Programming into Graphs of a New Generation and the Single Graphical Shell for All Languages

Velbitskiy, Igor

Glushkov's Fund
Kiev, Ukraine
e-mail: ivelbit@gmail.com

ABSTRACT

In programming into graphs of a new generation it is for the first time offered not to write and draw programs during their entire life cycle by using graphs from mathematics which are loaded with only horizontal arcs (ISO 8631: 1989). Drawing is easier and faster, especially for the touch screen. All the traditional (since 1947) machine-oriented operators like **if**, **for**, **goto** and brackets **begin-end** are excluded from programming. **They are outdated**. For a human they are too many, too complex, empirical (not strictly defined), and primitive (small power). *To neutralize* them a human spends too much efforts to create a huge number of languages, systems and environments of programming that are "supposedly easier," but in fact they divide specialists and make programming too complicated and inaccessible to all. Instead of this of all, only **one** (mathematically rigorous) simple graphical and *human-oriented essence* (**R-scheme**) is offered. As a result, programming is simplified, accelerated and improved many times, has a proof of the correctness of programs and the programming process, and self-documenting including documenting the motivation of made decisions, and supports mathematical derivation and automatic generation of programs and the tests for them. A software program depicted in this graphical notation is 100 or more times smaller and easier to read and comprehend compared with the traditionally recorded programs, in the form of texts in modern programming languages in flow-charts, and in UML-diagrams. Unlike the latter, R-scheme does not contain complex profiles (forms), only one graphical structure that is used throughout the life cycle of programs for recording and algorithms, as well as software projects, computer codes, the network schedule and program documentation. Programming for the first time gets the mathematical basis (the graph theory) and is available **to all**.

Keywords

Polyglot Programming, Visual programming, Programming into graphs, graphs loaded along arcs, RR*-schemes, a graphical shell of programs, compactness, easy programming, quick entry into the computer, proof of correctness of program, network diagrams, 3D programming, automatic programs and tests generation

1. INTRODUCTION

Programming by using graphs (GP) began to emerge in the '70s with the development of computer control systems and space missile complexes in former Soviet Union and the formal recognition of the need to document the process of software development to facilitate the rapid introduction of permanent fixes and improvements and also was facilitated by the work of Dijkstra [1], who for the first time has proven redundancy of traditional programming operators and primitive artisanal nature of the organization of work with them. As a result of GP development, it has been proposed to abandon all traditional computer-oriented statements of programming languages and use math graphs instead, which were called R-schemes, where «R» is from word «rational» [2]. A concept of drawing has been introduced into programming as a basic documentation unit for a program under development. This documentation, which is a visual or graphical representation of the product, coincides with the product itself - the program. This was not achieved in programming and in any traditional industries (aerospace, automotive, construction, etc.)

R- schemes are loaded only on the horizontal arcs and used throughout entire software development life cycle (SDLC) of development only of graphic programs. This greatly simplified and speeded up the development of programs to ensure their effectiveness from memory and CPU perspectives, making them self-documenting and compact 100 times or more.

Later it became clear the versatility of the proposed guidelines [3-8] and compatibility of GP with all the innovations of modern programming (OOP, AOP, WEB, etc.), which were carried out in order to neutralize the disadvantages of traditional operators of programming languages. New opportunities for GP, which are still absent in traditional programming, have been identified: proof of program correctness, automatic test generation, and automatic generation of programs, matching the type of programs (project, code and documentation) with the network graph of its development, the widespread use of color, 3D-programming, etc. It became clear continuity with what has been achieved before GP, and the possibility of introducing the new without destroying what is good in traditional programming.

2. THE ESSENCE OF PROGRAMMING INTO THE NEW GRAPHS

In GP (ISO 8631: 1989 [5]), instead of the traditional operators of the type if, for, go to, etc. (total number is about 10) only one horizontal arc - R- scheme – is encouraged to be used throughout the entire life cycle of a program development. Figure 1 presents an R-scheme with two nodes and one horizontal arc which has a Condition written above the arc, and one or more Actions written below the arc that are executed if the condition is "true." For condition and actions entries in one or more lines any language: English, Armenian, Russian, Chinese, etc., the language of mathematics, and any programming languages can be used.

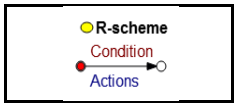

Figure 1. R-scheme (one arc to the right or to the left).

Any number of alternative arcs directed to the right and/or to the left may be drawn for a node in an R-scheme, as shown by Figures 2-7. These arcs are counted (and checked for a true condition) from top to bottom until the condition is true.

Then the appropriate action is executed and transition is performed along the corresponding directed arc into another node of R-scheme - into a new state of a program. If the condition is not on the arc, the actions are carried out under the arc undoubtedly.

Such structuring of the information on the arcs provides large visibility, supports compact programs, and eliminates the use of traditional block of brackets type: **begin end**, **{ }**, **<? ?>,** etc., refuse to write of the statements, one per line, and a ladder. Figure 2 shows an example of defining a new operator "3 (can be any number) Conditions and cycle". Figure 2a shows its R-scheme, Figure 2b shows its R* mathematical model without the implementation details, and Figure 2c - an equivalent recording in C++, where the red marked redundant characters are compared with the R-scheme in Fig. 2a.

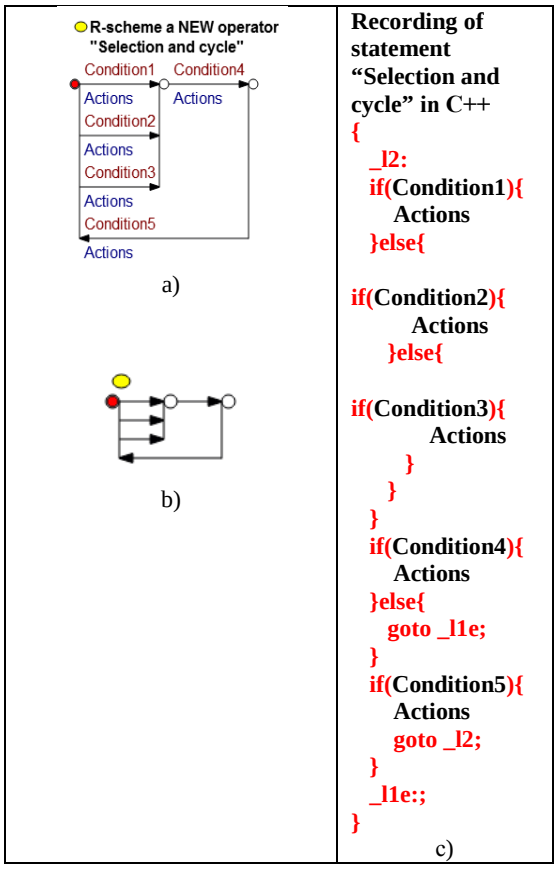

Figure 2. Recording of statement "Selection and cycle" in RR*-schemes and in C++. Highlighted in red in C++ are the symbols superfluous for an R-scheme.

There are 79 extra characters or 133 (about 62.6%) with indentation (with paragraph) and line feed. Recording of operator Figure 2a is 2.5 times more compact, and recoding of operator on Figure 2b - 8.6 times more compact than the recording in C ++ on Figure 2c. For entering the traditional operators (without the header above) on Figure 2c requires 225 keystrokes of keyboard whereas entering of the equivalent operator (without a title about the ellipse) on Figure 2a takes a total of 19 keystrokes keyboard, 11.8 times less.

Keywords such as **real**, **procedure**, **function**, **form**, and so on, can be used above an arc which are always true and set a new understanding and use of all elements of R-schemes. For example, Figure 3 presents parallel execution of all arcs coming from the top left node in accordance with the key word on the arc. The semantics of such task visually displayed in standard notation to Figure 3c. Many of the descriptions of variables and appropriate keywords can be excluded from the GP, because they can be easily calculated based on the structure of the recording of these variables on R-scheme arcs, as in classical mathematics.

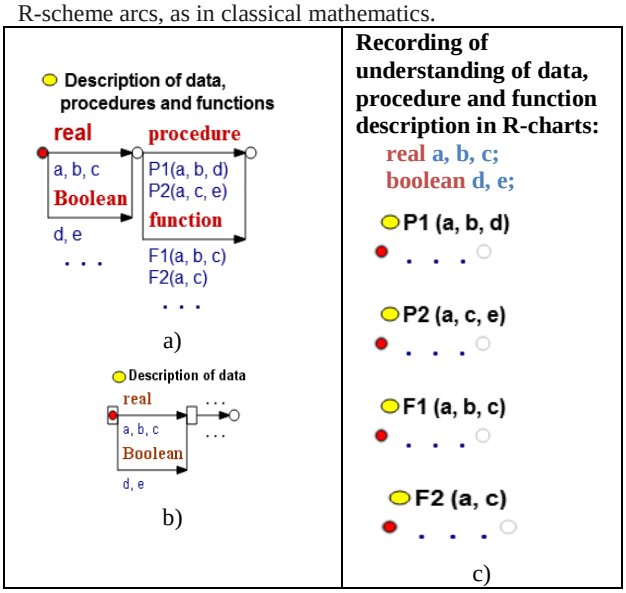

Figure 3. Definition of data, procedures and function.

The nodes of R-schemes have no name, but may have different configurations and color for the implementation of traffic lights in the program, the definition of &-arcs, 3D-programming, documenting the motivation of decisions, etc. For example, a red colored node always defines the beginning of the project, triangular of node - macro R-schemes, on Figure 3b-5 rectangle is used to record arcs that are executed in parallel, Figure 4 shows the principle of organization of 3D-programming and documenting the motivation of the decisions with the help of the node of graph in the form of a parallelogram.

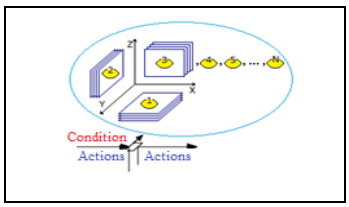

Figure 4. Organization of 3D-programming and documenting the motivation solutions.

Such a graph has a name that is written on top near the yellow ellipse in Figures 1-3,5,7,8. The name can be with or without parameters, Figure 5. A program is defined by any number of interrelated on behalf of such graphs, Figure 5. Unlike traditional recording programs such graph R (Figure 5) has a record of R * (Figure 6) mathematical

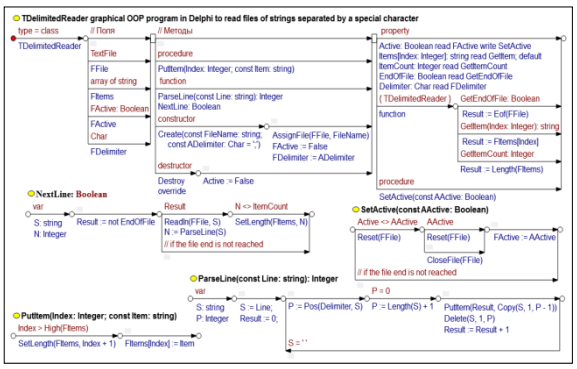

Figure 5. R-scheme of an OOP program is **7** times **more compact** than a typical program in Delphi [9]

abstraction program without records on the arcs (without implementation details). This recording (R *) allows you to define model programs and execute upon them the mathematical studies, identical transformation, optimization of various parameters - time, memory, classification for a new type of graphics software archives, etc. It is an order of magnitude and more compact circuits for the image of the algorithm (program), and significantly simplifies the design of programs. For example, in Figure 5 R=7 compactness, a compactness of Figure 6 R*=35,7 times more compared with the recording of this program conventional manner [9]. See also Figure 2b, 2a-2c, and 7c-7b-7a. Fixed for today compact real graphic shells (RR *) R=19 programs, R*= 130.

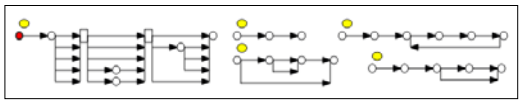

Figure 6. R*-scheme of a program with no implementation details is 35.7 times more compact then in Delphi.

Figure 7 shows a comparison (advantage) of R-schemes with widespread flowcharts - graphs loaded in the nodes (Fig. 7a). As a result, Figure 7bc - this compact executable (!) Program, and Figure 7a - not.

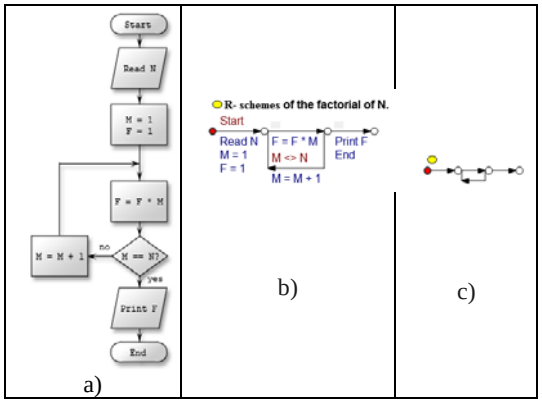

Figure 7. Example of algorithm for factorial of N recorded
in block diagrams (a) and RR*-schemes (b, c).

R-scheme the first one (three in one) allows you to 1) formally write logic of customer requirements and specification for development of a software project in a natural language; 2) write the algorithm, the draft program, the computer code and documentation of the program in any programming language, and 3) record the network schedule of project development, Figure 8. In programming into graphs final software product coincides with the documentation (Figure 5) and the network schedule for its development (the first graph of Figure 5). The supervisor the first time may record the artist name and estimated time on the arc of R-scheme in special brackets.

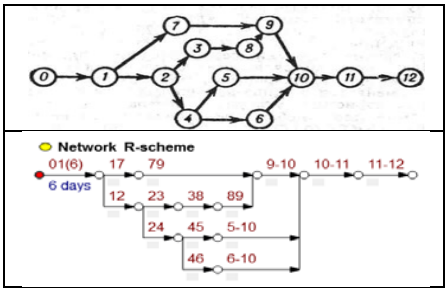

Figure 8. Recording of a network schedule in R-schemes

implementation of job and monitor the quality of work. That did not has in any industry.

Each arc of a graph is entered by a one-touch of a mouse or keyboard, or a finger on a touch screen, so that real programs are entered 1.5-3 times faster. For example, to draw the R-scheme in Figure 2a it will require a 4-touch of a mouse left button or keyboard, on Figure 3a – 3-touch, on Figure 6 – 1 + 2 + 6 + 5 + 2 + 3 + 4 + 3 = 26-touch, on Fig.7c – 2-touch, which is 1.6 times less than the number specified elements (ellipses and arcs) of this R-scheme program. The result is that to set the arc of graph requires less than one-touch keyboard or mouse. Setting texts on arcs is no different from setting texts in the traditional text editor.

Thus, the proposed single graphical environment for programming languages is formed of a single arc. Writing programs in this shell is easier, it does not use traditional operators, keywords, block of brackets and many punctuation marks. It reduces the need for the program narrative, simplifies translation and interpretation of programs. For the first time the program is the subject of mathematics and for the person, not only for computer. The development more corresponds to mathematical derivation of programs from well-defined concepts and rules without contradictions and problems of ambiguous understanding. RR*-schemes are ideal for reverse translation of programs in any language of programming into graphic form. As a result, the program from the "thing in itself", understandable only to the author (not always available), becomes transparent for the development and improvement, Graphic of record is 100 or more times compact, uses less memory to enter into the computer by 1.5-3 times faster (needs fewer keystrokes on the keyboard). The graphic recording is a polyglot (does not depend on the language). For the first time, you can record a program into graphs without implementation details (without records on arcs) for their mathematical analysis and synthesis in order to prove their correctness, optimize memory, speed, etc.

3. GENERATOR OF PROGRAMS IN GRAPHS

Generator of programs - a software, is designed for active programming by generating various embodiments and prompts for a person with the purpose of construction as a result of the graphics program in the form of R-schemes. The initiative is always from the generator. It immediately offers

a person (not knowing anything about his intentions) R-scheme, its future program in the form of Figure 1. The role of the human in the first place is to enter all information available to him about the task at hand and adequate response on the message of generator. If his task has been recorded at some formal programming language (COBOL, C, etc.), the generator will issue a decision in R-schemes automatically in a fraction of a second. Otherwise, the generator starts with a man dialogue in accordance with laid down in its program of generator. This program is constantly being improved (collects intelligence of interaction with a human) and specializes in the corresponding contingent of professionals. The generator is recorded in the R-schemes. Principles of programming in R-scheme allow to automate the process of self-improvement of the generator.

The essence of the work of the generator and the person is 1) in the logical analysis of all the information about the problem, 2) the selection of logic and actions, which in this case must be made, and 3) write formulation of the problem in graphic form into R-schemes. Technology to address these issues is extremely simple: «step by step from logic» or stepwise refinement of the logic of the problem, of the process, of algorithm, program, project, human thoughts, etc. In the future, language formulation of the problem is transformed into a natural human language (usually in the language of mathematics) and then - in the language of the computer. From it all ambiguity understanding is consistently removed. The generator ensures and directs the person in such a way that the human language would become clear for language of computer. Language of computer is constantly being improved and approaches the professional language for the staff of the development of programs.

It is important that the language of this description (R-scheme) is only one (!) for the computer, for the customer, and for all the performers of the project throughout its life cycle. R-scheme is alone, but it is most convenient to the executor corresponding to the project due to the specialization of records on the arcs which are performed for a specific task. R-scheme - the foundation of any project, and the first thing that is approved and fixed in the project. Further implementation details are refined and can be written in any language and an alphabet on the arc.

4. REALIZATION

Currently, a graphical software development environment is implemented as a pilot project (lab version of programmers: A.Hodakovsky and A.Gubov) which includes R-schemes editor – REditor for any texts on their arcs in any language, creation and storage of the project structure, a transformer of R-schemes to R* compact representation and back, a compiler of RR* into C++, etc. This graphical environment is implemented as REditor plug-in for Qt Creator. It is composed of five domains, which divide a computer screen into 5 areas. The main area (about 90% of screen space) that occupies the central part of the screen is Working Pane (WP) which is used for R-schemes creation and editing. Top three lines present Main Menu for creating an architecture and structure of the project environment (Trees, File, and Preferences). Third area is Toolbox (14 graphical icons). Fourth pane is the list of the open Working Panes of R-schemes for entire project. One WP is selected as active (visible). The last area occupies a narrow left (5%) strip of the monitor screen for storing tree-structure of R-schemes for all WPs for development project. It provides an operational (fast) view of WP for any part of the project. Thus, implemented graphical environment has generalized the existing experience of programming in graphs. It allows development of any projects and is complete enough (we estimate at 80%) for a commercial version of graphical programming environment to build on its basis.

5. CONCLUSION

Thus, this article suggests a new mathematical culture of programming which is interesting by its advantages, simplicity, humanity and the natural transition to it for all and not just for programmers. This culture allows you to bring the style proof into a professional programming, and mathematical methods of analysis and synthesis. New proposals are particularly effective for the primary education of programming, so they can be included in the system of compulsory education of professionals of all specialties. Programming should be part of universal literacy, and culture of the society. These proposals begin programming in the new twenty-first century and it is a great promise in the future to build human-computer society and its co-evolution with man. One of the users of R-schemes has said this on a very emotional note: "After two weeks of drawing R-schemes in the editor, I solved the problem, which R-schemes would never solve... I see R-schemes as a useful and beautiful tool not only in programming, but anywhere you need to understand the logic of interaction of parts of any problems or issues. Compared with the flow-chart and UML diagrams, they have considerable advantages. They are performed on a computer throughout entire life cycle of any work, more visually, more compact and are not cluttered with unnecessary details (figures) and suitable for nonlinear imaging algorithms or data structures (e.g. C ++ classes). Therefore, I do not understand why this tool does not enjoy crazy (wide) popularity" (V.Bushkevich, 04.05.2015).

It is known that in the human brain the connections between neurons and brain regions play a major role, as in the R-schemes - an arc between the nodes are not limited to the number, direction and 3D-dimension relations between the nodes of one and several R-schemes. Nothing is like this in modern programming, and on the parameters (more discrete and more clearer) they are better than connections in the human brain. Memory makes all this and the huge potential of modern microelectronics and devices today to reflect on the development strategy of our computers and to analyze the main shortcoming of them from 1947 - a complex and primitive organization of the programming process. Now it is the time for analysis and selection of the best COMPUTER and SOFTWARE of radically new of brain-like generation for the new millennium. We estimate the complexity of creating such a project in ½ years and plus ½ years for marketing and sales. These periods may be significantly less taking into account the implementation of the existing version of R-schemes, international experience in programming and phased implementation.

REFERENCES

[1] E.Dijkstra, «Letters to the editor: go to statement considered harmful», Communications of the ACM 11 (3): pp. 147–148. doi:10.1145/362929.362947.ISSN0001-0782, 1968.

[2] "SergeevVG – The chief designer of control systems of rocket and space complexes.To100-anniversary of the birth", Space-Inform, Kharkiv, 448 pp., 2014.

[3] V.M,Glushkov, I.V.Velbitskiy, «Programming technology and problems of its automation», USIM, №6, pp. 75-93, 1976.

[4] I.V.Velbitskiy, Programming technology. Техника; Ukraina, 279 pp., 1984.

[5] «Information technology, Programme constructs and convention for their Representation», International standard ISO/IEC 8631, Geneva 20, Second edition, 1989.
[6] W.K.McHenry, «Technology: A soviet visual programming». J. of Visual Languages and Computing, v.1, №2, pp.199-212, 1990.
[7] I.V.Velbitskiy, «Graphical Programming and Program Correctness Proof», CS and IT Proceedings of 9th International Conference, Erevan, *IEEE* Conf. DOI: 10.109/ CSITtchnol, 6710368 pp.1-8. 2013.
[8] I.V.Velbitskiy, «Graphical programming of new generation. Single universal graphical shell for all programming languages». Global IT 2015, Las Vegas, USA, p.1-8, 2015
[9] A.N.Valvachev «Programming in Delphi. Chapter 3.6» http://www.rsdn.ru/article/Delphi/Delph