95-101.

Papert, S. (1980). Mindstorms: Children, Computers and Powerful Ideas, New York: Basic Books.

Sleeman, D. (1987)