R-Technology of Programming: Basic Notions and Implementation

I. V. Vel'bitsky and A. L. Kovalev
(Technosoft, Kiev, USSR)

I. V. Kasatkina
(Kiev Polytechnical Institute, Kiev, USSR)

Wang Lei (王 勰)
(Tianjin University, Tianjin 300072, China)

Received September 10, 1990; revised June 10, 1991.

Abstract

This paper presents a kind of visual programming, which is called R-technology of programming. The R-technology is independent of any programming language or operating system and the R-chart is in accord with international standard (ISO 8631H). The package of R-technology has been applied to the high level languages such as PASCAL, C, ASSEMBLER, FORTRAN, PL/1, MODULA-2 and RTRAN. This package is applied to computers ranging from mainframes (IBM 370) and minis (VAX) to micros (IBM/PC).

1. Introduction

In recent years visual programming has been intensively developed in the United States, Western Europe, China, Japan and the Soviet Union. Visual programming refers both to manipulating visual objects, such as icons or images, and to representing objects, such as data structures, which do not have an inherent visual form, in visual terms. The field of visual programming has become fairly large. One major area of research focuses on the design of new visual languages, many of which include icons as fundamental programming objects [1]. Other languages and interfaces are being developed to create interfaces for database management systems, for forms and other applications [2]. There are also systems which support visual interaction in which the logical objects, such as arrays, are still represented by conventional programming language statements, but the control paths are represented visually. These include Visual LISP [3], ICONLISP [4], FORTH [5], C [6], PAD [7], R-technology [8]. Underlying formalisms include dataflow diagrams, HIPO charts, action diagrams, Nassi-Shneiderman [9] and R-charts [10].

This paper is devoted to the R-technology of programming which has been under development for more than two decades at the Institute of Cybernetics of the Ukrainian Academy of Sciences, Kiev, USSR.

In R-technology programs are formulated as a mixture of traditional linear text and text which has been written on a graph structure. The declaration sections of a program are written in the usual fashion, whereas the executable statements are written on a series of graphs, called R-charts.

R-technology was initially based on the concept of formalized R-grammar which can be described in terms of weighted, oriented graphs. Many programming languages, described in terms of R-grammars, could be subjected to a syntax analyzer and translated into any other desired form; thus a program is simply a set of rules to transform the data from one form to another. R-technology is well suited, therefore, to a particular class of translation-oriented problems. The idea of using the structure of the data to drive the process of programming along with visual programming is the central concept of R-technology.
2. Notion of R-Chart

Drawings are the most well-developed and integrated forms of knowledge representation and have been standardized for most subject domains. A number of standards have been introduced for representing program flow, i.e., flow chart in ISO 5807 according to which most programs are documented. The relationship between the flow chart and actual program is realized by the interpretation of a programmer which often leads to errors.

The notion of R-chart has been introduced to translate the flow-charts to actual program source code and a standard has been evolved which is adopted internationally (ISO 8631 H). The aim of the R-chart is to present visually (graphically) the logic structure or program flow independent of any program language or operating system, and to be used to compile programs in any language, e.g., assemblers or high level languages as Fortran or C.

Let us introduce the notion of R-chart formally.

The R-chart consists of arcs and nodes. Its arcs are represented by horizontal and vertical lines. Its nodes represent the states of computational process. On top of each arc is a condition; if the condition is true then the statement block underneath the arc is executable.

Let us introduce the notion of elementary structure.

The elementary structures in Fig. 1 are called direct, reverse and loop structures, denoted by \( m, n, k \), respectively, where \( m \) is the number of direct lines, \( n \) the number of reverse lines, \( k \) the number of arcs in the loop-structure. \( \alpha_i, i \in \{1, 2, \ldots, m\}; \beta_j, j \in \{1, 2, \ldots, n\}; \gamma_h, h \in \{1, 2, \ldots, k\} \) are conditions of passing through corresponding arcs. \( A_i, i \in \{1, 2, \ldots, m\}; B_j, j \in \{1, 2, \ldots, n\}; C_h, h \in \{1, 2, \ldots, k\} \) are statements to be executed when corresponding conditions are true.

![Fig. 1](image1)

More complex structures can be built up out of elementary structures by means of three connection operations: sequential, parallel and embedded. (For example, insert a loop structure to sequential connection). An example is given in Fig. 2.

![Fig. 2](image2)