Next generation visual programming technology

Velbitskiy I.V.
The Glushkov's Fund, Kyiv, Ukraine
ivelbit@gmail.com

Abstract

The next generation visual programming technology, which uses the so-called R-charts – graphs loaded through the arcs (VTR), is defined. R-charts have the standard ISO/IEC 8631. Main difference of VTR is simplicity and single graphic form for program writing during the entire life cycle, which ensures new principles of design, debugging, visual and compact (by more than one order) recording compared to writing in the existing programming languages (PL). Graphic paradigm of VTR is set including the basis single graphic shell of the existing PL. Three- and multidimensional programming is introduced. It is integrated well with available programming systems. New technology is accessible for wide group of specialists, and not only for the programmers. At present, Graphics Editor, R-chart translator in C++ and R-chart design system are realized in medium Qt-Criotor C++. This allowed carrying out the reasonable analysis of VTR advantages compared to traditional technologies using the modern PL, UML and OOP.

Introduction

For the first time we have introduced the concept of programming technology and orientation to industrial principles of development of avionics software for rocket and space systems of R-36M (SS-18 SATAN) type (of former USSR) at the end of 60s of the last century. The so-called R-technology of programming of broad use has been designed in the result of generalization of these works [1-4]. This technology was well-known in the USSR and CMEA states-members (more than 600 works were published related to various scopes of its application).

Now, twenty years later, specifically after 2009 the work related to visual R-technology was resumed. It was analysed and compared to what we have in programming at present. Consequently, a conclusion was made that the new concept of R-technology did not get out of date, but fits in with modern trends of programming development by forming next generation of visual programming technology with R-charts (VTR), which improves present approaches to programming by more than one order. A core of the discussed concept is that the R-chart is not a new (one more) PL, but **a graphic shell - single (!) for all** the known languages. R-technology is viewed not instead of, but **along** with everything that exists in modern programming by adding it with new and appealing features.

Determination of VTR basis

It is proposed not to write, but to draw the programs during their entire life cycle in the form of graphs consisting only of horizontal and vertical lines, Fig. 1. Such graphs are entered easily in the computer using a mouse only (or keyboard). In order to enter a graph (Fig. 1) consisting of 15 arcs, it is necessary to click 8 (n-1) times with the left mouse button, where n – number of horizontal graph arcs. The vertical arcs act as auxiliary on these graphs and connect a graph's nodes with the horizontal arcs, which are being main and loaded with information.

A node does not have the name and it assigns of the program state or the process of its development. Any number of arcs outgoing to different directions – to the right and/or to the left, may be connected to one node. The outgoing arcs at each node are being visible (read, understood, analysed, and executed) sequentially downward and from one node to another – by the appropriate arrow of the arc starting from the first graph's node on the left and to the last node on the right.

Condition of propagation along the arc is written on the arc top, and below – the Actions performed during this process. No limitations are placed on conditions and actions writing – they can be written in any language: Russian, English, Armenian, mathematical, programmer's, etc. in one or several lines. If Condition above the arc is true, then the Actions written below the arc are executed, and a move along the arc's arrow to the new state (node) is performed. If symbol "#" is written near the arc's arrow (Fig. 2.2), a move is made to condition behind a node to which such arrow is pointed to. The arc, which does not contain a Condition, is always true. Condition, which begins with the PL key word, is being always a true *logical constant of* R-chart (Fig. 3) executed in special way stipulated for determination of graphical shell. As a rule, logical constant is joined to its Action forming a single resultant code. However, more complicated Actions can be observed (see Fig. 3, on the right). If Condition is false, the Actions below the arc are not performed and the next outgoing downward arc is considered, and if it is not available, then the arc which follows a node to which it is pointed to (the arc with the last false condition). Each R-chart has its own name in the project, which is written near yellow ellipse (Fig. 2.3, 2.4 and 4). Arbitrary texts

on the R-chart's arcs (Fig. 4 and 2.4) can be selected with the right mouse button (a dotted frame, see Fig .4) and defined by another R-chart formed according to the command of VTR environment. Such graph is named as the R-chart (ISO/IEC 8631). Theoretically, the R-charts are equal to the Turing machine.

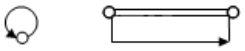
Examples of R-charts recording

R-chart recording of selection and loop statement and appropriate its record in C++ is given in Fig. 5. The R-chart is incomparably more visual, two-fold compact, and it is entered into computer six-fold quicker. The VTR's user does not see the right part of Fig. 5, which is given here as a formal definition (explanation) of R-chart in the symbols traditional for the readers. Recording of traditional loop statements (Fig. 6) in the R-charts is more visual and powerful if for no other reason than the R-chart arcs are loaded just partially for their recording – some of the arcs have neither conditions nor actions, or both.

Comparison of R-charts with other well-known graphical methods for algorithm writing of UML type, Flow Chart etc. is given in Fig. 7. The R-chart is the only one of graphical methods that is used along the entire life cycle of the programmes. It is more compact among the all known methods.

Basis enlargement

The VTR allows flexible development of drawing tools during the operational process. For example, *special* double arc without arrows (which looks like as an equal sign that connects two nodes) is used to represent a graph of loop type. This makes it possible to use only horizontal and vertical lines for such graph presentation:



Any information in any language can be written above or under the arc the same as on ordinary arc with an arrow. The name of the special arc and the corresponding construction are written above, and the Actions – below. If no any records are on the special arc (see Fig. 6.1.2), then this will correspond (on default) to graphical recording of traditional loop statement **while**.

The R-chart's nodes may have *special configuration* – small square, small diamond, rectangle, etc., see Fig. 8. It denotes a special condition of visual technology: parallel execution of outgoing arc branches, multidimensional representation of R-charts, connection of linguistic processor with special (in the form of grammar) interpretation of record on the R-chart's arcs, etc. It is needed to click twice on the left mouse button to move to the special configuration of the arc or node.

For many applications such as linear programming, 3D trajectory calculation, etc., it will be convenient to employ multidimensional R-charts: **1, 2, …, N**. This could be done in the VTR using a node of shape - a parallelogram, see Fig. 9.

The colour is used extensively in VTR to highlight nodes, arcs and records on the arcs, see Fig. 10. Red colour (Fig. 10) demonstrates a route of the arcs for test generation; and Green colour is used for marking of the nodes where program operation interruptions are permitted, etc.

Ability of *compact recording* when all inscriptions on the arcs are deleted from the program using one command is being one of the most powerful and newest possibilities of R-chart, see Fig. 1 and 10. For example, program writing given in Fig. 10 fits the scale M1:25 compared to its writing in C++ language. This allows seeing the program, its logic and structure like from the top. Such a view to the program helps him/her to define strategy of its design, to choose the point to continue its development, to determine an angle for demonstration of program operation without superfluous realization details, etc.

Figure 11 demonstrates a fragment for determination of TDelimit class in Delphi, which describes the objects for reading of elements divided with some symbols from the text file. R-chart ensured visualization and compactness for description of this OOP fragment. The *structure* of the defined class became visible (clear) immediately. Possibility to fix visually the definition structure for algorithm, program, and object – is being a new and important concept of graphical paradigm.

VTR graphical paradigm

The strategy of traditional programming development is to come to continuous *increase* (and complication) of the basis: a command, statement, function, module, PL, object, class, environment, etc. Every new step is random, unique, and it is based on "sheer empiricism", financial (advertising) capacities of the author or the company. Therefore, the "Tower of Babylon" of programming languages and styles are growing continuously in this model and it separates the programmers instead of uniting their achievements.

On the contrary, in the next generation technology the basis is *decreased* to the kernel (to one arc), to the essence of programming (it is possible to say that to Higgs boson – "the God's particle" in programming), which is understandable to each programmer and a specialist that not involved in programming as well as to a schoolchild, thus, providing real strategy of second (computer) literacy build. Hence, programming acquires single (the only one!) graphical equivalent (a core, graphical shell) for all the programming languages. A tool (graphics kernel) is adjusted and developed to the problem to be solved and to its

executors, instead of the problem to be transformed for a tool as it is used in traditional programming.

Key filler-words and the corresponding language structures such as **go to, if, for, while, break, labels, begin-end, {-}** etc. with strict and fixed syntax writing **are excluded** from programming. These structures are being the main source of errors and problems in modern programming. They are replaced in VTR with more powerful (!) and compact graphical R-charts without any key words. All information on the program is positioned visually on three parts: above the arc (Condition), below the arc (Actions) and continuation along the arc arrow. This simplifies understanding of the program and its development process, including OOP.

Conclusion

At present, VTR is the only programming concept, which satisfies the principle of the great physicist A. Einstein: "Everything should be made as simple as possible, but not simpler", because nothing simpler and more effective exists now. The following is considered as the basic advantages of the next generation VTR technology: Simplicity, Compactness, Visualization, Capacity, Prospects of improvement and development, Succession that is already there. This erases the problems of modern programming and opens the door to demonstrative programming and accumulation of professional experience.

References

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

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

[3] McHenry William R-Technology: A Soviet Visual Programming, Journal of Visual Languages and Computing Vol 1,#2, 1990. – pp.199-212.

[4] I.V. Velbitskiy, Next generation visual programming technology with R-charts. Plenary report, MEDIAS-2012. Dedicated to 100 anniversary of Alan Turing, Cyprus – 2012, p. xiv-xxxiv.

Figures for Paper

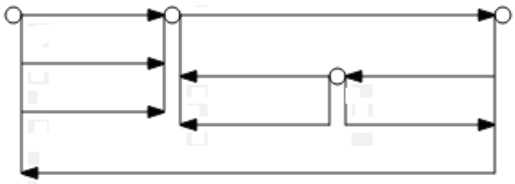


Fig. 1. Program or its chart in the VTR

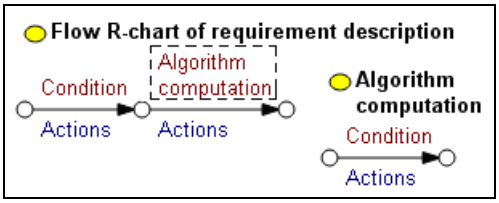


Fig.4. Connection of arbitrary texts with their defenitions in the R-charts

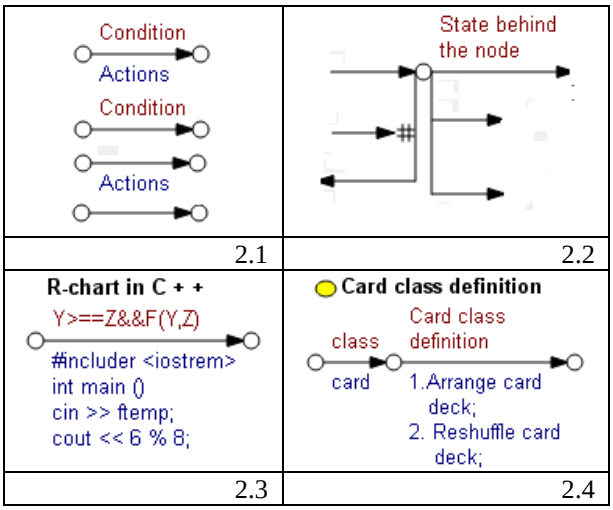


Fig. 2. Examples of writing on the graph arcs

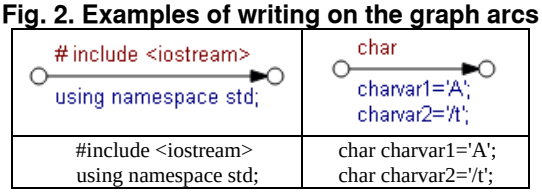


Fig. 3. Example of writing of R-chart's logical constants and codes C++ under them

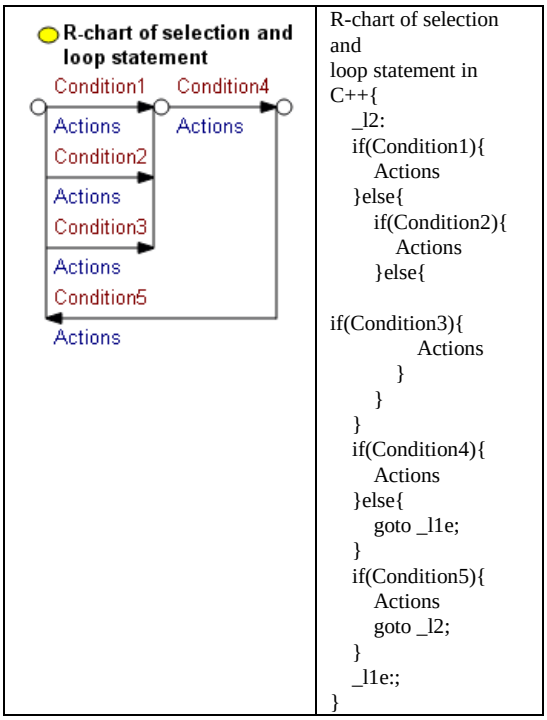


Fig. 5. Selection and loop statements recording in R-chart and C++

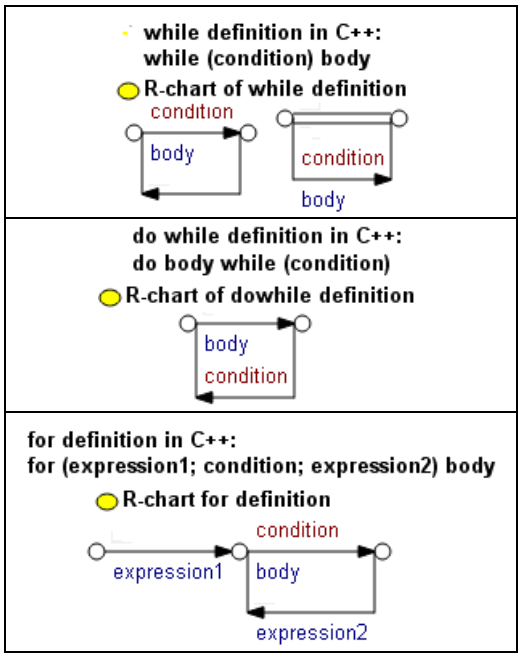


Fig. 6 Recording of loop statement definitions in C++ and R-charts

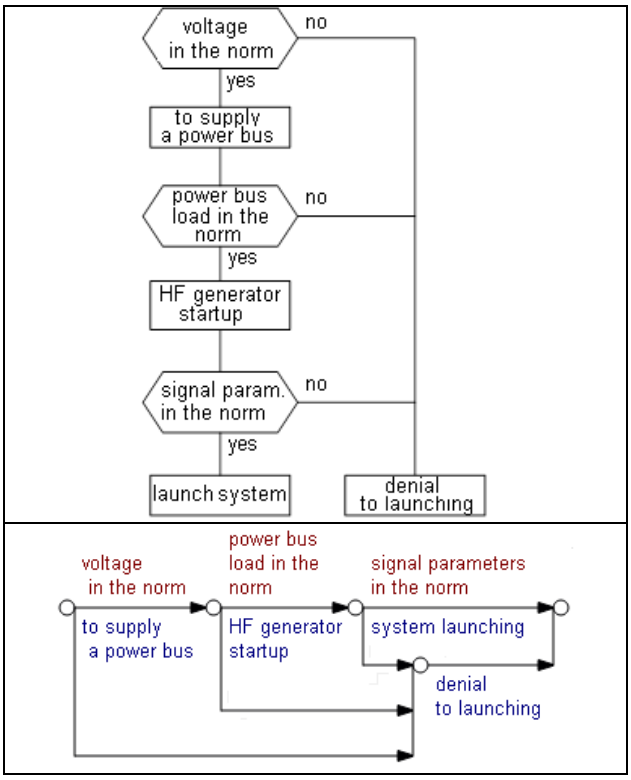


Fig. 7 Example of the fragment of prelaunch procedure for space shuttle "BURAN" written in UML (on the top) and in the R-chart

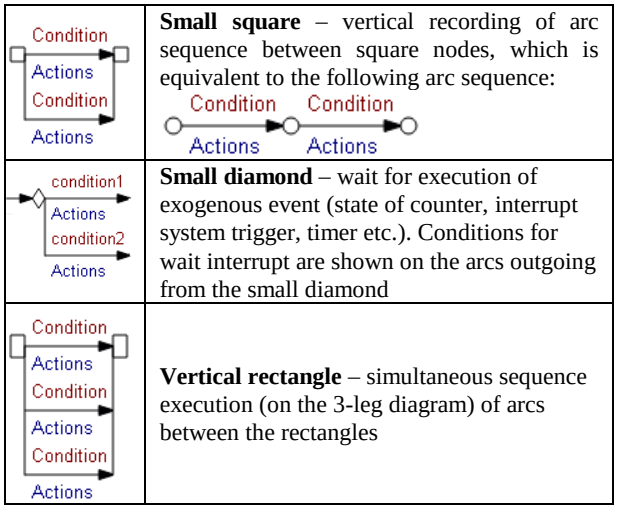


Fig. 8. Special use of R-chart nodes

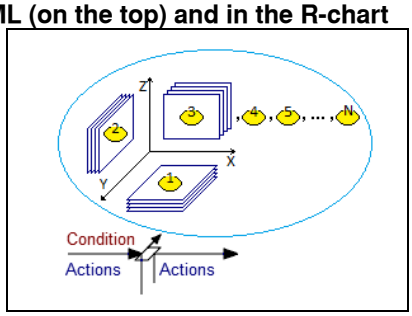


Fig. 9. Organization principle for three- and multidimensional calculations with R-charts

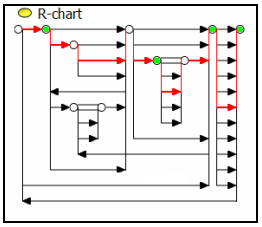


Fig. 10 Use of colours for graphical R-chart recording

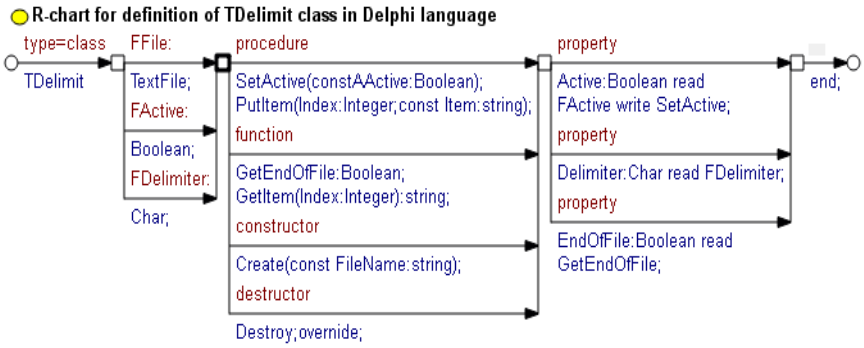


Fig. 11 R-chart for definition of some OOP class in Delphi