TRANSPORT PROBLEMS 2009
PROBLEMY TRANSPORTU Volume 4 Issue 3 Part 1

Jan PIECHA*, Marcin BERNAŚ
Informatics Systems Department of Transport, Faculty of Transport,
Silesian University of Technology
ul. Krasińskiego 8, 40-019 Katowice, Poland
**Corresponding author.* E-mail: jan.piecha@polsl.pl

DIGITAL CONTROL SYSTEMS TRAINING ON A DISTANCE LEARNING PLATFORM

Summary. The paper deals with new training technologies development based on approach to distance learning website, implemented in the laboratory of a Traffic Engineering study branch at Faculty of Transport. The discussed computing interface allows students complete knowledge of traffic controllers' architecture and machine language programming fundamentals. These training facilities are available at home; at their remote terminal. The training resources consist of electronic / computer based training; guidebooks and software units. The laboratory provides the students with an interface entering into simulation packages and programming interfaces, supporting the web training facilities. The courseware complexity selection is one of the most difficult factors in intelligent training unit's development. The dynamically configured application provides the user with his individually set structure of the training resources. The trainee controls the application structure and complexity, from the time he started. For simplifying the training process and studying activities, several unifications were provided. The introduced ideas need various standardisations, simplifying the e-learning units' development and application control processes [8], [9]. Further training facilities development concerns virtual laboratory environment organisation in laboratories of Transport Faculty.

PLATFORMA KSZTAŁCENIA NA ODLEGŁOŚĆ DLA CYFROWYCH SYSTEMÓW STEROWANIA

Streszczenie. Artykuł opisuje wybrane aspekty rozwoju nowych technik kształcenia opartych o technologie internetowe. Opisane prace zostały zaprojektowane i zaimplementowane w laboratorium Cyfrowych Systemów Sterowania, na specjalności Inżynieria Ruchu, kierunku Transport w Politechnice Śląskiej. Opisane elementy interfejsu użytkownika pozwalają dokonać indywidualnego doboru tematyki kursu / lekcji w zakresie podstaw architektury sterowników ruchu drogowego oraz zasad ich programowania na poziomie języka maszynowego. Oferowany serwis kształcenia można udostępnić do pracy domowej studenta; na jego odległym terminalu. Zasoby systemu zawierają zarówno elementy elektronicznego kształcenia jak i przewodniki tradycyjne, wspomagające samodzielną prace studenta. Portal komunikacyjny udostępnia studentowi blok symulacyjny sterownika wraz z narzędziami samodzielnej konfiguracji lekcji. Poziom złożoności lekcji zostaje dopasowany do aktualnego poziomu wiedzy użytkownika. To jedno z najtrudniejszych zagadnień projektowania inteligentnych systemów kształcenia. Dynamicznie konfigurowanie aplikacji umożliwia bieżące sterownie poziomem złożoności lekcji. Interakcje ćwiczącego sterują na bieżąco stopniem złożoności lekcji, od chwili jej uruchomienia. Dla uproszczenia procesów kształcenia oraz aktywności studenta zaproponowano szereg unifikacji systemu, prowadzące do uproszczenia stopnia złożoności dalszych aplikacji, w systemu kształcenia na odległość. Dalsze prace na tym polu będą rozwijane w kierunku Uniwersytetu Wirtualnego w laboratoriach Wydziału Transportu.

1. INTRODUCTION

Traffic engineering qualifications concern one of the most demanding education areas. Digital systems development defines new technologies and new implementation techniques. Active engineers are searching for a hot data of these new technologies with their description and equipment introduction.

The web training platforms and new training facilities provide the user with comfortable and flexible possibilities of his qualifications improvement [1], [2], [3], [5].

The paper introduces selected solutions implemented in virtual laboratory of Digital Control Systems that is under construction in the Department of Informatics, at the Faculty of Transport.

The syllabus requirements, introduced in a file of Transport Study standards, concern 8 and 16-bit processor architecture and their auto-code exploration with communication processes. Analysing their applications in a traffic control systems, data communication buses description and addressing techniques analysis, etc. Moreover the data communication protocols and industrial controlling devices implementation are discussed in these works as well. For fulfilling the users' needs the web training laboratory organisation was elaborated.

Till today many works in an e-learning and distance learning technologies were undertaken; both for training resources unifications and for training content distribution platform (LMS - Learning Management Systems) development [22], [24], [25].

For simplifying the applications structure and training results evidences organisation single or multiple choice and single fill - up format are still used; satisfying the training units needs.

However, one can find many examples where these data evidences are not satisfying the user. The presentation frame expresses more sophisticated content where traditional interactions are not effective enough. They are not giving enough evaluation mechanisms and statistics for the user's knowledge estimation.

For appropriately matched answers, defining the user's knowledge, several unification of the application structure, were elaborated. The obtained solutions are able to keep a track of the training process in accordance with a defined route though the database content.

The applications unifications are obtained by controlling facilities of the application development platform: the Multimedia Application Management Shell (MAMS) [4], [11], [12], [13] was created.

The laboratory network works under distribution platform *(ex. Moodle*) supported by a controlling unit, installed on a main machine, collecting knowledge of the users [20]. They are supported by various management functions, defined according to the scheme presented in Fig. 1. The presentation and administration services were installed on a client's machine – placed on a local server.

The local machine makes conclusions on a separate layer that is independent from LMS, although they are finally joined on a main machine via communication interfaces.

For the distance learning services development several common technologies were applied: Apache 2.0 [38], PHP 5.0 [38], Tomcat 5.5 [34] and JAVA 1.5 EE [35]. The system e-content database was designed on a freeware Postgres database solution; version 8.

The services were installed on Linux Debian free distribution, however this product has a structure that corresponds with a standard multiplatform technology; with simple migration to any available operating system.

The JAVA environment supports the network evaluation algorithms, running on a main controlling machine. PHP technology provide us with interfaces between controlling modules of the system; simplifying the Moodle [30], [42] or USE-LMS [24] platforms integration.

Several additional supporting technologies were used as well:
- SAX [39] and DOM [39], for XML processing engines and JAVA services description,
- JGRAPH[35], for operations support,
- JGRAPHT[37], for the multi directed graph definition and application units relations description,
- FUZZY ENGINE[36], for the conclusions support,
- JSCI [40], for the statistics measures and their estimations.

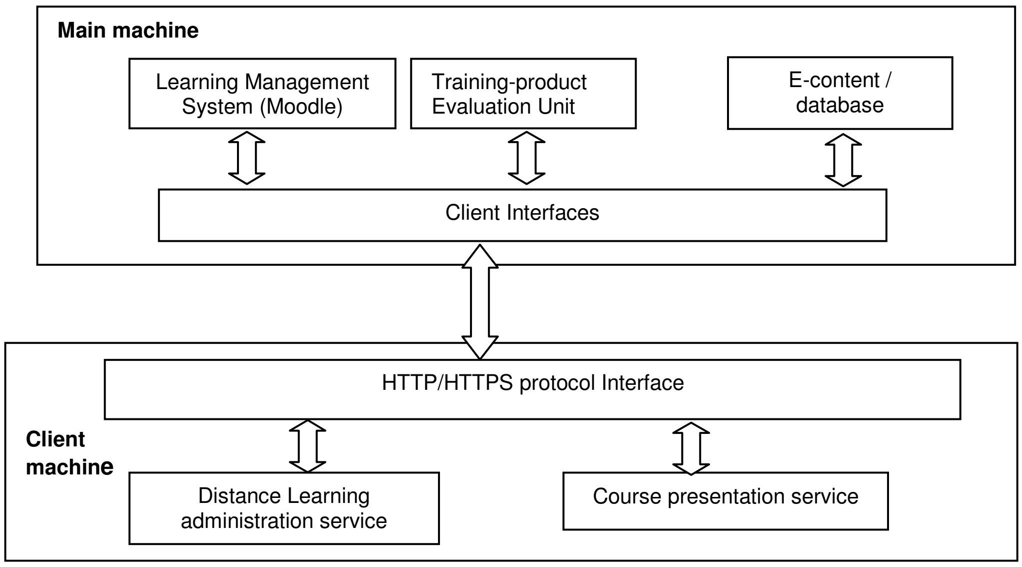

Main machine
Learning Management System (Moodle)
Training-product Evaluation Unit
E-content / database
Client Interfaces
HTTP/HTTPS protocol Interface
Client machine
Distance Learning administration service
Course presentation service

Fig. 1. The training management distribution functions
Rys. 1. Funkcje zarządzania i dystrybucji zasobów

2. CONCLUSIONS ENGINE

A fundamental part of the training platform determines all relations of the e-content units [17] with evidences of the user's skills. A tree structure defines the courses relations, as in example solution available in SCORM standard [31], [32], [33]. It is based on semantic descriptors, used in the majority of thesauruses and syllabuses, using RDF [27] (*Resource Description Framework*) and TESE [41] (*Thesaurus of European Search Engine*) formats.

The formal representation of a directed multigraph [18] [21] [23] is assigned as:

$$G= (V,E) \tag{1}$$

where:

$V = \{v_1, v_2, ..., v_n\}$, is a set of vertices identified with system objects (O),

$E = \{e_1, e_2, ..., e_n\}$, is a set of edges describes relations and their functions (F).

Vertices within the directed multigraph are divided into three separate sets, according to the defined layers (L):

$$V = R \cup T \cup U \tag{2}$$

$$R \cap T = R \cap U = T \cap U = \phi \tag{3}$$

where:

R - finite set of objects, which represents e-content frames stored in system:

$$R= \{r_1,r_2,\ldots,r_k\} \tag{4}$$

T - finite set of objects; represents information gathered in frames, described by semi-natural language:

$$T= \{t_1,t_2,\ldots,t_m\} \tag{5}$$

U - finite set of objects, which represents system users:

$$U = \{u_1,u_2, \ldots,u_n\} \tag{6}$$

Layers R, T, U in graph structure are defined as follows:

$$\begin{aligned}
&V_L = \{v_1, v_2,\ldots, v_n\}\\
&C_L = \{c_1^L, c_2^L,\ldots, c_k^L,\}\\
&P_{c_L} = \forall c_j^L \; P_{cl} = \{p_1, p_2,\ldots, p_m\}\\
&F^V : V_L \times C_L \to P_{c_L}\\
&\lambda \; : C_L \to k : k = [0,1]
\end{aligned} \tag{7}$$

where:

- v_i - object (vertices) in L layer,
- L - distinct layer with object and features,
- C_L - L layer finite set of features (attributes),
- P_{c_L} - values set for layers attributes,
- F^V - default function, assigns value p_{cj} for objects v_i and feature c_j,
- λ - quality function, influence on evaluation and lesson selection process (default "1") of specified feature.

Directional multi-graph assigns an edge for ordered pair of vertices:

$$G : e_i \Rightarrow V \times V \tag{8}$$

The graph edges are based on Cartesian product of two layers vertices. Products with all relations define all dependences in the application database. Relations of the Cartesian square product of the layer is called internal, otherwise it is called external. Edges definition (presented in equation 4) is similar to the layer definition edge, as:

$$\begin{aligned}
&E_{L'\times L''} : \{(v_i, v_j) : (v_i, v_j) \in L'\times L''\},\\
&C_{L'\times L''} = \{c_1^{L'\times L''}, c_2^{L'\times L''},\ldots, c_k^{L'\times L''},\}\\
&P_{c_{L'\times L''}} = \forall c_j^{L'\times L''} \; P_{cL'\times L''} = \{p_1, p_2,\ldots, p_m\}\\
&F^E : E_{L'xL''} \times C_{L'xL''} \to \{p_i : p_i \in P_{C_{L'xL''}}\}\\
&\lambda \; : C_{L'xL''} \to k : k = [0,1]
\end{aligned} \tag{9}$$

where:

- $E_{L'\times L''}$ - edge defined as ordered pair of vertices,
- F^E - default function, assigns value p_{cj} for edge e_i and feature c_j,
- $C_{L'xL''}$ - finite set of features (attributes) for Cartesian product (layers L'xL''),
- $P_{C_{L'xL''}}$ - values set for distinct attributes. Values are within [0,1] range,
- λ - quality function, defines influence on evaluation and lesson selection process (default "1") of a feature.

Function F^V and function F^E are defined by the Cartesian products. Reorganising vertices function into the edge function we simplify the application structure into one class of features only (the all discriminated features were described already in many works [19] [28]).

The graph relation is an entry of the conclusions making algorithm that generates the lesson structure, appropriately to the target user, found on the U layer. The algorithm is finding an optimal lesson structure spotting the graph relations, based on maximal values of an aim functions; where the aim for these conclusions concerns term t_j and its user u_i.

$$\max(F_{c_{grade}}(t_j)|u_i) \tag{10}$$

The algorithm of the lesson construction consists of three main blocks; as it is presented in Fig. 2 [28].

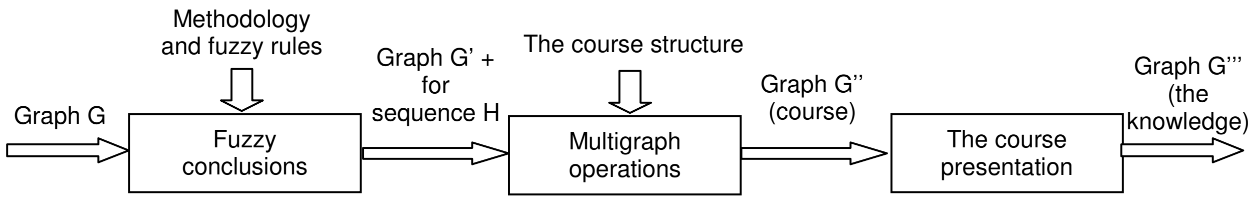


Fig. 2. The algorithm diagram of lesson structure composition
Rys. 2. Schemat blokowy struktury organizacyjnej lekcji

The redundant data is removed by a-priori filter, rejecting all relations, not joined with the user's lesson.

The methodological model classifies all characteristic features, modifying their weights (λ). Moreover, new connection in the graph (based on the edges in R layer) is searched.

The application characteristic features are expressed by means of fuzzy sets. These values membership function (μA) is defined by the following relations, where:

- 1: member belongs to a given set,
- (0,1): partial membership of the element in the given set,
- 0: member not belonging to the given set.

Fuzzy conclusion concerning more than one feature, requires additional definitions by well known axioms [29]. They allow carrying multiplication (AND) and addition (OR) of the features.

Multiplication, by "T–norm": TN(μA'(z), μA''(z)) = min(μA'(z), μA''(z)), addition, by "S-norm": SN(μA'(z), μA"(z))=max(μA'(z), μA''(z)).

The linguistic variables are expressed by trapezoid membership functions; with four factors, defining the values range (11).

$$\mu_A^{abcd}(\mathbf{z}) = \begin{cases} \frac{z-a}{b-a} & \text{dla} \quad b > \mathbf{z} > a \\ 1 & \text{dla} \quad c \geq \mathbf{z} \geq b \\ \frac{d-z}{d-c} & dla \quad d > \mathbf{z} > c \\ 0 \quad dla & z \leq a \vee z \geq d \end{cases} \tag{11}$$

If the membership functions were not defined, they are set automatically to low, medium and high ranges. Fig. 3 presents the default variable ranges definition.

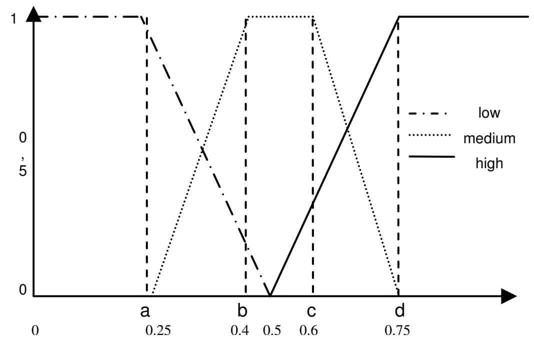


Fig. 3. The default membership functions definition
Rys. 3. Domyślna funkcja przynależności dla pojęć lingwistycznych zbioru rozmytego *A* oraz *A*'

Conclusion making process is done according to the "modus ponens' rule [43]. The formal description of the chosen method is defined by:

$$\mu_{A'}(z') = \sup_{z\in \mathbf{Z}}\left[\mu_{A'}(\mathbf{z}) \overset{T}{*} \mu_{A\to A'}(\mathbf{z}, z')\right] \quad (12)$$

In the application discussed in the contribution the following functions were implemented:
- Łukaszewicz rule [43]:

$$\mu_{A\to A'}(z,z') = \mu_R(x,y) = \min[1, 1-\mu_A(z) + \mu_{A'}(z')] \quad (13)$$

- and max–min rule, introduced by Zadeh [29]:

$$\mu_{A\to A'}(z,z') = \mu_R(z,z') = \max\{\min[\mu_A(z), \mu_A(z')], 1-\mu_A(z)\} \quad (14)$$

Finally the new relation value or the control vector, for the application structure is obtained by so called defuzzification process [43]. From many defuzzification functions the Centroid Average [29] was selected; the less complex calculation algorithm.

The controlling sequence (H=<h_1, h_2,…,h_n>) is obtained from the application database, as well as new relations set, processed in a second block.

This second block describes types of graph operations needed for the lesson's structure definition:
- sum, where: $G_{suma} = G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$, (15)
- multiplication, where: $G_{iloczyn} = G_1 \cap G_2 = (V_1 \cap V_2, E_1 \cap E_2)$, (16)
- α–cut, takes all vertices, where at least one edge value is greater than α
  $$G_{cut} : v_i \in V', F(E_{V'xV''}) > \alpha), \quad (17)$$
- sub-graph $(\cup C)$, where result graph relation is only relation of a defined feature:
  $$G_{podgraf} : e \in E, F : E\times C \neq \phi, \quad (18)$$
- the shortest path algorithm, was based on modified Dijkstra – algorithm [16],
- extended path algorithm,
- maximal flow algorithm, based on Ford–Fulkerson algorithm [46].

They define operations of G'' graph that is a sub-graph of G', containing only a lesson structure, was divided into several tasks of the lesson. The tasks are defined by their functions, as:

$$FL_{path}, FL_{repetition}, FL_{evaluation}, FL_{verification} \quad (19)$$

If the presentation time of the lesson's frames is exceeding the maximal limit (defined by steering item h_j) the aim t_i is treated as a course key descriptor t_l for l=1..m units, defining t_i , treated as the lesson's aims:

$$\xi t_i = \bigcup_{l=1}^{m} \Gamma_{t_l}, \quad (20)$$

The functions FL are used as indicator to follow the lesson's presentation process and for verifying purposes. The lesson definition and controlling processes are illustrated by Fig. 4.

The algorithm of the lesson organisation is performed by graphs, defined as vertices and edges of *path* functions. The starting point of the lesson defines free vertices of a first key descriptor (t_x). The repetition number *lr* is placed in the complexity features of the frame (p_r) and h_i item as:

$$lr = \begin{cases} 1 : p_r <= h_i \\ \dfrac{p_r}{h_i} : p_r > h_i \end{cases} \quad (21)$$

The selected frames return the user's interaction sequence into the R layer. In case the results are not matched with a cognitive definition the algorithm directs the course into a block 2 and sets the conclusions into to the verification unit [16].

If requirements, specified above are fulfilled, the next frame of the lesson is selected, in accordance with the following algorithm steps:
- evaluation of user's interactions, by the features defined for functionality evaluation,
- fetch a next edge E(r_x,r_i) from the *repetition* functions, were r_x defines a currently evaluated frame and r_i is an expected next frame,

- if a value of the *evaluated* edge $E(r_x,r_i)$ function is smaller than the value of the edge $E(r_i,r_x)$, for the *repetition* function, then frame r_i is added into the set of candidates for the next frame finding, in a path (ZK),
- if exist uncheck edge for vertices r_x, go to the second step,
- if ZK is empty, set the next frame r_i as it is appointed by the highest value of edges $E(r_x,r_i)$ for the *path* function find and closing the algorithm,
- select the frame form ZK set, having the highest value in the repetition product of the edge multiplied by its feature λ weight.

3. THE COURSES EXAMPLE

Many examples, describing programming techniques, can be found [6] [7] [14] [17]. However the majority of them illustrate high level languages techniques. For industrial controllers (among them traffic controllers) programming a low level language or machine code is used. The developed training units provide the user with the computer supported interface illustrating many aspects of this complex approach to traffic engineering.

The discussed above conclusions making engine was implemented and evaluated in a course of **16-bit processors architecture and programming study**.

The programme implementation has to be predicted by several introductory works; specifying the time duration of the course / lesson [4] [5] [8] [9].

The application is constructed in accordance with the multilayered Kay's strategy [4], [7], [9]. The application graph controls the application content twofold: for the layers relations, the content frames features definition, as well as dependencies between them (by an XML files and syllabuses). The basic structure of the course was defined by the expert (in an administration panel) that allows to modify the application structure.

The methodological aspects of an expected knowledge level are expressed in the fuzzy value range definitions. The system allows us to define the methodological measures as well; that are not strictly described. These rules are defined by a sequence of the following extended statements:

if feature(layer, layer) is value AND/OR feature(layer, layer) is value ...
THEN feature(layer, layer) is value.

The layers within these statements can be pointed out by the specific objects (of the layer) or set by their objects, as in example:

- User-result.user(**T**,**U**) **-** will determine the result for the given user, in accordance with the specific term,
- User-result.user(**T**,*****) **-** will consider knowledge of the all users, which have knowledge relation to T terms.

A final step of the development process is the MAMS pattern definition, by the XML file in a following example structure:

```
<lesson scheme=”name”>
        <function name=”path/repetition/grade/verification”>
                        <steering value=”hi” activation=”range”>
                                <operation type=”graph operation”>
                                        <item type=”G”>Graph</item>
                                        <item type=”V”>node vi</item>
                                        <item type=”E”>edge</item>
                                        <item type=”H”>steering value</item>
                                        <item type=”O”>graph operation</item>
                                <operation>
                        </steering>
            </function>
</lesson>
```

These relations and a given data define the input of the performed evaluation procedures. For the given course (structure of the lesson) has been introduced by the block diagram in Fig. 4.

The presented graph can run in several execution modes. First is called guiding course, where the user is led (step by step) through the lesson's content (CAI – Computer Assisted Instructions). This mode is provided for beginners, starting this training. A second mode concerns interactions (CAL – Computer Aided Learning) where the course flow and repetitions depend on the user's interactions. The expert, having the satisfying knowledge, can start from his skills checking. His entering into the course content concerns only part of the course. The course construction interface was introduced in Fig. 5.

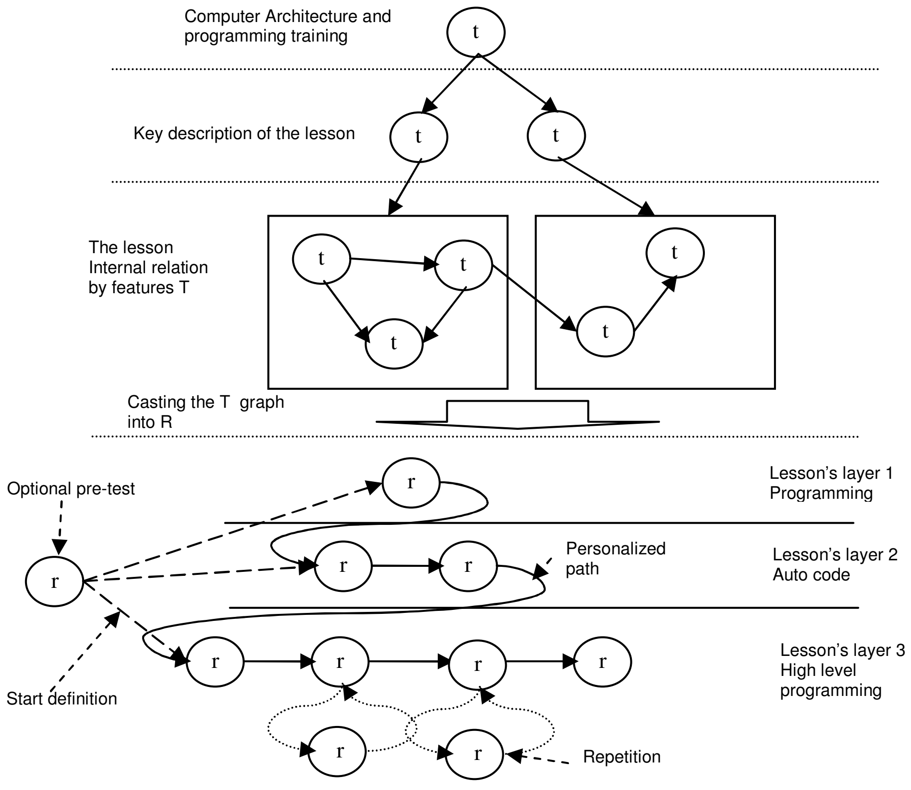
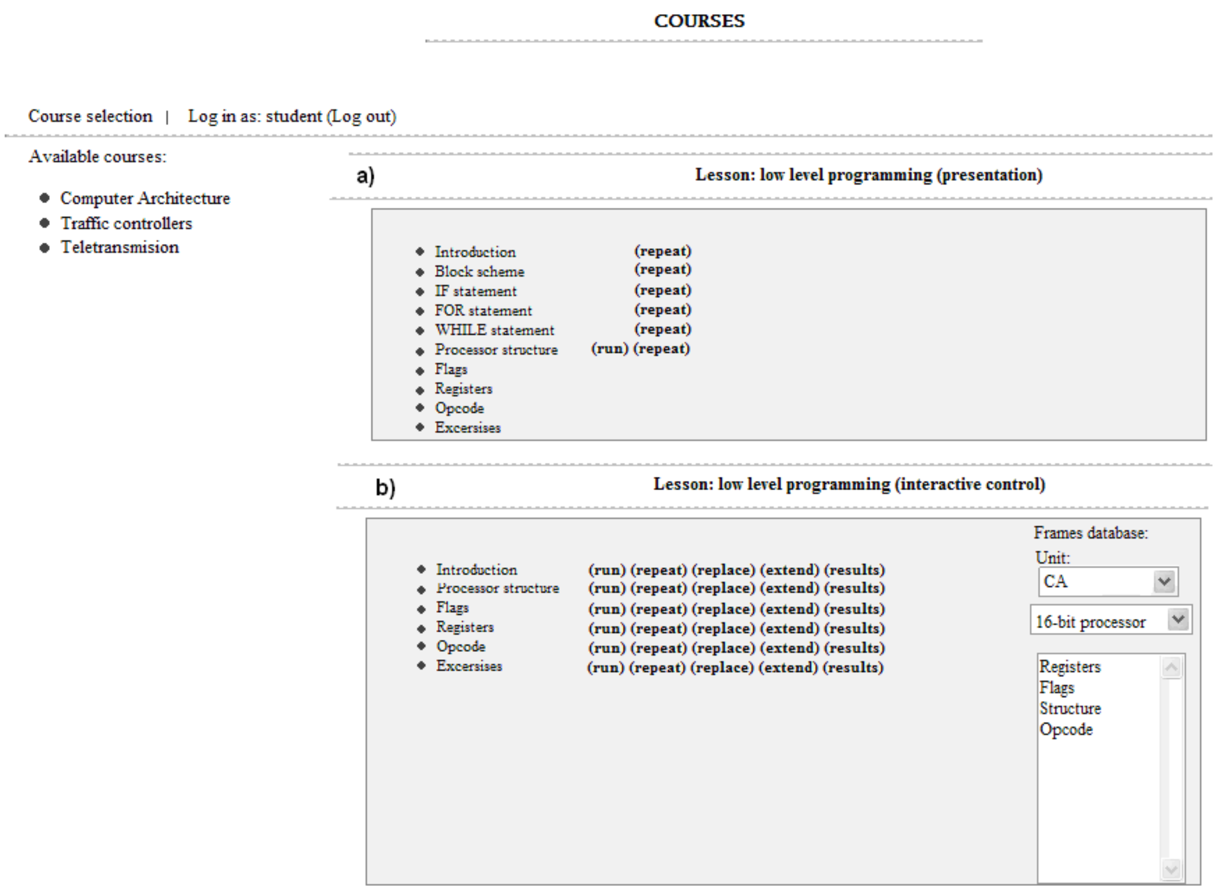

Computer Architecture and programming training
t
Key description of the lesson
t
t
The lesson Internal relation by features T
t
t
t
t
t
Casting the T graph into R
Optional pre-test
r
Lesson's layer 1 Programming
r
r
r
Personalized path
Lesson's layer 2 Auto code
Start definition
r
r
r
r
Lesson's layer 3 High level programming
r
r
Repetition

Fig. 4. Block diagram of the lesson organisation, based on the G'' graph
Rys. 4. Schemat blokowy organizacji lekcji, na podstawie grafu G''

4. THE COURSE EXAMPLE FRAMES

The learning process model implemented in MAMS platform (by means of Key) was provided by fuzzy conclusion technique [3] [4] [5] [6]. The conclusion rules related to layers T, R and U were illustrated by the example given bellow (for two variables: knowledge threshold (h_3) and complexity level(h_2)):

if „UU:WO"(U,T) is low then HH:h_3 is low
if „UU:WO"(U,T) is medium then HH:h_3 is medium
if „UU:WO"(U,T) is high then HH:h_3 is high
if "UU: TS"(U,U) is high and „UU:WO"(U,T) is high then HH:h_2 is high
if "UU: TS"(U,U) is medium and „UU:WO"(U,T) is high then HH:h_2 is high
if "UU: TS"(U,U) is high and „UU:WO"(U,T) is medium then HH:h_2 is high
if "UU: TS"(U,U) is medium and „UU:WO"(U,T) is medium then HH:h_2 is medium
if "UU: TS"(U,U) is low and „UU:WO"(U,T) is medium then HH:h_2 is medium
if "UU: TS"(U,U) is low and „UU:WO"(U,T) is low then HH:h_2 is low



COURSES

Course selection | Log in as: student (Log out)

Available courses:

- Computer Architecture
- Traffic controllers
- Teletransmision

a) Lesson: low level programming (presentation)

Introduction	(repeat)
Block scheme	(repeat)
IF statement	(repeat)
FOR statement	(repeat)
WHILE statement	(repeat)
Processor structure	(run) (repeat)
Flags	
Registers	
Opcode	
Excersises	

b) Lesson: low level programming (interactive control)

Introduction	(run) (repeat) (replace) (extend) (results)
Processor structure	(run) (repeat) (replace) (extend) (results)
Flags	(run) (repeat) (replace) (extend) (results)
Registers	(run) (repeat) (replace) (extend) (results)
Opcode	(run) (repeat) (replace) (extend) (results)
Excersises	(run) (repeat) (replace) (extend) (results)

Frames database:
Unit: CA
16-bit processor
Registers
Flags
Structure
Opcode

Fig. 5. The course mode selection interface: a) for presentation, b) for interactive control of the application
Rys. 5. Przykład okna selekcji trybu organizacji lekcji: a) prezentacyjny, b) sterowanej interaktywnie

The above rules produce, together with the lesson scheme, the three layer course structure.

The structure is described by the directed graphs, produced form the following XML scheme:

```
<function name="path">
    <item type=”O” name=”shortest path”>
        <item type=”O” name=”v-cut”>
            <item type=”O” name=”v-cut”>
                <item type=”O” name=”sum”
                    <item type=”G”>isPartOf</item>
                    <item type=”G”>next</item>
                    <item type="G">broaden</item>
                </item>
                <item type=”O” name=”cut”>
                    <item type=”G”>HH: h3</item>
                    <item type=”G”>grade</item>
                </item>
            </item>

            <item type=”G”>HH: h2</item>
        </item>
    </item>
</function>
```

The introduced rules and the lesson’s scheme are defined in the system once. After that the application structure is enriched by new relations, for every new course. The course structure is created by the expressions of semi-natural language, within its RDF features. In Fig. 6 the course organisation structure, concerning the assembly language training, was presented.

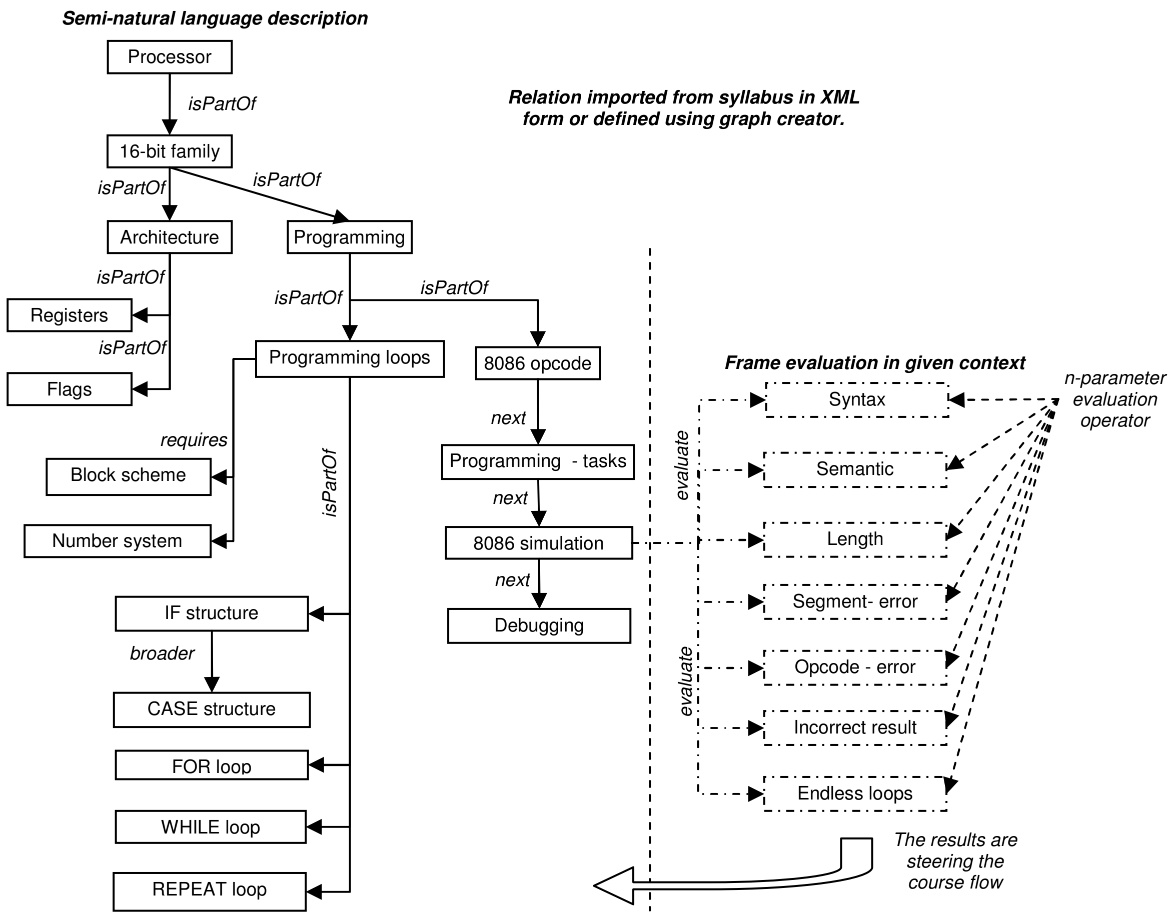
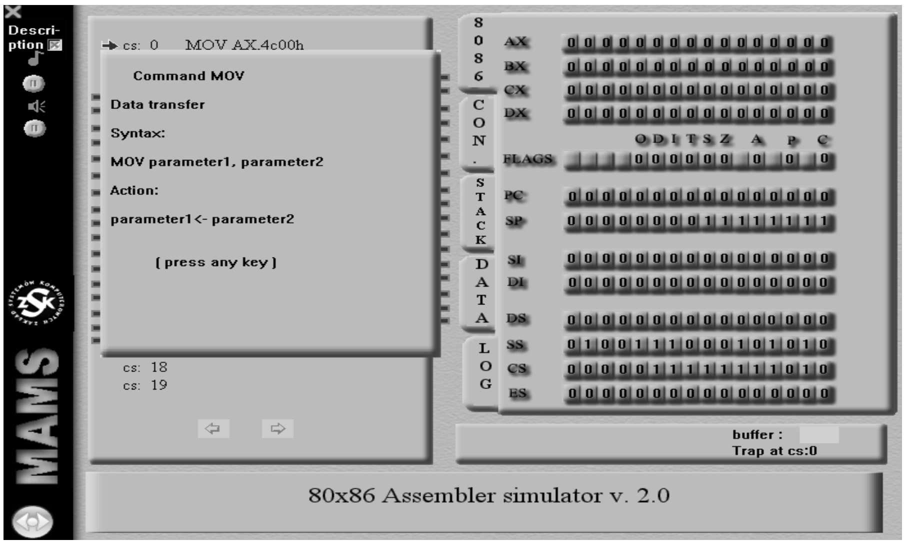


Fig. 6. The assembly language training course structure
Rys. 6. Struktura organizacyjna kursu - język asembler.

The application interface is co-responding with the user's knowledge and the selected mode of the application [20]. The most complex level of the discussed application starts from an assembly language programming interface that discusses not only programming principles. This interface also presents and explains the programme codes execution, on a register level of the processor (Fig.7). All execution items were illustrated by a debugging environment available on-line after the programmes are executed.



```
→ cs: 0    MOV AX.4c00h

    Command MOV
Data transfer
Syntax:
MOV parameter1, parameter2
Action:
parameter1<- parameter2

      [ press any key ]

cs: 18
cs: 19
```

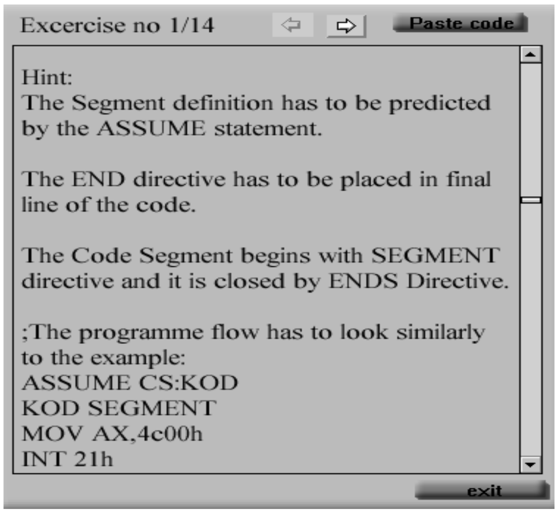
```
AX     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
BX     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
CX     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
DX     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
               O D I T S Z   A   P   C
FLAGS          0 0 0 0 0 0   0   0   0
PC     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SP     0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
SI     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
DI     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
DS     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SS     0 1 0 0 1 1 1 0 0 0 1 0 1 0 1 0
CS     0 0 0 0 0 1 1 1 1 1 1 1 1 0 1 0
ES     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

buffer :
Trap at cs:0

80x86 Assembler simulator v. 2.0
```

Fig. 7. The debugging interface
Rys. 7. Interfejs debuggera lekcji

The above tool helps the trainee, programmes writing and analysis, in a very easy way. The analysed tasks are divided into several units that can analysed separately and recalled any time we want. They can be automatically copied into editing window (Fig. 8) of any other applications.



Hint:
The Segment definition has to be predicted by the ASSUME statement.

The END directive has to be placed in final line of the code.

The Code Segment begins with SEGMENT directive and it is closed by ENDS Directive.

```
;The programme flow has to look similarly
to the example:
ASSUME CS:KOD
KOD SEGMENT
MOV AX,4c00h
INT 21h
```

Fig. 8. The programming window example
Rys. 8. Przykład okna programu

Many additional debugging facilities are available as well. The trainee is able to modify the processor states then coming back into the description mode, which displays short info about current actions of the processor. The programme execution results are not only visible in the processor registers; we can also change the observation panel to the memory occupation analysis and into the console status recognition.

The aim of the course is to learn writing programmes in assembly language. The application user can write all programme codes then the application will point out any mistakes. The build in menu can be used for writing codes by clicking the mouse.

The knowledge analysis results are available in simulator, where twenty parameters describe the knowledge level. If the questions level occurs too difficult for the user (according to interactions results), the application complexity automatically goes down.

The example courseware consists of following subjects:

- processor structure explanation with fundamental programming facilities,
- programme segments of memory space description,
- the assembly language instructions analysis,
- available loops of the assembly language,
- the loop-solutions and implementations.

In the application unit the following presentation modes are available:

- an electronic book,
- tutorial packages,
- simulation units in linear presentation sequences,
- branched, interactive presentation units,
- quizzes, etc.

Every frame of the course / lesson is evaluated separately, by multiple measures defined as n-operator [27] [28]. The results are interpreted in the layer T, analysing relations in a context defining the course essential terms. The evaluation scheme for a single frame illustrates a block diagram in Fig. 9.

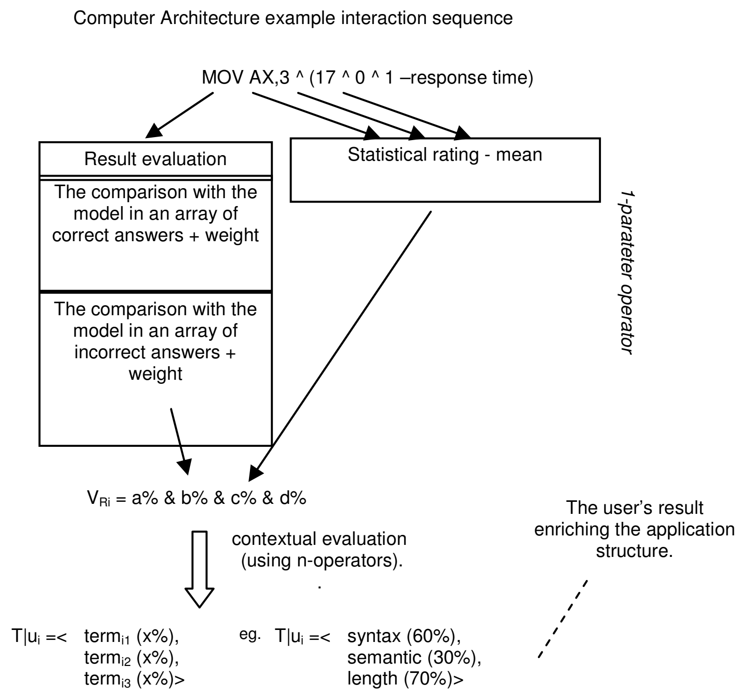

Computer Architecture example interaction sequence
MOV AX,3 ^ (17 ^ 0 ^ 1 –response time)
Result evaluation
The comparison with the model in an array of correct answers + weight
The comparison with the model in an array of incorrect answers + weight
Statistical rating - mean
1-parateter operator
V_Ri = a% & b% & c% & d%
contextual evaluation (using n-operators).
The user's result enriching the application structure.
T|u_i =< term_i1 (x%), term_i2 (x%), term_i3 (x%)>
eg. T|u_i =< syntax (60%), semantic (30%), length (70%)>

Fig. 9. The MAMS evaluation process by $^1\Phi$ and $^n\Phi$ operators
Rys. 9. Schemat oceny na platformie MAMS – operatory $^1\Phi$ i $^n\Phi$

The analysis results are obtained from the repetition functions, of the course graph repetition, producing results in a second step of the conclusion making algorithm. The electronic means of the course are supported by guide books for both, the user and for the trainer (or the system administrator) in the so called Virtual University.

5. CONCLUSIONS

The paper presents some proposals of an automatic courseware controlling processes, using high quality measures; evaluated for one subject, also available for many other items.

The application platform complexity can be friendly user in case the applications developer will use the elaborated interfaces.

The given solutions unifies features of application database into layers and graph structures, connecting users' profiles, e-content database as well as their description in semi-natural language items.

The user's interactions, within the application layers, can be evaluated in several different modes; using the same presentation frames. The layer User, together with the layer Term, provides us with the knowledge measures in the description context.

Thanks to the graph data assignment, there are no boundaries in the units' type connections. The lesson structures can be modified on-line, using the same evaluation base (tree and sequence structures as well).

The unique authors' solutions provide the user with a hybrid system, for ITS (intelligent tutoring system) development.

Several example of courses were empirically evaluated in a full cycle of a study. These results and variety of possible implementation are worth noticing. Solutions can be easily incorporated into other subjects.

References

1. Burke R. L.: *CAI Source Book.* Englewood Cliffs, Prentice- Hall, New York, 1982.
2. Eberts R. E.: *Learning strategies in CAI design.* The International Journal of Applied Engineering Education, No. 2/86, Oxford, 1986, pp. 113-121.
3. Piecha J.: *The multilevel model for microcomputer ICAI systems.* The International Journal of Applied Engineering Education, Vol.5, No 3, Pergammon Press, Oxford, 1989, pp. 391-395.
4. Piecha J.: *The programmable shell for multimedia applications development.* Journal of Applied Computer Science, Vol.7, No 2, Łódź, 1999, pp. 31-43.
5. Piecha J.: *The Intranet Databases and some Approach Troubles into Multimedia Files.* Proc. Int. Conf. "Computer Based Learning In Science" - CBLIS'99, Enschede, 1999, pp. 271-279.
6. Esteves M.: *Contextualization of Programming Learning: A Virtual Environment Study.* 38th ASEE/IEEE Frontiers in Education Conference, New York, 2008. http://fie-conference.org/fie2008/papers/1575.pdf
7. Wey Chen J. Chih-Cheng L.: *A Van Hiele Web-based Learning System with Knowledge Management for Teaching Programming.* Proceedings of the Sixth International Conference on Advanced Learning Technologies (ICALT'06), 2006. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01652381
8. Piecha J.: *Data units selection within sequential database.* Journal of Medical Informatics and Technologies, Vol.2/2001, pp. 155-162.
9. Para T., Piecha J., Pawełczyk P.: *The e-content distribution platform for Internet Virtual University services.* Research Reports on Distance Learning Technologies, Katowice, 2006, pp. 140-147.
10. Kuo-En C., Bea-Chu C., Sei-Wang C.: *A Programming Learning System for Beginners-a Completion Strategy Approach.* IEEE Transactions on Education, VOL. 43, No 2, 2000. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00848075
11. Piecha J., Król R.: *The sequential database attributes for interactive applications management.* Journal of Medical Informatics and Technologies, Vol.3/2002, pp. 129-134.
12. Klir G. J. and Yuan B.: *Fuzzy Sets and Fuzzy Logic: Theory and Application.* Prentice Hall, New York, 1995.
13. Piecha J. Król R.: *The MAMS an interactive applications management engine.* Proc. of Int. Distance Learning Workshop, Katowice, 2004, pp. 25-37.
14. Marcelino M., Mihaylov T., Mendes A.: *H-SICAS, Handheld Algorithm Animation and Simulation Tool to Support Initial Programming Learning.* 38th ASEE/IEEE Frontiers in Education Conference, New York, 2008. http://fie-conference.org/fie2008/papers/1544.pdf
15. Piecha J.: *Principles of e-learning applications characteristics and evaluation procedures.* Proc. Distance Learning Workshop, Katowice, 2004, pp. 76-82.
16. Motwani R., Raghavan P.: *Randomized Algorithms.* Cambridge University Press, 1995.
17. Carchiolo V., Longheu A., Malgeri M., Mangioni-Rusing G.: *Web-Based Pesonalized Learning System in Academic Context.* Proceedings of the Fourth International Conference on Computer and Information Technology (CIT), 2004. http://ieeexplore.ieee.org/ielx5/9381/29791/01357216.pdf?arnumber=1357216
18. Banachowski L., Diks K., Rytter W.: *Algorytmy i struktury danych*, Wydawnictwa Naukowo - Techniczne, Warszawa, 2006.
19. Furas Ł., Piecha J.: *The interrupt services simulation for computer subroutines in on-line learning system.* Proc. of Int. Conf. ICT in Education, Rożnov, 2005, pp. 155-161.
20. Piecha J., Bernaś M., Furas Ł.: *The e-content structure fundamentals in the example application of an on-line learning system.* Proc. of Int. Conf. ICT in Education, Rożnov, 2005, pp. 182-188.
21. Forbus K. D., Hinrisch. T. R.: *Companion cognitive systems: A step toward human-level AI.* AI Magazine Vol. 27, 2006, pp. 83–95.

22. Piecha J.: *Web-training resources construction principles and their interactive links with content distribution platform.* Advances in Soft Computing Springer Verlag, Berlin, Heidelberg, 2007, pp. 292-297.
23. Tadeusiewicz R., Ogiela M. R., Ogiela L.: *New Approach to the Computer Support of Strategic Decision Making in Enterprises by Means of New Class of Understanding Based Management Support Systems.* Proceedings 6th International Conference on Computer Information Systems and Industrial Mangement Applictaions, IEEE Computer Society, California, 2007, pp. 9 – 13.
24. Piecha J.: *The distribution and content platforms coordination for applications management.* Research Reports on e-Content and Distribution Platforms Technologies, Katowice, 2006, pp. 132-139.
25. Pawełczyk P., Piecha J., Bernaś M.: *The content protection services development for distance learning network.* Research Rep. on e-Content & Distribution Platforms Technologies, Katowice, 2006, pp. 141-148.
26. Skowron A.: *Approximate Reasoning in MAS: Rough Set Approach.* Proc. of IEEE/WIC/ACM Conf. on Web Intelligence, 2006. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4061335&isnumber=4061322
27. Ahmad R., Rahimi S.: *A Perception Based, Domain Specific Expert System for Question-Answering Support.* Proc. of IEEE/WIC/ACM Conf. on Web Intelligence, 2006, pp. 454-459.
28. Bernaś M., Piecha J.: *Interactions Validation Methods for Training Resources Control Engine Development*. Central European Conference on Information and Intelligent Systems, Croatia, 2008, pp. 312-323.
29. Rutkowski L.: *Metody i techniki sztucznej inteligencji.* Wydawnictwo Naukowe, PWN, Warszawa 2006.
30. *Moodle overview*. http://docs.moodle.org/en/About_Moodle.
31. *Specification: SCORM 2004 3rd Edition Overview*. Advanced Distributed Learning 2006. http://ADLnet.gov.
32. *Specification: SCORM 2004 3rd Edition Content Aggregation Model (CAM) Version 1.0.* Advanced Distributed Learning 2006. http://ADLnet.gov.
33. *Specification: SCORM 2004 3rd Edition Run-Time Environment (RTE) Version 1.0.* Advanced Distributed Learning 2006. http://ADLnet.gov.
34. *Tomcat 5.0 specification*. www.jacarta.tomcat.com.
35. *Java 1.5 development guide*. www.sun.org.
36. *Fuzzy engine project*. www.fuzzy.sourceforge.net.
37. *Graph basic library*. Jgrapht project. sourceforge.com.
38. *Apache2 system specification*. www.apache.org/documentation
39. Eckel B.: *Thinking in Java, 4'th edition*. Prentice Hall, New York, 2004.
40. *Statistical measures library*. www.jscsi.com.
41. *European Thesaurus of Education System*. www.eurydyce.org.
42. *Concurrent Versioning System for Moodle*. http://docs.moodle.org/en/CVS_%28developer%29.
43. Cichosz P.: *Systemy uczące się*. Wydawnictwa Naukowo-Techniczne, Warszawa, 2000.

Received 14.11.2008; accepted in revised form 12.06.2009