

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).