An Experimental Toolbox for Advanced Interactive Learning Environments

May 1993
LISP Users and Vendors Conference (LUV-1993)

D.SCHNEIDER, B.BORCIC, P. DILLENBOURG, M. HILARIO, P. MENDELSOHN

TECFA
University of Geneva
Faculté de Psychologie et des Sciences de l'Education
9 route de Drize, 1227 Carouge, Switzerland

schneide@divsun.unige.ch
Phone.: ..41 22 705 96 94
Fax.: ..41 22 342 89 24

WWW-server: http://tecfa.unige.ch/www/welcome.html
Gopher & Anonymous Ftp: tecfa.unige.ch

Abstract

The design of educational software has evolved during three decades, reflecting technical advances and changes in theories of instruction. In this project, we implemented some features of advanced interactive learning environments (ILEs): the multiplicity of teaching styles (the same content may be taught in several ways), the multiplicity of learning sources (experience, coaching, hypertext browsing), the use of a rich interface allowing direct manipulation and free exploration, the use of artificial intelligence techniques for creating complex problem situations and for supporting pedagogical reasoning.

We developed an Experimental Toolbox for Interactive Learning Environments (ETOILE) and an application (MEMOLAB). ETOILE provides designers with a set of tools and resources allowing them to create their own ILE. Globally, ETOILE provides the pedagogical architecture while the designer provides the domain expertise and creates the specific parts of the interface. ETOILE is not an authoring tool that can be mastered by any genuine author, it is a toolbox to be used by designers with programming skills in Common Lisp and in CLIM (Common Lisp Interface Manager). ETOILE namely includes an Object-Oriented Production System, a hypertext, various interface management facilities and pedagogical knowledge bases.

MEMOLAB is a learning environment for acquiring basic skills in experimentation methodology for human sciences. It illustrates the kind of systems that can be designed with ETOILE. MEMOLAB includes several components. The 'Lab' allows learners to build an experiment on human memory. The 'Simulation' produces the results of this experiment. The 'Data Tools' allow the learner to visualize the results and to compute basic statistics. During their activities, learners receive the assistance of a tutor. The system includes five different tutors, implementing various teaching styles. The learner may also explore the various hypertextes in which he can find knowledge about designing experiments and knowledge about human memory.

The design of ETOILE and MEMOLAB reflects our efforts to translate a theory of development in terms of system specifications, namely to structure a learning session into a sequence of micro-worlds, each microworld being characterized by a specific interface language. The psychological theory is encoded in the relationship between the languages of successive microworlds. This kind of learning environment architecture is supported by the knowledge structures embedded in ETOILE.

ETOILE and MEMOLAB currently have Beta Status. Both are available to the research community from our FTP server (Please contact the authors via e-mail for more information). A User Guide, and partially completed User Manual and Reference Manual are available.

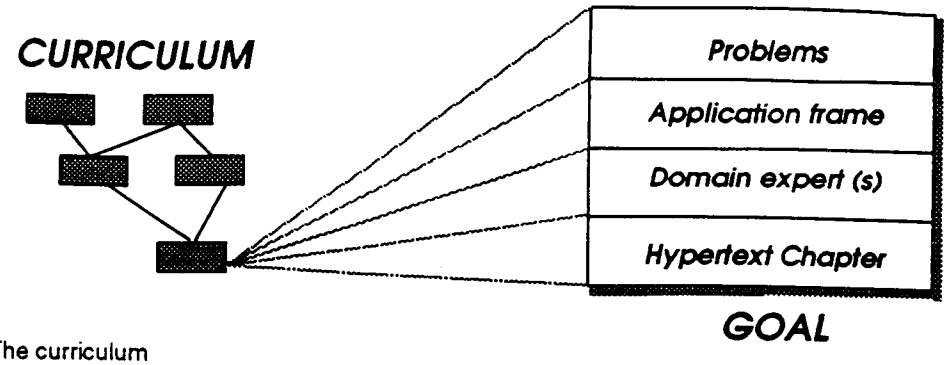

Figure 1: The curriculum

to the learner when the problem is selected. The author should indicate the difficulty of each problem (with an integer from 1 to 5). The author may add domain-specific knowledge, by creating a sub-class of 'problem', with additional slots, specific to his system (as we did for Memolab).

2. The **application frame:** Learners solve problems through an interface which is specific to the ILE to be created. This interface is basically one or more windows, with a set of commands (menus, buttons, object manipulations,...). ETOILE is based on the Common Lisp Interface Manager (CLIM) which provides us with the concept of the "application frame". An application frame precisely gathers all the information that defines a learning situation, e.g. the window, the command tables, etc. In many learning environments, the problem solving situation remains the same throughout the learning process. ETOILE allows the author to diversify the situation, for instance to ask learners to solve the same kind of problems with a new set of commands (as it is the case in MEMOLAB). Such "application frames" are some kind of microworld.
3. The **domain expert**(s): An expert for goal-X is a rule base able to solve any problem stored in goal-X. Its role is crucial for the interaction between the learner and the machine.For instance, diagnosis is carried out by searching for an expert's rule which implies the interaction (command) that the learner has just performed. The conclusions of the expert's rules include commands which belong to the command table (stored in the goal's situation frame). This relation between the rules and the interface objects is made possible by the object-oriented inference engine (OOPS) we have implemented. The author may define several experts for the same goal. This would support the implementation of multiple viewpoints.
4. The **hypertext:** Finally, the designer is expected to add to the goal a more readable representation of the expert's knowledge in hypertext form. The learner will explore freely the information relative to the skills he must acquire. This information must be structured in granularity layers, in such a way that the learner can deepen some points and pass quickly over others. The functionalities offered by the hypertext are described in section 2.4.

2.2 Tutors, coach and experts

ETOILE implements a variety of tutoring styles, each of which is implemented with an independent OOPS rule-base. Rule-bases are hermetically sealed. This guarantees the internal consistency of teaching styles and enabled us to reduce the number and complexity of rules and thereby to get a better control of the system behavior. Five tutors (i.e. five teaching styles) have been defined: Papert (free discovery), Piaget (guided discovery), Vygotsky (apprenticeship), Bloom (mastery learning) and Skinner (programmed learning). The teaching style (i.e. the tutor) is selected by the coach. Selection of a tutor is based on efficiency criteria and on the concept of **'pedagogical drift'**. Let us imagine that a learner follows the apprenticeship style, but fails often, receives a lot of feed-back, and asks frequently for help. In this case, the actual interaction will look more like mastery learning. This is what we refer to as a drift. The coach could then decide to 'jump' to mastery learning, rather than conduct an interaction which does not correspond to any style.

How do Coach, tutors, experts and the learner interact? (see Figure 2, "Coach, Tutors and Experts," on page 4) Since we have several tutors, there must be somebody above them who will decide which tutor should be activated. The coach first selects a goal, according to the goals previously mastered by the learner. Then, the coach selects a tutor. If the learner has been successful, the coach will for instance select a tutor which provides less assistance to the learner. The coach will avoid selecting tutors that previously have not been very efficient with that learner.Then the selected tutor will ask the learner to solve one of the available problems in collaboration with the expert. The tutor will determine the mode of collaboration. He may also encourage the learner to read specific parts of the hypertext documents. The coach is implemented as a rule base. You may inspect these rules and, if you feel comfortable with their syntax and semantics you may modify them or add new rules.

2.3 The collaboration between the learner and the expert

Courseware produced with ETOILE is based on the "learning by doing" idea, i.e. the learner solving prob-

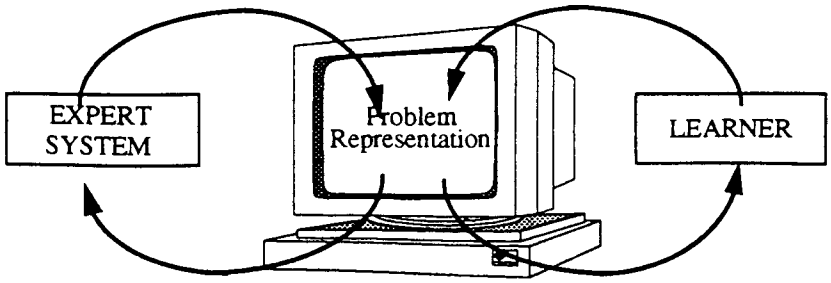

Figure 3: The interface: a "shared" problem representation

- The ***number of steps*** performed by the expert matches the global-collaboration mode. A step is a sequence of rules firing in which the last one concludes to an interface command.
- If ***no more rule*** is fireable, it is supposed to mean that the problem has been solved.
- Between two 'propose' steps, the learner can interrupt the process. The expert displays a pop-up window, shows the message stored with the last instantiated rule and asks the learner whether he wants to continue or to stop.

When the local mode is 'observe', the expert compares the command performed by the learner with the commands it would have performed itself. This comparison does not really correspond to the student modelling process that is performed in intelligent tutoring systems. Student modelling aims to detect the fundamental misconception which generates the learner's erroneous behavior and it can be compared to a psychologist analyzing the learner's behavior. The diagnosis in ETOILE rather corresponds to what one does when one works with somebody. You maintain some representation of your partner, you have some ideas about why he does something and you try to understand what he does. Deep understanding would be too time consuming. In our '***shallow diagnosis***' the expert tries to get the minimal understanding of the learner command to continue the task and to inform the tutor whether this behavior seems correct or not. This diagnosis is less ambitious than the general use of the term diagnosis. It is however improved by the fact that it is ***frequent and local***, i.e. carried out at every single step of the solution process and bounded by the context of this single step.

When the expert compares the learner's command, it considers all the rules that it could have fired in the similar situation. This comparison leads to three outputs:

- if the learner command (instantiated) corresponds to the command that appears in the action part of one of the expert's activable rules, then the message is set to 'positive-diagnosis';
- if the learner command does not match a rule but a repair-rule, i.e. a rule that you have declared as incorrect, the message is 'negative-diagnosis';
- If the learner command does not match any activable rule, the message is 'unknown-diagnosis'

The tutor uses the expert's message as rule condition to decide what to do next. The OOPS-engine has a 'preview-rulebase' function which allows to determine the set of activable rules within a rulebase. Since the learner command changes the state of the problem, the rules activable during the diagnosis process may not be the same than those activable just before. Hence, the preview-rulebase is performed before the learner does something and its result is stored until the learner does something and the comparison can occur.

Another implication of this approach, is that the tutor must have the possibility to say 'your action is incorrect' and to restore the problem to its state before the last learner command. This means that each command must be associated to a 'undo' command which has exactly the opposite effects on the problem state display. The other components of ETOILE are also implemented in CLOS. Because of this high integration of the various components, a rule action may open an hypertext chapter or an hypertext chapter may contain a runnable example. ETOILE has an "open" architecture.

2.4 Hypertext

A hypertext module was deemed the most appropriate computational infrastructure for the knowledge resources offered to the learner. The hypertext approach views a document as a set of text chunks corresponding to conceptual nodes in a database. A system of machine-supported links between mouse-sensitive strings (buttons) and chunks (nodes) allows interactive branching within and between documents. Conceptual entities are displayed as scrollable windows of text which may contain any number of integrated buttons representing links to other nodes. The user can navigate freely around the hyperdocument by clicking on any of these graphics pointers.

ETOILE's hypertext is inspired by Genera's Document Examiner. A hypertext is written by using a markup language (i.e. a small subset of Latex commands) which then have to be compiled for on-line use. We included book structure commands (chapter, section,...). This way an author can combine the advantages of linear and 'web' text. A graphic browser allows to display the linear structure of a hypertext and to jump to a section. Because of the use of a Latex-like syntax, hypertext documents can be printed easily. For instance,

A challenge was to articulate a computational architecture with psychological theories of learning. ETOILE supports the cognitive architecture of MEMOLAB, which is explained below, but does not coerce the designers to do the same. Our cognitive architecture divides the learning cycle into stages visualized by the "pyramid" metaphor. Actually, in MEMOLAB, each goal does not have a completely different application frame (interface). For each goal, the problem solving situation is the LAB, i.e. a workbench where learners can assemble experiments and run them. However, the set of commands available is not the same for each goal. The LAB is the central place of interaction. It is connected to subordinate frames, such as the data-tools (where learners analyze the data collected through the experiment) and the simulation. The simulation is a set of procedures that approximate the results of the learner's experiment by comparing this experiment with the most similar experiment found in a database in which we stored experiments described in scientific literature. The simulation method is inspired by case-based reasoning techniques (see 3.2).

Finally, MEMOLAB includes two hypertextes. The 'handbook of methodology' includes several chapters, connected to the various goals of the system. It includes a theoretical introduction to the design of well-formed experiments. The 'encyclopedia of memory' is another hypertext, independent from any goal, i.e. that learners can access at any time, and which provides them with the knowledge about human memory they could need, for instance, for determining the relevant variables.

3.1 The Cognitive Architecture

Designing an intelligent learning environment (ILE) involves implementing some theory of learning and teaching. However, most available theories do not have the level of operationality required for implementation work. Designing a ILE is real research work. We are developing an intermediate framework that builds a bridge between theories and implementations by translating psychological knowledge into terminology more relevant to computer scientists. It specifies the cognitive architecture of systems like MEMOLAB. Let's examine two key concepts: the *pyramid metaphor* and the *language shift* mechanism.

The "pyramid" metaphor represents the concepts and skills to be acquired by the learner, ranked bottom-up according to their level of "hierarchical integration". Learning consists in moving up in the pyramid. Each level of the pyramid is defined by two languages: the *command language* and the *description language*. The command language vocabulary is the set of elementary actions that the learner is allowed to do at some stage of interaction. The command language syntax defines how the learner composes sequences of elementary actions. The description language is the set of symbols (strings, graphics,...) used by the computer to show the learner some description of her behavior. This description reifies some abstract features of the learner's behavior in order to make them explicitly available for metacognitive activities (Collins and Brown, 1988).

The command and description languages are different at each level of the pyramid, but each level integrates its lower neighbor. This integration is encompassed in the relationship between the languages used at successive levels: if a description language at level L is used as a new command language at level L+1, then the learner is compelled to use explicitly the concepts that have been reified at level L.This is what we called the *language shift* mechanism (Dillenbourg, 1992): when she receives a new command language, the learner must explicitly use the concepts that were implicit in her behavior. The meaning of the new commands has been induced at the previous level by associating the learner's behavior with some representation. This representation is now the new command.

The process by which properties that are implicit at some level of knowledge can be abstracted and explicitly reached at the higher level has been studied under the label of *reflected abstraction* (Piaget, 1971). The language shift mechanism has two uses. Firstly, it translates this psychological concept in a terminology more relevant for ILE designers. Secondly, it describes a pedagogical strategy (mainly inductive) to trigger reflected abstraction. By applying the framework to ILE design, we not only ground the structure of learning environments in a model of cognitive development. But such models of development can be *tested* through the difficult process of implementation. We found that this intermediate framework can be used to "interface" several theoretical backgrounds. Most psychological theories address actually only a specific facet of learning while an ILE designer must consider learning in its globality and complexity. Therefore, an intermediate framework should integrate multiple theoretical bodies of knowledge, each relevant for some aspect of reality. An educational computing system must account for the importance of discovery, for the role of practice and for the effect of coaching, because all of them occur at some stage of learning in the real world. The framework we propose can be read from different theoretical perspectives.

From Campbell and Bickhard's (1986) viewpoint, the language shift mechanism can be viewed as a process of inducing *interaction patterns*. An elementary interaction associates some sequence of user's actions and the computer's description of this sequence. Inferring the meaning of the description language can indeed be described as the result of inducing the relationship between the actions performed and their representation (Dillenbourg, 1992). This corresponds to a view of knowledge as something that stands in the interaction between the subject and her environment. It creates a bridge between our model and current research on *situated learning* (Brown,1990), a "hot" issue in AI and Education.

Our intermediate framework also introduces the designer to the theories of Vygotsky. The *apprenticeship* idea is reified in the pyramid model by sharing control between the coach and the learner: when the learner is able to perform at some level L, the tutor must guide her activities at level L+1. This level L+1 corresponds to the concept of zone of proximal development (Vygotsky, 1978). At each language shift, the learner will assume a more important control of his solution process and the coach's guidance will be reduced.

At level one, the novice sequences experimental events on the lab workbench. At the end of level 1, the learner receives challenges that induce the need for comparisons.The concept of sequence is reified at level one and used at level two as the way of designing new experiments. At the end of the second level, the necessity of taking into account the interaction of effects is induced by new challenges.The concept of plan is presented at level two and used for designing experiments at level 3.

The shift from one level to another, i.e. to shift from one language to another corresponds to some *qualitative* jump in learning. Within each level, we defined four sub-levels that are discriminated by *quantitative* differences. These differences result form an increase in the difficulty of the challenges proposed by the coach. More complex challenges compel the learner to handle a larger number of dimensions and hence increase the working memory load. At the end of the second sub-level, the learner receives challenges that already belong to the next level. This shows the learner the necessity to have more powerful control structures to solve the proposed challenge (As in Case theory sub-level i.4 is equivalent to sub-level i+1.0).The "reunitarisation" of the objects used at some level in a new more powerful object frees the memory resources necessary to solve the problem.

3.2 The Case Based Simulation in Memolab

The Memory Machine (MM) represents an example of an additional module that can be added to the ETOILE system. The input of the simulation in MEMOLAB is the experiment that the learner has built in the LAB. Pseudo-subjects are supposed to memorize words or other items and later to remember them. The output of the simulation is the list of words that each "subject" has remembered. MM's design choices are related to its pedagogical function and to the nature of available knowledge.

The simulation has to produce results similar to the ones that one would obtain if the experiment was performed on real subjects. However, psychological experiments have de natura a low fidelity. A simulation is clearly not a substitution for real experimentation. Therefore, the validity of simulated results must be assessed in the light of pedagogical goals: to acquire basic skills in methodology of experimentation. If the learner builds an experimental plan with a factor F and based on a paradigm P, and if the literature includes knowledge on the effects of factor F, within the paradigm P, then the simulation must produce data in which the effects F can be identified. More than being precise, the results must be explainable. Several ILE designers (White and Frederiksen, 1988; Roschelle, 1988) have shown that a qualitative simulation, although less efficient, is better suited to pedagogical purposes. Simulating an experiment involves an analysis of the structure and content of this experiment. The simulation produces a trace that can be shown to the learner. It includes hypertext links to which point to abstracts of the concerned literature.

Human mnemonic behavior cannot be predicted by a set of formulas or rules. Available theories do not constitute a consistent and exhaustive body of knowledge and the literature does not cover the very large space of experiments that can be designed in MEMOLAB. Within the space of possible experiments, psychologists have concentrated their work on some avenues, or paradigms. "Knowledge" in this domain of psychology is distributed among a large set of experiments. Therefore, learners' experiments will be simulated by comparison to a similar experiment from the literature. This process is implemented with cased-based reasoning (CBR) techniques (Riesbeck and Schank, 1989).

The comparison of experiments within a paradigm brings more information than comparing experiments which differ in too many ways. Therefore, the case-retrieval process is not a simple search through a flat set of cases, with some similarity metrics. The set of cases (literature experiments) is partitioned into paradigms and a main stage during the simulation identifies the relevant paradigm. Another peculiarity of this domain is that the adaptation of the retrieved case to the target case cannot be driven by universal rules. Let us imagine that the learner builds an experiment where subjects have to memorize a long list of words and that MM retrieves an experiment which is similar on all points but the list length. The effect of list length is not universal, but varies according to other factors such as the delay between the memorizing and recall events. Therefore, case-adaptation is performed by a set of rules (named Vertical Adapters) that differ for each paradigm. In short, case-adaptation in MM is governed by rules that are themselves case-dependent.

The case library includes a paradigm discrimination tree and a set of experimental sequences (cases). Each leaf of the paradigm tree corresponds to a subset of sequences found in the literature. Once we identify the leaf node corresponding to the learner's sequence, we retrieve the experiment which is the most similar to that of the learner. The vertical adapters (VA) are stored at different levels of the tree, according to their generality. Indeed a set of VA forms a simple production system.

MM must reason at the same time about the content and the structure of experiments. The content of the experiment refers to the sequences composing the experiment: what are the tasks, the material features, etc. This information is necessary to determine the paradigm to which the learner's experiment belongs. The structure of an experiment refers to the relationship between sequences. The fact that, for instance, two experimental sequences are identical on all points but the material length indicates that the learner intends to observe the effect of the material length. This 'factor' must be identified in order to respect the pedagogical constraints. The structural analysis returns the factors identified within the learner's plan and the value (modality)of this factor in each sequence. This knowledge feeds the paradigmatic analysis process which identifies the most specific paradigm (leaf node) corresponding to the learner's sequence.