The Language Shift : a mechanism for triggering metacognitive activities.

Pierre Dillenbourg
TECFA
Faculté de Psychologie et des Sciences de l'Education
University of Geneva (Switzerland)

Abstract

This chapter presents a metaphor for designing educational computing systems (ECSs) that progressively transfer to the learner an increasing amount of control in the problem solving process. The continuous variation of learner's control is segmented in a few levels. The transition from some level i to the next higher level i+1 results from the internalization of the concepts necessary to control the activities at level i. The use of reflection tools is proposed for supporting the internalization process. These reflection tools reify the control features of the learner's activities, i.e. they make concrete some abstract features of her behaviour. The next level is reached when the learner is able to use aspects refied at level i to interact with the system at level i+1. A same control concept is hence used firstly as a description language (by the system) at some level i and then as a command language (by the learner) at the level i+1. This language shift mechanism elevates the learner's level of control and her level of abstraction. It is described by analogy with an elevator that would move inside a pyramid. A floor of the pyramid corresponds to some control level. We use a formal notation to look inside the language shift mechanism and relate it to various psychological theories and current ECSs.

in P. WINNE and M. JONES (Eds). *Adaptive Learning Envirtonments (pp. 287-315)*. Berlin: Springer-Verlag

1. Introduction

Metacognition refers to "*understanding of knowledge, an understanding that can be reflected in either effective use or overt description of the knowledge in question*" (Brown, 1987). This definition covers the two main facets of metacognition : regulation of action and knowledge about knowledge. Metacognition is a very complex concept because it touches on many fundamental issues in cognitive psychology, such as learning, understanding, reading, consciousness, self-image etc.. This over extended meaning may lead to give the word "metacognition" the image of an empty concept.

The complexity and looseness of this concept does not question its importance (Schoenfeld, 1987). This complexity corresponds to a particular stage in research : the various mechanisms that exist under the "metacognition" label have still to be isolated and labelled (Brown, 1987). We do believe that metacognition research has the potential to improve actual classroom practice. Schoenfeld (1987), and Palincsar and Brown (1984) have demonstrated concrete and efficient forms of metacognitive training. Metacognitive research has "*reawakened an interest in the role of consciousness, or awareness, or understanding, in thinking and problem-solving*" (Campione, 1987). This not for scientific pleasure : it concerns thousands of children who are wasting their live at the back of some classrooms. Learning disabilities appear to be connected with poor metacognitive skills (Wong, 1985) and can be reduced by metacognitive development programs (Campione 1987).

What is the link between metacognition and design of an ECS? If metacognition is related to learning, it is still more closely related to self-instruction (Weinert and Kluwe,1987), and therefore to learning with an ECS. The debate about student control that opposes LOGO defenders with partisans of strong tutoring is typically a metacognitive issue. Moreover, metacognition is also related to motivation - through the self-image (Weinert and Kluwe, 1987) - and to the issue of transfer, two key issues in ECS evaluation.

The development of ECSs field is still obstructed by narrow-minded people that can only consider two possibilities : either you build a very directive teaching system that will be very efficient and centred on some specific content, either you leave the students playing around, roughly speaking learning nothing about vague cognitive skills. Metacognitive investigations goes beyond this stupid discrimination and opens the door on ECSs that acknowledge the context-dependency of knowledge and the importance of basic procedural skills.

This chapter is based on a continuous view of metacognition: there is no one single cognition level plus a control level, but instead an (infinite) tower of levels, each level controlling the hierarchically inferior level (Brown, 1987; Maes,1988). Recursion is the only way to implement this infinite tower within a finite device, the human brain. We hence discard the idea of having some metacognitive box that controls some other cognitive box. Metacognition is rather the ermergent property of our cognitive system when it processes information about itself. In this contribution, we analyse the metacognitive processes in terms of inductive, deductive and analogical processes that are not specific to metacognitive activities.

This chapter will focus on an important concept in metacognition : reflection. The word "reflection" has two meanings. The psychological reflection is the cognitive activity that leads to some consciousness of one's own knowledge. The physical reflection consists in returning to something its own image. This chapter investigates how the computer's physical reflection can support the learner's psychological reflection by presenting her some reified description of her behaviour (Brown,1985; Collins and Brown,1988). *Reification* means making concrete abstract aspects of behaviour, or, in Wenger's terms (1987), creating a written notation for a process. We won't consider the computer's reflection in the psychological sense, i.e. a fiel of research better known under the label "computational reflection" (Maes, 1988).

2. Framework for interaction design

It is our ambition to present a lecture that goes beyond the simple description of systems that care about metacognition or the enumeration of a few empirical rules. We have attempted to describe our approach in a formal way. The result of this attempt, although incomplete, constitutes a first step in the long-term development of *computational mathetics*, i.e. a computational formalisation of the learning process and of the ECS design process (Self, 1990). The specificity of our discipline is precisely the relationship between the learning process and the design process. So far, these processes have been separated : at one hand, we have some theories about learning, and at the other hand, we have some principles about how to design an ECS that promotes leaning. The latter is supposed to be consistent with the former, but they are still separated. The challenge is to integrate the later in the former : the ECS features have to be incorporated in the theory of "learning with an ECS". The quest for such an integrated theory is the major difference between research efforts and development work. Computational mathetics aims to be this unified theory, under some computational form. We have already attempted to develop a similar framework in the domain of student modelling approaches (Dillenbourg and Self, 1990).

This formal notation is not completed in the sense that, at the end, reasoning with the proposed notation is not simpler than reasoning on real problems. Theorems are not available for formal deductions. However, we believe it is still worth to present as a first step towards computational mathetics. We will firstly describe the notation and then the language shift mechanism with the associated pyramid metaphor. An example, theoretical foundations and relations with similar work will be described later on.

3. Terminology and Notation

This section introduces some minimal notation that will be used in this attempted framework. We restrict ourselves to a few concepts related to metacognition and learner-system interaction. We use uppercases to represent sets and lowercases for elements. Variables will be denoted _variable.

The atoms of the framework are actions. An *action* is any act performed by a agent through the interface : a sentence typed in by the learner, a window open by the system, a button pushed by the learner,... A action will be noted by a predicate :

```
action (_agent, _problem, _level, _time)
```

An *agent* is a cognitive system able to use an interface. In this framework, we restrict ourselves to two agents, the learner or student and the system or computer. This framework may be extended to multi-agents systems : for instance systems with several real learners or systems where computer plays the role of a collaborative learner (Dillenbourg and Self, to appear). Hence we have :

action (**learner**, _problem, _level, _time) or
action (**system**, _problem, _level, _time)

A problem is defined by its appartenance to a probem class. A *problem class* is a set of problems that have similar intrinsic features, i.e. the knowledge required to solve, the solution process, etc. A class may be subdivided into several subclasses. Within each *subclass*, problems have similar extrinsic features, i.e. a similar expression, within a similar context,... The solution of each problem requires activating knowledge specific to its subclass plus the knowledge shared by the all class. For instance, if the class is "fault diagnosis", we can have many subclasses such as "car engine diagnosis", "hard disk diagnosis", "water pump diagnosis",... The first subclass can again be subdivided into "Porshe engine diagnosis", "Fiat engine diagnosis", ... This distinction between problems classes and sub-classes will be later necessary for discussing the transfer issue. Each class can obviously be considered a subclass of some larger set of problems. It is considered as a class if it is the largest subclass among the pedagogical objectives. A problem class is noted **P** and its subclasses $\mathbf{P_i}$. A problem is associated with a number in subscript :

Example : `action (_agent,` $\mathbf{problem_1}$`, _level, _time)`

The "level" slot refers to the agent's reasoning. For the purpose of this paper, we define the agent's reasoning as a *hierarchy of control levels*. This hierarchy looks more like a pyramid than like a inverted tree (with roots in the sky) because each decision at one level does not necessarily map to a few at the inferior level. Sometimes, control may be more global, e.g. when chosing a main approach to some problem. An example of segmentation in control levels is presented in section 8. Levels will be noted by an integer that counts the levels from the bottom of the pyramid. It must be said that the concept of control level is broader that in the chunking theory, the internalization of some concept requiring some abstraction in addition to a simple compilation process.

Example : `action (_agent, _problem,` **`level-1`**`, _time)`

The *time* is here defined as the position of an action within a sequence of actions. We do not represent the length of intervals. Time is represented by an integer :

Example : `action (_agent, _problem, _level,` **`5`**`)`

For the simplicity of writing this complex predicate will sometimes we denoted simply by writing its time slot in subscript :

$$\mathrm{action}_5 = \mathrm{action}\ (\_agent,\ \_problem,\ \_level,\ \mathbf{5})$$

The *action space* is the set of actions from which some agent may select its actions. The action space is defined by a triplet :

```
Action-Space (agent, level, pedagogical-approach)
```

The set of actions available vary according the agent, the level and the pedagogical approach. The learner’s action space is determined by the system according to its current pedagogical approach. For instance, the action "consulting the dictionary" may be accepted at some stage of the interaction, and forbidden during the evaluation stage. The system’s action space is defined by the designer and the interaction. Determining the learner's action space is important for the purpose of this paper, since it represents the system's ability to tune the learner’s control of the system: roughly speaking, the complexity of student's control increases with the size of her action space. The choice of a pedagogical methods will not be discussed in this framework, but it could be introduced in relation to the action space concept.

An action space can be viewed as a language with a set of basic actions as vocabulary and with a few syntactical rules that govern sentence building. According to the *grain-size* of analysis, an action may be one word long (such as a key press or a click) or some more complex sentence (e.g. click on object A, drag it on top of object B and select the "connect" menu option.). We will specify later on different kinds of language.

A *behaviour* of an agent X is a subset of X's action space, i.e. a set of actions that have the same agent-slot X and have been performed for a same problem :

```
Behaviour (agent_x, problem_i) =

      {    action (agent_x, problem_i, _level, 1),
           action (agent_x, problem_i, _level, 2),
           action (agent_x, problem_i, _level, 3),
           action (agent_x, problem_i, _level, 4),
           ... }
```

The *interaction* is a sequence of actions performed by agents through the interface. Each action belong to the current action space of each agent. We only consider student-computer interactions but the notation leaves the gate open for other classes of interactions, e.g.interaction between two learners. The interaction between two agents X and Y for some problem Z is denoted

```
Interaction (agent-X, agent-Y, problem-Z)
```

Example :

```
Interaction (learner, system, problem_i) =

    { action (learner, problem_i, _level, 1),
      action (system, problem_i, _level, 2),
      action (learner, problem_i, _level, 3),
      action (system, problem_i, _level, 4),
      ... }
```

Finally, the representation that some agent has of some object is denoted

```
representation (agent, object)
```

4. The "lift in the pyramid" metaphor

Figure 1 shows the metaphor we will use in this chapter : a lift within a pyramid. The pyramid represents the hierarchy of controls described in the previous section and each floor corresponds to some level of control. The ground floor is called floor 1. The lift cabin represents the activity of the learner. The lift position within the pyramid represents how the reasoning is shared between the learner and the system. The area above the lift is the part of the reasoning process which is regulated by the system (the tutor component). In the area below the lift, one finds the functionnalities offered by the system to solve problems, e.g. basic calculus tools. The size of the lift represents the number of decision levels assumed by the learner, i.e. the cognitive load on the learner’s shoulders.

It is always possible to imagine a level beneath or above the level considered.The definition of a ground floor is contex-dependent and arbitrary. The ground floor of an ECS that teaches equations solving may for instance be the top floor of another system on elementary calculus.

The *command language* defines the learner's action space : the words of the language determine the available actions and the language syntax sets the possible combinations of actions. At the outset, this command language defines the baseline of the pyramid, but this paper will show that we can elevate the command language to higher floors. The *description language* is a set of words and syntactical rules used by the computer to present a representation of learner's action. A representation language is not neutral but emphasizes some features. The description language is supposed to be some graphical language, composed of objects and relation between objects. The objects represent the learner's actions. The relations between objects hence reify the relations between actions, i.e. the control performed at the lower floor.

The system action space is composed of direct answers to learner's commands plus sentences from description language. For example, in Algebraland (Brown, 1985), if the learner use the command "distribute" on the equation 3(x -5) = 7 , then the system will have two actions : returning the new equation 3x - (3*5)= 7 and drawing a line that reifies the transfornation link between these states.

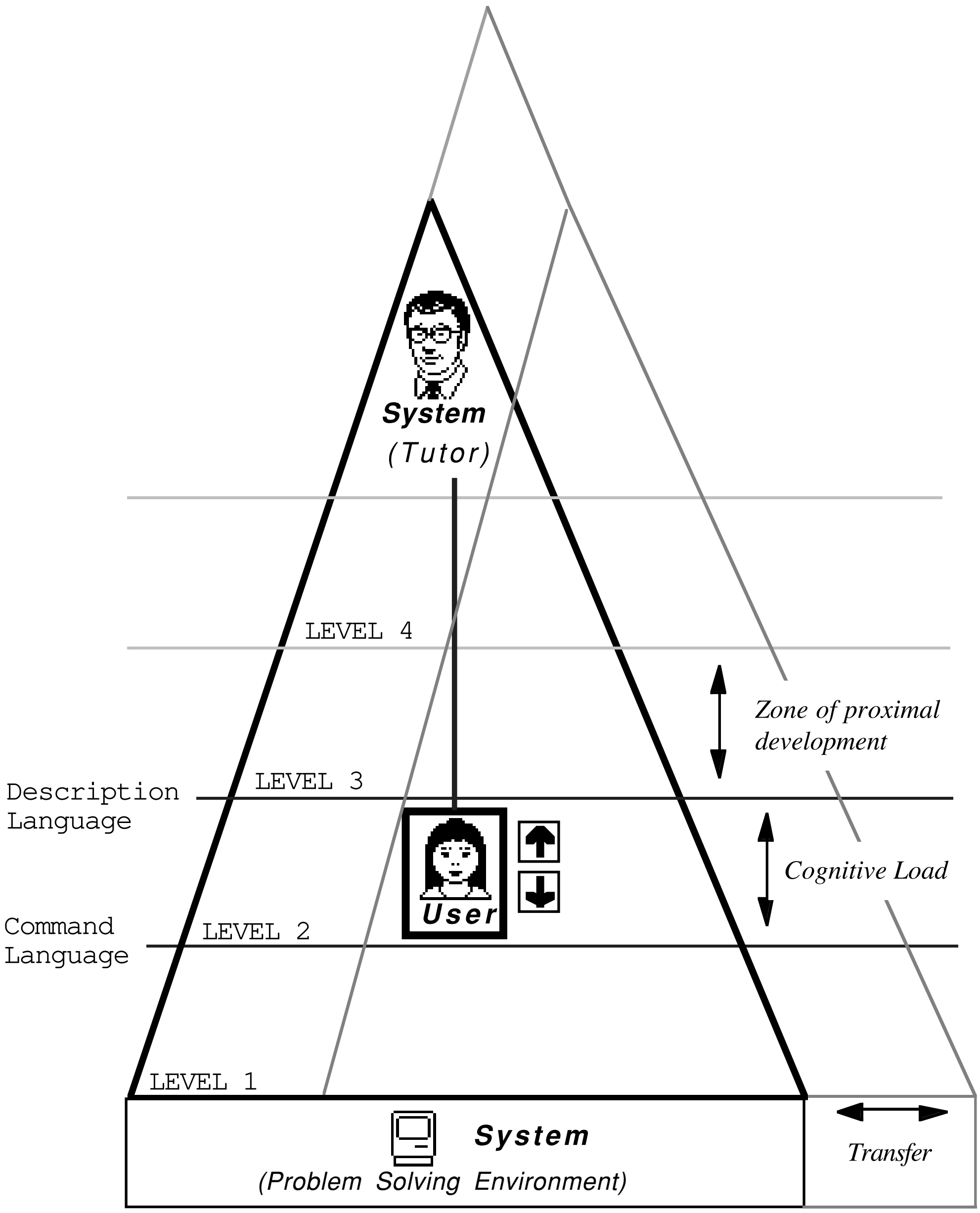

Figure 1 : The "Lift in the Pyramid" metaphor

In our notation, the action spaces can hence be defined as follow :

$$(\text{Action-space}(\text{learner}, \text{level}_N, \pi)) = \{\text{action}(\text{learner},\_,\text{level}_N,\_) \mid \text{action}(\text{learner},\_,\text{level}_N,\_) \in \text{command-language}\}$$

$$(\text{Action-space}(\text{system}, \text{level-N}, \pi)) = \{\text{action}(\text{learner},\_,\text{level}_N,\_) \mid \text{action}(\text{learner},\_,\text{level}_N,\_) \Rightarrow \text{action}(\text{system},\_,\_,\_)\} \cup \{\text{action}(\text{system},\_,\text{level}_{N+1},\_) \mid \text{action}(\text{system},\_,\text{level}_{N+1},\_) \in \text{description-language}\}$$

5. The ascending mechanisms

Teachers aim that learners perform better in higher levels of control and abstraction. Metaphorically, the goal is that the lift reaches the highest floors. It must be said that, for many tasks, it is probably difficult to imagine more than three or four levels and that the relationship between two levels is probably not identical all through the pyramid: "*Nature may not always be theoretically aesthetic*" (Wenger, 1987). We examine the mechanisms of shifting to the next level (up), which can recursively lead to this goal.

5.1. The structure of experience.

Lets imagine that the learner has to solve a problem 1, at the first floor of the pyramid. His behaviour is :

$$\begin{aligned}\text{Behaviour}(\text{learner}, \text{problem}_1) = \\ \{\ & \text{action}(\text{learner}, \text{problem}_1, \text{level}_1, 1), \\ & \text{action}(\text{learner}, \text{problem}_1, \text{level}_1, 2), \\ & \text{action}(\text{learner}, \text{problem}_1, \text{level}_1, 3), \\ & \text{action}(\text{learner}, \text{problem}_1, \text{level}_1, 4), \\ & \ldots\ \}\end{aligned}$$

where actions are sentences of the command language

The microworld may give an answer to each learner's action. If the action is simply typing text into a window, then the answer is neutrally bringing characters on the screen. But if the action consists in using some facility offered by the microworld, then the system response may be the result of some longer computation. The action of the computer may also be to do nothing.

$$\begin{aligned}\text{Interaction}(\text{learner}, \text{system}, \text{problem}_1) = \\ \{\ & \text{action}(\text{learner}, \text{problem}_1, \text{level}_1, 1), \\ & \text{action}(\text{system}, \text{problem}_1, \text{level}_1, 2), \\ & \text{action}(\text{learner}, \text{problem}_1, \text{level}_1, 3) \\ & \text{action}(\text{system}, \text{problem}_1, \text{level}_1, 4), \\ & \text{action}(\text{learner}, \text{problem}_1, \text{level}_1, 5), \\ & \ldots\ \}\end{aligned}$$

For reifying the learner's behaviour, the system has supplementary actions, expressed with the description language. These actions describe the relation between the last **n** learner's actions, where n varies according to the problem, the level and the time. In the simplest case, the description language reifies the relation between the two last learner's actions and the intermediate system's action. The reifications actions belong indeed to a higher level, here level 2, since they represent the control feature of the lower level. Subsequently, an interaction with systematic reification of the control aspects of learner's behaviour becomes something like :

```
Interaction (learner, system, problem1) =
      {     action (learner, problem1, level1, 1),
            action (system, problem1, level1, 2),
            action (learner, problem1, level1, 3),
            action (system, problem1, level2, 4),
            action (system, problem1, level1, 5),
            action (learner, problem1, level-1, 6),
            action (system, problem1, level2, 7),
            ... }
```

where the level-2 actions are a sentence from the description language and, in this example :

```
action (system, problem1, level2, 4) =
Representation (system, relation (action1,action2,action3))
```

and

```
action (system, problem1, level2, 7) =
Representation (system, relation (action3,action5,action6))
```

The description language sentences constitute the punctuation that enables the learner to split the interaction sequence into *interaction segments.* An interaction segment is a subset of the interaction described by a description language sentence:

```
Interaction-Segment (learner, system, problemi, i) =
 {    ...
      action (learner, problemi, levelN-1, M-3),
      action (system, problemi, levelN-1, M-2),
      action (learner, problemi, levelN-1, M-1),
      action (system, problemi, levelN, M)       }
```

If we imagine that the problem is to add fractions, here is an example of interaction fragment :

```
Interaction-Segment (learner, system, "(2/3 + 4/9)", i) = {

action (learner, "(2/3 + 4/9)", level2, 5) =
clicks on fraction 2/3 and asks "mutl-den-by" with arg.3

action (system, "(2/3 + 4/9)", level2, 6) =
replaces 3 by 9

action (learner, "(2/3 + 4/9)", level2, 7) =
clicks on fraction 2/3 and asks "mutl-num-by" with arg.3

action (system, "(2/3 + 4/9)", level3, 8)} =
graphical representation of "make-equivalent-fraction"      }
```

This segment shold be preceded by another one where the learner determines the common denominator and followed by other ones where she performs the addition and eventual simplification. This example is at level 2 since the learner does not perform the aritmetic operations by itself, but asks the system to do it via the command language. At a higher level, this segment would become a part of a segment called "reduce-to-the-same-denominator".

The length of a interaction segment may vary. The system action between the two learner's actions ($\texttt{action}_{m-2}$) is necessary to understand the relationship between the learner's actions only if this system's actions brings new information necessary for the choice of the learner's second action. Otherwise, this action can be removed from the segment. This model further simplifies reality by the fact that the student's actions are considered to constitute a flat set. Things may be more complex (e.g `action3` resulting from `action1` instead of resulting from `action2`). It is clear that a model needs to make some simplifications to be useful, it is indeed these simplifications that create its heuristic power.

This interaction will be repeated for successive problems of P. After some time, the learner has observed a large set of interaction segments.The learner's experience is a collection of these segments, denoted E :

```
E = { Interaction-Segment(learner, system, problemi, n)
      Interaction-Segment(learner, system, problemi, m)
      Interaction-Segment(learner, system, problemj, p)
      Interaction-Segment(learner, system, problemk, q)
      ... }
```

5.2. The induction postulate

The interaction gives the learner the opportunity to observe the computer's representation of control features. The aim is that the learner can build her own representation as a result of the internalisation process :

```
(internalisation
     (Representation
          (system, relation (...,actionM-3,actionM-2,actionM-1))
⇒
(Representation (learner, relation (...,actionM-3,actionM-2,actionM-1))
```

This section and the next ones analyses how reflection can help this internalisation process.The role of reflection is to organize and structure experience. We base our work on the postulate that this process is mainly inductive. Campbell and Bickhard (1986) proposed the concept of environmental *interaction patterns* as a basis for cognitive development. In the current problem, interaction patterns are sets of similarities between segments. These patterns will be internalized as metacognitive concepts. But this induction process is very difficult since it concerns relational concepts (concepts describing relations between actions).

We will see how the computer can help this induction. This inductive process includes two stages : first clustering, i.e. grouping segments into classes, then pure induction, i.e. searching for similarities between objects within each cluster. In natural learning-concept-from-example situations, clustering and inducing are intertwined : making classes is guided by the common properties of the objects in the classes and dicovering these properties is oriented by some idea of what the concept could be. Educational settings make the learner's life easier by artificially separating these two stages : the tutor classifies objects into positive and negative examples of the concept to be acquired or sets up an environment in which this classification appears clearly to the learner. Then, in a third stage, the integration of the acquired concept can be improved by constraining the learner to use the description language as a new command language. These three stages are described in the next sections.

5.3. Clustering stage

In our case, clustering means regrouping segments that correspond to a similar relation between student's actions. The description language includes the information necessary to perform this task. Remember that elements of E_S (to be clustered) are of the form :

```
{     ...
      action (learner, problemi, levelN-1, M-3),
      action (system, problemi, levelN-1, M-2),
      action (learner, problemi, levelN-1, M-1),
      action (system, problemi, levelN, M)      }
```

According to our definition of the syntax of the description language, the last element of this segment is a representation of the relation between the previous elements :

```
action (system, problem1, levelN, M) =
Representation (system, relation (...,actionM-3,actionM-2,actionM-1))
```

This equation reduces the complexity of the clustering stage because induction is performed on a single element ($\mathtt{action}_m$) instead of on whole segments and also because this element has concrete (visual) features (provided that $\mathtt{action}_m$ belongs to a graphical description language).

If the description language must guide the clustering process, its syntax should be designed to ease this process. Let's imagine a description language composed of objects. Any pair of objects (o_1, o_2) can be connected according to four rules : up, down, left, right. Then the learner's experience may be partitionned as follow :

```
E = { Es.up, Es.down,Es.left,Es.right} where

Eup={Interaction-Segment (learner,system,_,n)| actionm = up(o1,o2)}
Edown={Interaction-Segment (learner,system,_,m)| actionm = down(o1,o2)}
Eleft={Interaction-Segment (learner,system,_,p)| actionp = left(o1,o2)}
Eright={Interaction-Segment (learner,system,_,q)| actionq = right(o1,o2)}
```

... or in short :

```
E={Erel.1,Erel.2,...|Erel.x={Interaction-Segment(learner,system,_,n)|actionm
=rel.X(o1,o2)}}
```

Given that the syntax of the description language has to guide the clustering process, the designer has to choose a syntax that is sufficient but simple. If the possible relations are numerous or complex or not easy to perceive, the clustering process will be more complex.

5.4. Inducing stage

The task of the learner is now to perform induction on each cluster of EI. One cluster is :

```
Erel.x={Interaction-Segment(learner,system,_,n)|actionm =rel.X(o1,o2)}
where
relx(o1,o2)= representation (system, relation(...,actionm-3,actionm-2,actionm-1))
```

Since machine learning has passed through its "glasnost", we know pure induction does not exist, because the role of background knowledge is very important (Dietterich,1986). We use in induction a lot of knowledge from the context, from the usefulness of the concept to be acquired, from our previous experiences, etc. The learner does not perform induction in a knowledge-free environment. An ECS is precisely a environment in which the amount of available knowledge can be partially controlled.

The semantics of the description language may contain -or not - this information : a concrete (visual) relation between objects which represents metaphorically the real relation between the learner's actions. If this metaphor is very obvious, the learner can deduce the common features that she is expected to find within the cluster. For instance, in figure 2a, it can be logically deduced that this pattern includes A $\Rightarrow$ B action sequences; while figure 2b describes a pattern where the relation between A and B is that they both result from a third action X.

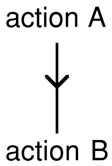

Figure 2.a

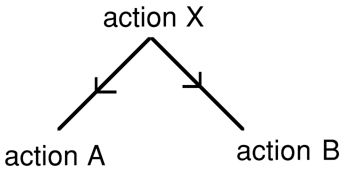

Figure 2.b

In other words, the design of the description language must take into account its analogical role in guiding induction. This analogy does not exempt the learner from induction : analogy is far from being an error-free process (Hall,1989). The learner has to check on the cluster objects and consider the context in order to give a particular meaning to relation described by the language

5.5. Integration : the language shift mechanism

When the learner has acquired a new concept about her own behavioural regulation, the system should constrain her to use these concepts for later problems. This can be done by using the description language as a new command language (and offering a new description language). This is what we called the "*language shift*". This mechanism is defined by the following equation :

$$\textbf{description language at level}_n = \textbf{command language at level}_{n+1}$$

After language shift, the learner has new tools for solving the problem, i.e. her solution process is changed. The new tools can be implemented by means such as new options in menus. Naming a new concept (as menu item) anchors this concept in learner's memory. In the pyramid metaphor, this language shift corresponds to the lift moving one level up, when one constrains the learner to describe her behaviour in terms of how it is regulated instead of by performing the actions directly .

A corollary of the above equation is that the system has to perform the actions the learner performed at the previous level :

$$\text{Action-Space (system,level}_N,\pi) = \text{Action-Space (learner,level}_{N+1},\pi)$$

The learner's cognitive effort necessary for reaching the next floor is somewhat lessened by the fact that the bottom boundary of the learner’s activities goes up simultaneously : she no longer has to perform the computation required by her actions, since the computer does it for her. She now has more cognitive resources available for concentrating on the control aspects. This is represented metaphorically on the pyramid by the fact that the lift remains the same size.The final goal of an ECS is obviously that the learner becomes able to perform alone all the control covered by the pyramid. However, by allocating the automatization of subskills to the system instead of to the learner, the focus on metacognitive aspects can be made sooner and more efficiently. An alternative approach to the computer-made automatization of subskills is the learner-made automatization of these subskills, i.e. the systematic practice of these skills, until they are compiled in quasi-automatic procedures.

6. Inside the language shift : action sub-spaces.

We have described the language shift as a all-or-none process. This process can be analyzed more finely if we stratify the learner's actions space into ordered subsets. We have empirically determined five sub-spaces according to the kind of activities that the learner is allowed to perform on the description of her behaviour :

observing	the learner just looks at the description
editing	the learner has some tools to edit the description
orienting	the learner use the description to select a problem state
composing	the learner creates the description by combining items
creating	the learner builds the description from scratch

If the interaction description is a tree, the levels of activities might be :

observing	looking at the three
editing	annotating nodes by justifications
orienting	selects a node for backtracking or ask for replay
composing	builds the tree from a bank of parts
creating	drawing the tree with lines and boxes

These action sub-spaces constitue way successive steps within the language shift mechanism: the observation level clearly corresponds to the description language, while the orientation activities begin to use the description as a new command language.

7. The role of problem variations

Transfer and lift ascension are viewed as two facets of a similar inductive mechanism, built around the description language.

In the pyramid metaphor, focusing on transfer means working on partially overlapping pyramids that correspond to new problems subclasses (or classes in case of far transfer). The union of two pyramids form a new one. The height of the new pyramid is proportional to its breadth. We try now to analyse this intuitive rule and to relate it with the mechanisms of lift elevation previously described.

In learning concepts from examples, differences between examples are very important because they enable one to eliminate features that are not relevant to the concept. For instance, for teaching the concept of square, the examples set must include non-horizontal squares otherwise learners will consider that a square may only have only horizontal and vertical sides.

Varying the subclass of problems aims to the same effect on interaction clusters as non-horizontal squares : to guarantee that features shared by examples are relevant components of the concept, i.e they insure eliminating non-relevant similarities.

Lets consider two segments :

$$\text{Interaction-Segment(learner, system, problem}_i\text{, n)}$$
$$\text{Interaction-Segment(learner, system, problem}_j\text{, m)}$$

Remember that the last element of each segment represents the relationship between the previous ones :

$$\text{action}_n = \text{representation (system, relation (...,action}_{n-3}\text{,action}_{n-2}\text{,action}_{n-1}\text{))}$$
$$\text{action}_m = \text{representation (system, relation (...,action}_{m-3}\text{,action}_{m-2}\text{,action}_{nm-1}\text{))}$$

The condition for transfer is that the learner detects some isomorphism the two segments $(\ldots, action_{n-3}, action_{n-2}, action_{n-1})$ and $(\ldots, action_{m-3}, action_{m-2}, action_{nm-1})$. This isomorphism is represented in the description language by the concrete similarities between the actions in the left term :

$$
\begin{aligned}
&relation\ (action_n\ ,\ action_m\ ) = \\
&reprentation \\
&\quad (system,\ relation \\
&\qquad ((\ldots,action_{n-3},action_{n-2},action_{n-1}), \\
&\qquad (\ldots,action_{m-3},action_{m-2},action_{nm-1}))
\end{aligned}
$$

This equation has important corollaries for the relation between the transfer process and the description language. If the description language does not include elements specific to some subclass P_i , we have :

$$action_n = action_m$$

In this case, the learner is directly informed by the system of the opportunity for transfer. If the representation used for the description language contains elements specific to some subclass but that these elements can be easily isolated from those that are shared, the transfer will be guided by searching for $(action_n \cap action_m)$ since

$$
\begin{aligned}
&(action_n \cap action_m) = \\
&reprentation \\
&\quad (system,(\ldots,action_{n-3},action_{n-2},action_{n-1}) \cap \\
&\qquad (\ldots,action_{m-3},action_{m-2},action_{nm-1}))
\end{aligned}
$$

Reciprocally, transfer mechanisms are important for the description language : the learner will demonstrate that she has integrated the description language when she will use it explicitly to perform transfer. This is important to detect the right time to start a language shift.

8. The MEMOLAB example

We are currently designing a system that applies the language shift mechanism to the complexitiy of a real world problem : "*nobody grows up by solving the tower of Hanoï problem !*" (Dillenbourg et al,1990a). This learning environment is called MEMOLAB. The pedagogical goals are that students acquire the fundamental concepts necessary to perform experimentation in psychology. These concepts are experienced by designing and running experiments on human memory, in an artificial lab. The four-floors pyramid is built around the concepts of experimentation. Peripheral concepts from statistics and human memory are attached to attached around thse core concepts. The four floors are :

1 : Building non-comparative experiments (single sample)
2 : Building simple comparative experiments (one factor, several samples)
3 : Building uni-dimensional experimental plans
4 : Building multi-dimensional experimental plans

Building an experiment means assembling concrete events on the lab workbench. An event is the association of some subjects with some task and a measurement device. A task is defined by an activity, a material and a few parameters. Building a plan means creating a non-intanciated single experiment in the workbench and generating the experiment by defining a n-dimensional table of factors (n being the number of independent variables).

At level one, the dumb beginner will sequence event-frames on the lab workbench. Each event-frame includes some part of the information that defined the event (subjects,task,...). The learners decide what information has to be displayed in each event frame. The elevation to the second floor implies that students understand that observing an effect means creating a variation between groups and that these inter-group variations must be more important that the intra-group variations. The description language will reify the differential nature of the experimental method. The system will regroup experiments that can be compared and redisplay the events in order to enlighten the differences.

The differenciation nature of the first floor description language defines the second floor command language. At the second floor, learners will use these differences to build experiments. For instance, they will describe the whole sequence of activities for a first sample. Then the second sample sequence will be built by editing the first sample sequence (e.g. augmenting the length of words that subjects of second group have to memorize), i.e. it will be defined by its difference with the first sequence.

The passage to the third floor implies a new discrimination by the learner between the concrete structure of the experiment (a sequence of events) and the variables (dependent and independent)that instanciate this structure. The description language will redisplay the experiments by separating the paradigm and its associated plan. The third floor command language will then offer two workbenchs, one similar to lower floors, to build experimental paradigms, and a new one, to define experimental plans. Finally, the concept of interaction between effects is necessary to reach the fourth floor. This concept will be suggested by comparing plans of the third floor, in the same way that experiments have been compared at the floor1-floor2 transition.

9. Theoretical Bridges

This framework can be read from several theoretical and practical perspectives. We review related work in psychology, in machine learning and in research on ECSs.

9.1. Psychologists visiting the pyramid.

9.1. 1. Metacognition and reflected abstraction (Piaget)

The mechanism of language shift is closely related to the piagetian concept of *reflected abstraction* (Piaget, 1971). Reflected abstraction is a process by which properties that are implicit at some level of knowledge can be abstracted and explicitly accessed at a higher level. The general relationship between metacognition and reflected abstraction is a complex issue of genetic psychology that goes beyond the scope of this paper (Blanchet, 1990). However, Campbell and Bickhard (1986) - although rather in conlfict with the piagetian structuralism -proposed an interesting analysis of this relationship that pertains to our approach. Let us remember that the induction of interaction patterns that is the basis of the language shift is a key notion of Campbell and Bickhard's theory.

These authors describe developmental stages in terms of going through an hierarchy of *knowing levels*, where each level "knows" the properties of the subordinated level. For Campbell and Bickhard, reflective abstraction is a mechanism of *knowing-level ascension*. Such an ascension through an hierarchy of levels matches our definition of the language shift mechanism. This parallelism is emphasized by the following paragraph (where the authors use the word "system" for referring to a cognitive system, i.e. an agent in our terminology) :

> "Suppose that a system could learn to create external indicators of various points in and aspects of its own internal process - as it was actually engaged in those processes. Then the indicators would manifest properties of the organization and functioning of those internal processes. Being external, the indicators would be available for examination by the system itself. From the indicators, the system could then abstract the properties of the processes that yielded those indicators. That is, using such indicators, a system could reflectively abstract its own properties. "
>
> (Campbell and Bickhard, 1986, p. 86)

If we stay in the piagetian lineage, the lift metaphor is very close to the neopiagetian theories developed by Robbie Case (1985). In this approach, the process of stage transition consists in integrating independent control structures into a more complex one, in a way very similar to our metaphor. In the MEMOLAB example, the objects of one floor are reorganised in order to reduce the task complexity. A group a single-group experiments (level 1) is re-unitarised into comparative exepriments (level 2). Sets of comparative experiments are re-unitarised as experimental-plans (level 3) and groups of plans are re-unitarised as multi-dimensional plans (level 4).The mechanism that trigger this re-unitarisation process is partly linked to the limits of working memory. Interestingly, we know the limits of working memory are related with the issue of awareness through the concept of focus of attention, more precisely the control function that actively maintains information in working memory (Jackendoff, 1987).

9.1.2. Internalization of regulation processes (Vygotsky)

According to Vygostsky (1978), the development of higher psychological processes results from the *internalisation* of inter-individual processes. Metacognition would also have social origins, self-regulation being viewed as the result of adult-child regulation (Wertsch, 1985). The vigotskian concept of "zone of proximal development" (ZPD) is present in our framework as the stage just above the lift position : this stage includes control activities that the learner is not able to perform alone but can do with the help of the computer:

ZPD = `Action-space (system+user,level`$_{N+1}$`,coaching)`

`if we have Action-space (learner,levelN,_)`

Vigostky outlined the linguistic nature of the internalisation mechanism when he declared that "*verbal thought is the transferral of speech to an internal level*" and that "*reflection is the transferral of argumentation to an internal level*"(Vigotsky,1981, quoted by Wertsch, 1985). This linguistic dimension is important for our purpose because it relates the type of user-computer interaction to our interest for metacognitive issues. Models of mental activities are indeed generally identified through verbs such as to think, to know, to ignore,... (Wellman, 1985). These linguistic aspects of internalization have been analysed by Wertsch (1985). He suggested that some linguistic mechanisms allow the performance of intra-psychological processes at some stage of the inter-psychological level. Wertsch observed mothers monitoring their children solving a puzzle. He found that mothers change their referential perspective according to the child's skills. When identifying a piece of the puzzle, beginner's mothers refer to piece attributes that both agents can observe (e.g. "a green piece"), while later, mothers tend to refer to the strategy used (e.g. "a piece with another colour"). According to Wertsch, *strategy-based* referential means that "an understanding of the strategy was required to interpret a referring expression appropriately". In the pyramid metaphor, the role of the description language corresponds to the mother's discourse, i.e. it describes the learner's behaviour in terms of the underlying strategy (Dillenbourg, to appear).

9.1.3. Transfer and metacognition

Several researchers have analysed the relationship between reflection and transfer. Rozin (quoted by Brown,1987) related reflective access to knowledge (the key to metacognition) and multiple access of knowledge (the key to transfer). This link supposes that "*accessing is connecting previously isolated components, or mental faculties, and passing information between them.*" (Campbell and Bickhard, 1986,p. 89). Ann Brown (1987) reports research on transfer between isomorphic problems by three- and four-year-old children. These children exhibited a greater transfer when, after each problem, they were required to describe their solution to "Kermit the frog" so that he could reproduce it. Multiplicity and reflection hence become central issues in the design of ITS. The difference between Brown's experiments and our approach is that we explore the reflection-transfer link in the opposite direction : how transfer can be used to promote metacognition.

The framework may also be related to Anderson theories of transfer within the ACT* framework (Singley and Anderson, 1989). Each segment of the learner's experience can be repressed as a production rule :

$$
\{ \quad \ldots \\
\text{action (learner, problem}_i\text{, level}_{N-1}\text{, M-3),} \\
\text{action (system, problem}_i\text{, level}_{N-1}\text{, M-2),} \\
\text{action (learner, problem}_i\text{,level}_{N-1}\text{, M-1),} \\
\text{action (system, problem}_i\text{, } \mathbf{level_N} \text{ , M)} \quad \}
$$

correspond to the production rule :

$$
\text{IF ...} \\
\text{AND action (learner, problem}_i\text{,level}_{N-1}\text{, M-3)} \\
\text{AND action (system, problem}_i\text{, level}_{N-1}\text{, M-2)} \\
\text{THEN action (learner, problem}_i\text{, level}_{N-1}\text{, M-1)}
$$

And this rule is represented by :

$$\text{action (system, problem}_i\text{, } \mathbf{level_N}\text{, M)}$$

The control aspects are compiled in these rules. At the opposite, the raison d'être of action (system, problem$_i$, **level$_N$**, M) is precisely to give some access to these underlying control features. The lift ascension in other words corresponds to a process that inverts Anderson's compilation process. The student task would be rather to extract the hidden metaknowledge in a similar way to a knowledge engineer (Clancey, 1988) attempting to build a deep expert system (Steels, 1990). This fundamental difference of approach implies another measure of transfer. Singley and Anderson estimate the difficulty of transfer as proportional to the overlapping between the production sets corresponding to the source and target problems. This overlapping is quantified in terms of number of identical production (with some weighting). In our approach, the overlapping includes not only identical segments but also segments sharing a same representation, i.e. a same abstraction. This view is also supported by computational applications in the so-called *abstraction-based analogy* field (Greiner, 1988; Shinn, 1988).

9.2. Computers scientists looking at the pyramid

With the last subsection, we touched on the relation between the pyramid and production systems. Several machine learning systems attempt to learn from some representation of experience called "problem solving path" (Langley, 1983). It is a trace of the operators applied during the solution process, associated with the state of the problem and some measure of the operator efficiency. *Learning from solving path* has already found applications in the design of ECSs such as ACM (Langley and Olhsson, 1984) or PROTO-TEG (Dillenbourg, 1989). Interestingly, Klahr (1984) relates this kind of machine learning with the concept of reflected abstraction. By time line, Klahr refers to something similar to the solution path or what J.S. Brown (1985) called "audit trail" :

> "The time line and its associated analyzers can be viewed as an attempt to further specify Piaget's appealing but mysterious notion of "reflective abstraction"; it provides a means for the system to ruminate about the efficacy of its own processing episodes."
>
> (Klahr, 1984, p. 117)

These systems learn by searching for conditions that improve the efficiency of operators, i.e. they aim to improve control skills involved in operator selection. For that, they attempt to induce the features that characterise states where the operator X has been efficient and discriminates these states from those where the same operator has failed. The key of this inductive process is the *generalisation hierarchy*, i.e. a graph that informs the system that two different features can be induced as instances of a more general concept. This generalisation hierarchy is often encompassed in the language used to describe the instances and the concepts. Researchers agree on the crucial role of this description language in inductive learning (Kodratoff,1986; Utgoff,1986; Banerji, 1989). This language determines the concept search space by selecting the object features that will be taken into account and how features can be generalised.

The role of the description language in the pyramid corresponds the role of the description language in machine learning. The set of instances is the sequence of command language sentences, i.e. the first three elements of each triplet in E_S. The description of these instances is provided by the description language, i.e. the last element of each segment.

The semantics of the description language provides the deductive component necessary for induction. For many years, machine learning theorists mistakenly believed that learning could be purely inductive and neglected the deductive aspects involved (e.g. the description language role). Later the deductive aspects were overestimated by the explanation-based learning stream (Dejong and Mooney, 1986; Mitchell et al, 1986). Today, the complementarity of inductive, deductive and analogical sources of knowledge is well accepted. According to their relative weight, these sources of knowledge paint the learning process is a more-inductive colour or more-deductive colour. Implementations differ on the scale of the task distribution between knowledge sources : Devi (1990) proposed to combine inductive and deductive main phases while Dillenbourg and Self (to appear) view learning as a complex and built-on-the-fly sequence of small inductive, deductive and analogical steps. This *microscopic* view of learning fits with the learner task in her lift : the meaning of some `action (system, problem`$_i$`,` **level**$_N$`, M)` can help to find similarities between interaction patterns or inversely, interaction patterns can help to understand the meaning of representational aspects. Both mechanisms occur probably in parallel or iteratively.

9.3. ECSs around the pyramid

In the field of ECSs, several pieces of work may be related to the pyramid lift metaphor. Two ECSs are often quoted when speaking about reflection : ALGEBRALAND (Brown, 1985) and the GEOMETRY tutor (Anderson et al, 1985). ALGEBRALAND may be described within our framework as a lift locked at the second floor. The users have to solve linear equations. They have a command language at their disposal that performs the ground floor computations : add-to-both-side, distribute, expand, ... The description language is a inverted tree whose nodes are the intermediate transformations of the equation. Branches correspond to applications of an operator. This representation reifies the heuristic aspects of a solving process : the multiple solution paths, the backtracking steps, etc.

To extend ALGEBRALAND along the pyramid metaphor, we should add a ground floor where the learner types in some equation (command language = set of algebraïc symbols). Learners have access in ALGEBRALAND to some editing activities (such as annotating the tree) that consitute the basis for adding a third floor to this system. At this third floor, the command language would for instance act directly on the description tree with commands such as "select the node with the fewest occurrence of x". Such a command would compel the learner to make its strategy more explicit.

This approach of ECS is based on fundamental issues that distinguish it from the first generation of CBT material : designers pay more attention to the process of learning than to its products (Brown,1985).The holy law of immediate feedback, inherited from the behaviourist grounds of CBT, is called into question in the interest of giving the learner the time of inspect its behavioural trace (Foss, 1987; Wenger, 1987). The pyramid metaphor emphasizes the complementarity of immediate and delayed feedback : immediate feedback is necessary to install low-level skills that need to be automated, while delayed feedback is necessary when the focus is the learner's consciousness of some features of her own behaviour.

The framework is also very close to the work of Derry and her colleagues on the TAPS II system (Derry, 1990). In TAPS II, the students concentrate on the representation of the word problem to solve instead as on the representation of the problem solving strategy. As the author points out, these representations are intertwined :

> "...a person's conceptualization of a problem may actually emerge as a by-product of the problem-solving strategy itself. Simultaneously, the way in which a problem is conceptualized will suggest a strategic ordering of problem-solving operations that can be applied to solve the problem. It is precisely this sort of reciprocal, dynamic interaction between representation and operation that must be supported if computers are to serve as conceptualizing tools in instruction."
> (Derry,1990,p.4)

Learners are invited to build a schema of the problem solution by selecting general schemas into a bank and instanciating them with the problem data ("composition" level of activities). TAPSII occupies the three first floors of the pyramid : simple arithmetic operations may be performed by the system at the ground floor, the assembled schemas are used as a description language of the second floor but also as a command language of the third floor. This results from the fact that the learner has to build herself the representation of her activity. Some self-monitoring strategies are available at the top of the third floor for guiding the student but do not form a new description language.

The major difference between the "lift in the pyramid" metaphor and the systems described here is that it concentrates on the *dynamic* aspect of a system. This system can be thought as a sequence of microworlds. The learner's metacognitive development (for a particlar task) parallels her road through these increasingly abstract worlds. Let's note that this sequence may be nested within a sequence of increasingly complex tasks (e.g. considering the number of parameters affecting a problem difficulty) or within a larger-scale sequence that cover various parts of the curriculum.

The progressivity defined by the pyramid metaphor instanciates the apprencticeship paradigm (Wenger, 1987) : the tutor assumes the part of control that the learner cannot yet perform and progressively transfer control to the learner. This approach has got a growing popularity both in education - in the vygostskian line - and in machine learning (Wilkins, 1998)

According to Cumming and Self (1989), one major challenge of ECSs is to decouple the *task level* and the *discussion level*. At the task level, the student works to solve the particular problem or task proposed by the system. At the discussion level, she reflects on the task level and interacts with the system about some aspects of this reflection (e.g. her plans, her failures, ...). The language shift mecanism recursively exploits this dicrimination at each level of the pyramid.

Let's for instance consider the concept of knowledge negociation system (Moyse, 1989), created by opposition to the paradigm of knowledge communication that dominated our field during two decades (Wenger,1987). In our understanding, the terms "knowledge negotiation" refers to some relationship between the task and the discussion level: the multiplicity of equally valid viewpoints at the task level requires the existence of some discussion level where explicit negotiation or articulation of viewpoints can occur. This explicit negociation movement is attested within the literature by significative shift of focus from "tutorial discourse" to "tutorial dialogue" (Petrie-Brown, 1990; Baker, 1989).

Finally, we believe that the pyramid is consistent with the idea of *constructorium*, i.e. the conception of the computer as a lab where learners observe the construction of their own knowledge.(Dillenbourg, to appear). After many years of passive schooling, students naturally acquire a passive model of learning. Learning comes to mean listening a teacher, reading a textbook or repeating a definition. ECSs would gain an innovative power by removing this baneful image, by reflecting to the learner a better image of its learning activities. This image should emphasize that learning is an active process of constructing knowledge.

10. Conclusions.

ECSs can advantageously offer tools for reflecting to the learner some abstract aspects of her behaviour. However educationalist have for long be confronted to the issue that presenting some information to the learner does not mean that this information will be processed and assimilated. Assimilation results of transforming this information during some activities. The fact that the computer enables such activities does not imply that learners do actually perform these activities ! The idea behind the language shift is to compel the user to perform such activities : the assimilation of the concepts of the description language is the premisse of their use as a new command language.

The detailed implementation of the language shift is beyond the scope of this paper. It raises problems that have been with us for many years : the psychological validity of the description language (and all the related student modelling issues) and who decides the language shift. Our current work on MEMOLAB (Dillenbourg et al, 1990c) aims to bring answers to these questions and to confront the consistency of the pyramid and language shift ideas to the complexity of real world problems. It is clear however that by the pyramid metaphor does not describe a complete learning environment, but only its core structure. Around the pyramid, one needs to implement complementary tools that stimulate the necessary internalization process and introduce the peripheral concepts. In MEMOLAB for instance, these tools will constitute a complex learning society : soft coaching agents, statistical tools, specialised counselors, reflection tools and an encyclopedia of human memory (written in some hypertext form). If the scientific interest of a learning is related to the aesthetic of its design priciples, its pedagogical power is related to the richness of multiplicity of experiences.

Acknowledgements.

This research is partially funded by the Swiss NRP23 research program on AI and robotics. Thanks to A. Brandeis, P. Mendlsohn, D. Schneider, J. Self and all the workshop participants for their comments on previous drafts.

References

ANDERSON J.R., BOYLE C.F. and YOST G. (1985) *The Geometry Tutor.* Proceedings of the Ninth International Joint Conference on Artificial Intelligence. Los Angeles. Vol.1

BAKER M.. (1989) *A model for Tutorial Dialogues based on Critical Argument.* Proceedings of the 4th AI and Education Conference. IOS. Amsterdam.pp. 2-8.

BANERJI (1989) Solving the linguistic problems in learning. *Fundamenta Informatica XII*, 51-78.

BLANCHET A. (1990) *Abstraction réfléchissante et Metacognition.* Paper presented at the "Archives Jean Piaget", Geneve,23/04/90.

BROWN A. (1987) Metacognition, Executive Control, Self-Regulation and Other More Mysterious Mechanisms. in F.E. Weinert and R.H. Kluwe (Eds) *Metacognition, Motivation and Understanding.* Lawrence Erlbaum. Hillsdale, NJ,pp. 65-115.

BROWN J.S. (1985) Process versus Product : A Perspective on Tools for Communal and Informal Electronic Learning, *Journal of Computing Research*, Vol. 1(2).

CAMPBELL R.L. and BICKHARD M.H. (1986) *Knowing Levels and Developmental Stages*. Karger. Basel.

CAMPIONE J.C. (1987) Metacognitive Components of Instructional Research with Problem Learners. in in F.E. Weinert and R.H. Kluwe (Eds) *Metacognition, Motivation and Understanding*. Lawrence Erlbaum. Hillsdale, NJ,pp. 117-140.

CASE R. *Intellectual development, birth to adulthood.* New York. Academic Press.

CLANCEY W.J. (1988) The Knowledge Engineer as Student : Metacognitive Bases for Asking Good Questions. in H. Mandl and A. Lesgold (Eds) *Learning issues for Intelligent Tutoring Systems*. Springer-Verlag. New-York, pp. 80-113.

COLLINS and BROWN J.S. (1988) The computer as a tool for learning through reflection. in H. Mandl and A. Lesgold (Eds) *Learning issues for Intelligent Tutoring Systems.* Springer-Verlag. New-York, pp. 1-18.

CUMMING G. and SELF J. (1989) *Collaborative Intelligent Educational System.* Proceedings of the 4th AI and Education Conference. IOS. Amsterdam.pp. 73-80.

DEJONG G. & MOONEY R. (1986) Explanation-Based Learning : An Alternative View., *Machine Learning,* (1),145-176

DEVI R. (1989) *Machine learning and Tutoring Systems.* Open University, CITE Report 61. UK.

DILLENBOURG P. (1989) Designing a self-improving tutor : PROTO-TEG. *Instructional Science*, 18, 193-216.

DILLENBOURG P. (to appear) The computer as a contructorium : tools for observing one's own learning. in R. MOYSE and M. Elsom-Cook (Eds) *Knowledge Negociation*. Paul Chapman.London.

DILLENBOURG P. and SELF J.A. (1990) *A framework for learner modelling*. TECFA Document 90-5, University of Geneva.

DILLENBOURG and SELF J.A: (to appear) Designing human-computer collaborative learning. in C.E. O'Malley (Ed), *Computer-Supported Collaborative Learning.* Wiley & Suns

DILLENBOURG P., MENDELSOHN P. and SCHNEIDER D. (1990a) *The Geneva Manifesto of Intelligent Learning Environments*. Document TECFA 90-3. Université de Genève.

DILLENBOURG P., HILARIO M., MENDELSOHN P. and SCHNEIDER D. (1990b) *Training Transfer : A Bridge between the theory-oriented and product-oriented approaches to ITS design*. Document TECFA 90-3. Université de Genève.

DILLENBOURG P., HILARIO M., MENDELSOHN P. and SCHNEIDER D. (1990c) *The MEMOLAB Project* Document TECFA 90-4. Université de Genève.

DIETTERICH T.G. (1986) Learning at the knowledge level. in *Machine Learning*, Vol.1, 3,pp.287-316.

FOSS C. (1987) *Acquisition of error management skills.* Paper presented at Third International Conference on Artificial Intelligence and Education. Pittsburgh, pp. 27.

GREINER R.(1988) Abstraction-Based Analogical Inference in D. H. Helman (Ed) *Analogical Reasoning.* Kluwer Academic Publishers. Doordrecht. The Netherlands. pp.147-170.

JACKENDOFF R.S. (1987) *Consciousness and the computational mind.* MIT Press, Cambridge, Massachussets.

MOYSE R. (1989) *Knowledge Negociation*. Working Paper. IET, Open University. UK

HALL R.P. (1989) Computational Approaches to Analogical Reasoning : A Comparative Analysis. *Artificial Intelligence*, 39,pp.39-120.

KLAHR D. (1984) Transition Processes in Quantitative Development. in R.J. Sternberg (Ed) *Mechanisms of Cognitive Development*. Freeman & Company. New York

KODRATOFF Y and GANASCIA J.G. (1986) Improving the generalisation step in learning. R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds) *Machine Learning*. Vol II, Morgan Kaufman, Los Altos,pp.215-244

LANGLEY P. (1983) Learning search strategies through discrimination. *International Journal of Man-Machine Studies,* 18, 513-541.

LANGLEY and OHLSSON (1984) *Automated cognitive modelling*, Proceedings of National Conference on Artificial Intelligence. Austin, Texas, pp 193-197.

LOCHHEAD J. (1985) Teaching Analytic Reasoning Skills Through Pair Problem Solving. in J.W. Segal, S.F Chipman and R. Glaser *Thinking and learning skills*. Volume 1 : Relating instruction to research. Laurence Erlbaum, Hillsdale, NJ,pp. 109-131.

MAES O. (1988) Issues in Computational Reflection in P.Maes and D. Nardi (Eds) *Meta-level Architectures and Reflection*. Elsevier. Amsterdam.pp.21-36.

MICHALSKI R.S: (1986) Understanding the nature of learning : Issues and Research directions in R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds) *Machine Learning*. Vol II, pp.3 -26, Morgan Kaufman, Los Altos.

MITCHELL, T.M., KELLER, R.M. & KEDAR-CABELLI S.T. (1986) Explanation-Based Generalization : A Unifying View., *Machine Learning*, (1), 47-80.

PALINCSAR A.S. and BROWN A.L. (1984) Reciprocal Teaching of Comprehension-Fostering and Comprehension-Monitoring Activities. *Cognition and Instruction,* vol.1, nº2, pp. 117-175.

PERTRIE-BROWN A.M. (1990) Discourse and dialogue : concepts in Intelligent Tutoring Interactions. *Journal of Artificial Intelligence in Education*, Vol.1, (2), pp. 21-30.

PIAGET J. (1971) *Biology and Knowledge*. The University of Chicago Press. Chicago.

REIMANN P. (1990) *Problem Solving Models of Scientific Discovery Learning Process*. Peter Lang. Frankfurt.

SCHOENFELD A.H. (1987) What's all the fuss about metacognition? in A.H. Schoenfeld (Ed) *Cognitive Science and mathematics education*. Laurence Erlbaum Associates. Hillsdale. NJ., pp. 189-215.

SELF J.A. (1990) *An introduction to computational mathetics.* Working Paper. Computing Department, University of Lancaster.

SHINN H.S. (1988) Abstractional Analogy : A Model of Analogical Reasoning. Proceedings of a Workshop on Case-Based Reasoning. Florida.pp.370-387.

SINGLEY M.K. and ANDERSON J.R. (1989) *The Transfer of Cognitive Skills.* Harvard University Press. Cambridge, Massachussets.

STEELS L (1990) Components of Expertise. *AI Magazine*, vol.11, 2,pp. 28-49.

UTGOFF P.E.(1986) Shift of bias for Inductive Concept Learning. R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds) *Machine Learning*. Vol II, Morgan Kaufman, Los Altos, pp. 107-148.

VYGOTSKY L.S. (written in the 30', but edited by M. Cole, V. John-Steiner, S. Scribner and E. Souberman, in 78), *Mind in Society*. The Development of Higher Psychological Processes. Harvard University Press. Cambridge, Massachussets.

WELLMAN H.M. (1985) The Child's Theory of Mind : The Development of Conceptions of Cognition. in S.R. YUSSEN (Ed). *The growth of reflection in Children. Academic Press*. Madison, Wisconsin.pp. 169-206.

WERTSCH J.V. (1985) Adult-Child Interaction as a Source of Self-Regulation in S.R Yusen (Ed).*The growth of reflection in Children. Academic Press*. Madison, Wisconsin.pp. 69-97.

WENGER E. (1987) *Artificial Intelligence and Tutoring Systems*. Computational and Cognitive Approaches to the Communication of Knowledge. Morgan Kaufman, Los Altos, California.

WILKINS D.C. (1988) Apprenticeship learning techniques for Knowledge Based Systems. Doctoral Dissertaion. Report STAN-CS-88-142. Stanford University, CA.

WONG B.Y.L (1985) Metacognition and Learning Disabilities. in D.L. Forrest-Presley, G.E. MacKinnon and T.G. Waller Metacognition, *Cognition and Human Performance, Academic Press*, NewYork, Vol.2, pp.137-180