Figure 6: The tutor solves a Nth conflict between the learner and the expert.
Figure 7: The tutor opens the hypertext because the expert told him that he had to `repair' what the learner did.
Figure 8: If the expert cannot really attribute the learner actions to one of his own goals, he asks the learner what was his goal.
Figure 1: The learner builds an experiment (at level 1) by assembling `events' on the workbench. An event defines the task (read, listen, recall, wait,...) of a group of subjects using some material (e.g. a list of 12 words). Panes on both sides of the workbench provide libraries where the learner builds and select the components that he assembles on the workbench.
Figure 2:The Analysis of chosen items in the make-material tool

The distribution of pedagogical roles in a multi-agent learning environment.

P. Dillenbourg, P. Mendelsohn and D. Schneider
TECFA, Faculty of Psychology and Education, University of Geneva.

We describe a learning environment (MEMOLAB) that illustrates the distribution of roles among several agents. The learner solves problems in interaction with an expert, i.e. an agent who is able to solve the same problems through the same interface. The degree of assistance provided by the expert is tuned by another agent, the tutor, which monitors the interaction. MEMOLAB includes several tutors corresponding to various teaching styles. These tutors are selected by their superior, called `the coach'. This distribution of roles between the agents has been conceived in such a way that some agents (the tutors and the coach) are not directly concerned by the specific teaching domain and hence can be reused to build other learning environments. The set of domain-independent components constitute ETOILE, an Experimental TOolbox for Interactive Learning Environments. Its originality is that authors do not build a software application by writing questions and feedback, but by designing domain-specific agents that will interact with the agents provided by the toolbox.

1. Introduction

The term `intelligent learning environment' (ILE) refers to a category of educational software in which the learner is `put' into a problem solving situation. A learning environment is quite different from traditional courseware based on a sequence of questions, answers and feedback. The best known example of a learning environment is a flight simulator: the learner does not answer questions about how to pilot an aircraft, he learns how to behave like a "real" pilot in a rich flying context. Experience with learning environments (like LOGO) showed that those systems gain efficiency if the learner is not left on his own but receives some assistance. This assistance may be provided by a human tutor or by some system components. In our flight simulator example, the future pilot would gain from discussing his actions with an experienced pilot. In summary, we use the word `intelligent learning environment' for learning environments which include (1) a problem solving situation and (2) one or more agents that assist the learner in his task and monitor his learning.

This chapter describes how we distributed roles among agents in such a way that part of the pedagogical knowledge encapsulated by the agents could be reused for building other ILEs. We present two systems:

… MEMOLAB is a particular learning environment for the acquisition of basic methodological skills in experimental psychology.

… ETOILE (Experimental Toolbox for Interactive Learning Environments) is a toolbox that enables advanced programmers to build an ILE in another domain, but based on the same principles and architecture as MEMOLAB.

We started with the design and the implementation of MEMOLAB and we progressively abstracted the toolbox called ETOILE. It could appear more logical to build first the authoring tool and then to use it for creating an ILE. However, in this case, the researchers have to start with the over-general question: "What functionalities are shared by any educational software?". This approach generates constraints at a high level that unavoidably lead to a set of neutral interface tools and to procedures for specifying the sequence of learning activities. By proceeding inversely, we succeeded in designing a generic tool which encapsulates pedagogical knowledge.

ETOILE is not a proper authoring tool, with an author interface. It is a prototype to be used by programmers with advanced programming skills in Common Lisp. It provides them with the bricks of an ILE, but it is the author's task to assemble them into a coherent system. It supports the design process, it does not automate it.
ETOILE and MEMOLAB are implemented with Allegro Common Lisp and run on Sun Sparcstation. Since the whole systems are based on an object-oriented approach, we use the object-oriented features of Common Lisp, i.e. CLOS. For the interface functions, we use the beta-release of the `Common Lisp Interface Manager' (Clim 2.0). A full description of these systems can be found in Dillenbourg et al. (1993).

2. MEMOLAB

The pedagogical domain of MEMOLAB is the acquisition of the methodology in experimental psychology, i.e. experimental design. During the design of MEMOLAb, we concentrated on four qualities. Those qualities do not result from a formal study but arose from our experience in educational software. We illustrate these four qualities with MEMOLAB. In the next section, we will concentrate on the fourth quality, which justifies the multi-agent architecture. Together, these four qualities depict the profile of the products that can be built with ETOILE.

2.1. Quality #1: The learner must be active.

MEMOLAB targets the acquisition of the methodology of experimental design in psychology. Nowadays, this methodology is taught in most universities by presenting methodological principles and by discussing experiments from the literature. Students have no real design experience until they have to create an experience with real subjects for their own thesis. At that stage, any mistake in the experimental plan becomes very costly. This strategy for training psychologists could be compared to asking future pilots to fly real aircraft immediately after theory courses, skipping the flight simulator stage. MEMOLAB is an artificial lab where future psychologists can experience (some aspects of) experimental design, without facing the risk of spoiling their subjects. In Memolab, the experimental subject domain is the study of human memory. Basically, the learner's activities form a cycle: design an experiment, run it, and analyze the output data. Here is an example of an experiment that the learner can design: two groups of subjects will study a list of 20 words, and then later on try to remember these words. Between the encoding and the recall, the first group will wait during 10 seconds, while the second group will wait for 40 seconds. The comparison of their recall performance indicates the effect of the delay length. This experiment is illustrated by Figure 1. It illustrates how the learner designs an experiment within the first lab (we will see later that MEMOLAB includes several labs). An experiment is constructed by positioning a set of events on a workbench. The vertical axis represents the time dimension. The activities of a particular group of subjects are vertically aligned: In our example, they first encode some material, then there is some delay and then some attempt to recall what they studied before. When an experiment has been fully designed, the learner may ask to simulate the results. The details of the simulation are described in Dillenbourg et al. (1993).

2.2. Quality #2: The environment must be rich and complex

Any psychological or pedagogical theory investigates learning from a specific viewpoint. For instance, some of them pay attention to the effect of practice, but neglect the importance of understanding. Some concentrate on discovery, and forget about imitation. However, when one observes an actual learning process, it becomes clear that learning does not result from a single activity but from the integration of multiple activities, based on multiple sources of information. A learning environment must takes this complexity into account by offering a wide range of learning resources that will accommodate the variety of individual learning styles and learning needs. Therefore MEMOLAB includes several tools:

… The make-material tool

A difficult aspect in designing an experiment is to select carefully the items that the subjects will have to remember. A list of words has several features that the novice experimenter is not aware of, but which affect memory performance: the length of words (number of syllables), the frequency of words in our everyday language, the phonological or semantic homogeneity of a list of words. Figure 2 shows the "Analysis" tool of the make-material window.

… The hypertext `Methodology'

The very idea of a learning environment is that the acquisition of procedural knowledge requires active problem solving (by contrast to questions-based courseware that focuses

on declarative knowledge). However, the weakness of microworld designers has been to underestimate the role of domain specific knowledge. MEMOLAB corrects this bias by giving access to declarative knowledge (theories, laws, facts,...) about the methodology of experimentation and theories on human memory. The hypertext structure enables the learner to find information at the level of granularity he is interested in. Our hypertext facility includes the features now common to most similar applications (trace of read nodes, web view, book view, footnotes,...).

… The hypertext `Encyclopaedia of Memory'

Designing an experiment on human memory requires both methodological skills and some knowledge specific to the domain of memory. For instance, learners must know the concept of "independent variable", but they must also know that `word frequency' is an interesting independent variable linked to `word length'. MEMOLAB includes an hypertext that provides theoretical and experimental data on human memory.

… The simulation & data analysis tool.

Research on the use of simulation in educational research (Roschelle,1988; Fredriksen & White, 1988) has shown that the simulation method itself is important. The role of a simulation is not simply to produce realistic results. The simulation process must be carried out in such a way that the learner can understand the results. The simulation built in Memolab produces a commented trace of the simulation process that enable the learner to understand how these results have been computed. Furthermore, a set of statistical results is computed and the learner can inspect the generated data..

The richness of a learning environment is not only a function of the number of tools but of the extent to which those tools are integrated. We made this integration possible by using a homogeneous object-oriented internal representation scheme. Let us illustrate this with the simulation. The results of an experiment are computed by analogy with those of similar experiments. MEMOLAB explains how the results have been computed referring to these experiments (`Simulation Trace'). As illustrated by Figure 3, the learner may double-click on one of these references and jump directly into the hypertext node where the referenced experiment is described. The hypertext is also connected with the rule-based agents. Most current research on hypertext systems has so far been concerned with the features of the hypertext itself. The originality of our approach is to have considered the place of the hypertext within a complex system, to situate the information search within a context that makes this information useful.

2.3. Quality #3: The environment must be structured

If the richness of a learning environment is a quality, its complexity may reduce learning. The designer must structure his learning environment in a way that the learner may benefit from this richness. The `structured environment' idea illustrates a fundamental difference of perspective between learning environments and traditional courseware: the meaning of `teaching' shifts from asking questions and providing feedback (although this is not excluded) towards structuring the environment in a way that provides optimal learning conditions.

The structure we have adopted is a sequence of increasingly complex microworlds (or problem solving situations, or laboratories for Memolab). This sequence leads to a "final" microworld that should be as close as possible to the real context in which the learner will have to apply the acquired skills. For instance, the final lab in Memolab uses the language that is commonly used by the community of experimenters in their exchanges (papers, conferences,.). However such a microworld is probably too complex to be mastered directly by a naive learner (as in the real world). It should be preceded by simpler microworlds. In the first one, the learner will have to solve simple problems, considering only some basic parameters. In the next one, he will manipulate more powerful operators and to use additional resources available in order to cope with more complex problems.

The architecture of MEMOLAB includes three microworlds, referred hereafter as `levels'. Each level concerns a different kind of experiment and uses a partially different interface:

… At level 1, an experiment compares n groups of subjects along one dimension. The interface emphasizes the concrete, elementary and chronological aspects of an experiment.

… At level 2, the learner describes an experiment as a set of sequences. A sequence is a generic treatment that can be replicated several times with equivalent materials. The learner designs experiments that test the interaction between two parameters.

… At level 3, the interface reifies the logical structure of the experiment as a

plan.
By using a sequence of microworlds, one must address the issue of the learner's transition between microworlds. To smooth this transition, we designed the interface in such a way that these microworlds partially overlap. Each microworld uses a different set of commands for creating and for modifying the components of an experiment (events, sequences or plans). When an experiment has been completed at level N, the system translates this experiment into the representation that it would have at level N+1. The future representation scheme is therefore systematically associated with the actual scheme mastered by the learner. After some time, the learner is invited to express himself directly with the new language (N+1). This principle has been called the `language shift' (Dillenbourg, 1992).
We used the image of the pyramid in order to emphasize the hierarchical relationship between operators that are used in successive microworlds (Dillenbourg & Mendelsohn, 1992). This notion of hierarchical integration is based on the neo-piagetian theory of Robbie Case (1985). In brief, this theory states that the qualitative improvement of children's cognitive structures is impaired by quantitative limitations in controlling elementary structures. To by-pass this limitation, children restructure their knowledge into higher order schemes that recombine patterns of previously existing sub-schemes. The pyramid metaphor illustrates our conception of system design. In itself, the pyramid is not a very original image. It simply helps the designer to `view' his own system within a theoretical framework, for instance to think of the relationship between two corresponding system operators at different levels as the relationship between two schemas in Case's theory.

2.4. Quality #4: The learner must interact with agents

Learning does not only result from solving problems or using tools, but also from interacting with agents about these on-going activities. Even if the environment is rich and structured, some learners may need help while they solve problems. Some learners need to be pushed forward, otherwise they repeatedly use the same inefficient methods. An agent can of course be human, e.g. a tutor who monitors the learner's work or some peer learner sharing a desk. Regarding computational agents, we distinguish two types of agents:

… Agents that conduct domain specific interactions (i.e. help to build an experiment). These agents can be domain experts or a computerized collaborative learner (Dillenbourg & Self, 1992b). One can also have several experts that solve the same problem with different viewpoints (Moyse & Elsom-Cook, 1992). The interaction between the expert and the human learner concerns the problem to be solved.

… Agents that select and monitor learning activities or help the learner to manage his own learning process. In MEMOLAB, we discriminated two kinds of pedagogical agents, the tutors and the coach who is a kind of "tutor in chief". The various tutors have the same function but fulfill it differently according to their style.

This discrimination between two classes of agents has been essential in separating pedagogical knowledge from domain-specific knowledge and therefore to extract from MEMOLAB the components that constitute the toolbox. This architecture will be described in the next section.

3. Agents in ETOILE

3.1. Role distribution

We can summarize the previous section by saying that a learning environment is structured along two dimensions: A sequence of microworlds (with the described features) and a set of agents who interact with the learner within each microworld. By specifying several agents with various roles, we have been able to separate pedagogical knowledge from domain specific knowledge. The separation idea itself had two reasons:

… first, we aimed to build a domain-independent toolbox, i.e. to apply the same pedagogical knowledge on various contents;

… second, we wanted ETOILE to integrate various teaching strategies, i.e. to apply various teaching styles to the same content.

We should point out that the terms `domain-independent' and `pedagogical' are not synonymous. There exists some domain-specific pedagogical knowledge, for instance knowing that problem-23 is more difficult than problem-24. Therefore, we have to cope with the fact that the boundary between what is domain specific and what is pedagogical is not a straight line.
The originality of ETOILE with respect to other AI-based authoring shells such as DOCENT

(Winne and Kramer, 1988), IDE (Russell, Moran and Jordan, 1988), ISD (Merrill and Li, 1989) or CML (Jones and Wipond, 1991) is the approach chosen for separating pedagogical knowledge from domain-specific expertise. The common solution to this problem is to create a terminology for describing any subject domain (by using abstractions such as units, chunks, concepts, rules, principles,...). Variables in common pedagogical rules refer to this kind of terminology. The author's task is to describe the domain using those building blocks. However, building a terminology that fits such diverse domains as astronomy, music and engineering is a challenge that educators and even philosophers have been facing for 2000 years. Therefore, we advocate another approach. The pedagogical rules never refer to the domain. The pedagogical rules refer to the interaction between the learner and a domain-specific agent. These interactions are described in terms of agreement, disagreement, repair, and so forth. The domain-specific agent is (in the current implementation) an expert system able to solve any problem submitted to the learner. The domain specific terminology used by this expert is unknown to the tutor, it is confined to the expert (although shared with the learner) and therefore escapes from generality constraints. We will describe how the tutor influences the expert-learner interactions, but in order to do that we must first explain how the learner and the expert collaborate.

3.2. How to implement the interaction between the learner and a computational agent?

Imagine two expert systems that collaborate closely. Each expert has its own rule base, but they share the same facts representing the problem state. Any action performed by one expert changes the set of facts and is thus noticed by the other expert. Thereby, they permanently have a common representation of the current problem. However, our challenge was to get the expert system to collaborate with a human user, not with another expert system. Nevertheless, we applied the same principle of "sharing" facts: instead of an internal representation, based on predicates, the expert uses an external representation, based on interface objects. If the learner changes the problem state, the expert's representation is updated. Conversely, if the expert changes the problem state the learner can see the change.

The main implication for the expert's rule base is that most rule conditions refer to the problem state displayed on the screen and most rule conclusions change this problem-state display. The relationship between the expert rules and screen objects has been possible because we use an object-oriented inference engine. In short, each rule variable is related to a class and can only be instantiated by the instances of that class. When a instance of class X is displayed on the screen, we stored this object in an instance of another class called DISPLAYED-X. By this way, if rule variables refer only to `displayed' classes, we make sure they can only be instantiated by objects displayed on the screen.

The need for shared workspace appeared during our experiments with a previous collaborative system (Dillenbourg & Self, 1992b). In this system (PEOPLE POWER), human-machine collaboration was based on a dialogue about the task (instead on shared task-level actions). Within a shared workspace, if one partner wants to help the other, he can see the other's partial solution and provide concrete help by performing the next operation (Terveen et al, 1991).

The collaboration between the expert and the learner is performed step-by-step. A machine step is a sequence of rule firings that ends when an activated rule executes a `main command'. A learner step is a sequence of interface actions that ends with a `main command'. Consequently, a main command is a command that can be performed by both the learner and the expert. It serves as a basis for comparing the learner's actions with the expert's potential actions. At each step, the expert may either `propose' an action or `observe' the learner's action. It is generally the tutor who decides whether the next step should be performed by the expert or by the learner. In our terminology, the tutor decides the `local interaction mode' (LIM; `local' meaning `a single step in the solution path').

The global balance of learner's and expert's steps along a problem solving path determines the degree of collaboration between the learner and the expert. We refer to it as the global interaction mode (GIM). Its state is represented by an integer between 0 and 5. A global interaction mode equal to 0 corresponds to a demonstration done by the expert (the expert performing all the steps). A global interaction mode equal to 5 corresponds to a simple exercise that the learner performs alone. As it is implemented, the interaction mode is based on a quantitative criterion, i.e. the number of steps

performed by the learner and the expert. This is of course a limitation of our system because during problem solving some steps may be more critical than others (Dillenbourg, to appear) and it represents an interesting issue for future research.
Let us make clear that collaboration in the system remains flexible. The learner may always interrupt the expert in order to continue the task by himself. Conversely, when the learner is asked by the tutor to perform several successive steps, he is allowed to ask the expert to take over the task. The tutor may disagree with these learner's initiatives and coerce the learner to proceed with the same interaction mode. However, in case of repeated disagreement, the tutor will resign and ask the coach to select another tutor, more adapted to the learner's requests (see the notion of pedagogical drift in section 3.6. "Selecting teaching styles").
The expert has no pedagogical intention. Of course, the design of the expert is biased by pedagogical criteria. They can be found for instance in the granularity of rules or in the explanatory power of the comments associated with each rule. Nevertheless, during interaction, the expert's only goal is to solve the problem. It tries to maintain the minimal level of mutual understanding necessary to the joint accomplishment of this task. The expert does not try to teach, or to decide what is good for the learner. He asks questions to get answers, not to see if the learner knows the answers. Pedagogical decisions are the tutor's business.

3.3. Diagnosis or Agreement

The role of an expert in a learning environment is not only to guide or interact with the learner, but to detect the learner's mistakes and eventually to understand the cause of these mistakes in terms of missing or erroneous knowledge. This diagnosis process is often referred to as `learner modelling'. Diagnosis in ETOILE is inspired by the model tracing paradigm (Anderson, 1984). However, within the more collaborative style of ETOILE, we should rather use the word `agreement' instead of `learner modelling'. We consider three types of diagnosis:

… `Positive' diagnosis

The expert agrees with the learner if one of the rules that he could have activated leads to the same interface command as the command performed by the learner (same command and same arguments).

… `Unknown' diagnosis

If none of the rules that the expert was ready to activate corresponds to the learner's action, the expert states that he does not understand the learner.

… `Negative' diagnosis

If, after the learner's action, the expert triggers a `repair rule' (explained below), this means that the learner's previous action produced something wrong, something to be repaired.
The expert has a small set of `repair rules'. The concept of repair rule is close to the concept of malrule that has been intensively used in student modelling (Dillenbourg & Self, 1992a). The expert's normal rulebase implements a preferred optimal solution path for a task. However, interaction with the learner may move the expert away from optimal to other possible solutions. Outside the solution path, we encounter erroneous problem states that are not covered by the expert's normal rules. Hence, we added a few `repair' rules enabling the expert to recover from these situations. Determining these situations a priori implies to anticipate the classical learner mistakes. For instance, in MEMOLAB, the expert can repair a list of words which is unevenly distributed (all the long words at the beginning and the short one at the end). This diagnosis technique raised a set of technical problems that led us to modify some aspects of the inference engine.

3.4. The learner-expert-tutor triangle

The tutor monitors the expert-learner interactions: he receives messages from the learner or from the tutor about what is going on. On the basis of this information, the tutor takes a pedagogical decision. For instance, the expert may send a comment that the learner is wrong, and the tutor will decide whether, in case of error, the learner should be interrupted or not. This decision depends on the tutor's teaching style: one tutor could decide an intervention while the other would not care. We illustrate this triangular relationship by two short scenarios. Messages from and to the learner are materialized as popup windows, messages between the expert and the tutor are not visible on the screen
In the first scenario (Figure 6), the learner performs some action not understood by the expert (`unknown' diagnosis). The expert then asks the learner whether he can `undo' what

the learner did. He also sends a message to the tutor to inform him that he did not understand the learner (step 1). The learner has the possibility to refuse that the expert undoes his action. He answers "no" to the expert, this refusal being perceived by the tutor (step 2). The tutor then has to solve a conflict between the learner and the expert. In Figure 6, the tutor enforces the expert's initiative: he tells the learner "listen to the expert" and he asks the expert to continue (step 3). In order to solve this expert-learner conflict, the tutor does not try to determine who is right. He could not take such a decision since he hasn't any domain knowledge. By definition the expert is right. However, the tutor may decide for pedagogical reasons that despite the learner' mistake, it may be better to leave him exploring his own ideas. The tutor illustrated in Figure 6 (called Vygotsky) actually hands control to the learner at the first conflict, but gives preference to the expert at the second one (for the same problem). The dialogue illustrated in Figure 6 corresponds to the second time the learner refuses to listen the expert.

Figure 7 illustrates another mini-scenario. The expert fires a repair rule and informs the learner about the reasons why he changed something on the screen (step 1). The expert sends another message to the tutor to say that he had to repair something that was wrong. After the learner has read the expert's message (step 2) and has clicked on `ok', the tutor processes the expert's message. In the dialogue illustrated by Figure 7, the tutor's decision is to provide the learner with some theory related to the learner's mistake. The expert's repair rules which correspond to classical mistakes, are connected to specific nodes of the hypertext. Another tutor could decide not to open this hypertext and to leave the learner trying to discover these facts by himself.

This last scenario illustrates the debate about the separation of pedagogical and domain knowledge. The expert is an egocentric agent: he `repairs' the experiment because he wants to build a good experiment, not because he wants to teach the learner. The pedagogical decision is taken by the tutor. However, expert design is not completely neutral from a pedagogical viewpoint since the anticipation of the typical learner's major mistakes are encoded in the rules.

3.5. Teaching styles

The tutor has to take several pedagogical decisions, some of which already have been described:

… to choose the value of the global interaction mode;
… to choose the value of the local interaction mode;
… to solve conflicts between the learner and the expert (`undo');
… to choose what to do in a case of a `repair' by the expert;
… to choose the next problem to be proposed to the learner and the expert;
… to decide when to present theory (from the hypertext), i.e. to prefer an inductive (examples first, theory next) or deductive (theory first, examples next) approach;
… to decide when to end the session concerning a specific goal (success or failure);
… to decide to resign before the end of a goal session (explained in section 4.6. on page 12).

Each of these decisions implicitly define an axis along which tutoring behaviour can vary. Originally, in our system, tutor behaviour was summarized by a set of parameters, corresponding more or less to these axes. The value of each parameter was chosen by a set of rules. When we wrote these rules, we learned that the parameters were actually not independent from each other. It is for instance difficult to marry a inductive approach with a high level of interruptiveness. The concept of teaching style stems precisely from the consistency among a set of pedagogical decisions.

With all those teaching parameters it is theoretically possible to define a very large number of teaching styles. However, we reduced this large space to a few styles in order to define styles that differ significantly from each other. There are two reasons for working with a few contrasted teaching styles. The first reason is that if the learner fails to reach a goal with a tutor applying some style, there is a low probability that he would succeed with an almost identical style. The system does better to choose a radically different approach. The second reason for having clear differences between teaching styles is that the learner should perceive these differences (Elsom-Cook, 1991), namely be able to choose his `favourite' tutor later on.

The five teaching styles we have defined are respectively labelled Skinner, Bloom,

Vygotsky, Piaget and Papert. These names do not imply that the we succeeded in translating the theories of these five key figures in the history of education and psychology. We rather use these terms as flags: we defined a tutor that roughly teaches `ý la Vygotsky', another one `ý la Papert' and so forth. Each teaching style corresponds to a tutor and is implemented as an independent rule base. Each rulebase is itself divided in three rule sets: (1) selection of activities and problems, (2) monitoring of joint problem solving and (3) ending a session. There is of course some overlap between the knowledge of each tutor, but it is easier to manage them as separate rule bases.
The only tutor that has so far by-passed the stage of an early implementation and goes through some experimentation is Vygotsky. Before investing in the full implementation of other tutors, we need to solve the problems related to the expert-learner interaction (see section 4. "Experiments"). From the outset, Vygotsky selects rather difficult problems. The expert initially performs most of the problem solving steps (GIM = 2). Then Vygotsky progressively increases the global interaction mode, i.e. the expert fades out and the learner plays an increasingly important role in the solution process, until he solves the problem alone (GIM = 5). In other words, the Vygotsky tutor is characterized by a particular GIM curve (GIM X Time), while Skinner would be depicted by a `problem difficulty' curve.
We must acknowledge that the design of teaching styles is bounded by the general architecture of ETOILE. The various components of ETOILE already have some pedagogical bias: the coach's goal selection process is inspired by mastery learning, the expert-learner interaction is inspired by apprenticeship, and the environment facilities are inspired by the microworld philosophy. If we had wanted to push to its end the logic of each teaching style, ETOILE would become much more complex. For instance, the notion of diagnosis is not the same (a) in a behaviourist feedback based on the learner's performance, (b) in a constructivist approach based on the learner's cognitive structure, and (c) in the apprenticeship mode where the expert attempts to integrate the learner's action into his own framework.

3.6. Selecting teaching styles

Since ETOILE includes several tutors, it needs a hierarchically superior agent which selects the tutors. His name is `the coach'. The coach also selects the goals in the pedagogical curriculum. The goal selection process and tutor selection process have been implemented in the simplest and most transparent way. The coach selects the first goal whose pre-requisite goals have all been mastered (remember that the curriculum is a network of goals with prerequisite links). When the learner succeeds that goal, its status is set to `mastered' and the goal selection process is repeated.
After a goal has been selected, the coach selects a tutor. The mission of a tutor is always relative to a single goal. Goal selection is mainly based on the learner's preferences and his performance history. If he has already failed the current goal with tutor T1, the coach will select a tutor T2 that provides more guidance than T1. On the contrary, if the learner succeeded the previous goal with tutor T1, the coach will select a tutor that provides less guidance. Our five tutors are sorted by decreasing level of directiveness: Skinner works step by step, Bloom makes larger steps but with close control of mastery, Vygotsky is based on participation, Piaget intervenes only to point out some problems and Papert does not interrupt the learner.
This general principle is however shadowed by more specific rules that recommend to avoid activating a tutor that has been repeatedly inefficient with a particular learner. The learner may also select a tutor. These preferences are recorded and taken into account in later selections. The learner (or a human teacher) can also remove a tutor from ETOILE's set of tutors for the rest of the session.
The tutor has rules to determine when the learner has mastered or not a goal and he can close a session concerning a goal. In order to provide more flexibility to the system, tutors can resign before the end of their `contract'. If a tutor like Piaget observes that the learner is frequently asking for help or for theory, this means that his teaching style does not fit the learner needs. Conversely, if the learner being taught by the tutor `Skinner' always asks to continue the problem on his own, he would certainly be more efficient with another tutor. We refer to this process a `pedagogical drift', because the actual interaction with the learner may lead the tutor to `drift away' from the behavior space for which he has been designed for and within which he is supposed to be efficient. In case of drift the tutor returns the control to the coach and indicates the type of drift that he observed. The coach will then select another tutor.

This monitoring of pedagogical drift is implemented in a very simple way: each learner command that has an effect on the locus of learner activities' control creates as a side-effect an instance of the class `initiative'. This class possesses two subclasses `want-more-guidance-initiative' and `want-less-guidance-initiative'. Some commands like seeking expert help or asking to read the hypertext generate instances of `want-more-guidance-initiative'. Some learner actions, like `I want to continue', `I want to select the problem', `I refuse to undo', create instances of `want-less-guidance-initiative'. All significant commands in ETOILE are characterized with respect to these two subclasses. Therefore, all domain-specific commands relevant at this level must create an instance of one of these two sub-classes. The tutor compares the number of instances in each subclass and resigns if there is a strong disequilibrium between the two sets of used commands.

4. Experiments

We have conducted preliminary experiments with a few subjects. These experiments were too early to draw conclusions regarding the efficiency of MEMOLAB. Before extending experiments to samples supporting statistical inference, we have to incorporate these preliminary results into our systems. Experimentation pointed out several difficulties in the learner-expert interaction and this is important because this interaction is a key factor for role distribution among ETOILE's agents.

When the learner creates an event on MEMOLAB's workbench, the expert tries to map this event to task goals related to his own problem solving activity. In other words, the expert attempts to understand what the learner does. This interaction is illustrated by Figure 9. Bijective linking between objects on the screen and task goals was implemented in order to optimize the expert, but it raises several problems when an event may be attached to several task goals. If the expert has planned that Group-1 will wait for 10 seconds and Group-2 will wait for 40 seconds, the learner may very well have in mind that Group-1 will wait for 40 seconds and Group-2 for 10 seconds. We use the term `goal commutativity' to express the fact that the allocation of parameters to experimental samples is arbitrary, and that the expert must be able to permute his goals.

However, mapping student actions to task goals raised several issues. Firstly, learners don't necessarily have goals, but instead start to think about goals when they are proposed. This is not a bad point. The second issue concerns the way in which the expert describes his goals to the learner. For instance, in Figure 9, the expert's goal described in `choice 1' is actually that Group-2 waits for 40 seconds, but it would be nonsense to propose a goal concerning Group-2 when referring to an event concerning Group-1. Hence, the expert generalizes his specific goal in order to cover the learner's intentions. This led to ambiguous situations where the expert was suggesting different goals that were actually expressed by the same sentence, generating a misunderstanding between the learner and the expert.

However, these difficulties gave us the opportunity to observe instances of `social grounding'. Social grounding is the mechanism through which each participant in a dialogue elaborates the belief that his partner has understood what he said, to a level sufficient for continuing the conversation (Clark & Brennan, 1991). In the protocols we encountered utterances (produced in French and translated by us) showing that the learner monitors the understanding that the system has about himself:

"He supposes that I wanted the subjects to do something during 40 seconds. I wanted the subjects to do nothing."

"He does not understand why I did that? "

"Precisely, what I want is that the subjects do nothing, I think that it is what he thinks"

"I am sure he will tell me again that I am wrong"

"He will ask me what I wanted to do. And the..., since... he'll base himself on wrong things since this is not what I want to do."

In other words, the learner-expert interaction is not simply based on mutual diagnosis with nested levels of beliefs: the expert diagnoses the learner, the learner diagnoses the expert, the learner diagnoses the expert's diagnosis,... From a theoretical viewpoint, one could go on about the diagnosis of the diagnosis of the diagnosis... However, in human-human dialogue, this cascade is not very long because human conversation is very rich in resources to detect (e.g. facial expressions) and repair (e.g. pointing) communication breakdowns. As soon as we reach the level "diagnosis on diagnosis", one speaker can repair the dialogue. In the fictitious example below, speaker

1 repairs the understanding that speaker 2 has about his first utterance.
Speaker 1: "Numbers ending with 4 or 6 are even numbers"
Speaker 2: "778 is also an even number"
Speaker 1: "I did not say that those ending with 8 are not even"
When a speaker refines a previous statement to correct some misunderstanding, quite often he does not simply repeat what he said but reinterprets it from the point of view of his partner. The fact that the learner reinterprets his own actions within the expert's conceptual framework corresponds to a learning mechanism that has been emphasized in the socio-cultural approach: participation changes understanding (Newman, 1989; Rogoff, 1991). The benefits from collaborative work seem to be grounded in the mechanisms engaged to maintain a shared understanding of the problem (Roschelle,1990).
Through these results, the process of cognitive diagnosis appears as mutual and reciprocal. Instead of having one partner who builds a diagnosis of the other, we have two collaborators who try to build a mutual understanding (Bull, Pain & Brna, 1993). A similar evolution can be observed in work on explanation: the explanation is no longer viewed as a structure built by one partner and delivered to the other, but as a new structure jointly constructed by two partners during their interaction (Baker, 1993; Cawsey, 1993). This evolution of diagnosis and explanation techniques drives the evolution of cognitive science towards the idea of distributed cognition (Resnick et al, 1991; Shrage, 1991; Dillenbourg, to appear).

5. Synthesis

We developed an original solution for separating domain-independent from domain-specific knowledge. Instead of searching for a domain-independent language able to describe any domain, we describe pedagogical knowledge in terms of interactions among agents. A learning environment is a society of agents for which we have specified (1) the role distribution over agents (coach, tutor, experts) and (2) protocols of communication among agents (global and local interaction mode, repair, conflicts,...). Some of the choices made in ETOILE are still arbitrary. For instance, the only domain-specific agents we implemented are `experts' while we could integrate `peer learners' or semi-experts. Another limitations is that `agents' refer to a simple data-structure, including namely a rulebase that drives his behavior. Current agents do not have the autonomy and goal-orientedness that characterize a proper multi-agent system (Soham, 1993). The specification of well-defined communication protocols among agents would enable the integration of heterogeneous agents, namely agents which do not rely on rule bases or agents written in a different programming language. This approach would also extend ETOILE's scope to designers outside the LISP community.
By separating pedagogical from domain knowledge, we are not only capable of permuting the domain knowledge associated with some teaching strategy, but conversely, we are also able to apply various teaching styles to a particular domain expertise. Instead of arguing which teaching style is the best, we prefer to cover a large range of teaching styles. Of course, this raises the issue of selecting the teaching style. Despite this issue being non-trivial from some theoretical viewpoint, we solved it with a simple pragmatic principle: if the learner fails, change the teaching style! We would like to develop future research on this learner's perception of teaching styles.
Another achievement concerns the integration of multiple tools within a learning environment, especially the integration of a hypertext with rulebase agents (Schneider et al., 1993). Nowadays, hypertext systems are a very popular research topic. We believe that the pedagogical power of a hypertext cannot be understood without considering the context that justifies information search in a hypertext.
Regarding the interaction between the leaner and a computerized agent, we applied is very simple principle: human-computer interaction should concern what is ON the screen. The design of the expert, has been determined by this principle: the rule conditions `read' the problem state on the screen, the rule conclusions change the problem state display. This design principle produces a nice opportunistic expert-learner collaboration and makes this interaction inspectable and tunable by the tutor. Experimentation with users helped us to find problems related to this approach, namely the difficulty of interacting about goals. But they also gave us some insight for implementing mechanisms of social grounding between a human user and a machine. Social grounding is a mechanism of joint and reciprocal modelling between two partners, based on the negotiation of shared meanings. A limitation in our current architecture is that the rule variables are instantiated by screen objects without any ambiguity. A new research avenue would test

interactions where the computer's concepts were not directly connected to the displayed objects (the learner concepts are neither), and where these mappings could be negotiated with the learner through various pointing mechanisms.

6. Bibliography

ANDERSON, J.R (1984) Cognitive Psychology and Intelligent Tutoring. Proceedings of the Cognitive Science Society Conference, Boulder, Colorado, pp. 37-43.

BAKER, M. (1992) The collaborative construction of explanations. Paper presented to "DeuxiËmes JournÈes Explication du PRC-GDR-IA du CNRS", Sophia-Antipolis, June 17-19, 1992.

BULL, S., PAIN, H. & BRNA, P. (1993) Collaboration and Reflection in the Construction of a Student Model for Intelligent Computer Assisted Language Learning. Proceedings of the Seventh International PEG Conference, Edinburgh, July 1993.

CASE, R. (1985) Intellectual Development form Birth to Adulthood. New York: Academic Press.

CAWSEY A. (1993) Planning Interactive Explanations. International Journal of Man-Machine Studies, 38, 169-199.

CLARK, H.H. & BRENNAN S.E. (1991) Grounding in Communication. In L. Resnick, J. Levine and S. Teasley. Perspectives on Socially Shared Cognition (pp. 127-149). Hyattsville, MD: American Psychological Association.

DILLENBOURG, P. (1992) The Language Shift: A mechanism for Triggering Metacognitive Activities. In M.Jones and P. Winne (Eds.), Adaptative Learning Environments (pp. 287-316). Berlin: Springer Verlag.

DILLENBOURG, P. (To appear) Distributing Cognition over Humans and Machines. in S. Vosniadou, E. De Corte, R. Glaser and H. Mandl (Eds) International Perspectives on the Psychological Foundations of Technology-Based Learning Environments. New York: Springer-Verlag.

DILLENBOURG, P. & MENDELSOHN, P. (1992) The Genetic Structure of the Interaction Space. In E.Costa (Ed.), New Directions for Intelligent Tutoring Systems (pp 15-27). Berlin: Springer-Verlag.

P. DILLENBOURG, M. HILARIO, P. MENDELSOHN, D. SCHNEIDER, B. BORCIC (1993) Intelligent Learning Environments. Research Report. TECFA, FacultÈ de Psychologie et des Sciences de l'Education, UniversitÈ de GenËve

SCHNEIDER, D., DILLENBOURG, P., MENDELSOHN, P., HILARIO, M. & BORCIC, B. (1993). IntÈgration d'un hypertexte dans un environnement d'apprentissage ý initiative mixte. Colloques HypermÈdias et Apprentissages - Third European Congress - Multi MÈdia, Intelligence Artificielle et Formation. Lille, APPLICA-93, 22-25.

DILLENBOURG, P. & SELF, J.A. (1992a) A framework for learner modelling. Interactive Learning Environments, 2(2), 111-137.

DILLENBOURG, P. & SELF, J.A. (1992b) A computational approach to socially distributed cognition. European Journal of Psychology of Education, 3 (4), pp. 353-372.

ELSOM-COOK M.T. (1991) Dialogue and teaching styles. In P. Goodyear (Ed). Teaching knowledge and intelligent tutoring (pp. 61-84). Norwood (NJ): Ablex.

FREDERIKSEN, J. & WHITE, B. (1988) Intelligent Learning Systems for Science Education. Proceedings of the Internatioal Conference on Intelligent Tutoring Systems. ITS-88. Montreal, Canada. June. pp. 250-257.

JONES M. & WIPOND K, (1991) Intelligent environments for curriculum and course development. In P. Goodyear (Ed). Teaching knowledge and intelligent tutoring (pp. 379-395). Norwood (NJ): Ablex.

MERRILL M.D & LI Z. (1989) An instructional Design Expert System. Journal of Computer-Based Instruction, Vol. 16, 3, pp. 95-101.

MOYSE, R. & ELSOM-COOK, M. (1992) Knowledge Negotiation. London: Academic Press

NEWMAN, D. (1989) Is a student model necessary ? Apprenticeship as a model for ITS. Proceedings of the 4th AI & Education Conference (pp 177-184), Amsterdam: IOS.

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

ROGOFF, B. (1991) Social interaction as apprenticeship in thinking: guided participation in spatial planning. In L. Resnick, J. Levine and S. Teasley. Perspectives on Socially Shared Cognition (pp. 349-364). Hyattsville, MD: American Psychological Association.

ROSCHELLE, J. (1990) Designing for Conversations. Paper presented at the AAAI Symposium on Knowledge-Based Environments for Learning and Teaching, March. Stanford,CA.

ROSCHELLE, J. (1988) Sufficiency and utility of physics problem solving processes. Proceedings of the Internatioal Conference on Intelligent Tutoring Systems (ITS-88).

Montreal, Canada. June. pp. 132-139
RESNICK L., LEVINE J., and TEASLEY S. (1991) Perspectives on Socially Shared Cognition. Hyattsville, MD: American Psychological Association.
RUSSELL D.M., MORAN T.P., JORDAN D.S. (1988) The Instructional-Design Environment. in J. Psotka, L.D. Massey and S.A. Mutter (Eds) Intelligent Tutoring Systems. Lessons Learned. (pp. 203-228). Hillsdale, NJ. Lawrence Erlbaum
SCHRAGE M. (1990) Shared Minds. The new technologies of collaboration. New York: Random House
SOHAM, Y. (1993). Agent-oriented programming. Artificial Intelligence, 60, pp. 51-92.
TERVEEN, L.G.. WROBLEWSKI, D.A. & TIGHE S.N. (1991) Intelligent Assistance through Collaborative Manipulation. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-91). pp. 9-14. Sidney, Australia.
VYGOTSKY, L.S. (1978) The Development of Higher Psychological Processes (edited by M.Cole, V. John-Steiner, S.Cribner and E. Souberman). Harvard University Press. Cambridge. Mass.
WERTSCH, J. (1985) Adult-Child Interaction as a Source of Self-Regulation in Children. In R. Yussen (Ed). The growth of Reflection in Children (pp. 69-97). New York: Academic Press.
WINNE P. & KRAMER L.L. (1988) Representing and inferencing with knowledge about teaching: DOCENT- an artificially intelligent planning system for teachers, Proceedings of Intelligent Tutoring Systems (ITS-88), (pp.7-15), Montreal, Canada.

Figure 3:Integration of the hypertext and the simulation: the learner clicks on an experiment referred in the simulation trace and gets the hypertext node where this experiment is described.

Figure 4: The learner-tutor-expert triangle in ETOILE: the tutor monitors the interaction between the learner and a domain-specific agent.

Figure 5: Step-by-step collaboration between the learner and the expert