Classifications' Constructing by Means of a Task-Oriented Environment

I. Zheliazkova, A. Borodzhieva

Key Words: *Classification; constructing; task; UML project; implementation; teacher's tool; preliminary study.*

Abstract. *Classifications are frequently used declarative knowledge units that facilitate understanding different subject concepts by professionals and researchers. Almost each lecture course includes different classifications so their effective teaching and learning by means of special purpose applications is of research interest. The current paper deals with a Task-Oriented Environment for Constructing Classifications to support both teacher and learner. An ontology model of a task for this purpose and mining its pedagogical parameters from teacher's and learner's side are proposed. The results of a preliminary study for verification of the proposed metrics are also discussed.*

1. Introduction

Classifications are abstract concepts introduced for the human's better understanding, systematization, recognition, comparison, and contradiction of real world objects. Together with other declarative knowledge units, such as definitions and descriptions they form the corresponding professional language. The popular object-oriented methodology in the software engineering domain relies on the concepts class, object, and class diagram. In INTERNET for many professional domains classifications are accessible with a big number of classes, hierarchical structure, and hyper-links.

However only one commercial application CA-CL [16] was found for classifications constructing and using in different domains. This application offers a lot of user functions from simple (rename, change, merge and split classes) through moderate (checking for the classification status, single and multiple attributes, class hierarchy, inherited attributes, restriction of the assumed values) to complex (inconsistency in the class hierarchy, multiple inheritance, syntax in the object dependences, executing string search). From the teaching point of view some of the functions are not so important, others not easy to use. But the main disadvantage of CA-CL is that there are no relationships among different types of classes.

For the needs of researchers Dubois et al. [4] introduced the concept fuzzy classification. The difference between this type of classifications and the professional ones in that the degree of belonging of a sub-class to the superclass, as well as of belonging of an example attribute to the typical one, can be less than 1. The main proposal of the cited authors is a combination of the object-centered presentation with the theory of probabilities. The object-oriented presentation allows to create a compact, understandable and easy for processing a homogeneous knowledge base, making inferences in a way similar to the expert's one. The probability theory ensures appropriate reasoning to cope with the intypicality, uncertainty, indeterminacy in these knowledge units. The above mentioned features are far from the real human's model of the classifications and the way they are presented and manipulated in the teacher's mind.

The development of common cognitive skills for classifications begins in the primary education and continues in the secondary one. A taught classification consists of a limited number of classes, attributes, and examples and a good school practice of classifications teaching includes principles and recommendations for a test-like lesson [19].

Obviously the teacher needs an intelligent tool to construct qualitative classifications, compare the learner's classification with the expert's one, provide precise knowledge assessment and diagnostics, as well as assign subtasks for misconceptions remedial. The only system found for these needs in the primary schools is EpiList I [7]. Its first version was based on instruction without loop, e.g. the learner explicitly performs classification tasks from particularity to generality in the form of a game. The system explains the learner's errors implicitly by means of comparisons and generalizations. The second version EpiList II presents a more effective adaptive control system with a close loop [11] that explicitly monitors the learner's skills for inferences and instruct him/her for their acquisition. The system registers how many times the learner has attempted correctly or incorrectly to take part in the game. In such a way EpiList II monitors 11 skills, namely: generalization, comparison, and contradiction with three levels of difficulty, as well as positive and negative examples. An overlay learner's model is embedded to identify four levels of cognitive competency (skills missing, unknown competency in one skill, insufficient competency, and competency).

In the higher education to instruct subject classifications of the basic objects (algorithms, programs, languages, schemes, etc.) the lecturer relies on the learned common cognitive skills for classifications. In the lecture material these objects are classified according to different features and each class is connected with typical attributes, their values and examples. Such a classification is presented by the lecturer usually in a mixed style, e.g. as a text fragment, table, classification tree. The relationships to sub-classifications of the produced objects with other classifications of the same objects but according to other features are reminded. The learned classifications facilitate understand-

ing, application, analysis, and synthesis of the procedural knowledge units, such as formulae, algorithms, schemes, systems, etc.

It is not clear enough from the known e-learning systems if the classifications present separated lecture resources, constructing by both lecturer and student. The current paper deals with a subject-independent Task-Oriented Environment for Classifications Constructing (TOECC) aimed to improve the process of their teaching and learning. In the next section a platform-independent architecture of the environment is presented. Then an ontology model of a task for classification constructing and the concept qualitative classification are introduced. Section 4 deals with mining the author's pedagogical parameters, and sections 5 describes the implementation of the teacher's tool prototype. An algorithm for generation of subtasks is proposed in section 6. The learner's pedagogical parameters are presented in the next section. Section 8 contains the results of a preliminary study for verification of the proposed metrics on the programming languages classification are given. Finally, the authors' team results are summarized.

2. Architecture of the Task-Oriented Environment for Classifications Constructing

By the author of the lecture material the TOECC is seen as a repository of classifications separated from the other taught knowledge units such as algorithms, programs, schemes, etc. The environment helps the student to perform the task for classification constructing and its subtasks, as well as to assess his/her performance in comparison with the author's one. TOECC also facilitates the instructor in planning, monitoring, and assessing a test-like practical exercise.

Similar to other task-oriented environments developed by Zheliazkova's team [1,9,13] TOECC architecture consists of standard and specialized tools supporting a homogeneous Knowledge Base (KB) integrated in the environment Data Base (DB) (*figure 1*). Four standard editors are included: a word processor (e.g., MS Word), a text editor (e.g., MS Notepad), a graphics editor (e.g., MS Paint), and a help file creator (e.g., MS HTML Help Workshop). The examples with .bmp files (images, schemes, diagrams) prepared by means of the graphical editor are imported in the Word document of the lecture material. It is hierarchically structured by the author and converted to a .hlp file for a context-dependent help.

The specialized tools are three, called respectively teacher's tool, student's tool, and task manager. The author, student and instructor operate with these tools through a highly interactive and intuitive user interface. By means of the first two tools the author's and student's classification knowledge is extracted and text files with a fixed structure and extension .cls are generated. From the environment's point of view the author's .cls file is seen as a separated text resource. The student's .cls file presents the student's overlay model at the task level.

The task manager extracts the instructor's pedagogical knowledge and store it in a text file with a fixed structure and extension .ecs. It serves as a program/plan for a test-like exercise, invoking subprograms, e.g. .cls files.

The interpreter-evaluator parses the .cls file to compute the author's/task's parameters (knowledge volume, degree of difficulty, planned time). After a construction task is performed by the student, the tool computes his/her results relatively to the author's ones. The tool also provides diagnostics of the student's knowledge refreshing his/her task model.

The author's .cls and instructor's .ecs files are stored in the KB that can easily be extended with other classifications and examples. Such files can be opened and edited by means of Notepad to avoid the slow process of their editing-generating. In such a way different equivalent tasks and exercises can be created for different students and groups.

The main purpose of the task manager is to present and sort the exercise tasks according to instructor's preferences and to interpret his/her directives for intervention during the plan execution. After an exercise finishes, its parameters similar to the task ones are accumulated as statistical experimental data in the Task Base (TB). This base together with the KB and students' models are integrated in the DB. It is supposed to be used for the needs of an environment for individualized planned teaching in different professional domains [19].

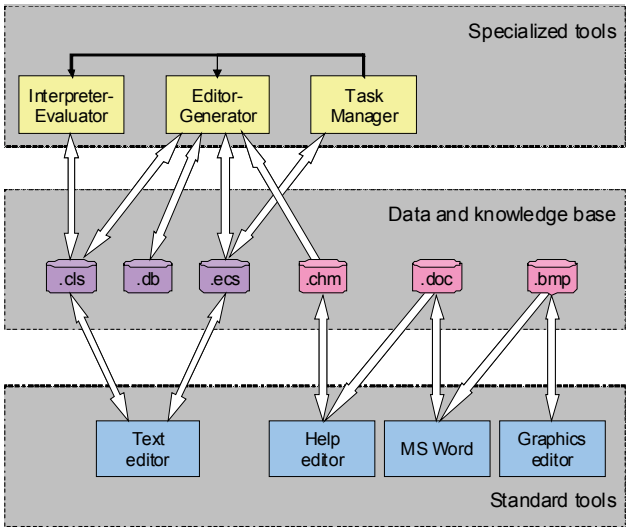


Figure 1. Architecture of TOECC

3. Ontology Model of a Task for Classification Constructing

According to Hendler [17] "An ontology comprises a set of knowledge terms, including the vocabulary, the semantic interconnections, and some simple rules of inference

and logic for some particular topic. Under its implementation the subject contents is integrated with this structure for faster, deeper, and substantial knowledge acquisition." A lot of researchers in the area of ontology-based courseware [2,3,6,8] share his opinion.

A subject-independent ontology model of the task for classification constructing is shown in *figure 2* as a special type of a root tree. The correspondence of its components to the terms of object-centered representation follows:

- The root node c^0 corresponds to the basic class at the hierarchical level 0, it has no super-class and it is filled with the object name plus the feature name for classification;
- Its right nodes-children $\{a_1^0, a_2^0, ..., a_m^0\}$ at the same level present the rang A^0 of the typical attributes values, specified for that abstract object and are filled with string values;
- Its left nodes-children $\{p_1^0, p_2^0, ..., p_r^0\}$ at the same level present the set P^0 with other classifications of the same object but according to other features and are filled with pointers to those knowledge blocks;
- Its lower children $\{c_1^1, c_2^1, ..., c_n^1\}$ at the next level 1 present the list C^1 of sub-classes and are filled with their names. Generally each internal node, for example, C_i^1, corresponds to a sub-class, has a list A_i^1 of right children $\{a_{i1}^1, a_{i2}^1, ..., a_{im}^1\}$ and a list C_{ij}^1 of sub-classes (in *figure 2*) there are no children and a pointer to a knowledge block, for example, p_n^1, presents a pointer to a knowledge block, containing a sub-classification of this object. The terminal nodes $\{r_{i1}^2, r_{i2}^2, ..., r_{in}^2\}$ of a super-class C_i^1 correspond to a set R_i^2 of example representatives of the supper-class and are filled with the names of the real entities.

In a classification tree $T = (N, L)$ the set of nodes N consists of four intersecting sets, filled respectively with classes, attributes, representatives (examples), pointers and images for a concrete classification: $N = C \cup A \cup R \cup P \cup I$. The set of arcs L of this tree also consists of four intersecting sets E,T,B,X, including respectively deepness, wideness, practical, and external relationships, e.g. $L = E \cup T \cup B \cup X$.

The kinds of relationships in the ontology model in *figure 2* that are four will be discussed hereinafter.

By default, a deepness relationship $e_k = < c_i, c_j >$ between a super-class c_i and its children class c_j is "divided-into" relationship. An wideness relationship $t_k = < c_i, a_j >$ between class c_i and attribute value a_j is "has-a-property" relationship. A practical relationship $b_k = < c_i, r_j >$ between a class c_i and its example r_j is of type "is-represented". An external relationship $x_k = < c_i, p_j >$ between a class c_i and a pointer p_j to the knowledge block is of type "is-related-to".

The process of classification constructing can be viewed as filling the empty nodes (slots) of the classification tree with the elements of the subsets C, A, R, P, I. This mental activity is performed first by the author, as a source of the knowledge, and later by the student, as a receiver of the knowledge. For this purpose the checking of the tree structural correctness is more than desirable. In the context of classifications this means: validation of the main features for a root tree, namely: connectivity, availability of a root node, reachability of each terminal node through a simple path and acyclity. Secondary, two special restrictions also have to meet each classification tree: 1) Each element of the above-mentioned subsets has to be pointed out only one time as super-class; 2) Only the four early mentioned relationships types are permitted.

The well-known classical algorithms for the root tree editing, searching and sorting cannot be directly applied to the proposed ontology model. There is a need of a new interactive algorithm, taking into account not only the special features of the classification tree but mining the pedagogical parameters values from the author's and student's performances too.

4. Mining the Author's Pedagogical Parameters

It is reasonable to compute the knowledge volume Q_1 in the author's classification as the sum of both nodes and relationships numbers: $Q_1 = |N_1| + |L_1| = 2.|N_1| - 1$. The last is a result from the main statement of the common theory of graphs according to which $|L_1| = |N_1| - 1$.

It is supposed that the subsets C_1, A_1, R_1, P_1, I_1 are viewed when the task for constructing is performed by the student. This means some degree of system's prompt, respectively decreases the system's confidence in his/her knowledge. The degree of system's prompt C_p determines what part of the author's knowledge is accessible to the student, when he/she constructs this knowledge unit. In

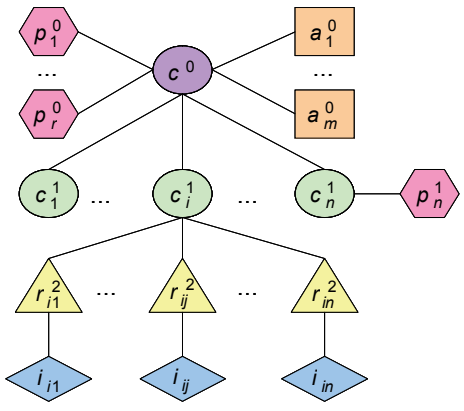


Figure 2. Ontology model of a classification task

case of a classification with significant volume C_p is determined as $C_p = |N_1| / Q_1 = |N_1| / (2.|N_1| - 1) \approx 0.5$. If $Y_j(c_i)$, $j = 1,...,5$, means that local subsets respectively $E_1(c_i), T_1(c_i), B_1(c_i), X_1(c_i)$, c_i $(i = 1,...,n)$, belonging to the i_{th} class, then the quality of the author's classification is defined as $C_q = \sum_{i=1}^{n} \sum_{j=1}^{5} \sigma[Y_j(c_i)]$, where the function

$$\sigma[Y_j(c_i)] = \begin{cases} 0, & \text{if } Y_j(c_i) \neq \varnothing \\ 1, & \text{if } Y_j(c_i) = \varnothing \end{cases}.$$

When $\sum_{i=1}^{n} \sigma[Y_1(c_i)] > 0, (j = 1,...,5)$ the classification is well balanced respectively its deepness, spacity, usefulness and connection with other classifications of the same object, but according to other features. Informally this means availability at least of one connection from each type, that well concurs with the human presentation about good classifications. In particular cases when: 1) $\sum_{i=1}^{n} \sigma[Y_2(c_i)] = 0$ classifications can be called non-spaced; 2) $\sum_{i=1}^{n} \sigma[Y_3(c_i)] = 0$ classifications can be called non-practical; 3) $\sum_{i=1}^{n} \sigma[Y_4(c_i)] = 0$ classifications can be called isolated. By definition $\sum_{i=1}^{n} \sigma[Y_1(c_i)] > 0$, close but never equal to 1, as a classification tree has at least two terminal nodes, e.g. without children.

The main temporal parameter of author's performance is the time of constructing, i.e. the time interval D$t1 = t1f - t1s$, where $t1s$ ($t1f$) is the current system time when teacher starts (finishes) the classification constructing. The main derivative parameter is the rate of constructing, defined as $v1 = Q1/Dt1$. It is a function of the author's personal characteristics and the system's user interface too.

5. Implementation of the Author's Tool Prototype

Nowadays the most popular subject-independent system for creating different types of ontologies is Protйgй [2] but some difficulties and restrictions have been found using it for the teaching purpose [5]. That is why to verify the above-proposed metrics a WINDOWS-based tool prototype has been implemented by the authors of this work. DELPHI 7.0 programming environment has been chosen for this purpose due to highly intuitive visual programming with high degree of code generation and availability of a rich Visual Component Library (VCL) [10].

Before implementing the tool's prototype the UML project of TOECC was developed [9]. For representation of a classification in the computer memory the VCL tree interface components were used. Three kinds of diagrams were drawn to visualize the functional requirements, teacher's activity and dialog forms as special kind of classes. The tool's use case diagram is shown in *figure 3* and the activity diagram in *figure 4*.

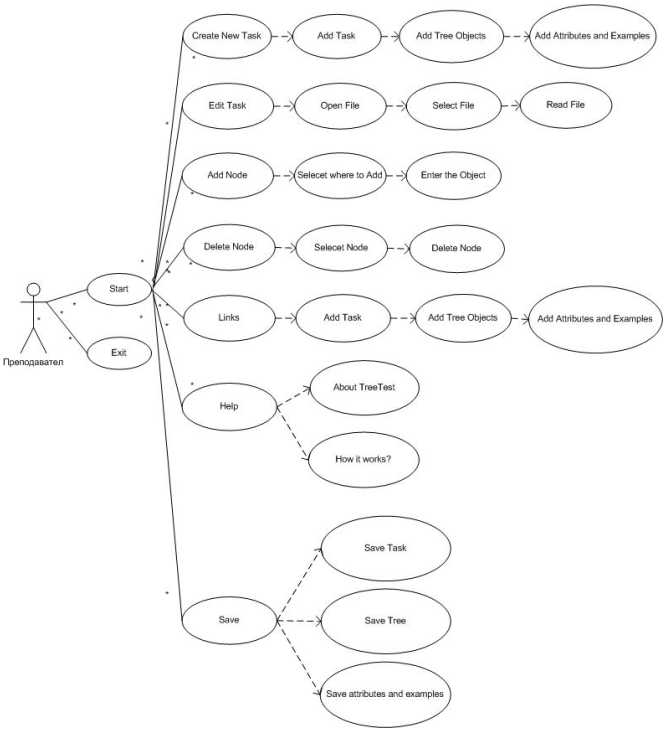


Figure 3. Use Case diagram of the teacher's tool

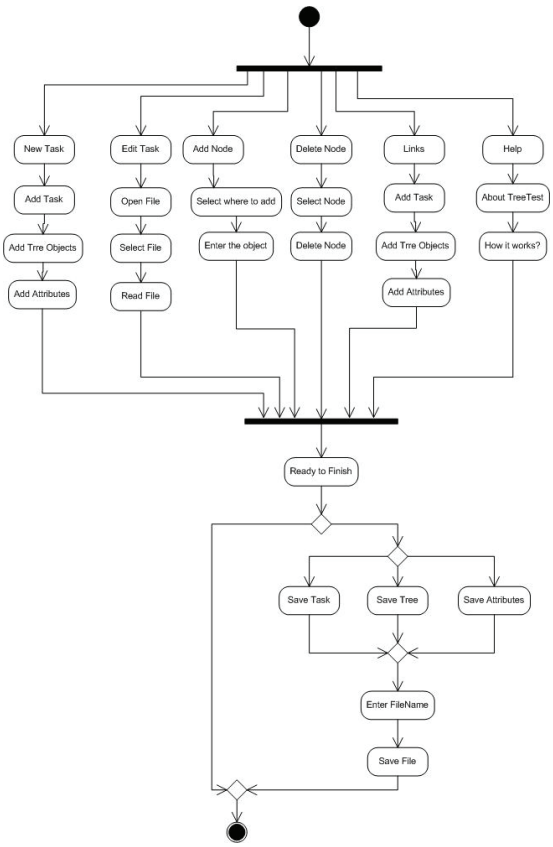


Figure 4. Activity diagram of the teacher's tool

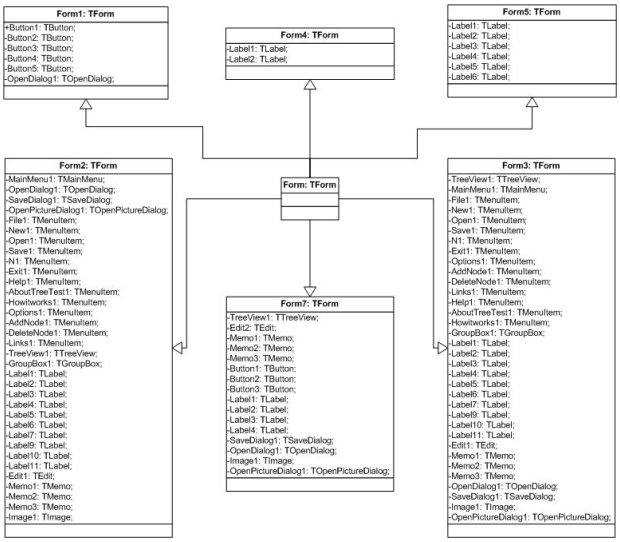


Figure 5. Class diagram of the tool prototype

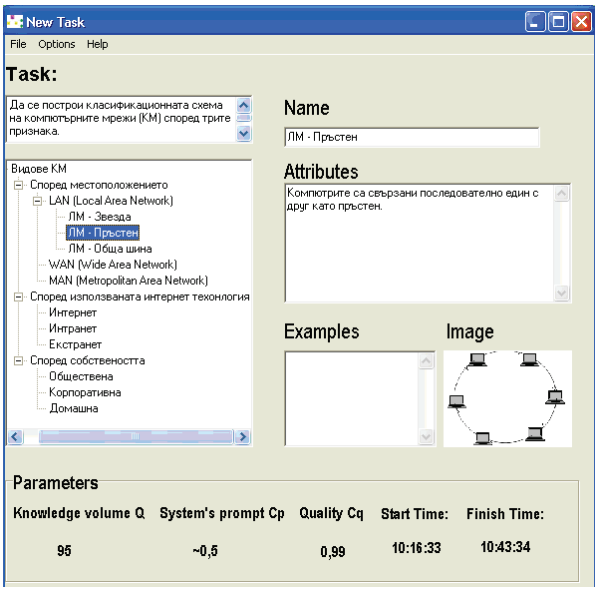


Figure 6. Tool's screen

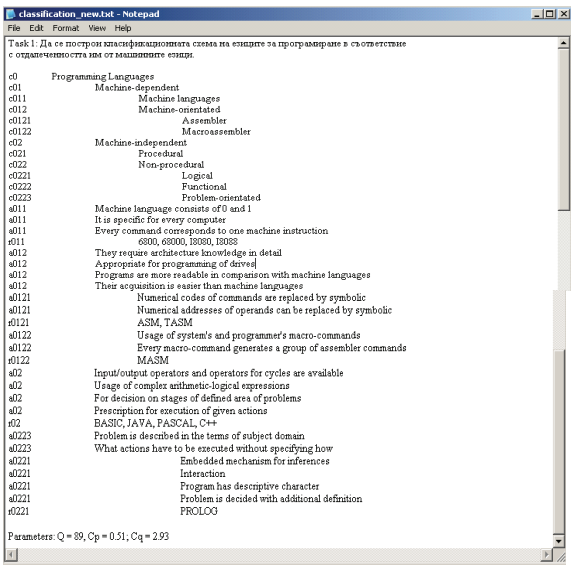


Figure 7. Generated text file

Figure 5 presents the class diagram in which a rectangle has three sections respectively for the class name, attributes, and methods. Almost all methods present event-handlers, invoked when an user action (button pressing, mouse moving, menu choice, timer signal, and so on) occurred.

The tool's user interface is simple and easy for use even by the teachers or non-professional programmers (*figure 6*).

The tool offers to describe the task in a free text format (the field *Task*), and its classification tree. For each node the following data can be set up: name (*Name*), attributes (*Attributes*), representative (*Examples*) and image (*Image*). The menu-item *File* contains standard items for opening an existing file and creating a new file, to save it and exit. From the menu-item *Options* the commands *Add Node* and *Delete Node*, as well as *Links* to other classifications for the same objects but according another feature are available. In the panel *Parameters* the values of the pedagogical parameters (knowledge volume, degree of system prompt, quality of classification, start time and finish time of the constructing task) are given. As a result of the task preparation a standard text file with a fixed format is generated (*figure 7*).

6. Mining the Student's Pedagogical Parameters

As a precise and sensible measure for knowledge acquisition the degree of proximity of the student's classification tree to the teacher's one can be used [14,15]. It is reasonable for an intelligent system to interpret this problem solution as syntax pattern recognition of two classification trees in the range between 0 and 1. This is in line with the statement that the criteria for deciding whether the student has mastered a skill can be described in terms of percentage of a perfect score (e.g., 80%, 90%, 100%). The classical syntax methods in Pattern Recognition turned to be inappropriate for computation of the degree of proximity. Through several formal definitions we will come to an original formula for it.

Definition 1. A syntax node type c is the ordered couple $nci = (i, S)$, where i stands for its number in the list of such nodes and S stands for a text constant. The degree of proximity of two c-type syntax nodes $n1ci$ and $n2cj$, belonging to two different lists $L1$ and $L2$ is $C_b(n1_c^i, n2_c^i) = \begin{cases} 0, & if\ S_1 \neq S_2 \\ 1, & if\ S_1 = S_2 \end{cases}$. If $C_b(n1_c^i, n2_c^i) = 1$, those nodes are called equivalent.

Definition 2. Syntax node type we will call a given couple $L_t = (t, L)$ where $t \in (C, V, I)$ means the type of its elements, and L is a traditional list of the string constants. The power of a syntax list $L_t = |L|$.

Definition 3. The coefficient of the proximity of two syntax lists $L1_t$ and $L2_t$, belonging to different trees $T1$ and $T2$:

$$C_p(L1_t, L2_t) = \frac{|L1| \cap |L2|}{\max(|L1|, |L2|)}$$

Informally, it gives the number of the equivalent syntax lists in the L's tree $T2$ respectively to the maximal power of the powers on the author's list $L1$ and the learner's list $L2$.

Definition 4. An adjacency of a syntax node n_c^i is a list $A^i_c = (n^i_c, L_v, L_p, L_i)$, where each of the lists $L_t (t \in \{C, V, I, P\})$ could be empty and there is a link between each element $n_i \in L_t$ and n_c^i.

The power of the adjacency is $|A_c^i| = 1 + |L_v| + |L_p| + |L_I|$.

Definition 5. The degree of proximity of two adjacencies $A1_c^i$ and $A2_c^i$ belonging to different trees $T1$ and $T2$ is:

$$C_p(A1_c^i, A2_c^i) = \frac{1 + \sum_{t \in \{V,I,P\}} C_p(L1_t, L2_t)}{\max(|A1_c^i|, |A2_c^i|)}$$

Informally it gives the number of the equivalent syntax nodes in an adjacency of the learner's tree in proportion to the maximum of powers in the author's adjacency $A1_c^i$ and of the L's one $A2_c^i$.

Definition 6. A syntax path presents a list of adjacencies $P^i = (i, A_c^1, A_c^2, ..., A_c^{l-1})$, where i is a code identifying a path in a tree. $A_c^j (j = 1,...,l-1)$ belongs to a node at the level j and has a connection between n_c^k and n_c^{k+1} $(k = 1,2,...,l-1)$. The power of such a path is $|P^i| = l - 1$.

Definition 7. Coefficients of the proximity of semantic paths $P1^i$ and $P2^i$, belonging to two different trees $T1$ and $T2$ ($i = j$ is not obligatory):

$$C_p(P1^i, P2^j) = \frac{\sum_{k=1}^{l-1} C_p(L1_t, L2_t)}{\max(|P1^i|, |P2^i|)}$$

Informally, it gives the number of the equivalent nodes in a path of the learner's tree in proportion to the maximal of the powers in the author's path $P1^i$ and $P2^i$.

Definition 8. The semantic tree is a list $T = (P_1^1, P_2^2, ..., P_m^m)$, where m is the number of the paths in this tree, e.g. $|T| = m$.

Definition 9. The coefficient of the proximity of $T1$ and $T2$ is the proportion:

$$R_1 = C_p(T1, T2) = \frac{\sum_{i=1}^{m} C_p(P1^i, P2^i)}{\max(|T1|, |T2|)}$$

Informally, it gives the total number of the equivalent nodes in the learner's tree in proportion to the maximal of the powers of the author's tree $T1$ and the learner's one $T2$.

Definition 10. A syntax node represents a misconception, if it exists as a child class for a given parent class in the student's classification, and does not exist as a child class for the same parent class in the teacher's classification. If m is the number of such nodes then the function of misconception excess is

$$R_2 = \begin{cases} 0, \text{ if } m = 0 \\ 1 = (R_{2min} - 1)(m/n)^f. \end{cases}$$

Here $R2_{min}$ corresponds to the minimal value of R2 under $m = n$, and the value of f defines the kind of the function change (*figure 8*).

By means of $0 \le R2 \le 1$ the knowledge evaluation R1 = Cp(T1, T2) can be decreased thus punishing the L for his/her missing and/or wrong knowledge.

Definition 11. The function R3 of rudeness of the misconception excess is

$$R_3 = \begin{cases} 1, \text{ if } m = 0 \\ 1 - b^{a-L} \end{cases} \quad \text{where} \quad L = \frac{\sum_{i=1}^{m} Level(c_i)}{m}$$

is the average level of misconception excess. The constant a is interpreted as the minimal value of R3 under the minimal value of L, and the constant b as the maximal value of R3 under the maximal value of learner (*figure 9*). The rudeness of the misconceptions at the higher levels is greater than those at the lower levels.

Definition 12. The final assessment R0 = R1·R2·R3 will give more precise and sensible L's knowledge evaluation as it takes into account three indicators: correct knowledge, misconception excess, and rudeness of the misconception excess.

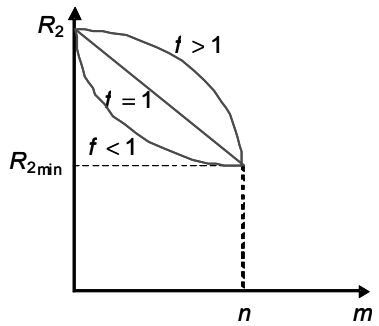


Figure 8. Graphical interpretation of misconception excess

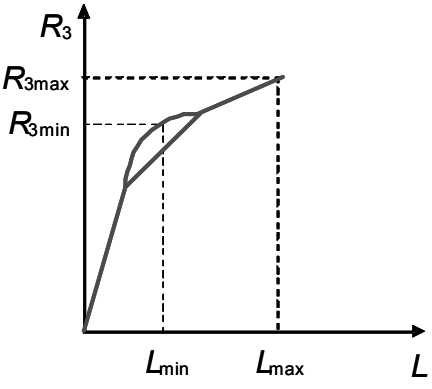


Figure 9. Graphical interpretation of its rudeness

7. Generation of Subtasks Formulation

The subtasks levels in increasing order of their complexity may concern: a node, list of nodes, occurrence, path of occurrences, and sub-tree of paths. The same levels of the *L*'s errors will be also valid in increasing order of their roughness. Normally all levels of the errors are expected in the *L*'s performance. That is why of interest is the generation of the subtask formulation in the limited natural language and strategy for the errors remedial under limited time. *Table 1* presents in pseudo-code subject and language independent frames for generation of the subtasks formulation at different levels. $S_k(k=1,...,4)$ stands for the text constants; &I a slot for the system variable denoted as $_i$; + symbol for concatenation of the text constants.

Together with computation of the coefficient of proximity a typical list L_l is generated with the values of the micro variables for a given level of errors.

The structure of its elements depending on l is also given in the table. The order of the elements as well as their

Table 1. The algorithm for remedial in pseudo code

```
IF performance = failed
    Generation of a step-by-step report
    Current level := highest
    WHILE (rest time = enough)
    AND (current level > lowest -1) DO
        Current subtask := first subtask;
        WHILE (rest time = enough) AND
        (current subtask > last subtask -1) DO
            Generating current subtask;
            IF current subtask = failed
            THEN
              Consultation with the teacher;
            ENDIF
            Current level := next level
    ENDWHILE
ENDIF
```

order under the concatenation depends on the used natural language. In this relation some of the slots S_k could be empty. The learner's performance is successful, if there are no errors discovered. In other case a detail report for the missing and wrong relationships is generated. The remedial begins with the first error at the higher level, for which remedial the rest of time will be enough.

A level will be generated, if he/she has enough time to do it. Under the assumption for a constant rate of the knowledge acquisition during the session the rest time is $t_r = t_p - t_c$, if it is bigger then $(Q/Q1).t_c$, where Q is the knowledge volume of a given subtask.

In *table 2* a simple algorithm is perceived for formulation of a subtask $d(l)$ for classification at a given level *l*. If the current subtask has been performed unsuccessfully, the learner possibly needs a consultation with the teacher. When all subtasks at the given level are performed by the

Table 2. Algorithm for generation of a subtask

```
d(l) := ''
Current operand := first operand;
WHILE current operand ≠ last operand + 1 DO
    IF current operand = S_k THEN
            d(l) := d(l) + S_k
    ELSE
            Extracting the value of the micro variable
            &v from L_l;
            d(l) := d(l) + v;
    ENDIF
    Current operand := next operand;
ENDWHILE
```

L, he/she will get lower level. This process finishes when the planned time is over, or the last subtask at the lowest level is performed. The following algorithm in pseudo-code models more precisely the above-described heuristics strategy for remedial.

8. A preliminary Study

To put the environment in the practice a preliminary study was carried out with the aim to test the validity of the formulae for the author's and student's parameters. The subject area *Programming Languages* was selected. The author's task performance was simulated by a professional programmer during one computer session. Three variants of the programming languages classification were chosen for comparison (*figure 7*): the first one – class hierarchy plus attributes' values and instances, the second one – class hierarchy plus attributes' values, and the third one – only class hierarchy. There were no pointers and images in the three variants.

The results shown in *table 3* for knowledge volume are in the scope of the human teacher expectation. The system's prompt is approximately constant as the knowledge volume is considerable for the tree variants. The quality of the first variant is highest but less than 1 as for some classes attributes' values are absent and the classification is isolated. The quality of the third variant is lowest as it does not contain any knowledge about attributes and representatives. The time of constructing decreases as knowledge volume decreases, and the rate of constructing is approximately constant as it is supposed that the personal

Table 3. The computed pedagogical parameters

Variant	Knowledge volume, Q_1	System's prompt, C_p	Quality, C_q	Time, min:sec	Rate, min^{-1}
First	103	0.504	0.875	11:45	8.7
Second	87	0.505	0.500	20:30	8.3
Third	23	0.521	0.208	02:33	9.2

Table 4. The computed parameters

Parameter	Constant	Value
R2	f	1
R2	$R2_{min}$	0.5
R3	L_{min}	1
R3	L_{max}	7
R3	$R3_{min}$	0.9
R3	$R3_{max}$	0.99

characteristics could not be changed during one short session. The figures give the descriptive knowledge representation in the way the system stores it in a standard text file.

The validation of the formulae for computing the parameter R0 of the *L*'s performance under the accepted values of the parameter constants (*table 4*) also was tested.

The performances were simulated by an undergraduate student during one computer session. The author's classification tree for this experiment is given in *figure 10*.

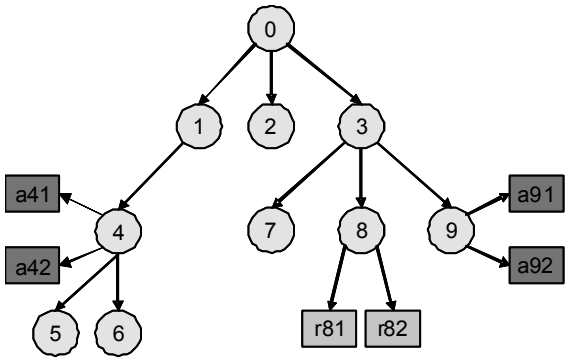
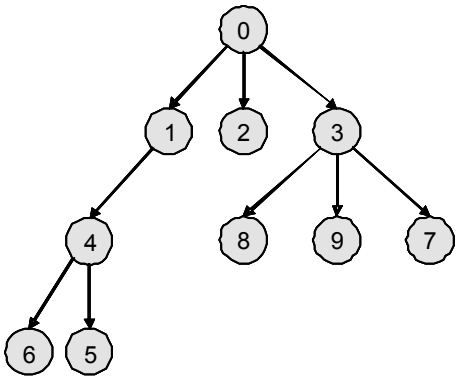
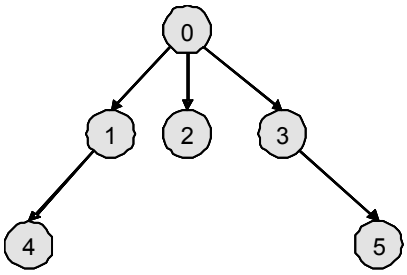


Figure 10. Experimental *A*'s classification tree

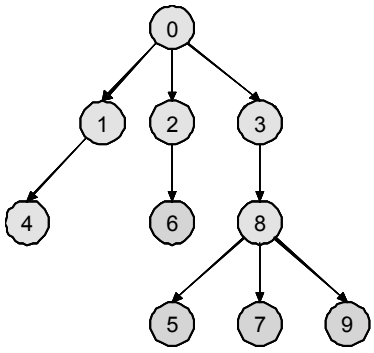
In this and next *figures 11a-f* a recognized class node is represented by a numbered circle, a recognized attribute node *j* to class node *k* by a rectangle named akj, a recognized instance node *j* to class node *k* by a rectangle named "ikj", a misconception class node by a thick circle, a misconception attribute or instance node by a thick rectangle. *Figures 11a* and *11b* illustrate the influence of the recognized class nodes on the degree of proximity R0. For *figure 11a* it is less but closer to 1.00 because all class nodes had been recognized by the *L*. For *figure 11b* it is less because only a part of the class nodes had been recognized. The *L* is punished with a decrease of the evaluation when the *L*'s classification tree has misconception class nodes (*figure 11c*). *Figures 11c* and *11d* demonstrate the influence of the position of the misconception class nodes in the hierarchy. The conclusion is that at higher levels this influence is stronger. Examples in *figures 11e* and *11f* show the weaker influence of the misconception attribute and instance nodes on the *L*'s evaluation.



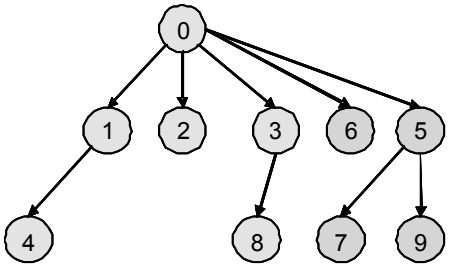
a) $R_0 = 0.850$



b) $R_0 = 0.500$



c) $R_0 = 0.430$



d) $R_0 = 0.416$

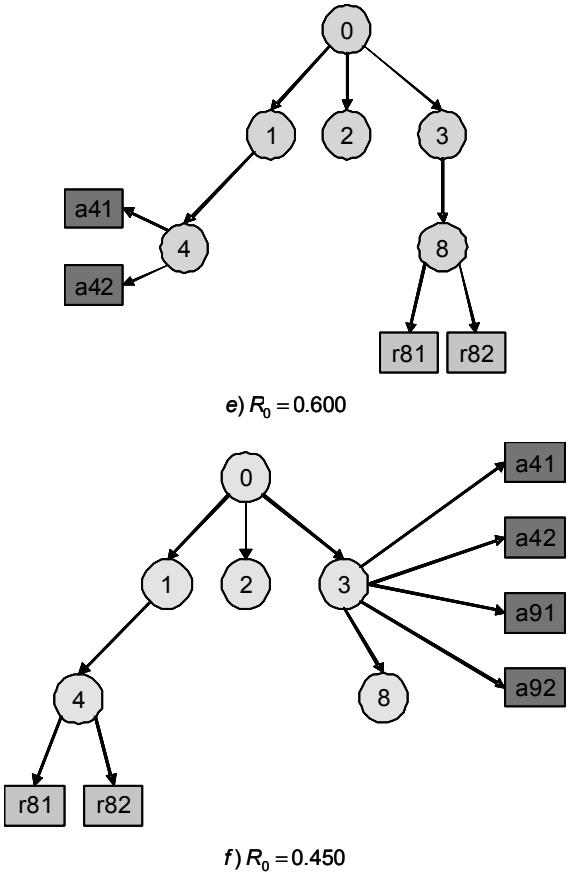


Figure 11. Different *L*'s task performance

To find the computed results correlation with the human's ones 10 teachers (T1 ÷ T10 columns in *table 5*) were asked to order the *L*'s trees by their proximity to the *T*'s tree. The conclusion made is that the measure (the first column M) reproduces well the "correct" ordering of the teachers.

Table 5. The ordering of the Ts' trees

M	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
a	a	a	a	a	a	a	a	a	a	a
e	e	e	e	e	e	e	e	e	e	e
b	b	b	b	b	b	b	b	b	b	b
f	c	f	c	c	c	c	f	f	c	f
c	f	c	f	f	f	f	c	c	f	c
d	d	d	d	d	d	d	d	d	d	d

9. Conclusions

The ontology-based task for classification constructing is performed by the student in a way closer to the author's one. Metrics for computation of several pedagogical parameters are proposed for both author's and student's performances. These parameters can be useful for comparison of the classifications of different authors, controlling the step execution of the student's plan, setting up the goal of a lecture course, generation of a more realistic individual plan and its correction during execution.

TOECC has a well-organized, structurally correct and compact homogeneous knowledge base, task base including generated subtasks for the student's misconceptions remedial. The precise and sensible evaluation of the student's performance relatively to the author's one is precise and sensible, and takes into account a set of parameters. The instructor is involved not only in exercise monitoring, but in setting the input parameters, and analyzing the results of the student's task and exercise performance.

The tool prototype is subject-independent and requires elementary computer competency from the author. The classifications storage in a text file separated from other knowledge units allows their coauthoring, reusing as well as and integration in a web-based environment for lecturing.

Acknowledgments

The authors are grateful to Merel Iumer, a student in the specialty Computer Systems and Technologies from the University of Ruse for program implementation of the tool's prototype in the framework of her bachelor degree diploma project.

References

1. Atanasova, P. V. Creating and Using a Task-Oriented Environment for Structural Schemes Constructing. Theses of PhD Dissertation, Ruse University, 2012 (in Bulgarian).

2. Delijska, B. A. A Subject Ontology of Computer Networks. – *Journal of Computer Engineering*, 1, 2007, No. 2, 32-36.

3. Devedzic, V. Education and the Semantic Web. – *Int. J. Artificial Intelligence in Education*, 14, 2004, 39-65.

4. Dubois, D., H. Prade and J. P. Rossazza. Vagueness, Typicality, and Uncertainty in Class Hierarchies. – *Int. J. Intelligent Systems*, 6, 1991, 167-183.

5. Eremin, E. About Application of Ontologies for Representation of Course Programs. Proceedings of the Second International Conference „Modern E-learning", Varna, Bulgaria, 2007, 41-47 (in Russian).

6. Geleverya, T., O. Malinivskaya, T. Gavrilova, M. Kurochkin. System VITA II for Prototyping Teaching Courses on the Base of Ontologies. International Conference „Modern E-Learning", Varna, Bulgaria, 2006, 123-129 (in Russian).

7. Goh, G. M., C. Quek, D. L. Maskell. EpiList II: Closing the Loop in the Development of Generic Cognitive Skills. – IEEE Transactions on Systems, Man, and Cybernetics – *Part A: Systems and Humans,* 40, July 2010, No. 4, 676-685.

8. Gribov, V. V., A. V. Tarasov. Ontology Model of the Subject Domain „Graphical User Interface". – *Russian Journal of Intelligent Systems*, 2005, No. 1 (9), 80-90 (in Russian).

9. Georgiev, G. T., I. I. Zheliazkova. Representation and Processing of Domain Knowledge for Simulation-based Training in Complex Dynamic Systems. – *Int. J. Intelligent Systems,* 2000, No. 10, 255-277.

10. Faronov, B. B. DELPHI – Programming in a Higher Level Language. Sankt-Peterburg, Piter, 2004 (in Russian).

11. Rastrigin, L. A. Teaching as a Control Process. Cybernetics Methods and Tools in Teaching Process Control in a Higher School. Riga Polytechnic Institute, 1985, 16-36 (in Russian).

13. Atanasova, G. E., I. I. Zheliazkova. An UML Project of a Task-Oriented Environment for Teaching Algorithms. – *Information Science & Computing,* 2008, No. 6, 31-38.

14. Zheliazkova, I. I. Knowledge Constructing of Class Hierarchies. Proceedings of SAER'95, Varna, Bulgaria, 1995, 367-371.

15. Zheliazkova, I. I. Evaluation of the Teacher's Classifications for the Needs of an Intelligent Teaching and Learning Support Environment. Proceedings of SAER'96, 1996, Varna, 255-259.

16. http://help.sap.com/printdocu/core/print46c/en/data/pdf/CACL/CACL.pdf.

17. http://en.wikipedia.org/wiki/James_Hendler.

18. www.indiana.edu/~idtheory/document/m3.doc.

19. iaeipt.com.

Manuscript received on 12.8.2012

Assoc. Prof. ***Irina Zhelyazkova-Valkova*** *is with the Department of Computer Systems and Technologies, Ruse University. She graduated from the Kiev Polytechnic University in 1973. She defended her PhD thesis at the same university in 1976. Her present interests are in the area of discrete structures and modeling, e-learning and e-teaching, intelligent learning environments, software engineering.*

Contacts:
University of Ruse
8 Studentska Street
7017 Ruse
Bulgaria
e-mail: irina@ecs.uni-ruse.bg

Assist. Prof. ***Adriana Borodzhieva*** *is graduated from Ruse University, Department of Telecommunications in 1998. At the present time she is an assistant professor at the same department. She is a PhD student in the Department of Informatics and Information Technologies. Her present fields of interest include communications circuits, logic design, channel coding and cryptography, and digital signal processing.*

Contacts:
University of Ruse
8 Studentska Street
7017 Ruse
Bulgaria
e-mail: aborodjieva@ecs.uni-ruse.bg