AN UML PROJECT OF A TASK-ORIENTED ENVIRONMENT FOR TEACHING ALGORITHMS

Galina Atanasova, Irina Zheliazkova

Abstract: This paper continues the author's team research on development, implementation, and experimentation of a task-oriented environment for teaching and learning algorithms. This environment is a part of a large-scale environment for course teaching in different domains. The paper deals only with the UML project of the teaching team's side of the environment.. The implementation of the project ideas is demonstrated on a WINDOWS-based environment's prototype.

Keywords: problem solving support, adaptive and intelligent technologies, UML, task-oriented environment, algorithm flowchart

ACM Classification Keywords: K.3.2 Computer and Information Science Education-Conference proceedings

Conference: The paper is selected from Third International Conference "Modern (e-) Learning" MeL 2008, Varna, Bulgaria, June-July 2008

Introduction

For about 20 years the main *Intelligent Tutoring Systems* (*ITS*) research was centered on *Problem Solving Support Technologies* (*PSST*). The main goal of a PSST is to help the learner (*L*) in the process of solving a domain problem. The delivering and sequencing of the background material (*BM*) was supposed to be performed outside the ITS usually by a human teacher. In his well-known survey [Brusilovsky, 1999] identified three *PSST* and called them respectively: intelligent analysis of learner's solutions, interactive problem solving, and example-based problem solving support.

Intelligent Analysis of Learner Solutions (*IALS*) deals with the *L*'s final solution of domain problems no matter how this solution was obtained. To be considered as intelligent, a solution analyzer has to decide whether the solution is correct or not, find out what exactly is wrong or incomplete, and possibly identify which missing or incorrect knowledge may be the reason for the error. In the special literature the last functionality is referred to as knowledge diagnosis. An intelligent analyzer can ensure information for the *T* to update the *BM*. Moreover these analyzers can provide the *L* with an extensive error feedback. This functionality is referred to as knowledge remediation.

Interactive Problem Solving Support Technology (IPSST) is a more recent and powerful technology inherited also from standalone non-web ITS. Instead of waiting for the final solution, this technology can provide the *L* with an intelligent help on each step of the problem solving. The level of help can vary from simple signaling about a wrong step to giving a hint for executing the next step for the *L*. The systems, which implement this technology (often referred to as interactive tutors) can capture the L's actions, understand them, and use this understanding to provide help and/or to update the *BM*.

Example-Based Problem Solving Technology (*EBPST*) that is the newest one is helping the *L* to solve new problems not by articulating their errors, but by sequencing them relevant successful problem solving cases from his/her earlier experience.

Both IALS and EBPST although passive, e.g. working by the *L* request, appear to be very natural and useful. Moreover, an old standalone adaptive and intelligent learning system, which uses these technologies, could be relatively easy imported to the web by implementing a *Common Graphical Interface* (*CGI*) gateway to the old standalone system. An important benefit of these technologies in the web context is their low interactivity. This is especially important in the case of slow INTERNET connection. Currently, these technologies dominate in the web context over the more powerful *IPSST* – the last technology migrated to the web.

The web put the begging of a new technology that hasn't parents among the standalone non-web ITS. In the mentioned survey this technology is called *Learner Model Matching Technology (LMMT)* due to its ability to analyze and match many learners' models at one and the same time. The author also identified two kinds of *LMMT* and respectively called them *Adaptive Collaboration Support Technology* (*ACST*) and *Intelligent Class Monitoring Technology (ICMT)*. The goal of the *ACST* is to use system's knowledge about different learners to form a matching group for different kinds of collaboration. The *ICMT* is also based on the ability of web to compare models of different learners. However, instead of searching for a match, the system searches for a mismatch. The goal is to identify the learners who are essentially different from their classmates due to variety of reasons. For example, these learners cold be progressing too fast, or too slow, or simply have passed much less BM than others. In any case, such learners more than others ones need a special teacher's attention, more detailed BM presentation and easier problems to solve.

For several years we are working on development, implementation, and experimentation of a *Task-Oriented Environment for Teaching and Learning Algorithms* (*TOETLA*) in which a combination of the above-presented technologies is incorporated [Zheliazkova I. & G. Atanasova, 2004]. The present paper presents the environment project in the *Unified Modeling Language (UML)* mainly from the course team's point of view. The rest of the paper is organized as follows. The common use case diagram for all kinds of environment users is discussed in the next section. The administrator's activity diagrams are considered in the third section. The next two sections present both UML activity and sequence diagrams for the author and teacher respectively. In section five the user interface of the WINDOWS-based prototype of the TOETLA is shown by means of several screenshots. The conclusion outlines the paper contributions and the authors plan for a near future.

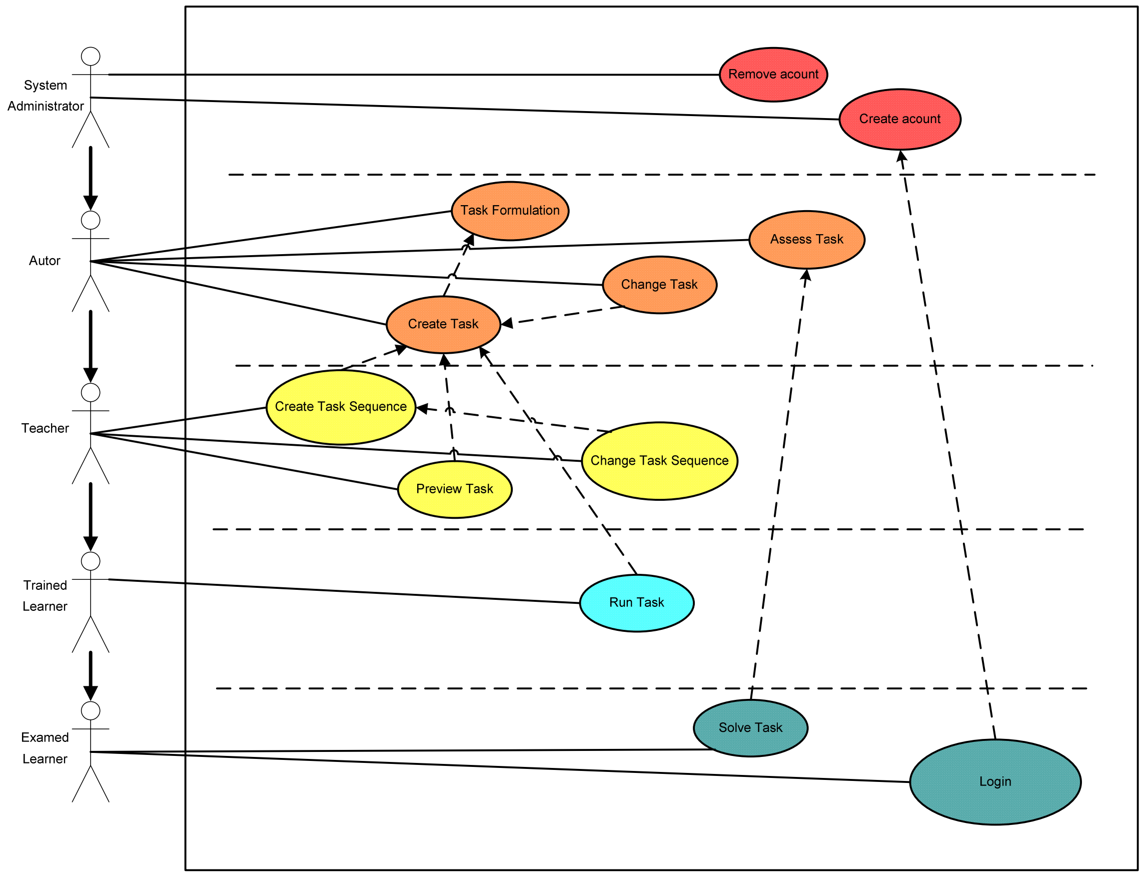


Figure 1. Environment Use Case Diagram

Use Case Diagram of the Environment Project

Nowadays the UML is accepted as a standard language for software project description due to the platform-, technology-, and program-independence of this graphical language [Fowler, 2004]. The common use case diagram presenting the functional features of the TOETLA and the roles of its users (also referred to as actors) are given on fig. 1. The environment has four kinds of users, namely: *System Administrator* (*SA*), *Author* (*A*), *Teacher* (*T*) and *Learner* (*L*), e.g. all participants in the teaching process. The strongly defined hierarchy in their roles and access rights is depicted with vertical arrows. That means that each registered user on upper level can play the roles of the lower levels users. Therefore, the SA can play the role of the A, T, and L, the A the role of the T and L, and the T the role of the L. Also an upper level user can authorize the next level user to play his/her role for a time period. In the context of TOETLA the A is a person who creates tasks for construction of flowcharts used for presentation, training, and assessment of the L's task solution by the environment. The T is responsible for planning, monitoring, and assessing the exercise consisting of this type of tasks and performed by an individual L or a group of learners. The L registered in the TOTLEA without user name and password, e.g. as a guest [Shoikova E. et all., 2006] we called trainee. He/she can monitor the demonstration of the A's task solution step by step. The L registered with user name and password we called examinee. He/she can perform an exercise in a way similar to the A for a fixed time in order to be assessed in accordance with the T didactic skills and preferences.

Administrator's Activity Diagrams

It is essential for consistency and reliability of a teaching and learning environment to control the user access preventing no-authorized users from accessing the system information resources and to use only the resources that are permitted [Noninska I., 2007]. In the TOTLEA only the SA is allowed to register and authorize all other users by creating and removing the user accounts with the corresponding rights and to control their access to the environment. The simple UML activity diagrams presenting the sequence of his/her actions to create a user account and remove it (if necessary) is shown respectively on fig. 2 and fig. 3.

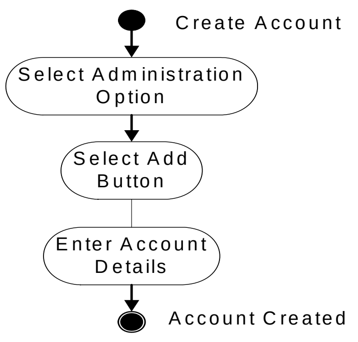


Figure 2. Create Account Diagram

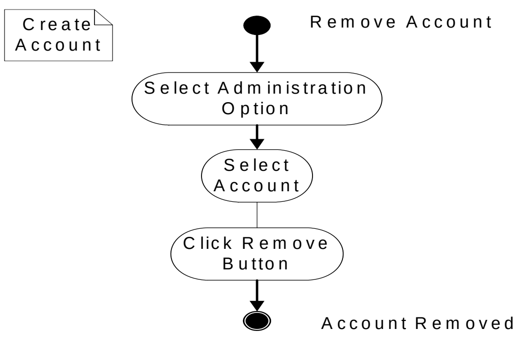


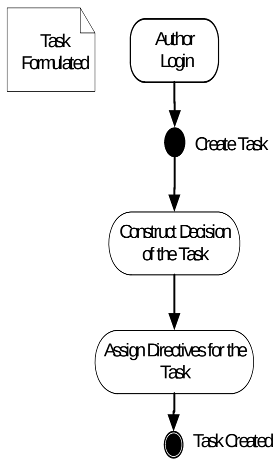
Figure 3. Remove Account Diagram

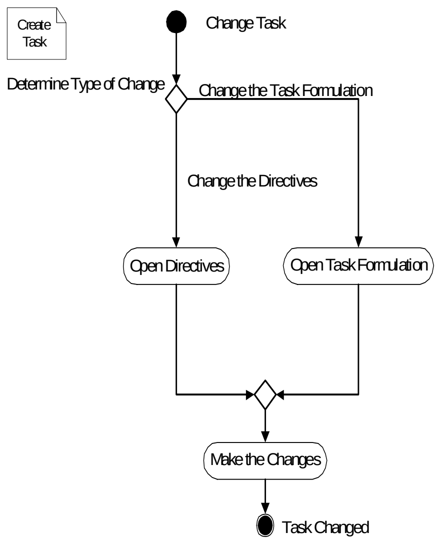
Activity and Sequence Diagrams of the A

The TOTLEA is helping the A with support of a large homogeneous base consisting of tasks for flowchart construction. The domains taught and the type of the corresponding algorithms (whether computational or for decision making support) does not matter. That means that this base could be common for several couses.

The A's activity diagrams to create, change and assess a base task are shown respectively on fig. 4, fig. 5, and fig. 6. The fig. 7 presents the A's sequence diagram. To become clearer for the reader these diagrams needs some comment. An essential environment feature is it allows the A to give the T some didactic recommendations.

The A can use several local key directives to allow or permit the L while solving a task to print the A's solution, to save the L's solution, to see the A's algorithm description, and so on. Other recommendations concern the planned time for the task performance, kind of the task assessment, assessment scale, and so on.


Task Formulated
Author Login
Create Task
Construct Decision of the Task
Assign Directives for the Task
Task Created



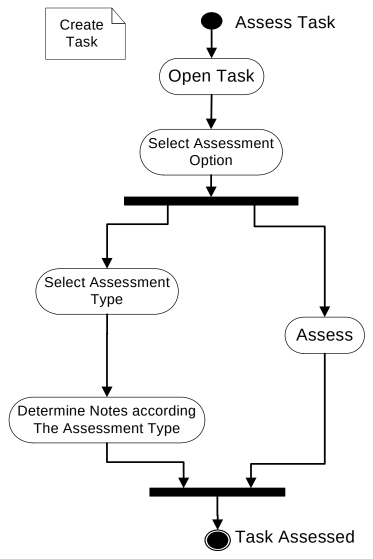


Figure 4. Create Task Activity Diagram

Figure 5. Change Task Activity Diagram

Figure 6. Assess Task Activity Diagram.

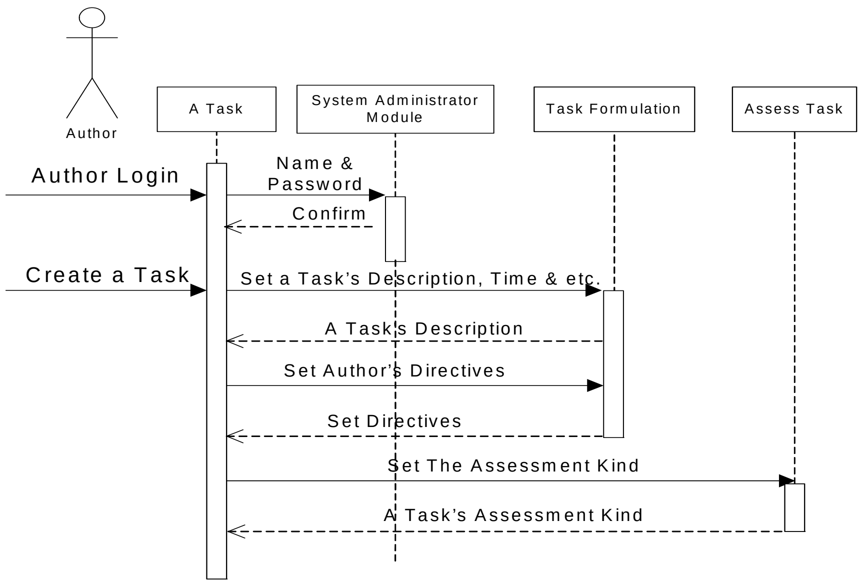


Figure 7. Author's Sequence Diagram

Activity and Sequence Diagrams of the Teaching Team

When work alone the T has access to the task base in order to preview it and select appropriate tasks for each exercise (whether individual or for a group of learners). He/she can add or remove a task in an exercise, choose the number of its tasks and change their sequence. In accordance with the exercise goal, A's preferences, and his/her recommendations the T plans the exercise, changing its automatically computed parameters as well as the A's key directives. The sequence diagrams on fig. 8 and fig. 9 correspond the situation when the A and T work in a parallel to create and change the exercise tasks. The sequence diagram on fig. 10 also corresponds the teamwork but in a long time process depicted as a vertical rectangle. On the top of the figure the environment's units for the process implementation are shown. The solid arrows are used for the users (A and T), and the dash ones for the environment units' reactions.

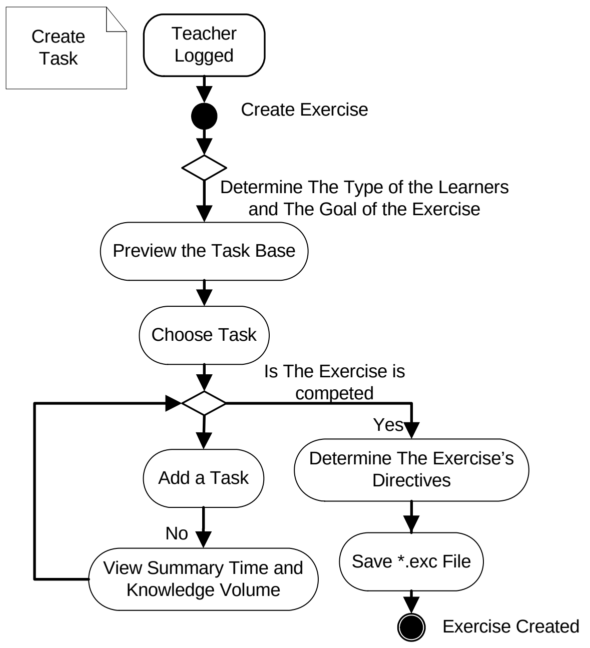

Create Task
Teacher Logged
Create Exercise
Determine The Type of the Learners and The Goal of the Exercise
Preview the Task Base
Choose Task
Is The Exercise is competed
Yes
Add a Task
Determine The Exercise's Directives
No
View Summary Time and Knowledge Volume
Save *.exc File
Exercise Created

Figure 8. Create Exercise Activity Diagram

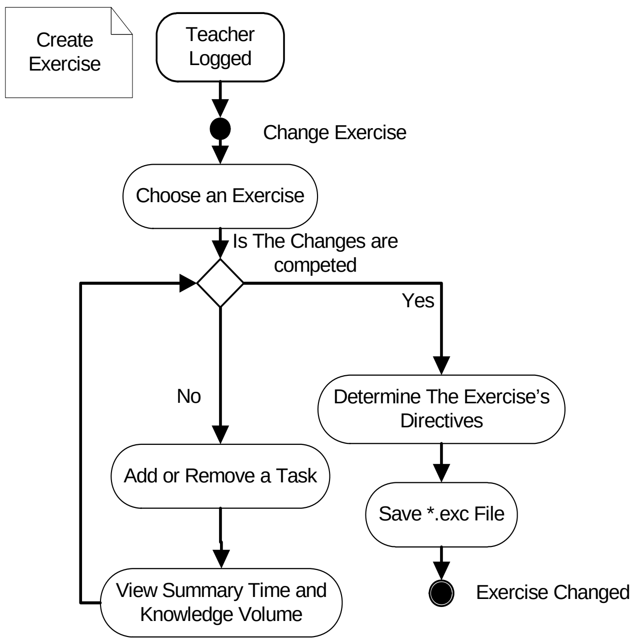


Figure 9. Change Exercise Activity Diagram

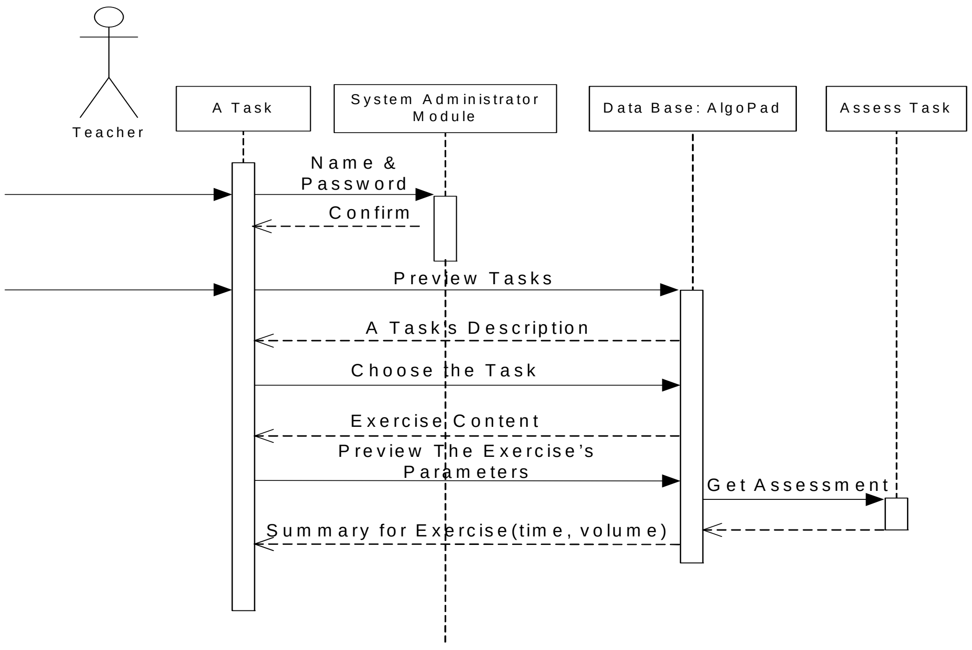
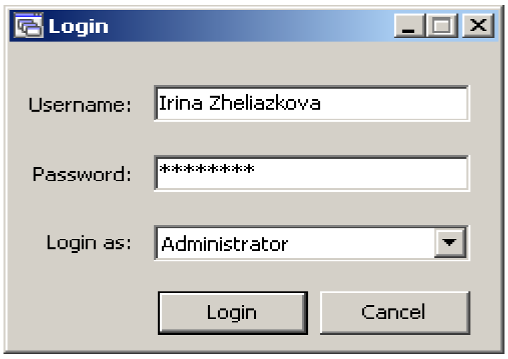


Figure 10. Teacher's Sequence Diagram

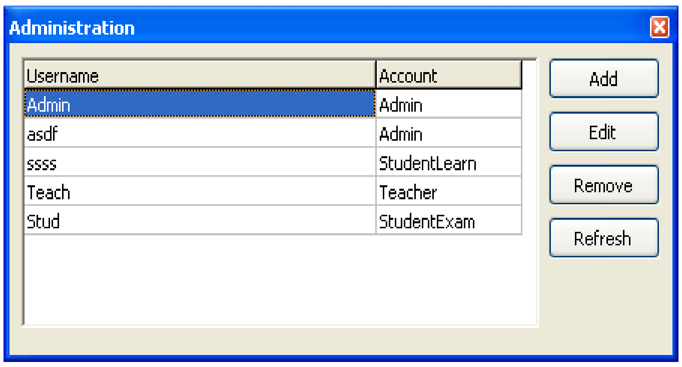
Interface of the WINDOWS-based Prototype

For implementation of the WINDOWS-based environment prototype DELPHI visual programming environment had been preferred over other ones such as VISUAL BASIC and VISUAL C++. Several reasons can be pointed out for this choice, namely: its power component library, intuitive visual programming, different databases support, class diagrams generation, and so on. As a whole the prototype has an intuitive and easy to use interface for all kinds of users. The screenshots presenting on fig.11-18 confirm this statement without any comments.



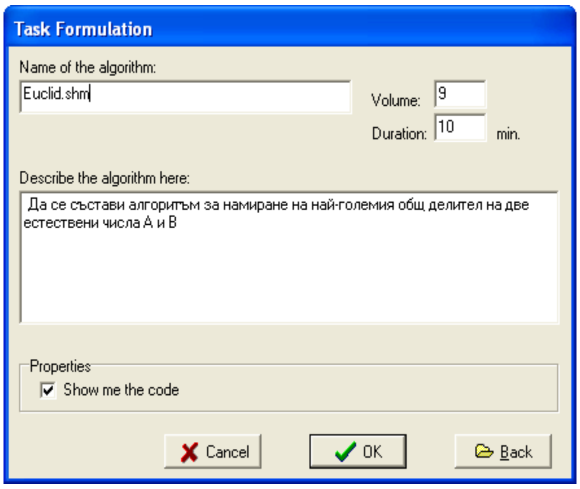
Username: Irina Zheliazkova
Password: ********
Login as: Administrator

Figure 11. Login Window



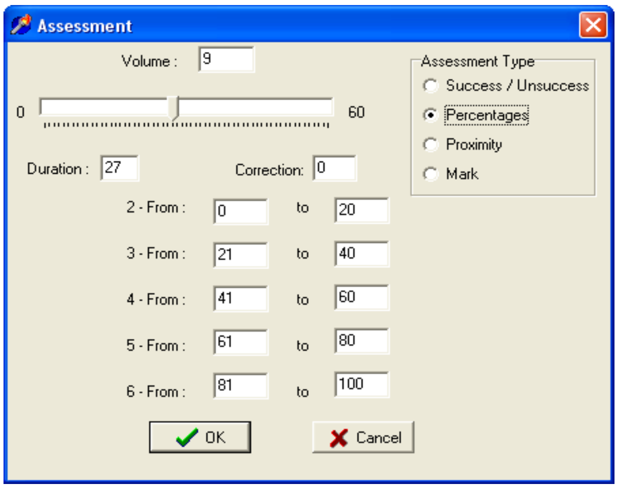
Username	Account
Admin	Admin
asdf	Admin
ssss	StudentLearn
Teach	Teacher
Stud	StudentExam

Figure 12. Administration Window



Name of the algorithm: Euclid.shm
Volume: 9
Duration: 10 min.
Describe the algorithm here:
Да се състави алгоритъм за намиране на най-големия общ делител на две естествени числа А и В
Properties
[✓ Show me the code]

Figure 13. Task Formulation Window



Volume : 9
Duration : 27
Correction: 0

2 - From :	0	to	20
3 - From :	21	to	40
4 - From :	41	to	60
5 - From :	61	to	80
6 - From :	81	to	100

Assessment Type
Success / Unsuccess [✓ Percentages] Proximity Mark

Figure 14. Assessment Window

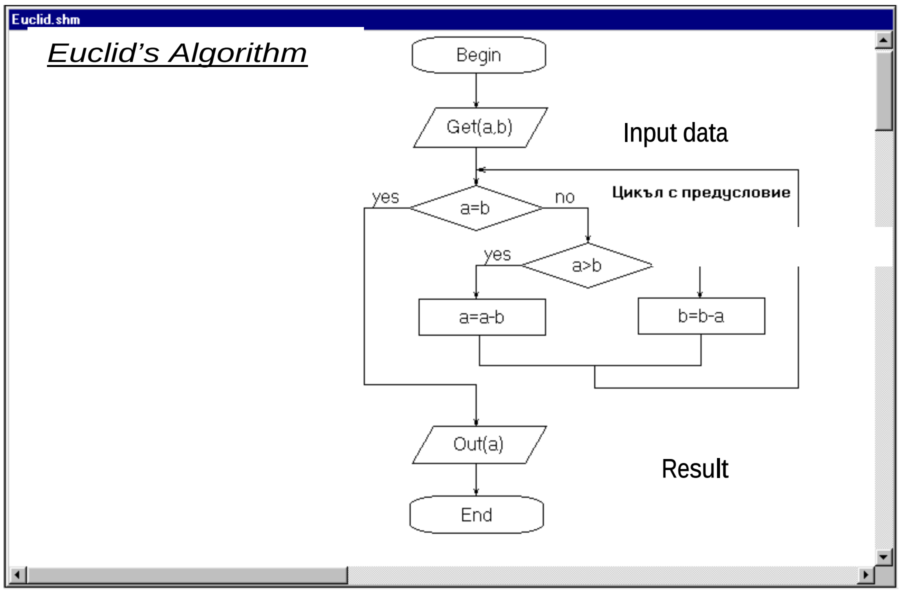
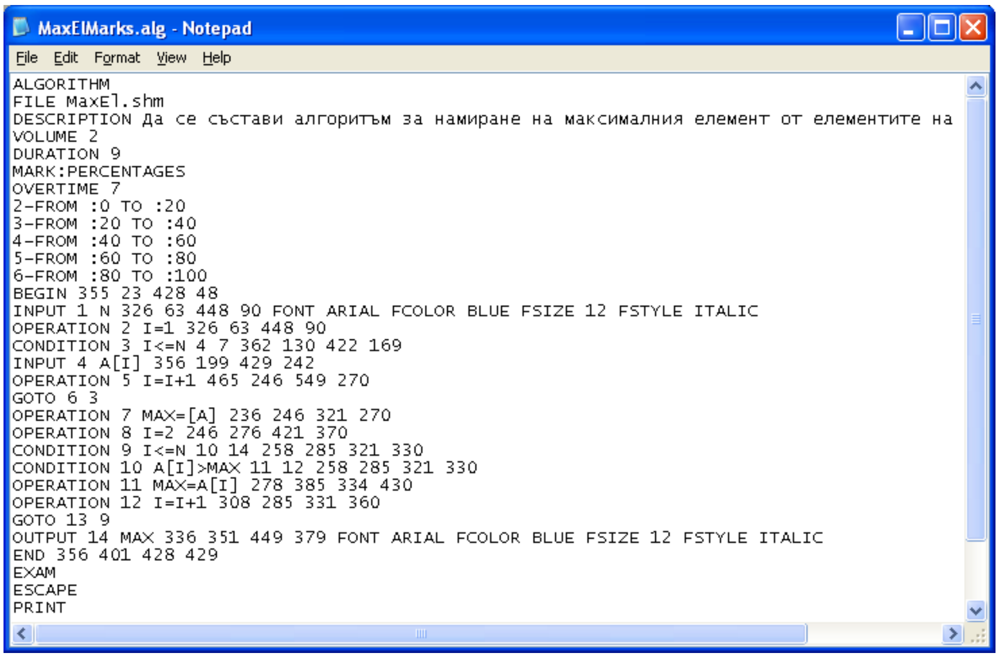


Figure 15. Window with the Euclid's Algorithm Flowchart

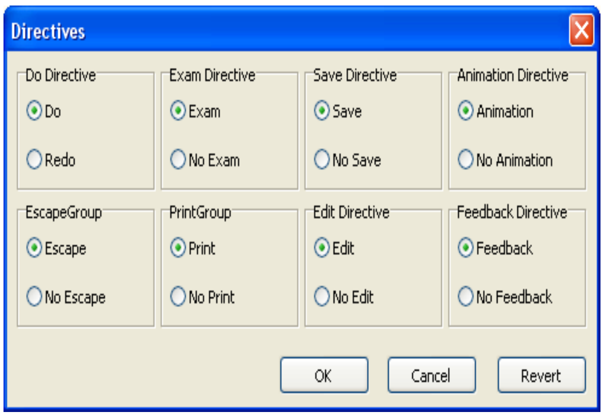


```
ALGORITHM
FILE MaxEl.shm
DESCRIPTION Да се състави алгоритъм за намиране на максималния елемент от елементите на
VOLUME 2
DURATION 9
MARK:PERCENTAGES
OVERTIME 7
2-FROM :0 TO :20
3-FROM :20 TO :40
4-FROM :40 TO :60
5-FROM :60 TO :80
6-FROM :80 TO :100
BEGIN 355 23 428 48
INPUT 1 N 326 63 448 90 FONT ARIAL FCOLOR BLUE FSIZE 12 FSTYLE ITALIC
OPERATION 2 I=1 326 63 448 90
CONDITION 3 I<=N 4 7 362 130 422 169
INPUT 4 A[I] 356 199 429 242
OPERATION 5 I=I+1 465 246 549 270
GOTO 6 3
OPERATION 7 MAX=[A] 236 246 321 270
OPERATION 8 I=2 246 276 421 370
CONDITION 9 I<=N 10 14 258 285 321 330
CONDITION 10 A[I]>MAX 11 12 258 285 321 330
OPERATION 11 MAX=A[I] 278 385 334 430
OPERATION 12 I=I+1 308 285 331 360
GOTO 13 9
OUTPUT 14 MAX 336 351 449 379 FONT ARIAL FCOLOR BLUE FSIZE 12 FSTYLE ITALIC
END 356 401 428 429
EXAM
ESCAPE
PRINT
```

Figure 16. Alg file opened in Notepad

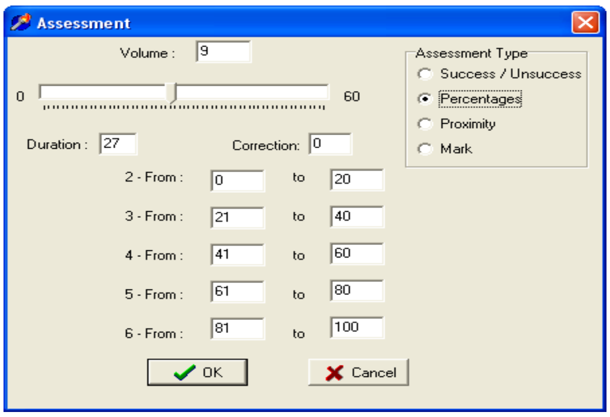
Bellow two special futures of the environment ensuring its intelligence and adaptation to the A and T are commented. In the process of a task created by the A two files with the same name but different extensions respectively *.alg and *.jpg are generated. The first file can be seen as a subprogram in a specialized script language for algorithm knowledge description called *AlgolScript*. Its syntax and semantics can be found in our earlier paper [Zheliazkova I. & G. Atanasova, 2004]. Also in the process of an exercise created by the T an ordinary text file with an extension *.ecs is generated. This file can be seen as a program in a specialized script language for the T's didactic skills description called *ExerciseScript*. Its syntax and semantics have been already defined [Zheliazkova I. et al., 2007]. The syntax and semantic of the ExerciseScript can be found in another paper [Stoyanova E., 2007]. An .alg or .ecs file may be imported in an ordinary text editor like Notepad. Fig. 16 show an .alg file opened in Notepad.

Respectively the generated .jpg file can be open in an ordinary graphical editor like Paint and then imported in the topic BM file. The A/T can edit the resource file and save it as a new file avoiding in such a way the slow process of file generation of individual but equivalent tasks/exercises for a group of learners.



Do Directive: [✓ Do] Redo
Exam Directive: [✓ Exam] No Exam
Save Directive: [✓ Save] No Save
Animation Directive: [✓ Animation] No Animation
EscapeGroup: [✓ Escape] No Escape
PrintGroup: [✓ Print] No Print
Edit Directive: [✓ Edit] No Edit
Feedback Directive: [✓ Feedback] No Feedback

Figure 17. Directives Window



Volume : 9
Duration : 27
Correction: 0
2 - From : 0 to 20
3 - From : 21 to 40
4 - From : 41 to 60
5 - From : 61 to 80
6 - From : 81 to 100
Assessment Type: Success / Unsuccess [✓ Percentages] Proximity Mark

Figure 18 .Assessment Window

Conclusion

The paper presents a UML project of a task-oriented environment for teaching algorithms. It combines three adaptive and intelligent technologies available to the teaching team, namely: interactive problem solving support, example-based problem solving and collaboration support. They are demonstrated on a WINDOWS-based prototype of the environment. The project learner's part will be discussed in a separated paper. We plan to start work on the implementation of the web-based environment that will add the learner's model matching technology specific only for web.

Bibliography

[Brusilovsky, P., 1999] Adaptive and Intelligent Technologies for Web-based Education. In: C. Rollinger and C. Peylo (eds.), Special Issue on Intelligent Systems and Teleteaching, Kinstliche Intelligenz, 4, 19-25.

[Fowler, M., 2004] UML Distilled: A brief guide to the standard object modeling language, 3rd edition, Addison Wesley, 2004.

[Noninska I., 2007] R. Romansky, Data Control in E-Learning Systems, In: Proceedings of the E-learning Conference'07, Instanbul, Bancesehir Publication, 27-28 April, 2007, 52-54.

[Shojkova E. & M. Ivanova, 2007] In: Bulgarian Journal of Automatics and Informatics, 4, 2007, 55-61.

[Stoyanova E., 2007] P.Valkova, I. Zheliazkova, A Domain-Independent Ontology-Based Approach to Representation of Courseware Knowledge, In: Int. J. Information Technologies & Knowledge, 1, 2007, 237-244.

[Zheliazkova I. I., 2004] Atanasova G. E., A Visual Language for Algorithm Knowledge, In: Proceedings of the International Conference on Computer Systems and Technologies (e-Learning), Rousse, Bulgaria, 17-18 June, 2004, IV.24-1- IV.24-6.

[Zheliazkova I., 2004] G. Atanasova, A Technology of Using the TODEA (Task-Oriented Design Environment for Algorithms), In: Newsletter of the Union of Scientists – Rousse, section "Mathematics, Informatics, and Physics", 2004, 83- 89 (in Bulgarian).

Authors' Information

Galina Atanasova – Senior Assistant, Department of Informatics and Information Technologies, University of Rousse, Phone: +359 82 888 470, e-mail: gea@ami.ru.acad.bg

Irina Zheliazkova – Assoc. Prof. Irina Zheliazkova, PhD, Department of Computer Systems and Technologies, University of Rousse, Phone: +359 82 888 744, e-mail: irina@ecs.ru.acad.bg