Software Agents for Cooperative Learning

Guy A. Boy

A crucial issue for the integration of new information technology (IT) in the education system is the enhancement of the access to knowledge and culture in order for the education system to improve its role of knowledge transfer and citizen training. It can be used for at least three reasons:

- to develop autonomy and individual learning;
- to remove barriers caused by geographical isolation;
- to open the education system to the external world and facilitate synergy with local resources.

In this chapter, it is assumed that IT is typically supported by CD-Roms or INTERNET. This project fits with the current Cognitive Science initiative of the French Ministry of Research. A crucial chapter of this initiative is devoted to cultural technologies. The main issues that are currently raised are the following: Would the computer-mediated information technology give birth to a new world? What would this world be like? What would be the role of human beings in this world? It often happens that humans are the victims of new information technology because they do not assimilate or integrate it in the right way, and/or at the right time. The use of the new IT leads to the creation of new artifacts enabling the management of knowledge. It seems that implicit behavior characterizing traditional culture is evolving towards explicit behavior with the use of new IT. A major issue is the extension of the human memory to
external memories. Computer technology enables knowledge management and storage. New concepts such as corporate memory or organizational memory are emerging. The education system is certainly a good example of a generator of corporate knowledge that is reused for the benefit of students.

This paper introduces a concept of educational memory (EM), i.e., corporate memory (CM) for the education system. CM work currently developed at EURISCO is multidisciplinary and multidomain, even if it is currently focused on the construction of CM concepts for the aeronautical industry (Attipoe & Boy, 1995; Boy, 1995; Durstewitz, 1994). In many ways, CM problems encountered in the industry domain are very similar to the CM problems encountered in the education domain. CM is also related to the development of information highways. Information highways will enable massive information transfer. But they do not solve the major problem of existence or availability of the right information at the right time in the right place, and in the right understandable format. In this perspective, we propose the use of software agents as intelligent assistant systems (Boy, 1991b) that would facilitate human-computer interaction, as well as human-human interaction through new IT.

The agent-orientation of human-machine interaction is not new. Autopilots have been commonly used since the 1930’s. Such agents perform tasks that human pilots usually perform, such as following a flight track, maintaining an altitude, etc. Transferring such tasks to the machine modifies the original task of the human operator. The job of the human operator evolves from a manipulation task (usually involving sensory-motor skills) to a supervisory task (involving cognitive processing and situation awareness skills) (Sheridan, 1992). Creating software agents involves new cooperation and coordination processes that were not explicitly obvious before. Agents are taken in the sense of Minsky’s terminology (Minsky, 1985).

We would like to convince the reader that the software agent technology enables the understanding and learning of various kinds of concepts since they involve active behavior of the users. They enable users to center their interactions at the content level (semantics) partially removing syntactic difficulties. They also enable to index (contextualize) content to specific situations that they understand better (pragmatism).

Section 2 presents our view on computer-supported cooperative learning (CSCL). The evolution of learning technology shows that we are heading towards the construction of pedagogical tools that enhance current educational materials. A specific case of cooperative learning in physics is given. Section 3 focuses on the requirements for an educational environment based on the construction and exchange of active documents. Examples of software agents for cooperative learning are provided in section 4. An application is developed in section 5. The balance of the paper is devoted to a discussion on and the perspectives of the current research effort.
1. Computer-supported cooperative learning

1.1. Cooperation via electronic documents

Using new IT involves the exchange of electronic documents of various types. These documents can be ephemeral, draft, or edited. An ephemeral document is a document providing useful information that can be forgotten after it has been used. It is usually called a "message" in the electronic mail language. In addition, sending a few lines using electronic mail has become acceptable, which is not the current way of doing things using a standard letter format. The fact that an electronic mail message can be very precise but informal facilitates communication between people. A draft document is a working document that will be annotated and modified. It is usually exchanged back and forth between people until it becomes satisfactory. An edited document is a crisp, completed and non-modifiable document.

Documents can be considered as mediating tools that support exchanges of viewpoints and concepts. Several viewpoints can be merged into a concept when a common agreement has been reached. They can be left as they are otherwise. In practice, documents are organized into chunks, such as paragraphs, sections, chapters, etc. These chunks include consistent viewpoints or concepts. In any case, each viewpoint should be contextualized, e.g., it should be labeled by the name of the author of the viewpoint, his/her main interests, etc. Context is usually related to other entities such as situation, behavior, point of view, relationships among agents, discourse, dialogue, etc.

Cooperation via electronic documents is a new activity that will involve new tools. These tools are information-intensive. They are called software agents. Traditional writing and reading becomes human-computer interaction. Furthermore, we claim that introducing such artificial assistants into the education system in effect will contribute to changing the relations between the various actors.

Sometimes documents are designed in such a way that they end up being complex because either technology does not allow them to be simpler, or designers do not have enough human factors knowledge and training to design for simplicity. Software agents should reduce the complexity to improve situation awareness, understanding, and performance. For this reason, we should keep software agents as small applications (Rappaport, 1995). Software agents assist users to design, produce, manage, access, choose, and interpret documents.

We make a distinction between a shallow and a deep interface. A shallow interface includes the necessary devices and displays (handled using metaphors) to enable a user-friendly human-machine interaction. Metaphors are designed to improve the interface transparency. A deep interface consists of internal mechanisms (see section 2.3). These internal mechanisms help users in the management of complexity of the tasks and the system itself.
For instance, in the University of Michigan Digital Library, Birmingham et al. (1994) developed a comprehensive agent-based architecture. They focused on the construction of particular agents and protocols: (1) user-interface agents including an interviewing agent; (2) supporting query-processing agents incorporating linguistic retrieval and providing information integration; (3) mediators to better allocate resources; (4) ontologies and protocols to federate any collection of independently generated information sources in a common language; (5) collection-interface agents maintaining the link between autonomous data repositories and the rest of the system.

1.2. The evolution of the learning technology

The initial learning technology focused on individualized instruction, i.e., standalone tutoring. The current view has evolved to the point where training and education must support inquiry-based learning, collaboration and learning as it is integrated into the doing and using. What does it mean to learn? What is a learning environment? This section describes the evolution of the learning technology from conventional computer-based training to cooperative learning.

Computer-based training (CBT) concerns training where students and teachers use computers to improve conventional training. Each instruction method is based on a model (Boy, 1993). This model involves knowledge that needs to be learned, the student and the way knowledge will be conveyed to the student. Thus, there are at least three major issues in CBT: (1) knowledge representation and elicitation; (2) student modeling; (3) computer-student interface. Domain knowledge representation can be more or less formal according to its degree of complexity. It is important to capture student knowledge in order to improve training (the feedback issue). In addition, student background and personality need to be taken into account. The computer-student interface should include both domain and student knowledge.

Intelligent tutoring systems (ITS) have been studied for almost two decades. They involve CBT and include several human-like features in their software. An ITS has explicit models of tutoring and domain knowledge. It is more flexible in its system's response. The major problem in the ITS approach is not to improve intrinsic knowledge of the computer. This is due to the philosophy of the industrial age where the current model supports the fact that learning is knowledge transfer. This model does not work today because we need to be change-tolerant, as the world changes every day. In the information age, we need to go from facts to process, and from isolation to cooperation (Soloway, 1993).

Interactive learning systems should enable the student to manipulate cognitive artifacts (Norman, 1992) from several perspectives or viewpoints. Viewpoints can be shallow (interface level) or deep (interaction level). For instance, an airplane artifact can be seen from several viewpoints: a picture or a text explaining how it should be used (user viewpoint); the way it is built (engineering viewpoint); or how much it costs (financial viewpoint).

Cooperative learning systems provide students with access to other people's ideas and concepts (SIGCUE, 1992). They make it possible to
By using cooperative learning systems, we drastically depart from the usual one-directional way of learning. Students are placed in a dynamic environment where they can express their own viewpoints, and incrementally adapt initial viewpoints to more informed and mastered concepts. In addition, cooperative learning systems are mediating tools enhancing cooperation between students, teachers, parents and other people involved in the education system. In this paper, we define computer-supported cooperative learning (CSCL) as an environment where knowledge is exchanged via electronic documents.

Learning is an active and constructive social process. An essential aspect of knowledge is that it is contextualized. This is the reason why knowledge is so difficult to acquire and represent. It is vivid. The paradox is that when we think that we have formalized it (e.g., written on paper), it is already deactivated! We have to recontextualize it to adapt it to a new situation and make it vivid again. Contextualization can be seen as indexing in the sense of connecting chunks of knowledge. The contextualization process is facilitated when people learn by doing. It follows that learning technology needs to be highly interactive.

1.3. A specific case of cooperative learning in physics

It is mandatory to perform a careful analysis of the current practice and the available experience in the use of the learning technology to provide the teachers with the appropriate tools. The Toulouse Board of Education is currently working on a cooperative distant learning system in physics. The aim of this experimentation is to help pedagogical teams to cooperatively design physics lab exercises according to global educational directives, obtaining pedagogical consistency between schools, and training teachers and students to use modern communication tools.

The current project involves 72 physics teachers from 28 schools distributed in 14 sites. Each site has a coordinator. At least three sites work on the same topic. Each site produces a set of pedagogical sheets that include the directives and necessary materials for lab experiments. Site coordinators meet every month to examine and improve the various productions before testing them in the classroom. Each site is equipped with an interactive workstation used for the design of the pedagogical sheets cooperatively with two other sites. This cooperative work leads to the construction of a pedagogical sheet database that is accessible to the entire education community.

This experimentation is very well received by the teachers. A conscious effort has been made to keep this experimentation within a reasonable framework at least in the initial stage. The volume and the quality of the productions show the real interest of the various participants. Teachers who do not participate in this experimentation ask for the productions in order to use them in their classroom. EURISCO currently develops a CSCL environment called ACTIDOC using an approach of cooperation via active documents. The next two sections of this paper present such an environment based on the construction and evaluation of software agents.
2. Towards an environment based on active documents

It is common sense to say that "it is much better to get someone who knows than a hundred people who search". From this perspective, the concept of knowledgeable agent is essential. Knowledge acquisition is usually handled through books and human-human interactions. On one hand, books require a reader to infer vivid knowledge from the texts. On the other hand, even when knowledgeable people are available at the right time in the right place (this is extremely rare), they are not necessarily able to explain what they know because knowledge elicitation is an extremely difficult process.

2.1. Requirements for active documents

Software agents for cooperative learning are designed to transform electronic document basic content into active documents. They convey vivid behavior to static documents. This section provides four requirements for active documents.

Documents have been used for ages to transfer information and knowledge from person to person. These documents have various forms such as text or graphics. But a book without a reader is just ink on paper. The active part of a text is the reader. Sometimes, the reader experiences difficulties in understanding what the writer wanted to convey through the text. Instead of mobilizing user cognition on basic interaction problems, the larger part of user's cognitive activity should be focused on learning the content of documents. Thus, an active document includes artifacts that provide the appropriate illusion useful and natural to understand its content (first requirement).

Even if we claim that a natural interaction tends to improve information access, the shallow user interface level is not enough to qualify an active document. In particular, intelligent indexing, such as providing good metaphors supported by appropriate search mechanisms, improves human-machine interaction. Indexing is used in the sense of connecting a document with other documents that are likely to enhance understanding. Current hypertext technology (Nielsen, 1990) enables the user to navigate in a document space from the consultation of an initial document. This capacity not only extends the user's memory, but it is a suggestive facility that enhances associative learning. Thus, an active document has appropriate indexing mechanisms (second requirement).

Document user-friendliness and indexing facilitate exploration. In addition, the agent-orientation provides a framework for automatic adaptivity to users. In the Computer Integrated Documentation (CID) project, an adaptive indexing software agent incrementally updates the context of the information retrieval. The notion of context has been used to tailor documentation to users' information requirements (Boy, 1991a). Thus, an active document is adaptive to users (third requirement).

Finally, it often requires cognitive efforts to figure out dynamic aspects of complex matters when they are available on a static medium such as paper. Computer programs enable the simulation of dynamic features that paper technology cannot provide. Computer simulations enable better
understanding of such dynamic aspects. Thus, an active document enables dynamic simulations (fourth requirement).

2.2. Integrating software agents into active documents

As we have already advocated, one way to avoid extra training is to produce documents that can be naturally used by people. Direct manipulation is likely to improve the design and use of (active) documents. A user-centered answer to facilitate the integration of CSCL environments is to satisfy conditions such as consistency of knowledge, internal consistency of the system that insures human reliability, context-sensitivity to the task, expert advice when it is needed, etc.

Current documents are constructed from a variety of knowledge sources. They may have various formats according to the target and the available technology. The form and content of a document are both task-dependent (context of use) and domain-dependent (content). One of the main difficulties in designing active documents is to anticipate a very large number of contexts of use. Context of use is usually related to other entities such as situation, behavior, viewpoint, relationships among agents, discourse, dialogue, etc. Contextualization is extremely difficult using the conventional paper technology. It is easier using computer technology when appropriate software agents are available or easy to construct. The CID project is an example of integration of software agents into active documents.

If active documents are understood by the user without external help, then they are self-explanatory. Complementary documents are commonly used to understand original documents. In active documents, explanation should be formalized and transferred into a software agent that will help the user to better understand. For instance, in the physics lab exercises, diagrams are presented to the students with missing parts that the students need to add in order to complete a consistent electrical circuit. On paper, these diagrams are presented to the student with a text explanation to explain what he/she needs to do. Using the computer, the same diagrams are active, so that by clicking on each part of them, hypertextual information (text or graphics) appears and explains what to do.

2.3. SRAR, blocks and software agents

Knowledge is not contextual, it is contextualized. In practice, there must be a voluntary act of contextualizing acquired knowledge, i.e., knowledge is incrementally modified to fit with contextual requirements. This act is not trivial. Context is incrementally constructed and added to current knowledge. This raises the context representation issue. What is context? In this paper, context is a set of conditions on world parameters that are persistent in a variation domain. For instance, a flight phase such as "cruise" defines persistent conditions such as "the aircraft is in the air within a prescribed flight level envelope".

The Situation Recognition and Analytical Reasoning model (SRAR) was developed to take into account the way operations procedures were used in the control of complex systems (space shuttle, aircraft, industrial process, etc.) The SRAR model has been described extensively in (Boy, 1991a), and
used in KAoS (Bradshaw, 1995). In SRAR, it is assumed that knowledge is divided into chunks that consist of a situational part (including context) and an analytical part. When a situation is recognized, analytical reasoning is performed. It is very difficult (often impossible) to elicit situational knowledge. Situational knowledge is domain-specific and incrementally acquired by training. It consists of compiled expertise at the skill level in Rasmussen's sense (Rasmussen, 1986). Analytical knowledge is elicited when we ask someone to explain what he/she knows about a domain. The underlying principle of SRAR is that situational knowledge is incrementally constructed from analytical knowledge integrated with situated explanations and observations. This learning process is performed by experimentation (Boy & Rappaport, 1987). We have shown that some analytical knowledge chunks used in a novice’s analytical reasoning were compiled into expert’s situational patterns for solving the same problem (Boy, 1987; Boy & Caminel, 1989). Furthermore, such situation patterns are active interface agents that represent local pragmatism useful in human-machine interaction.

Knowledge blocks were born from the need to model the construction of situational knowledge (Boy & Caminel, 1989; Boy, 1990; Mathé, 1990). They were designed to enable the representation of situated analytical chunks at any representational grain level. A knowledge block consists of a problem statement including a context, triggering preconditions, abnormal conditions and a goal, and a procedure (a solution to the problem statement) to reach the goal. Contextual conditions (context) are distinguished from triggering conditions because they are more persistent in time and/or space. Context (situational patterns) is incrementally acquired and refined (Boy, 1991ab). It is used to label and organize knowledge blocks.

Smith et al. already defined an agent as a persistent software entity dedicated to a specific purpose (Smith, Cypher & Spohrer, 1994). In this paper, an agent is a software entity that can be represented by a knowledge block with an interface metaphor (appearance). Even if these kinds of agents can be "generic", they must be flexible enough to be tailored to each individual. It is recognized that the most flexible way to tailor a software agent is to program it. However, programming must be natural and easy to do. If users need to learn a conventional programming language, most of them will not develop such agents. Our software agents research agenda focuses on two directions:

- the construction of internal mechanisms based on knowledge blocks that enable users to manipulate natural concepts without knowing what is going on behind the scene (what really happens is machine learning

---

1Why is the concept of “agent” useful for capturing intelligence in the interaction between human and machine? Interaction can be divided into two conceptual entities: the interface between agents and the procedures that they use to interact through this interface. Thus, we claim that intelligence is both in the interface and in the procedures. Usually, the more the interface makes the system transparent to the user (i.e., the user understands how to use the system to perform what he/she needs to do), the less he/she needs procedures (i.e., direct interaction is sufficient without any external help). In many cases however, it is hardly possible to avoid the use of procedures because of the complexity of the tasks and the system itself.
"crunching"); natural concepts manipulated by users are for instance "I like that", "this is not what I need at the moment", "this is related to another concern which is...", etc.;

- the construction of metaphors that enable users to think and play with "natural" entities such as simulation objects, consistent symbols and signs (specific to the domain), etc.; these metaphors are not necessarily human-like expressions, but rather symbolic, meaningful and consistent interface entities; they are mainly focused on the adaptation of appropriate media to various kinds of knowledge that need to be transferred for supportability purposes.

2.4. The ACTIDOC Environment

ACTIDOC (ACTIve DOCuments) is a CSCL environment. It is based on the principle that conventional documents, usually paper-based, can be improved by the addition of appropriate software agents. Currently, ACTIDOC is implemented on top of HyperCard (HC).

![Figure 1. ACTIDOC-Author typical entry.](image)

An ACTIDOC document is an ordered set of pages (typically HC cards in the current implementation). A page consists of a content and software agents that make the content vivid (or active). The content of a page is common knowledge that is shared by a community of people, e.g., teachers,

---

1According to Donald Norman, "People will be more accepting of intelligent agents if their expectations are consistent with reality. This is achieved by presenting an appropriate conceptual model—a system image—that accurately depicts the capabilities and actions." (Norman, 1994)
students, parents, etc. For instance, it comes from a textbook. Software agents are overlaid on top of the edited part of an ACTIDOC page.

An ACTIDOC agent is an active knowledge block that is composed of:
- a name;
- a context that is a list of triplets {author(s); time of creation/modification; reasons of creation/modification};
- a set of triggering conditions (TCs): a TC is usually reduced to a mouse click or a message coming from another agent;
- a set of internal mechanisms (IMs) typically implemented in HyperTalk, each IM is associated to a TC;
- a set of interface metaphors, typically an agent has an initial interface metaphor that can be modified by one of its internal mechanisms.

ACTIDOC-Author is a system that enables the creation and modification of ACTIDOC agents (Figure 1).

3. Examples of software agents for cooperative learning

A major strength of the French education system is its unity, i.e., teachers must follow an official program elaborated by the National Education Ministry. It is also a strong constraint. If it is followed to the letter, it turns out to be a page-turning activity. Such an alternative assumes that the student must absorb the information contained in each page and answer questions before going on to the next page. We believe that the assimilation and accommodation of these linear courses can be improved if appropriate pragmatism is used. In our approach, pragmatism is introduced using software agents that assist both the learner and the teacher in the assimilation and accommodation of the content. In this section, we introduce both a methodology and examples of agents that support this approach.

3.1. Schank’s learning architectures as classes of agents for the design of active documents

The main problem in building such software agents is to put the student in an appropriate learning situation that affords exploration, understanding and remembering. Each active document includes a learning situation that must satisfy the following requirements: timely information, appropriate content and appropriate presentation and support. Schank and Juna have proposed six learning architectures that we use to assist the design of software agents supporting active documents (Schank & Jona, 1991).

Case-based learning software agent. This architecture is based on two ideas. The first idea is that the expert memory includes facts. Doctors and lawyers, for instance, remember cases that were processed in the past and reuse them in analogous situations. Situation recognition is then extremely important. It is essential to recognize analogies with previous cases. The second idea, dear to Roger Schank, is that people are good story tellers. In particular, when students are ready to listen a story, they are ready to learn from this story. The task of a software agent that would use this architecture would be to motivate the student to listen to the story to encourage him/her
to look for more information, ask questions (exploration), and eventually criticize the story.

*Incidental learning software agent.* Sometimes people are not motivated to learn in specific situations. When specific knowledge is essential in everyday life, people remember it by heart, e.g., multiplication tables. They remember essential knowledge in context by recalling some of its attributes. The trick in incidental learning is to provide the student with the attributes that are sufficiently motivating for him/her. An incidental learning software agent enables the presentation of interesting situations about a concept that is not easy to learn because it is not intrinsically motivating.

*Problem solving software agent.* Problem solving is usually learned by giving the students rules to apply and situations-problems in which these rules can be useful. We then increase the complexity of a problem, and hope that students will be able to adapt. Each student must perform an exercise before being able to go to the next one. An architecture including a set of cascading problems includes a problem space where problems are related to each other by various degrees of complexity. In other words, if you cannot solve problem A, you will not be able to solve problem B, because B is a logical follow up of A. There are problems that we need to be able to solve before others. A software agent that enables the implementation of this architecture must provide a library of cascading problems, with semantic connections between them. Each problem is decomposed into sub-problems with lower granularity, thus easier to solve. This assumes a sufficiently fine analysis of the domain to elicit a hierarchy of sub-problems.

*Software agent that enables exploration directed by video databases.* The best teachers who taught us were good story tellers. Human beings use their memory to index and retrieve anecdotes at will. Video databases would be very interesting instruments that would serve as story tellers. The main problem is to index them in such a way that the student could easily browse them and retrieve useful information. The corresponding software agent helps the user to index and retrieve video images in the right context.

*Software agent that enables learning by doing based on simulations.* The simulation technique enables students to learn by exploration. It is becoming easy to implement as computer technology improves. Most microcomputers are equipped with games involving learning by doing and trial-and-error. Such simulation constitutes an environment in which students do not fear making errors. Rapid prototypes of simulation could help in the development of global skills. Heavier simulators are often necessary to learn fine grain concepts and situations. This type of software agent is more complex because it must be integrated into the simulation.

*Suggestive questions software agent.* This technique is based on the fact that everyone can be his/her own teacher. We often have a good idea, a problem or just a question that we would like to discuss. A suggestive questions software agent is a knowledge-based system that helps in learning from interaction by listening to the student and asking good questions. It must be able to ask a student questions about one of his/her problems or ideas. These questions are often questions that the student would not think of, even if he/she knows the topic very well. This mechanism enables him/her to learn from his/her own ideas. A software agent that would ask
such questions would be useful even if it does not understand the student's ideas.

3.2. Evaluation software agent (student level)
An evaluation software agent manages the strengths and weaknesses of the student. This software agent is a knowledge-based system. An expert instructor very rarely discovers many new cases that he has not experienced before. There are observable patterns of student behavior (i.e., his/her strengths and weaknesses) that the instructor is able to recognize. Recognition of such patterns leads to prelearned appropriate answers. This representation allows the implementation of production rules that a computer expert system can run. Such a computer evaluation mechanism would allow the instructor to focus more on one of his/her main tasks that is to answer more appropriately student needs. Thus, students would benefit from this human feedback.

The construction of the evaluation software agent that assists the instructor in the evaluation task should improve the relevance and completeness of the feedback. Students would perceive and correct their flaws faster. Performance would be much improved. The evaluation software agent should enable the analysis of student needs and further improve instructor acts. It should make it possible to focus on the students' most important strengths and weaknesses.

The evaluation software agent would allow to find classes of students according to their strengths and weaknesses. This classification capability would enhance better use of scarce pedagogical resources. The efficiency of training would be improved.

3.3. Instructor aid software agent
An instructor aid software agent should satisfy the following requirements:
- reducing instructor workload of tedious, repetitive and time-consuming tasks such as preparing lectures, correcting exercises; it should be connected to office automation software agents that are extremely relevant here; video hardware should be directly accessible from the computer;
- facilitating understanding and learning from his/her own experience; the instructor may consult a database including his/her own teaching strengths and weaknesses; he/she would improve his/her courses by considering the students' overall performance; this would be the same as transforming training into a self-correction activity; this feedback system would enable him/her to incrementally improve his/her courses by experimentation;
- offering access to other experiences; today the exchange of experience between instructors is very limited, due to weak exchange support; an instructor aid software agent should improve this support and provide an appropriate framework facilitating human-human interactions.

Thus an instructor aid software agent facilitates course management including course authoring, storing and retrieving experience, and accessing the experience of other instructors.
3.4. General purpose networking software agent

Concepts that we have already developed would not be applied if the psychological and social reality were not taken into account. A danger would be to implement these software agents in a centralized system. The overall system should be decentralized and interconnected. The last thing an instructor would like is to be spied by a system. Conversely, if he/she perceives the benefits of the cooperation with other colleagues (including coordinators), he/she would certainly interact with equals within an interconnected system. Electronic mail systems are good examples of such interconnected communication systems.

This leads to the definition of a general purpose networking software agent that would enable anyone in a domain-specific network to benefit from the experience of others. A general purpose networking software agent used by a coordinator would provide generalizations of experience for the overall training network. It is a knowledge sharing tool for the overall training system in the application domain. It enables the management of cases that are incrementally stored when they are discovered by educators or instructors. It enables access to evaluation records from any location in the network at any time. It enable to compare various performances. Local practices would be kept, but the definition of a common ontology would improve the understanding of student backgrounds and capabilities.

4. Developing an example

In the following example, software agents are added to existing documents to enhance their usability. Software agents provide pragmatism to the existing documents where syntax and semantics is already fixed and will not be modified. This feature corresponds to the French unified school program. Even if this approach fits well with the French education system, we think that the separation of semantics and pragmatism is a general and useful concept for the design of active documents.

4.1. An example in physics

Let us take an example of a formal course on electrical tension. In this example, we show how a conventional physics exercise can be transformed into an active document by the addition of appropriate software agents. A conventional page describing the notion of potential difference or tension follows (Figure 2).

Teachers may add appropriate agents such as denotation agents that show relevant parts of graphics explained in the text. These agents associate text descriptions to corresponding graphical objects, and conversely. For instance, by dragging the mouse on the sentence "We observe a river water current", the denotation agent shows the relevant part (Figure 3).
NOTION OF POTENTIAL DIFFERENCE OR TENSION

We observe a river water current. The altitude difference between two points of the river causes the existence of a water current between these two points.

In the same way, we observe an electrical current in a closed circuit. The potential difference between two points of the circuit causes the existence of an electrical current between these two points.

This analogy is displayed in the following figure:

![Diagram of water and electrical currents](image1)

Figure 2. Basic pedagogical document.

NOTION OF POTENTIAL DIFFERENCE OR TENSION

We observe a river water current. The altitude difference between two points of the river causes the existence of a water current between these two points.

In the same way, we observe an electrical current in a closed circuit. The potential difference between two points of the circuit causes the existence of an electrical current between these two points.

This analogy is displayed in the following figure:

![Diagram of water and electrical currents](image2)

Figure 3. Use of a denotation agent.
In the same way, a definition agent can be programmed to establish a correspondence between a text description and a mathematical formula. When the text "altitude difference between two points" is activated by putting the mouse on top of it, a mathematical formula appears in context. The context is defined by the corresponding picture and the denotation of the two points A and B (Figure 4).

**NOTION OF POTENTIAL DIFFERENCE OR TENSION**

We observe a river water current. The altitude difference between two points of the river causes the existence of a water current between these two points.

In the same way, we observe an electrical current in a closed circuit. The potential difference between two points of the circuit causes the existence of an electrical current between these two points.

This analogy is displayed in the following figure:

![Figure 4. Use of a definition agent.](image)

An analogy agent gives the equivalence between various entities such as VA and ZA. By dragging the mouse on top of VA, the altitude ZA is highlighted and shows the analogy (Figure 5).

These are very simple software agents that enhance the pragmatics of already designed physics courses. In this particular case, agents are basically hypermedia links between objects. Objects can be overlaid on top of graphical or textual parts of a conventional document to create active documents. There is a toolbox of agent types that the teacher can choose to program his own agents by analogy. Agent types can be: denotation, definition, analogy, suggestive question, problem solving (decomposition of a problem into sub-problems), video management, evaluation, hypermedia link to another active document, etc. Once, the teacher has chosen an agent type in the tool box, a procedure helps him/her to design the corresponding agent by clicking on appropriate objects or locations on the screen.

When in use, both students and teachers browse at their own speed active documents related to the lesson of the day. Individual backtracking is possible and encouraged. Eventually new agents can be created to enhance...
understanding of the concept to be learned. Students practice exercises by solving problems presented in active document exercise. In these documents, problem statements are put in context using agents in the same way as presented above. Suggestive questions guide the students. Hypermedia links to other relevant documents enable the student to remember concepts previously learned. An evaluation agent records students paths in the various active documents, as well as the answers to the questions posed. By the end of a session active documents are collected and analyzed by the teacher either on-line with the students, or off-line.

### NOTION OF POTENTIAL DIFFERENCE OR TENSION

We observe a river water current. The altitude difference between two points of the river causes the existence of a water current between these two points.

In the same way, we observe an electrical current in a closed circuit. The potential difference between two points of the circuit causes the existence of an electrical current between these two points.

This analogy is displayed in the following figure:

![Figure 5. Use of an analogical agent.](image)

#### 4.2. An educational memory in use

Typical active documents such as those described in section 4.1 are exchanged between teachers, students, parents, schools and homes. An educational memory is not a dead body of information but an actively growing accumulation of beliefs that have been put together (related or not) by people involved in the education process. These beliefs may evolve with time according to tests. An active document cannot become a stable and trustable knowledge entity until it has been adequately communicated to and approved by other people. This is a reason to enhance the educational memory interactivity both within the education system itself, and with other

---

1 A trustable knowledge entity is guarantied to work in a given context of validity. This is the case of physics formula such as Newton's law "f=ma" to measure forces at the surface of the Earth.
parties such as industry and the civil organizations. The educational memory can be seen as a large set of interconnected active documents that are logically and historically organized. This logical and historical organization is performed using contextual descriptions of the documents as described previously. It also includes a classification of software agents. This classification is incrementally acquired using a concept clustering process applied to software agents constructed by teachers. The block representation handles the construction and re-construction of such documents' organization (Boy, 1991a).

The second requirement described in section 2.1 states that active documents should have appropriate indexing mechanisms. In the CID system, we have already developed an indexing mechanism that is suitable for incrementally updating descriptors of documents and attaching context to these descriptors. A descriptor is a partial description of the document that defines a particular semantic direction of search. A document is always partially described. This is why information retrieval is an abduction process (Peirce, 1931-1958). Context is added to the descriptors within a document to reduce the uncertainty characterizing the information retrieval process. Context is usually added either positively or negatively to descriptors after successful or unsuccessful document retrievals. When a document is retrieved, it not only provides content knowledge, but also contextual information such as who designed it, why it was designed the way it is (design rationale), who used it, who did not like it (user feedback), etc.

Let us take a scenario of active document search and reuse. First, a physics teacher decides to give a course on the notion of potential difference and tension. She decides to retrieve active documents generated by other people. She makes the assumption that using the educational memory, she will get interesting active documents that she can reuse and adapt to her course. She tries to describe what she needs by specifying a list of descriptors such as "potential difference" or "tension". After a first information retrieval attempt, she gets more than 100 active documents. She does not have time to examine the whole set. She then decides to add some context to the descriptors by specifying "tenth grade" and "physics course". She then gets 7 documents that she can browse. She sees that some of the documents mention that the evaluation feedback provided by other teachers on 4 of them is not acceptable. She decides not to consider these anymore. To decide which one of the 3 remaining documents she will keep, she reads the annotations provided by other teachers and uses the documents themselves. Once she has used the selected active document, she provides feedback information on her own use of it. She may say that some children could not understand some parts of it. Thus, she has made some modifications that are included and contextualized in the current active document. The document is returned to the educational memory.

In addition, a physics teacher may design a set of software agents that he/she can send to the educational memory for experimentation. Other teachers may use them and give their feedback. We think that this is a way to converge towards a normalization of pragmatics in the teaching of physics. The main problem is for the teachers to carefully annotate the
active documents that they create, modify or use. In the current project, we try to better understand the human factors involved in the use of such an educational memory, as well as the underlying mechanisms that are required to support it.

5. Discussion and perspectives

This paper presents an approach of cooperative learning that is based on the design and use of software agents. Alain Rappaport (1988) was one of the first researchers to introduce the concept of cognitive primitives for the design of user interfaces and expert systems. In cooperative learning, cognitive primitives (basic software agents) are essential to better respond to students' needs and problems, and improve cooperative learning. Teachers should be faced with software agents that they understand directly in order to better generate active documents. Both teachers and students should be able to easily create new agents, as well as modify old ones. To facilitate active document design and publishing, libraries of software agents need to be created and maintained.

In the aeronautical domain, Airbus Training has implemented a procedure used by instructors that enables them to provide experience feedback, i.e., instructors ask for improvements or corrections of flaws in training tools based on the experience they have on these tools. Experience feedback is based on positive or negative experience that is interpreted by training specialists to generate or modify corporate knowledge. A corporate memory of the description of the various pedagogical tools is maintained using this procedure. The main point of such an organization is the optimization of the end product destined for the students.

The education system needs to better understand the notion of a product designed by a team of people for the needs of an evolving society. What do we want our children to learn? For doing what? Design rationale of educational products should be more explicit. In these conditions, teachers will be able to communicate about concrete descriptions of their pedagogical requirements by exchanging, using and refining software agents.

Standardization is a key factor to enhancing communication between people within a professional community. In the education system, the need for standardization is a trigger for appropriating new technologies, contrary to the aeronautical domain where standardization is a late, but crucial issue. In other words, it is now important to develop an ontology of the education domain that enables teachers and instructors to establish workable communication links, especially with upcoming software agents.

1A major issue is the interoperability of software developed in a specific software environment. Software agents should be platform-independent. Furthermore, the combination of object-oriented techniques (a software agent is a software object) and component-based software has some essential benefits listed by Rappaport (1995): reuse, extensibility, customization, distributability, and standardization. An example of standardization of agent-based software is given in (General Magic, 1994).
Standardization is consistent with the engineering approach. One of the drawbacks in standardization in education is the risk of the reduction of natural concepts.

In this paper, three main concepts have emerged: active documents, software agents and organization. Active documents are used as repositories of pedagogical knowledge. Software agents are observers, information processors, and proposers. They can be active entities added to conventional documents transcribed into an electronic form. Some of them observe user's interactions. They are able to produce actions from the data they have acquired from the user. The action performed by a software agent ranges from the activation of other agents to the execution of (computational) operations. They implicitly or explicitly include user models. Software agents are easy to manipulate and relate to each others. They provide vivid behavior to a user interface. They can be visible (audible), or invisible (inaudible). When they are sensorial they have a presentation shape (usually called a metaphor) on the screen, or a sound, and a behavior. Otherwise, the user does not know that they exist. In the field of electronic documentation, agent adaptivity has been shown to be extremely useful in information retrieval (Boy, 1991a). In this case, software agents are knowledge-based mechanisms that enable the management of active documents.

Technology is not a panacea for education. But, it can serve the proximal cause for mobilizing folks to actions (Soloway, 1995).

By manipulating active documents, it is anticipated that the education organization will evolve. It will produce a shareable memory that can be capitalized by the corpus of the teaching profession. We claim that humans will experience several changes in their professional life, because of technology changes as well as job changes. Training is no longer only a matter of an initial learning phase, but is becoming a life-time continuous education process that is based on intelligent assistance (Boy, 1987, 1991b) or performance support. Even if initial training (including theoretical courses) enables the acquisition of conceptual frameworks, intelligent assistance based on software agents can be seen as hands-on training with the possibility of zooming into deeper knowledge. Training as intelligent assistance is intrinsically cooperative. This paper proposes several dimensions that are relevant to this cooperation factor. In particular, it is claimed that system agent-orientation is not neutral to the evolution of our information-based world. Since software agents carry various viewpoints, these viewpoints need to be documented. This requires more human-human coordination, participation and distributed decision making. This project is ongoing and further results will become available over the next few years.

Summary

This paper describes an approach to the design of software agents for cooperative learning. This approach is based on a careful analysis of current education practices, e.g., user needs and cultural constraints, bearing in mind the technological possibilities and goals. We claim that information technology (IT) should be designed to preserve a reasonable continuity with
current practice, to facilitate knowledge transfer and access, to show a good cost/benefit ratio, where cost includes financial cost as well as additional workload. In the ACTIDOC environment, active documents are generated and managed using current learning documents improved with a pragmatic layer of appropriate software agents. Such agents for cooperative learning are proposed and discussed using typical examples.

Acknowledgments

Many thanks to Jeffrey Bradshaw and Alain Rappaport for many useful and vivid exchanges on the topic. Thanks to Helen Wilson, Rob Jaspers, Martin Hollender and Said Tazi who provided astute comments towards improving the quality of this paper. Some of the views presented in this section have been developed over the last thirteen years by the author in cooperation with others. Some results from other researchers are presented because they have been judged important for this paper and the comprehension of the whole.

References


