Implementation of Computer Tutoring Program Adaptation Methods

Stanislav I. Pedan
National Aerospace University
Kharkiv aviation institute, UKRAINE

Andriy G. Chukhray
National Aerospace University
Kharkiv aviation institute, UKRAINE

Steven D. Hall
Texas A&M University-Corpus Christi

Anatoliy S. Kulik
National Aerospace University
Kharkiv aviation institute, UKRAINE

Implementation of computer tutoring programs and adaptation to the level of the user competence are described. Three methods of next task choice at outer loop level are the results. The algorithm of automatic formation of the decision-making model for inner loop task is presented. Method execution is shown with an example of two linear equations solving the problem. The process of developed adaptation methods implementation using a graphic method of tutoring programs creation in the universal environment is described.

INTRODUCTION

The growth of intelligent computer tutoring programs (ICTP) in the educational process is connected with the possibility of overcoming traditional teaching disadvantages: orientation to the "average" trainee, lack of teaching material adapted to the trainee's competence level, trainee's individual learning style and other factors. ICTP are capable of providing educational programs for the individual student based on their competence level, and allow the individual to proceed at the optimal speed for mastering the knowledge and skills. However adapting a program each time a new ICTP appears seems a rather labor-consuming problem. Moreover, frequently the teachers who could create ICTP do not possess the programming knowledge necessary to develop the ICTP.

In the present research the following problems are solved:

1) Development of ICTP flowcharts which do not require programming skills;
2) Development of ICTP outer loop adaptation methods;
3) Development of adaptation method in ICTP tasks inner loop.

METHOD OF ICTP GRAPHIC CREATION

Presently, authoring tools (AT) for computer tutoring programs (CTP) development are widely available. Examples of AT are DIAG, RIDES, SIMQUEST, XAIDA, Demonstr8, D3, TRAINER, ASPIRE, GTE, REDEEM, Eon, Interbook, MetaLinks, CALAT, CTAT and others (Woolf, 2009). One of the most widely known AT in the virtual learning environment is Moodle (Moodle Service Network) and in the mathematical environment there is ActiveMath (LeActivieMath). The creation of an ICTP in Moodle is difficult and time-consuming. The author must develop his own methods and tools to create feedback loops for the user. ActiveMath is restricted to mathematics only. There is a need for a universal AT with a wide spectrum of various educational subjects that will allow quick formation of ICTP with intelligent and adaptive functions. This software should not require programming skills. The ICTP development process should require minimal time and provide simplicity of ICTP modification. The ideal ICTP should provide an individual approach using basic pedagogical principles, and adaptation of the tutoring process required of the subject domain and the skill level of the user. This program will provide a universal environment for the creation of ICTP. Proceeding from the premise, it is possible to formulate the basic requirements of a universal environment ICTP creation program:

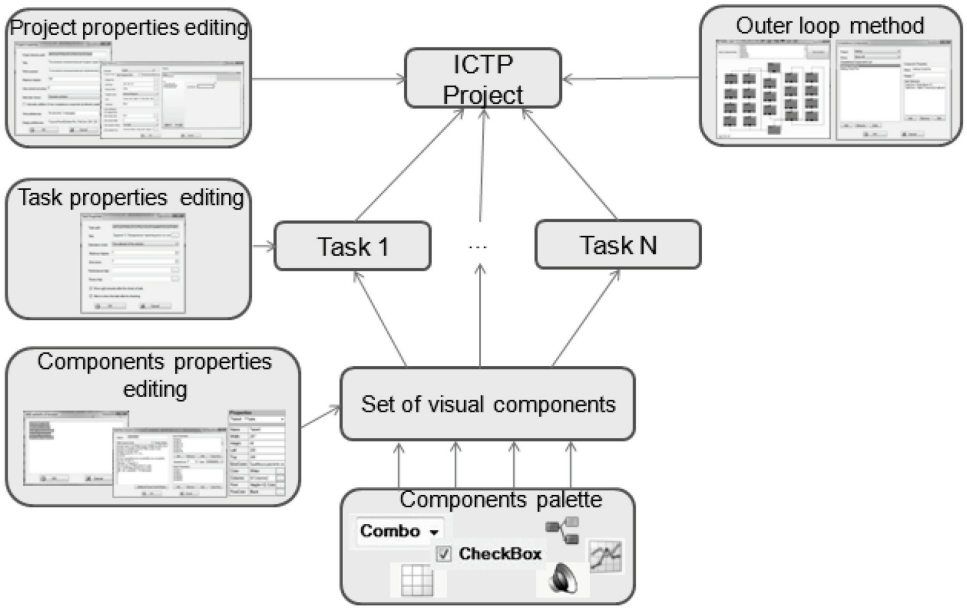
1) Support of ICTP graphic assemblage (the majority of people learn visually);
2) ICTP tasks construction should be structured and easily modified;
3) ICTP execution should be directed on an individual approach.

The universal environment should provide means for intuitive support of created programs with intelligent possibilities of the trainee's competence control under specific subject domain.

The scheme of ICTP graphic creation process is presented in figure 1. Creation of the tutoring program consists in several stages:

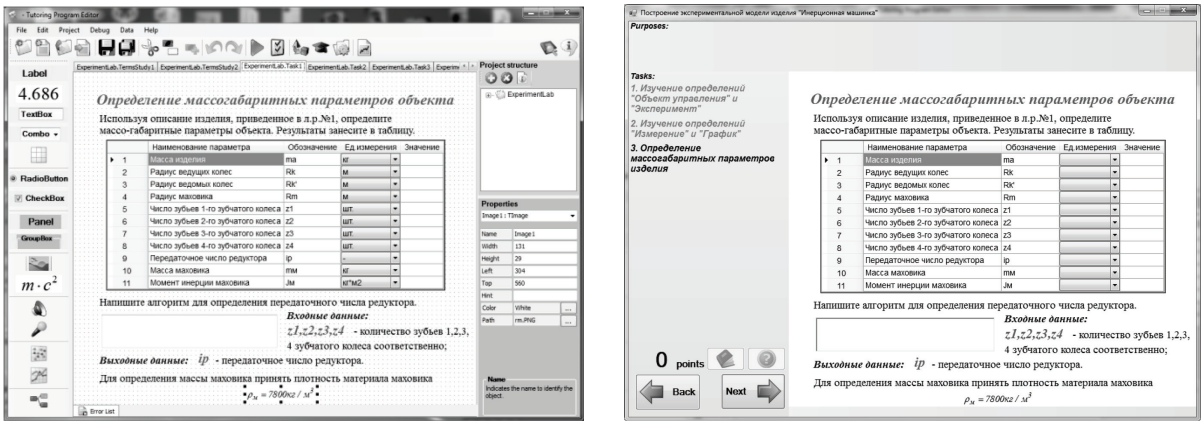
1) Creating of ICTP tasks set. Each of tasks consists from several visual components which define its appearance. Visual components are divided on 2 categories: display elements (for example, labels, images, function graph construction component) and input elements (text input fields, tables, combo boxes, elements of mathematical formulas assemblage). After the placement of components on the task form we need to define the properties of them such as width, height, text, background color, font, etc. For input elements the compulsory requirements are the setting of entered data etalon values or algorithms of their calculation in the form of mathematical formulas or program code.
2) Defining of tasks properties. So, after creating of set of ICTP tasks we need to set some of their properties such as title, maximum grade, price of error, estimation mode, etc.
3) Defining of project properties. All created tasks are united in one project which represents the ICTP. Project properties are title, ICTP execution time limitations, design preferences, etc. But the most important property is a next task choice mode or outer loop method which is described in the following section.

FIGURE 1
SCHEME OF ICTP GRAPHICAL CREATION



All described steps are performed in the universal environment ICTP creation program. After execution of these steps of ICTP creation we receive a tutoring program project which can be executed by the corresponded program of ICTP translation.

FIGURE 2
VIEW OF ICTP IN UNIVERSAL ENVIRONMENT
(CREATION AND TRANSLATION MODES)



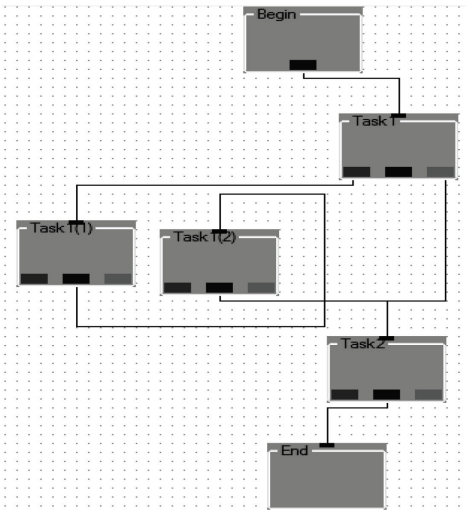
Thus, tutoring programs creation process consists in formation of tasks set as collections of visual components of various types, settings of each tasks properties and their association in the project. Such structure of ICTP project provides the implementation of the second requirement that of allowing fast addition of new tasks. Let's consider in the details, the issue of supporting of ICTP adaptation, under the varying levels of competence of each trainee which is based on the implementation of two loops: outer ICTP loop and inner loop of its tasks (VanLehn, 2006).

OUTER LOOP FORMATION METHOD

Outer loop is responsible for decision-making on what is the next task for execution by the user. Decision-making in the outer loop should be aimed at mastering by the trainee of the set of competence components (CC). The universal environment offers some variants of mechanisms of inner loop decision-making:

1) Pedagogical scenario of the tasks to be performed. The teacher designs the context-dependent scenario of tasks execution in a special graphic editor. The pedagogical scenario is a tool of studied material, which mastering corresponds to the trainee competence. In another cases scenario scheme serves for setting of the order of the stages. The scheme presented on fig. 3, displays ICTP pedagogical scenario, consisting of two tasks Task1 and Task2.

FIGURE 3
EXAMPLE OF THE SCENARIO SCHEME



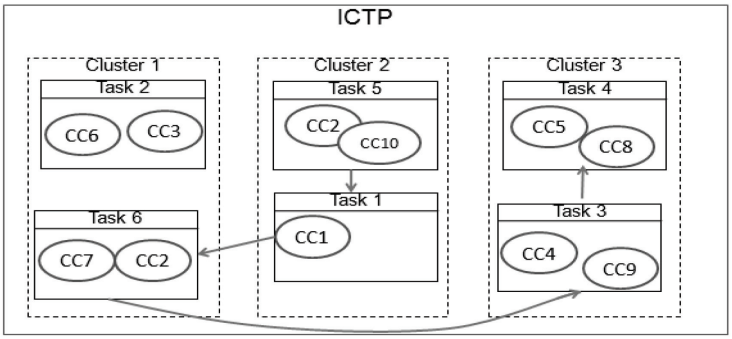
Tasks for the scheme are represented by blocks with one input and three outputs (defining the success of the task performance) which are connected by communication lines, defining the conditional transitions between the stages of the pedagogical scenario. On the scheme it is defined that in case of an error on solving Task1 the trainee should execute the two additional tasks Task1 (1) and Task1 (2) and only after successful execution go to Task2.

According to scheme the call of task performance is implemented. Thus, the teacher independently, proceeding from their own experience, offers the scheme of possible tasks of the trainees. The advantage of such method is that the task choice is performed from teacher own experience. The drawback to the system is that it does not always possess the current level of trainee competence.

2) Decision-making about the next problem for the user. The conscious trainee knows what level of CCs he needs first of all. Therefore, the universal environment should offer to the trainee the whole list of possible tasks and the independent formation by the trainee of the tutoring sequence. But frequently the level of chosen tasks complexity does not correspond to current level of the trainee competence and actual tutoring goals.
3) Decision-making about the next task by means of ICTP adaptation. The universal environment allows to define ICTP CCs in a special CCs editor, the specify elements of input which influence the degree of confidence of CC possession, and also define for each of the tasks a set of CCs required for its successful completion. In the beginning of the ICTP execution tasks are grouped in similar clusters. Inside of a cluster tasks are ordered on their CCs total complexity. In the case of successful task performance the next task from the same

cluster must be greater on total CCs complexity level. If the task has been executed unsuccessful the next task should be smaller on total CCs complexity on the minimum value. In case of mastering by the trainee of required CCs volume in current cluster there is a choice of one of the following cluster tasks.

FIGURE 4
EXAMPLE OF POSSIBLE SCHEME OF ICTP OUTER LOOP



As a result of the described next task decision-making there is a formation of an individual outer loop of tasks adapted under the level of trainee competence. For example, for the task of some physical object mathematical model designing, decision-making mechanism of the outer loop can form the following sequence of tasks: studying of forces, operating on object, formation of mathematical model construction algorithm, execution on calculation according to constructed model, testing of the model. This is one of possible task sequences. In another case this sequence can contain a larger quantity of tasks or a task with another level of complexity. It depends on the trainee's competence level and his/her progress at ICTP tasks executions.

The maximum effect of the tasks outer loop execution can be reached by a combination of three described mechanisms of decision-making. Formation of studying the individual program and the education material adaptation for each of trainees can be reached not only on the outer loop level but also on the ICTP tasks inner loops. The method of inner loop formation is described in the following section.

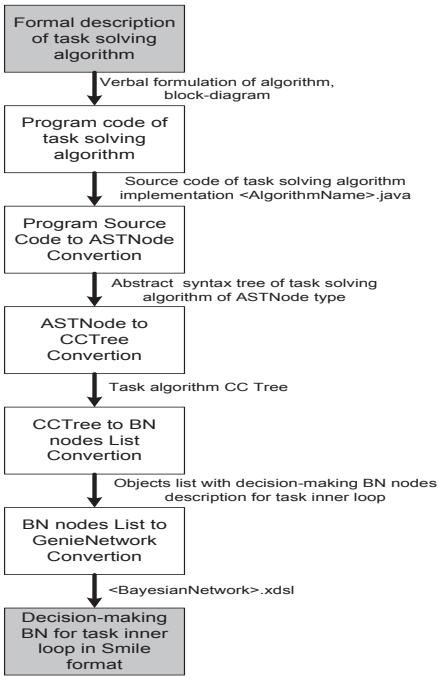
INNER LOOP FORMATION METHOD

In case of simple input elements, such as a table, text input field, component of graphs designing, combo box etc., it is simple enough to define on which CCs they influence as a result of user input. But the universal environment allows the creating of tasks which require from the user a more complex answer input - algorithm of some problem solving in form of its program code. Task inner loop is responsible for the intelligent analysis of its solving algorithm program code for the purpose of revealing of CCs containing in it and the communications between them and formation of the adapted sub-tasks sequence compensating of trainee knowledge or lack thereof. As a model of inner loop decision-making (Murray, Lehn, & Mostow, 2006) Bayesian networks approach has been chosen. This model allows to estimate the probabilities of each tasks steps execution success and to choose the most useful of them from the pedagogical point of view. Principles of inner loop execution can be applied to solving of various technical problems which algorithm can be formalized and presented in the form of a program code. To each of the algorithm CC the probability node corresponds to defined degree of system confidence in trainee possession by it, and also utility node defines importance of this CC possession check in the inner loop. Decision-making about the inner loop next sub-task representing studying of some CC is carried out proceeding from definition of CC with the maximum value in its probability node and utility node values product. The algorithm of automatic generation of task inner loop model is developed. It performs the receiving of the task solving algorithm abstract syntactic tree (AST), AST

transformation to CC tree, and synthesis from CC tree of the decision-making BN(Kulik,2011). Let's consider in details this algorithm.

The first stage of the process of the inner loop transition model automatic formation is based on a program code of the problem solving algorithm is its parsing. It is aimed at identification of CC, necessary for task performance. After that on the basis of the task CC structure there is construction of the decision-making BN, forming rational sequence of task steps execution due to decision-making about the most useful alternative after receiving of the next evidences about tutoring process progress. In figure 5 the scheme of decision-making BN automatic formation stages for ICTP inner loop is presented.

FIGURE 5
THE SCHEME OF DECISION-MAKING BN AUTOMATIC FORMATION STAGES FOR ICTP INNER LOOP



The initial data for work of the transition model automatic formation system is the formal description of the problem solving algorithm in the form of the verbal information on sequence of its stages, block diagrams, formulas of mathematical calculations.

The result of the system work is the decision-making BN for tutoring program inner loop. BN is represented in SMILE format (GeNIe) (platform-independent library of classes for implementation of graphic probabilistic models, and also decision-making models), convenient for the decision of its analysis of problems and solving of probabilistic inference problems.

For receiving of final decision-making BN it is necessary to execute a number of actions:

1) The initial problem algorithm is represented in the form of program code written in high level language, in our case in Java. Java provides wide tools for program code syntax analysis. The given function is carried out by the developer of the tutoring program or by available automated tools of problem solving algorithm graphic representation transformation in a corresponding program code. Result of this stage of performance is the file with a Java-code of the initial problem solving.
2) The construction of an abstract syntax tree (AST) of problem solving program code is made. The object of ASTParser class is created for this purpose. ASTParser is an element of the Eclipse JDT API-functions (The Eclipse Foundation) for program code AST construction,

manipulations by it, detection of its errors, compilation and execution. AST is the final, oriented tree in which internal tops are associated with operators of a programming language, and leaves with corresponding operands. Thus, leaves are empty operators and represent only variables and constants. After reference of program source code the object of ASTParser class will transform it to hierarchical structure of ASTNode type. Every Java language construction can be presented by node of corresponding type with various nesting degree concerning root node of a tree. For example, the function containing the declaration of two variables, can be presented at top level by MethodDeclaration type node (ASTNode successor) and two nodes of VariableDeclarationStatement type nested in it. As a result the source program java-code will be transformed to corresponding hierarchical structure of AST nodes.

3) Algorithm AST will be transformed to a problem CC tree (CCT). The given tree represents the hierarchical structure consisting of terminal type nodes (operators and names of variables of a source code) and non-terminal, containing the information on the list of the nodes nested in them. CCT can be received from AST by application of idea of Composite pattern (Gamma, Helm, Johnson, & Vlissides, 1995). Thus for CCT construction AST nodes connected only with CC interested to us are analyzed, other nodes are ignored.
4) CCT will be transformed to objects array with the information on decision-making BN nodes for an inner loop of a problem. Thus, if CCT contained relations between the nodes, hierarchies going from the top nodes to the nodes nested in it the direction of BN relations changes from more nested to the top nodes of hierarchy. It is connected with features of decision-making BN probabilistic inference as large CC acquirement depends on results of performance of its components.
5) On an available array of the information on BN nodes the BN object of Network type which is a copy of API SMILE class is constructed. The given object can be saved in a file of Genie format (graphic environment for synthesis and the analysis of probabilistic models), and also is used for the internal program analysis and solving of BN probabilistic inference problems.

Let's consider the process of decision-making BN automatic construction for a problem of tutoring to the solving of the linear equations systems.

The universal environment allows us to organize task inner loop execution by the addition in it of inner loop visual component. For such component ICTP developer should define a set of solving etalon algorithms. We will consider stages of inner loop mechanism creation and its execution on an example of problem of two linear equations system solving.

Let the system of two linear equations in a general view looks as follows:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = b_1, \\ a_{21}x_1 + a_{22}x_2 = b_2 \end{cases} \quad (1)$$

Where $a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2$ - some known integers, $a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2 \in Z$. It is necessary to define values of unknowns $x_1, x_2 \in R$.

There is a set of ways of the given linear equations system solving. Let's use Kramer's method and find a system determinant, i.e. the determinant of the second order made of factors at unknowns:

$$V = \begin{vmatrix} a_{11} a_{12} \\ a_{21} a_{22} \end{vmatrix} \quad (2)$$

Let's make two more determinants from (1), having replaced in the first of them the first, and in the second - the second columns with the column made of free members b_1, b_2 :

$$V_1 = \begin{vmatrix} b_1 a_{12} \\ b_2 a_{22} \end{vmatrix} \qquad V_2 = \begin{vmatrix} a_{11} b_1 \\ a_{21} b_2 \end{vmatrix} \tag{3}$$

Then, according to Kramer's rule, if a system determinant $\Delta \neq 0$ the considered system has one and only one decision, and:

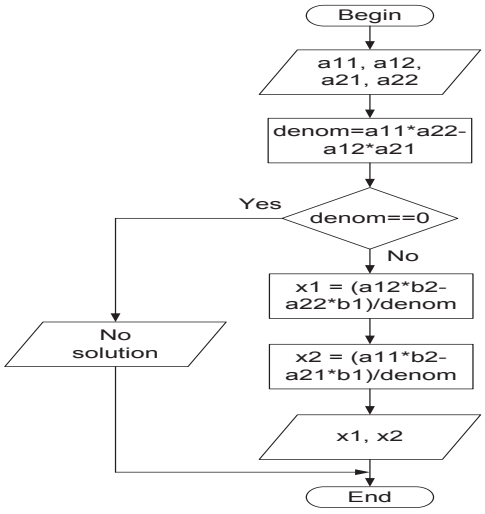
$$x_1 = \frac{V_1}{V} \qquad x_2 = \frac{V_2}{V} \tag{4}$$

Thus, for finding of linear equations system unknowns it is necessary:

1) To find value of system determinant according to the formula (2);
2) To check up a condition of system decision existence ($\Delta \neq 0$);
3) To construct 2 private system determinants according to the formula (3);
4) To find values of the first and second unknowns according to the formula (4).

Having the verbal description of solving algorithm of system from two linear equations, we can make the block diagram of system solving program algorithm:

FIGURE 6
ALGORITHM OF TWO LINEAR EQUATIONS SYSTEM SOLVING

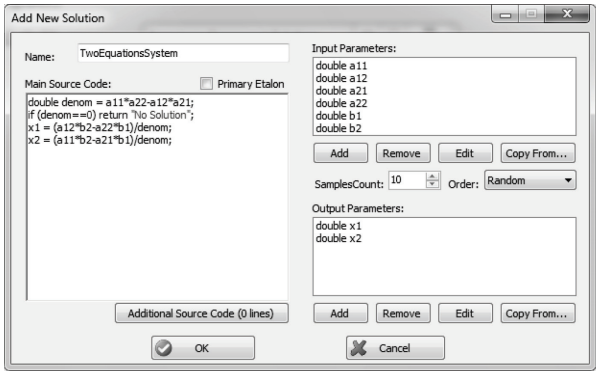


According to the algorithm made above it is possible to realize its implementation in Java language. The program code by system unknowns finding (1) will have the following view: double denom = a11*a22-a12*a21; if (denom = = 0) return "No Solution"; x1 = (a12*b2-a22*b1)/denom; x2 = (a11*b2-a21*b1)/denom;

Algorithm input data is variables a11, a12, a21, a22, b1, b2, and output data is x1 and x2. We can check the correctness of trainee algorithm work by comparison of its execution results with results of some etalon algorithm execution on a set of input data test values. In other words, we should generate set of input parameters a11, a12, a21, a22, b1, b2 values, submit them to algorithms input, receive values of the output variables x1 and x2 for the trainee and etalon algorithms and compare them. In case the deviation of trainee and etalon output data exceeds admissible norm execution of inner loop decision-making mechanism begins. The purpose of its execution is compensation of trainee knowledge and skills

lack in algorithm construction. We will consider in more details process of creation of etalon algorithm creation for two linear equations system problem solving.

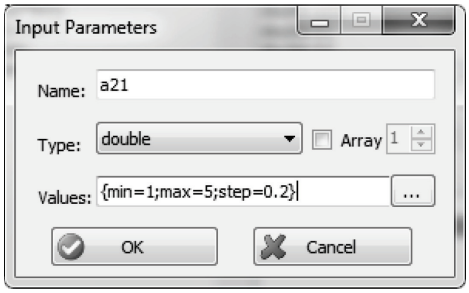
FIGURE 7
CREATION OF TASK SOLVING ETALON ALGORITHM



In figure 7 the screenshot of etalon algorithm creation editor for two linear equations solving problem is presented. Algorithm parameters are:

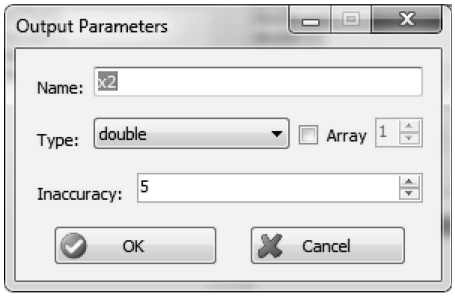
1) Program code of the problem solving;
2) The list of algorithm input parameters. For input parameters it is necessary to specify their name, type and set of test values for check of algorithm work.

FIGURE 8
EXAMPLE OF NEW INPUT PARAMETER a21 ADDITION



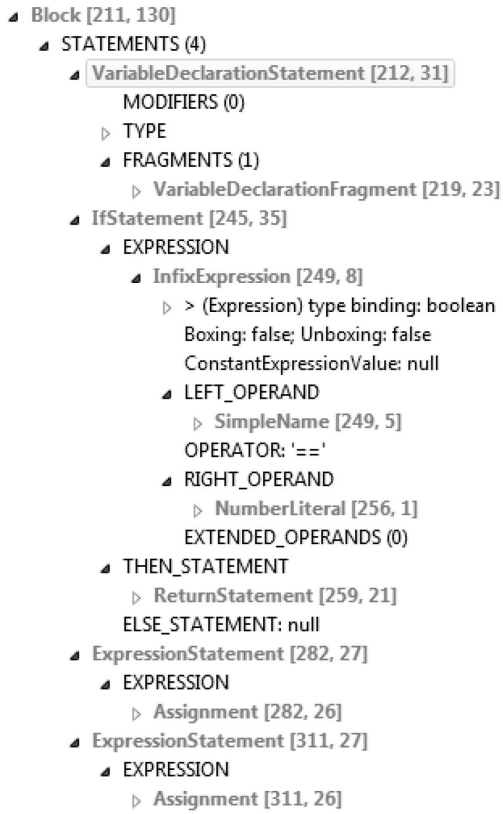
3) Quantity of samples of input parameters test values and method of their formation (consecutive choice or random). For formation of a correct conclusion about correctness of trainee algorithm work it is necessary to check up it's functioning on set of various test values of input parameters.
4) The list of output parameters. For output parameters it is necessary to specify their type and value of its maximum deviation in percentage from the etalon value.

FIGURE 9
EXAMPLE OF NEW OUTPUT PARAMETER x2 ADDITION



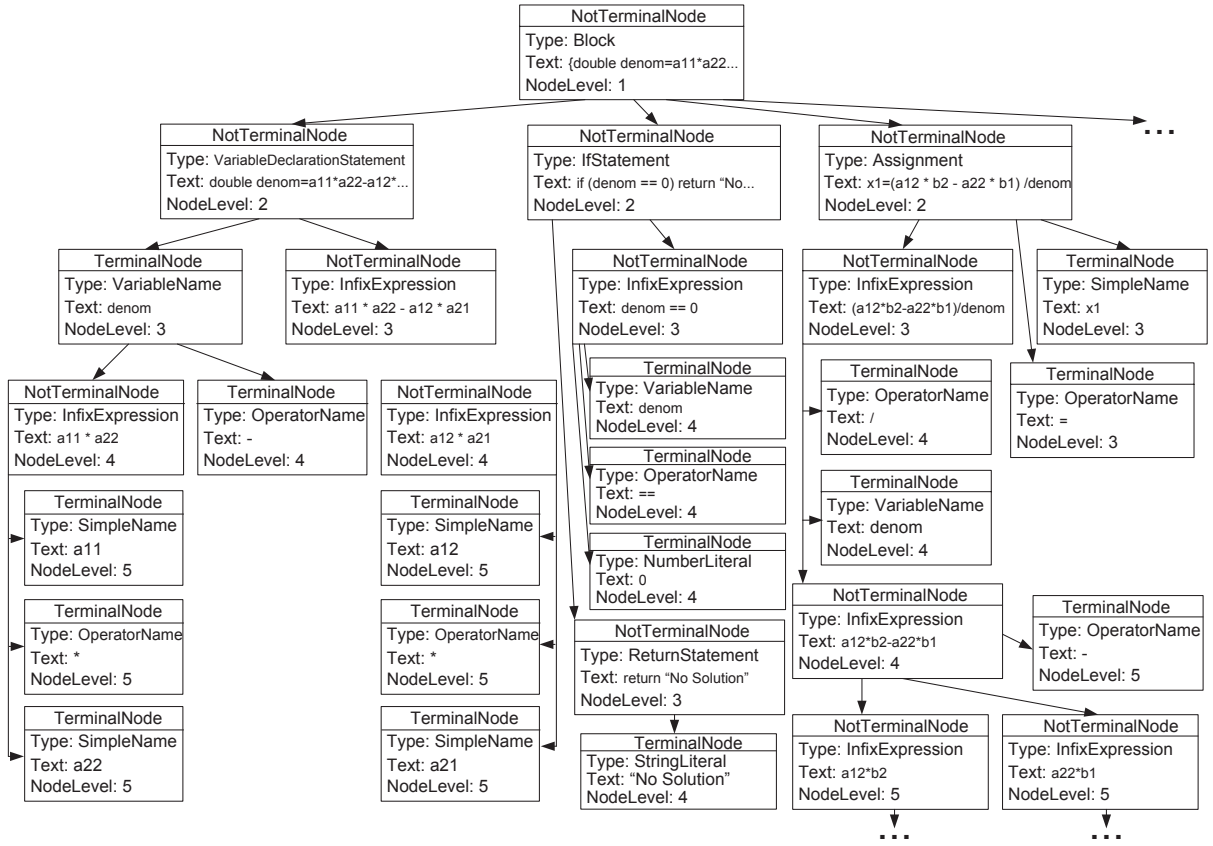
After setting of etalon algorithm parameters there is a formation of decision-making model for problem inner loop in form of Bayesian network. For creation of AST it is necessary to create object of ASTParser class and by means of a function setSource call to specify the java-code of system solving described above. By means of utility ASTView it is possible to receive visual AST representation.

FIGURE 10
THE FRAGMENT OF AST STRUCTURE OF EQUATIONS SYSTEM SOLVING ALGORITHM



It is necessary to transform received AST algorithm in CCT which includes only interesting to us AST nodes and their some characteristics. CCT represents hierarchical CC structure which the trainee should acquire for the linear equations system solving. Received CCT for a considered problem will have the following view:

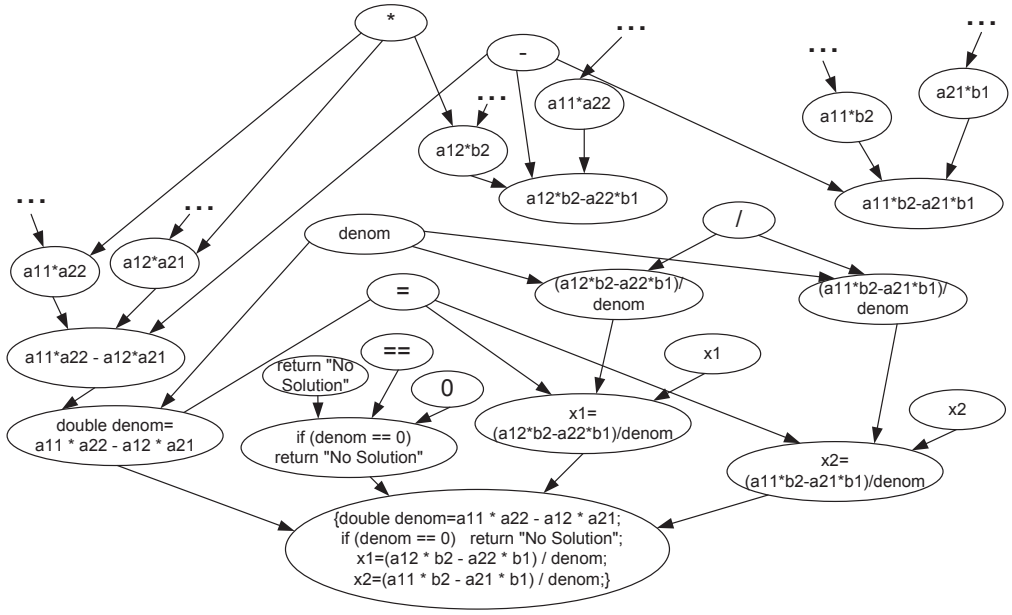
FIGURE 11
THE FRAGMENT OF PROBLEM CCT



CCT consists of two types nodes: terminal (not having descendants, private CC) and non-terminal (having descendants, compound CC). Nodes have an indicator of nesting level concerning root node. For example, the node with the text «a12» has the fifth nesting level concerning the uppermost node of a method main at which the given indicator is equal to 1. Each private CC has the corresponding weight factor defining degree of its importance in a problem solving concerning other components. The weight of compound CC is equal to the sum of weight factors of its components plus its own weight.

Apparently from the scheme represented in figure 11, CCT has some nodes duplicating each other located in different branches of a tree, for example terminal nodes «denom» or «*». In BN made on the basis of CCT duplicated nodes unite in one node, thus a direction of connections between all nodes changes on opposite, i.e. from nodes with biggest nesting level to the smaller ones. Structure of a considered problem BN is presented in a following picture.

FIGURE 12
BN OF PROBLEM SOLVING



Described BN consists of chance nodes, each of which has two conditions: Satisfied (acquirement of CC) and Violated (absence of CC acquirement). Each node has the table of conditional probabilities connected with it (CPT). Values of the table for node A having n ancestors $a_1, a_2, ..., a_n$ are formed as follows:

$$P(A=Satisfied|a_1,a_2,..,a_n)=\lambda,\ P(A=Violated|a_1,a_2,..,a_n)=1-\lambda, \quad (5)$$

At $(a_1=$ Satisfied)^$(a_2=$ Satisfied)^...^$(a_n=$ Satisfied), $\lambda \epsilon (0,1)$.

$$P(A=Satisfied|a_1,a_2,\ldots,a_n)=0,\ P(A=Violated|a_1,a_2,\ldots,a_n)=1, \quad (6)$$

At $a_j=$ Violated, $j \epsilon \{1,2,\ldots,n\}$.

Thus, on the basis of received statistics data, it is 90% of confidence that the trainee acquires compound CC if he acquires all its private CCs. In case the trainee does not acquire one of private CC it is obvious that he will not acquire also a compound component.

For example, for node «a12*b2», having three ancestors «a12», «*» and «b2», CPT will have the following view:

TABLE 1
CPT OF «a12*b2» NODE

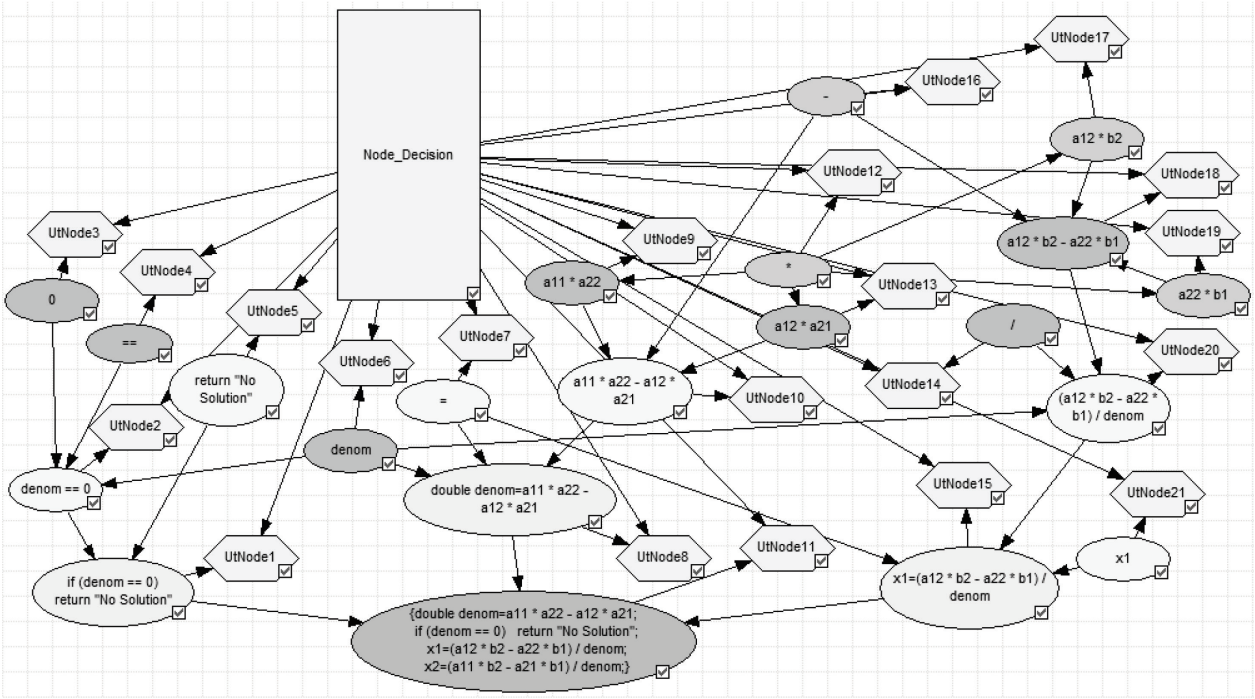
a12	Satisfied				Violated			
b2	Satisfied		Violated		Satisfied		Violated	
*	Satisf.	Violated	Satisf.	Violated	Satisf.	Violated	Satisf.	Violated
Satisfied	0.9	0	0	0	0	0	0	0
Violated	0.1	1	1	1	1	1	1	1

For implementation of function of a choice by system of the most useful next task execution step from the list of alternative variants, BN represented in figure 12, should be transformed in decision-making BN. For construction of the given network, and also possibility of solving of its various

probabilistic inference problems, we will use the tools of platform-independent library SMILE. It is necessary to complement present BN with decision node which list of conditions (alternatives of a choice) will include all network chance nodes. Besides it, it is necessary to connect utility node corresponding to it with each chance node of a network. Utility values of a_i node if decision node has n alternatives a_1, a_2,…,a_n, are defined as follows:

Utility (a_i=Violated) = weight (a_i), utility values of other variants are equal 0, where weight (a_i) - weight value of a_i node, equal to the sum weight of all its private nodes and initial weight value of a_i node. Created through API SMILE interface decision-making BN can be seen by means of graphic editor Genie 2.0.

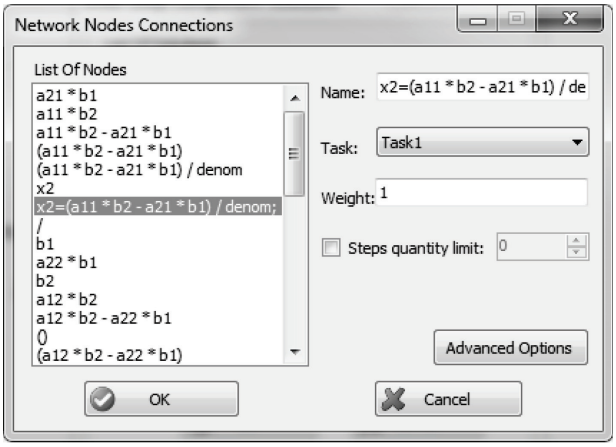
FIGURE 13
THE FRAGMENT OF DECISION-MAKING BN IN GENIE 2.0 EDITOR



In figure 13 chance nodes of the ellipse form are shown by different colors depending on their nesting level in initial CCT. With each of such nodes the utility node is connected with names UtNode1 … UtNode21. The decision block "Node_Decision" has 21 alternative conditions Node1 … Node21, which are corresponded by network nodes and CC associated with them.

After BN formation we receive the list of algorithm CC. Further, at inner loop execution, on each next step the most useful for check CC will be chosen. Therefore it is necessary to define for interesting to us CC tasks for their check and its weight in problem solving algorithm.

FIGURE 14
EDITOR FOR SETTING OF INNER LOOP SUB-TASKS



Thus, from set of the tasks created in the universal environment, some of them will be chosen for outer loop, and a part for inner loop.

After the user answer input in the form of a program code in the inner loop visual component there is execution of its correctness checking algorithm. The algorithm includes some stages represented on figure 15:

1) Formation of a program code for reception of trainee and etalon algorithms output data. The check program is formed of following blocks: 1a - block of input parameters test values; 1b - cycle of testing function call and transfer to it of input data test values; 2a - testing function. It accepts as the input data sample of input parameters test values, passes them through algorithm (trainee's or etalon) and returns values of algorithm output parameters in output stream; 2b_u - program code of trainee algorithm; 2b_e - program code of etalon algorithm; 2c - block of output parameters values returning to output stream of testing function; 3 - program code of the additional functions called from the main algorithm.
2) Fixation of output data of algorithms execution
3) Comparison of the output data of algorithms execution taking into account an admissible error in their divergence
4) Conclusion about correctness of trainee algorithm execution. In case of detection of inadmissible divergence in values of algorithms output parameters the decision on the inner loop execution beginning is made. Thus, there is a loading of etalon algorithm decision-making BN.

Let's consider an example of inner loop work for a problem of two linear equations system solving. In figure 15 it is possible to see differences in blocks 2b_e and 2b_u. In the trainee's algorithm case, when the main system determinant is equal to zero (the system has no decision) is not analyzed. Therefore on the input data second sample we have received the system message about division by zero. Comparison system has made a conclusion that the trainee's algorithm is wrong and has given a command of task inner loop start. The system loads decision-making BN of etalon algorithm which has been generated even at designing of inner loop visual component. After execution of the inner loop task there is an updating of BN nodes conditions including a decision-making node. End of inner loop execution occurs after performance of all its sub-tasks or when values of possible decisions of decision-making node are equal to zero. We will consider one of the possible inner loop scenarios, a two linear equations systems problem.

FIGURE 15
COMPARISON OF RESULTS OF TRAINEE AND ETALON ALGORITHMS EXECUTION

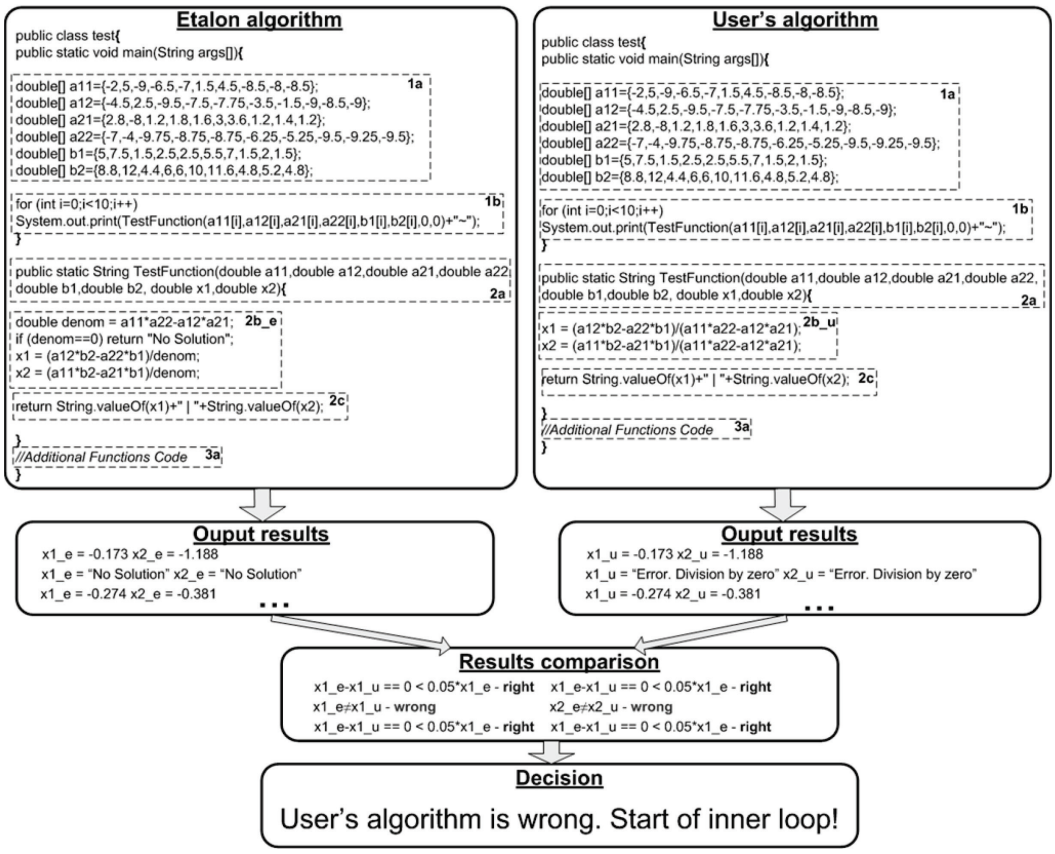


TABLE 2
THE DESCRIPTION OF THE ICTP POSSIBLE INNER LOOP SCENARIO EXECUTION

№	Current step	Result	Next step decision
1	Examination of Kramer's method in a whole, for example restoration of all actions and formulas sequence of unknowns calculation	Violated	x1=(a12*b2 - a22*b1) / denom; or x2=(a11*b2 - a21*b1) / denom;
2	Task on x1 calculation formula assemblage from separate blocks	Satisfied	x1=(a12*b2 - a22*b1) / denom;
3	Task on x2 calculation formula assemblage from separate blocks	Satisfied	if (denom = = 0) return "No Solution"
4	Task for knowledge check of condition of equations system solution existence	Violated	denom = = 0
5	Task for situation studying when the main determinant is equal to zero	Satisfied	Exit form inner loop

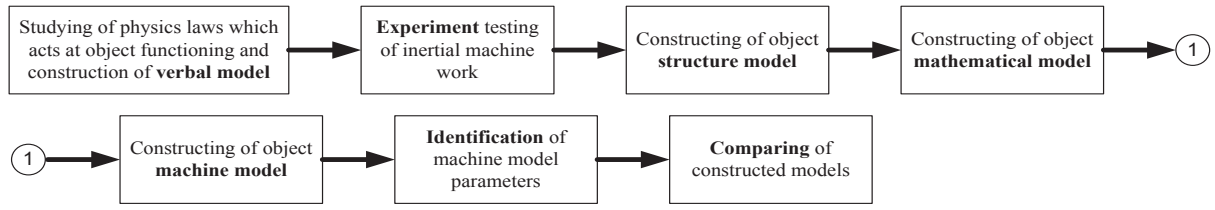
After the system decision about the trainee's algorithm program code to be incorrect there is a decision-making on providing of the task for Kramer's method knowledge examination in a whole. After its wrong execution recalculation of BN nodes values leads to conclusion that the greatest utility have CC of x1 and x2 unknowns calculation. The system makes the decision on displaying of the task for x1

calculation formula assemblage. After the successful performance the knowledge of the x2 calculation formula is checked. We model a situation at which the user possesses x1 and x2 calculation skills. That's why the decision on knowledge checking of condition of equations system solution existence is made. The final task of the inner loop is graphic demonstration to the user of a situation which arises when the main system determinant is equal to zero.

IMPLEMENTATION OF SYSTEM MODELING COURSE IN UNIVERSAL ENVIRONMENT

The system modeling course (Kulik, Chukhray, Pedan, & Anzenberger, Development of the automated laboratory practical work at the course <Modeling of Systems>, 2008) was created with the help of universal environment. Course is aimed to studying of modeling on example of such simple physical object as inertial machine. The trainee should model process of machine movement on an inclined plane and its stops on a horizontal site. Course structure is presented on figure 16.

FIGURE 16
STRUCTURE OF SYSTEM MODELIING COURSE



Course consists from seven ICTP, which implement intelligent method of its outer loop formation and includes a lot of inner loop tasks. Due to inner loop component trainee has a big freedom in designing of studied object mathematical model. Using of universal environment at system modeling course construction and testing of created ICTPs has led to considerable economy of time expenses and has shown high learning efficiency due to realization of the developed adaptation methods of teaching material to the trainee's competence level.

CONCLUSIONS

As a result of performed research the universal environment for ICTPs graphic creation has been developed. The universal environment provides adaptation of the tutoring process to the level of competence of each trainee at two levels: the outer loop of ICTP and the inner loops of its tasks. Mechanisms of individual studying program formation for each of the loops are described. By means of the universal environment the intelligent tutoring program on system modeling has been implemented. ICTP testing in real conditions of the tutoring process has shown a high degree of trainee current competence level estimation, allowing the development of individual sequence in tutoring to achieve the planned educational results.

REFERENCES

Chukhray, A., Pedan, S., & Vagin, I. (2010). Modeling of Engineering Skills Acquirement Process Via Dynamic Bayesian Nets. *In Proceedings of the East-West Fuzzy Colloquium*, 257-263.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.

GeNIe. *Decision Systems Laboratory*. (n.d.). Retrieved from University of Pittsburgh: http://genie.sis.pitt.edu/

Kulik, A., Chukhray, A., Pedan, S., & Anzenberger, P. (2008, September). Development of the automated laboratory practical work at the course <Modeling of Systems>. *In proceedings of the Interactive Computer Aided Learning Conference*, 24-26.

Kulik, A., Chukhray, A., Pedan, S., & Hall, S. (2011, February). The method of computer tutoring program pedagogical actions formation for its inner loop tasks. *Proceedings of ASBBS Annual Conference*, 18, (1), 626-637.

LeActivieMath. (n.d.). Retrieved from Language-enhanced,User-Adaptive, Interactive eLearning for Mathematics: www.leactivemath.org.

Moodle Service Network. (n.d.). Retrieved from www.moodle.com.

Murray, R. C., Lehn, K. V., & Mostow, J. (2006). Looking Ahead to Select Tutorial Actions:A Decision-Theoretic Approach. *International Journal of Artifical Intelligence in Education,* 14, (3,4), 235-278.

Russell, S. J., & Norvig, P. (2003). *Artificial Intelligence: Amodem Approach (2nd ed)*. Upper Saddle River, New Jersey: Prentice Hall.

The Eclipse Foundation. (n.d.). Retrieved from http://www.eclipse.org/

VanLehn, K. (2006). The Behavior of Tutoring Systems. *International Journal of Artifical Intelligence in Education,* 16, (3), 227-265.

Woolf, B. (2009). *Building intelligent interactive tutors:Student-centered straregies for revolutionizing e-learning.* Burlington MA: Morgan Kaufman Publishers.