Next generation visual programming technology

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

Abstract

The next generation of visual programming technology, which uses so-called ***R-charts (ISO/IEC 8631)*** *is defined how VTR. R-charts (or R-scheme) - is oriented graph, which is loaded only on the arcs (on/under arc). The main difference of BTP is her simplicity and only graphic representation during the entire program life cycle, which ensures new principles of design, debugging, visual and* ***compact (by more than one order)*** *recording compared to writing in existing programming languages (PL). Graphic paradigm of VTR is set, which is based on a single graphical shell for existing PLs* ***without*** **goto, if, for, while, break, labels,** *parentheses* **begin-end, {-}** *etc. Three- and multidimensional programming is introduced. It is integrated well with available programming systems. It 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 was realized in Qt-Criotor C++ environment, which allowed to identify of VTR advantages compared to PL, UML, OOP.*

1. Introduction (history and advantages)

We have introduced for the first time 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 last century. 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 and one (1) international conferences had been conducted dedicated to R–technology issues as well as tens of special workshops, and more than 600 works were published related to various scopes of its application. Before the breakup of the Soviet Union and CMEA, R-technology won a contest of technologies 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]. This technology was analysed and valuated high by the American experts in 1989 [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 media. New concept was realized on modern platforms and discussed at the international conferences [5, 6].

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 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 **graphic shell - single (!) for** all the known languages. R-technology is viewed not instead of, but **along** [6] with everything that exists 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 graphical shell. However, this arc is more powerful compared to traditional PL operators. 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 graphic shell is applicable in all the languages and during the whole life cycle of the programs.

Compactness. Graphic recording of the programs is at faster compared to traditional recording and fits better to information presentation form acceptable in classical mathematics. There is a chance (for the first time) to increase compactness of recording **by more**

than one order providing view on the whole programming project like from the top – bird's eye view.

Visualization. Software, 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,** parentheses **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.

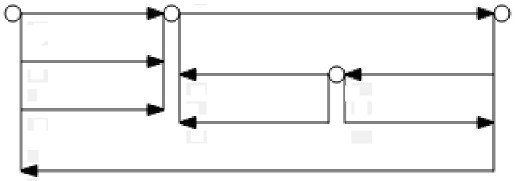
Capacity. Theoretically, R-chart is equal to the Turing machine. It means that one reduced arc of the R-chart is enough to write any algorithm. In practice any PL may have the specified simple graphical shell that is being single(!) for all the languages. It means that everything that may be written 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. So the graphic interface is very simple, it is very efficient broadcasting in the sense of memory and speed of result codes, as well as simplify the hardware architecture of computers. 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 operators (tool) as it happens now in traditional programming. This erases the problems of modern programming and opens the door to demonstrative 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. The processes of algorithm design from its informal writing in related documents are automated. Program debugging process becomes demonstrable. Simplification of the basis reduces also its add-on – visual programming environment that can be user-customized for the first time with memorization, accumulation and development of its intellect. New prospects of distributed Internet—design techniques are open.

Continuity. No needs to break anything and to make differently. Everything that exists in the programming will remain and just a graphic 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 the 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 in the graphical shell. It means that the user can employ *linear* structures of programming languages (procedures, functions, descriptions, expressions etc.) adopted in programming from the classical mathematics, in the habitual form and, consequently, protected better from the errors, on the R-chart arcs.

2. Determination of VTR basis

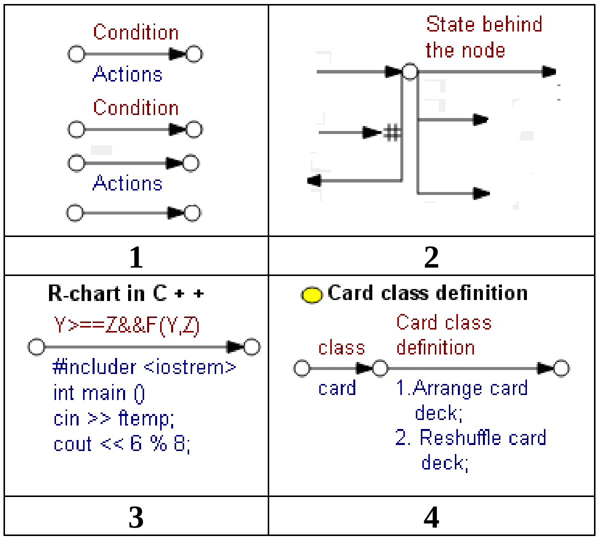
It is proposed not to write, but to draw the programs during their entire life cycle in the form of graphs consisting only of horizontal and vertical lines, Fig. 1. Such graphs are entered easily in the computer using a mouse only (there is equivalent effective procedure in place as to tracking through the nodes of R-chart and drawing a graph using keyboard means 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 nodes with the horizontal arcs, which are being main and loaded with information.



[illegible]

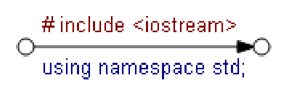
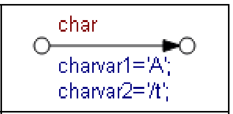
A node does not have the name and it assigns of the program state or the process of its development. Any number of arcs outgoing to different directions – to the right and/or to the left, may be connected to one node. The outgoing arcs at each node are being visible (read, understood, analysed, and executed)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

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 is written on the arc top, and below – the Actions performed during this process. No any limitations are placed on conditions and actions writing – they can be written at any language: Russian, English, Chinese, mathematical, programmer's, etc. in one or several lines.



[illegible]

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 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 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 stipulated for determination of 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).

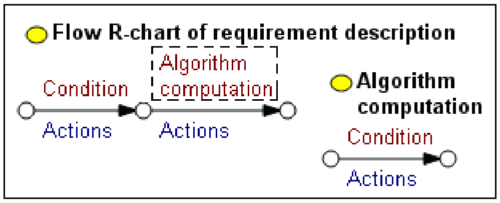
	

#include <iostream>
using namespace std;
	char charvar1='A';
	char charvar2='/t';

[illegible]

[illegible]

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.



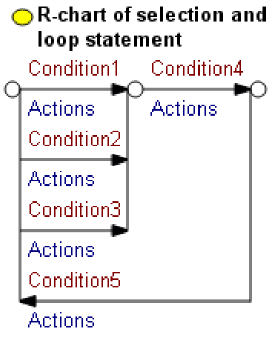
[illegible]

3. Examples of R-charts recording

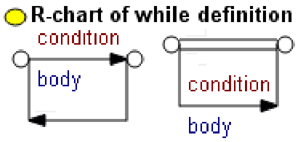
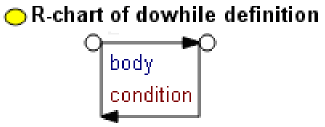
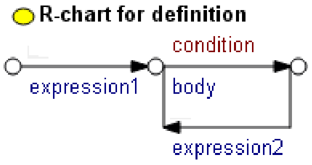
R-chart recording of selection and loop statement and appropriate its record in C++ is given in Fig. 5. The R-chart is incomparably more visual, two-fold compact, and it is entered into computer six-fold quicker. The VTR's user does not see the right part of Fig. 5, which is given here as a formal definition (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. The R-charts fit better to human thought process for a problem to be solved, since they set a repetition pattern described in the task, but not a modification of such pattern for the appropriate statements of existing PL with strict syntax. This is considered as a prime cause of errors generation and main problems of existing programming, including OOP.

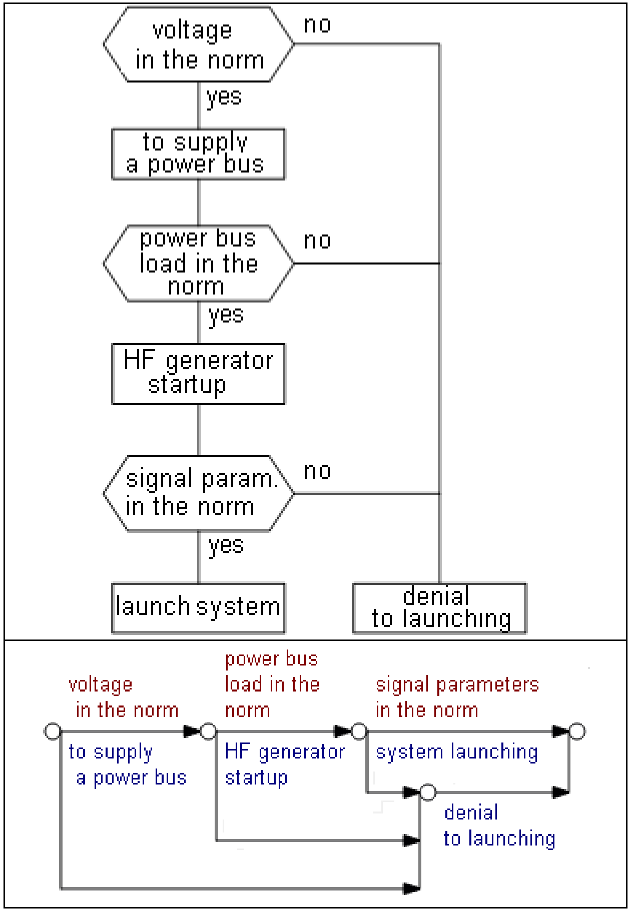
Comparison of R-charts with other well-known graphical methods for algorithm writing of UML type, Flow Chart etc. is given in Fig. 7. The R-chart is the only one of graphical methods that is used along the entire life cycle of the programmes. It is more compact among the all known methods.

 R-chart of selection and loop statement
Condition1
Condition4
Actions
Actions
Condition2
Actions
Condition3
Actions
Condition5
Actions
	R-chart of selection and loop statement in C++{
	_l2:
	if(Condition1){
	Actions
	}else{
	if(Condition2){
	Actions
	}else{
	if(Condition3){
	Actions
	}
	}
	}
	if(Condition4){
	Actions
	}else{
	goto _l1e;
	}
	if(Condition5){
	Actions
	goto _l2;
	}
	_l1e:;
	}

[illegible]

1	while definition in C++: while (condition) body  R-chart of while definition condition body condition body
2	do while definition in C++: do body while (condition) 
3	for definition in C++: for (expression1; condition; expression2) body  R-chart for definition condition expression1 body expression2

[illegible]

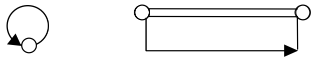


[illegible]

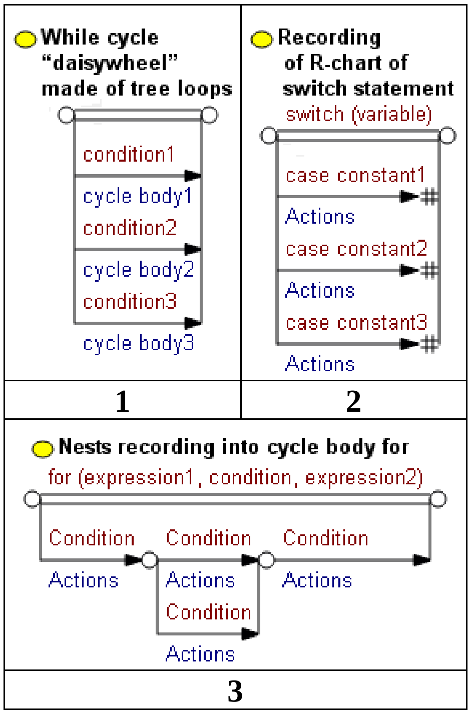
Accordingly, in the first place the VTR visualizes that part of existing traditional programming technique that serves as the main source of meshing, errors and difficulties in program design, in particular, transitions, loops and branching. Linear parts of program writing such as expressions, functions,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

4. Basis enlargement

The VTR allows of flexible development of drawing tools during 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:



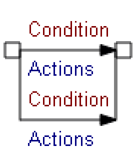
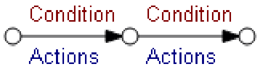
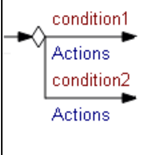
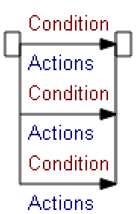
Any information in any language can be written above or under the arc the same as on ordinary arc with an arrow. The name of special arc and corresponding construct are written above, and the Actions – below, see Fig. 8. If no any records are on the special arc (see Fig. 8.1 and 6.1.2), then this will correspond (on default) to graphical recording of traditional loop statement **while**. In case with 8.2, three statements **while** are written in one R-chart and thy can be executed in parallel.



[illegible]

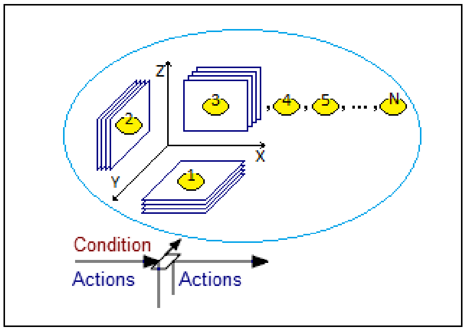
If a key word is written on the special arc (see 8.2, 8.3), it will serve for recording of special language constructs, for example, **case** and **for,** respectively. Such record is useful for reverse translation from existing PL to the R-charts. This record is also convenient to utilize for generation of new and often used graphical structure.

The R-chart's nodes 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 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 special configuration of the arc or node.

	**Small square** – vertical recording of arc sequence between square nodes, 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
 Condition Actions Condition Actions Condition Actions	**Vertical rectangle** – simultaneous sequence execution (on the 3-leg diagram) of arcs between the rectangles

[illegible]

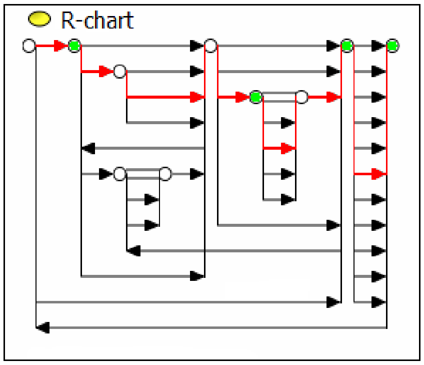
For many applications such as linear programming, 3D trajectory calculation, parametric representation of multidimensional sculpted surfaces, etc., it will be convenient to employ multidimensional R-charts: **1, 2, …, N**. This could be done in the VTR using a node of specific shape - a parallelogram,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



[illegible]

The source R-chart with a node of parallelogram shape is *on default* in multidimensional space **No.0** (usually n-1). Each multidimensional space **1, 2, …, N** consists of two-dimensional R-charts and begins with the new Document (an ellipse with the number). Any amount of R-charts and Documents may be in this space and they are recorded in the Design Tree. At completion of sequential (or parallel) execution of all the multidimensional R-charts, return to the source R-chart of multidimensional space **No.(n-1)** have place to the point behind a parallelogram-shaped node.

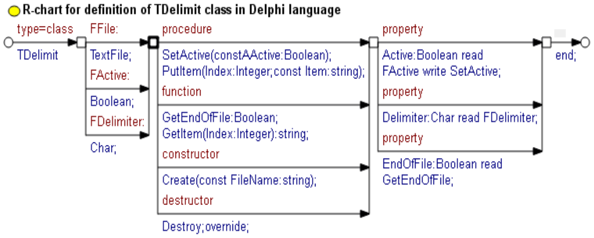
The colour is used extensively in VTR to highlight nodes, arcs and records on the arcs, see Fig. 11. Usually, **Yellow** colour is used for marking of an ellipse before the name of each new R-chart. Brown colour is used for writing of Conditions; Blue – for writing of the Actions. Red colour (Fig. 11) 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.



[illegible]

Ability of *compact recording* when all inscriptions on the arcs are deleted from the program using one command on the 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. This allows seeing the program, its logic and structure like from the top – bird's eye view. Taking into account the fact that a programmer remembers well "what its program does", such 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. For the first time, the program receives "its face", its hieroglyph that can be used to understand at a glance and to discuss design decisions to be taken.

Figure 12 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. This example demonstrates effectiveness of visualization of bad linear part (not winning for demonstration of VTR advantages), which does not contain complicated and convoluted flow, in the R-charts. R-chart ensured visualization and compactness for description of this OOP fragment. The *structure* of defined class became visible (clear) immediately: column (3 arcs) for data definition; column for method definition – of two procedures, two functions, two special procedures – constructor and destructor, and column (3 arcs) for class properties definition. Possibility to fix visually the definition structure for algorithm, program, and object – is being new and important concept of graphical paradigm.



[illegible]

5. VTR graphical paradigm

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. This model has neither core nor common development scheme, or the future. 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 and human thought process (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. This simplifies considerably and solves existing problems of traditional programming, including OOP.

6. Correlation with classical mathematics

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

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. Recommendations to use a statement **goto**, if it is included into the program by the appropriate automation system without human participation, are less known. This allows elevating performance effectiveness of the resultant program. Moreover, one proposal does not contradict another one.

The next generation VTR technology **excludes** the abovementioned problem key words and the appropriate structures of switch and loop statements from the programming, and also the brackets of type **begin-end, {-}** etc. It is proposed to use the graph theory from the classical mathematics instead of them. There are two types of graphs available in the classical mathematics – a graph loaded through the vertexes (Flow chart, UML, DRAGON etc.) and a graph loaded through the arcs (Network graphs, R-charts etc.). The latter is taken as a basis for VTR and ensures the advantages specified above along the entire lifecycle of the programs.

7. 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 in traditional programming are changed continuously and rewritten, or it could be even worse (which happens often) – they are recorded nowhere and remain in a programmer's head in order he can write an invented algorithm in the statements of one of the existing programming languages (PL). Originating documents are fixed (axiomatized) in the VTR and a tool (R-chart) of program design process is modified. This is done using 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 recorded in optional symbols. Further it is defined more exactly with a "step by step from logic", 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, the VTR neutralizes drawbacks of existing ODs and on return it allows receiving more qualitative software 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. We shall take definition of factorial of an integer given in the Encyclopaedia of Mathematics in the Internet as a basis:

> **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. 14.1). 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. Further he will work with it until finds his understanding of the computer program or becomes more confused in his thoughts. If such confusion has place, 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 the Encyclopaedia of Mathematics).

It is worth to mention that existing Graphics Editor allows 1) to construct quite complex and tangled R-chart in any language of the customer and the developer; 2) to carry out specification easily for any concept of R-chart used by them at the next designing level as it is for example shown in Fig. 14.2); and 3) to collect automatically all R-charts with specifications into single resultant R-chart.

At the second and all subsequent design stages modification of R-chart of preceding level (Fig. 14.1) 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. This stage is named as "Problem statement digitization" in Fig. 14.2. It is not important whether this name successful or not, but it is clear to the developer. Based on his understanding of R-chart designing concept at the second stage (Fig. 14.2), he did the following: 1) changed the condition on the first arc by connecting its text with condition on the third arc, and edited slightly terminology "convenient for him". In the developer's viewpoint, these inscriptions are equivalent, and he brought them to one and more understandable to him; 2) changed the action on the second arc; 3) changed the condition on the third arc; 4) rewrote the condition on the first arc under the third arc, and replaced operation of multiplication by " x " with " * ", which is more understandable in programmer's circle; and 5) withdrew the result at the end of R-chart by adding the appropriate last arc.

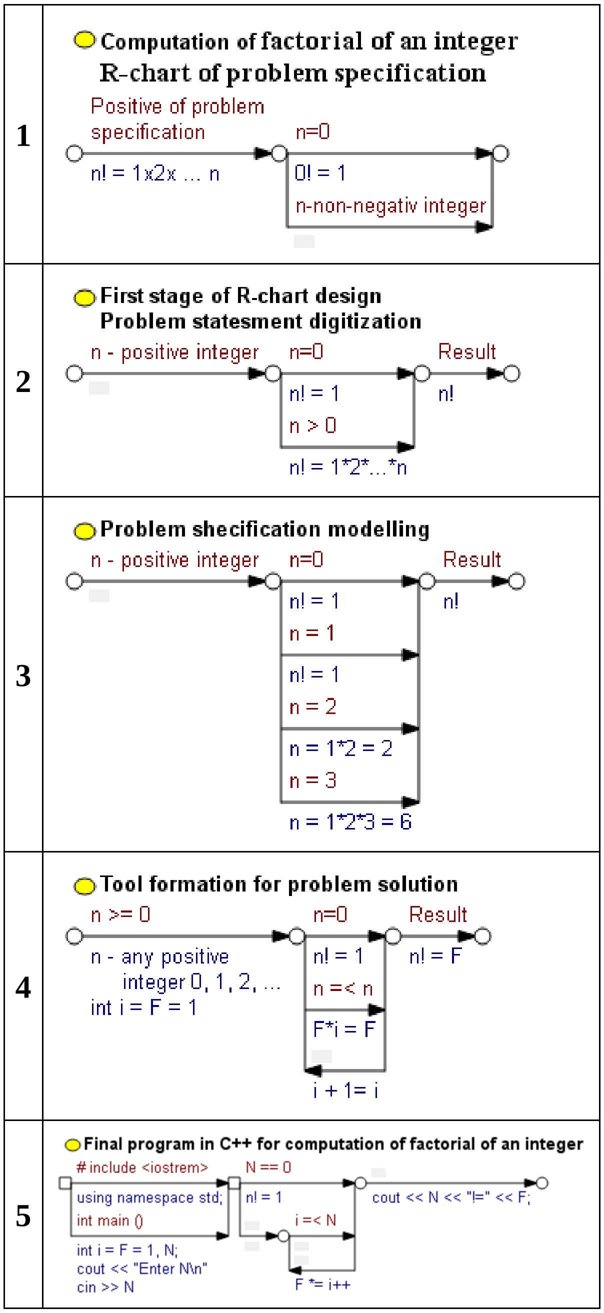


Fig. 14 Five algorithm design stages for n! (factorial) computation

At the third stage, the developer decided to examine the computation formula for **n!** (and to model it). In order to perform this, he copied the R-chart from Fig.

14.2 into Fig. 14.3, then he added two arcs after the third arc, and on the last three arcs he calculated manually a value for **n!** for n=1,2,3 using the formula given Fig. 14.2. This modelling was enough to him to understand how to organize further factorial computation for any **n.**

At the last stage (Fig. 14.4), 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 on natural, mathematical and programming languages. The resultant R-chart is being a tool for execution of the source problem and computation of factorial of an integer. In order to transform it into a computer program, programming language has to be chosen then and **records on the arcs have to be specified** using a standard method (Fig. 14.5). A developer himself or a professional programmer can make such specification. Nowadays, it is easily to contact any professional programmer through the Internet 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 lifecycle. R-chart is only one, but it is convenient to maximum for the appropriate executor of the project **due to specialization of the tool** (R-chart) for specific task performed at every step of the VTR technology. Project executor writes on the arcs without limitations using language understandable to him and applies logic of information to maximum. At the same time (!), **all the executors understand** these records thanks to ODs, earlier made R-charts and agreed terminology starting from the initial informal description of the active task. The R-chart documents logic of motivation of adopted project solutions at every step of the VTR technology starting from ODs received from the customer. Each developer working in the international team of the specialists may write loads on the arcs using the language and alphabet convenient 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.

8. 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 to now.

Now VTR is implemented in the environment Qt-Creator C++(Programmer Gubov A.M.). 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 with texts on the arcs of R-chart (source code: www.glushkov.org). This is the first, not commercial implementation. She had two objectives: 1) to determine how time-consuming to move to a new generation of VTR and 2) comparison of the VTR that is already there, and is it worth it to go if you have a traditional (non-graphic) technology, a well-established and familiar for many? Our final conclusion - is worth it! and inevitably visualization solutions logically challenges of the future. All that is best in modern programming is stored in the BTP and takes it very important feature of the visual of the entire life cycle of software that allows you to get rid of so many problems of modern programming. Of course, there are drawbacks - it's unfamiliarity and lack of commercial sale, but ... this is a temporary illness of all new and are too small and primitive compared to opening new opportunities.

Recall that in theory the VTR has no restrictions on the use of effective, and almost twenty years ago, has more than 600 publications for its effective application in various fields. Conducted in the present analysis and pilot implementation is enough to recommend the Project R-graphical programming schemes for large-scale commercialization.

9. 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] INTERNATIONAL STANDART ISO/IEC 8631. Information technology-Program constructs and convention for their representation – Second edition 1989.08.01– Geneve 20, Switzerland:ISO/IEC Copyright Office, 1989-7p.

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

[5] 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.

[6] L.F. Drobushevich, Common use of UML and P-chart notations in the training process for software system development methods, MEDIAS-2010, Cyprus, p.73-77.