Uniform universal graphical shell to record programs in any languages. New generation visual programming technology.

Velbitskiy I.V.
GLUSHKOV Fund
Kiev, Ukraine
ivelbit@gmail.com

***Abstract*— New generation visual programming technology, which uses so-called R-charts (ISO/IEC 8631:1989) is defined (VTR). R-charts are direct graphs, loaded through arcs. The main difference of VTR is the simplicity and uniform graphical form to record programs in the whole life cycle, which provides with the new principles for designing, debugging, visual and compact (more than by one order) record in comparison with the record in existing programming languages. Graphics program input is easier and quicker by an order. Due to its simplicity it is available to a great number of specialists and not only to programmers. The new graphic paradigm of VTR is defined; it is based on the uniform graphical shell for all languages, consisting only of one horizontal arc. All traditional statements: goto, if, for, while, etc., labels and brackets of begin-end, {-} type are excluded from programming. The new technical principles, linguistic, parallel, 3D and multi-dimensional graphic programming, correctness proof and self-descriptiveness of graphics programs are introduced. The color is widely used in record of graphics programs. It is well integrated into existing programming systems. At present moment VTR graphic programming has been implemented in Qt-Criotor C++ environment (www.glushkov.org). It allowed quite reasonably performing the analysis of VTR advantages in comparison with traditional technologies that use modern programming languages, UML, OOP, SOA, etc.**

Keywords – graphic programming, programming without statements of goto, if. for, while, etc. type, color in programming, metalanguages, net graphs, parallel (3D and multi-dimensional) programming, program self-descriptiveness, program correctness proof.

I. INTRODUCTION: HISTORY

At the end of the 60-ies in last century for the first time we introduced the concept of programming technology and orientation to technical principles for development of on-board programs for aerospace systems of P-36M (SATANA) type in the former USSR. So-called R-technology of programming for wide use has been developed as a result from generalization of those works [1-3,6]. This technology was widely known in the USSR and COMECON member-states. Three all-Union and one international conference, tens of specialized seminars were performed on issues of R-technology, as well as more than 600 papers in different fields of its application were published. Before breakup of the Soviet Union and COMECON R-technology won at the competition of technologies for joint development of uniform COMECON Technology in all 10 countries of the Community. In 1988 it received the international standard **ISO/IEC 8631:1989** [3].

Today, twenty years afterwards, the works on R-technology were renewed. Its analysis and comparison with achievements in programming was performed. The concept was revised taking into consideration the development of technologies, new languages and environments. The new concept was implemented on modern platforms and discussed at International conferences [7-9].

As a result, we made the conclusion that the new concept of R-technology has not become obsolete but it fits well into modern tendencies in development of programming, upgrading them, establishing the new generation Visual Programming Technology with R-charts (VTR), which improves the modern approaches to programming by some times. The main thing in suggested concept is that R-chart is not the new (one more) language but **the universal graphical shell – being the uniform (!) for** all known languages: Fortran, Cobol, C++, Java, Delphi, etc. At the same time it is more powerful and much easier, more visual and compact than those languages and simplifies the existing process and software development environment – IDE.

II. DEFINITION OF VPT BASIS

The graphical shell of new generation consists of charts. The elementary chart, which is called *R-chart*, consists of one horizontal arc between two tops, see Fig. 1. The arc has the direction only to the right or to the left. Any number of arcs to the right and/or to the left may come out of the R-chart top, see Fig. 2. The arcs that come out of the top are viewed (read, understood, analyzed and executed) successively downward and from top to top along the corresponding arrow of the arc, starting from the first top on the left and finishing with the last top on the right. Vertical arcs in R-charts are auxiliary, they connect tops with horizontal arcs, which are the main ones and are downloaded with information. The top has no name and sets *condition* of program or process of its development. The graph input is made only by input of horizontal arcs around selected one or two tops. For example, it is necessary to have 12(n-1) presses of mouse or keyboard buttons to input R*-chart in Figure 2 consisting of 13(n) arcs. The first arc of R-chart and captions on it (Fig. 1) are always drawn automatically.

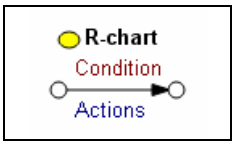


Figure 1. Elementary R-chart (one arc to the right or to the left) to record any program in any language.

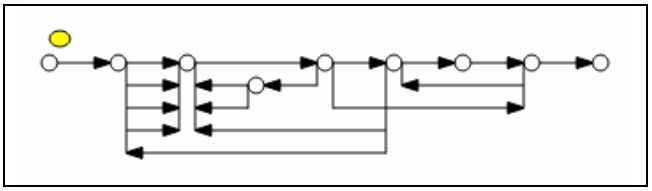


Figure 2. Record of R*-chart for 4 cycles without limitation of a number of alternative arcs and without details of implementation (without record of information on arcs).

R-charts on the arc at the top have *Condition* to pass along the arc and at the bottom – *Actions*, being made at this, see Fig. 1. Condition and/or Actions may be absent. No restrictions are imposed on record of Conditions and Actions (Fig. 1, 3-5) – they can be recorded in any language: Russian, English, Chinese, Mathematical, Programming language, etc. in one or some lines.

If Condition is true or absent, so, Actions under this arc are not made and the next downward arc that comes out of this top is considered, and if it is absent, then, the arc that follows after the top, to which it (arc with the last false condition) points.

If Condition starts from *key* word, so, this condition is always the true *Boolean constant of R-chart* and is made by special way, stipulated at definition of graphical shell (Fig. 4). As a rule, Boolean constant *is joined* to its Action, forming the single result code but there can be more complex actions (See the second column on the right in Fig. 4). Such capability provides with efficient graphical setting the descriptions of data and metadata.

If Condition is a free text (Fig. 5 in dotted frame), so, the value of condition is defined by another R-chart with the name from the frame.

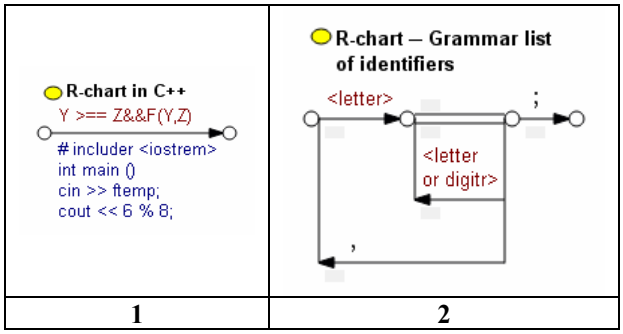


Figure 3. R-chart for fragment of program in C++ language and grammar list of identifiers.

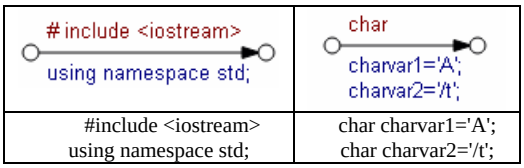


Figure 4. Record of Boolean constants for R-charts and their corresponding codes in C++ (below in the column)

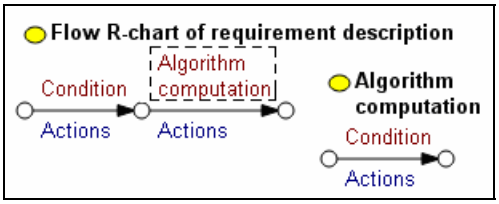


Figure 5. Principle for connection of free texts (in dashed rectangle) with definition of R-charts.

By default a free text in dashed rectangle is considered to be true after return from another R-chart and the actions below it are executed. Such capability provides with efficient development of large software projects in single language of R-charts.

Any chart is built from elementary R-charts and is used **in all (!) life cycle** of program. Each R-chart has its name, which is recorded around yellow ellipse (Fig. 5). A new R-chart is set by double click of left mouse button on free space of Work field in *Graphic editor – GIDE*. According to ISO/IEC 8631 such graph is called R-chart. In theory R-charts are equal to Turing machine.

III. EXAMPLES FOR RECORD OF R-CHARTS FOR PROGRAMS

Fig. 6 has the record of R-chart for exit and cycle statements and its corresponding record in C++ language. R-chart is incomparably more visual, more compact by two times and it is input into computer by **37** times quicker (only by **5** key presses). ½ of odd symbols in C++ is excluded from record of R-chart (they are in red color on Fig. 6). A VPT user does not see the right part of Fig. 6, which is shown only as the explanation of R-chart in traditional notations for reader.

The record in R-charts of traditional cycle statements (Fig. 7) is more visual and powerful not least because the arcs for their records are loaded only partially – some arcs have no conditions or actions, or both.

Fig. 8 shows the comparison of R-charts with other known graphical methods to record algorithms, such as UML, Flow Chart, etc. R-chart is the only graphical method, which is used in all life cycle of programs and is the most compact one among all known.

R-chart (graph, loaded along arcs) is a universal way to set the information in mathematics. Historically such graphs has been used long ago in *Project management system,*

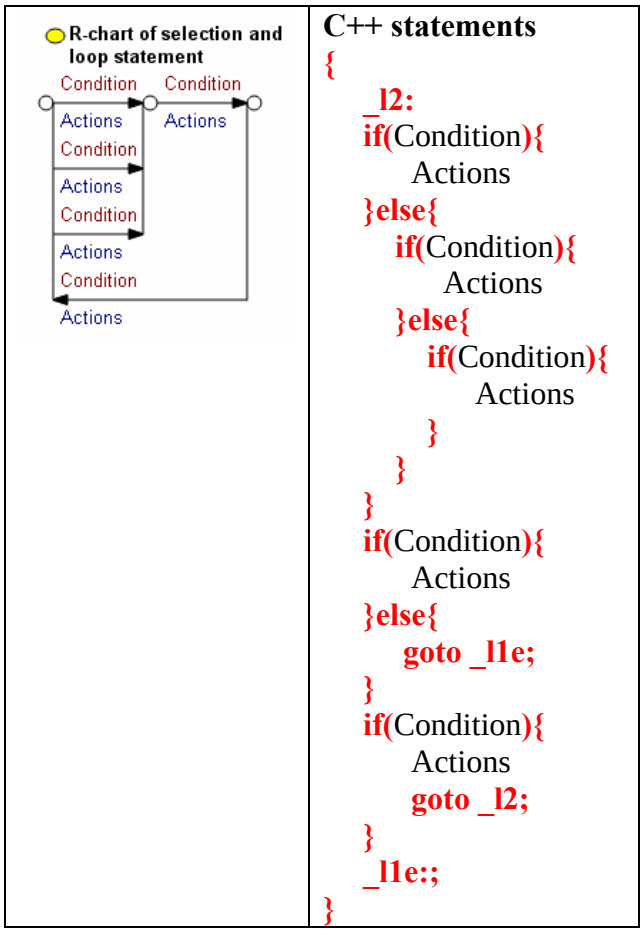


Figure 6. Record of exit and cycle in R-charts and C++.

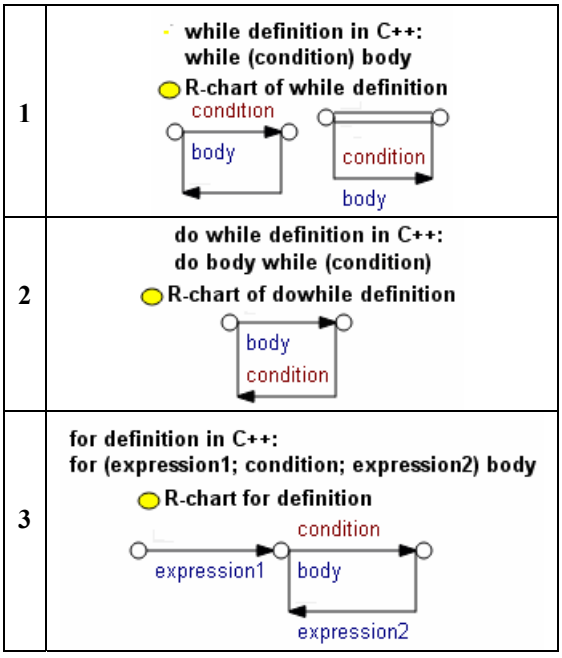


Figure 7. Record of cycles definitions in C++ and R-charts.

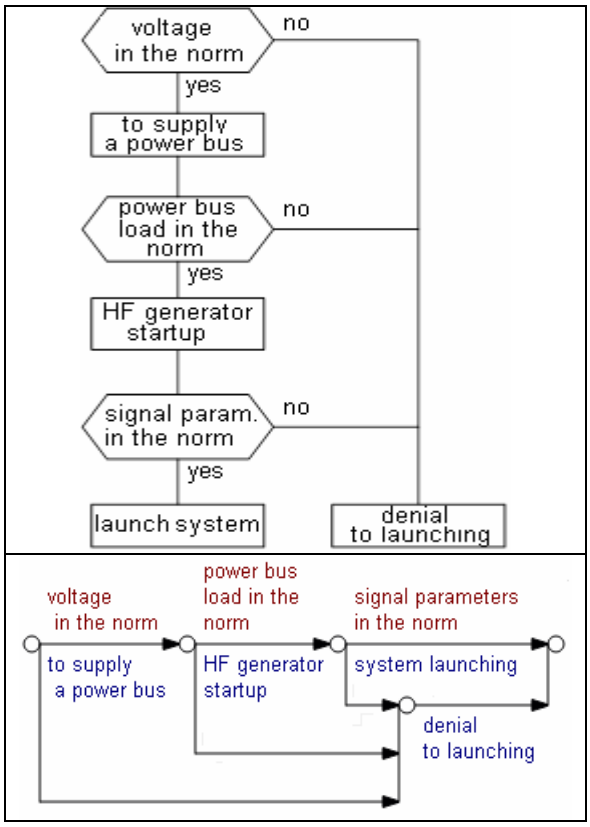


Figure 8. Fragment for pre-launch preparation of reusable space shuttle BURAN, recorded in UML and in R-charts.

providing with the most visual presentation of progress of works, their provision and completion, see Fig. 9. The name of works is usually recorded on the arc above such graphs: 01, 11, 12, etc. and below or in brackets (see the first arc of R-chart, Fig. 9) – time for their execution. The use of such systems opens the large perspectives for organization of

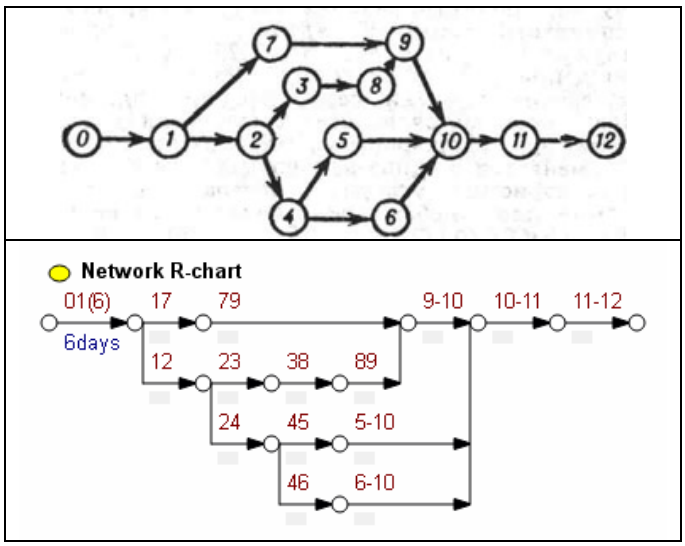


Figure 9. Record of network graph (on top) in R-charts

planned industry on development of software projects in the uniform system of notations: desires of Customer, Program objects and Executive system of their development.

Thus, VTR visualizes first of all that part of existing traditional programming languages, which serves as the main source for meshing, errors and difficulties in development of program – so-called language statements. Linear parts to record programs: expressions, functions, assignment, data description, input, output statements and other structures of programming languages are embedded into graphical shell almost without changes. Such record of linear parts provides with the visualization (demonstrativeness, compactness) and consistency with existing programming languages.

IV. BASIS EXTENSION

VTR allows the flexible development of drawing tools. For example, *special* double arc without arrows is used to draw graph loop type (Fig. 10) that allows more vividly to draw **while** type statement, see Fig. 7.1 and very many situations while recording the real algorithms and macro definitions.

The tops of R-charts may also have the special configuration – square, diamond, rectangle, etc., see Fig. 11. For many applications: calculation of 3D trajectories, parameter representation of multi-dimensional space. It means the special condition of R-chart: parallel execution

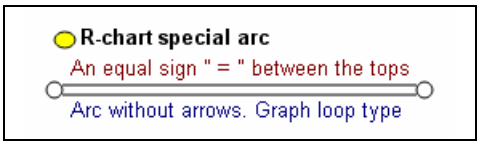


Figure 10. Arc without arrows to record graph loop type.

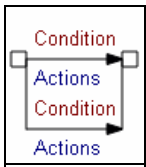
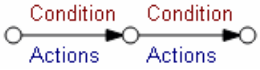
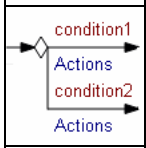
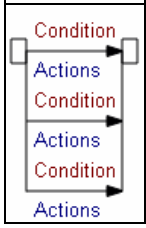
	**Small square** – vertical writing of arc sequence between square vertices, which is equivalent to the following arc sequence: 
	**Small diamond** – wait for execution of exogenous event (state of counter, interrupt system trigger, timer etc.). Conditions for wait interrupt are shown on the arcs outgoing from the small diamond
	**Vertical rectangle** – simultaneous sequence execution (on the 3-leg diagram) of arcs between the rectangles

Figure 11. Examples for special use of tops

of arcs outgoing branches, 3D and multi-dimensional image of R-charts, connection of linguistic processor with special interpretation of records on arcs of R-charts (in the form of grammar), see Fig. 3.2, etc. Click twice by left mouse button in order to pass to special configuration of top or arc.

For many applications: calculation of 3D trajectories, parameter representation of multi-dimensional space surfaces, linear programming, etc. it is convenient to pass to multi-dimensional R-charts with the top in the form of special parallelogram, see Fig. 12.

VTR widely uses the color to mark the tops, arcs and records on arcs, see Fig. 13. Usually the ellipse before the name of each new R-chart is marked by *yellow* color. *Brown* color is used to record Conditions; *blue* color – to record Actions. *Red* color on Fig. 13 marks the route of arcs for automatic generation of test; *green* color – tops, where the program activity interrupts, etc. are allowed.

One of the most powerful capabilities of graphical programming, which is absent in traditional programming, is the use of forms of graphs: R and R*. All captions on arcs (details of execution) from R-chart are deleted in R*-chart, and hereby its (R*) compactness is significantly increased; see Fig. 2, 13 and 14-15 below. For example, R*-program on Fig. 13 is **by 65 times more compact** than its traditional record in C++ language. Such record allows considering a program, its logic and structure as if from the top – "bird's eye panorama". It helps a programmer defining the strategy for continuation of works and discussing a program without extra details of execution, connecting them (details of execution) as soon as necessary.

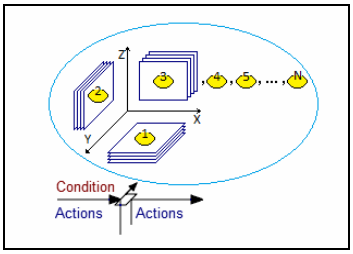


Figure 12. Principle for organization of 3D and multi-dimensional calculations using R-charts.

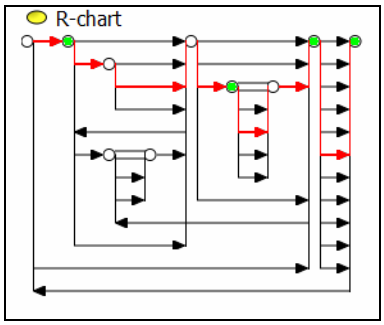


Figure 13. Use of color to record R-chart

V. REALISTIC EXAMPLE

Fig. 14,15 below contain R, R*-charts of realistic program in Delphi language, designed to process a certain class of special files [4]. R-chart of example demonstrates the self-descriptiveness of new generation technology. The first R-chart contains the architecture (directory comment) of the whole program and procedure for its definition. The first column has its name and characteristics: **type=class.** The second column defines the data (**//fields**), etc. All methods (names) in program are saved in Graphical development environment – GIDE and further set the succession (network graph) for their definition. Traditional programming does not have such capabilities.

The traditional record of this program in Delphi language [4] p.17-19**:** takes 2.5 pages, 121 lines, 2080 symbols without comments. The graphics program (Fig. 13) contains 24 (by **5.3** times less) equivalents of text lines in this article, 1220 (by **1.7** times less) symbols and **37** horizontal arcs, for which input it is required by **23.2** times of key presses less. As a result **860** (**41.3%**) symbols, which became extra (hash) for graphical record of R-charts, were deleted from traditional record of this program.

Using only one chart (arc) the graphical programming for the first time can draw the structure (idea, architecture) of software project, without details of execution, Fig. 15. Such record is **by 45.1** times less than traditional record of this program in Delphi language.

The larger the realistic program is, the more the advantage of compact graphical (R*) program is, i.e. by 50, 100 and more times. Compact R*-chart simplifies the development, protection and demonstration of project. For the first time it allows starting the development from idea, architecture, project sketch, gradually increasing and designing such basis (skeleton) with details and capabilities of execution. The work may be held in any concept (OOP, COP, etc.) and in any language, from which only the simplest, linear objects and concepts for type of function, procedure, expressions, formulas, structures of data, etc., established in classical mathematics, are used. The input of such program into computer is carried out by **15.4** times quicker. For example, it is necessary **in total 37** (number of horizontal arcs) presses of mouse or keyboard buttons to input a program on Fig. 15 and 13.

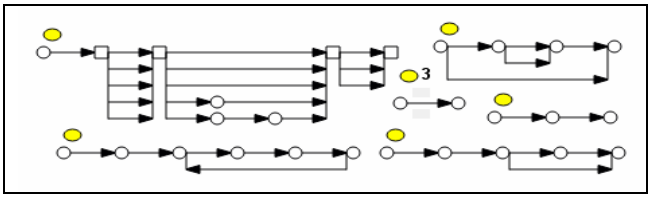


Fig. 15. R*-chart of the same program (Fig. 14) without details of execution is **by 45.1 times more compact**.

VI. VTR GRAPHICAL PARADIGM

The first program was written more than 65 years ago. It is a short period for evolution in formation of fundamental programming principles.

The essential strategy for development of traditional programming is reduced to constant *increase* (and complication) of basis: command, statement, function, module, object, class, environment, etc. Each new step is accidental, unique and is based on “bare empiricism”, possibilities to finance (to advertise) an author or firm. Therefore, “Babel tower” of programming languages and styles, which separates the programmers and does not unite their achievements, is constantly increased in this model of development.

Right from the start programming was based on fundamental principles of mathematics that have the history of development for many centuries. Programming derived the main principles to record expressions, formulas, functions, data descriptions, etc. from mathematics. As a rule these records provide with comprehensibility (visualization) of programming languages and reliable record of programs without errors. The main difficulties and problems in programming arise from use of special language statements of **goto, if, for, while** type, etc., which do not have any direct analogs in classical mathematics.

E. Dextra in his time suggested the structural program design without **goto.** In opinion of many experts – it is the most efficient offer in programming technology until now.

The contrary situation is in the new generation technology: basis *is decreased* to the engine (to one arc), to the essence of programming process and human thinking (we can say, to Higgs boson – “God's particles” in programming), which is understood by a first-time programmer and specialist, who is not engaged into programming, and by a schoolboy, providing everyone with the real strategy to build the second (computer) competence.

New generation VPT **excludes** the branch, exit and cycle statements of **goto, for, while** type, as well as **labels** and brackets of **begin-end, {-}** type, etc. For today they are rather 1) oriented to computer (to set commands to it) than to a human (to provide with convenience and quality in process of his/her thinking process while executing his/her concrete task); 2) they are primitive due to their capabilities, 3) they have the complex syntax, 4) they use a great number of extra symbols-parasites of **';'** type, **labels,** keywords of **then**, **else**, **do**, **continue**,**...** type, brackets of **begin-end**, **{-}** type, etc. 5) they do not allow documenting the thoughts, motivation of programmer’s actions at development of his/her program, within this purpose it became necessary to use the second additional language of diagrams UML, and finally, 6) they require to hash-up conditions of task, which should be programmed, for themselves. Therefore, they complicate and confuse, generally speaking, the simple programming process, create the situation when “one cannot see the forest behind the trees”, one cannot see the idea of algorithm and program. Due to this reason these symbol

statements **goto**, **if**, **for,** etc. are completely excluded from new generation Graphical programming technology, instead of them it is suggested to use the theory of graphs, loaded along the arcs, from classical mathematics.

As a result, programming acquires the single (one and only!) graphical equivalent (engine, graphical shell) for all programming languages). **A tool (graphics engine) in new programming is developed, adapted to the task, being solved, and its executors and the task is not transformed to a tool (language statements) as in traditional programming. It significantly simplifies and solves the existing problems of traditional programming, including OOP.**

VII. SEVEN ADVANTAGES

1. Simplicity. Only one horizontal arc is in the foundation (basis) of VTR graphical shell. But this arc is more powerful than traditional statements of programming languages. You do not need any keyword to set it; it is easier and simpler to input it into computer. The description of VTR essence takes one page. VTR graphical shell is applicable to all languages and in the whole life cycle of program. As the graphical shell is very simple, its translation is very efficient in terms of space and running speed of result codes. It also allows performing the efficient hardware implementation.

2. Compactness. The compactness is from 3 to 100 and more times better in comparison with Traditional program record. The larger a software project is, the higher the compactness is. The main advantage of this form – one can see the chart of the whole project at once, it helps to understand (to tell, to discuss, to protect, to develop, to add, to approve, to resend, etc.) the project and it turned out to be very important in professional programming.

3. Visualization. A program, its structure, architecture, idea of algorithm is covered at one sight. For the first time a programming language (R-chart) provides with *self-determination* of what it describes. All keywords-parasites for determination of language statements are *excluded* from programming and they correspond to 40-50% of all program symbols. All these capabilities significantly simplify understanding a program and process of its development, which *automatically arises* from the first steps of program definition.

4. Data definition. R-charts provide with conceptual unity, mathematical rigor and graphical visualization of data storage and processing systems, improving their intellectual component. The graphical record of descriptive part in modern programming languages is by 1.5-2 times more compact and incomparably more visual. The capability for visual description is provided for large data storages in graphs of their structure (catalogues, metadata) with flexible, quick mechanism to address to them in order to transfer to processing into information systems, which have the same graphical R-charts for processing of these data.

5. Power. Theoretically R-chart is equivalent to Turing machine. It means that one arc of R-chart is enough to record any algorithm. Practically any programming language may have the mentioned simplest graphical shell, uniform (!) for all languages. It means that all that may be recorded in these languages, including OOP, can be written more efficiently in their graphical shell, connecting more powerful visual apparatus of human associative thinking to programming.

A tool (R-chart) in the new graphical paradigm is adapted, developed for the task, being solved, and its executors but the task is not transformed to existing fixed language statements (tool) as it happens now in traditional programming. It erases many problems in modern programming and opens the road to evidence-based programming and accumulation of professional experience by programmers. The great advantage of the new graphical shell, which is absent in traditional programming, is that it provides with the single semiotic system for Customer, Programmer, Data Manager and Software Products Commercial Design Manager.

6. Perspectives. VTR is especially efficient for solution of logically difficult (complicated) tasks and hereby it improves and develops modern programming, including OOP. It just sets the evidence-based, 3D and multi-dimensional graphical programming. Parallel, linguistic and other special processors, which are programmed by single language of R-charts, are rapidly activated during execution of program by natural way. The processes to design and to dedicate the classes of objects are simplified. Program checkout process becomes evidence-based. The simplification of basis simplifies its add-in – visual programming environment, which for the first time may be personalized to a user with memorization, accumulation and development of his/her intellect. The new perspectives of distributed internet-technologies for program development are opened.

7. Succession. One does not need to break anything and to make “by another way”. All that exists in programming remains and only the graphical shell is added to what we have. The professional development of graphical shell “to what we have” can be made during one month (in the future this process will be automated). A user has the right of choice how to work, using the graphical shell, or by the way he/she used to do. He/she will widely use the experience from previous generations of programmers in the graphical shell. It means that he/she can use *linear* structures of programming languages on arcs of R-charts (procedures, functions, descriptions, expressions, etc.) that came to programming from classical mathematics, in their usual form and therefore they are better protected against errors.

VIII. CONCLUSION

It is necessary to start from foundation in order to construct a good building. What we offer – it is the new foundation for programming, which provides with another, simpler and natural (human) beauty for development of the whole buildup. We were based on the fact that one cannot construct a good building, all time compensating (neutralizing) the disadvantages of its foundation as it is made now in programming in IDE, SOA, etc. systems. The existing foundation has not practically been changed since the end of the 40-ies when the first programs were written in Assembler and Fortran. Its main disadvantage is the machine-based, operational, complex and primitive (low-powered) organization. It has become obsolete for more than 65 years and does not correspond to the tasks of time.

The foundation is more human (not machine-oriented), simple and powerful. It is simply defined on Fig. 1, 2 and by text that takes ½ of page in section II, the rest – illustrations of advantages and comparison with what is known. Everything is made according to A. Einstein: "do as simple as you can but not simpler than it is" because there is nothing simpler and more efficient until now yet. As a result the culture and efficiency of programming is sharply improved by some times. Programming has been simplified and became available to kids, schoolchildren and specialists-non-programmers. We hope that even "blessed trinity [5] as a standard of modern professional complexity and beauty (SOAP, WSDL and UDDI)+SOA" will only win from it – it will be simpler and understandable to people and programmers.

REFERENCES

[1] V.M. Glushkov and I.V. Velbitskiy, "Programming technology and problems of its automation", *USIM*, Kyiv, no. 6, pp. 75-93, 1976.

[2] I.V. Velbitskiy, "Programming technology", *Technika*, Kyiv, p. 279, 1984.

[3] *Information technology, Program constructs and convention for their Representation*, International standart ISO/IEC 8631, Second edition 1989 Geneva 20, Switzerland, ISO/IEC Copyright Office, p.7, 1989.

[4] A.N. Valvachev etc. "Programming in Delphi. Textbook. Chapter 3 Object-oriented programming", *INTERNET*, p. 19, 2005.

[5] Leonid Cherniak, "SOA – step beyond the horizon. Open systems: IT management", *Open systems #09/2013*, p.12 Article in a journal:

[6] W.K. McHenry. "Technology: A soviet visual programming", Journal *of Visual Languages and Computing*, vol.1, no. 2, pp. 199-212, 1990.

Article in a conference proceedings:

[7] I.V. Velbitskiy, "Next generation visual programming technology with R-charts", *Plenary report, MEDIAS-2012. Dedicated to 100 anniversary of Alan Turing (IEEE)*, Cyprus, pp. xiv-xxxiv, 2012.

[8] L.F. Drobushevich, "Common use of UML and R-chart notations in the training process for software system development methods", *MEDIAS*-2010, Cyprus, pp.73-77, 2010.

[9] I.V. Velbitskiy, " Graphical Programming and Program Correctness Proof", *Plenary report, 9th IEEE Computer Science & Information Technologiens Conference*, Yerevan, Armenia, 23-27 Sept.2013, IEEEXplore,CSIT-20

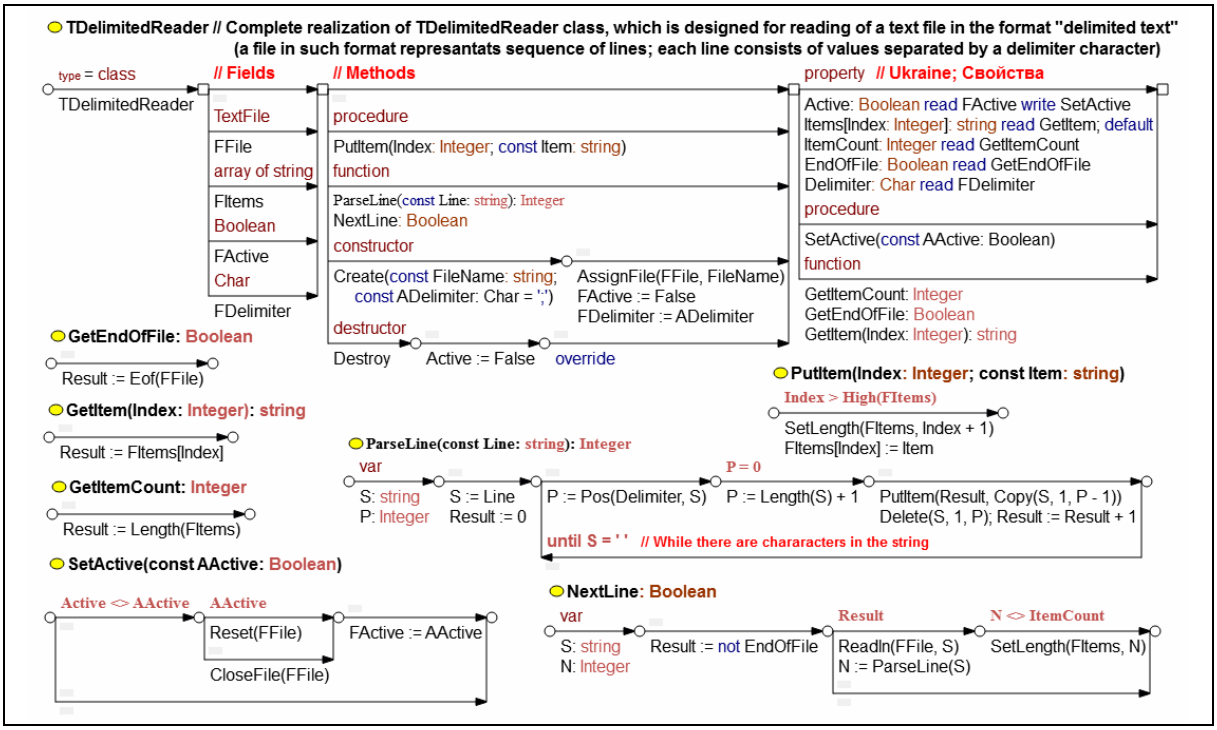


Figure 14. Graphical R-chart **is by 5.6 times more compact** and is incomparably more visual than the record of program by traditional in Delphi [4], which is removed from the **860 (41,3%)** of unnecessary characters.