The framework for easy intelligent tutor programs creation

Anatoly Kulik, Andrey Chukhray, Vitaly Symon

The framework for easy intelligent tutor programs creation

A. Kulik, A. Chukhray, V. Symon

National Aerospace University (Ukraine)

Key words: *framework, intelligent tutor program*

Abstract:

In this paper the framework for intelligent tutor programs creation is described. It separates the logical part of tutor program and technical parts like user interface, database connectivity, etc. The intelligent program is built in specially developed graphical tool by means of performing primitive actions to create task model as a graph with states and conditional transitions. So it may be composed by a teacher with base programming skills.

1. Introduction

Creation of quality intelligent tutor program is a very complicated task. Such program has many criteria of quality: e.g. quality of tutor content, intelligence with deep diagnosing and adaptive remediation of knowledges and skills [1,2], usability, scalable, performance, etc. Therefore, the developers should have experience in complicated software development, knowledges in application domain the program is creating for and methods of training. In this paper, the framework for easy creating intelligent tutor programs by a teacher, who is a not professional programmer, is considered.

2. Problem definition

This problem can be solved by creation of universal framework for intelligent tutor programs. This framework should separate the logical part of tutor program and other, technical parts, like user interface, database connectivity, etc. In this case, teacher, the person who knows the application domain of tutor program, doesn't require some programming skills.

The program should be flexible, scalable and easy for understanding. It consists of some modules, and every module provides a special functionality. For example, user interface is provided by user interface agent and the tutor workflow is supported by control agent. All modules can be plugged or unplugged in every moment and that will not damage the workflow or state of some other module. These modules can be provided by third-party developers as well. All modules communicate through data container called "global facts dictionary". That will provide high flexibility of the program.

The tutor workflow will be designed by teachers, and no programming skills are required. The workflow consists of some little actions and these actions can be described using very simple expressions like "a=2" or "d=sqrt (b*b-4*a*c)". Teacher can bind these actions using some graphical interface without using some special programming language expressions and operators like "if", "goto", "while" and so on.

The tutor program can appear as desktop application or web-application. Change user interface agent is enough to switch the type of application. Web is the best way to create m-learning solutions.

3. Our solution

Most of requirements concern tutor program architecture. Tutor program is considered to be a sequence of some basic actions. Actions are scripts, written in JScript.NET. Every action has some requirement to run. It is some simple expression that can be “true” or “false”. If result of evaluating the expression is “false” the action will not be performed. Every action changes the state of the program using “global facts dictionary”. Action implements IRunnable interface. It means that action object can be executed. IRunnable interface is built using “composite” design pattern(fig. 1)[3].

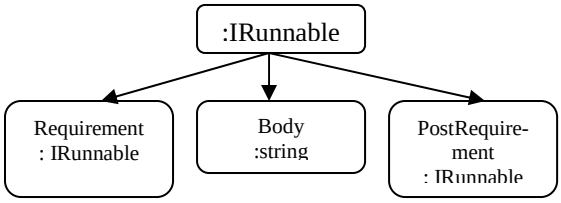

Figure 1. IRunnable interface

Also there are agents in the system. Agent responds for some special functionality. For example user interface, network collaboration can be implemented using agents. Agents are executed after any action (not IRunnable) execution completed. Agent class implements IRunnable too. Agents can be loaded with tutor program automatically or manually by administrator. For example, administrator can enable or disable network collaboration.

Every IRunnable has its unique name and is identified using this name. The recommended name of IRunnable is: *zone.domain.project.method.name* like *edu.khai.sample.QuadraticEquation.InitAction*. These names are required to identify IRunnable when it is stored or restored, some other IRunnable references it. Also these names are used to create namespaces. For example, the variable “d” can be “determinant” in some action and “discriminator” in other. These two actions work in one program. At some moment value of this variable is “discriminator” or “determinant”. So one of the actions gets wrong information. But if these actions use namespaces there is no any conflict: variables names will be *action1.d* and *action2.d*, every action will get access to its local “d”.

Collaboration of agents and actions is provided through “global facts dictionary”. Global facts dictionary is a dictionary that contains objects as values and strings as keys. This dictionary is visible to all modules of program. Agents and actions communicate through this dictionary.

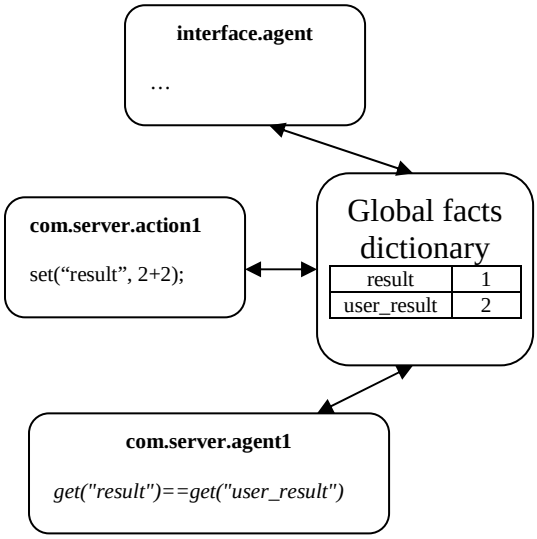

Figure 2. Communication through global facts dictionary.

Functions “get” and “set” are used in scripts to get access to “global facts dictionary”. Example using “get”: *get("result")==get("user_result") // check if the answer is true.*
The symbol @ in front of variable will enable using local namespace for this variable. For example: expression *set(“@myvar”)* will store the variable using name <*action name*>.*myvar* (*action name* is name of action the expression was stored in).
There are some standard namespaces for communication between modules. They are: *interface* – to build interface/get data from it, *student* – to get/set data about student, *network* – for cooperation using network and some other.

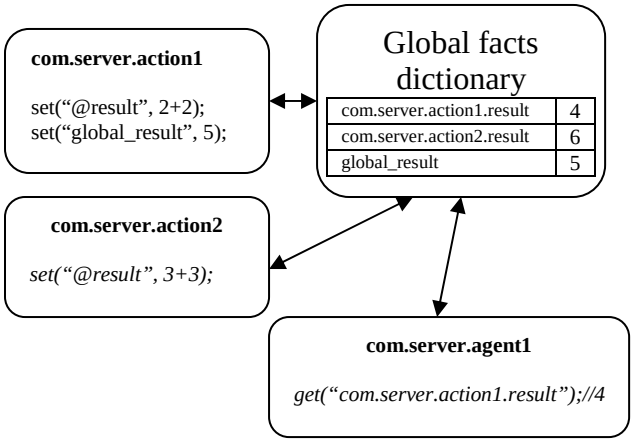

Figure 3. Using namespaces in global facts dictionary.

All objects in tutor program are controlled by control agent. This is the main object of the program. It responds for executing actions, plugging and unplugging agents, global facts dictionary.

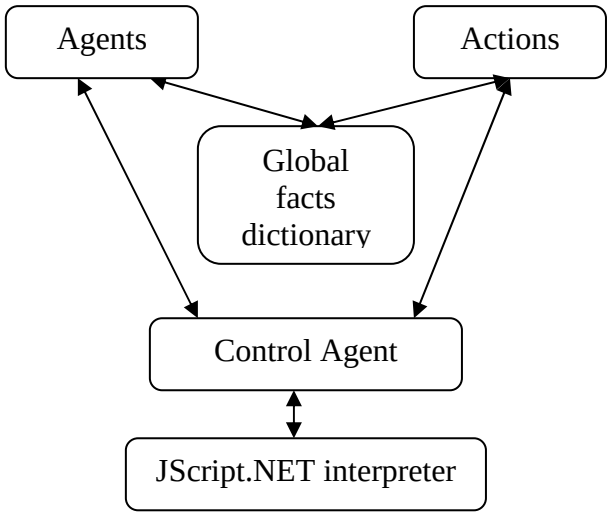

Figure 4. Control agent

Control agent runs the action following to the sequence. After each action is executed, control agent runs all agents, loaded in that moment. All scripts are executed using JScript.NET interpreter.

4. Sample tutor program

Let’s create sample tutor program for quadratic equation.
The program will offer to find roots of the quadratic equation. First of all, it is necessary to create task model of the program (fig.5):

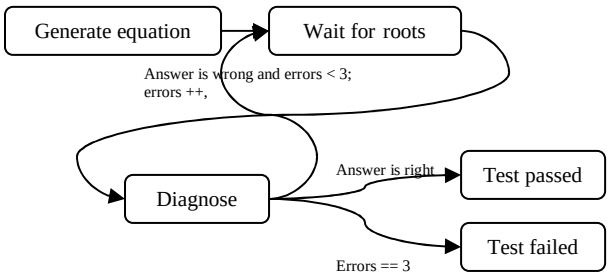

Figure 5. Task model

This diagram should be implemented using special tool that is a part of framework (fig.6). The actions can be added using context menu. Connections can be created clicking right mouse button on two actions. Condition of the connection is set in dialog that appears after double-click on the connection. Example of such condition:
!get("right")&&(get("errors")<3) – this connection is valid only when answer is wrong and number of errors is below 3.
After the sequence is created the actions scripts should be written. To edit script of action press "edit action" button. Example simple window creation using GUI agent:
set("interface.invoke", true) – sets flag for GUI agent;
set("interface.form.my", true) – creates form with the name "my";
set("interface.form.my.text", "sample form") – set caption;
…
set("interface.textbox.nroots", true) – create textbox and show it in the window;
set("interface.textbox.nroots.text", "0");
set("interface.textbox.nroots.x", 100);
set("interface.textbox.nroots.y", 50);
set("interface.textbox.nroots.parent", "my");

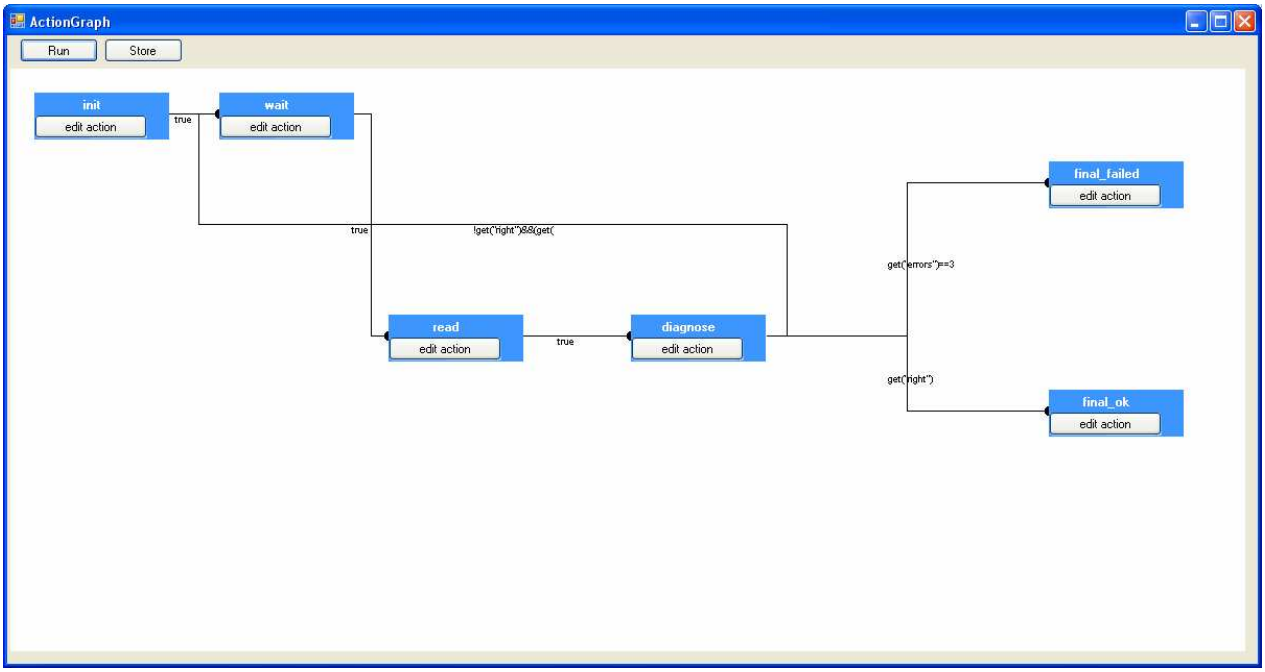

Figure.6. Action sequence designer

Created window is presented on fig.7.

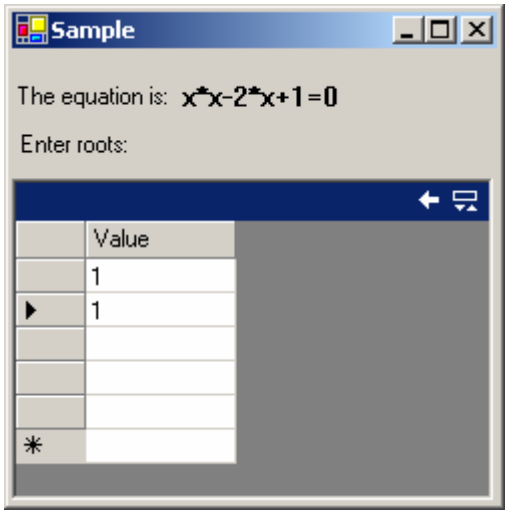

Figure.7. Sample form of intelligent tutor program

After all scripts are ready press “Run” button on form, presented on fig.6. Enter answers into new dialog that will appear (fig. 7) and then close it. If your answer is right program finish its work, otherwise give you one more try. After three tries the test ends with negative result.

5. The Conclusion

The main contribution of the paper is the idea and the features of the framework for intelligent tutor programs easy creation by a teacher who is not a qualified programmer. The architecture of such framework is considered.
Now we are implementing different intelligent tutor programs for mathematic tasks solving training in such a way.

References:

[1] Kulik, A.; Chukhray, A.; Chukhray, M.: Diagnostic models of intelligent tutor system for teaching skills to solve algebraic equations // In Proceedings of the Interactive Computer Aided Learning Conference, Villach, Austria, Sept. 26-28 2006, International Journal of Emerging Technologies in Learning.– 2007. –Vol. 2. – № 1, http://www.i-jet.org.

[2] Kulik, A.; Chukhray, A.: Intelligent tutor system for teaching skills to solve algebraic equation // In Proceedings of the East-West Fuzzy Colloquium, Germany, Zittau, IPM, 2006, P. 120-128.

[3] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series, 1995. – 395 p.

Author(s):

Anatoly Kulik, Doctor of Technical Science, Professor, Laureate of Ukraine State Prize, head of Control Systems department, dean of Control Systems faculty in National Aerospace University, Kharkov, Ukraine (e-mail: kulik@d3.khai.edu).
Andrey Chukhray, Candidate of Technical Science, deputy head of Control Systems department, deputy dean of Control Systems faculty in National Aerospace University, Kharkov, Ukraine (e-mail: chukhray@d3.khai.edu).
Vitaly Symon, student of National Aerospace University, Kharkov, Ukraine (e-mail: symon@gmail.com).