Designing a Decision Engine for Adaptive Training Simulators

Aneliya Ivanova, Elena Yakimova

Abstract: *The paper describes an approach to improvement of existing training simulators, integrated in Virtual laboratory on Computer Organization. The main goal of the initiative is making the simulators adaptive to students' ability and level of knowledge. The motivation for this upgrade is discussed and key elements of research, design and implementation are introduced. Since the core component that ensures achievement of adaptivity is the decision engine of the simulators, the discussions of this paper are focused to its design and implementation.*

Key words: *Training simulator, Adaptive, Training Scenario, Difficulty Level, Decision Engine, Simulation Parameters, Rule Base, Rule Based System.*

INTRODUCTION

It is well known that training simulations are one of the most appropriate ways to increase learner's engagement. A survey, described in [1] shows that using an interactive simulation exercise results in increased student engagement levels. Iverson [2] also suggests that building of learner engagement requires incorporation of simulations into training process. This largely applies to the Millennial students that distinctly show preferences to first-person learning (by doing, exploring, and discovery) [3].

Having incorporated simulations in the training process, we have to go further and consider how to increase the students' engagement to the simulations themselves. What is the motivation for this?

The Virtual Laboratory (VLab) on Computer Organization was implemented in 2002 and has been intensively used within practical workshops on the corresponding course. The simulators, included in the VLab were designed for highly motivated students having preliminary knowledge on the subject matter. During the first years of adoption the simulators used to achieve a good deal of user engagement. The students were enthusiastic to control the simulations and when completed an operation they were curious to play the simulation again and to accomplish the task with better performance – with less or no mistakes, and for shorter time.

Recently we have remarked that the students' engagement and motivation decreases year after year. More and more students find the simulators difficult and do not feel like trying again to perform better.

Out of doubt, our students have changed and we have to revise and reorganize our training tools in order to keep them engaged. The Millennial students would appreciate game alike training tools – not in terms of GUI, but in terms of user interactions.

It is obvious we could not teach foundations of Computer Arithmetic using 3D virtual worlds, but we could adopt an interaction manner, inherent for a computer game (CG). Gaming elements provide motivation, structure, and a goal, and also create a competitive environment for learning. Including gaming elements, like score-keeping, competition, and surprise variables can increase the entertainment value and fun of a simulation [4]. Most of CG offer to the players to select a level of difficulty and after a successful action, the player gets a kind of a bonus. He (she) is also provided with instructions and directions on each step of the game.

Each simulation must have a way to judge and direct learners. In technology-based simulations, the design and sophistication of decision engine is the key to the simulation's success [4]. In this paper we describe our approach to improve the existing training simulators on Computer Organization in the context of presented above discussions, and to make them adaptable to students' abilities. This will be achieved through designing a new decision engine for the simulators and upgrading their user interface (UI).

RESEARCH

Why is necessary to seek after adaptivity? That is because the Millennial students are addicted to environments adapting to their interests, needs and abilities, so they naturally expect the same from their training tools. Before designing the set of training scenarios, we have to consider the level of difficulty for each one. Niehaus and Riedl [5] present the concept that Zone of Proximal Development (ZPD) (Fig.1) being a developmental theory, could be applied to training scenarios as well. It provides a representation of the balance between learner ability and scenario difficulty. The ideal situation is when the learner does not leave out the ZPD into the zone of confusion or zone of boredom. A training scenario is considered effective if falls within the ZPD (the patterned zone in Fig.1). A similar idea shares Sierra [6] providing a visual representation of "challenge/ability" balance (Fig.2). According to her, the best user engagement is achieved when the challenge, provided by an application is adequate to user's ability.

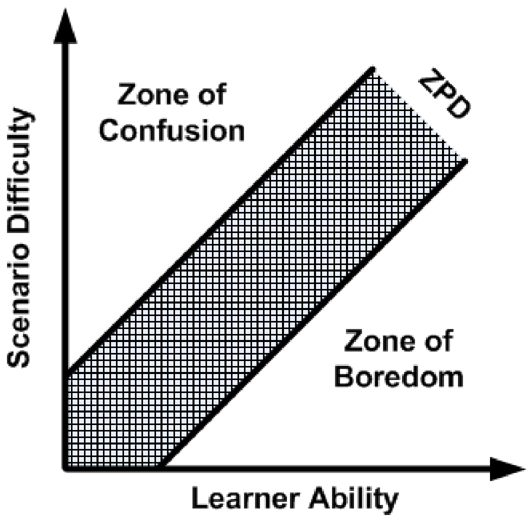

Fig.1. The Zone of Proximal Development

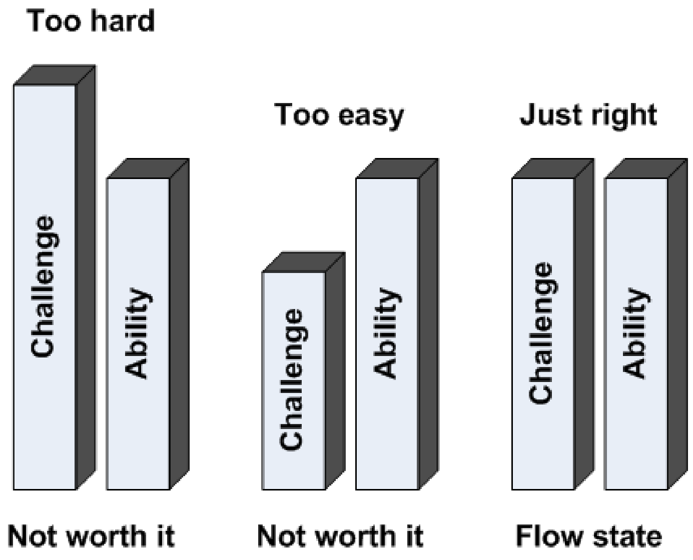

Fig.2. Keeping users engaged

The next consideration is about the way of selecting the most appropriate to student's ability scenario. There are two approaches possible. The first one gives the responsibility of level selection to the system. In this case the student completes a test, and on the basis of the test results, the system defines the appropriate level of difficulty. Giving account to the short attention spans and expectation of instant gratification that are inherent for Millennial students as well as their trend to overestimate their efforts, this approach is not likely to work at 100%, because the students may refuse to complete the test. The Millennial students prefer to have at disposal an array of choices, so the idea to put the level selection to their responsibility, seems to work better.

The last consideration is focused to the user interface. It should be reorganized to users with visual-kinesthetic learning style (that is actually the Millennial's one). In this connection we have to seek after: graphical representation of concepts, visual rather textual hints and help, enhanced animation. Since the most of our future student are game-addicted and take the learning as a fun, they would appreciate getting rewards and interaction style that is not so academically rigorous.

DESIGN AND IMPLEMENTATION

The existing simulators, included in VLab are implemented as separate applications of two types – those that simulate how the arithmetic operations are executed in ALU, and simulators of the main CPU blocks and of various types of CPU. In this paper we discuss the improvement of the first type simulators. The "arithmetic" simulators will be integrated into a single application. The decision engine will operate in two layers – external, that controls application's behaviour, and internal – that drives the selected simulation itself. Therefore, the design is organized in two flows. The ***external layer*** of decision engine (Fig.3) deals with selection of operation that is to be simulated, selection of micro-algorithm for operation execution, selection of difficulty level and at the end of the session – saving the student's performance (student ID, date, time, scores achieved).

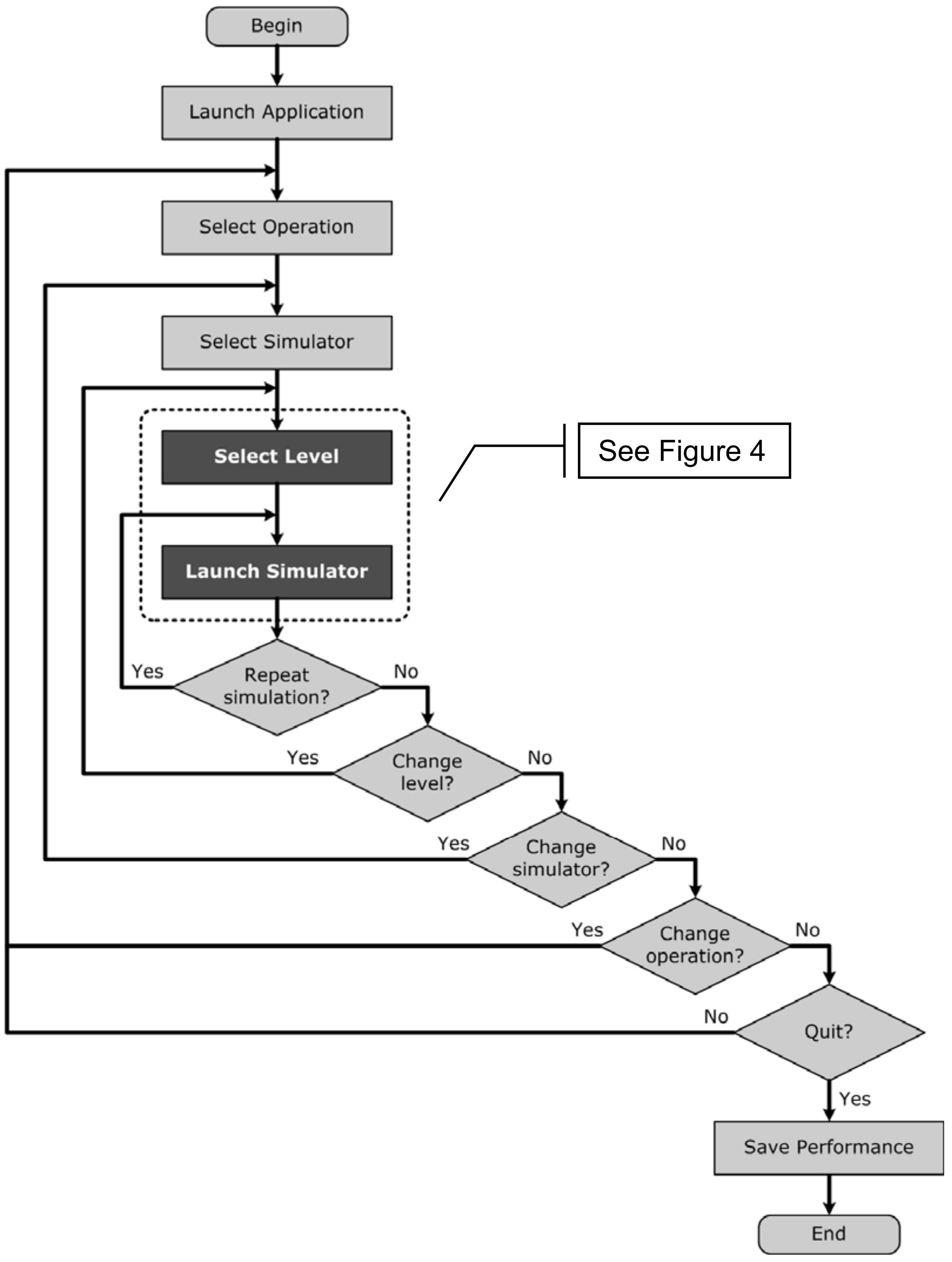

Fig.3. Algorithm of operation for decision engine's external layer

As could be seen on Figure 3, when the application is launched a start screen appears and the student is able to select one of the basic arithmetic operations, executed by ALU (addition, multiplication, division, etc.) that is of interest to him (her). After selection of operation, the student is offered to select a simulator that reproduces one of the known micro-algorithms for executing the selected operation. After the simulator has been selected, it is necessary to be defined the simulator's level of difficulty. There are 5 levels provided (Fig.4) and they are actually tied together with the self-estimated level of knowledge of the student. Level 1 is relevant to "Lack of knowledge" option, as level 5 fits to "Good knowledge" option. To each level is assigned a scenario, describing user–simulation interaction. When the difficulty level is defined, the simulator is started in the appropriate mode. To avoid student's self-overestimation, when a level is selected, and the previous one is not already passed by the student, he (she) will be asked to play it once to show ability to manage with the desired level.

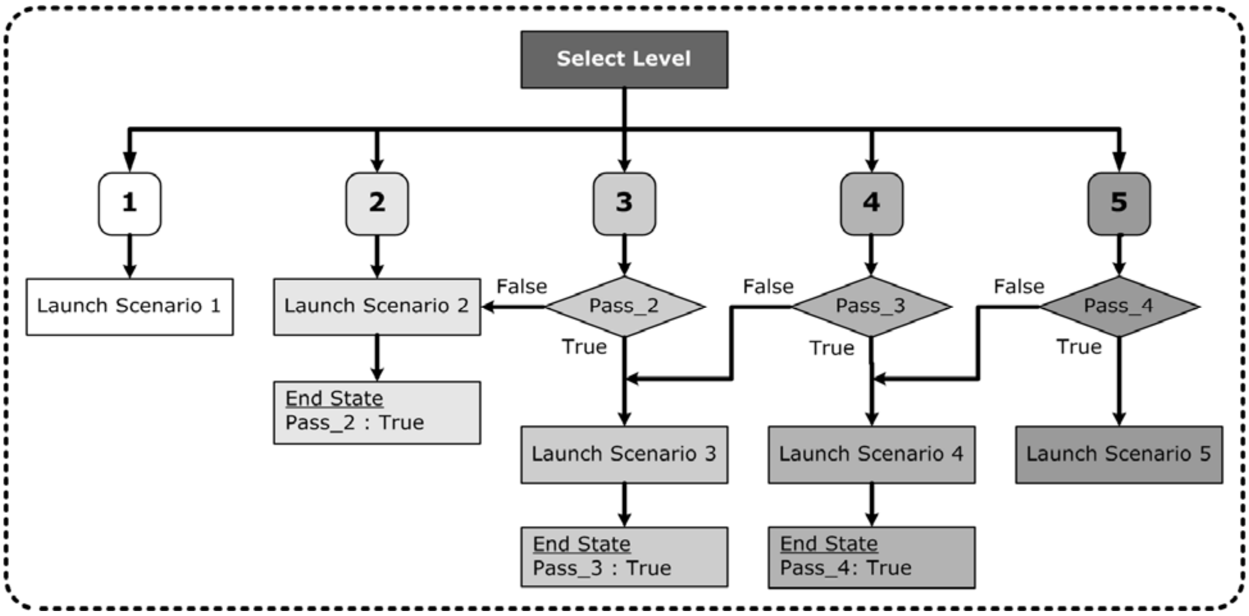

Fig.4. Algorithm for level selection and scenario launch

After the simulator is launched, the decision engine switches the control to the ***internal layer***. When working with arithmetic simulators of VLab the students are expected to click control buttons simulating the signals that enable execution of operation's micro-algorithm. The internal layer of decision engine tracks the student's actions, checks them for consistency and depending on situation, provides help or visualizes the animation. Doing this, the engine follows the simulation parameters, set for the selected scenario.

The formal representation of a scenario embraces two sets of simulation parameters. The first one deals with user interaction and describes how the hints, help and rewards should be provided and errors handled. These parameters will be referred as ***"interaction"*** ones. The set of interaction parameters is independent of selected simulator, it is common for each scenario, as only the parameters' values differ.

The second set of parameters is unique for each simulator and describes the micro-algorithm steps as well as the content of hint and help messages. These parameters will be referred as ***"algorithmic"*** ones (Fig.5).

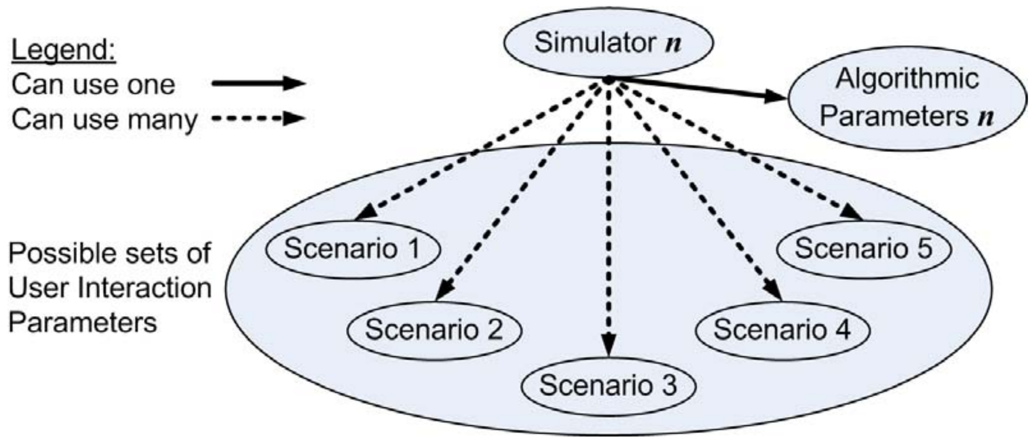

Fig.5. Communication between simulator and simulation parameters

Taking into account the above considerations, it is advisable the decision engine to be based on the production systems concept and to be implemented as a rule based system. Since the operation of external layer was depicted by the algorithm on Fig.3, on Figure 6 is represented the block diagram of the internal layer of Decision engine. It may be viewed as consisting of three basic components: a rule base (set of rules), a workspace (set of facts) and an inference engine (interpreter of the rules).

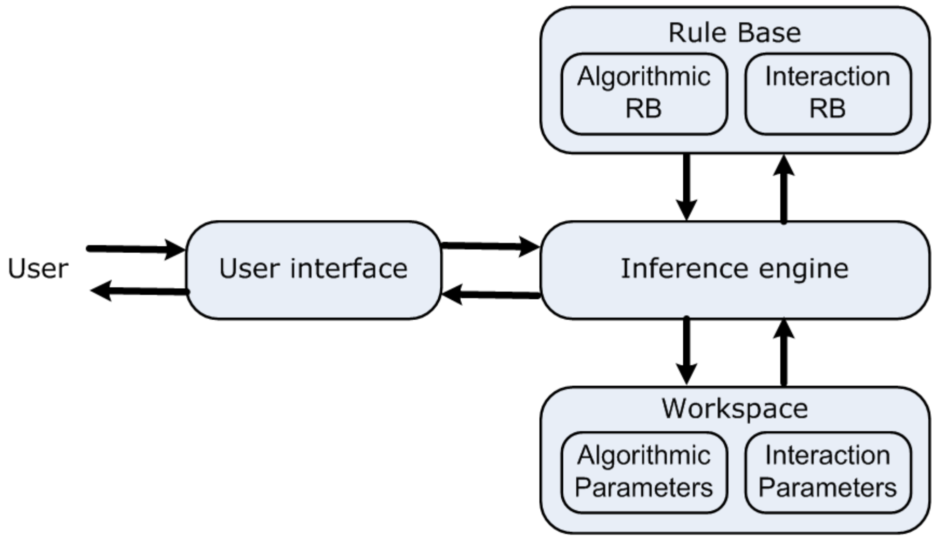

Fig.6. Block diagram of the internal layer of Decision engine

Interface enables the user to communicate with the system and provides access to the inference engine.

The rule base determines how the data stored in workspace modifies. Rule base is a set of rules in the form of if-then statements, representing relationships between the facts in workspace. Each rule is a conditional statement that links given conditions to conclusions or outcomes and has the form:

IF ***(conditions are satisfied)***
THEN ***(take an action or deduce new conclusions)***

A rule may have more than one premises, usually combined either by AND or by OR, and more than one conclusion. The rules are entered as separate statements and it is the inference engine that uses them together to draw conclusions. In this case the RB is composed by two bases – containing algorithmic and Interaction rules. Below are presented some of the algorithmic rules for the simulator of micro-algorithm for multiplication by right shift of the Product:

```
Rule 1: IF ((Ld_Rx : Enabled)
            AND (Control ID = Ld_Rx))
```

```
        THEN SET_PARAMETER (Ld_Rx : Disabled ; Ld_Ry : Enabled)

Rule 2: IF ((Ld_Rx : Enabled)
            AND (Control ID <> Ld_Rx))
        THEN SET_HINT (Hint : 'The correct MO is: Load Rx.')
...

Rule 5: IF (Xn = 1)
        THEN SET_PARAMETER (Out_Rz : Enabled)

Rule 6: IF (Xn = 0)
        THEN SET_PARAMETER (Out_Rz : Disabled ; Out_Ry : Disabled;
                                              SR_Rz : Enabled)

Rule 7: IF ((Out_Rz : Enabled)
            AND (Control ID = Out_Rz))
        THEN SET_PARAMETER (Out_Rz : Disabled ; Out_Ry : Enabled)

Rule 8: IF ((Out_Rz : Enabled)
            AND (Control ID <> Out_Rz))
        THEN SET_HINT (Hint : 'The correct MO is: Output code Rz.')
...

Rule 12: IF ((SR_Rz : Enabled)
            AND (Control ID <> SR_Rz))
        THEN SET_HINT (Hint : 'The correct MO is: Shift right Rz.')
```

The workspace contains data that is to be used to derive a conclusion. It stores information about values of interaction and algorithmic parameters at any time in the form:

<Parameter ID : Value>.

An example is provided in Table 1. It describes the combination of micro-algorithm (MA) for multiplication by right shift of the Product and Difficulty Level 2, as the current step of MA is: "Output code from Product Register (Rz) to the Adder".

Table 1. Simulation parameters, stored in workspace

Interaction parameters	**Algorithmic parameters**
Auto_Mode : False	Ld_Rx : Disabled
MA_Blink : True	Ld_Ry : Disabled
MO_Descr : True	Out_Rz : Enabled
Button_Blink : True	Out_Ry : Disabled
Error_Count : True	St_Rz : Disabled
Error_Mess : True	SR_Rz : Disabled
Error_Hint1 : True	SR_Rx : Disabled
Error_Hint2 : False	
Bonus_Count : False	

The Inference engine (rule interpreter) applies rules to the available information in the workspace. The rule interpreter uses forward chaining inference method – starts from the initial content of the workspace and works forward to the conclusions. It searches the rule base until it finds a rule where the IF clause is known to be true. When found it can conclude, or infer, the THEN clause, resulting in the addition of new information to the workspace.

CONCLUSIONS AND FUTURE WORK

In this development we have chosen to put the responsibility of difficulty level selection to the students, basing on the assumption that this solution is more relevant to Millennials' learning style. As future development we are planning to design and implement a decision engine that determines what level of difficulty is most appropriate to student's ability after analyzing the results, achieved by the student on completion of preliminary test. Such a test should be completed on each level change as the decision engine will select random test questions from a database. The final idea is a comparison of these two approaches and pointing out the better and more effective one. The conclusion will be drawn on the base of an inquiry among the students, as well as the stored students' performance.

REFERENCES

[1] Bulger, M., R. Mayer , K. Almeroth, S. Blau. Measuring Learner Engagement in Computer-Equipped College Classrooms. Journal of Educational Multimedia and Hypermedia. 17 (2), 2009, pp. 129-143. Chesapeake, VA: AACE.
Retrieved from http://www.editlib.org/p/23524.

[2] Iverson, K. Engaging the E-Learner: Interaction is Not Education. Learning Solutions e-Magazine, February 25, 2008.

[3] Oblinger,D.G., J.L.Oblinger. Educating the Net Generation, EDUCAUSE, 2005. Available online at: www.educause.edu/educatingthenetgen/.

[4] Vaughan, M., K. Himmel. FORCES SHAPING OUR FUTURE - ELEMENTS OF SIMULATION , White Paper, Regis Learning Solutions, 2006.

[5] Niehaus, J., M. O. Riedl. Scenario Adaptation: An Approach to Customizing Computer-Based Training Games and Simulations. Proceedings of the AIED 2009 Workshop on Intelligent Educational Games, Brighton, UK, 2009.

[6] Sierra, K. Can you have too much ease-of-use? Creating passionate users, Available online at:
http://headrush.typepad.com/creating_passionate_users/2005/03/can_you_have_t o_1.html

ABOUT THE AUTHORS

Aneliya Ivanova, Principal Lecturer, PhD, Department of Computing, University of Rousse, Phone: +359 82 888 827, E-mail: AIvanova@ecs.ru.acad.bg.

Elena Yakimova, Principal Lecturer, PhD Student, Department of Computing, University of Rousse, Phone: +359 82 888 276, E-mail: ESimeonova@ecs.ru.acad.bg.

ACKNOWLEDGEMENT

This work was partially supported by the European Social Fund and the Bulgarian Ministry of Education and Science under the Operation programme "Human Resources Development", Grant # BG051PO001/07/3.3-02/7.