UDC 004.8: 004.421.2
DOI: 10.20535/SRIT.2308-8893.2021.4.02

INFORMATION TECHNOLOGY FOR CREATING INTELLIGENT COMPUTER PROGRAMS FOR TRAINING IN ALGORITHMIC TASKS. PART 1: MATHEMATICAL FOUNDATIONS

A.S. KULIK, A.G. CHUKHRAY, O.V. HAVRYLENKO

Abstract. The existing education system (in particular higher education) due to its focus on basic knowledge is quite inert and cannot satisfy the needs of the modern labor market, which is rapidly developing. Some professions transform or disappear, while the others appear almost every day. Today the employers need specialists with certain skills and abilities, who are able to develop them and adapt to specific projects. That is why short-term courses are very popular today, especially online and with a mentor — a specialist in a particular field. At the same time, graduates of such courses are mostly unable to solve complex problems and make competent decisions on their own. There is a requirement of creation of training programs for testing the development and implementation of tools for productive knowledge and skills transferring in a particular field. The article shows a possible approach to provide some interactivity to computer tutoring tools in addition to the game principle, information visualization and other techniques that have already proven themselves in information systems. It will give an opportunity to create a platform that can accumulate new technologies, integrating them into a digital tutoring environment that can be adapted to each student..

Keywords: intelligent tutor system, algorithmic tasks, diagnostic models, bayesian networks, student model.

INTRODUCTION

The exponential growth of knowledge and skills which labor market demands requires new approaches to training and its intensification to ensure the necessary quality of training and retraining. The traditional approach cannot provide such quality, as it is characterized by a number of destabilizing factors. Among them: disturbing influences on students and mentors, weak professional and pedagogical training of individual mentors, low starting level of knowledge and skills and insufficient motivation of individual students. Moreover, in the conditions of specialists group education, traditional training cannot be adaptive. This follows from the objective laws of our brain, which limit our perception of seven (plus or minus two) static objects and three simultaneously solved problems [1].

Promising areas for solving this problem are a systematic approach to poorly structured processes [5, 10], the use of principles of rational control of complex systems in conditions of uncertainty [2] and the creation of software for intelligent tutor systems (ITS). Such programs can have virtually unlimited memory and performance resources for the effective formation of professional competencies and be characterized by high adaptability to the specifics of a particular learner.

The first theoretical research in the field of ITS dates back to the fifties of the last century. For now leading centers of the computer-based tutoring tools development are Carnegie Mellon University, the University of Pittsburgh, USA, Canterbury University, New Zealand. For example, Carnegie Mellon created the product Algebra Cognitive Tutor [3], a feature of which is intelligence tutoring to solve mathematical problems. The University of Canterbury has created a product SQL-Tutor [3] to teach how to compile SQL-queries. However, none of the developed ITS has yet approached the effect of individual studying, described in 1984 by American scientist B. Bloom: the average success of students individually may be better than the success of 98% of students in the traditional form: one mentor for thirty people [3]. Existing approaches to ITS are characterized by a number of theoretical and, consequently, practical shortcomings. For instance, there are some difficulties in the development of flexible enough production rules which required in a cognitive ITS for comparison with the result obtained by the student. Besides this, there is no reliable solution of the first and second kind errors appearance in the ITS class using constraint-based modeling approach (CBM). Therefore, the development and experimental research of new approaches, ITS models and methods remain open. It is also clear that unique universal approach to the ITS creation for any field of activity cannot be proposed.

At the National Aerospace University "Kharkiv Aviation Institute" scientists of the aircraft control systems and mathematical modeling and artificial intelligence departments since 2004 certain steps in the development, implementation and improvement of ITS in various disciplines have been making. Developments are based on an approach to the rational control of objects in conditions of partial uncertainty [2]. As a subject area of study in this article algorithmic tasks (AT) which are characterized by properties of determinism, mass and efficiency are considered [3]. The authors have identified two main classes of AT: 1) algorithm execution tasks (calculation tasks); 2) algorithm development task.

Goal and objectives. The goal of the work is to improve the quality of training in AT solving by training individualization improvement through the creation and implementation of a set of principles, models, methods, algorithms and software for each process stage of knowledge and competencies transferring.

According to the goal it is necessary to perform:

1) analysis of the of the ITS development problem state;

2) development of ITS concept and principles concerning AT solving;

3) development of a task model and student model for computational algorithmic task in the demonstration and training mode, compiled-interpreted ITS model, data models of ITS, models of the training process in ITS, classification models of algorithms and SQL-queries constructed by the student;

4) development of the automatic assignments generation and automatic task performance method for the calculation algorithmic task, the operands omissions

diagnostic models automatic method, a method of addition of new components of knowledge and skills in ITS, the objects fuzzy search method, methods of automatic diagnosing the algorithms and SQL-queries compiled by a student;

5) development of mathematical tools, algorithms and software for ITS in algorithmic tasks solving, their practical implementation and efficiency investigation.

Analysis of existing approaches to the ITS creation. There are various approaches to the automation of tutoring processes – from electronic textbooks to universal platforms for online courses. In this work, modern systems related to mathematical and algorithmic tasks solving were analyzed, including foreign ITS – Algebra Cognitive Tutor, SQL-Tutor, Steve, Andes, AutoTutor and others, as well as native ITS – Term, GRAN, DG and other.

The analysis of modern approaches to ITS creation for a wide class of tasks is carried out. There are two main, most cited, approaches:

1) application of ACT-R theory by J. Anderson of Carnegie Mellon University (USA) to create cognitive ITS;

2) using the approach of S. Olson from the University of Illinois (USA) based on the constraint-based modeling.

The disadvantages of cognitive ITS include: 1) the complexity of production rules set development, needed to lead from the initial data to the reference task for comparison with the solution obtained by the student; 2) for weakly formalized domains or tasks, the development of such a set may be impossible; 3) the buggy-rules' developing complexity for the formation of intelligent feedback in response to errors of the student; 4) the impossibility of forming a complete set of buggy-rules due to the unlimited space of possible mistakes made by students.

Cognitive ITSs have also been criticized for students' freedom limitation and forcing them to think strictly according to the certain task solution approach. In cognitive ITSs, there is a possibility that the student could make a step that does not conform either to the correct rule or to the buggy rule. In this case, it is assumed that the wrong step has made. However, such a step may be absolutely correct, but not in line with the strategy that is embedded in the system. Thus, the correct step of the student may be rejected.

The cognitive ITSs shortcomings were also noted in the 2011 report by the US Department of Education and quoted in the New York Times. Thus, despite the assurances of software manufacturers – the company "Carnegie Learning" that their cognitive ITS provide revolutionary math courses and revolutionary results, a careful analysis of their implementation results showed something else: Carnegie Learning Curricula and flagship software – Cognitive Tutor – have no significant effects in the study of mathematics by high school students in the United States.

The disadvantages of CBM and of corresponding ITS are as follows:

1) inconsistencies in the strategies for solving problems by a student, as the CBM does not support problem-solving strategies, but just assesses a current state of the problem solving (in difference to the current action assessment in cognitive ITS);

2) such ITSs cannot give a clue about the strategy for a problem solving;

3) there is incorrectness and inconsistency of certain restrictions used in ITS;

4) correct solutions might be concerned as incorrect and vice versa;

5) a number of restrictions are deal with syntactic comparison of only one reference solution with the student solution.

Thus, the question of development and experimental research of new approaches to ITS creation for different domains is quite actual. Such approaches should include the both of the above approaches advantages, as well as implement the time-tested and experimental studied other specific ITS intelligence functions and features. Among such functions, the most promising is the organization of the "external cycle", i.e. the choice of the next task for a particular student. To optimize and to consider the student's individual characteristics and preferences during this process, it is advisable to use modern methods of multi-criteria estimation [6]. The functions that implement an "internal cycle", i.e. the student assistance in a specific task step performing are relevant as well [4].

The process of automated student tutoring. The algorithmic tasks automated solving might be structured as shown in Fig. 1.

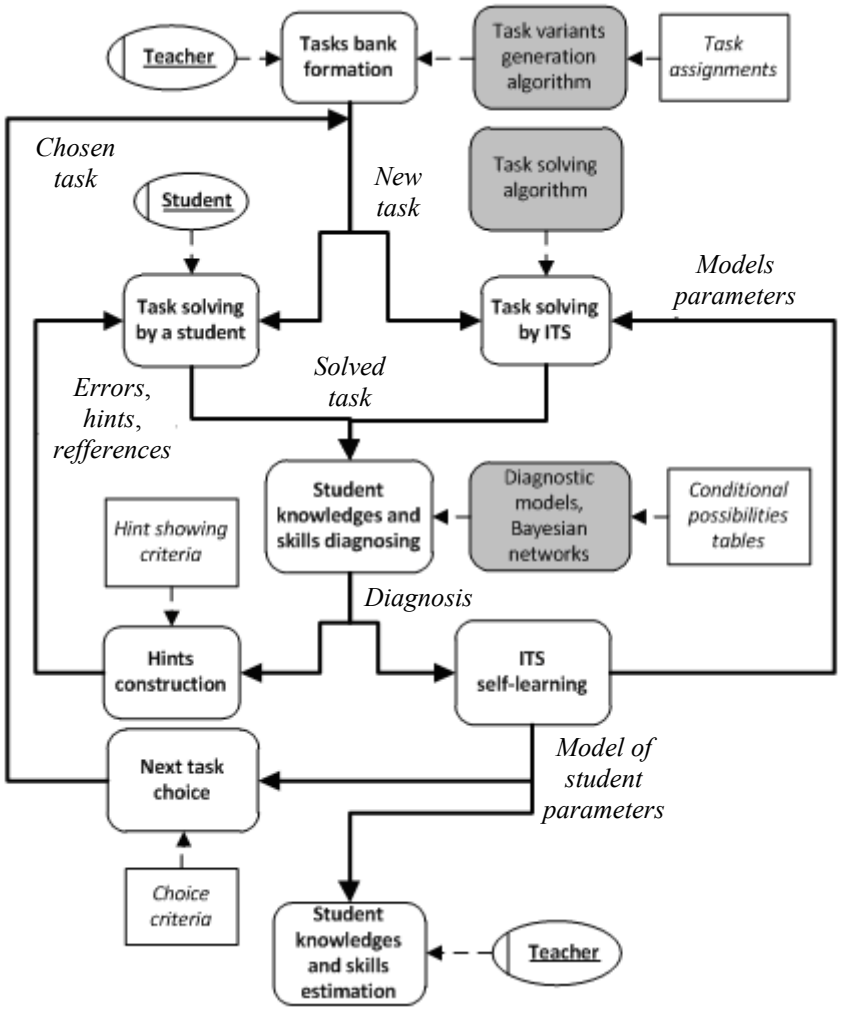

Fig. 1. The process of students automated tutoring for algorithmic tasks solving

At the stage the tasks bank formation the teacher can be helped by the corresponding algorithm of tasks variants set formation on the basis of the tasks assignments templates and input data restrictions customized by him. Automation of

this stage is extremely important, however, it is not fully provided with models and modern tools due to the formalization complexity.

A new task in a case of first step or selected in the case of next step from the formed task bank should be solved in parallel by the student and the ITS with the intermediate and final results fixation in the system on the certain steps. This process is detailed and controlled by a certain algorithm of the internal tutoring cycle, but the diagram shows it as a separate activity of the system, which ends when a student has solved the problem.

Then all the data obtained on the time spent, actions taken, data entered by the student should be used to diagnose his knowledge and skills, i.e. the level of mastery of the competencies inherent in the problem and the mistakes he made in the solution process.

In this paper, the authors propose to use the apparatus of Bayesian networks in combination with diagnostic models used in technical diagnostics. Such a network contains information both on the level of mastering certain competencies by the student, and on the reasons for the discrepancy of his answers with the standard calculated by the ITS.

This will, on the one hand, teach the network in the process of obtaining new data about the student, on the other hand, use the data of network nodes to form tips and explanations for the student [3]. In addition, the resulting state of the Bayesian network can provide quite complete information to assess the level of knowledge of the student after solving a certain sequence of tasks, or to monitor the tutoring process.

The choice of the next step requires software implementation in ITS with minimal teacher intervention in terms of setting the selection criteria in some cases. This process is also proposed to build on the basis of Bayesian network data and based on the goal of building the most effective study trajectory for each student.

As a conceptual basis for the creation of models, methods, algorithmic and software ITS in the study used such principles as redundancy, decomposition of knowledge and skills components (KSC), assessment of knowledge and skills taking into account uncertainties, transition between tasks, automatic task generation, diagnosing knowledge and skills, adaptability, many tips, coverage of the class of tasks, self-learning ITS, learning through the game, recognizing the path of the algorithm by the student and pulling up to the most similar reference algorithm, the orientation of training programs to work via the Internet, rapid automated creation of ITS.

For training of each AT it is offered to use three modes of ITS: demonstration, training and test. In demo mode, ITS demonstrates how to perform tasks by automatically filling in the appropriate input fields on the screen form. In this case, each time the program performs a new task, generating data for the condition, calculating intermediate data and solutions. In the second and third modes the task is formulated by the program, and the student has to execute it. The differences are that in the second mode the system helps the student if necessary, and in the third – no. In all three modes, the student's model changes.

AT model in training mode. The generalized model of the calculated AT is given by a tuple:

$$ModelOfCT_1 = (condct_1, objch_1, objct_{12}, \ldots, objct_{1n}),$$

where $condct_1$ – task condition, $objct_{1i}$ is an object given by an ordered seven components $(uid, name, type, value, format, alg, note)$, where *uid* – unique identifier, *name* – name of the object, *type* – value type, *value* – value, *format* – value format (for example, floating point), *alg* – value calculation algorithm *value*, *note* – designation of the object. To perform the calculated AT, the student forms a tuple of n objects, and $objct1_{n-l}$, $objct1_{n-l} + 1, \ldots, objct_{1n}$ – problem solving, $l \in \{1,2,\ldots,k\}$, $k < n$, other objects — intermediate; $IE\_CT1\_D = \{iect1d_1, iect1d_2, \ldots, iect1d_a\}$ – screen form input elements set; $OBJ = \{objch_{11}, objct_{12}, \ldots, objct_{1n}\}$. The relation $F1$ between sets OBJ and IE_CT1_D – surjective, injective and, therefore, bijective. In demo mode for each item $iect1d_j$ several IE_CT1_DITS enters the appropriate value $objct1_i$. Every $objct1_i$ may be preceded by zero, one or more $objct1_j$, the values of which must be calculated according to the algorithm given above. To formalize such relations, an oriented graph $G = (E, D)$ is constructed. Then each vertex is assigned an ordinal function and subsets B_k, $k = \overline{0, r}$ are defined, which form a of the original graph vertices set E partition and represent its levels. The Demukron method is used to find the levels of the graph G.

Method of objects generation and calculation. If level 0 objects that are independent are to be generated, then objects of all levels other than level 0, i.e. dependent objects, must be calculated. Then the method for generating and calculating objects looks like this:

For each object *Y* level 0
The beginning of the cycle
  Repeat
  Generate values *V* for *Y*;
  Until then, the value of *V* will be valid for *Y*
End of cycle
For each level *R* from 1 to *N*
The beginning of the cycle
  For each object *X* of level *R*
  The beginning of the cycle
    If the value of *V* for *X* is not calculated, then
     Calculate the value *V* for *X*;
    If the value of *V* is not valid for *X*, then
     Beginning
     $N_{attempts}$: = 0;
     Repeat
       ConjunctiveAcceptability: = Truth;
       Procedure 1;
       $N_{attempts}$: = Ntrial + 1;
     Until ConjunctiveAcceptance or Ntrial = Nmax

The end

End of cycle

End of cycle

Procedure 1.

1. Randomly select a level 0 Y object on which X depends.

Form a set of 0th level Y objects, on which X depends both directly and transitively:

$$A = \{Y \mid Y \in B_0, X \subset \Gamma^g Y\}, \quad g \in \{1,2,...,N\},$$

$\Gamma^1 Y = \Gamma Y, \Gamma^2 Y = \Gamma\{\Gamma Y\}, \Gamma^3 Y = \Gamma\{\Gamma\{\Gamma Y\}\}, \ldots$. From the set A randomly select one object: Y:= random (A).

2. Generate a new value $\Pi_{value} Y$ by the algorithm $\Pi_{a\lg} Y$.

3. Calculate the values of all Y-dependent objects, i.e. $\forall z \in \{Z \mid Z = \Gamma^1 Y \cup \Gamma^2 Y \cup ... \cup \Gamma^N Y\}$. Calculate $\Pi_{value} z$ by algorithms $\Pi_{a\lg} z$ and ConjunctiveAcceptability := ConjunctiveAcceptability ^ Acceptability (z).

The method of objects generation and calculation is applied to the ITS demo mode of the task to provide confidence that the value for each object is calculated and it is valid, i.e. the problem has a solution.

The model of the student in the demonstration mode. In the demonstration mode, the possibility of explaining certain steps taken by the program is also realized. To do this, the program contains a button “Explain”, when you click on which reveals the features of the calculation of a value – from the ordered graph of objects G for object Y, select all such objects that make up the set. In addition to the values of each object belonging to the set, ITS displays an algorithm for calculating the value of $Y.\Gamma^{-1} Y\Gamma^{-1} Y$.

In the demonstration mode for each type of task, the student can make several attempts to view the solution. For each attempt, the total time of the student-driven demonstration is saved. The student controls the demonstration of the task using the “Next Step” button and the “Explain” button. Clicking on the “Next Step” button is possible only when the program has finished entering a value in the previous input element*iect*1djand the student considered this step. When you click this button, the program enters the value in the next input element*iect*1dk. Clicking on the “Explain” button is associated with the active input element, which has already entered a value, which thus determines the possibility of pressing it in the interval between two consecutive clicks of the “Next Step” button, and its effect applies only to the current step taken in this interval.

The time sequence of the student’s activity in the demonstration mode is shown graphically in Fig. 2.

In the model of the student all specified moments of time with the corresponding associations are fixed: for any student for any class of settlement AT for any attempt of the student to master this class of tasks within the demonstration mode for any object the tuple remains. $T_i = (t_j, t_{j_{+1}}, t_{j_{+2}}, t_{j_{+3}})$. Based on the accumulated tuples T_i are students who do not work, but play with the program

(if $(t_{j+3} - t_j < \sigma_{i1}) \wedge (t_{j_{+3}} - t_{j+2} < \sigma_{i2})$, where σ_{i1}, σ_{i2} – some thresholds for the so-called "gaming"), cheaters who are not motivated to think independently (when $(t_{j+2} - t_{j+1} < \sigma_{i3}) \wedge \left(\frac{t_{j+3} - t_{j+2}}{t_{j+2} - t_{j+1}} > \sigma_{i4} \right)$, where σ_{13} — threshold of non-independence; σ_{14} – threshold for the ratio of the independence of the student to his independence).

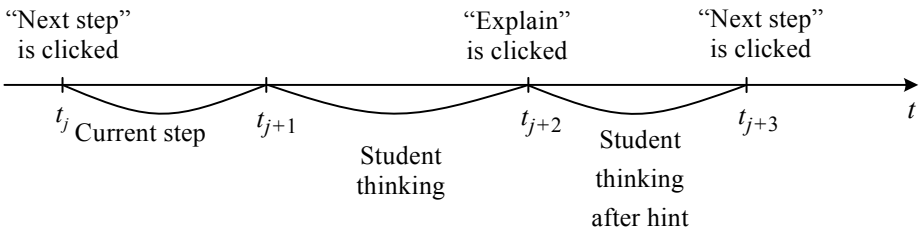

Fig. 2. Time sequence of the student's activity in the demonstration mode

Model of the calculation task in the training mode. The model of the calculation task in the training mode is based on the model of the calculation task in the demonstration mode. In addition to the model of the calculation task in the demonstration mode there is a parallel calculation: both the student and the program calculate the values of objects $objct1_i$. Then for each object these values are compared. In the event of a discrepancy, the reasons for such discrepancy should be clarified to determine the adaptive tutoring sequence for the particular student. Causes of errors are gaps in the knowledge or skills of the student, as well as incorrect knowledge or incorrect skills, consequences – incorrect results of his calculations within a specific task. It is necessary to solve the inverse problem: as a result to find the reason in order to reflect it in the model of the student and to choose the correct restorative training sequence.

Peculiarities of solving these inverse problems in the field of ITS are as follows:

1) a set of reasons is unknown in advance – gaps in knowledge or skills, as well as incorrect knowledge or skills. For each student, they can be their own, special;

2) the set of consequences – infinite when the student performs the task on paper, and when using a computer program is limited only by the number of different values of the data type in a particular input element;

3) the same consequence can correspond to many reasons;

4) several errors can be made in the same calculation (potentially, each initial value and each operation may be invalid);

5) even if the student does not know and cannot answer, he might calculate, guessing or using a hint ("guess");

6) knowing and being able to solve a task, the student might fault due to carelessness ("slip").

To model such processes, it is sometimes advisable to use cognitive maps to identify bottlenecks based on expert data [7]. However, in this paper, models and methods were based on empirical research. The results of an experiment to identify errors of KHAI students in finding the roots of *n*-order algebraic equations by

the Lobachevsky–Greffe–Dandelen (LGD) method to determine the stability of automatic control systems using O. Lyapunov's first method helped to formulate the following error classes:

– errors in calculating the imaginary part of a complex root;
– non-fulfillment of the condition of the end of squaring of roots;
– errors related to misunderstanding of root squaring;
– data recording errors;
– errors in calculating the double product of coefficients;
– rounding errors;
– use of the inverted formula to calculate the roots;
– ignorance of the conditions of existence of complex roots;
– loss of a sign at calculation;
– excessive iteration;
– incorrect raising of the coefficient to the degree;
– ignoring the accuracy of calculations;
– errors in calculating the root2gdegree;
– calculation of the offset share; errors in calculating exponents;
– other errors.

Diagnostic models. Obviously, even for one class of tasks, not to mention different classes, there is no generalized and formalized representation of the diagnostic model (DM) other than the form "IF conditions, then diagnosis is possible". From here for the following researchers it is possible to offer only a way of construction of DM on the basis of examples both for the described class of tasks, and for other tasks.

Examples of DM for the LGD method are:

$$r\_f(\tilde{x},h) \neq r\_f(\hat{x},h),$$

DM to detect the fact of error, where – the reference value calculated by ITS — the value calculated by the student, $\tilde{x}\tilde{x}$

$$r\_f(x,h) = (-1)^t \cdot (z_0 + z_1 \cdot 10^{-1} + z_2 \cdot 10^{-2} + ... + z_h \cdot 10^{-h}) \cdot 10^p$$

a function that is a real number according to the rules of rounding x floating point format up to h decimal places

$$z_b \in \{0,1,...,9\}, \quad t \in \{1,2\}, \quad p \in Z, \ (z_0 > 0) \oplus, \ \oplus(\forall b\, z_b = 0),$$

$\hat{A}_{(k,j)} = r\_f(\hat{A}^2_{(k-1,j)} + 2\sum_{s=1}^{j}(-1)^s \hat{A}_{(k-1,j-s)}\hat{A}_{(k-1,j+s)}, h)$ — coefficients of the matrix of the LGD method, $\hat{A}_{(k-1,c)} = 0$ if $(c < 0) \oplus \ \oplus (c > n)$,

$$(r\_f(\tilde{x},h) - r\_f(\hat{x},h) = -1 \cdot 10^{ex(r\_f(\hat{x},h))-h}) \wedge (\hat{z}_{h+1} \geq 5).$$

DM to determine rounding errors, where $ex(r\_f(x,h)) = p$:

$$\forall j \forall s \in \{g-2, g-1\}((\hat{A}_{(s,j)} > 0) \Rightarrow (r\_f(\hat{A}_{(g-1,j)},h) = r\_f(\hat{A}^2_{(g-2,j)},h))).$$

DM to detect errors of the class “Excessive iteration”.

$$y_l = \pm r\_f\left(\sqrt[2^g]{\frac{m(r\_f(\hat{A}_{(g,l)},h))}{m(r\_f(\hat{A}_{(g,l-1)},h))}\cdot 10^{\frac{ex(r\_f(\hat{A}_{(g,l)},h))}{ex(r\_f(\hat{A}_{(g,l-1)},h))}}},h\right).$$

DM to determine the errors of the class “Incorrect division of degrees”, where $m(r\_f(x,h)) = z_0 + z_1\cdot 10^{-1} + z_2\cdot 10^{-2} + ... + z_h\cdot 10^{-h}$, – the real root of the equation y_l.

Obviously, in this way it is impossible to analyze the actions in the performance of any task for all the students. However, computer training programs must be open and easily modified to introduce new DMs. To implement the possibility of flexible modification of ITS in terms of both adding new DM and changing or supplementing the user interface using the interpreted program code, a compiled-interpreted program model was chosen.

Method of operand skipping diagnostic models automatic construction. As the analysis of student errors in performing tasks by the LGD method and other tasks showed, one of the most common errors is the error of skipping operands. Therefore, part of the DM can be obtained automatically. The method of automatic construction of diagnostic models of operand skipping is given below. It consists of two stages:

1) the expression is translated using the method of “sorting station” E. Dijkstra into the Reverse Polish notation (RPN);

2) by means of the modified calculation of values in the RPN with use of stacks there is a formation of necessary set of diagnostic models.

The second part of the method is given below:

$i := 1$;
Repeat
The beginning of the cycle
Poz_perekr_totoch_operati: = pos_operati;
For j from 1 to n
The beginning of the cycle
Potoch _lex: = lex$_j$;
If Potoch_lex $\notin$ Operators, then
Beginning
Place in Stek_perekr_livor Potoch_lex;
Place in Stack_transfer_dream Potoch_lex;
The end
Else
Beginning
Operand_right_break_left := extract Stack_break_left;
Operand_left_break_left := extract Stack_break_left;
Operand_right_break_right := extract Stack_break_right;
Operand_left_break_right := extract Stack_break_rigft;
If Pos_flow_lex = Pos_over_flow_operat, Then

```
Select Potoch_lex From
 '+', '-':
   Beginning
    Res_break_left := '0' + Potoch_lex + Operand_right_break_left;
    Res_break_right := Operand_left_break_right + Potoch_lex + '0';
   The end
       '*', '/', '^':
 Beginning
         Res_break_left := '1' + Potoch_lex + Operand_right_ break_left;
   Res_break_right := Operand_left_break_right + Potoch_lex + '1';
  The end
End of selection
   Else
   Beginning
Res_break_left := Operand_left_break_left + Potoch_lex + Operand_
     right_break_left;
Res _break_right: = Operand_left_break_right + Potoch_lex + Operand_
     right_break_right;
    The end
 Place in Stack_of_break_left '(' + Res_break_left + ')';
 Place in 'Stack_of_break_right ' ('+ Res_break_right + ') ';
The end
End of cycle
DM_left [i]: = extract Stack_break_left;
DM_right [i]: = extract Stack_break_right;
i: = i + 1;
End of cycle
Until i = m + 1,
```

Where $Operators = \{'+','-','*','/','^'\}$, s – a string to which the formula in the RPN corresponds, $lex = (lex_1, lex_2, \ldots, lex_n)$ – tokens selected from line s, $pos\_lex = (pos\_lex_1, pos\_lex_2, \ldots, pos\_lex_n)$ – positions of the tokens from the tuple lex in the source $line$ s, $pos\_operat = (pos\_operat_1, pos\_operat_2, \ldots, pos\_operat_m)$ – the positions of the operators in the line s.

In the case of several different DMs operation, the ITS asks the student to clarify the diagnosis, offering him several options for calculating the wrong value for different diagnostic models, as well as the ability to enter their own, which does not match the proposed, calculation option that serves as a signal to find a new DM.

The student's model for the calculation task in the training mode. The central place in the student's model for the calculation task in the training mode is occupied by KSC. It is from their values that the adaptive tutoring sequence for a particular student depends. However, there is another inverse problem, how to determine the results of the student's work with the program KSC or what should

be the relationship between the steps of the method – objects $objct1_i$, the value of which is formed by the student, and KSC.

The probabilistic approach and the use of Bayesian networks (BN) are chosen as the approach to the modeling of the student [8, 9]. In the famous work of the American researcher K. VanLehn, it is proposed to use not one BN, but set of BN to model the student. But since the ITS has additional information about the KSC of students in the form of a DM set, such a set of BN could be constructed as represented in Fig. 3.

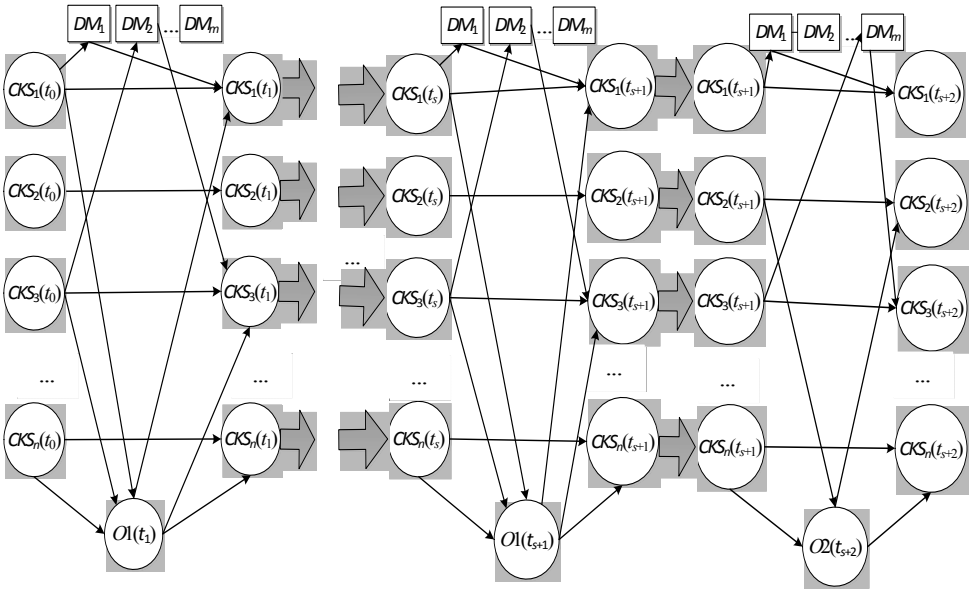

Fig. 3. BN structure for the student's KSC modeling with DM: $CKS_i(t_k)$ — *i*-th KSC at time t_k; $O_j(t_s)$ — an object entered by the student into the system at time t_s; DM_j — *j*-th diagnostic model

During the simulation, such probability values were formed in the tables of conditional probabilities, which in case of incorrect step of the student and operation of some DM associated with a particular KSC, reduce the probability of owning this KSC compared to the case of incorrect step and failure of this DM. When adding a task and its KSC to the system, you should check whether there are such components for previously added tasks in order to use as a priori probabilities of possession of a component of a new task a posteriori probabilities for tasks that the student has already performed. Since KSCs differ from each other in names and probability values, it is necessary to search for similar KSC by a given name to avoid duplicate and quasi-duplicate components in the system. Due to possible operator errors, as well as possible permutation of words or the use of abbreviations, several metrics should be applied simultaneously for similar strings.

Thus, it is necessary to form a set

$$SSim = SSim_1 \cup SSim_2 \cup SSim_3,$$

where $SSim_1 = \{sn_i \mid d_{lev}(s_0, sn_i) \le \theta\}$, $SSim_2 = \left\{ sn_j \mid \frac{q\_joint(sn_j, s_0)}{q\_avg(sn_j, s_0)} \ge \mu \right\}$,

$SSim_3 = \{sn_k \mid d_{abb}(s_0, sn_k) \le \sigma\}$, $SN = (sn_1, sn_2, \ldots, sn_m)$ — a set of KSC names

strings; s_0 – the name of KSC entered; $d_{lev}(s_0, sn_i)$ — Levenstein distance between lines s_0 and sn_i; $q\_joint(sn_j, s_0)$ — number of common q-grams in lines sn_j and s_0; $q\_avg(sn_j, s_0)$ — average number of q-grams in lines sn_j and s_0; $d_{abb}(s_0, sn_k)$ — the distance of editing abbreviations for strings s_0 and sn_j; θ, μ, σ – some thresholds.

NearestHash method. The NearestHash method is used to solve the first subtask. In common, the problem statement is following: a given editing distance δ between objects of some class Cl, which satisfies conditions:

$$\begin{cases} \delta(X,Y) \geq 0 \text{ — not negative,} \\ \delta(X,X) = 0 \text{ — is null,} \\ \delta(X,Y) = \delta(Y,X) \text{ — simmetry,} \\ \delta(X,Z) \leq \delta(X,Y) + \delta(Y,Z) \text{ — triangle sides relation.} \end{cases} \quad (1)$$

Let object rt of class Cl and set of objects $ET = \{et_1, et_2, ..., et_n\}$ of the same class are given. It is required to find $ET_s = \{et_{s1}, et_{s2}, ..., et_{sl}\}$, such as $\forall et_{si} \in ET_s \subseteq ET : \delta(et_{si}, rt) \leq \lambda, \ \lambda \in N, \ l \leq n$.

The proposed NearestHash method consists of two steps.

Step 1. From the set of ET randomly selected k elements $o_1, o_2, ..., o_k$, $(k \leq n)$. These elements are further associated with the k axes of the k-dimensional Euclidean space E^k. After that, each element et_i of the set ET is assigned a point E^k, the coordinates of which are equal to the distances to the axes, i.e. $P(et_i)_j = \delta(et_i, o_j), \ i = \overline{1,n}, \ j = \overline{1,k}$.

Step 2. The object rt is also placed in accordance with E^k point with coordinates $P(rt)_j = \delta(rt, o_j), \ j = \overline{1,k}$. In this step, the distances are calculated only between rt and objects whose corresponding points are located close in E^k to the point $P(rt)$.

The necessary conditions for the similarity of objects X, Y and Z of class Cl are proved.

Proposition 1. For given objects et_i and et_j of class Cl, the distance between which δ satisfies conditions (1) and does not exceed a certain threshold λ, according to the method NearestHash:

a) points $P(et_i)$ and $P(et_j)$ of the space E^k, corresponding to the source objects, removed in E^k from each other at a distance of not more than $\lambda\sqrt{k}$, i.e. $\forall i \forall j \neq i \ \delta(et_i, et_j) \leq \lambda : \rho(P(et_i), P(et_j)) \leq \lambda\sqrt{k}$;

b) the point $P(et_j)$ is placed in E^k within the hypercube with the center at the point $P(et_i)$ and the side length 2λ;

c) the absolute value of the difference between the points $P(et_i)$ and $P(et_j)$ to the origin in E^k does not exceed $\lambda\sqrt{k}$, i.e. $|\rho(P(et_i),0)-\rho(P(et_j),0)|\leq \leq \lambda\sqrt{k}$;

d) if the absolute value of the difference between the sizes of objects et_i and et_j is greater than λ, then the distance between these objects is greater than λ, i.e. $(|\overline{et_i}-\overline{et_j}|>\lambda)\Rightarrow(\delta(et_i,et_j)>\lambda)$.

CONCLUSIONS

The article presents one of the possible ways to solve the problem of adaptive tutoring with modern knowledge and skills through the creation of intelligent computer training programs. The concept, methods and models of ITS are presented. The second part will present practical results for the development and implementation of specific ITSs.

REFERENCES

1. T. Klingberg, *The Overflowing Brain: Information Overload and the Limits of Working Memory*. Oxford University Press, 2009, 216 p.
2. A. Kulik, "Rational intellectualization of aircraft control: Resources-saving safety improvement", *Studies in Systems, Decision and Control*, **105**, pp. 173–192, 2017.
3. A. Chukhray, *Methodology for learning algorithms: monograph*. National Aerospace University "KhAI", 2017, 336 p.
4. J.P. Martínez Bastida, O. Havrylenko, and A. Chukhray, "Developing a self-regulation environment in an open learning model with higher fidelity assessment", *Communications in Computer and Information Science*, **826**, pp. 112–131, 2018.
5. M.Z. Zgurovsky and N.D. Pankratova, *System analysis: problems, methodology, applications*. Kiev: Nauk. opinion, 2011.
6. N.D. Pankratova and N.I. Nedashkovskaya, "Hybrid method of multicriteria evaluation of decision alternatives", *Cybernetics and Systems Analysis*, vol. 50 (5), pp. 701–711, 2014.
7. G. Gorelova and N. Pankratova, "Strategy of complex systems development based on the synthesis of foresight and cognitive modeling methodologies", *IEEE First International Conference on System Analysis & Intelligent Computing* (*SAIC 2018*), pp. 1–6.
8. M.Z. Zgurovsky, P.I. Bidyuk, and A.N. Terentyev, "Methods for constructing Bayesian networks based on estimating functions", *Cybernetics and Systems Analysis*, no. 2, pp. 81–88, 2008.
9. M.Z. Zgurovsky, P.I. Bidyuk, O.M. Terentyev, T.I. Prosyankina-Zharova, *Bayesian Networks in Decision Support Systems*. Edelweiss Publishing Company, 2015, 300 p.
10. N.D. Pankratova, "System analysis in the dynamics of diagnosing complex technical systems", *System research and information technology*, no. 4, pp. 33–49, 2008.

Received 01.08.2021

INFORMATION ON THE ARTICLE

Anatoliy S. Kulik, ORCID: 0000-0001-8253-8784, National Aerospace University "Kharkiv aviation institute", Ukraine, e-mail: anatolykulik@gmail.com

Andrey G. Chukhray, ORCID: 0000-0002-8075-3664, National Aerospace University "Kharkiv aviation institute", Ukraine, e-mail: achukhray@gmail.com

Olena V. Havrylenko, ORCID: 0000-0001-5227-9742, National Aerospace University "Kharkiv aviation institute", Ukraine, e-mail: lm77191220@gmail.com

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ СТВОРЕННЯ ІНТЕЛЕКТУАЛЬНИХ КОМП'ЮТЕРНИХ ПРОГРАМ ДЛЯ НАВЧАННЯ АЛГОРИТМІЧНИМ ЗАВДАННЯМ. Частина 1: Математичні основи / А.С. Кулік, А.Г. Чухрай, О.В. Гавриленко

Анотація. Існуюча система освіти (зокрема вища освіта) через орієнтацію на базові знання досить інертна і не спроможна забезпечувати потреби сучасного ринку праці, що стрімко розвивається. Деякі професії трансформуються або зникають, з'являються нові. Роботодавцям сьогодні потрібні фахівці з певними навиками і вміннями, які здатні розвивати їх та адаптувати до конкретних проектів. Саме тому популярними є короткострокові курси, особливо онлайн та з наставником — фахівцем у певній галузі. Утім випускники таких курсів здебільшого не в змозі самостійно вирішувати складні завдання та приймати грамотні рішення. Постає потреба у створенні навчальних програм для перевірки розробки та впровадження засобів продуктивного передавання знань і навичок у конкретній галузі. Показано можливий підхід до забезпечення певної інтерактивності засобів комп'ютерного навчання як додаток до ігрового принципу, візуалізації інформації та інших прийомів, застосовних в інформаційних системах. Це дозволить створити платформу, яка зможе акумулювати нові технології, інтегруючи їх у цифрове навчальне середовище, яке може бути адаптованим для кожного студента.

Ключові слова: інтелектуальна система навчання, алгоритмічні завдання, діагностичні моделі, байєсівські мережі, модель студента.

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ СОЗДАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ КОМПЬЮТЕРНЫХ ПРОГРАММ ДЛЯ ОБУЧЕНИЯ АЛГОРИТМИЧЕСКИМ ЗАДАНИЯМ. Часть 1. Математические основы / А.С. Кулик, А.Г. Чухрай, Е.В. Гавриленко

Аннотация. Существующая система образования (в частности, высшее образование) из-за ориентации на базовые знания довольно инертная и не может удовлетворять потребности современного рынка труда, что стремительно развивается. Некоторые профессии трансформируются или исчезают, появляются новые. Работодателям сегодня нужны специалисты с определенными навыками и умениями, которые способны их развивать и адаптировать к конкретным проектам. Именно поэтому популярны краткосрочные курсы, особенно онлайн и с наставником — специалистом в определенной области. Но выпускники таких курсов не могут самостоятельно решать сложные задания и принимать грамотные решения. Требуется создание обучающих программ для тестирования разработки и внедрения инструментов продуктивной передачи знаний и навыков в определенной области. Показан возможный подход к обеспечению некоторой интерактивности средств компьютерного обучения в дополнение к игровому принципу, визуализации информации и другим методам, применимым в информационных системах. Это позволит создать платформу, способную аккумулировать новые технологии, интегрируя их в цифровую среду обучения, которая может быть адаптирована для каждого ученика.

Ключевые слова: интеллектуальная система обучения, алгоритмические задания, диагностические модели, байесовские сети, модель студента.