

Automatic Generation of Exercises for Self-testing in Adaptive E-Learning Systems: Exercises on AC Circuits

Paul CRISTEA, Rodica TUDUCE

*University "POLITEHNICA" of Bucharest, Faculty of Electrical Engineering
Spl. Independentei 313, Bucharest, Romania, pcratea@dsp.pub.ro*

Abstract. The paper presents aspects of knowledge assessment in the framework of adaptive e-learning systems, focussing on aspect related to self-testing, as a way of providing the input necessary for both system adaptation and user informed decisions. A model of up-ward propagation of evaluation credits and down-ward propagation of validation and acceptance is presented. A prototype example of automatic test generation is described in some details for the special case of electric ac circuits analysis.

Introduction

The spreading of e-learning tools usage is steadily advancing, but it is still lagging far behind expectations. E-learning does not seem to fulfil its promise to become the most important learning methodology, especially in the context of the increased role of continuous and life-long learning. This is often explained by the fact that, despite their recent impressive developments, most of the currently available e-learning tools and environments are still less appealing than the traditional face-to-face teaching methods for both students and tutors [1-2]. From the student's point of view, there are two quite opposite main reproaches: either the complaint about the "lack of human touch", the rigidity of most e-learning tools resulting in the same web pages presented to any user intending to acquire a certain item of knowledge, or the protest against the "over-coaching", the system behaving as knowing better than the user what are his/her needs. The problem of what should be part of a "learner's profile", what and how much of the learner's specific objectives, interests, preferences, and even current state of mind should be tracked as part of the concept, without rising sensitive privacy invasion issues, is still an open question. The natural answer would be to give all the control to the user, to decide how much and what type of help (s)he prefers. Still, there are cases when this approach is simply not feasible, sometimes for the mere fact that the number of parameters to set for controlling the intricacies of the behaviour of an intelligent e-learning environment is too large and full control becomes tedious and repelling in itself. From the tutor's point of view, the main drawback of current e-learning systems is that they tend to require more effort in authoring teaching materials and in preparing lessons, tests and examinations than their classical counterparts, while effectively isolating the student by hiding the whole educational environment, teacher and peers included, behind a machine. The capacity of largely re-using teaching materials, while still filtering them through the tutor personality to an extent that gives the sense of recognized authorship, the permanent synchronous and asynchronous contact among learners and between learners and teachers to a level that

helps establishing an effective cooperative learning environment seems to be the way to the answer in this case. Adequate authoring tools are needed to help the teachers in preparing both learning materials and tests for evaluating the advancement of the students towards their teaching goals. Traditional knowledge assessment is known to be demanding from both parts involved, time consuming, rising emotional aspects that do not contribute to the co-operation among the tutors and students and are prone to trigger a negative attitude from the learners. Nevertheless, a large variety of testing and evaluation tools are required by any intelligent e-learning environment to get the feedback necessary for the adaptation of the teaching system to guide the student along the learning process. Such tools operate best when they are perceived by the user as self-testing helps, giving him/her an effective support and remaining continuously under his/her own control. The learner's full cooperation makes the testing tools useful and an active part of the learning environment.

The paper briefly presents an ILE that implements the computer-supported collaborative work model, focussing on aspects referring to automatic authoring of the self tests for student evaluation the tools used to evaluate and guide the students' advancement towards their learning goals.. A prototype automatic generator of ac circuit exercise is presented in some details, stressing on the general points that could be used for test and exercises in any field of science and engineering involving quantitative analysis.

1. Intelligent e-Learning Systems

The vast majority of the currently used web-based educational systems are powerful integrated systems, like Blackboard [3] or WebCT [4] that provide a large variety of support services to both learners and teachers, but lack adaptability, belonging to the class of Learning Management Systems (LMS). Significant research and implementation effort has been dedicated to develop Intelligent Tutoring Systems [5] and Adaptive Hypermedia [6, 7], able to adapt to learner's objectives, interests, and preferences, i.e., to Learner Profile (LP). Currently, such systems offer remarkable adaptability, but only for specific tasks, lacking integration. Improved distributed architectures, centred on the concept of Intelligent Learning Environments, have been suggested to allow combining the efficiency of LMS with the flexibility of adaptive systems [8-10]. To implement the adaptivity feature, an ILE requires a quite complex structure, with several parallel version of the same learning item (LI), allowing many different learning paths to be selected in accordance with the LPs. This approach requires considerable additional effort in elaborating teaching materials, might require several authors and might need institutional support, but brings the advantage of true flexibility and adaptability. A course, as a teaching material, is no longer a flat juxtaposition of learning items, but a multilevel structure with many parallel branches, along which the ILE recommends an optimal path for a user or for a class of users. As mentioned in [10], it has been argued that authoring learning material and building the structure of adaptive systems tends to become too complicated for the average teacher. Correspondingly, portability – the ability to deploy the content of a system on any other system, and reusability – the ability to search and retrieve learning items (LIs), including lessons, modules, exercises, activities and to reuse them, become strictly necessary features for an efficient implementation of the ILE concept and for the success of the wide scale implementation of this approach. The problem has already been addressed for the case of LMS: the DOD's Sharable Content Object Reference Model (SCORM) ADL initiative [11] aims to provide a comprehensive set of e-learning capabilities enabling interoperability, accessibility and reuse of Web-based learning content. Similar efforts are under way to provide a harmonized set of guidelines, specification and standards for Intelligent Tutoring Systems (ITS) and Adaptive Hypermedia (AH) systems [12, 13]. At the

same time, significant changes in the basic methodology of teaching/learning is to be expected as result of the current advancement and convergence of cognitive psychology and computing science. Paradigms such as just-in-time learning, constructivist approaches, student-centred learning and collaborative approaches have emerged, and are being supported by technological advancements including simulations, virtual reality and multi-agents systems.

2. Methodology, Architecture and Implementation

The system is learner centered, all human and artificial agents being focused on achieving the learning-training tasks. Human agents include, aside the students, authors of teaching materials, tutors, course administrators, and system administrator(s). The pilot web oriented ILE has the server-client distributed multiagent hybrid architecture shown in Figure 1 [14].

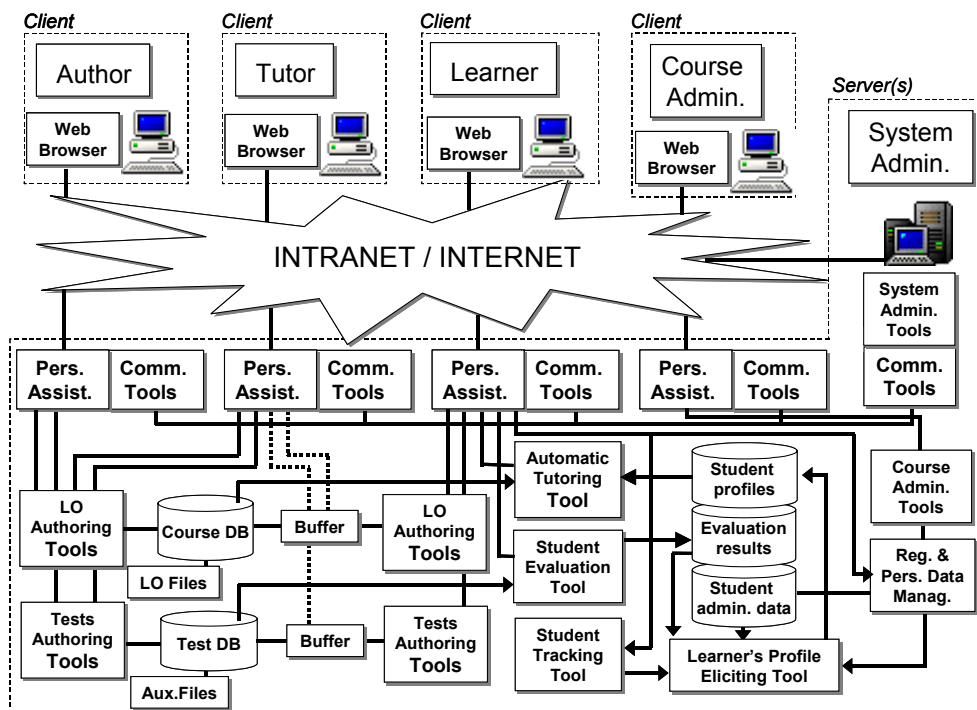


Fig. 1: Architecture of the ILE pilot

The client side requirements are kept to a minimum; each human agent accesses the system via an intranet or internet connection, by using a standard web browser. On the server side, each human agent has a personal assistant (agent) that inter-operates with the other artificial agents providing the system functionality. All personal assistants are linked to communication tools that facilitate the contact among human actors in the system, and provide support to the collaboration among the students. The learner personal assistant (LPA) controls, in the first place, the access to the agents providing the main functions – the Automatic Tutoring Tool and the Student Evaluation Tool, but also to the Learning Item (LI) Authoring Tool and Tests Authoring Tool – thus providing support to a participative learning approach. Students get credits not only for their results in the tests, but also for their involvement in the course development – by contributing with new versions to existing *LIs*, building new questions for tests addressing various *LIs*, etc. When submitted, the student contributions are stored in a buffer, waiting for tutor's validation to be introduced in the course and test databases. The contributions are marked similarly to the successful passing of regular tests. The LPA gives also access to the Registration and

Personal Data Management Tool, and helps tracking student participation and results, passing the corresponding data to the Learner's Profile Eliciting Tool (LPET). The tutor has also a Tutor Personal Assistant (TPA) that provides an interface to all system functionalities offered to a tutor: keeping track of class enrolment, following learners' progress in the training process, reading synthetic data on learners' profile, etc. Authors and tutors have access to authoring tools, to update the course, tests and any other teaching materials. Course administrators have access to all relevant data about courses and students. The system administrator monitors the usage of the resources and takes care of the smooth functioning of the system. The distribution of the functional roles among human actors can be managed from an administrator platform accessed via root privileges.

The server has been implemented as an in-house developed J2EE compatible JAVA web-application, running on the Tomcat Apache Server and using the MySQL database server, both largely accessible and available on UNIX and WINDOWS platforms. The modular approach allows an easy restructuring of the system; e.g., the change of the database affects only the module controlling the database. Database access speed is increased by using a connection pool. Data access (some contained in the MySQL database, some in a system of related directory files) is made through interactive web pages implemented with Java Server Pages (JSP) and Servlets. Each JSP is controlled by a JavaBean which handles the security tasks and data manipulation. Every JavaBean within a module extends the class corresponding to an actor (learner, author, tutor, course and system administrator), being able to control authorized access.

3. Learning Evaluation

Learning evaluation is based on test results, but also on the student involvement in proactive learning, including the contribution to the course continuous development [15]. Care is taken to stimulate students to do more than just recognize textbook material when responding to tests. Students are encouraged to formulate themselves new questions and are given credits when their contribution is included in the question data base (QDB).

To make the system closer to a classic examination that does not require the checking every item in a course, the passing of a higher level *LI* is not conditioned by the passing of all lower *LIs* it contains, but only by the

sum of points exceeding the threshold for that level. The acceptance of a higher level *LI* gives credit for all *LIs* below it in the course structure. This results in an up-wards propagation of the awarded points and a downward propagation of the acceptance, as shown in Figure 2. The acceptance of an *LI* is irreversible, so that only warnings and advices are issued if later errors signal possible weaknesses in an already assessed *LI*. The tests are built

by randomly choosing from the question data base (QDB) a specified number of questions referring to the given tag (*LI* address or keyword). A test referring to a *LI* can comprise questions attached either directly to that specific *LI*, or to any *LI* placed below it in the course tree structure. The points obtained when making a choice *C* from the set of options $O(Q)$ pertinent to question *Q* are recorded at the *LI* to which the question is attached and

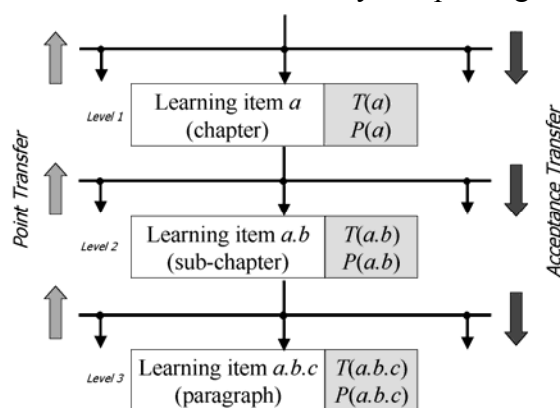


Fig. 2: Evaluation points and acceptance transfers

transferred upwards. The sum of points $P(Q)$ for a question Q results by adding the points for all options selected at Q :

$$SP(Q) = \sum_{C \in S(Q)} P(C), \quad (1)$$

where $S(Q) \subseteq O(Q)$ is the set of options the student has selected at question Q . The correct choices are awarded positive points, the wrong answers – negative points. Assigning negative points to the wrong choices contributes to discourage guessing, but the system success relies on student motivation. The points $P(Q)$ acknowledged for question Q is given by:

$$P(Q) = \begin{cases} SP(Q), & \text{if } SP(Q) < 0, \\ 0, & \text{if } 0 \leq SP(Q) < T(Q), \\ SP(Q), & \text{if } SP(Q) \geq T(Q), \end{cases} \quad (2)$$

where $T(Q)$ is the threshold required for the acceptance of the reply to Q . As a consequence, points obtained for a question are taken into account only when exceeding a certain requested minimum. Such a learning appraisal aims at a robust understanding and proper using of the tested knowledge, by discouraging superficial and fragmentary learning. Each LI is not evaluated independently, but in the context of the course, in relation with the related LIs . The sum of points $SP(LI)$ for a certain learning item LI consists not only of the sum of the points obtained for the questions Q referring directly to LI , but, also, of the sum of the acknowledged points transferred to LI from its children – the learning items LI' placed one level below it in the course structure:

$$SP(LI) = \sum_{Q \in LI} P(Q) + \sum_{LI' \in C(LI)} P(LI'), \quad (3)$$

where $C(LI)$ are the children of LI . Equation (3) ensures that the points obtained for a certain LI are transferred upwards, to all its ascendants, without double counting.

The points $P(LI)$ acknowledged for a learning item LI depend also on the passing of a certain acceptance threshold $T(LI)$, but, in this case, there is an award $A(LI)$ for the successful completion of the study of LI marked by the passing of $T(LI)$:

$$P(LI) = \begin{cases} SP(LI), & \text{if } SP(LI) < T(LI), \\ SP(LI) + A(LI), & \text{if } SP(LI) \geq T(LI). \end{cases} \quad (4)$$

The status $S(LI)$ of LI changes from 0 – *pending* to 1 – *studied*, when its threshold, or the threshold of its ascendant, have been passed:

$$S(LI) = \begin{cases} 0, & \text{if } (SP(LI) < T(LI)) \text{ and} \\ & (S(LI') = 0, LI \in C(LI')), \\ 1, & \text{if } (SP(LI) \geq T(LI)) \text{ or} \\ & (S(LI') = 1, LI \in C(LI')), \end{cases} \quad (5)$$

where $C(LI')$ are the children of LI' .

Relation (3) shows the up-propagation of the points in the tree-like course structure, while relation (5) corresponds to the down-propagation of the acquired knowledge recognition along the same structure.

4. Automatic Problem Generation

We present in this section a tool that automatically generates sets of ac electrical circuit analysis problems with given complexity. The initial window of the system asks the user, tutor or student, for the global description of the desired set of problems, comprising the global parameters (set label, the number of problems, number of nodes, number of branches, the range of values for the chord current intensities, twig and chord resistances, twig *emfs*, the number of branches with current source, and the number and type of controlled sources). For each problem in the set, the system generates a distinct topology, starting with building a tree (Fig.3 and Fig.4) and continuing by introducing the chords to generate the circuit graph in Fig. 6 and establish the corresponding essential incidence matrix $[\Lambda_{tc}]$ that describes the circuit topology. In the second part, the circuit parameters (resistances, reactances, independent and controlled sources parameters) and complex variables (complex branch currents and voltages) are chosen or computed sequentially.

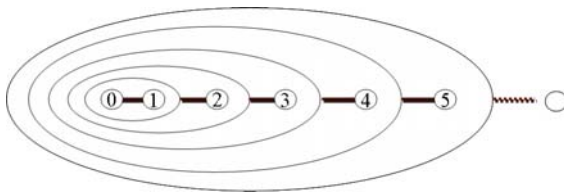


Fig.3. Tree generation: Each additional node is connected by a twig pointing towards previously linked nodes.

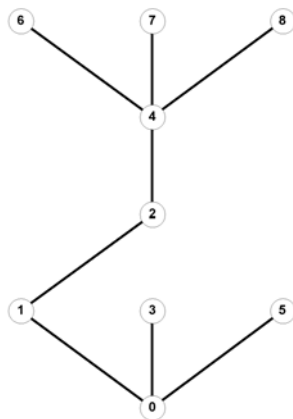


Fig. 4. Example of automatically generated tree with nine nodes

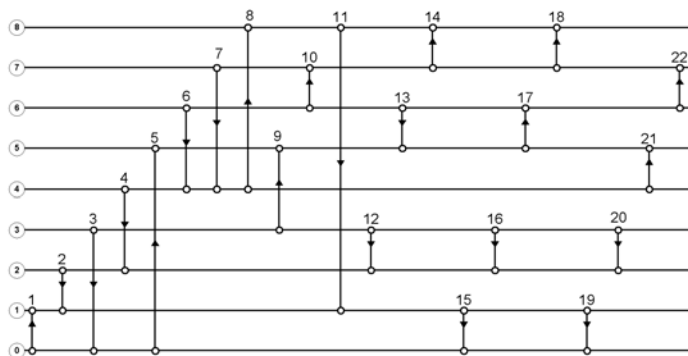


Fig. 5. Example of automatically generated network graph comprising the 8 twigs shown in Fig. 4 and 14 chords. The nine nodes labeled 0 to 8 are shown as connection bars.

The basic idea of the method for automatically generating sets of problems, without the need of solving systems of equations (or, equivalently, of inverting matrices), is to start the procedure from points that allow the simple chain computing off all needed elements. This method can be applied to any similar type of analysis problems, in which the parameters of the system are given, while the variables result as solutions of equations that describe the system.

In the following, all current, voltages, *emfs*, source currents and impedances are complex magnitudes. Square or rectangular matrices are represented by the symbol of the magnitude between square brackets, while column vectors are represented by a single square bracket to the right of the symbol. The subscript 't' corresponds to magnitudes attached to the twigs (the branches of the tree), while the subscript 'c' – to the chords (the branches of the co-tree).

The successive steps of the algorithm are:

1. Randomly choose the chord current vector I_c ;
2. Compute the twig current vector: $I_t = [\Lambda_{tc}] I_c$;
3. Randomly choose twig complex impedances (resistances and reactances) and complex emfs: $[Z_t], [Z_c], E_t$;
4. Compute the twig voltage vector: $U_t = [Z_t] I_t - E_t$;
5. Compute $U_c = [\Lambda_{tc}]^T U_t$;
6. Compute the chord complex emfs: $E_c = [Z_c] I_c - U_c$;
7. Concatenate the variables and parameters for all branches:

$$U = \begin{bmatrix} U_t \\ U_c \end{bmatrix} \quad I = \begin{bmatrix} I_t \\ I_c \end{bmatrix} \quad E = \begin{bmatrix} E_t \\ E_c \end{bmatrix} \quad J = \begin{bmatrix} J_t \\ J_c \end{bmatrix}$$

8. If independent current sources desired, partially or totally convert some of the independent voltage sources as shown in Fig. 6: $E^{new} = E - [Z] J$;

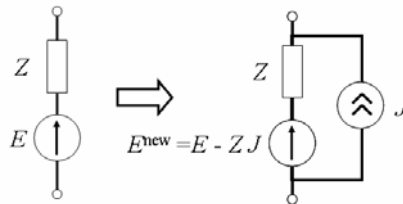


Fig. 6. Partial conversion of an independent voltage source into an independent current source

9. If cross links between branches (controlled sources and mutual inductive couplings) desired, convert some of the independent voltage sources into the selected cross links:

Controlled source (change of independent voltage source emf,s)

$$E^{controlled} = [Z_t] I \quad E^{new} = E - [Z_t] I$$

$$J^{controlled} = [Y_t] U \quad E^{new} = E - [Z][Y_t] U$$

$$E^{controlled} = [A] U \quad E^{new} = E - [A] U$$

$$J^{controlled} = [B] I \quad E^{new} = E - [Z][B] I$$

Mutual reactances

$$E^{induced} = [-jX_M] I \quad E^{new} = E - E^{induced}$$

Figure 7 gives the matrix flowgraph representation of circuit equations which are used to generate circuit parameters and variables. The methodology for fast generating circuit variables and parameters implies cutting the loop in the equation flowgraph, so that no matrix inversion is required.

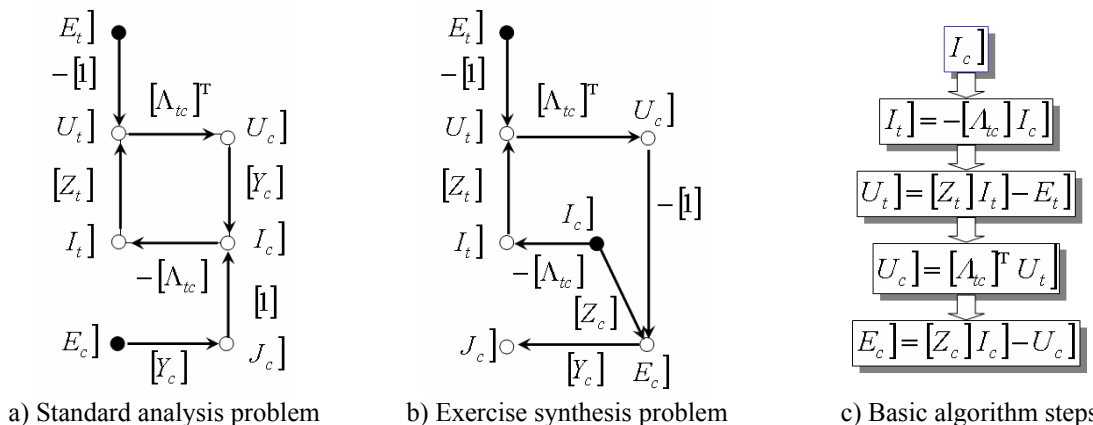


Fig.7. Methodology for converting an analysis problem (a) into a procedure for automatic generation of exercises (b). Notice the cut of the loop in the matrix flow graph resulting from the change of the initial data.

Because the whole procedure involves no division, it is possible to generate problems that use only integer values, which is a debatable advantage from the engineering training point of view, but can produce more attractive tests. A similar approach can be used for a large class of problems, by choosing variables that open the loops in the system equations flow graph.

5. Conclusions

The paper presents a pilot implementation of an Intelligent e-learning environment (*ILE*) able to adapt to the learner's profile (*LP*) and to encourage active and cooperative learning. Special attention is given to support the authoring of learning items and tests, especially important in an *ILE* context to assess the students advance towards their learning objectives. Adequate authoring tools can make the tutors task easier, contributing to a better acceptance of e-learning systems, despite the required extra work and can stimulate a pro-active attitude of students. An example of automatic problem generation is given for the case study of a.c. electric circuit analysis. A similar methodology can be applied to convert any engineering analysis problem into a procedure for automatic generation of tests. This approach has the advantage of greatly simplifying the problem of synthesizing system analysis exercises, to the point of making them available to both tutors and students, in the later case for self-testing and computational experiments purposes.

References

- [1] D. A. Norman, J.D.Spohrer, Learner-centered education, *Comm. of the ACM*, 39(4), 1996, pp. 24-27.
- [2] M. J. Wooldrige, N.R.Jennings, Intelligent agents: Theory and practice, *The Knowledge Engineering Review*, 10(2), 1995, pp. 115-152.
- [3] Blackboard Inc, Blackboard Course Management System, <http://www.blackboard.com/>.
- [4] WebCT, Course Management System, <http://www.webct.com/>.
- [5] C. Frasson, G. Gauthier (editors), *Intelligent Tutoring Systems - At the Cross-roads of Artificial Intelligence and Education*, Ablex Publishing Corporation. Norwood, New Jersey, 1990.
- [6] De Bra, P., Aerts, A., Berden, B., De Lange, B., Rousseau, B., Santic, T., Smits, D. & Stash, N., *AHA! The Adaptive Hypermedia Architecture*, *Proc. of ACM Hypertext Conference*, Nottingham, UK, August 2003, pp. 81-84.
- [7] P. Brusilovsky, Adaptive hypermedia, *User Modeling and User Adapted Interaction*, Ten Year Anniversary Issue (Alfred Kobsa, ed.) 11(1/2), 2002, pp. 87-110.
- [8] P. Brusilovsky, Student model centered architecture for intelligent learning environments, *Proc. of Fourth international conference on User Modeling*, 15-19 August, Hyannis, MA, USA. User Modeling Inc, 1994, pp. 31-36.
- [9] O. Conlan, C. Hockemeyer, V. Wade, D.Albert, D., & M. Gargan, An architecture for integrating adaptive hypermedia service with open learning environments, In *Proc. of ED-MEDIA 2002*, vol. 1 of World Conference on Educational Multimedia, Hypermedia & Telecommunications, pp. 344-350.
- [10] P. Brusilovsky, A Distributed Architecture for Adaptive and Intelligent Learning Management Systems, *Proc. of AI in Education (AIED2003)*, vol.4., Workshop Towards Intelligent Learning Management Systems, Sydney, Australia, July 2003.
- [11] Advanced Distributed Learning (ADL) Initiative, Sharable Content Object Reference Model (SCORM), <http://www.adlnet.org/>.
- [12] A.I. Cristea & A. De Mooij, Designer Adaptation in Adaptive Hypermedia Authoring, *Proc. ITCC'03*, Las Vegas, US, IEEE Computer Science, 2003, pp. 444-448.
- [13] D. Dicheva, L. Aroyo, Alexandra Cristea, Collaborative Courseware Authoring Support, *Proc WEB-CATE 2002 – Computers and Advanced Technology in Education*, Cancun, Mexico, 2002, pp. 52-57.
- [14] P. D. Cristea, Rodica Tuduce, Pilot Implementation of an Intelligent e-Learning Environment, *eChallenges e-2004*, Vienna, Austria, October 27-29, 2004.
- [15] P. Cristea, Rodica Tuduce, Test Authoring for Intelligent e-Learning Environments, *Proc. IASTED Intl. Conf., Web-Based Education*, Innsbruck, Austria, 2004, pp. 402-407.