

A COMPUTATIONAL APPROACH TO THE STUDY OF HUMAN SKILL ACQUISITION.

Jan J. Elshout
Laboratory for Experimental Psychology,
University of Amsterdam
Weesperplein 8
Amsterdam, The Netherlands

Bob J. Wielinga
Laboratory for Experimental Psychology
University of Amsterdam
Weesperplein 8
Amsterdam, The Netherlands

A general model for "learning by doing" is presented which accounts for a number of phenomena observed in protocols of human subjects solving a series of similar problems. The model is based on the conjecture that skill acquisition is a process of progressive development of problem directed knowledge structures which are to a large extent domain specific. A program is described for encoding protocols of human subjects in terms of the model.

When confronted with a series of similar problems to solve, humans usually improve their performance with experience. To investigate this "spontaneous" process of skill acquisition, we have chosen a methodology similar to that of [1]: taking protocols of human subjects solving a series of problems and constructing a program which is capable of coding these protocols, in terms of a computational model, possibly with the help of a human coder. The kind of problem we have chosen is the two waterjug problem: given two waterjugs with known capacities, a tap and a drain, produce a given quantity by filling, emptying and pouring the jugs. This type of problem is generally considered to be "semantically poor", but it has the advantage that the transition from novice to expert in the domain is a matter of only a few hours exercise. Moreover, analysis of protocols of novices shows that many "semantic problems" have to be solved before expert behaviour emerges [2].

Detailed analysis of some of the protocols by hand led to three major observations: 1) many processes apparent in the protocols can best be described in computational terms (eg. planning, debugging, global/local control, demons), 2) "learning by doing" is a complex interaction between a variety of problem solving processes, and is not easily described by a set of independent learning mechanisms and 3) learning manifests itself as a qualitative change in the problem solving process: things are not just done faster, they are done differently. The latter observation gives rise to the basic con-

jecture that underlies our research: improved performance is a manifestation of the progressive development of problem directed knowledge structures (schemata) which are to a large extent domain specific.

On the basis of these observations (and a number of more specific ones, see [2]), we have developed a general model of the skill acquisition process. In our model we distinguish a number of functional components (cf. [3]): ORIENTATE, SOLVE, REFLECT and REALIZE, each represented by a process. Processes may be simultaneously active and communicate via a common working memory of which the processes themselves are part. Most control decisions in the system are taken locally by specific processes, but in some cases (e.g. major errors or break throughs) focus of attention has to be shifted from one process to another. This more global control function is performed by a separate process: the "Research Director" [4]. The Research Director process has global knowledge about the ongoing activities, the distribution of resources, the current goals, a global measure of progress and a set of rules describing what to do when something unexpected happens. The basic control functions of the Research Director are: initiating activities, providing these activities with advice on the currently appropriate mode of operation (e.g. "sloppy", "reflective", "mechanic", "careful"), interpreting general comments or gripes from processes and acting accordingly if necessary.

When confronted with a new problem the Research Director will activate ORIENTATE. The main task of ORIENTATE is to construct (or retrieve from

memory) a representation of the problem: the problem conception [5]. In the problem conception information is stored about the specific problem parameters, possibly applicable plans, information about bugs that occurred previously etc. In the protocol fragment shown in Appendix 1, orientation is restricted to assimilating problem parameters (line 3). Other protocols show more elaborate orientation such as computing various relations between the numbers involved.

Given a problem conception which is well enough specified to attempt a solution of the problem, the Research Director will activate SOLVE. The performance of SOLVE is primarily dependent on the availability of problem directed schemata (plans, algorithms, rules). When an algorithm or a plan is available in the problem conception it will be executed. Planning (which is a sub-process of SOLVE) is often performed when the problem (as the subject sees it) allows some form of abstraction or when a new result is obtained. In the waterjug problem domain planning amounts to solving the problem in number space using arithmetic operations. Many subjects use plans (or algorithms) which are not based on abstractions of the operators, but on generalization of previous solutions. The subject who produced the example protocol, uses an algorithm developed while solving previous problems. This algorithm consists of two parts: an "Opening" (fill the smaller jug, pour it over, fill the larger jug up, (lines 4 to 11), and a "keep-plan" (create a new quantity by pouring and save it in the smaller jug; lines 12-15). When no plans or algorithms are available, SOLVE has to resort to search, which may be guided by domain specific rules such as "never fill up two jugs", "new numbers are winners".

The activities of SOLVE are monitored by the function REFLECT and its processes. REFLECT evaluates the rate of progress, checks the efficiency of the problem solving process (e.g. by detecting redundant steps; see lines 6-8 in the example), diagnoses errors in plans and algorithms (cf. HACKER [6]), suggests therapy for buggy plans, and tries to generate abstractions of existing schemata (plans, operations). The REALIZE component of the model has as explicit task to modify existing knowledge structures or to create new ones. REALIZE fixes bugs in plans according to the diagnosis and therapy suggested by REFLECT. REALIZE also builds plans from generalizations of previous solutions.

On the basis of the general model sketched above we have constructed a protocol encoding program: PANKAN. PANKAN consists of two basic components:

1) a collection of processes representing the functions in the model and 2) a component which relates the behaviour of the model to utterances in the protocol. The implementations of the functional components of the model differ in their degree of detail. ORIENTATE simply retrieves a problem conception schema from memory and fills the appropriate slots with the actual problem parameters. SOLVE accommodates several options: simple planning, execution of plans and algorithms, search. REFLECT currently performs the following functions: evaluation of obtained quantities according to criteria such as "newness", "known subgoal", etc.; demon triggered evaluation of "phantom numbers" (empty space in a partially filled jug), which causes the Research Director to initiate a new problem solving process; detection of redundant steps; diagnosis of errors due to "clobbered preconditions". REALIZE is currently able to implement the suggestions given by REFLECT.

Apart from these few exceptions, the current versions of REFLECT and REALIZE are not able to change knowledge structures or create new ones. The program has to be provided with the knowledge that is needed to account for the behaviour of subjects in various stages of the learning process.

The second component of PANKAN (ANALYSE) monitors the behaviour of the learning and problem solving processes and tries to relate this behaviour to the utterances in the protocol, by either interpreting the protocol on the basis of keywords or by asking questions to a human coder (see Appendix 2 for examples). When the behaviour of the model is not in accordance with the protocol, ANALYSE can take a variety of actions. If the model allows a choice of behaviours, ANALYSE will redirect the activity of the model. Another possibility is that there is a need for a specific schema which is not part of the database of the model. ANALYSE will try to find one in a library of schemata. A third action that ANALYSE can take is to shift the focus of attention (e.g. from planning to search). When all of these actions fail to bring model and protocol in agreement, the program fails to encode (part of) the protocol.

When provided with the appropriate knowledge structures, the current version of PANKAN is able to analyse many "simple" protocols, i.e. protocols reflecting more or less expert level, where no serious deviations of the normal execution of plan occur. Of the more interesting protocols i.e. where complex process interactions occur, the current program can only analyse a few. So far we have not been able to

analyse a complete series of 15 protocols. Problems that are hampering such an analysis include: the limited power of the system to predict and resolve confusions (e.g. confusing empty space in a jug with its content; errors due to neglect of preconditions), limited capability of creating and selecting the appropriate knowledge structures, problems with major shifts in attention and the corresponding control problems.

In spite of these difficulties we believe our effort to be a fruitful one. Using concepts and ideas from AI research, has considered enhanced our capabilities to analyse protocols in such detail that interesting learning phenomena become evident. The construction of a protocol analysis program has forced us to formalize both the framework on which our protocol analysis is built and specific analytic methods.

APPENDIX 1.

Analysis of a protocol fragment made by PANKAN. Jugs are 7 and 6, goal is 4. Only main activities are given.

```

1      4
1      now, the pile is getting thinner,
      (conversation because of new Datum)
      (Research director active)
      (approach is Reflective)
      (orientation)
2      but it is still going slowly...
      (conversation because of New Datum)
3      7 6 4 (problemconception is Pc7a1)
      (assimilation of new data)
      (Demon active)
      (solving)
      (Reflect active)
4      hum, fill this one
      (using Plan7a1)
      (using Openin1)
      (Fill of right jug)
5      6 ... (Newnumber 6)
      (current state is 0 6)
6      you don't have to transfer this one to
      the other side
      (Evaluate active)
      (hold all)
7      that doesn't make sense,
      (result of Evaluate is: redundancy)
8      hum, I should have filled the other one...
      (Therapist active)
      (Therap\l seen as a Therapy it has plan =
      Openin1
      replacement = FillA) suggested)
      (Realize active)
      (Opening changed)

```

```

9      anyway, 6 0
      (continue all)
      (Pour of right jug)
      (current state is 6 0)
10     you fill this one up
      (Fill of left jug)
11     7 0 (Newnumber 7)
      (current state is 7 0)
12     you pour it over
      (using Keep.pl)
      (Pour of left jug)
      (Newnumber 1)
      (current state is 1 6)
13     and you empty that one
      (Empty right jug)
14     1 0 (current state is 1 0)
15     and transfer it to the other side...
      (Keep of left jug)(Current state is 0 1)

```

APPENDIX 2

Questions asked to the coder and answers given (underlined) during analysis of the example protocol. Numbers indicate lines.

line 1: do you make conversation not related to the problem?

yes, 1 2

line 8: do you make an evaluation?

yes

do you criticize the current plan?

yes

line 11: do you finish the evaluation here?

yes

line 12: do you want to pour the right jug into the left one?

yes, 1 2

REFERENCES

- [1] Bhaskar, R. & Simon, H.A. "Problem solving in semantically rich domains: an example from engineering thermodynamics." *Cognitive Science*. 1:2 (1977) 193-215.
- [2] Elshout, J.J. & Wielinga, B.J. "A computational approach to the study of Human Skill Acquisition." University of Amsterdam, ICO.256, 1979.
- [3] Smith, R.G., Mitchell, T.M., Chestedi, R.A. & Buchanan, B.G. "A model for learning systems" In *Proc. IJCAI-77*. MIT, Cambridge, Mass., August, 1977, pp. 338-343.
- [4] Hardy, S. "Synthesis of LISP functions from examples." Ph.D. thesis, University of Essex, 1976.
- [5] de Groot, A.D. "Thought and choice in chess". The Hague: Mouton, 1965.
- [6] Sussman, G.J. "A computer model of skill acquisition", New York: American Elsevier, 1975.