Applying Intelligent Agent Technology to Create Instruction Programs

Dr. Vladimir V. Tarasov

Department of Mathematics and Data Analysis,
Karelian Research Centre of the Russian Academy of Sciences

Pushkinskaya St., 11, Petrozavodsk, Republic of Karelia, 185610, Russia

E-mail: tarasov@post.krc.karelia.ru

Abstract

This article concerns the problem of creating intellectuallized instruction programs. We suggest applying intelligent agent and multi-agent technologies for the creation of such programs. Two examples, COACH and IDLE, are given of existing systems of this kind. Then we describe the LimTUTOR system which is developed to support the process of teaching students to solve tasks of searching for function limits. Finally we extend LimTUTOR to a multi-agent system to cover different aspects of teaching a discipline.

1 Introduction

Nowadays one can observe a considerable increase of employing computer aids supporting educational process. Among them there are instruction computer programs that allow to improve the education efficiency by providing each student with a "personal digital teacher" [1, 3, 5] (like a "personal digital assistant" [4]). In this case the success of education depends on the knowledge the computer program possesses and on the behaviour of the "personal digital teacher". The deeper the knowledge possessed by

the program and the better it simulates the human–teacher behaviour, the more successful education process goes on.

Recently researchers have tried to exploit knowledge–based system technology to develop such programs. This approach gave the developers a good tool for representing comparatively large volumes of domain and teaching knowledge, for taking into account the user's expertise level, but left aside the behavioural aspect of the teacher's work. Besides, these systems are usually locally available and have too much control over the student's actions. Now the focus has shifted to intelligent agent and multi–agent system technologies. These ones allow not only for more precise simulation of the teacher's work, but also for design of distance learning systems working in distributed environments, for instance, in the Internet.

A number of works is being carried out along this direction presently, for example, the WebDidact system developed at the Polytechnic Institute of Losanne (Switzerland) [3], the system for learning digital electronics designed at the University of Genoa (Italy) [3], the system COACH for teaching students to program in Lisp created at IBM Almaden Research Centre and Stanford University [5], the adaptive system IDLE is being developed at the Institute for High Performance Computing and Databases (SPb, Russia) [3].

To determine what advantages can be provided by applying intelligent agent technology to computer–aided instruction, we should have a general look at the instruction process. Usually, three basic tasks emerge during the process of teaching some discipline:

1. Provide the student with theoretical knowledge.
2. Teach the student to apply theoretical knowledge to solving practical tasks.
3. Check up the theoretical and practical knowledge acquired by the student.

Consequently, the second task may be divided into two phases: demonstration of the process of solving sample tasks to the student; guidance of the student through the process of solving tasks independently.

Let us firstly consider the second problem in more detail. In order to implement both phases of teaching students how to solve practical problems efficiently, it is necessary to create a program system having the

following characteristics. Firstly, the system should posses adequate domain knowledge to demonstrate how to solve problems and to check up the student's way of solving tasks. Secondly, the system should have knowledge on the methodology of teaching how to solve problems. Thirdly, and this is the most important characteristic, the system should be able to watch the student's actions implicitly, to evaluate the method, chosen by the trainee, of solving a task, and, when needed, to direct the student's method of solving a task towards the solution.

One of the efficient techniques of building such a program system is intelligent agent technology [8]. Employing this technology allows not only for providing a program with required knowledge, but also for taking into account the behavioural aspect of the work of a "personal digital teacher". This is corroborated by the following agent properties: autonomy, allowing for guidance of a student's work without explicit interventions, social ability, permitting active interaction with the student and other applications, reactivity, allowing for respond in a timely fashion to student actions, pro–activeness, making it possible for the system to take the initiative during the instructing process [2, 8].

But to encompass the overall process of teaching, it is expedient to use multi–agent system approach. In this case every task emerging during the computer–aided instruction may be accomplished by a separate agent: one being responsible for the theoretical part, the other one teaching to solve practical tasks, the third one checking up the knowledge acquired by the students, and the last one supervising the process of teaching. And a special agent could help the tutor to gain the whole picture of how the instruction process is going on. So the set of the agents would act as a multi–agent system forming "a complex digital teacher". Such agent specialization can improve the system's performance and flexibility and this multi–agent system might be also used for distance learning in the distributed computational settings.

2 COACH and IDLE—Examples of Employing Intelligent Agent and Multi–Agent System Technologies

Now we shall dwell upon two examples of employing intelligent agent and multi–agent system technologies. The first system is COACH which stands for COgnitive Adaptive Computer Help [5]. This system was created at IBM Almaden Research Centre and Stanford University for teaching students to program in Lisp. COACH is an advisory system that does not interfere with the user's actions but comments them implicitly to help the user along. COACH watches the user's actions to build *Adaptive User Model* which chooses the appropriate advice, that is COACH records user experience to create personalized user help. COACH might choose to use description, example, syntax, topic, style and level of help according to user–demonstrated experience and proficiency.

Figure 1 depicts the system's architecture [5]. *Domain Knowledge* is represented in the help system subject frames, adaptive frames and the parser grammar. *Subject Frames* contain knowledge about the skill domain a user is trying to learn (e.g., Lisp). User background is stored in *Adaptive Frames*. *Coaching Knowledge* is presented in rules that create and control the adaptive frames and the help presentation blackboard. *Update Rules* control the recording of user experience for the adaptive user model. *Consistency Rules* contain knowledge about how to build the adaptive user model. These two rule sets work to update the adaptive user model. *Presentation Rules* determine the help that will be provided to the user. The *Reasoning System* is made up of a forward–chaining production system that interprets rules and a *Blackboard Mechanism* that resolves presentation goal conflicts. Together they operate on the adaptive user model by referring to domain knowledge and using coaching knowledge to make decisions.

The second system is IDLE which stands for Intelligent Distance Learning Environment [3]. This system is being developed at the Institute for High Performance Computing and Databases (SPb, Russia). IDLE is a multi–agent adaptive system for distance learning. The system's adaptivity, i.e. its ability to adjust itself to personal characteristics and preferences of its user, is based on *User Model*, which is built accord-

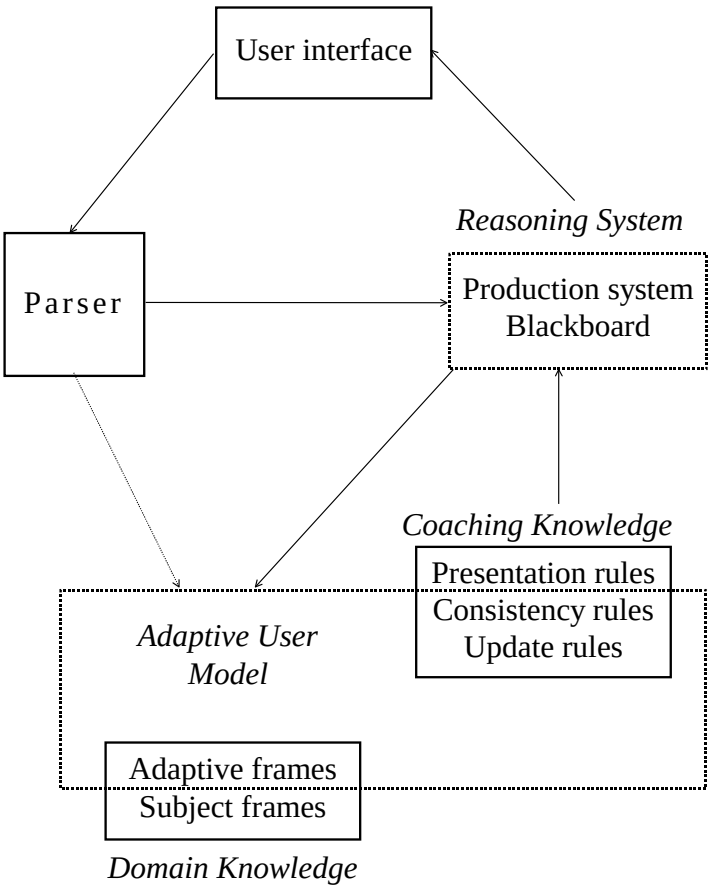

Figure 1. COACH Architecture

ing to the results of prior testing and is adjusted dynamically during the user's work. A prototype of this system has been developed and the technology of program testing was taken as a sample domain to be tutored by IDLE.

Figure 2 depicts the system's architecture [3]. Among the agents comprising IDLE are the following. *Expert Tutor* controls the learning process, applies different educational strategies according to the cognitive and

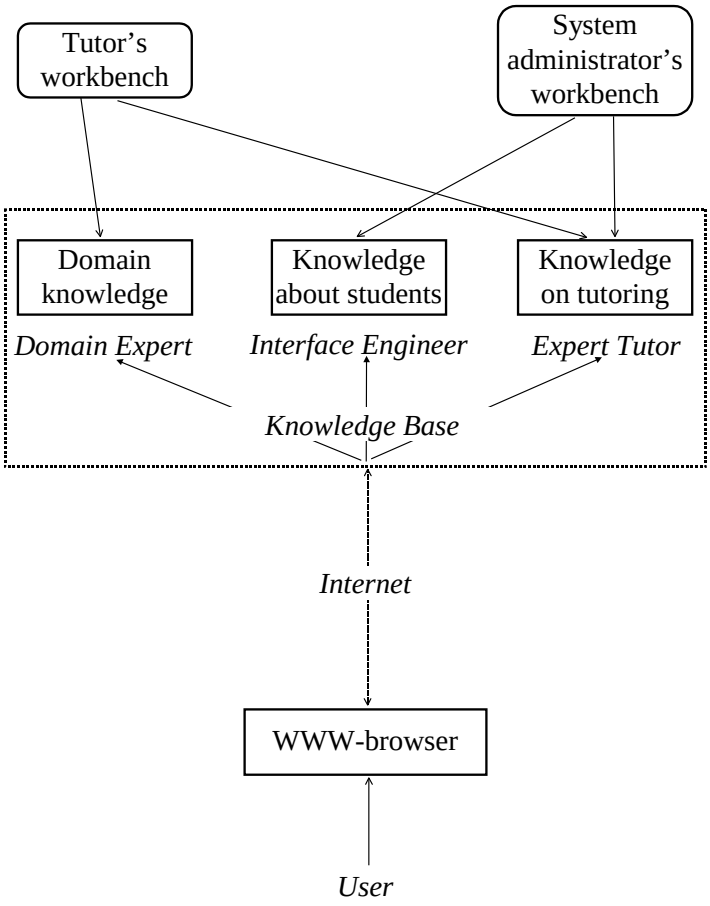

Figure 2. IDLE Architecture

characterological features of the student together with his/her instruction progress. *Interface Engineer* maintains User Model for the current student and provides it to the human users and other agents. *Domain Expert* accumulates and provides knowledge on subject domain, maintains testing exercises, provides these data to its users and other agents, analyzes the student "feedback" in domain terms.

3 LimTUTOR—an Intelligent Agent Teaching How to Search for Function Limits

Now we shall describe the LimTUTOR system which is being created at the Department of Mathematics and Data Analysis of the Karelian Research Centre of the Russian Academy of Sciences [7]. LimTUTOR is intended to support the process of teaching students to solve tasks of searching for function limits. The entire instruction process encompasses two phases. At the first phase the system shows a student how to solve tasks of searching for function limits. To do this, the user formulates the task of searching for a function limit, and during solving the task the system exhibits the user not only the found solution, but also the entire way of its (system's) reasoning including detailed explanations for why that or another action was undertaken. This is done in a step–wise manner for the trainee could smoothly follow the system's way of reasoning.

Figure 3 depicts the scheme of interaction between LimTUTOR, a specialized mathematical editor, and a user [7]. It should be noted that we can use our own custom mathematical editor or exploit available editors such as Mathematica, Mapple V and so on. LimTUTOR is designed to be flexible and useful for the existing software.

At the second phase, being the most interesting with regard to the system's operation, the student is given opportunity to solve tasks autonomously, LimTUTOR watching the student's way of doing this and, if required, correcting and adjusting him. The user starts his work by entering an initial expression into the specialized mathematical editor. As necessary transformations are carried out, new expressions are entered into the editor. All the time LimTUTOR remains active, observing the user's actions. Upon entering a new expression, LimTUTOR reads it in

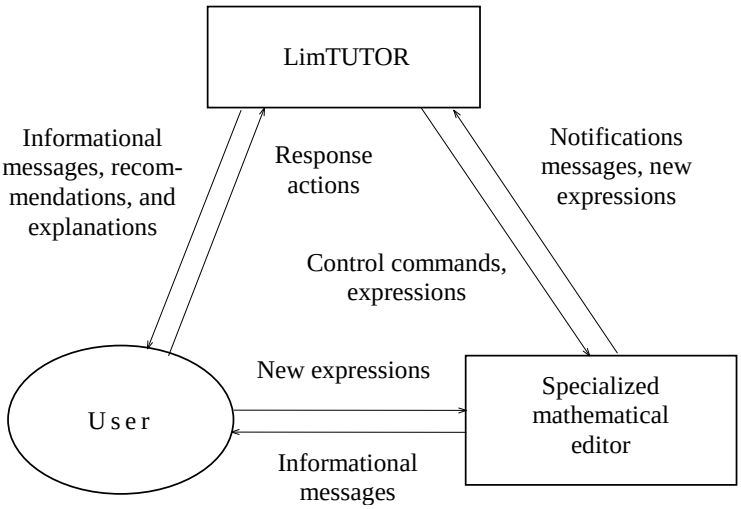

Figure 3. Interaction between LimTUTOR, the specialized mathematical editor, and the user during the instruction process

from the editor for analysis and pauses the editor's operation. If the step done by the user drives us to the solution, the editor's operation is resumed. Otherwise, LimTUTOR informs the user his step is incorrect and hints him some feasible ways of continuing the search for a solution.

It is worth noting that LimTUTOR resembles the COACH system [5], and yet there is a considerable difference. While COACH focuses on constructing an adaptive user model to select the most appropriate advice according to the user's proficiency, LimTUTOR concentrates on the process of teaching how to solve hard practical problems. To accomplish this, sophisticated reasoning techniques is used, which allow for making prognosis on further event development and for composing pertinent recommendation.

4 Architecture of the LimTUTOR System

In general terms, LimTUTOR consists of six parts: user and editor interface components, a control component, a knowledge base, and two inference engines. This structure of the LimTUTOR system is drawn in Figure 4 [7].

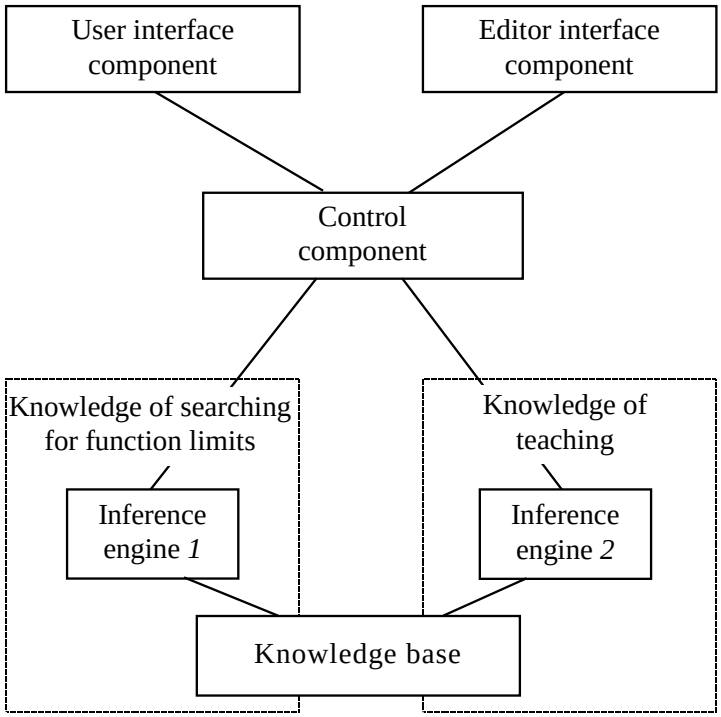

Figure 4. Architecture of the LimTUTOR system

The *Control Component* is responsible for the system's autonomous functioning, for switching it from the demonstration mode of solving tasks into the mode of observation and correction of this process and vice versa, and for coordinating the operation of the separate subsystems. The *User Interface Component* is intended to output explanations, recommendations, and various messages to the user. The interaction with the special-

ized mathematical editor is arranged with the help of the *Editor Interface Component*. This interaction includes putting expressions into the editor, getting new expressions from the editor, sending commands to control the operation of the editor, and receiving its notification messages.

The *Knowledge Base* contains basic inference rules which formalize the knowledge of searching for function limits and of teaching. *Inference Engine 1* is designed to solve particular tasks of searching for function limits and is used throughout the both instruction phases. *Inference Engine 2* directs the instruction process at the second phase and activates Inference engine *1* as the need arises.

5 Teaching Knowledge Representation

For representation of the teaching knowledge in LimTUTOR we utilized the inference search control language (ICL) allowing for description of solution search process as a sequence of inference metarules which formalize the methods of applying basic domain knowledge to searching for solutions [6].

The first inference engine is designed to solve the task of searching for function limits. Corresponding metarules formalize different ways of using basic rules of transformation of the expressions falling under the limit sign, this being necessary to search for a solution. As an example we shall cite the following basic rule applying a formal substitution and metarule. The natural language description comes first, then the formal description goes.

BR43. Square difference. $a^2 - b^2 = (a - b)(a + b)$.

```
TSUBST(POL_SqrDif; subtr(pow(s,2),pow(t,2));
       mult(subtr(s,t),add(s,t)); PrepareTerm)
```

MR2. Single Inference. If the current inference consists of one formula and the expression falling under the limit sign and included in this formula of the current inference is product, difference or sum of functions, then mark the formula of the current inference with the label "single inference" and apply the basic rules about quotient, product, difference and sum of functions.

```
SingleInf :
   If((SingleInf(cur_inf)&ArithmOper(first_form)))
   Apply(label(single_inf),CL_FuncQuot,CL_FuncProd,
         CL_FuncDif,CL_FuncSum);
```

The second inference engine is employed to watch a student solving tasks. Upon receiving the subsequent message of entering an expression from the editor, the performed transformation is examined for validity. Then the second inference engine invokes the first one to construct a solution starting from the point reached by a student. If the construction is successful, then the accomplished step is stored. If either of the conditions is not satisfied, the second inference engine backtracks to the previous expression and invokes the first one to find, starting from this point, the next step leading to a solution. After doing this, the user is given the message about what next step might be undertaken and why. Figure 5 shows the interaction between the inference engines during this stage.

6 Multi–Agent System for Teaching Different Aspects of a Discipline

Now that we described LimTUTOR we can see that intelligent agent technology may be applied to teaching how to solve practical tasks, but we left aside other educational aspects which appear during the instruction process. In order to cover different aspects of teaching a discipline, it is reasonable to extend our instruction agent into a set of agents, each of them being responsible for a particular educational task.

Figure 6 shows a feasible architecture of such a system. The systems consists of five agents. *Theory Agent* provides the user with theoretical knowledge on the discipline. *Practice Agent* demonstrates and teaches the user how to solve practical tasks. *Check–up Agent* accomplishes the check–up of the level of the student's assimilation of the knowledge. *Supervising Agent* controls the overall process of instruction and *Tutor's Assistant Agent* assists the human tutor to collect information about how the instruction process is going on and data concerning particular students.

The agents might use some information about the user's expertise and record his activities. To use this multi–agent system in distributed

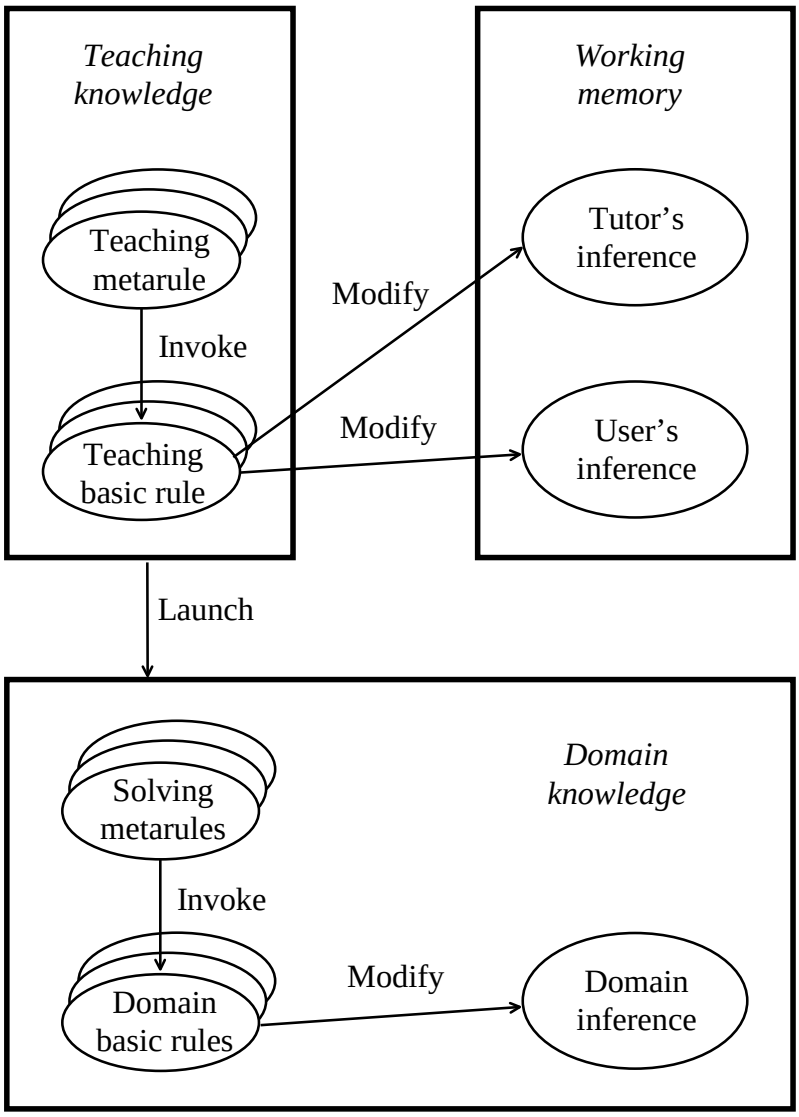

Figure 5. Teaching knowledge representation

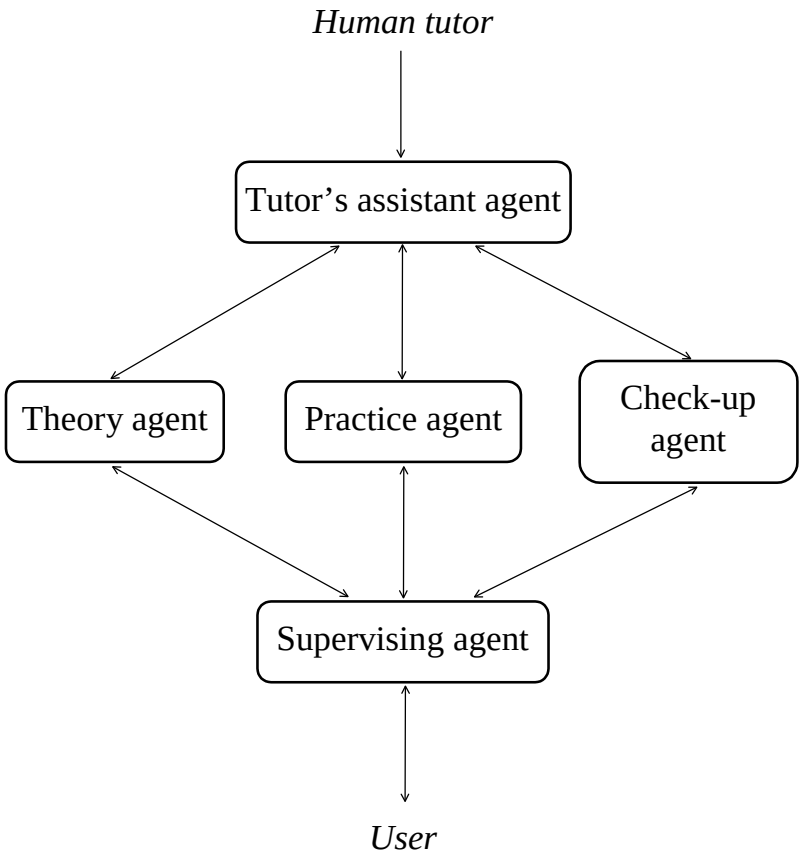

Figure 6. Multi–agent system for teaching different aspects of a discipline

computational environments, like the Internet, we may employ Java for implementing some parts of the system and KQML for exchanging messages between the agents. In this case this multi–agent system could be used for distance learning.

7 Conclusion

This paper has described the application of intelligent agent technology to the design and development of instruction programs. We considered two existing examples of such programs and the developed system LimTUTOR, which is designed to support the process of teaching students to practically solve tasks of searching for function limits. LimTUTOR embodies the both instruction phases: demonstration of how to solve problems and observation of the student solving problems, the needed corrections of the trainee's actions being done. This system is under development now and has been partially tested within the special course for fifth year students of mathematics and computer science at Karelian State Pedagogical University. A graduate student of the computer science department of Petrozavodsk State University has taken part in the project.

Utilization of intelligent agent technology has permitted us to simulate the "student–teacher" relationship and to create "personal digital teacher" that imperceptibly watches the student working and interferes with this process only if need be. This approach may be applied to teaching how to solve problems related to other mathematical areas, for instance, how to solve tasks of integration.

It would also be expedient to organize interaction between LimTUTOR and another agents capable of providing a trainee with theoretical knowledge and of checking up the knowledge obtained by the student. So, applying multi–agent system technology will enable us to cover the overall process of teaching a subject, every instruction task being accomplished by a separate agent. This multi–agent system can function in distributed environments like the Internet and, therefore, may be used for distance learning.

References

[1] *Berestova V.I. et al.* Using expert system technology for creation of intelligent teaching programs. In Conference "East–West" on New Informational Technologies in Education (abstracts of reports). Moscow, 1992 (in Russian).

[2] *Braspenning P.J.* Plant–like, animal–like and humanoid agents and corresponding multi–agent systems. In Proc. of the Int. Workshop "Distributed Artificial Intelligence and Multi–Agent Systems" (DAIMAS'97). St.Petersburg, 1997, pp.64–77.

[3] *Gavrilova T.A., Voinov A.V., Danil'chenko I.I.* Multi–agency—based project on a distance learning system for program testing. In Proc. of the Int. Workshop "Distributed Artificial Intelligence and Multi–Agent Systems" (DAIMAS'97). St.Petersburg, 1997, pp. 125–136.

[4] *Maes P.* Agents that reduce work and information overload. Communications of the ACM, July 1994, vol. 37:7, pp. 30–40.

[5] *Selker T.* COACH: A teaching agent that learns. Communications of the ACM, July 1994, vol. 37:7, pp. 92–99.

[6] *Tarasov V. V.* Using explicit representation of inference search metarules for inference search control in expert systems. In Int. Conference on Informatics and Control (ICI&C'97), vol. 2, St.Petersburg, 1997, pp. 421–428.

[7] *Tarasov V. V.* Using intelligent agent technology for the development of a system teaching how to search for function limits. In Proc. of the Third Inter–Karelian Conference on Teaching Mathematics and Physics in Secondary and Higher Education, Petrozavodsk, 1998, pp. 248–253.

[8] *Wooldridge M., Jennings N.* Intelligent agent: theory and practice. The Knowledge Engineering Review, vol. 10:2, 1995, pp. 115–152.