A New Approach on Flight Training: ASIMIL

Michel Aka, Claude Frasson

A New Approach on Flight Training: ASIMIL

Michel Aka and Claude Frasson

Département d'Informatique et de Recherche Opérationnelle
Université de Montréal
C.P.6128, Succ. Centre-Ville
Montréal, Québec, Canada H3C 3J7
E-mail: akakoasm, frasson @iro.umontreal.ca

Abstract

This paper proposes a new approach to deal with flight training. We will analyze the various methods used during the implementation of the software (Virtual reality and Case Based Reasoning). This approach introduces a client/server architecture that enables the user to have automated real-time assistance in addition to normal error detection.

Résumé

Cet article propose une nouvelle approche pour l'entraînement au pilotage, Nous analyserons les différentes méthodes utilisées pour l'implantation du système basé sur la réalité virtuelle et le raisonnement à base de cas. Cette approche utilise aussi une architecture client-serveur permettant à l'usager de recevoir de l'aide en temps réel en cas de détection d'erreur.

Keywords

Virtual Reality, Case-Based Reasoning, Flight Training

Introduction

The fast evolution of computer technology along with the spread of a global network (Internet), has made distance learning a reality. We can categorize training in two main groups: theory and practice.

Flight training is the main objective of the ASIMIL (Aero user-friendly SIMulation-based distance Learning) project. ASIMIL aims at developing a tool that will train and sharpen the skills of pilots in the Aeronautical domain. By combining Virtual Reality (VR) and Case Based Reasoning (CBR) we hope to enhance the traditional training processes.

In "Functional requirements of a simulator prototype in Virtual Reality" [1], we defined the following requirements, covering most of ASIMIL's aspects:

· The Control Requirements of the flight simulator describe the Physical aspects that have to be reproduced in the virtual environment. Control requirements also define what the learner can control during the flight.

· The Display requirements of the flight simulator describe the visual manifestation of instruments and the outside world. They also emphasize the importance quality and fluidity of the simulation.

· The Basic architecture requirements are defined in order to lead the implementation of the application. This application is required to work over TCP/IP (Transfer control protocol / Internet Protocol).

In order to meet these requirements we decided to implement a client/server application. Students (also called learners) practice by completing exercises that run on the client side of the application. Teachers (also known as instructors or experts) can monitor the learners' activities on the server side of the application. The exercises consist of accomplishing multiple tasks in a three-dimensional (3D) simulation. Rules have to be respected and conditions have to be met in order to complete the exercises.

One of the most innovative features proposed in this project is real time learner assistance. During the execution of an exercise, students may commit errors by not respecting the predefined sequence. In real life, when such a situation occurs, the human flight instructor is able to detect the error. But most importantly, he is able to determine the cause of this error in order to optimize the learner's knowledge acquisition. This is a very important factor in the learning process because the student is practicing and can be corrected at the same time; taking advantage of the instructor's experience. ASIMIL tries to mimic the presence of the human instructor.

The Virtual Aeronautical Instructor (VAI) is one of the main modules in ASIMIL. It identifies and reacts to all errors committed during the execution of the exercise. VAI's implementation is based upon Case Based Reasoning (CBR) technology. As an Artificial Intelligence (AI) technique, CBR enables us to solve problems by correctly identifying them and associating them to their corresponding solutions. VAI is not only used to understand mistakes but also to prevent them. A human Instructor is able, in some cases, to anticipate a failure or identify a situation that leads to an eminent error. VAI acquires its anticipation technique by monitoring the advices given by human instructors and by associating these advices to the situations in which they were given.

Through the course of this paper we will familiarize ourselves with VR and CBR. We will then focus on the implementation technique used to achieve the goals of this project.

Combining Virtual Reality and Case-Based Reasoning

Virtual Reality

Many definitions try to define what Virtual Reality is, we will state only one that says: Virtual Reality is a discipline that enables human beings to interface with multi-dimensional environments created by computer data [2]. By the alteration of senses (vision, sound and touch), Virtual Reality mimics a real environment or creates a new one that superimposes itself on to the real world. The user of a VR application is said to be immerged in a computer generated world [15].

Case-Based Reasoning

Case Base Reasoning (CBR) is a problem-solving paradigm [3][12]. Unlike other major AI approaches, CBR is able to utilize specific knowledge of previously experienced problems/situations (cases). It can generate a solution for a new problem by relying on similar past cases, and every time CBR solves a case or fails to do so, it memorizes it. As it encounters more and more new cases, CBR is able to gain experience. It is said to be revolutionary technique because it mimics the way humans learn by using past experience [4][14].
The seeds of case based reasoning can be found in the AI works of Roger Schank on Dynamic Memory [5]. Janet Kolodner is the conceptor of the first system that can be qualified as a Case Based Reasoner [6]. The system was called CYRUS and was developed at Yale University using Schank's dynamic memory model.

Case memory is the structure where all cases of the CBR system are stored. There are two classes of Case Memory:
Dynamic memory model: here cases are organized in a hierarchical structure. Cases sharing similar properties are under the same general structure [6].
Category and exemplar model: as an alternative way to organize memory, Ray Bareiss and Bruce Porter [7][8], proposed the Category and Exemplar model. The Category and Exemplar model has a network-oriented architecture. In this architecture features are linked to Exemplars and Categories. Exemplars represent sets of cases sharing common features. In the graph representing the Case Memory, these features are linked (directly or indirectly) to the Exemplar.

We distinguish 4 main steps in a CBR problem solving situation:
· Case Retrieval: retrieve the most similar case (or cases) comparing the case to the library of past cases.
· Case Reuse: reuse the retrieved case to try to solve the current problem.
· Case Revision: revise and adapt the proposed solution if necessary.
· Case Retrain: retain the final solution as part of a new case. After determining the new case's solution, a new index entry is created for the case if needed and the new case's structure is created and added in memory.

Architecture

General Architecture

ASIMIL is a client/server application that operates over TCP/IP-compatible networks.
If we keep in mind that ASIMIL has to be a distance learning tool using simulation for training, then learner/instructor communication has to be made possible and fluidity has to be imperative during the learner use of the simulation. For these reasons we decided to implement a client/server architecture where the learner uses simulation based training to perform sequential exercises on the client shell. The exercise DB and the exercise evaluator (Analyzer) are also located on the client side of the application (see Fig.1). The presence of a Rule Based system does not affect the fluidity of the simulation since its main work only occurs at the end of the exercise and is relatively fast.

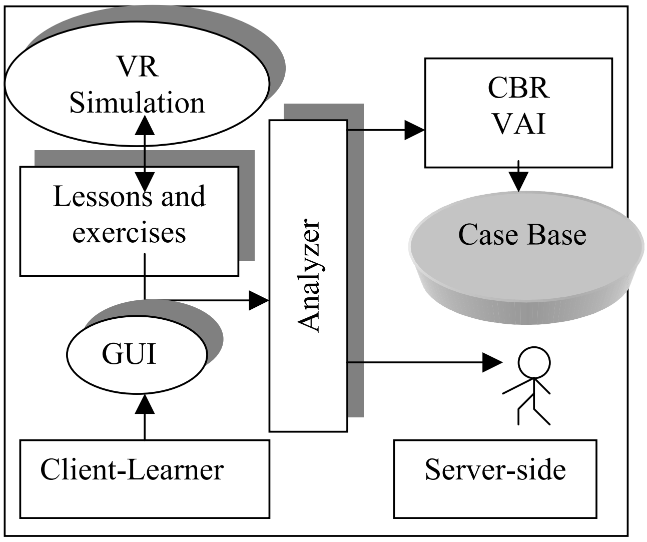


Fig. 1. Client-Server Architecture

The user gets and provides input/output information from both the Graphical User Interface (GUI) and the Simulation module. The user interfaces with these modules to input the server's address, communicate, perform actions in the simulation and review lessons.

On the server side, the expert is able to follow the student thanks to tools that we will see later. The CBR system is also located on the server for providing assistance for all connected users.

The learner can receive automatic assistance from the CBR which can eventually find the most appropriate case or receive advices from the expert who can follow online the exercises experimented by the learner. learn and practice his piloting skills. The complete systems operates over TCP/IP-compatible networks.

ASIMIL is equipped with an agent that reads out the all-incoming text messages from server. This enables the learner to have knowledge of the instructor's words without taking his eyes off the cockpit.

Implementation

The Exercise corrector: The VR module

In ASIMIL the VR module creates the virtual environment inside which the learner will undertake his exercises.
This virtual environment is created in 2 steps. First of all, we have to model all the objects to be present in the simulation. Secondly, we have to regroup these various objects into a scene.

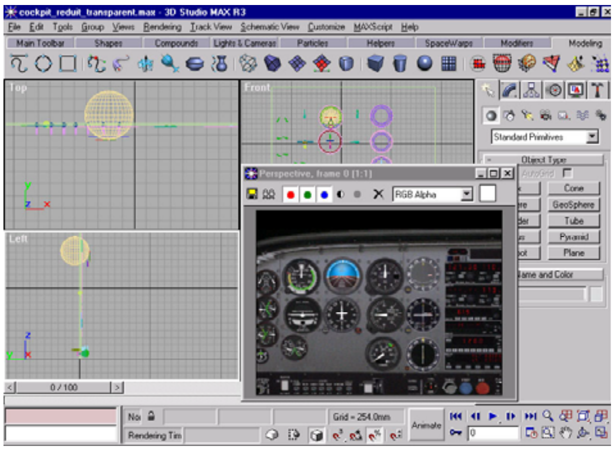


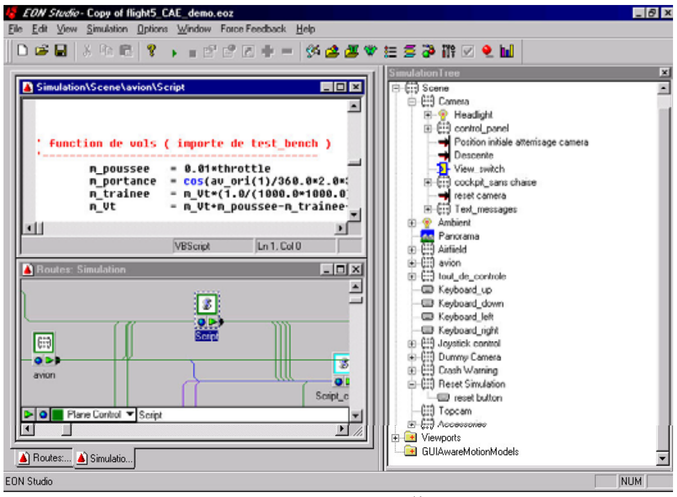
Fig. 2. Creation of objects

The creation of the objects present in the simulation (modeling) was done using 3D studio MAX (Fig. 2) which is a software that enables us to build the binary data set representing the virtual objects through a graphical interface. With the help of this software we are can modify or add various attributes such as light effect, polygon detail and texture. The landing strip, the plane, the control tower and all the objects seen in the simulation were modeled in 3D studio Max.

.

A great deal of attention given to realism and file size. In an effort to prevent lag (delay between user input and simulation response), the simulation was stripped of a lot of details. For example, high detailed textures were replaced by textures that provide acceptable level of detail. Also, objects like trees or cars that account for decoration, were not added.

The second step of the virtual world creation is the one where the objects' properties and behaviors are defined.
This step was made possible thanks to a software called Eon studio (Fig. 3).


```
' function de vols ( importe de test_bench )
'-------------------------------------------
        m_poussee   = 0.01*throttle
        m_portance  = cos(av_ori(1)/360.0*2.0*
        m_trainee   = m_Vt*(1.0/(1000.0*1000.0
        m_Vt        = m_Vt+m_poussee-m_trainee-
```
Fig. 3. EON Studio

With about 80 pre-programmed nodes, Eon gives the user the control needed over the 3D environment. These nodes allow the creator of a 3D environment to add or modify properties of the virtual world objects [13]. For example rotation, translation, transparency, brightness, sound effects and many other more are properties that can be added to the virtual objects of the scene.

The simulation itself is completely independent from the rest of the software. It can be run as a normal flight simulator, where one would fly the plane without any precise objective.

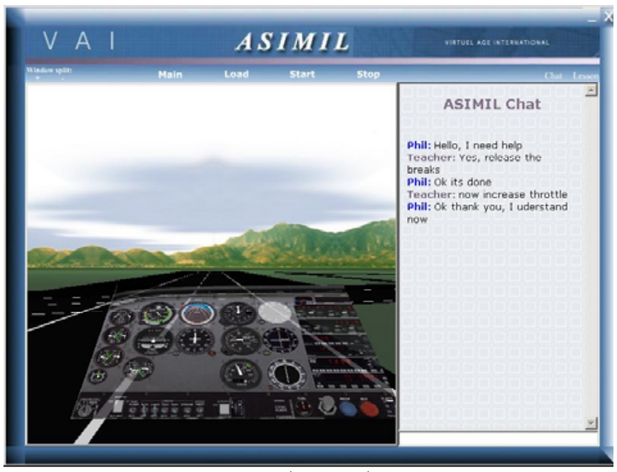
In ASIMIL, the simulation communicates with VAI by sending and receiving messages. These messages are related to: the aircraft's position, the aircraft's orientation, the aircraft's speed, the aircraft's vertical speed, the flap's positions, the state of the Brakes, the state of the Gears, throttle, wind force, wind direction and visibility.

The Flight Dynamic Model (FDM) is the set of calculations and rules that regulates the aircraft's behavior in the virtual environment (Simulation). The earlier versions of the FDM were not adequate because they tried to simulate the aircraft's behavior instead of trying to make the plane move accordingly to the rules that make up its environment. In other words we simulated the laws of physics in the virtual environment including the forces that have an impact on the plane: gravity, lift, drag and thrust.

In the current model in order for the plane to fly it has to overcome the force of gravity. Hence his lift has to be greater than that of the force that gravity permanently applied on it [11]. The computation required in ASIMIL's flight dynamic model happens in a cyclic manner. The forces are first computed to helps us determine the aircraft's ground speed and vertical speed. At the end of each cycle the new aircraft position is calculated. The cycle then repeats itself.

Two types of simulations can be distinguished in ASIMIL. They are the as passive and active simulations.

The **active simulation** is used when the learner starts an exercise and flies an aircraft in the virtual environment. The active simulation is able to capture the learner's input and modify the simulation accordingly.



Phil: Hello, I need help
Teacher: Yes, release the breaks
Phil: Ok its done
Teacher: now increase throttle
Phil: Ok thank you, I uderstand now

Fig. 4. Learning environment

One of Eon's most important features that was used during the conception of both ASIML simulation is the script. Eon script-nodes are special nodes containing programmer-defined scripts. Thanks to these script-nodes we can extend the functionalities of Eon by implementing properties or behaviors that are not offered by the default nodes. In ASIMIL script-nodes were used to define and control almost every aspect of the plane's behavior.

The simulation itself is completely independent from the rest of the software. It can be run as a normal flight simulator, where one would fly the plane without any precise objective.

Remote viewing is the process through which the expert can follow the student's actions inside the virtual environment. The remote viewing takes place on the server side of the application and is made possible through the use of a passive simulation.

The **passive simulation** is only used on the server side of the application. Once the expert activates the remote viewing, the passive simulation is loaded. The passive simulation is very similar to the active simulation except it cannot be flown; instead it receives its input over the network from the client side of ASIMIL..

Analyzer module

An exercise (Lift-off, Landing, Taxi, Straight level flight…) is a simulation that has to follow a predefined sequence in order for it to be carried out properly. A sequence is said to be an arrangement of items according to a specific set of rules, for example items arranged alphabetically, numerically, or chronologically [9].

In ASIMIL, a sequence is an arranged set of states that defines the precise chronological order that the learner's aircraft has to follow. In the following section we will take a look at the implementation of exercises, error detection, and post exercise assistance.
Definition of Current State and Concept

We define a state as a set of parameters that describe the simulation at a precise moment in time. The parameters can be divided into three categories:
Aircraft's coordinates parameters
- Aircraft's latitude and longitude
- Aircraft's altitude
- Aircraft's heading, pitch and roll

Control panel's parameters
- Aircraft's speed
- Aircraft's vertical speed
- Flaps' position
- Breaks' state
- Gears' state
- Throttle

Environmental parameters
- Wind's direction
- Wind's speed
- Visibility

If St_i is the state of the simulation at time i we can say that:

$$St_i=([P_i],[CP_i],[ENV_i]) \ .$$

Where $[P_i]$, $[CR_i]$ and $[ENV_i]$ are respectively the set of values that represent the aircraft's coordinates, control panel parameters and environmental parameters at time i. These parameters are either Booleans or numerical values. When a parameter is a numerical value, it is represented by an interval. This form of representation (allowed interval of the parameter) is only used in the sequence and not used to describe the current state of the simulation. It is useful because it helps us generalize a sequence's state (a step) as an amalgam of acceptable states.

We call transition the transformation which, when applied to a state, leads us to the following state. Given a current state St_i, we can obtain St_{i+1} by applying the required T_i transformation at time i on St_i:

$$T_i(([P_i],[CP_i],[ENV_i]))=([P_{i+1}],[CP_{i+1}],[ENV_{i+1}]) \quad . \tag{2}$$
$$T_i(St_i)=St_{i+1} \quad .$$

By analogy the exercise sequence is also the series of transformations required to go from the initial state to the final state. Each transformation corresponds to the set of user inputs that will transform a given state into the next.

A concept is the notion that is associated to a step of the sequence. Each step of the sequence is characterized by a combination of concepts. The only step of the sequence that is not associated with a concept is the last step. Most of the time, there is only one concept associated to a step of the sequence. This is because usually one notion is enough to describe the transition that takes place (Example: brakes removal or 45 degrees right turn). But occasionally a sequence's transition may require more than one concept in its definition.

The expert's decision is another reason that would explain the fact that a step is associated to more than one concept. During the exercise's conception, the expert may feel that the presence of a given step St_i is useless in that course. Therefore the expert might remove the step St_i and associates its set of concepts $\{C-i\}$ to St_{i-1}'s. St_{i-1} will then be associated to $\{C-(i-1)\}\cup\{C-i\}$.

We just saw how concepts are paired up with exercise sequence steps, we will now see how concepts are linked to lessons.

Concepts
Concepts are both linked to exercises and lessons in two databases (Concepts-Lessons and Concepts-Exercises). Notions relative to a specific concept can be present in more than one lesson.

We could directly link steps of the exercise's sequence to their corresponding lessons without the intermediate of concepts. We chose to use concepts because it helps us better categorize these relations for the expert, especially during exercise creation. It must be pointed out that the sequence-steps/concepts/lessons architecture is completely transparent to the learner. The learner is only conscious of the mistake he did and is given the correct lessons at the end of the exercise.
Error Detection and Lessons Designation

Error detection
In ASIMIL, error detection is the process through which the analyzer is able to detect the user's mistake during the execution of the exercise.

We previously saw that the steps in the sequence correspond to the different states that the simulation has to follow. We also saw that steps of the sequence can represent a set of acceptable states. Now we will see how the sequence is checked.

Let $St_{current}$ be the current state of the simulation. $St_{current}$ does not need to have intervals to describe its numerical values because the precise values are obtained from the simulation.

An error occurs when the current state of the simulation does not correspond to the current state described in the sequence. In other words, there is an error when the sequence is not respected.

During the error detection phase, the analyzer compares all the parameters of the current state with the parameters of the step of the sequence. If all the parameters match, nothing is done and we just move on to the next step. On the other hand, if two different parameters are found, the concepts associated to the current step of the sequence are noted. These concepts will find their use in the lesson designation phase.

Lesson designation
In this phase, the analyzer will determine all the lessons that the user must study according to the errors that he made during the exercise. Before the analyzer can define the list of lessons to be revised it must first get the list of all the concepts corresponding to the user's mistake. We use a function that accomplishes error detection given a state St_i and a current simulation state $St_{current}$. That function returns the set of concepts (broken concepts) that are associated to St_i.

Once we have all the broken concepts all the lessons that are linked to these concepts are added to a list. The list is forwarded to the GUI (Graphical User Interface) in order for them to be directly accessible by the user.

The Virtual Aeronautical Instructor

Error detection, error explanation and error anticipation are all done by the Virtual Aeronautical Instructor. VAI can also anticipate user-mistakes and provide the student with assistance that might prevent an exercise-threatening situation. When VAI anticipates an error it informs the expert (if any is present) by using color codes applied on the user's name. These colors indicate the importance of the mistake. VAI comes as an addition to the Analyzer module which main role is to correct and not assist.

A VAI case represents the state of the user's simulation at a given moment in time. Every case is associated with a paragraph that provides explanation and/or assistance.

Simulation update messages are sent to the server peridically by the client side of ASIMIL. When these informations are received VAI uses it to make a target case (a case to be searched inside Case Memory). It then utilizes its CBR engine to retrieve eventual error-anticipations. An error-anticipation is possible only if the expert had made a previous anticipation in a similar situation.

When the user commits an error, the state of the simulation is sent to the server along with the set of concepts that had been broken. Again VAI builds a target case and searches the Case Memory for an error corresponding to the situation, if none exists it generates a new solution.

To deal with the lag issue we use a very simple method. In message case of delay, the client side of the application still provides the student with the response from VAI. However in case of major message delay the response is simply disregarded. Non-the less the learner can carry on with his exercise since the analyzer module can determine what concepts were fouled.
VAI's Case Memory

A VAI case represents the state of the simulation at a given moment in time. VAI is able to detect and anticipate errors by comparing the current context of the simulation with similar contexts that led to errors present in the Case Memory. Every case is associated to the text that will be used to provide explanation to the learner. VAI's case Memory is the collection of all cases that are known to VAI. It is made of initial cases and the cases gained by experience. This Case Memory is organized in a Dynamic memory model pattern that has a hierarchical structure.

There are fourteen levels that constitute the hierarchy. The Case Memory has a maximum length of 14 branches. Each level uses a specific parameter for indexing (altitude, speed, heading, concepts…).

Let I_x represent the interval of possible integer values that the x^{th} parameter.
$I_x = \bigcup_{i=1}^{i=n} I_{ix}$ where I_{ix} is the i^{th} section of the interval I_x. It corresponds to the i^{th} branch of a node.

Let $P_x(C)$ be the value of the x^{th} parameter of the case C. During insertion and retrieval, in order to know the branch of the tree under which C is indexed, we look for the interval where:
$P_x(C) \in I_{ix}$.

At the x^{th} level of the case memory, C can be found under the sub-tree referenced by the i^{th} branch.

The interval division method was favored here because we were concerned about the size of case memory. If we had used the integer values of each parameter for indexing, we would have had a case memory structure too large to be implementable.

VAI's cycle is derived from the CBR cycle described by Aamodt and E. Plaza [3]. Through this cycle VAI is able to achieve problem identification, provide help to the user and gain experience. In order to find the appropriate phrases that will help the user, VAI has to retrieve the cases that resemble the most the target case. This is known as case retrieval.

Case Retrieval
Case Retrieval's importance lies on its ability to find the set of cases that are most similar or equal to the target case. Each step of the Case Based Reasoning cycle can be divided into sub steps of implementation. Case retrieval can be divided into Feature identification, Search and Select.

Feature identification: prior to performing a search in case memory, the information relative to the user's simulation are used to create a target case. Once our target case is complete the system can begin looking for it in the Case Memory.

The objective of the *Search* step is to find the set of cases that is identical to the target case. The search for the target case in Case Memory can result in success or failure. In case of failure the search returns the case that are the most similar to the target case.

During the search, VAI simply follows the tree like structure of the Case Memory and inspects the leafs in depth first order [18]. VAI goes from one level of the tree to the next by following the branch of the node that corresponds to the target case's parameter [17].

Case Reuse
The method used is very simple and depends on the number of different solutions obtained. When multiple solutions are proposed by Case Retrieval, VAI simply merges the solution and separates them by an "or".
When only one case is returned from the Case Retrieval we just copy that solution and assign it to the new case. However no new solution is generated when the returned case is the same the target case. When the Case Memory returns more that one case then the solutions are

concatenated and separated by "or". The target case is also marked as "not certified".

Case Revision
This step of VAI's cycle does not necessarily take place in real time. All the cases that are marked as "not certified" have to be reviewed by the expert. If the explanation and help do not correspond to the situation then the expert corrects the case. Cases that have recurrent complaints from user feedback are also marked as "not certified". This step helps VAI to better control its error analyzing capabilities.

Case Retain
Once a new solution has been generated for the target case VAI can add it to its case memory. The target case is added according to the indexing method that makes up the Case Memory.

Conclusion

ASIMIL is a tool combining Virtual Reality and Artificial Intelligence techniques in order to achieve distance learning. The assistance given to the user in real time and after each exercise when needed. The assistance provided in ASIMIL is mostly an automated process but it can also be given directly by the instructor thanks to the possibility of remotely viewing the learner's actions.

In future developments of ASIMIL we will add features like: user profile and we would like to improve the indexing method used by VAI. These improvements will also help in the automatic determination of the lessons that are to be taught to a user.

Acknowledgments. This work has been realized in the R&D department of Virtuel Age International. We would like to thank the company for its support [16].

References

1. Francesca De Crescenzio: Functional Requirements of a Simulator Prototype in Virtual Reality 4-9, in Proceedings for ASIMIL (2001).
2. Francesca De Crescenzio, Gouarderes G., Lefebvre P., Frasson C.: State of the art on Virtual Reality, in Proceedings for ASIMIL (2000).
3. Agnar Aamodt , Enric Plaza: Case Based Reasoning: Functional Issues, Methodological Variations, and System Approaches, in AI Communications 7-(1):39-59 (1994).
4. Anderson J. R.: The Architecture of cognition, Harvard University press, Cambridge (1983).
5. Schank R.: Dynamic memory; a theory of reminding a learning in computers and people. Cambridge University Press (1982).
6. Kolodner J.: Maintaining organization in dynamic long-term memory, in Cognitive Science Vol. 7 243-280 (1983).
7. Bareiss R.: Exemplar-Bases Knowledge Acquisition: A Unified Approach to Concept Representation, Classification and Learning. San Diego: Academic Press (1989).
8. Porter B., Bareiss R. and Holte R.: Concept learning and heuristic classification in weak theory domain. In Artificial Intelligence, vol. 45, no. 1-2, September 1990, pp229-263 (1990).
9. URL: http://www.its.bldrdoc.gov/fs-1037/dir-032/_4774.htm (1996).
10. Cognitive Systems: ReMind: Developer's Reference Manual, 220-230 Commercial St., Boston, MA 02109 (1992).
11. Shawn Blaszak,, Aaron Goldshall, George Suarez: http://library.thinkquest.org/2819/forces.htm (1996).
12. Jaczynski Michel: Etudes du Raisonnement par cas: recherche de cas similaire en utilisant des ensemble flous , Rapport de Stage de 3eme annee, DEA Informatique Universite de Nice (1993).
13. Eon Reality: Eon Reality Inc. . http://www.eonreality.com/ (2002).
14. E. L. Rissland, J. Kolodner and D. Waltz: Case based reasoning. Morgan Kaufman editor, DARPA 89: CBR workshop 1-13 (1989).
15. URL : http://www.isi.edu/isd/VET/vet.html
16. URL: http://www.virtuelage.com/
17. Berliner, H.: The B tree search algorithm: A best-first proof procedure, in B. Webber & N. Nilsson, eds, Readings in Artificial Intelligence, Morgan Kaufmann Publishers, Inc., pp. 79—87 1981.
18. Vinpin Kumar: Algorithm for constraints satisfaction: A survey, AI Magazine pp. 13(1)32-44 1992.