I. Zaretska
zaretskaya@karazin.ua
A. Radchenko
wsooh16@gmail.com
A. Minayev
alexlm555555@gmail.com

Algorithms Constructor: a Tool to Learn and Teach Algorithms

The paper presents an environment for creating, editing, tracing and running, saving and loading algorithms in a flowchart form. It can be used in any educational process whenever the basics of algorithms are to be learned. The requirements model, design of the system and its implementation are presented. Two modules are presented: for a trainer to fill in the system with the problems on creating algorithms and their right solutions in the form of flowchart, and for a trainee to solve these problems by creating and tracing algorithms and gaining skills in algorithms construction.

Key words: software system, object-oriented design, algorithm, flowchart.

І. Зарецька
zaretskaya@karazin.ua
А. Радченко
wsooh16@gmail.com
А.Мінаєв
alexlm555555@gmail.com

Конструктор алгоритмів: інструмент вивчення та навчання алгоритмам

У цій роботі подано середовище для розробки, редагування, трасування й виконання, збереження та завантаження алгоритмів у формі блок-схем. Воно може бути застосоване у будь-якому навчальному процесі під час вивчення основ алгоритмів. Наведені модель вимог, дизайн системи, а також його реалізація. Подано два модулі: модуль викладача для наповнення системи завданнями щодо створення алгоритмів та їх рішеннями у формі блок-схем; модуль учня для вирішення цих проблем під час розробки та трасування алгоритмів і отримання навичок у конструюванні алгоритмів.

Ключові слова: програмна система, об'єктно-орієнтований дизайн, алгоритм, блок-схема.

Introduction

In IT education wherever it takes place – at the elementary or the secondary school, university, any other secondary or higher educational institution or even just IT school – it is of utmost importance to gain fundamental knowledge and skills which form the base for further education. As our experience on teaching IT for more than 40 years shows it is absolutely justified to teach basics of algorithms, their structures and control flows as well as methods of creating before teaching programming and programming languages. When a student can develop an algorithm to solve the problem it is much easier for him or her to write a program using the syntax of some programming language. Unfortunately, many trainers begin teaching programming skipping this very important step which leads to partial understanding instead of general approach. To make the process of teaching and learning basics of algorithms more IT-like we developed a software system which in fact is an IDE (Integrated Development Environment) for

working with algorithms – creating, saving in files, loading from files, editing, tracing and running. The system has two modules – for trainees and for trainers.

The trainer's module allows user to formulate a problem and add it to an existing category or to a new one (all the problems are supposed to be divided into categories). Then a trainee creates a flowchart of an algorithm or a set of flowcharts (if there are several solutions) being the right solution to this problem by using flowchart blocks and filling them with commands. There is an option to add some extra blocks with commands aside from the right flowchart's blocks for a trainee to widen the variety of choices. The problem and its flowchart solutions should be saved.

The trainee module allows user to choose the problem from the list of problems of the certain category and to create the flowchart of an algorithm using blocks with commands created by a trainer. He or she can then check the solution, edit it and check again, run or trace the algorithm under different input data and see control paths and outputs, save it in any folder for further usage.

The proposed system was developed by request of the Ukrainian publishing agency "Ranok" in frame of the international project on joint school education as an electronic support for several textbooks on informatics for schools published by this agency. All these textbooks are recommended by the Ukrainian Ministry of Education and are used together with their electronic support systems in most Ukrainian schools. There are several types of electronic support materials developed together with these textbooks. In fact the authors of this paper are among the authors of all the textbooks and all the electronic materials. Part of them is developed as the interactive games strictly bounded to the materials and the design of the textbooks and covering the whole book including basics of algorithms. Opposite to those electronic materials the system presented in this paper is universal and can be used separately not only at schools but in any educational institution teaching IT. The first version of the system with Ukrainian interface only was disseminated two years ago via the site of the publishing agency "Ranok" among the Ukrainian schools. The feedback was analyzed and corrections and improvements implemented. The paper presents the new version of the system.

The system goes with the predefined initial set of about 50 problems and their solutions developed by the authors. All problems are divided into categories (linear algorithms, forks, iterations, and combined algorithms). New categories and problems can be easily added using trainer's module as well as any of the problems of the initial set can be deleted or edited by a trainer.

The system can be used for individual training as well as in the class work. In second case a teacher can assign different problems to different students and check their solutions saved in agreed network folder later, which considerably decreases teachers' loads.

1. Related Work

In the last decade of previous century the theory of R-charts was proposed by Russian professor I.V. Velbitskiy. He developed a new technology of visual programming by R-charts which allowed drawing programs instead of writing them. He implemented the universal graphical environment to create programs in any existing programming language [1–4]. His research was continued and enhanced with UML notation [5]. The R-charts were used mostly on industrial level for teaching the software technology in the process of software systems development. The theory and technology are quite complicated to be used at the introductory stage of learning algorithms.

Some research in the area were conducted in Kherson state university by professors A. Spivakovskii, M. Lvov and their followers [6–9]. They created a special instrumental language that allowed interpreting an algorithm. It was created especially for teaching students basics of algorithms but it had no graphical tool for constructing flowcharts.

A lot of European and American software companies worked on the flowcharts construction tools mostly for modelling business processes, work and data flows, activities, etc. In fact nowadays there are a lot of CASE-tools for flowcharts construction: any office suite includes something like Microsoft Visio. The truth is that they may be quite helpful for modelling and for experienced users but much less useful in the educational process for learning algorithms. They have quite complicated interfaces for the beginners. They do not allow user to check or to trace or to run an algorithm, explore different control paths under different input data, examine output data and make conclusions, which the process of learning requires. But they do have a lot of other options which are really excessive in the process of learning basics of algorithms. Not to mention the absence of incorporated problems to solve, which gives a teacher the freedom for creative work with students of different levels without increasing teacher's load.

There are also several web-applications for flowcharts construction like draw.io (https://www.draw.io/) or lucidchart.com (https://www.lucidchart.com/) but they have the same disadvantages as the CASE-tools mentioned above.

The only similar software system found by authors is Algorithm Flowchart Editor (AFCE) (https://github.com/viktor-zin/afce/tree/gh-pages/download) [10] which can be used in educational process. It offers the environment to construct, edit, save and load flowcharts but one cannot trace or run an algorithm to explore it. Instead it offers some pseudo-programming code implementing the algorithm, which may be useful for learning purposes. No set of problems and no interface to create them are proposed.

The authors do not know so far software systems which satisfy the requirements given in the next section.

2. Requirements for Algorithms Constructor

After multiple discussions with customers and potential users (mostly school teachers) and prototyping the following set of requirements was defined.

The system has two types of users with roles “trainee” and “trainer”.

A trainee uses the system to explore the process of creating flowcharts for the predefined set of problems, tracing and running the obtained algorithms with different data inputs to see the outputs. The right solutions for these problems have already been saved in the system by a trainer. So a trainee can check whether his or her solution is correct and if not, edit the flowchart and explore or check it again. To create or to edit a flowchart a trainee has to insert or delete blocks. To insert a block one has to choose (click) a block from the set of given ones and then choose (click) a place for a block on the flowchart. Deleting a block is a usual procedure of deleting a fragment. UnDo and ReDo commands are also available.

A trainer uses the system to fill it in with problems and their right solutions in the form of flowcharts. Several right solutions can be stored with a problem. A problem can include an image. Problems can be divided into categories if needed. A trainer can add or delete categories. Categories can form a tree with nested categories but no more than 3 levels in depth. The names of categories and problems, their texts, images and the right solutions are completely up to the trainer and completely his or her responsibility. The only validation offered by the system is checking that conditions are in their proper places in forks and iterations and blocks for data input and output (they have the special shape) contain the commands to enter or print variables.

The simplified use case diagrams for these functional requirements are shown in Fig. 1 and Fig. 2.

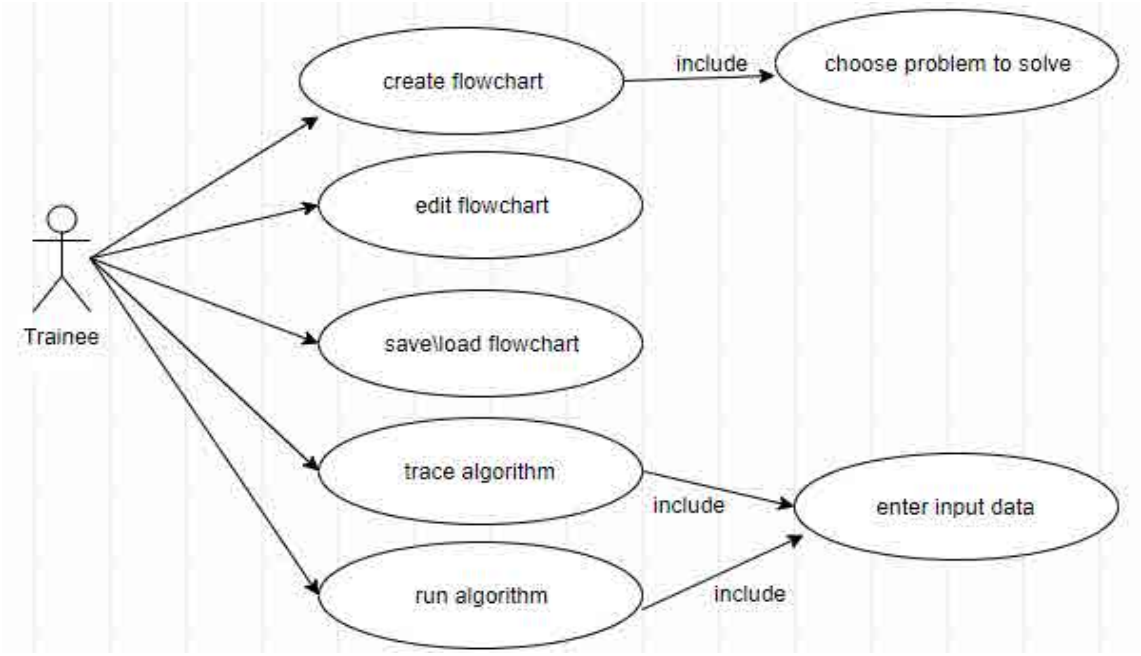

Fig. 1. Use case diagram for a trainee

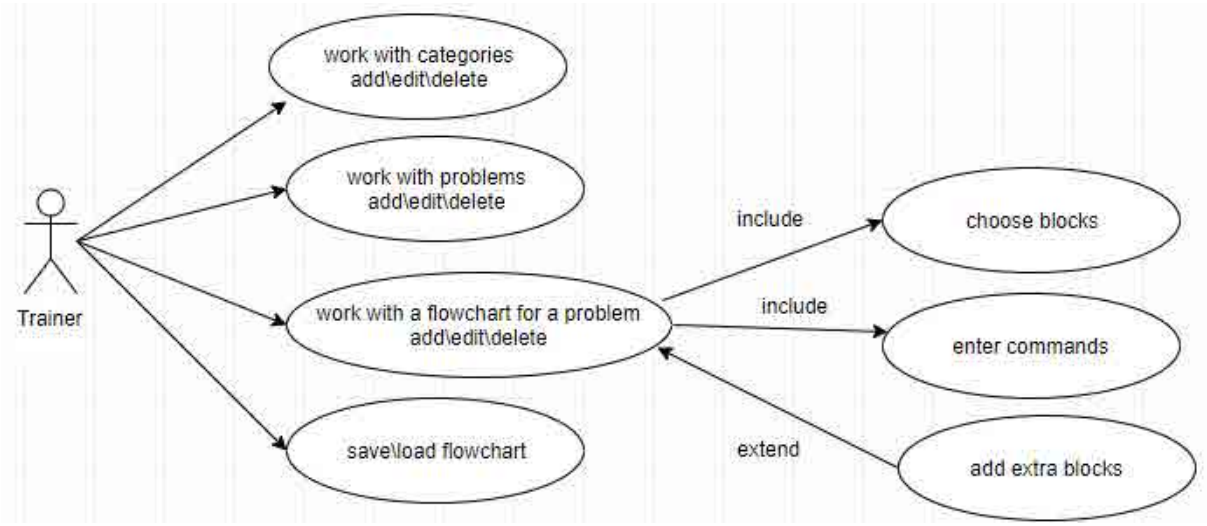

Fig. 2. Use case diagram for a trainer

The nonfunctional requirements are the following.

The system supports three interface languages – English, Ukrainian and Russian.

The prototype of the user interface design looks like shown in Fig. 3.

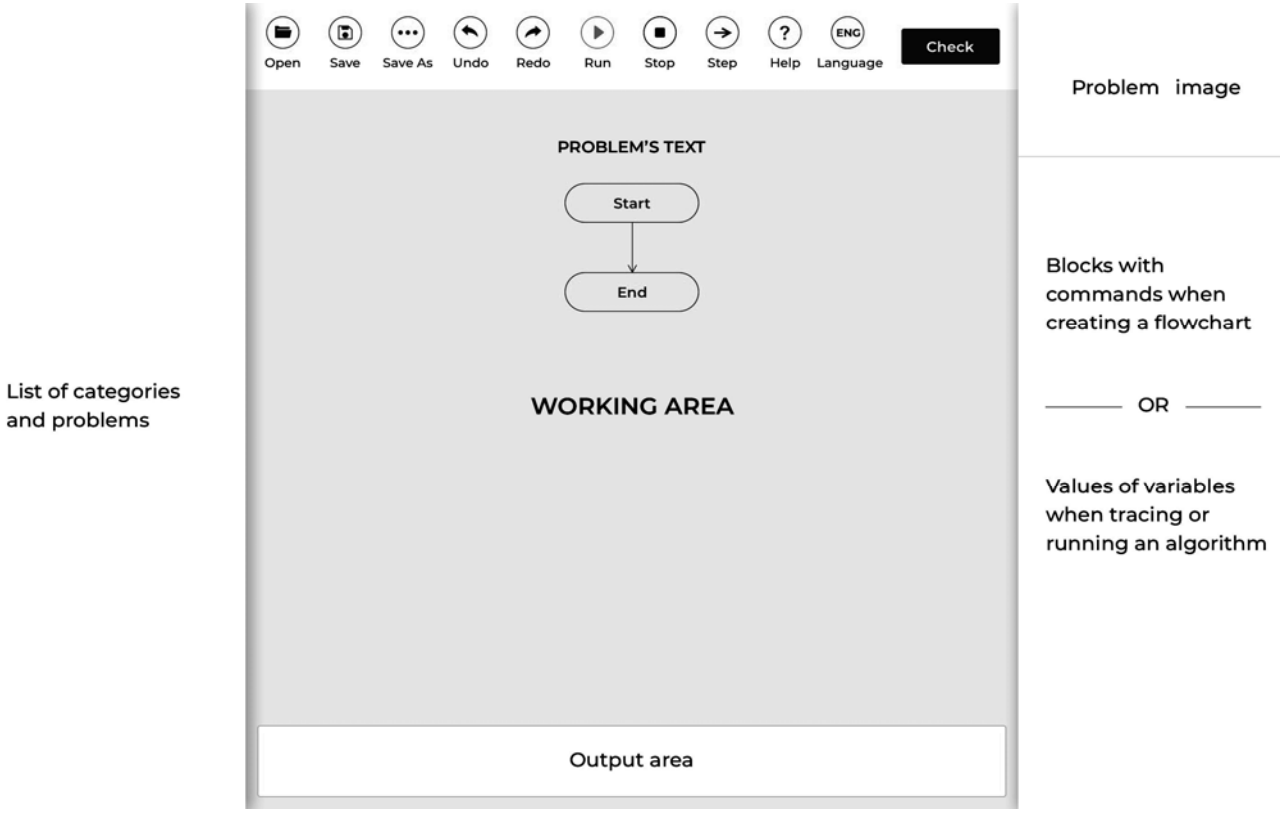

Fig. 3. User interface prototype

The system is a desktop application. The deliverable is an executable file (*.exe).

The presentation and business logic levels are implemented with Python. The data storage (problems, flowcharts, solutions) is implemented as json-format files.

3. Design of Algorithms Constructor

This section covers the most crucial part of the system, which is its design. We used an object-oriented approach and UML diagrams [11] to represent the static structure of the system (class diagrams) and its behavior (sequence diagrams). The main goal of the design was to make it reusable for both trainee and trainer modules and to follow SOLID principles [12, 13]. The whole number of classes exceeds 50, so we divided the class diagram into logically connected parts to present it in the paper. The sequence diagrams show the behavior of the system for each use case, but they are too big to be presented here, so we present here the main concepts of the system, the rendering classes, and the classes responsible for persistency. We also consider the GUI structure and models for json representation of the basic concepts.

3.1. Business logic and GUI structure

This section specifies the structural design of the system as UML class diagrams.

1. The main concept of the system is an Algorithm. Its structure is shown in Fig. 4. It is a container of classes derived from the abstract Operator base class. These derivative classes stand for all basic operators of any algorithm which are inputs and outputs, text commands, assignments, forks and loops of three types (with preconditions, postconditions or counters). The base class Condition and its derivative classes stand for different types of conditions in forks and loops.

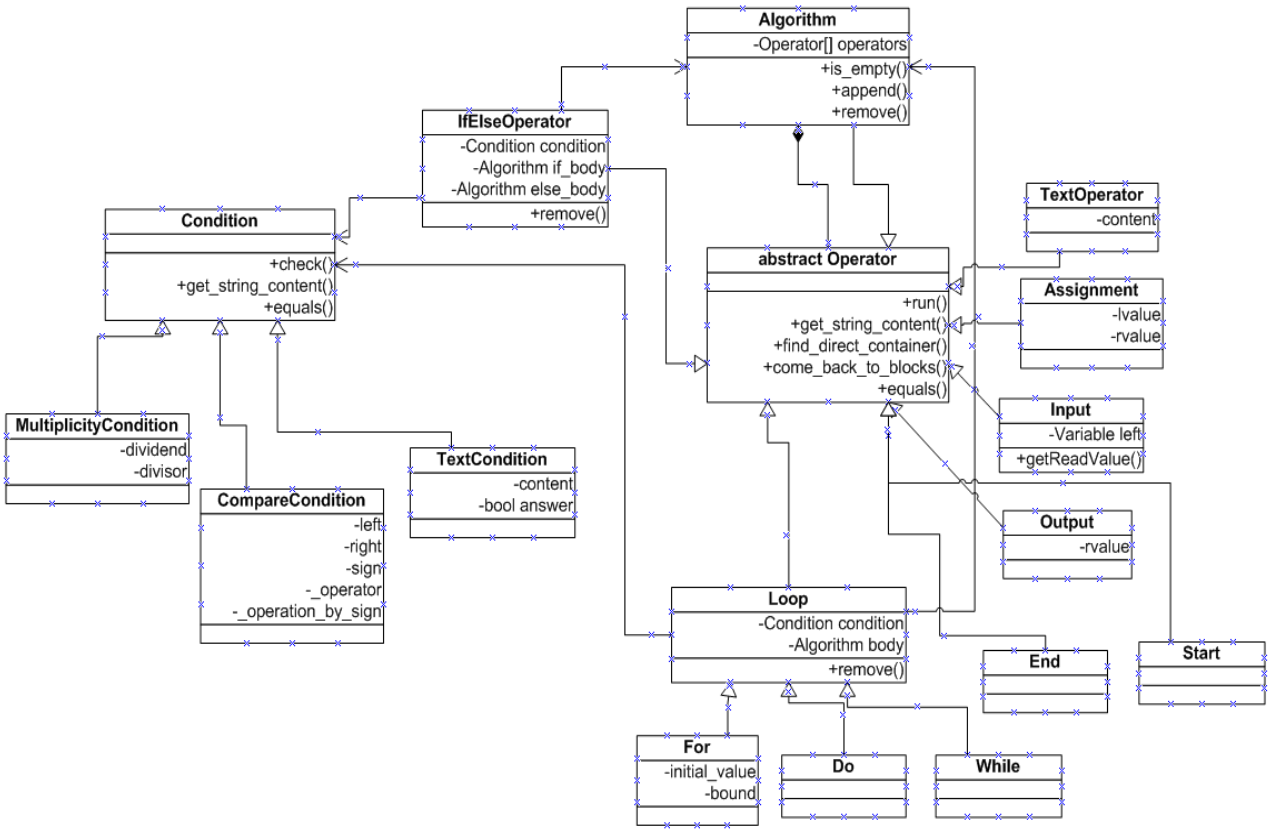

Fig. 4. Class diagram for Algorithm structure

2. The whole structure of the system is shown in Fig. 5. The class *System* contains all the problems (Fig. 3, left part), current algorithm (Fig. 3, center part) and blocks of its flowchart (Fig. 3, right part). A problem is implemented with the class *Task*. To run or trace an algorithm along different control paths the idiom of an iterator is used. The classes *RunIterator* and *TraceIterator* are responsible for running and tracing flowcharts respectively. The class *StateStack* is responsible for UnDo and ReDo commands in the process of the flowchart construction. The class *Comparator* is used for checking the current flowchart i. e. comparing it with the solution represented by the class *Solution*.

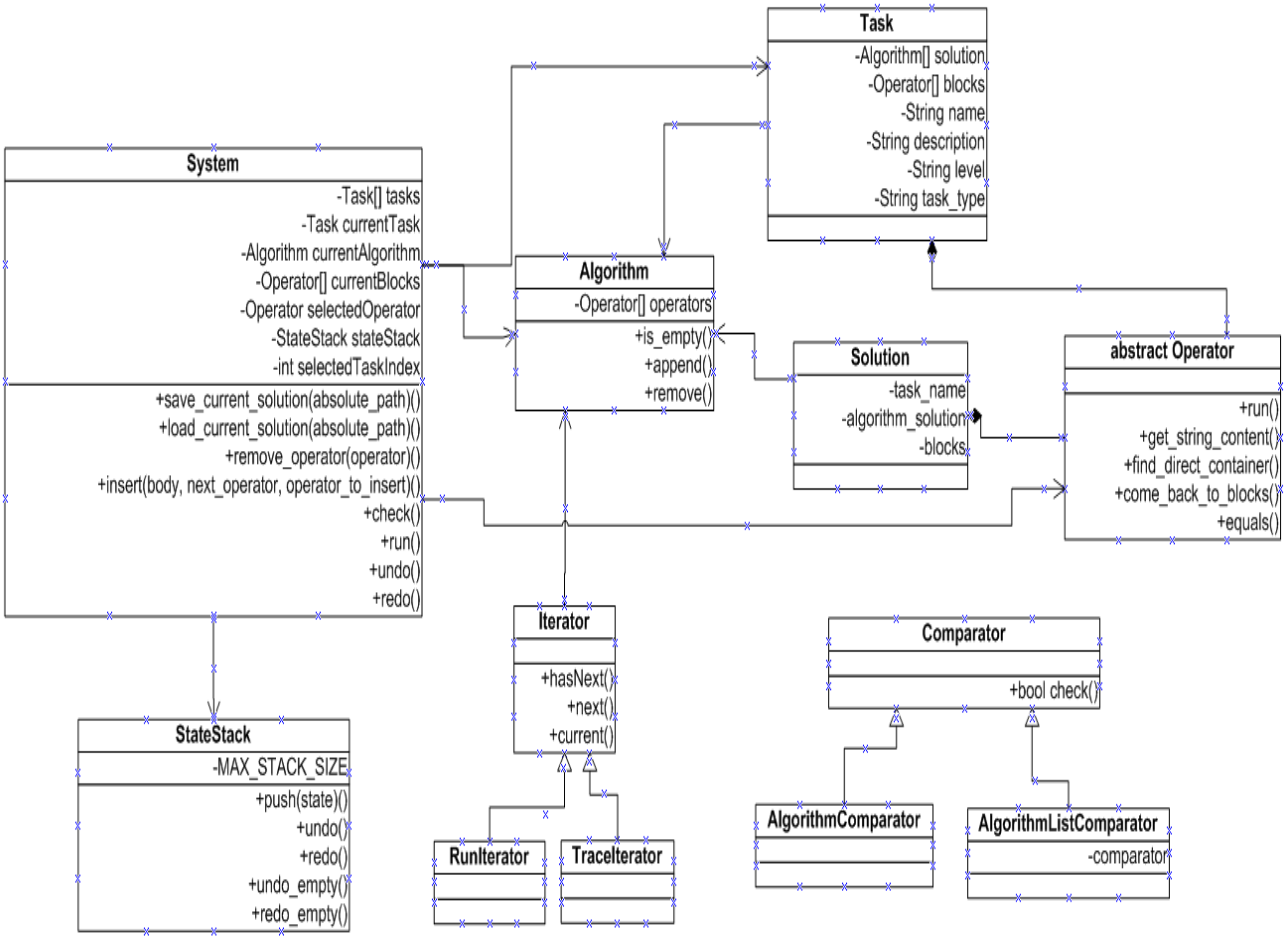

Fig. 5. Class diagram for the whole system structure

3. The diagram in Fig. 6 shows the classes responsible for running or tracing an algorithm. The variables and their current values are stored in the class *Memory* in the dictionary. The *Memory* and the *MemoryViewer* classes implement the Observer pattern [14].

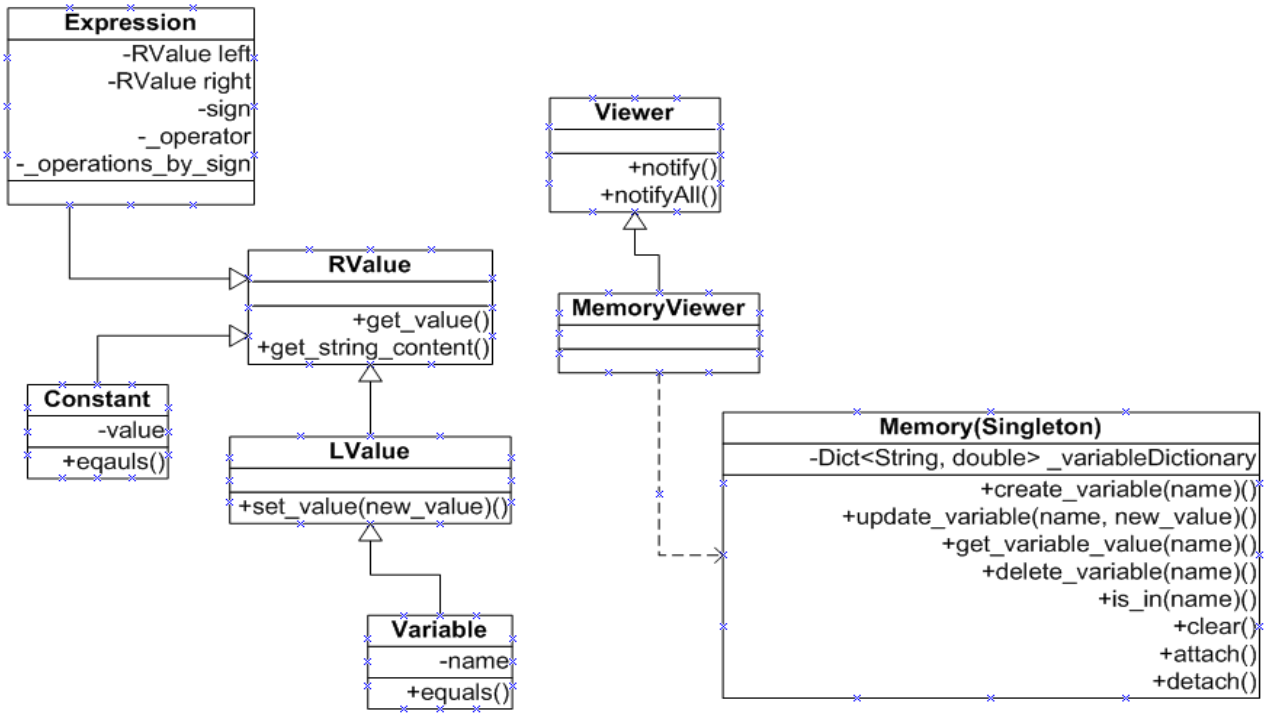

Fig. 6. Class diagram for tracing the values of variables

4. All the classes shown in Fig. 7 are responsible for proper rendering of flowcharts and their blocks in the working area.

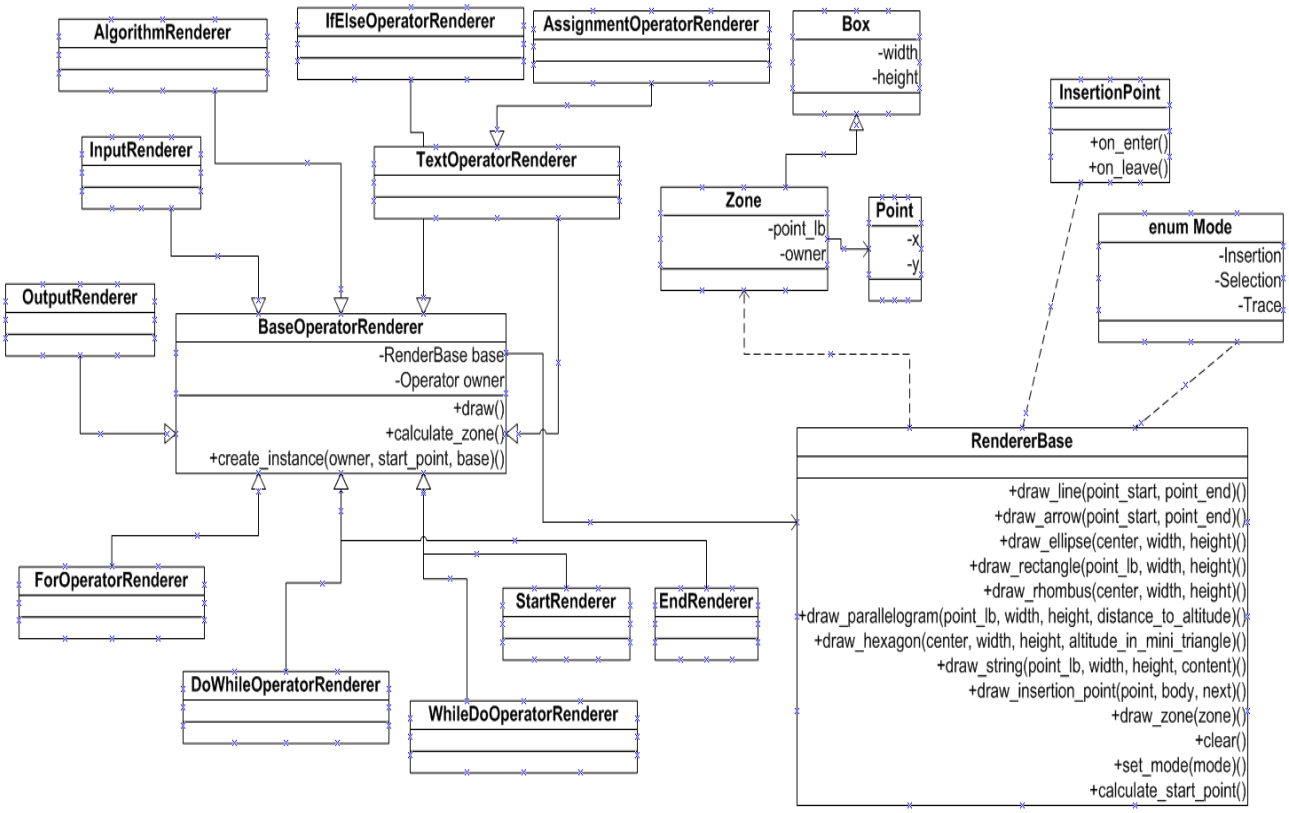

Fig. 7. Class diagram for rendering

5. The diagram shown in Fig. 8 is a simplified structure of the system's GUI.

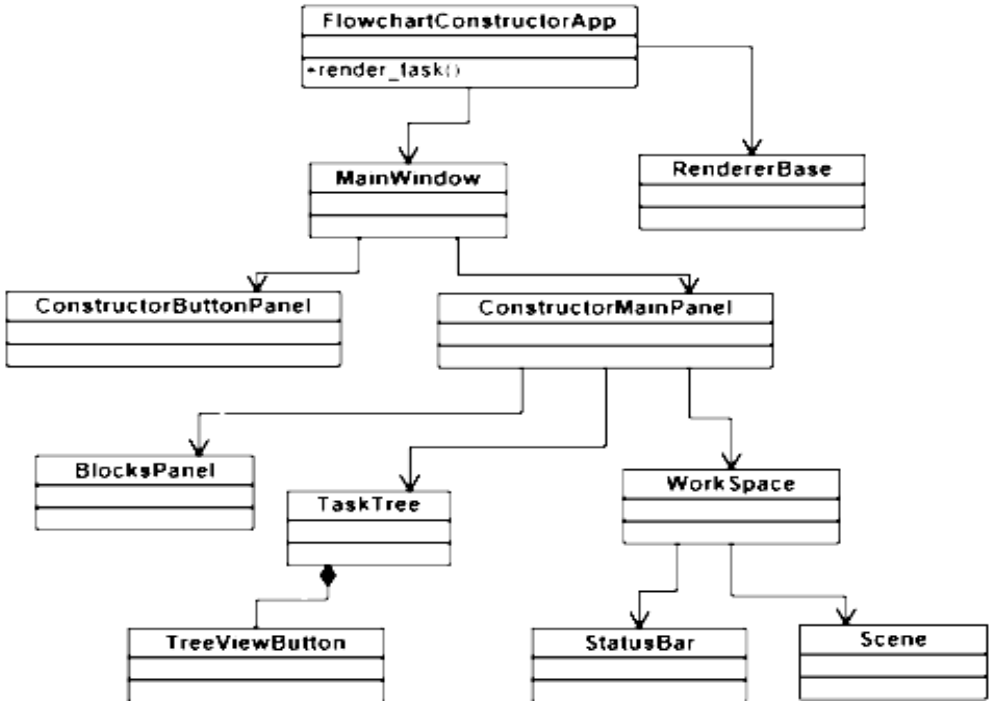

Fig. 8. Class diagram of GUI structure

6. The classes shown in Fig. 9 are responsible for storing data in json-format files. The class *Accessor* implements a bridge [14] between the business logic and the storage. The class *Logger* is an auxiliary class to follow errors in the process of system's operation.

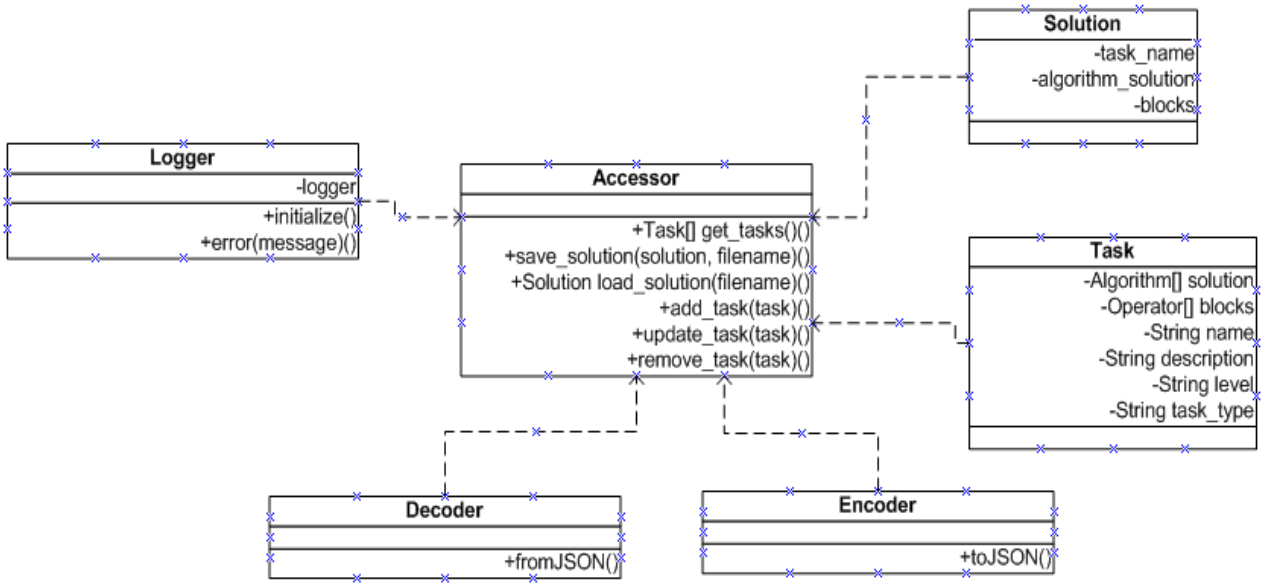

Fig. 9. Class diagram for access to data storage

4. Implementation of Algorithms Constructor

This section presents several snapshots of the trainee module to show the operation of the system.

The first snapshot in Fig. 10 shows the beginning of work: the left panel offers a problem to solve with the selected one being highlighted, the upper part contains language selector and menu, the right panel shows blocks of the future flowchart, and the working area contains the text of the problem and the blank flowchart with the Start and End blocks only.

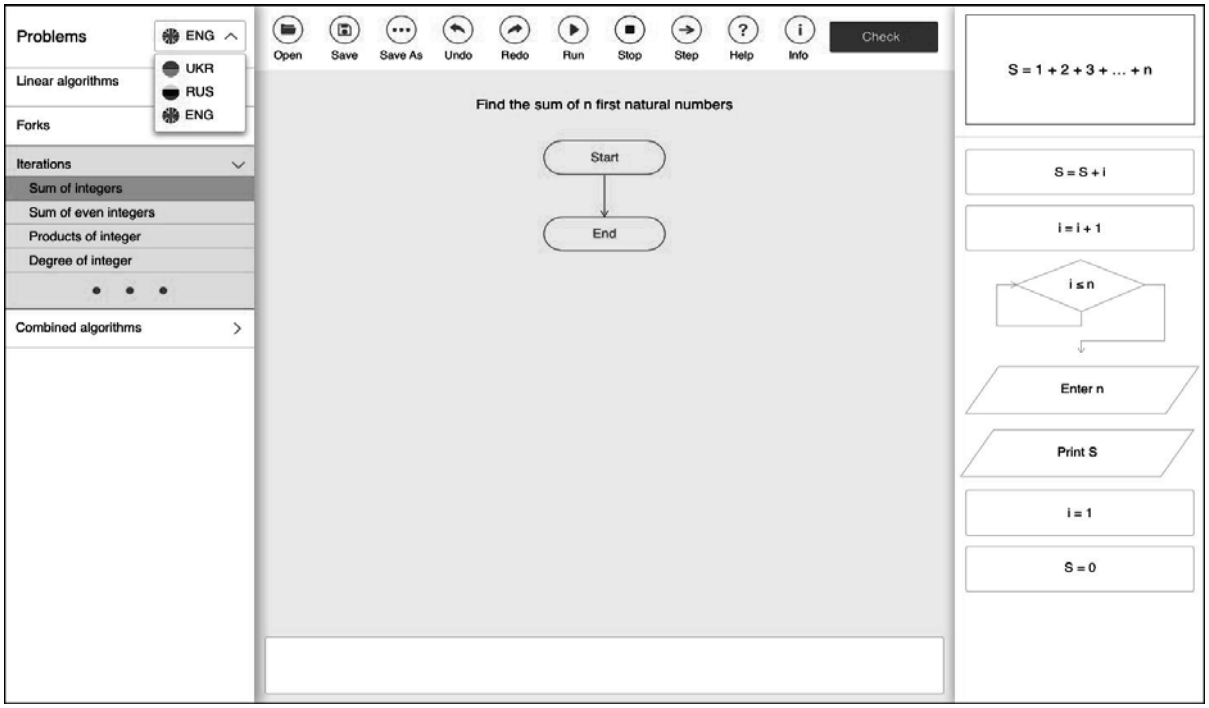

Fig. 10. Beginning of work

To insert a block into the flowchart one should first select it on the right panel, which results in bold points on all arrows where it can be inserted as shown in Fig. 11. Clicking one of these points results in inserting the selected block exactly into this place.

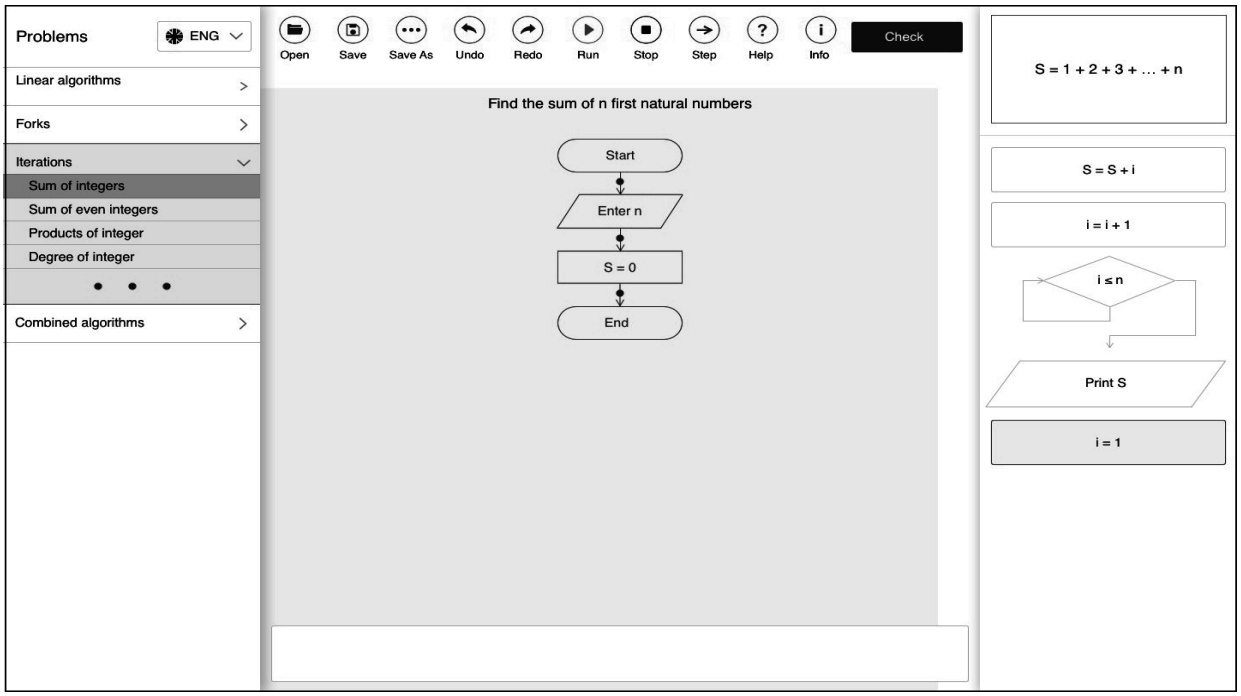

Fig. 11. Inserting the selected block

After finishing the flowchart construction one can check it just pressing the Check button and then run or trace the algorithm. The process is shown in Fig. 12. The right panel contains now the fields where the values of variables should be entered. The results of the algorithm's "execution" appear in the output area at the bottom of the screen.

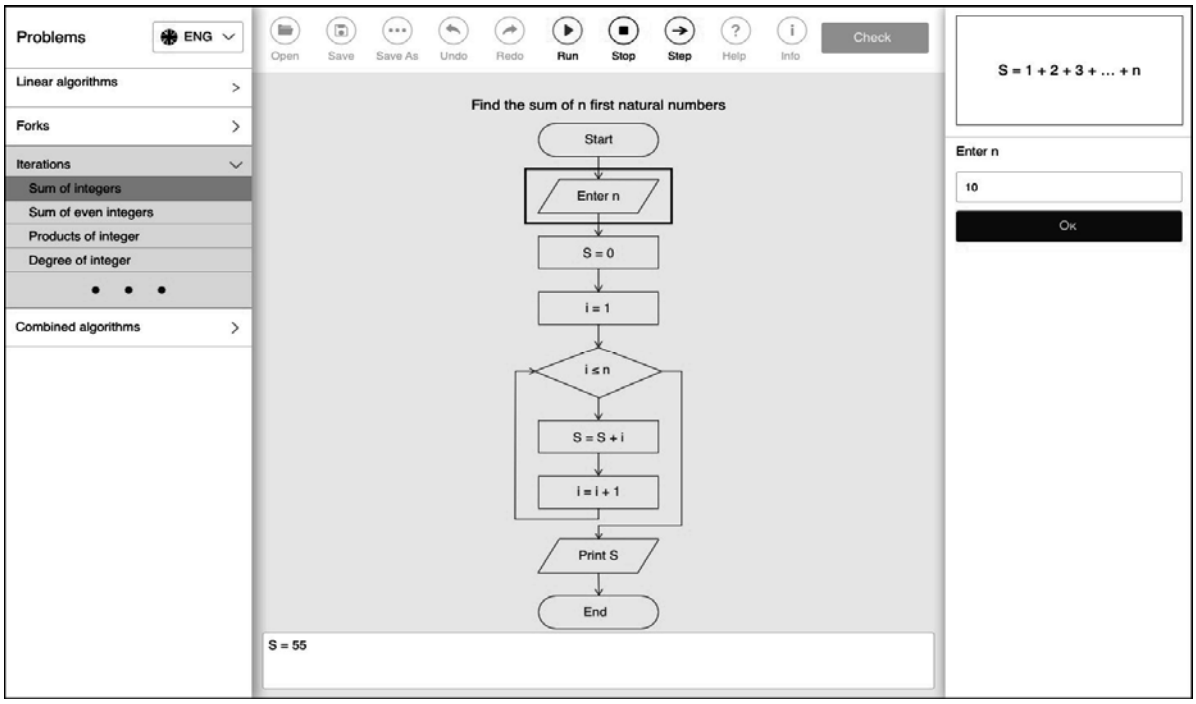

Fig. 12. Running an algorithm

In the process of tracing the algorithm step by step the current step is highlighted by the red border and the current values of all variables are shown on the right panel. For example Fig. 13 shows the current values of variables on the third iteration of the loop.

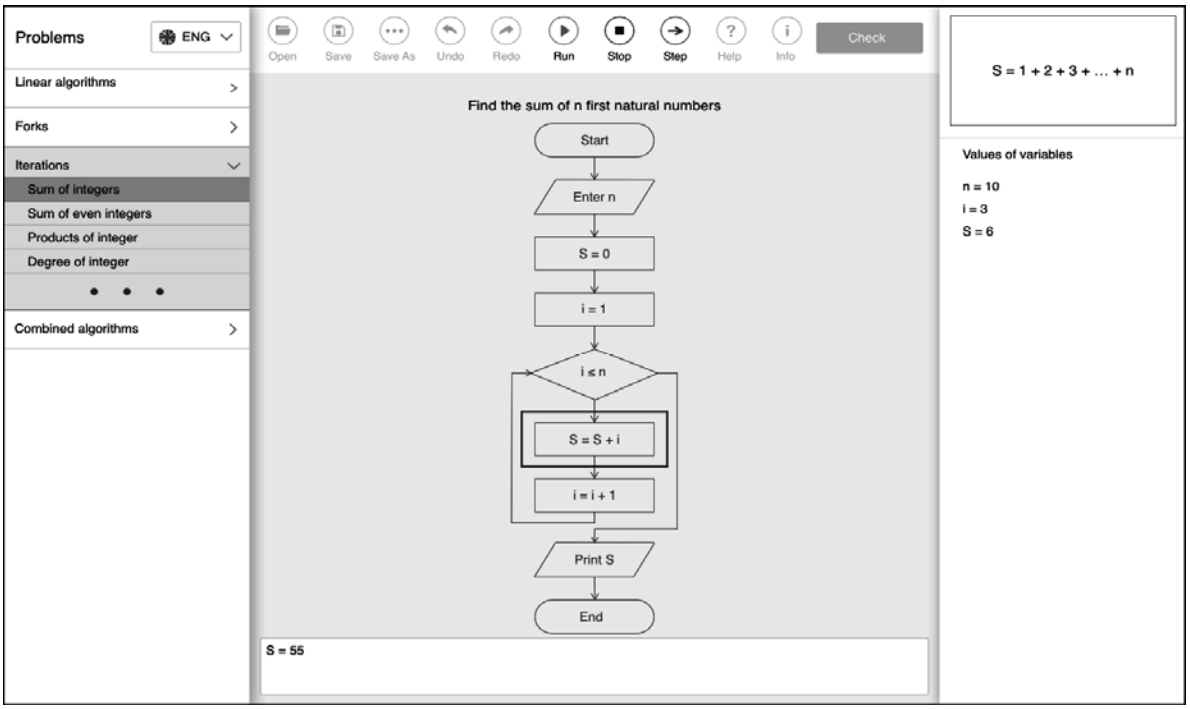

Fig. 13. Tracing an algorithm

5. Evaluation Results

The first version of the system has been in use in Ukrainian schools for about two years. Several webinars for teachers using it were organized by the publishing office "Ranok". All the feedback was analyzed and taken into account. The new version presented in this paper was spread out via the site of the publishing office "Ranok". There is positive response from teachers. Teachers confirm that it helps to motivate their students and greatly lessens the teachers' load.

Our university conducts courses for teachers where we also present this system. At our university we use the system in the course of introduction to algorithms for students of different specialties. The results show the system to be useful for better understanding of algorithms fundamentals.

Conclusions

The paper presents a software system for learning and teaching basics of algorithms through flowcharts construction. It can be used by trainees to learn how to create algorithms of different complexity levels using only basic algorithm's structures (forks and iterations of three types), "run" the constructed algorithms with different input data and explore all control paths. It also can be used by trainers to fill in the system with problems and their right solutions in the form of flowcharts. The situation with several right solutions to the problem is covered.

The system was developed exceptionally for educational purposes. It can be used in any educational institution from elementary school to university to learn and teach basics of algorithms and to understand basic algorithm structures.

The feedback from teachers using the system shows it to be useful exactly for the purposes it was developed.

References

1. McHenry W. K. R-Technology: A Soviet Visual Programming / W. K. McHenry // Journal of Visual Languages and Computing. – 1990. – Vol. 1. № 2. – P. 199–212.

2. Velbitskiy I. V. Next generation visual programming technology with R-charts / I. V. Velbitskiy // MEDIAS-2012. Dedicated to 100anniversary of Alan Turing (IEEE) : plenary report. – Cyprus, 2012. – P. 14–34.

3. Velbitskiy I. V. Next generation visual programming technology / I. V. Velbitskiy // 11-th IEEE EAST-WEST DESIGN & TEST SYMPOSIUM (IEEE EWDTS), Russia, Rostov on Don, Sept. 27–30 : plenary report. – Rostov on Don, 2013. – P. 404–410.

4. Velbitskiy I. V. Graphical programming and programs verification / I. V. Velbitskiy // 9-th IEEE COMPUTER SCIENCE & INFORMATION TECHNOLOGIES CONFERENCE, ARMENIA, YEREVAN. – YEREVAN, 2013.

5. Drobushevich L. F. Common use of UML and R-chart notations in the training process for software system development methods / L. F. Drobushevich // MEDIAS-2010. – Cyprus, 2010. – P. 73–77.

6. Lvov M. Video-interpreter of search and sorting algorithms / M. Lvov, A. Spivakovskii // Informatizatsiya osvity v Ukrayini: stan, problem, perspectivy. – Kherson, 2003. – P. 100–102.

7. Spivakovskii A. Web-environment for learning basics of algorithms and programming / A. Spivakovskii, N. Kolesnikova, N. Tkachuk, I. Tkachuk // Upravlyayushiye sistemy i mashiny. – Kiev, 2008. – P. 70–75.

8. Spivakovskii A. Video-interpreter of algorithms of integrated environment for learning basics of algorithms and programming / A. Spivakovskii, N. Kolesnikova // Third International conference "New information technologies in education for everybody". – Kiev, 2008. – P. 399–404.

9. Spivakovsky A. An integrated training environment for the university course "Basics of algoritmization and programming" / A. Spivakovskii, N. Kolesnikova, N. Tkachuk, I. Tkachuk // Information Technologies in education for all. – Kiev, 2007. – P. 240–248.