ICEIPT PROJECT– AN INTELLIGENT COLLABORATIVE ENVIRONMENT FOR INDIVIDUALIZED PLANNED TEACHING

Irina Zheliazkova

Abstract: Since the beginning of 90's in the framework of several national and institutional projects a research group from the University of Rousse has been working on development, implementation and investigation of an intelligent collaborative environment for courseware planned teaching (ICEIPT). This paper presents a brief overview of this project focusing on its background, client-server architecture, three level ontology and language.

Keywords: domain-independent, ontology, graph, language, courseware, knowledge.

ACM Classification Keywords: Computer and Information Science Education, Knowledge Representations of Formalisms and Methods

Introduction

Analyzing the process of human teaching a given course the following collaborative teacher's activities were identified [12]: structuring the subject domain, setting the group teaching goal, creating the group plan (course curriculum), preparing the course material (*CM*), monitoring the plan execution, correcting the *CM*, plan or goal. In reality the course static plan is not fully executable mainly because: the teaching environment is dynamic, the teacher's behavior cann't be predicted, the learners have different initial knowledge and personality, the teaching goal and the current achieved results are vague and uncertain, the necessary resources (human, equipment, financial) cann't be always ensured. In this context the models known for on-line modeling of this process such as the optimization model [9], Markov chain model [4], and so on are not enough adequate.

The above-listed disadvantages can be avoided by collaborative systems for support of general plan activities such as coding team tasks, allocating dynamic tasks, creating decision points, controlling constraints, and so on. According to [6] *TOP* is the earliest system investigated team-oriented programming and provided a plan description language. The capabilities of other reviewed systems (*STEAM*, *GRATE*, *RETSINA*, and *JACK* Teams) also are limited. To encode teamwork knowledge, experiment with inferred team intelligence, and specify an adaptive long-term plan process a logic language *MALLET* with a multi-agent architecture are proposed [6].

The intelligent and adaptive systems specially designed for web-based education are another alternative for increasing the efficiency of the human teaching process [2]. Forward three relevant projects briefly are presented. *IDEAL* [7] is a teamwork environment for individual planned teaching. Its multi-agent architecture consists of a user interface module on the client side, and course, teacher, and personal agent modules on the server side. The environment is based on a domain-independent ontology, learner and group's models. But lack of a knowledge description language limits flexibility and productivity of the teaching supported by this environment.

In *TILE* [5] project the courseware author (*A*) uses a combination of directed graphs and state-charts for visual graphical representation of relationships between the learning objects generating plans with *depth-first*, *top-down*, *bottom-up* or *breath-first* strategy. The *A* can annotate and index the multimedia *CM* using a flexible structure code language similar to the natural one. By means of a flexible code query language the *CM* suitable for a given learner (*L*) is delivered using semantic search in *INTERNET*.

In *IALMS* project [3] the adaptation to a given *L* is aimed to maximize the results of his/her learning. The system architecture consists of three subsystems (tracking, individualized, and adaptive) with an interface module. The *IALMS* procedural language is offered only the *A* and mainly for describing the *CM* structural tree that is passing bottom-up to choose the set of passive and active blocks with highest importance for a new session.

The aim of *ICEIPT* is to support collaborative teaching plan activities such as coding author, teacher and learner's tasks, supporting a common task base (*TB*), setting the courseware goal, creating decision points, assessing the degree of its achieving, and so on. This paper presents an overview of this project and is organized as follows. In the second section its theoretical background is presented. The client-server architecture of *ICEIPT* is commented in the third section. The next section present domain-independent ontology and language of the environment on course, structure, and session level. The last section summarizes the project contributions.

Theoretical Background of the Project

In *ICEIPT* project the author's team interface is viewed as a semantic graph (G^{env}) more connective than a simple combination of *TB* semantic subgraph (G^{tb}), subject structural tree (G^{st}), and teamwork subgraph (G^{tw}), e.g. $G^{env} \supset G^{tb} \cup G^{st} \cup G^{tw}$. G^{tb} consists of m components, where m is the number of different cognitive types of knowledge units (*KUs*) such as testing questions, learning concepts, constructing schemes, modeling systems, and even measuring and controlling real lab objects. The i^{th} component $G^{tb}(i)$ represents a homogeneous knowledge base (*KB*), j^{th} node of which presents a semantic subgraph generated during the ontology-based constructing of j^{th} *KU* i^{th} type. This node is labeled with a vector of local pedagogical parameters $< Q_{ij}^{kb}, C_{sij}^{kb}, T_{ij}^{kb}, C_{qij}^{kb} >$, where $Q_{ij}^{kb} \geq 1$ is *KU's* knowledge volume; $0 \leq C_{sij}^{kb} \leq 1$ its degree of system prompt; T_{ij}^{kb} the time planned for its ontology-based constructing, $0 \leq C_{dij}^{kb}$ its degree of difficulty. Different formulae for different types of *KUs* are applied to the generated subprogram tree [1,10,14,15,16] for computing the first two parameters. The last two parameters are updated after performance of a task constructing by a given learner (*L*).

The nodes of the domain structural tree G^{sd} are labeled with a pair <*i,j* >, where i is the number of the *KU*'s type*;* $i-$ position code, generated during the courseware structuring and composed from the numbers of the path structural units from the root node to the initial node of the given arc. The nodes of G^{tw} represent the teamwork knowledge, e.g. structure of the teams (author's, teaching, and learner's) involved in the teaching process and tools for their communication, and the arcs between them are navigation links only. More information about G^{sd} and G^{tw} can be found in the third section.

If i is the index code of the goal node, e.g. the root node of the structural subtree, chosen by a given teacher's team, a group goal can be expressed as [13]:

$$G^{gr}(i) = \bigcup_{j=1}^{n1} GTS^{gr}(j) \cap \bigcup_{j=1}^{n2} TKA^{gr}(j) \cap \bigcup_{j=1}^{n3} LTS^{gr}(j) \cap \bigcup_{j=1}^{n4} SOI^{gr} \cap \bigcup_{j=1}^{n5} CTE(j)$$

Here each predicate has the following meaning: $GTS^{gr}(j)$ choice of one of available $n1$ teaching strategies each one setting the way of passing the subtree; $TKA^{gl}(j)$ choice of one of $n2$ combinations of available types of *KUs*; $LTS^{gl}(j)$ choice of one of $n3$ available local teaching strategies each one setting a sequence of teaching operators (p - presentation of the author's *KU*, c - constructing its ontology by the *L*, r - remedial of

the learner's misconceptions and/or missing knowledge); $SOI^{gl}(j)$ choice of one of $n4$ combinations of the options for the teacher's intervention during the ontology design by a given *L*(refusal from the task for *KU* constructing, attempt to start it again, browse of all given type *KUs*, print the active *KU*, save it in a given *L* book emulator, and so on); $GTE^{gl}(P_j^{gl} \geq (\leq) P_j^*)$ choice of one of *n5* criteria combinations of the teaching efficiency; ($R_{sk}^{gl} \geq R_{sk}^*$ - minimal global degree of the learner's knowledge, $C_{sp}^{gl} \geq C_{sp}^*$ - minimal global degree of system prompt of the author's knowledge, $C_{df}^{gl} \geq C_{df}^*$ - minimal global degree of difficulty of the knowledge, $T_{tp}^{gl} \leq T_{tp}^*$ - maximal global time of the learning process, $T_{ds}^{gl} \leq T_{ds}^*$ - maximal duration of a session, e.g. lecture/test/exercise).

The individual learner's plan generated through dynamic disassembly planning is viewed as an unconnected graph G^{ip} which nodes represent subgraphs corresponding to the ontology for constructing the j^{th} *KU* i^{th} type. Passing the subtree bottom up each subgraph is labelled with a vector of parameters <$C_{sk}^{lc}(i,j)$, $C_{sc}^{lc}(i,j)$, $C_{df}^{lc}(i,j)$ $LTS^{lc}(i,j)$, $SOI^{lc}(i,j)$, $T_{tp}^{lc}(i,j)$, $I^{lc}(i,j)$>. The first five parameters are equivalent to the group parameters, $T_{tp}^{lc}(i,j)$ means time the computed initially under the assumption for a constant learning rate during a session $T_{tp}^{lc}(i,j) = \frac{\sum_{j=1}^{n} Q^{lc}(i,j)}{Q^{gl}(i)}.T_{tp}^*$, where n is the total number of the nodes of $GTS(i)$. The last parameter $I^{lc}(i,j)$ is a flag for decision point after the current session (initially all equal to *false*).

The semantic graph modeling a given *L* interface in the course of plan execution (G^{mp}) has several nodes added to G^{ip}, associated with the decision points for searching the last decision point, continuing the session, finishing the teaching process. The formulas for dynamically computation of the current real parameters (cr) on the base of their local values (lc) are simple:

$$C_{sk}^{cr}(i,j) = [C_{sk}^{cr}(i,j-1) + C_{sk}^{lc}(i,j)]/2; C_{sp}^{cr}(i,j) = [C_{sp}^{cr}(i,j-1) + C_{sp}^{lc}(i,j)]/2$$
$$T_{tp}^{cr}(i,j) = T_{tp}^{cr}(i,j-1) + T_{tp}^{lc}(i,j); Q^{cr}(i,j) = Q^{cr}(i,j-1) + Q^{lc}(i,j)$$

After performance of task j of type i both planned and achieved results are registered in the plan execution part and presented both the *T* and *L*. If $I^{lc}(i,j) = false$ and $I^{gl}(i) = false$ the learner-computer dialogue continues with performing the next $(j+1)^{th}$ task of the current session. If $I^{lc}(i,j) = true$ and $I^{gl}(i) = false$ the computed values are registered in the plan execution part and presented to both *T* and *L*.

The learning rate for the current session is computed and the inequality $\frac{Q^{gl}(i) - Q^{cr}(i,j)}{T_{tp}^{gl}(i) - T_{tp}^{cr}(i,j)} \leq \frac{\sum_{j=J1}^{J2} \frac{Q^{lc}(i,j)}{T_{tp}^{lc}(i,j)}}{J2 - J1}$ is checked. Here $J1$ and $J2$ are respectively the number of the initial G_{J1}^{ip} and final G_{J2}^{ip} subgraph of the current session. This is compared with the predicted rate during the rest of the planned time T_{tp}^*. Satisfaction of this inequality predicts success of the individual plan, so for this plan the next session will continue with the subgraph G_{J2+1}^{ip}. If the inequality is not satisfied the preplanning is recommended decreasing $C_{sp1}^{lc}(i,j)$ or the dialogue moves to the final state and final report with given and achieved global goal is presented all members of the team .

Multi-Agent Architecture

The architecture of the proposed environment (Fig.1) is a development of a well-known idea for dynamic planning/executing the teaching process [8]. A web-based single agent architecture of this idea has been proposed in [11]. The authoring environment of *ICEIPT* supports his/her work a long-term plan activities on the courseware level. The *A* works as a knowledge analyst and engineer making visible the course curriculum and conceptual structure of the taught domain.

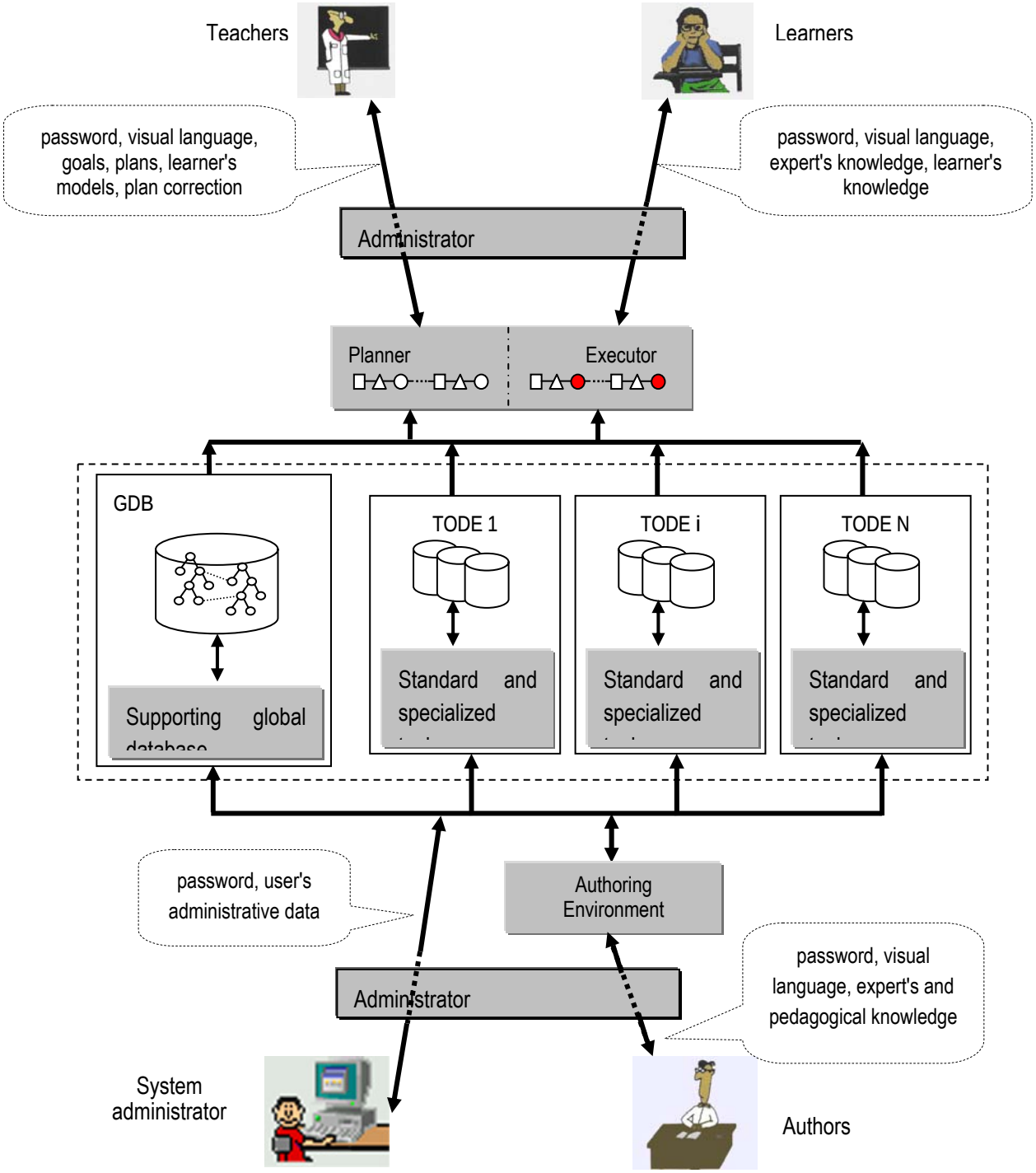
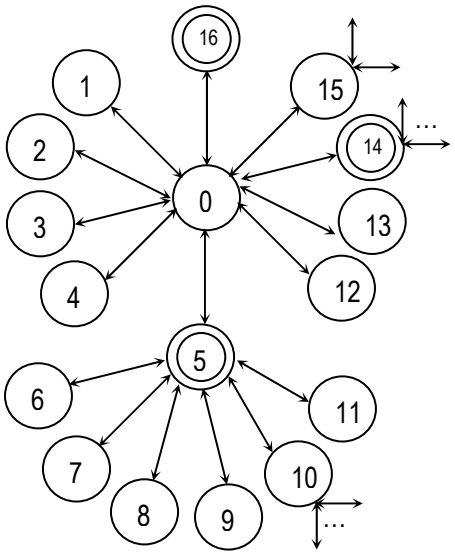


Fig. 1. Multi-agent architecture of ICEIPT

Except study plans with *depth-first*, *top-down*, *bottom-up* or *breath-first* the structure tree passing allows generating arbitrary combinations of them, e.g. more often used mixed strategy in the real teaching practice. A *TODE* facilitates the teacher (*T*) in planning/monitoring a lesson/test/exercise and helps the *L* in acquisition of the author's subject knowledge under the teacher's didactic one. Exept standard editors (text, graphical and so on) the architecture of each *TODE* includes three specialized tools: editor-generator (*EG*), interpreter-evaluator (*IE*), and task manager (*TM*) and offered the users a special purpose language. Such a language can be classified as a script internal visual very high-level mark-up language. By means of the *EG* the author and learner's subject knowledge is extracted and stored in resource text files with a fixed structure and extension. The tool interacts with the *T* to extract and store his/her didactic knowledge on session level in a resource text file with a fixed structure and extension. The resource files can be opened in a standard text editor to create different their variants. The *TB* is continuously extended with common and important parameters of the *LCs*/questions/tasks such as knowledge volume, degree of difficulty and system prompt, time expected for its completion, and so on. The *IE* based on the author's automatically computes these parameters lesson/question/task subprogram program tree. The tool also provides diagnostics of the learner's knowledge refreshing his/her model. After a construction task is performed by the *L*, the tool computes or registers his/her results relatively to the author's one. The main purpose of the *TM* is to sort the tasks for a given session according to teacher's preferences and to interpret the directives of the teacher's intervention in the course of the session's planned execution. After a session finishes, its parameters are accumulated as statistical experimental data in the *TB*. The ontology design is also used as a procedure for computing the parameters of the *CM* as well as precise assessment and diagnostics of the learner's knowledge relatively to the author's one.

Domain-Independent Ontology and Language

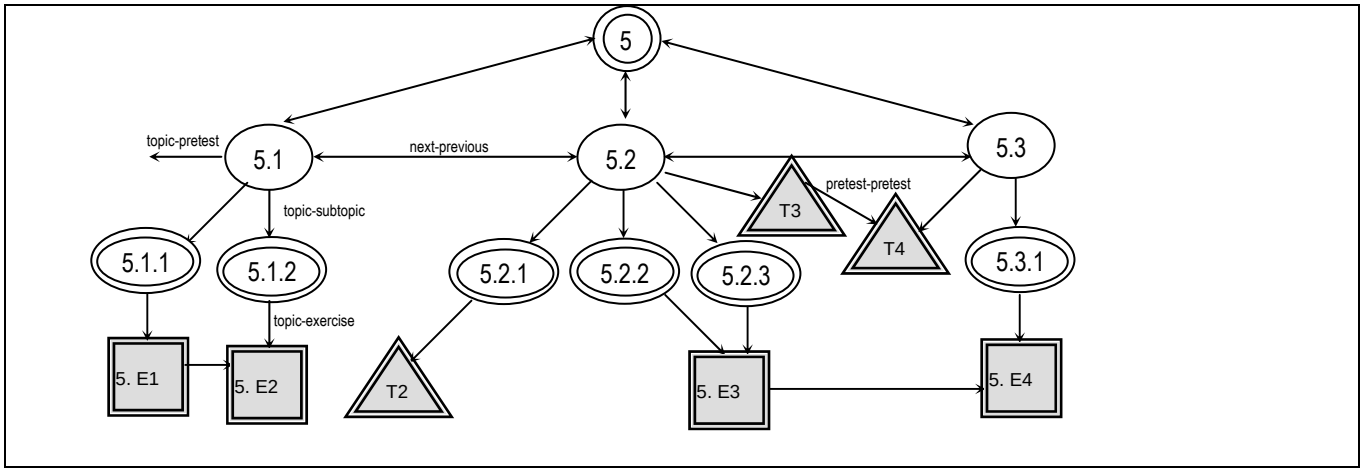
For each level the ontology is visually presented as a semantic graph/subgraph and a common structure program. The first level, concerning the teamwork knowledge, e.g. members, roles, communication tools, and so on. G^{tw} shown on fig. 2 has a form of two connected stars with *navigation* type links to the following pages: 0 – Courseware, 1 – Institution, 2 – Department, 3 - Author's team; 4 - Teacher's team, 5 - Initial course, 6 - Curriculum timetable, 7 - Curriculum annotation, 8 - Curriculum contents, 9 - Technology of teaching, 10 - References, 11 - Discussion forum, 12 - Search machine, 13 - Frequently Asked Questions, 14 - Terminological multilingual dictionary, 15 - Courseware map. Note, that double-circle nodes (5,14,16) stand for subgraphs considered in the next sections, 10 has a set of external links from/to some nodes of the subgraph on the third level, and 15 has links to/from nodes on three levels with an opportunity for downloading the lectures, tests, and exercises. The second level, concerning the *CM* structure (Fig. 3) is illustrated with the subgraph of node 5 on the previous level. For clear visual presentation the graphical primitives are added for: a topic (single ellipse), lesson (double ellipse), pretest (double triangle), exercise (double square). Besides explicit links "*topic-subtopic*" such a model contains other types such as: "*topic-test*", "*topic-lesson*", "*topic-exercise*", "*test-test*", and "*exercise-exercise*" the cardinality of which can be *1:N*. On the third level there are three separated semantic graphs respectively for a pretest, lecture, and exercise session. The goal a pretest is not only to assess the learner's lecture knowledge, but also to fill in the missing knowledge and correct the wrong one. The ontology of a test sesion with the initial node *P1* of the semantic graph is shown on fig. 4. Node *PA* stands for the administrative information; *PD* - key directives for the teacher's intervention during the learner testing; *PC* - criteria for the knowledge assessment; Except *next-previous* links between nodes questions *Qi* (*i =1,…,N*) questions external links from/to to the pages with the related lecture material also are shown. When the learner's test result is lower than a given threshold he/she has to perform the test again (the feedback *fail*). Otherwise the *L* is allowed to start the related practical exercise (the feedback *success*).



COURSE <*string*>
ORGANIZATION <*string*>
DEPARTMENT <*string*>
AUTHOR'S TEAM <*list_of_authors* >
TEACHER'S TEAM <*list_of_teachers* >
LEARNER'S GROUPS <*list_of_learners* >
TIMETABLE <*table*>

ANNOTATION <*memo*>
<*goal_description*>
TECHNOLOGY <*memo*>
REFERENCES <*list_of_references*>
FORUM | NOFORUM
SEACH | NOSEARCH
FAQ | NOFAQ
DICTIONARY | NODICTIONARY
MAP | NOMAP
LINKS <*list_of_home_pages*>
<structure_ description> END

Fig. 2. The ontology and program on the first level



<*structure_description*>:=
ROOT <*index*> <*string*>
{TOPIC <*index*> <*string*> INPUT|OUTPUT| INPUTANDOUTPUT <*list_of_indexes*>
[<*lesson_description*>]}
{TEST <*index*> INPUT|OUTPUT|
INPUTANDOUTPUT <*list_of_indexes*>

<*directives_description*>::=
ESCAPE | NOESCAPE
PRINT | NOPRINT
SAVE | NOSAVE
HEPL | NOHELP
DO | REDO
ASSESS | NOASSESS

<*test_description*>}

{EXERCISE <*index*> INPUT|OUTPUT| INPUTANDOUTPUT <*list_of_indexes*>

<*exercise_description*>} END

<*goal_description*> ::= ROOT <*index*>

GLOBAL STRATEGY *depth-first* | *breadth-first* | *top-down* | *bottom-up*

LOCAL STRATEGY *t* | *pt* | *ptr* | *cr*

<*directives_description*>

<*criteria_description*>

{<*sesion_description*>} END

TEACHER <*string*>

[<*local_goal_description*>] END

<*session_description*>::= LESSON | TEST | EXERCISE <*index*>

<*criteria_description*>::=

VOLUME <*integer*>

PROXIMITY > <*real*>

PROMPT > <*real*>

DIFICULTY <*real*>

DURATION <*hh.mm*>

TYPE SUCCESS|FAILURE| PERCENTAGE-

|SCALE <*number*>

CORRECTION <*real*>

MARK <string>

2-FROM: <integer1>TO:<integer2>

3-FROM: <integer1>TO:<integer2>

4-FROM: <integer1>TO:<integer2>

5-FROM: <integer1>TO:<integer2>

6-FROM: <integer1>TO:<integer2> END

Fig. 3. Second level ontology and language

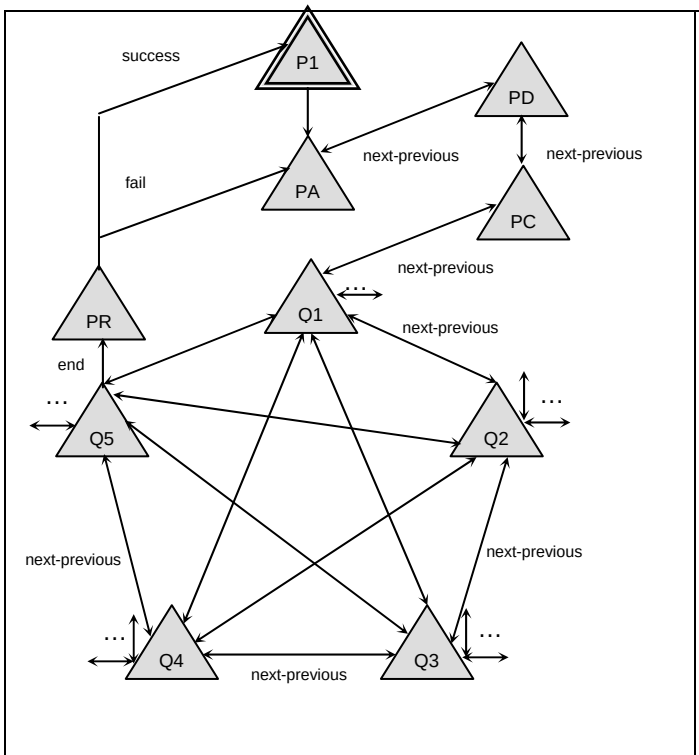


Fig. 4. Test ontology

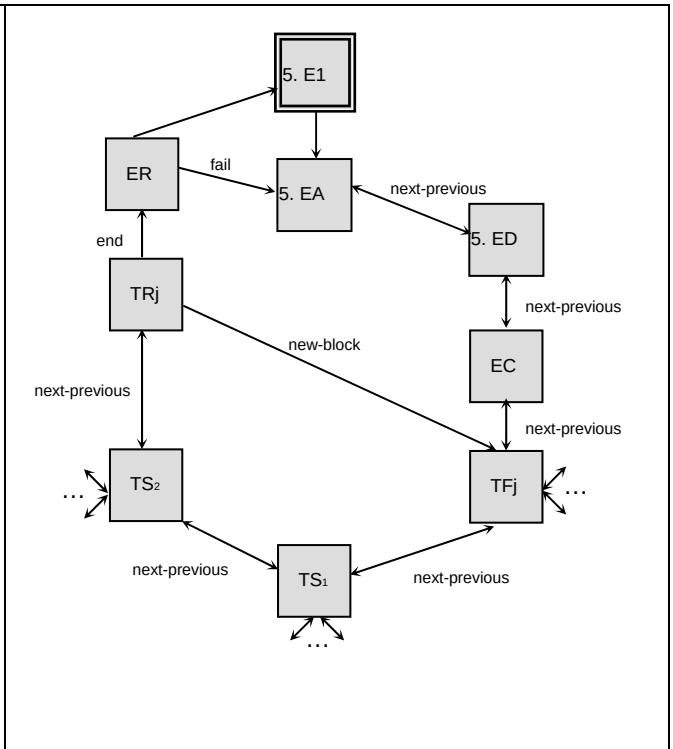
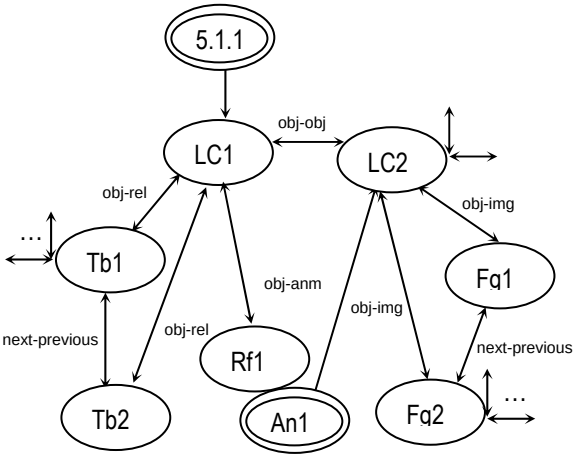


Fig. 6. Exercise ontology



lecture_description> ::= LESSON <*index*>[<*string*>]
[OBJECT <*index*> <*string*> INPUT | OUTPUT | INPUTANDOUTPUT <*list_of_indexes*>
CONTENT <link_to_HTML_document>]
[TABLE <*index*> <*string*> INPUT | OUTPUT | INPUTANDOUTPUT <*list_of_indexes*>
CONTENT <link_to_HTML_document>]
[{FIGURE <*index*> <*string*> INPUT | OUTPUT | INPUTANDOUTPUT <*list_of_indexes*>
CONTENT <link_to_HTML_dokument>}]
[{ANIMATION <*index*> <*string*> INPUT | OUTPUT | INPUTANDOUTPUT <*list_of_indexes*>
<*animation_task_description*>}] END

Fig. 5. Lecture ontology and program structure

On fig. 5 an example semantic graph of a lecture session with an animation style for teaching cognitively complex concepts the initial node 5.1.1 from fig. 3 is shown. Here nodes *LC1* and *LC2* stand for the textual description of two learning concepts, *Fg1* and *Fg2* for two images of *LC2*, *An1* for an animation of the concept, *Tb1* and *Tb2* for two relations of *LC1,* and *Rf1* for its references.. The common structure of an animation subprogram in a description language generated by a *TODE* for modeling with didactic Petri nets [10]. A sample of semantic graph corresponding to the square subgraph *E1* is presented on fig. 6. The *L* can read the exercise information about: administrative data (*EA*), directives for the teacher's intervention (*ED*), and the assessment criteria (*EC*). Node *TFi* corresponds to the *i*-th task formulation, and the next several nodes *TSj* (j=*1,M)* for step by step extraction of skills for its solving. Again except *next-previous* links the feedback to *TFi* is shown corresponding to the learner's moving on to the next task. If the mark in the exercise report (*ER*) is lower then a given threshold he/she has to perform the exercise again (the feedback "*fail*"). Otherwise he/she is allowed to continue with the next lecture topic. The languages for test and lecture description that are fully domain-independent the language for exercise task description depends on the task type such as algorithms [15], structural schemes [16], simulated systems [14].

Conclusion

In *ICEIPT* a subject task is viewed as capturing the author's knowledge constructing with automatically computing a set of important pedagogical parameters (knowledge volume, degree of system prompt, planned time for performance, and degree of difficulty). A group goal is expressed taking unto account logical functions, teacher's preferences, long-term plan constraints, and so on. Different goal-oriented study plans can be fully generated and flexible learning can be ensured to adapt towards learner's knowledge, preferences, and information needs.

The domain-independent ontology and language on course, structure, session and task level are expressive, understandable, and reusable. The lecture semantic graph differs availability of subgraphs for animation of complex learning concepts.

Bibliography

[1] Andreeva M., Models and Tools for Development of an Integrated Authoring Knowledge Testing Environment, *Thesis of PhD dissertation*, Rousse University, 2006 (in Bulgarian).

[2] Brusilovsky P., *Adaptive and Intelligent Technologies for Web-based Education*, Kunstiche Intelligenz, 1999.

[3] Ivanov K., Zabunov S., Individually Adaptive Learning Management System Project, *Internamional Conference on Computer Systems and Technologies CompSysTech'04*, Rousse, Bulgaria, 17-18 June, pp.IV.17-1 - IV.17-6.

[4] Jelev G. A. Strategy for E-learning in the Faculty of Computer Systems and Control in TU-Sofia, *International Conference on Computer Science*, Sofia, Bulgaria, 2004, pp. 263-267.

[5] Jesshope C., Heinrich E., D-r Kinshuk, Technology Integrated Learning Environments for Education at a Distance, DEANZ Conference, 26-29 April, 2000, Dunedin, New Zealand.

[6] Fan X., Yen J., Miller M., Ioerger T. R., Volz R., MALLET – A Multi-Agent Logic Language for Encoding Teamwork, *Transactions on Knowledge and Data Engineering*, Vol. 18. No. 1, 2006, pp. 123-138.

[7] http://www10.org/cdrom/papers/207/node3.html

[8] Peachey D. R., and McCalla G. I., Using Planning Techniques in Intelligent Tutoring Systems. *Int. J. Man-Machine Studies*, Vol.24, 1986, pp.77-98.

[9] Saveliev A. Y., Novikov V. A., Liubanov Iu. I., *Preparing the information for Computer-Based Systems*, Moscow, 1986 (in Russian).

[10] Stefanova S. P., *Application of Didactic Petri Nets for Teaching Purpose*, Thesis of PhD dissertation, Rousse, 2002 (in Bulgarian).

[11] Vassileva J., Deters R., Dynamic Courseware Generation on the WWW, *British Journal of Educational Technology*, Vol. 29, 1998, pp. 5-14.

[12] Zheliazkova I. I., A graph model of a dynamic planning of the teaching process, *Proceedings of the 8th International PEG Conference*, Sozopol, Bulgaria, May 30th-June 1st, 1997, pp. 97-104.

[13] Zheliazkova I. I., A Web-based multimedia intelligent environment for teaching, training and testing, *International Conference SAER,* Varna, Bulgaria, 2002, pp. 32-42.

[14] Zheliazkova I. I., Georgiev G. T., Representation and Processing of Domain Knowledge for Simulation-Based Training Systems*, Int. J. of Intelligent Systems*, 2000, Vol. 10, No.3, pp. 255-277.

[15] Zheliazkova I. I., Atanasova G., A Visual Language for Algorithm Knowledge, *Proceedings of the International Conference on Computer Systems and Technologies (e-Learning),* Rousse, 2004, pp. IV.24-1- IV.24-6.

[16] Zheliazkova I. I., Georgiev G. T., Valkova P. L., A Task-Oriented Environment for Structural Schemes Design, *International Journal of Information Technologies and Control*, Vol. 2, 2006, pp. 2-13.

Authors' Information

Irina Zheliazkova – Associate Professor; Rousse University, str. "Studentska" 8, Rousse 7017, Bulgaria; e-mail: irina@ecs.ru.acad.bg