Graphical Programming and Program Correctness Proof

Igor V. Velbitskiy

The Glushkov's Fund,
2, Mikhailivska str., of. 17, Ukraine, 01001Kiev
ivelbit@gmail.com

***Abstract*— The next generation visual programming technology (VTR), which uses so-called R-charts (ISO/IEC 8631) is defined here. R-charts – are being the oriented graphs loaded through the arcs. The main difference of VTR is its simplicity and single graphic form of the program writing during its entire life cycle, which ensures new principles of design, debugging, visual and compact (by more than one order) writing compared to writing in existing programming languages (PL). Due to its simplicity, it is acceptable to broad circle of the specialists, but not only to the programmers. Graphic paradigm of VTR is set, which is based on a single graphical shell for all existing PLs without goto, if, for, while, break, labels, brackets: begin-end, {-} etc. Evidentiary, linguistic, parallel, three- and multidimensional programming is introduced for controlling and computing systems. Colour is used extensively for writing of graphic programs. It is integrated well with available programming systems. At present, graphic programming VTR is realized in the environment Qt Creator C++ (www.glushkov.org). This allowed carrying out justified analysis of VTR advantages in comparison with traditional technologies that use modern PLs, UML, and OOP.**

***Keywords*— use only of graphs during the entire program life cycle; programming without goto, if, for, while, labels, brackets begin-end, {-} etc.; use of colour; programming of special processors – linguistic, control, parallel, three-dimensional, multi-dimensional etc.; program correctness proof**

I. Introduction (history and advantages)

At the end of 60s' of the last century we introduced for the first time the concept of programming technology and orientation to industrial principles for development of avionics software for rocket and space systems of type R-36M (SS-18 SATAN) (of former USSR). So-called R-technology of programming of broad use has been designed in a result of generalization of these works [1-4]. This technology was well-known in the USSR and CMEA states-members. Three (3) all-Union conferences and one (1) international conference had been held dedicated to the R–technology issues as well as tens of special workshops, and more than 600 works were published related to various fields of its application. Before the breakup of the Soviet Union and CMEA, R-technology won the technology contest in the Complex Program CP STP CMEA for joint development of single CMEA Technology in all 10 states of the Commonwealth. It received the international standard ISO 8631 in 1988 [3]. In 1989, the American experts analysed this technology and evaluated as high [4].

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. The concept was reviewed with consideration of technological expansion, introduction of new languages and environments. New concept was realized on the modern platforms and discussed at the international conferences [5-7].

Consequently, a conclusion was made that new concept of R-technology did not get out of date, but fits in with modern trends of programming development by forming the 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 **graphical shell - single (!) for** all the known languages. R-technology is viewed not instead of, but **along** [6] with everything available in modern programming by adding it with new and appealing features. The higher level of modern programming – OOP, AOP, UML etc. the more will be effectiveness of VTR and its graphical shell. The following is considered as the basic advantages of the next generation VTR technology:

Simplicity. Just one horizontal arc directed to the left or to the right is in the basis of VTR's graphical shell. However, this arc is more powerful compared to traditional PL statements. No any key words are required to create it, and its entering into the computer is done quicker. Description of VTR's core takes just one page. VTR's graphical shell is applicable to all the languages and during the entire life cycle of the programs. Since the graphical shell is very simple, its translation is quite effective as to memory space and operation speed of resultant codes. Simplification of the programming graphical shell basis allows simplifying hardware architecture of the computers (chips).

Compactness. Graphic writing of the programs is more compact compared to traditional writing and it fits better to information presentation (horizontal) acceptable in classical mathematics. There is a chance (for the first time) to increase compactness of writing **by more than one order** providing a view on the whole programming project like from the top – a bird's eye view.

Visualization. The program, its structure, architecture, and algorithm idea are gripped at a glance. Key filler-words and corresponding language structures such as **goto, if, for, while, break, 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 in 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.

Capacity. Theoretically, R-chart is equal to the Turing machine. It means that one reduced arc of the R-chart is enough to write any algorithm.

In practice any PL may have the described simple graphical shell that is being single(!) for all the languages. It means that everything that may be written in these languages, including OOP, can be written more effectively in their graphical shell with involvement of more powerful visual apparatus of human associative thinking to the programming.

In the new graphical paradigm, a tool (R-chart) is adjusted, developed to the problem to be solved and to its executors, but not a problem is transformed to the existing and fixed language statements (tool) as it happens now in traditional programming. This erases many problems of modern programming and opens the door to proof-programming and accumulation of professional experience.

Prospects. VTR is effective specifically for solution of complex (convoluted) logical problems and thereby it improves and develops modern programming, including OOP. It just creates a conclusive, three-, and multi-dimensional graphical programming. Parallel, linguistic and other special processors programmed with single R-chart language are turned on naturally and promptly during program execution. Design processes and extraction of object classes are simplified. Algorithm design processes are automated beginning from their informal writing in the originating documents. Program debugging process becomes demonstrable. Simplification of the basis reduces also its add-on – visual programming environment that can be a user-customized for the first time with memorization, accumulation and development of its intellect. New prospects of the distributed Internet-technologies are revealed.

Continuity. No needs to break anything and to make differently. Everything that exists in the programming will remain and just a graphical shell is added. Professional development of a graphical shell "for everything that is available already" can be done within one month (this process will be automated in the future). A user has the right of choice how he/she can work: to use a graphical shell, or to continue to operate as he/she got used to. The user will make use of experience of previous generation of the programmers as to graphical shell. It means that on the R-chart arcs the user can employ *the linear* structures of programming languages (procedures, functions, descriptions, expressions etc.) adopted in programming from the classical mathematics in the habitual form and, consequently, better protected from the errors.

II. DETERMINATION OF VTR BASIS

proposed not to write, but to draw the programs durinheir entire life cycle in the form of graphs consisting only of horizontal and vertical lines, see Fig. 1.

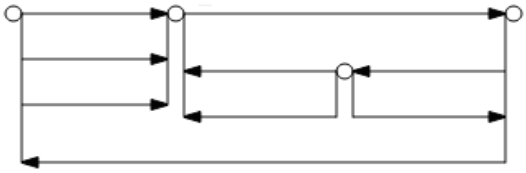


Fig. 1 Program or its scheme writing in the VTR

Such graphs are entered easily in the computer using a mouse only (there is an equivalent and effective procedure in place as to tracking through the vertices of R-chart and drawing a graph using the keyboard only). In order to enter a graph (Fig. 1) consisting of 15 arcs, it is necessary to click 8 times (n-1) 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 vertices with horizontal arcs, which are being main and loaded with information.

A vertex does not have the name and it sets state of the program 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 vertex. The outgoing arcs at each vertex are visible (read, understood, analysed, and executed) sequentially downward and from one vertex to another vertex – by the appropriate arrow of the arc beginning from the first graph's vertex on the left and to the last vertex on the right.

The VTR basis contains only one type of horizontal arc with direction to the right or to the left (Fig. 2). Condition of propagation along the arc

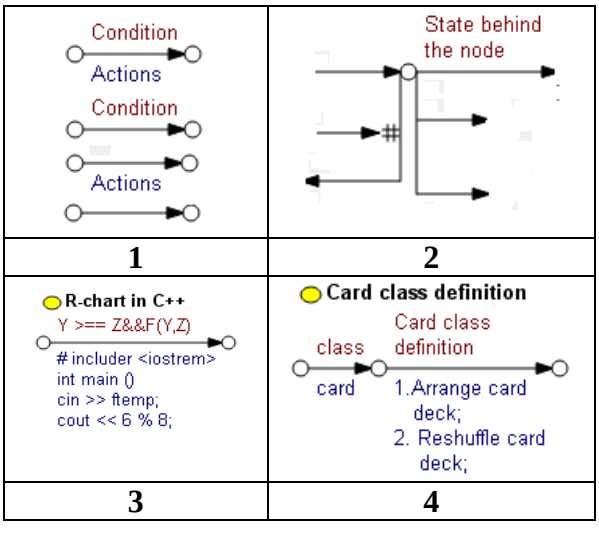


Fig. 2 Examples of writing on the graph arcs

is written on the arc top, and below – the Actions performed during this process. No any limitations are placed for

conditions and actions writing – they can be written at any language: Russian, English, Chinese, 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 a new state (vertex) is performed. If symbol "#" is written near the arc's arrow (Fig. 2.2), a move is made to condition behind the vertex to which such arrow is pointed to. The arc, which does not contain a Condition, is true always. Condition, which begins with the PL key word, is being always a true *logical constant of* R-chart (Fig. 3) executed in special way set for determination of the graphical shell. As a rule, logical constant is joined to its Action forming single resultant code. However, more complicated Actions can be observed (see Fig. 3, on the right).

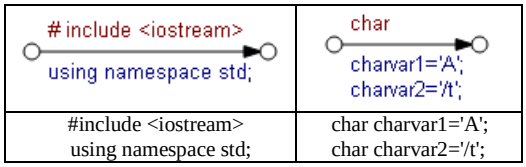


Fig. 3 Example of R-chart`s logical constants and codes C++ writing

If Condition is false, the Actions below the arc are not performed and the next outgoing downward arc is considered, but if it is not available, then the arc which follows a vertex 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 an R-chart (ISO/IEC 8631). Theoretically, the R-charts are equal to the Turing machine.

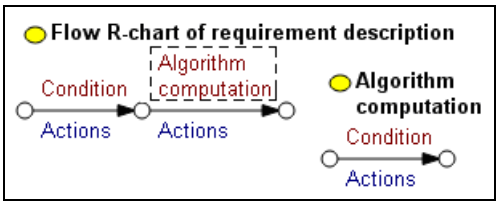


Fig. 4 Connection of arbitrary text with its definition in R-charts

III. EXAMPLES OF R-CHARTS WRITING

The writing of R-chart of selection and loop statement and its appropriate record in C++ language 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 traditional symbols for the readers.

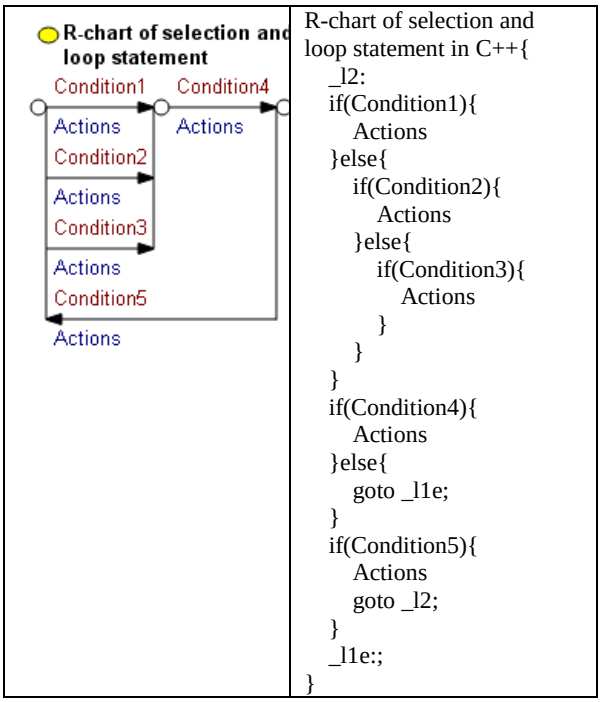


Fig. 5 Selection and loop statements written in R-charts and C++

Writing of traditional loop statements (Fig. 6) in R-charts is more visual and powerful if for no other reason than the R-chart arcs are loaded just partially for their writing – some of the arcs have neither conditions nor actions, or both.

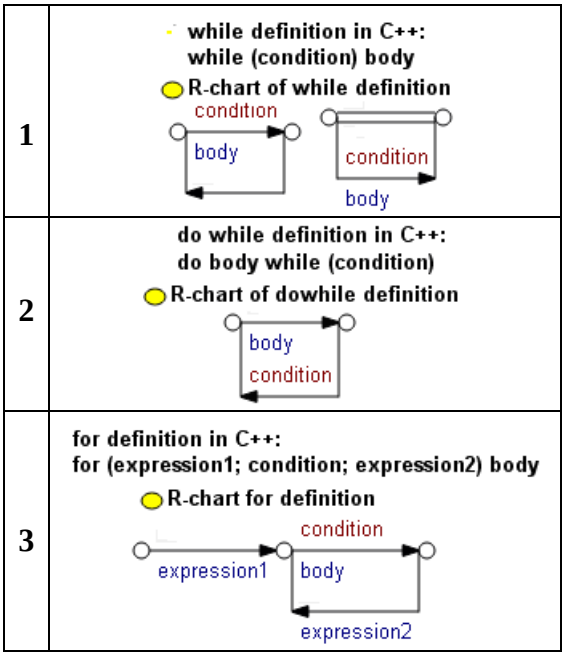


Fig. 6 Writing of loop statement definition in C++ and R-charts

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 programs. It is more compact among the all known methods.

Accordingly, in the first place VTR visualizes that part of existing traditional programming meshing, errors and difficulties in technologies that serves as the main source of program design, in particular, transitions, loops and branching. The linear parts of program writing such as expressions, functions, statements of code assignment, input and output, and other "safe" PL constructs remain unchanged in customary text format. However, the R-chart positions linear parts: Conditions of their use are written on the arc top, and an arrow assigns clearly the direction of work continuation. Such structure of linear statements ensures their visualization (and compactness) and continuity with existing programming systems.

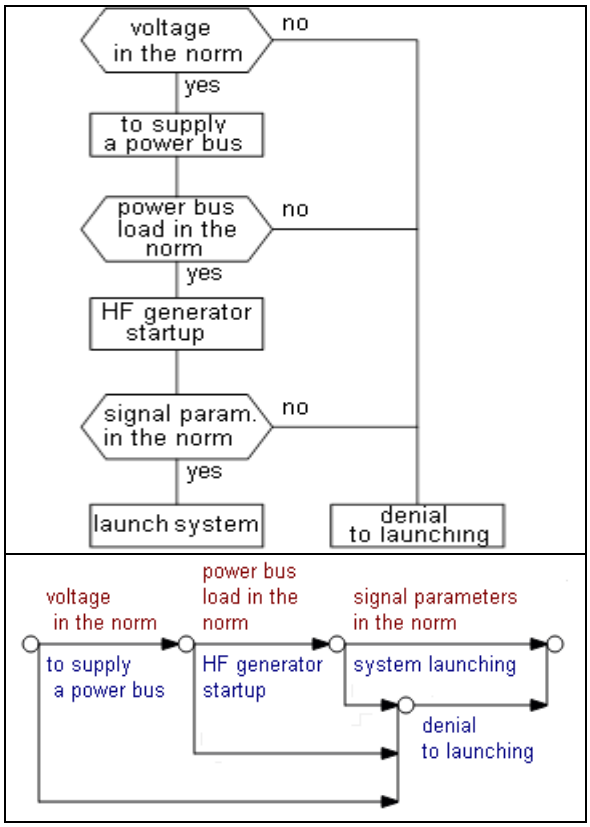
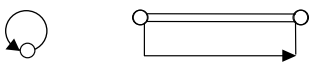


Fig. 7 Example of the fragment of prelaunch procedure for space shuttle "BURAN" written in UML (on the top) and in the R-chart

IV. BASIS ENLARGEMENT

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



Any information in any language can be written above or under special double arc the same as on ordinary arc with an arrow. The name of special arc and corresponding construct are written above, and the Actions – below, see Fig. 8. If no any records are on the special arc (see Fig. 8 and 6.1.2), then this will correspond (on default) to graphical writing of traditional loop statement **while**. Writing of a keyword on the special arc (Fig. 8, **for**) is useful for reverse translation from PL to R-charts as well as for generation of new frequently used graphic structures.

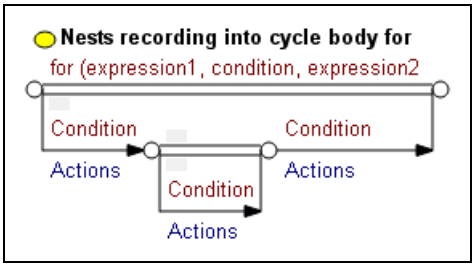


Fig. 8 Examples of writing on special arcs

The R-chart's vertices may have *special configuration* – small square, small diamond, triangle, rectangle etc., see Fig. 9. It denotes special condition of visual technology: parallel execution of outgoing arc branches, three- and multidimensional representation of R-charts, connection of

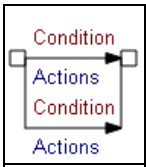
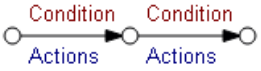
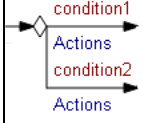
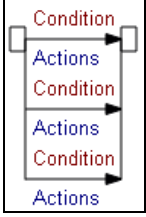
	**Small square** – vertical writing of arc sequence between square vertices, which is equivalent to the following arc sequence: 
	**Small diamond** – wait for execution of exogenous event (state of counter, interrupt system trigger, timer etc.). Conditions for wait interrupt are shown on the arcs outgoing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

Fig. 9 Special use of R-chart vertices

linguistic processor with special (in the form of grammar) interpretation of the records on the R-chart's arcs etc. It is needed to click twice on the left mouse button to move to special configuration of the arc or the vertex.

For many applications such as linear programming, 3D trajectory calculation, parametric representation of multidimensional sculpted surfaces, etc., it will be convenient to employ multidimensional R-charts: **1, 2, …, N**. This could be done in VTR using a vertex of specific shape - a parallelogram, see Fig. 10.

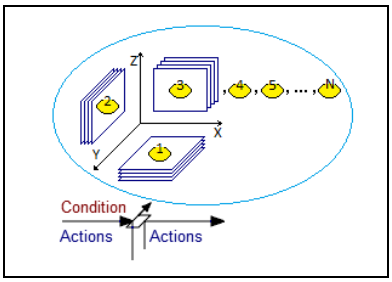


Fig. 10 Organization principle for three-and multidimensional calculations with R-charts

The colour is used extensively in VTR to highlight vertices, arcs and records on the arcs, see Fig. 11. Usually, *yellow* colour is used for marking an ellipse before the name of each new R-chart. *Brown* colour is used for writing of Conditions; *blue* – for writing of Actions; *red* colour (Fig. 11) demonstrates a route of the arcs for test generation; and *green* colour is used for marking the vertices

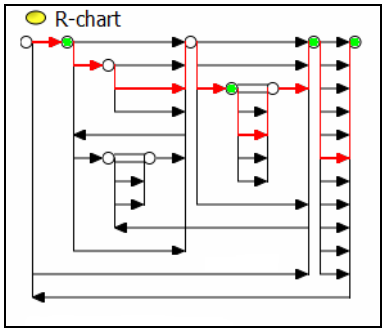


Fig. 11 Use of colours for graphical R-chart writing

where program operation interruptions are permitted etc.

Ability of compact writing when all inscriptions on the arcs are deleted from the program using one command on Control panel is being one of the most powerful and newest possibilities of R-chart, see Fig. 1 and 11. For example, program writing given in Fig. 11 fits to the scale M1:25 compared to its writing in C++ language. Such writing allows viewing the program, its logic and structure like from the top – from a bird's eye view. This helps a programmer to determine strategy of its development, selection of place for its development continuation as well as to define an angle of demonstration and discussion of the program without excessive details of realization etc.

A fragment for definition in TDelimit language of Delphi class that describes objects for reading of the elements from the text file is given Fig. 12. This example shows effectiveness of visualization in R-charts of "bad" (which does not demonstrate VTR advantages) linear part of definition, which does not contain complicated and convoluted logic. The R-chart ensures visualization (transparency) and compactness for description of this OOP fragment.

Possibility to fix visually the definition structure for algorithm, program, and object – is being new and important concept of the graphical paradigm.

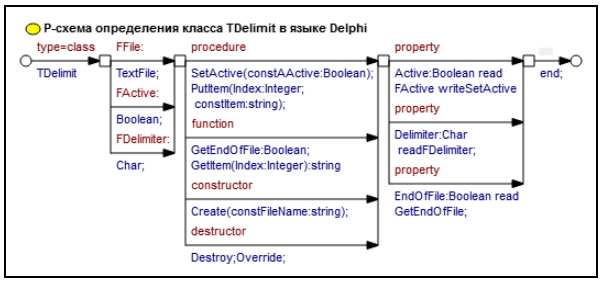


Fig. 12 R-chart for definition of some OOP class in Delphi

V. VTR GRAPHICAL PARADIGM

First program has been written a little over 65 years ago. This is quite short period for formation of fundamental programming principles.

The strategy of traditional programming development is come to continuous *increase* (and complication) of the basis: a command, statement, function, module, 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.

From the very beginning, programming was rest upon fundamental principles of the mathematics with the centuries-old history of development. Basic principles for writing of expressions, formulas, functions, data description etc. were transferred to programing from the mathematics. As a rule, these records ensure comprehensibility (visualization) of programming languages and reliable error-free program writing. Main difficulties and challenges in programming are related to use of specific statements of programming languages of type **goto, if, for, while** etc. that do not have direct analogy in classical mathematics.

At his time, Edsger Wybe Dijkstra proposed structured programming without **goto.** According to many experts, this is considered as the most effective proposal in programming technology up to date.

On the contrary, in the next generation technology the basis is *decreased* to the kernel (to one arc), to the essence of process of programming and human thought (it is possible to say that to Higgs boson – "the God's particle" in programming), which is understandable to each programmer and each specialist not involved in programming as well as to a schoolchild, thus providing real strategy of second (computer) literacy build.

The next generation VTR technology **excludes** statements of transition, selection and loop of type **goto**, **for, while** from programming and also **labels** and brackets of type **begin-end, {-}** etc. It is proposed to use the theory of oriented graphs loaded through the arcs from the classical mathematics instead of them.

Hence, programming acquires single (the only one!) graphical equivalent (a core, a graphical shell) for all programming languages. A tool (graphics kernel) is developed and adjusted to the problem to be solved and to its

executors, but not the problem is transformed under the tool (PL statements) as it is used in traditional programming. This simplifies considerably and solves existing problems of traditional programming, including OOP.

VI. DESIGN PROCESS AND PROGRAM CORRECTNESS PROOF

The VTR technology is distinguished uniquely with 1) automation of algorithm obtaining from informal record of problem statement or originating documents (ODs) for programming, and 2) program correctness proof. The main in the proof of program correctness is the system of axioms, which allows asserting about program correctness.

ODs for program construction and writing of formal process of program design from these documents are being the system of such axioms in the VTR. If ODs are changed continuously and rewritten in traditional programming, or it could be even worse (which happens often) – they are recorded nowhere and remain in a programmer's head to be written in the statements of one of the existing programming languages. Originating documents **remain always unchanged** in the VTR, but just a tool (R-chart) of program design process is modified. This is done through a "step by step from logic". At the very beginning, the logic is separated in the source text of ODs and the appropriate R-chart is written in optional symbols. Further, the logic of Conditions and Actions written in optional symbols in the R-chart is defined more exactly, or ODs are returned to the customer in order to carry out additional definition with specification of reasons for return in the customer's terms. In a result of such process, VTR neutralizes the specified drawbacks of ODs and on return it allows receiving more qualitative software (with proof) than in traditional programming.

Let's have a look at example of R-chart (algorithm) designing for evaluation of factorial of an integer in the VTR using a "step by step from logic" method. As a basis, we shall take definition of factorial of an integer given in the Encyclopaedia of Mathematics in the Internet:

> **Factorial of an integer *n*** (denoted by ***n*!**, it is read as *enn factorial*) — is the product of all positive integers less than or equal to **n**: **n! = 1x2x … x n.**
>
> The value **0! is 1** according to the convention for an empty product. Factorial is defined for non-negative integers only.
>
> This function is used often in combinatorics, theory of numbers and functional analysis.
>
> Sometimes an exclamation mark is named as a "factorial" informally.

It is important to us that 1) this definition was taken randomly by us, and it was not prepared (or juggled) for illustration of any advantages of algorithm design in the VTR, and 2) despite its triviality, this example demonstrates general principles of work organization for R-charts designing as well as distinction of this process from traditional technology.

This definition was written for human understanding, but not for organization of computations, and just two first paragraphs of this definition could be attributed conditionally to computing algorithm. Let's write logic of these paragraphs in R-chart as we understand it (Fig. 13.1). It does not matter that another person may write it differently, but what is important that a developer's flow of thoughts is documented as well as his terminology taken from the encyclopaedic description of a problem statement.

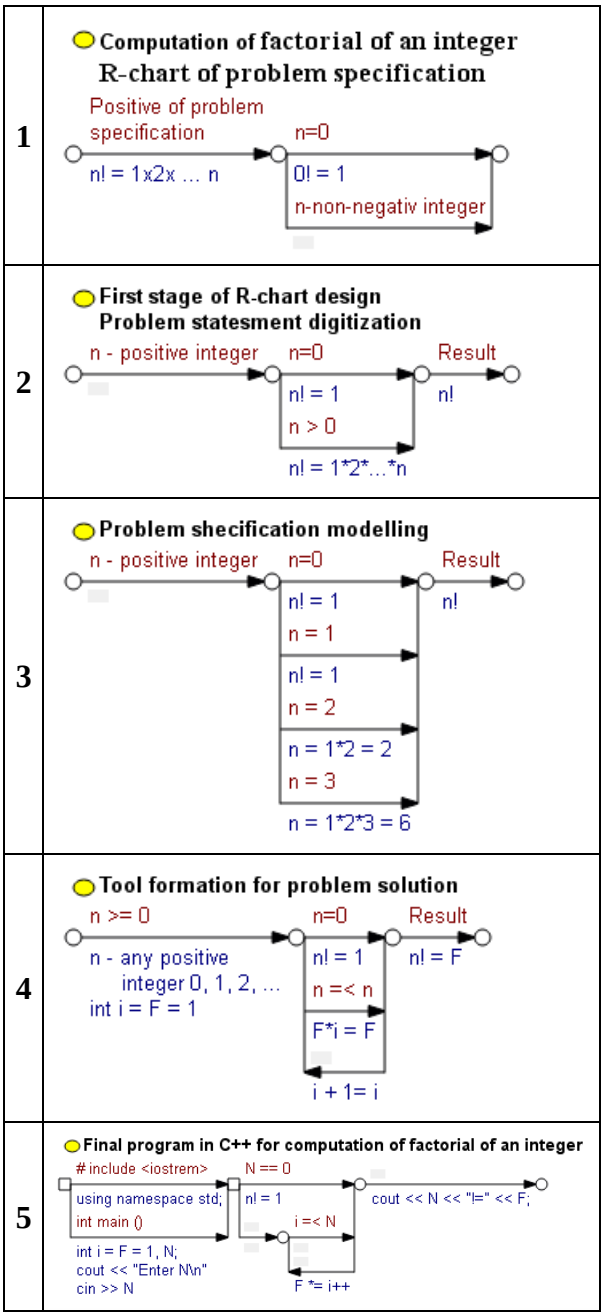


Fig. 13 Five algorithm design stages for n! (factorial) computation

Further he will work with it until finds his own understanding of the computer program or becomes more confused in his thoughts. If such confusion occurs, he will turn back and

repeat the entire development process with a trial-and-error method, which is quite understandable to a person, or he will formulate drawbacks of the ODs and return them to the customer for improvement (to Encyclopaedia of Mathematics).

At the second and all subsequent design stages modification of R-chart of preceding level (Fig. 13.1) is carried out using specification and editing of available records with addition or deletion of the arcs according to the declared strategy of design stage written near an ellipse of this stage.

At the third stage, the developer decided to look into the computing formula for n! (and to model it). In order to do this, he copied the R-chart from Fig. 13.2, added two more arcs and calculated manually a factorial for n=1,2,3 using the formula in Fig. 13.2. Apparently, this modelling was enough to him to understand how to organize a factorial computation for any **n** (in Fig. 13.4).

At the last stage (Fig. 13.4), the developer formulates an algorithm as he understands it and using the terms clear to him. In the considered case – this is a text mix in natural, mathematical and programming languages. The resultant R-chart is being a tool for execution of the source task and computation of factorial of an integer. In order to transform it into a computer program, programming language has to be chosen and **records on the arcs have to be specified** using a standard method (Fig. 13.5). A developer himself or a professional programmer can make such specification. Nowadays, it is easily to contact any professional programmer through the Internet even on the commercial basis.

It is worth to mention that the language of such description (R-chart) is the **same (!)** both for the computer, for the customer and for all the project executors during its entire life cycle. Each developer working with the international team of the specialists may write loads on the arcs using the language alphabet and convenient (native) to him. They are clear to everyone in the system: a record above the arc – Condition, under – Action, near ellipse – definition of earlier introduced notation etc. Described general method is applicable to any technology, including OOP.

VII. CONCLUSION

Now VTR is implemented in the environment Qt-Creator C++. It includes the graphical editor, a system of computer-aided design of R-chart, translator R-chart in C++ and module for the environment and Qt-Creator, which provides for the usual modern technology OOP programmer (www.glushkov.org).

This implementation demonstrates that the VTR is ideal for recording of the event handlers, which are executed in traditional (and visual, and OOP) technologies using programming languages (PL) in conventional textual form – C++, Java, Delphi etc. Since VTR is basically visual and much simpler approach to the traditional text-based PL, the corresponding visual ENVIRNMENT of the organization development process for VTR will be much simpler, more compact, more clearly, etc. existing ones. Above all, it will use the same graphic (visual) principles along the entire lifecycle of program development – a directed graph loaded through the arcs (instead of using the eclectic of text-based PLs, flow charts, UML-diagrams, a palette of the icon components etc.). Programming becomes conceptually uniform and independent of the Tower of Babylon of the programming languages and styles.

The 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 to now.

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 1989.08.01 Geneva 20, Switzerland, ISO/IEC Copyright Office, p.7, 1989.

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

[5] 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.

[6] 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.

[7] I.V. Velbitskiy, "Next generation visual programming technology", *11th IEEE EAST-WEST DESIGN & TEST SYMPOSIUM (IEEE EWDTS)*, Russia, Rostov on Don, Plenary report, Sept.27-30, pp. 404-410, 2013.