Visualization Metaphors

V. L. Averbukh

Institute for Mathematics and Mechanics

Urals Branch of Russian Academy of Science and

Ural State University,

16, S. Kovalevskoi, Ekaterinburg, 620219 Russia

e-mail: averbukh@imm.uran.ru

Abstract

In this paper the conceptions of Visualization Language and Visualization Metaphor are suggested. The structure of a metaphor and the some conception linked with visualization languages are considered. The paper includes the critical overview of metaphor used in CHI theory and in the modern practice of the interactive and visual environment design. Prototype tools for formal and empirical quality evaluation of visualization metaphors and languages based on the adequacy in visualization are described.

1. Introduction

"Visualization is a method of computing. It transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. Visualization offers a method for seeing the unseen. It enriches the process of scientific discovery and fosters profound and unexpected insights. In many fields it is already revolutionizing the way scientists do science." [23] This classical definition was published in 1987, which, certainly, does not mean that there was no visualization until 1987. In the 1970s--1980s, full-featured visualization systems were developed, whereas concrete examples of visualization are found before the first graphical packages appeared in the 1960s. Numerous existing general-purpose and special-purpose visualization systems are traditionally divided into three main classes:

- scientific visualization systems;

- information visualization systems;

- software visualization systems.

t is worth noting that, although visualization systems differ in purposes and implementation issues, they all have something in common, namely, that they manipulate some visual model of the phenomenon under consideration serving as a basis for translating a computer model into a concrete graphic representation. In its turn, this visual model is based on the idea of bridging the gap between different areas that exists in the mind of designers and users, i.e., on a metaphor.

Many designers and users implicitly assume that visualization is helpful only because it provides clear visual images, although some researches are skeptical about this still popular approach (for example, [27, 28]). At present, it is rarely discussed which visualization is needed (it is assumed that "good" visualization is needed, or standard methods are used). There is a number of works studying both design techniques and quality evaluation of visualization systems. However, these works are usually based on a priori assumptions or empiric estimates, and they hardly lean upon the sufficiently elaborated theory of visual languages.

We propose an approach offering a single framework for studying visual systems of various purposes and allowing a scientific basis for evaluating visual systems in order to provide a scientific foundation for their design. This approach rests on semiotic analysis and determination of the visualization language as the core of any visual system. The conception presented in our paper summarizes as follows:

(1) Each visualization system contains as its core the language considering as an unity of the vocabulary, syntax, semantics, and pragmatics.

(2) Visualization languages are built upon some basic idea of similarities between application domain entities with visual objects, i.e., upon a visualization metaphor.

(3) It is necessary to evaluate metaphors and visualization languages with respect to the main parameter, that is, their ability to satisfy user's needs to represent by visual way results of user's problem decisions, i.e., with respect to the adequacy in visualization..)

(4) While designing high-quality visualization systems, it is necessary to take into account the adequacy in visualization, which must be constructed as a function having as its parameters user's features and the application domain properties.

In Section 2, the conception of visualization language is considered. Section 3 gives a critical survey of how the conception of metaphor has been used in the theory of the human--computer interface and in the design of modern interactive and visual environments. The conception of visualization metaphor is introduced in the same section. In Section 4, its structure is considered. Section 5 is devoted to parameters of formal quality evaluation of visualization metaphors and languages. In Section 6, several prototype systems for empirical evaluation of visualization quality are examined. Next, results of experiments are discussed. In Conclusion, unsolved problems related to the development of the visualization metaphor theory, as well as prospects of further researches, are considered.

2. VISUALIZATION LANGUAGE

In order to define the conception of visualization, it is essential to consider the stage of mapping a computational model of the entity under study into some visual representation based on the mental model of this entity in the mind of the user and/or the developer of this visualization system.

Now, let us consider the conception of *model entity*, i.e., an object of the computational model to be studied, an object whose state and behavior, characteristics, attributes, and features are of interest to the researcher and, hence, are to be visualized.

We define a *view* as an abstraction of *graphical display* containing the specification of visual objects, their attributes, their interplacement, potential dynamics, and interaction techniques.

Visualization abstraction implies that *model entities* are related to the *view* so that the content, behavior, features, and attributes of model entities could be represented by a concrete graphical display exactly identifying all visual characteristics corresponding to attributes of the *view*.

Analysis of visualization systems reveals their similarity: from the set of graphic displays presented to the observer, it is possible to recognize some *visualization language* with the vocabulary consisting of views used in this system. Grammar rules for constructing the visualization language are specified by the order in which views replace each other. Obviously, considering visualization languages, one should not only specify the language on the basis of user manual, but also perform a subtler analysis of entities described by the system and methods for their representation including possibility for the user to manipulate visual objects on the screen. Significant units of the visualization language are 2D or 3D images and changes in the values of their graphical and nongraphical attributes. Analysis shows that the actual vocabulary may be larger than the vocabulary estimated by the designer because of unaccounted factors affecting the user. For semiotic analysis of the visualization language, it is necessary to properly define its spatial syntax and semantics. (An attempt at defining spatial syntax and semantics is found, e.g., in [10].) However, of prime importance is the specification of actual pragmatics of visualization languages, which can substantially differ from pragmatics estimated by the designers of the visualization system. When developing visualization languages, one should aim at the correct and natural interpretation of visual texts for representatives of specific national or professional culture. There is no pragmatics of visualization languages suitable for all applications and users. The problem of pragmatics is tightly related to the fact that perception of the visual text is subjective and is dependent on cultural, psychological, and physiological factors. As a rule, visualization languages are "reading languages", and significant units of their vocabulary assist the user with perception and further interpretation of visual phrases. The study of perception of texts in visualization languages led to the following observation: generally, visualization systems are successful if they deal with visualization of a limited number of entities and functions.

All this is also necessary for analyzing visual programming languages, which are essentially based on principles similar to those of visualization languages.

3. VISUALIZATION METAPHOR

The use of metaphors (figurative similarities of concepts) became inevitable as soon as modern computers emerged, since tools were needed that would allow describing completely novel phenomena and objects. Words for these descriptions had to be selected and adopted based on external or functional similarity. For example, a file as a container of a card-catalogue gave rise to a file as a container for punched cards carrying data and, finally, to a file of data; the concept of block diagram as a schematic diagram of an instrument or an electronic device is extended to a diagram representing the modular structure of the program; etc. New generations and new occupational categories of users (speaking different languages and belonging to different cultures) adopting these concepts, the metaphors fade out; i.e., words cease being metaphorical and evolve into ordinary designations. (By the way, this process of emergence and fading out of metaphors used for new concepts has repeatedly occurred in the history of the language.) However, our study is centered not on the metaphor in computer science, but rather on its particular case, namely, *visualization metaphor*.

Visualization is always based on figurative similarities of entities being studied and visual objects representing them, i.e., on some metaphoric representation. The development of this idea is the basis of every visualization language.

There are several approaches to understanding the metaphor from the viewpoint of the development of visualization systems and visual interface.

The conception of metaphor dominating at the moment is based on representing phenomena that are new or rather untypical for the user by means of phenomena familiar from everyday life; the latter phenomena must possess the same main properties as the phenomena they explain [25, 34]. The major achievement of this conception is a matter of common knowledge: this is the universally used desktop metaphor, which establishes relation between office life and programming concepts. This emphatic success was followed by a number of more or less effective attempts at creation of global metaphors of interface. They include the working room metaphor used for information visualization [32], flying and fish tank metaphors (as well as the hybrid flying and fish tank metaphor described in [15]), the geographic space metaphor used, e.g., for managing software development [11], metaphors of theater, cinema, and comics in systems of programming by demonstration [36]. In this case, the metaphor is a means for establishing relations between different fields of human activity and it is useful for stimulating associations. Such metaphors define the entire set of concepts with which users deal when solving problems. Visual symbols are used for describing ideas, actions, and commands. A global metaphor provides additional insight into interaction semantics; in addition, it offers visual representation of dialog objects and defines how the user may manipulate them. There are use cases of local problem-oriented visualization metaphors for monitoring parallel computations and corresponding operating systems also founded on universally accepted everyday and technical concepts [2, 25, 34].

Notice that the research and development results partially described in [2], which are related to the use of metaphors for specification of programming entities, are of particular interest, as they stem from long-standing work with students specializing in Software Engineering and parallel computing. The images used are of different imitation degrees: realistic images of a rotating magnetic disk, schemes and time diagrams, and primitive abstractions. Imitation degree affects perception, and the most realistic image is not always the best one, since details may hide the concepts to be learned. Sometimes, images used for visualization of concepts belong to the field that is quite distant from the one under consideration. For instance, synchronization, mutual exclusion, parallelism, semaphores, and other concepts were demonstrated by means of a model of a water conduit with tanks, pipes, and stops. The advantage of such approach is that it addresses everyday experience and generates interest among students, which facilitates understanding and learning the principles of the process under consideration. The shortcomings of this approach include the fact that it misses details and some specific concepts for which no analogues were found in the selected field, necessity to compare concepts from different fields during learning, as well as additional undesirable analogies related to everyday metaphors.

Metaphor is understood differently when speaking about a visual system supporting various graphic metaphors used in programming, in particular, finite automata, block diagrams, and dataflow graphs. The corresponding metaphor is typically supported by an ad hoc visual programming language constructed by choosing some aspect of the program, a graphic model, and relation between program aspects and graphics, which defines the behavior of the graphic model.

Lastly, let us point out some works containing profound analysis of the conception of metaphor and treating it from the viewpoint close to those existing in philology and philosophy.

Lastly, let us point out some works containing in-depth analysis of the conception of metaphor and treating it from the viewpoint close to those existing in philology and philosophy.

In [17], among other things, it is discussed how metaphors are employed in computer discourse, which, as it was already noted, requires universal usage of more or less appropriate metaphors for describing novel concepts. (A relatively fresh example is the metaphorical name of the new discipline Data Mining.) A systematic approach to metaphorical design is proposed in [22]. Theoretical foundations of methods for associating entities in metaphor selection are discussed. In [31], interconnections between the notions of metaphor and illusion are explored from the standpoint of user interface design and analysis for immersive visual reality systems.

Despite the emphatic success of metaphors in human--computer interface, one should not miss rather justified critique found, e.g., in [26, 30]. Of fundamental importance is the remark that the transfer of meaning, which supports visual metaphors by means of similarities or analogies to real-world situations, can be either positive or negative when limitations of real-world situations are imposed on the metaphorical meaning [30]. As mentioned above, the nature of phenomena could be understood too simplistically; details and some specific concepts for which no analogues were found within the bounds of the selected metaphor could be missed. The use of metaphor often leads to metaphorical artifacts; i.e., some properties of objects of the metaphor not existing in the context of the initial problem are transferred to the computer model. Additional undesirable analogies related to metaphors evolve in the mind of the user. Let us point out that it is not acceptable to choose a metaphor based on the similarity of words. Such metaphors often emerge from the programmer slang, and they are language-oriented or culture-oriented, which makes them incomprehensible and harmful to representatives of another language or culture. The simplest example of such a culture-dependent metaphor is using a "bug" icon for representing the function of error lookup; this metaphor is incomprehensible to many Russian-speaking students and programmers. Although some critical remarks could be arguable, the experience of using visual techniques shows that certain criticism is helpful as a warning to visual system designers. In this connection, we would like to note that this critique concerns the narrow understanding of metaphor as application of everyday experience. We believe that it is this narrow understanding of metaphor and not the use of metaphor that is the reason of unsuccessful decisions and a matter of critique. It is incorrect to transfer the understanding of metaphor successfully applied in human--computer interface to all cases of visualization. A wider understanding of visualization metaphor is needed that would include the current tradition of using metaphor in human--computer interface, but would not require to represent all details of the domain selected by the designer.

In this paper, it is proposed the approach to the understanding of metaphor as a main principle of mapping an application domain to visual universe. This approach extends the traditional one, and it is substantially based on the concepts of semiotics. This approach is likely to help formalizing search, designing, and generating views for visualization. Formalization of notions related to the metaphor would provide a systematic approach to quality evaluation of visualization languages and tools.

In our opinion there are no "metaphorless" visualizations of computer models and program entities. Long ago, it was observed in literature that any metaphor is picturesque, and, consequently, any graphic image occurring during visualization is metaphorical. Visualization is always a metaphor by its nature, since it associates concepts of the model to visual objects representing the former by means of the latter for proper interpretation by the user. In this connection, we can define a *visualization metaphor* as a mapping from concepts and objects of the application domain under modeling to a system of similarities and analogies generating a set of views and a set of techniques for interaction with visual objects.

4. VISUALIZATION METAPHOR STRUCTURE

In connection with the semiotic analysis of visualization languages it is necessary to define notions of a sign process and a sign system. Sign process is considered as a set of relation between the sign, the sign interpreter, his/her predisposition to the certain reaction on the sign, the sign signification and context. Sign system is considered as a set of signs, where relations between detonates are mapped by some way into internal relations between signs.

To choose a metaphor is to choose a sign system that will be used for visualization. Another function of the metaphor is to specify a context for better interpretation of elements of the given visualization language. Thus, the visualization metaphor helps understanding entities of the application domain that are being modeled, as well as creating new entities based on the internal logic of the metaphor. The components of the visual metaphor are the imagery it generates and actions it dictates for updating visual images and manipulating visual objects.

The analysis of the use of visualization systems reveals that the metaphor has a "focus" making the greatest impact on the user of the visual language generated by this metaphor. Sometimes, the metaphor focus is founded upon dissimilarity between metaphoric and model entities. In other cases, the metaphor affects the user by placing an object of the metaphor to a semantic context unusual to this object. Note that metaphors without focus are possible and the focus of the metaphor is always perceived subjectively and can be missed by some particular user.

Thus, the visualization metaphor can be specified as a set consisting of the following parts:

imagery of the metaphor; operations directed by metaphor both animation operations and user’s manipulations (in degenerated case the observation may be considered as these operations);

the *set of similarities* between model and metaphoric entities or elements of semantic discrepancy;

the *focus of the metaphor* responsible for the greatest part of the impact the metaphor makes on the user.

The visualization metaphor is the source of grammar, which in its turn generates the language and the system based on this language. Consequently, the description of the metaphor must contain the cores of the vocabulary, syntax, semantics, and pragmatics of the visual language. Let us consider the metaphor as an analogy (or a system of analogies) of the given application domain to another domain whose basic entities are of common knowledge and have conventional meanings. Applying analogy, we immediately define a possible visual vocabulary whose elements have conventional meanings. The perception of elements and phrases in this language is also determined by comparisons of the metaphor being used. The same is true for arranging language elements and the method for specifying relations between them. They are also mainly predefined by the arrangement of objects of the application domain used for the analogy. Thus, it can be said that the visualization language is a result of development of the cores of the vocabulary, syntax, semantics, and pragmatics contained in the description of the metaphor.

As an elementary example of using the approach described above, let us analyze the well-known metaphor of a block diagram employed by J. von Neumann at the dawn of the computer age and extensively used both in project documentation and in many visual programming languages as a tool for representing control-flow program graphs.

The class of languages that could be built upon the metaphor of the block diagram is rather rigidly defined. Obviously, a visual language designed around this metaphor must be a diagrammatic language based on the control flow. The set of images for representing program structures is dictated by tradition if not by existing standards for block diagrams developed in various countries. Methods of joining and arranging on the display are also rigidly defined by the very notion of the block diagram as a control-flow program graph.

Certainly, rich metaphors based on complex analogies are not that rigid and offer greater uncertainty and freedom in choice in visual language development. Therefore, formalized description of the metaphor is needed that would include descriptions of corresponding kernels. Formalization of description must create prerequisites for development of methods for quality evaluation of metaphors and visualization systems based on them.

5. PARAMETERS OF QUALITY EVALUATION OF VISUALIZATION SYSTEMS

Traditionally the goal of visualization is considered as "finding a graphical representation for program (or model) behavior which provides a good mapping to the way the programmers (or researchers) themselves tend to formulate solutions" [13]. However, despite a series of papers dedicated to methods for visualization quality evaluation (in particular, [18] and [21]), the problem of formalizing evaluation has not been solved. Various approaches to this problem are the subject of the following discussion.

Notions essential for developing approaches to systematic evaluation and analysis of visualization systems must be considered. They include the visual informativeness, visual expressiveness, and adequacy in visualization for corresponding systems.

We propose to informally define *informativeness* as a subjective characteristics of quantity of the useful information received by a user (receiver of the information) from the visual text.

Such an approach to defining informativeness reflects the a correlation of subjective and objective, syntactic and semantic properties of the message. In the traditional approach to information evaluation, the semantics of the message does not depend on the properties of the user, whereas the notion of informativeness makes it possible to define subjective semantics and evaluate pragmatic properties of the visual language.

Let us give a tentative definition of the *visual expressiveness* of a language as the possibility to express in this language the maximal number of meanings and shadings of meaning using the minimal number of language units. Evidently, this notion is related to the informativeness of a message. The more informative is an individual visual "word", the more expressive is the whole visual "text". The expressiveness of a text is defined by the number of images describing meanings, as opposed to those describing events or situations. As with informativeness, estimates of visual expressiveness and its quantifiable parameters can be obtained in experimental tests [4].

However, for evaluating actual systems, the user is concerned not only with informativeness or with visual expressiveness of the visualization language, but, primarily, with how well this language can satisfy his or her needs. It is necessary to be able to evaluate the speed and correctness of interpretation of the visual text phrases . If the language of the visualization system is laconic, informative, and well interpretable (admits translation into the mental language of the user), then this language is appropriate for the user needs. It can be stated that this language is adequate in visualization.

Adequacy in visualization defines the properties of the visual language that allow obtaining the best solution for a particular application problem by the given user or class of users. While on the subject of evaluation of adequacy in visualization, one must consider such characteristics as user perception of the visual language, precise interpretation of this language, and proper responses to requests of the interaction language. A negative factor is that phrases of the visual language under study can have an additional meaning (other than that intended by the designer of the language). The importance of adequacy in visualization depends on the visual informativeness and the visual expressiveness of the language. In order to describe adequacy in visualization, one should not only evaluate these characteristics, but also analyze the corresponding application domain and construct the model of the user or (if possible) the whole class of potential users of the system under study [5].

While analyzing the application domain, it is first necessary to use some visual metaphor to evaluate the possibility of mapping the application domain objects into some visual space generated by this metaphor. It is required to give a description of basic entities of the application domain taking into account the analysis of visual imagery and to choose basic concepts, possibly, establishing useful similarity to images generated by the metaphor. Next, key entities of the given application domain must be selected for visualization. These entities do not have to be basic in this application domain, but they must allow solving the visualization problem. In connection with visualization of the application domain, it is necessary to analyze, based on some metaphor, imagery of this domain and imagery generated by the metaphor (e.g., by similarity in terminology), possibility of using abstract symbols and icons, as well as images related to computing algorithms.

6. PROTOTYPE IMPLEMENTATIONS OF VISUALIZATION QUALITY EVALUATION

In our opinion, the key property of visualization systems is adequacy in visualization. In fact, this is adequacy in visualization and not visual expressiveness or other characteristics of the quality of visual metaphors that we tried to evaluate by indirect methods using model problems (described in [4]) to study the speed of the user reaction, the number of mistakes he or she makes, and his or her preferences. We attempted to obtain experimental evidence of this, as well as to work out and study experimental methods for evaluating adequacy in visualization.

Within the bounds of this research, a number of prototypes were implemented. In particular, the "user model" subsystem and several versions of subsystems serving for evaluating the perception quality of closely related visualization languages were implemented. At the next stage of the research, we studied the suitability of metaphors and languages used for visualizing parallel computations; i.e., we developed software tools for evaluation of their adequacy in visualization. The ParaVision system of visualization quality evaluation was developed and used in a series of experiments. The results of these experiments were taken into consideration at the next stage of prototype implementation of systems for evaluating visual languages, in particular, for sorting algorithms animation systems .

Later in this section, we discuss prototype systems and experiments on evaluation of visualization systems carried out with these prototypes.

6.1. User Model

In interactive systems, one way to take into account the context is to work out a user model, which is the subject of numerous researches. In the literature, it is covered in connection with applications, such as teaching systems, interactive systems, WWW browsers, artificial intelligence systems, etc. In studies of the user, models of different levels are considered, from general ones allowing for overall principles of human receptors to specific ones concerned with how individual users work with the mouse. In the process, the results of psychological researches are extensively employed, including special works on the psychology of human--computer interaction [35].

A user model in its general form is structured as follows:

- psychophysical qualities (including age-related and sex-related properties) and emotional characteristics;

- knowledge (general, specific, computer);

- computer or visual environments working experience (positive and negative);

- incentives and motivation of the user;

- national culture;

- professional culture.

In concrete research and development projects, certain points must be expanded into lists that are more detailed and be studied by means of various methods.

While developing prototype systems for quality evaluation of visualization languages, two approaches to the development and use of the user model were implemented. The first one consists in a priori specification and fixation of basic properties of the user in the system. The second approach requires tools for constructing the dynamic model of the actual user as the user is working; this model must allow for the competence of the user in the application domain, his or her psychological and physiological peculiarities, etc. In the process, the time history of the user characteristics, e.g., his or her attitude to the visualization system changing through learning, is monitored.

It was assumed from the outset that the users of the ParaVision system discussed below would be mathematical students with substantial computer science expertise. For systems of evaluating languages for animation of sorting algorithms, the type of the users was fixed similarly.

The second approach was intensively used in the construction of the subsystem "the user model". It was assumed that this model is similar to the model of the learner in intelligent teaching systems. This model is constructed in the process of teaching, and it represents the competence of the learner in the subject being studied, his or her personal characteristics, and the history of his or her communication with the teaching system. In our case, when the user model is constructed, it is important to consider the opinion of the user about the visual system, namely, his or her specific knowledge about the system, general knowledge, subjective estimate, and emotional attitude. For the given implementation of the model of the user, the following parameters were distinguished:

- perception speed;

- computer expertise;

- emotional appeal;

- capability of visual thinking;

- fitness to work.

In principle, the subsystem "the user model" was designed to obtain an environment adjustable to the user needs and capable of adjusting itself during operation. Its prototype implementation served for detecting the aspects of the model that, in our opinion, make the greatest impact on how the user evaluates the quality of visualization. At this stage, significant attention was paid to processing the results of the experiment. The results of the prototype implementation of the subsystem "the user model" served as the basis for implementation of other systems evaluating the quality of visualization metaphors and languages.

6.2. Experimental Evaluation of Visualization Languages

As part of the further research, we implemented a prototype system evaluating the perception quality of two visual languages with similar vocabularies and semantics, but different syntaxes. The users were presented with a model problem of delivering goods from producers to customers. The first language was a static iconic language; the second one was based on using animation images. The user was given a description of delivering various goods between several places. Goods may be delivered by car or by train. Goods differ in color and type of 2D primitives used for their visualization. After showing the next description, the systems asked the user to answer questions about the type of the goods, direction, and method of delivering. The system records the answers of the user.

During an experimental session, the following characteristics were evaluated and registered:

(a) perception quality;

(b) perception speed;

(c) aesthetic and emotional satisfaction.

Perception quality is defined taking into account the number of correct answers of the user about the list of delivered goods, producers, and customers shown during the experimental session.

Perception speed is defined by registering the total time of pauses made by the user during the experimental session.

Aesthetical and emotional satisfaction is defined by registering the next choice of the user. At the beginning and at the end of the experimental session, nine semiabstract human faces expressing various degrees of good, neutral, and bad mood are shown in random order. It is assumed that the user chooses the face reflecting his or her emotional state before and after the session.

The ideas realized in this subsystem and the results of experiments carried out with this system were used in the development of the ParaVision system for evaluating the quality of languages for visualizing parallel computations. The ParaVision system is described in the next subsection.

6.3. The ParaVision System

The ParaVision project [1, 6] is an attempt to develop software for evaluating adequacy in visualization. It was aimed at studying metaphors and languages used for visualization of parallel computing in order to be able to monitor and debug them. It was assumed that, offering a model problem to a large (and homogeneous) group of testers, we compare the average time it takes to solve the problem and determine which metaphor is the most suitable for visualization.

For the model problem in the ParaVision system, we took the classical problem of parallel programming, namely, that of dining philosophers, which helps illustrating many problems occurring in visual programming [3]. These problems include, e.g., the problem of shared resources, the problem of dead ends, and the problem of infinite postponement. In this situation, philosophers represent parallel processes and two forks used by philosophers to eat macaroni are shared resources.

The problem of dining philosophers was also used as a model problem for the Paradocs system of automated visualization of parallel computing [29]. Visual representations of Paradocs partly served as a model for one of the metaphors of the ParaVision system.

The ParaVision system realizes three visual metaphors for representing this problem.

The first metaphor is natural. The operation of the system is shown by a computer animation. Philosophers, forks, and states of the philosophers (waiting, thinking, eating) are represented by animated pictograms carrying natural images. When the state changes, both the pictogram of the state and the facial expression (!) in a philosopher pictogram change.

The second metaphor is symbolic. Philosophers and forks are represented by circles. Each philosopher is represented by an individual color. The processes of taking forks and eating are pictured by stripes.

The third metaphor is based on Gantt charts. The activity of each philosopher (process operation) is represented by means of a sequence of horizontal stripes. The color of a stripe indicates the state of the process. The number of the process is laid off on the vertical axis, and the time is laid off on the horizontal axis.

ParaVision makes it possible

- by the example of a model problem, to learn a number of basic notions of parallel programming, explore operation of a parallel system and some standard situations occurring in the process of operation of a parallel program that the developer has to face;

- to consider the same situation by means of various visual metaphors;

- to interfere in the operation of the program modifying the lengths of process phases;

- to test the user.

For our purposes, the latter feature is the most important, since a significant part of the system activity is gathering statistics on the time required for solving the same problem with different metaphors. After appropriate processing, these statistics become data for analysis of the quality of visual metaphors.

The user starts working with the ParaVision system by acquainting oneself with the features of the system. First, the essence of critical situations occurring in the parallel program being illustrated is explained to the user. The user or the tester chooses an appropriate metaphor, for which an animation is shown illustrating the solution of the problem with parameters specified by the system. As the test proceeds, the user may modify the parameters of the system to avoid the situations of dead-lock (!) and infinite postponement.

Using ParaVision, statistical data were gathered that allow evaluating the quality of visual metaphors used in the system for the model problem. Two groups of users were tested to obtain data. One group (about twenty people) consisted of students specializing in Software Engineering with almost identical expertise in visual environments and basic knowledge of parallel programming. The other group (more than ten people) consisted of students and master students mainly concerned with mathematical modeling problems. Besides, a small group of schoolchildren of middle and upper grades worked with the system. Before testing, none of the users was acquainted with the ParaVision system. For the first group, the best results were obtained with Gantt charts, for the second one, with the natural metaphor. Note that some testers did not cope with the task, among them, a few students of the first group and about a half of the second group. All schoolchildren solved the problems. With the schoolchildren, only natural metaphor was used.

ParaVision turned out to be something like a computer game; nevertheless, the schoolchildren managed to learn some concepts of parallel programming.

After a series of experiments, its participants outlined their impressions. They can be summed up as follows:

(1) The first (natural) metaphor is colorful. Abstraction is at the lowest level, and, hence, it can be easily understood even by an unprepared person. However, the corresponding visual representation contains information redundant from the standpoint of specifying a parallel programming problem. (The participants of the experiment suggested the term "superfluous visualization".) Visual representation is excessively static; it poorly describes processes native to the problem. In the opinion of the participants of the experiment, this metaphor could hardly provide a foundation for analyzing the state of the computational process corresponding to the model. Nevertheless, this variant is appropriate for familiarizing oneself with concepts of parallel programming.

(2) The participants of the experiment claimed that the second quasi-natural metaphor is essentially identical to the first one. Although it is more dynamic, it has almost all the disadvantages of the first metaphor.

(3) Lastly, in the opinion of the participants of the experiment, the third metaphor based on Gantt charts is the most convenient and informative.

While the first and second metaphors concern only with the current state of the process, the third metaphor presents information both on the current state and on the process history; i.e., it is possible to monitor the behavior of the process, where and why dead-locks and infinite postponements appeared. Thus, this mode allows realizing one of the requirements to visual debugging, namely, the availability of information on the process at any point in time.

Analyzing attempts at obtaining numerical characteristics of adequacy in visualization, one may conclude that using only one parameter (the time spent on solving the problem) is by no means sufficient. Supplementary tools are needed for studying other characteristics of the visual language. Probably, it is worth turning back to evaluation of such additional parameters as perception quality and speed, emotional satisfaction of the user, etc.

The principles of ParaVision were elaborated in the process of developing several experimental visual systems using Gantt charts. Among them, there is a system providing the user with the possibility to choose the direction of drawing and the color palette of charts. Spiral charts turned out to be a successful solution. Of some interest are 3D variants of charts describing parallel processes similar to charts employed in the VisuaLinda system [9].

6.4. Evaluation of Methods for Sorting Algorithms Animation

In the development of subsequent versions of systems for visualization languages evaluation, we decided to provide the user with the possibility of choosing a language for describing concrete program entities. We had to create tools for construction of visual languages from a set elements, rules, and techniques for linking language phrases and meanings. For model application domain, we considered the animation of sorting algorithms, for users, high school and first year university students. (Interestingly, the researches [33] and [12] were based on similar material: the sorting algorithms animation was taken for the application domain, and the research was carried out with college students.) Thus, this system rigidly fixed both the set of entities to be visualized and the means of their visualization. However, since animation was realized for several versions of sorting algorithms, it was possible to use different visualization languages and techniques and to compare afterwards their visual expressiveness and adequacy in visualization. A distinguishing feature of this system is visualization of the algorithmic operation "*compare*", which remains unimplemented in most other systems, as it is considered evident for those who watch the animation .

Thus, we developed a visual learning system for the elementary course of lectures on algorithms. Its other goal, the main one for our purposes, was to allow evaluating the quality of visualization language used for representation of sorting algorithms. Two versions of the system were implemented: for high school and for college students.

In the simplified version designed for high school students, the user is able to change almost nothing during visualization; he or she only controls the process of visualization. While an algorithm operates, it is animated in the special window on the screen and the text of the program implementing the algorithm is displayed as well. In the text window, the current line is highlighted. An attempt was made to visualize the "*compare*" algorithmic operation: the elements being compared are distinguished by color and texture (diagonal grid). In principle, the features of this version are similar to those of traditional visual debuggers.

In the version for college students, the user may choose methods for representation of basic entities describing a sorting algorithm, e.g., color, size, and how to visualize the "*compare*" operation. Simultaneous execution of several sorting processes is possible, which allows one to compare the speed of algorithms.

We have made studies into which visual elements are most suitable for a particular sorting algorithm. The size of a rectangular stripe was chosen as the basic method for describing the value. We decided to use color only in experiments with animation of the exchange sorting algorithm. Specific works were done in choosing the color whose gradations would denote the value of the element. We considered gradations of gray, as well as the variant of using the primary colors (red, green, and blue); we settled upon the variant with gradations of gray as the most suitable for recognition by human.

In different variants of sorting algorithms, several visualization techniques were used for the comparison operation. As an example, consider "a corner with an additional beam", which approaches the objects being compared from the right, touches the largest object, and moves down; or "a harpoon", which shoots upward from the current object of the complete sequence to the next object of the input sequence. If it hits the target, the current object of the complete sequence moves one position up.

As the result of these researches, methods for visual representation of sorting algorithms adequate for using in teaching were studied.

Obviously, the systems for evaluating visualization quality that we developed and described in this section are nothing but prototypes. However, note that the main goal of this study was to try out methods for empirical evaluation of adequacy in visualization for a specific application domain and a fixed class of users.

7. DISCUSSION OF THE RESULTS

Some principles of visualization considered above could be summarized as follows: the visualization language must be built upon the fundamental idea of comparison called visualization metaphor, and adequacy in visualization should be the main parameter of evaluation of visualization metaphors and languages. This parameter must be accounted for in designing visualization systems. Now, let us consider implications of violating these principles either in design or in construction of methods for system evaluation.

Visualization metaphor is the foundation of visual representation, conceptual basis for comparison, measuring quality, and design of visualization systems. As it was already mentioned, visualization languages are impossible without metaphors. In this relation, it is interesting to discuss the studies into evaluating visualization quality [9]. The object of this paper, which has much in common with the experiments from Section 6.2, is to find out to what extent certain languages are self-evident and useful for representing model entities. The authors consider a language based on quite a strange metaphor (a scheme of a mechanism is represented by means of starfish and other inhabitants of the seabed), as well as a visual language that they believe is not based on any metaphor and is used to represent a typical technical scheme of a device. A series of experiments has been meant to clarify whether metaphor increases the *usability* of visual languages. Even the authors admit that the results obtained are quite arguable. In our opinion, the reason is that no theoretical basis was provided for evaluation of usability. That is adequacy in visualization that sets forth what should be compared, what should be measured, and what should be optimized. Narrow understanding of metaphor as everyday analogies results in far-fetched comparisons. Sometimes, systems constructed upon this basis are funny rather than useful for programmers.

An example of such funny metaphor is the metaphor of a hotel described in [34] and used in [25]. The hotel stands for the complete multiprocessor system. Rooms in the hotel represent individual processors. The contents of a room are suitcases, which represent the processes that are mapped onto the corresponding processor; processes can be moved from one processor to another. The last level of the hierarchy is made up by visual representation of the program. For this purpose, jigsaw puzzle pieces being the contents of suitcases are used. Puzzles consist of various elements from a structured set of shapes for indicating types of program constructions; these elements have a fixed sequence. Imagery used here is similar to that described in the 1980s [16, 24].

Our experience shows that it is doubtful that a programmer debugging a parallel program is able to use these cumbersome analogies, which look unnatural from the standpoint of his or her problem. However, these analogies are often ideally suited for purposes of teaching. Let us again indicate much richer metaphors used for teaching in researches and developments partially covered in [2]. There is no visualization equally suitable for all problems and all users. Nevertheless, the thorough study of currently used visualization systems and our experiments show that, in many cases of software and scientific visualization, the most appropriate are abstract visualization techniques such as Gantt charts for representing parallel processes or complex techniques for representing abstract objects in visualization of differential games [7, 8].

Neglect of pragmatic properties of visualization described by the parameter "adequacy in visualization" is responsible for failures of even serious works on software visualization. For instance, an interesting system for visualizing the history of developing and supporting versions of

OS UNIX platform software systems is described in [20]. Photo-realistic graphics is used for representation. However, experts in this field claim that the choice of visualization objects is unsuccessful, since these objects are insufficiently informative for software system developing and debugging.

The approaches to the definition of the notion of visualization metaphor that have been proposed in this paper make it unnecessary to follow every minute (and often accidental) similarity between entities, as it happens when metaphor is understood traditionally. At the same time, our approach, which allows for choosing metaphors and languages adequate in visualization, combines traditional understanding of metaphor of human-computer interface and other situations of using analogies between entities in visualization. Thus, in the development of specialized scientific visualization systems for problems related to optimal control and differential games, significant benefit was gained from taking into account adequacy in visualization in construction of abstract views of visual representation and techniques for interaction with visual objects [7, 8]. Specialized visualization systems supplying new views are indispensable at the stage of constructing computer models, because model objects under study often require new visual representation techniques. These problems arise also in designing specialized tools for scientific and information visualization for the purposes of biomedicine. Note that, working with current visual systems created for office use, medical professionals (as well as other professionals) have to employ alien metaphors and visual languages, which is quite acceptable for many applied problems. However, these representation techniques become insufficient and even unacceptable when they interfere with solving purely medical problems, e.g., an important problem of analyzing a large amount of "raw" data on patients and the run of the disease. A demand arises for complex views including, e.g., statistical graphic elements and natural images of the innards of the diseased. Let us indicate that metaphors and views drift from information visualization to software visualization systems. Recall that information visualization systems borrowed these views from statistical graphics.

8.CONCLUSION

In our earlier papers, we posed problems of formal description of adequacy in visualization and its dependence on the application domain and class of users of the visualization system [4]. Later, attempts were made to formally describe visualization metaphor as a mapping of the initial set of objects of the application model onto the target set of visual objects. In principle, a formal approach is possible to the process of generation of visualization metaphors and languages built upon these metaphors. There is an extensive literature on the formal theory of visual languages, their syntax and semantics. Important results can be found, e.g., in [14].

For designers of specialized systems, it is essential to create a technique for developing views consistent with ideas the actual user has about the nature of entities of the application domain, his or her purposes, and his or her methods for problem solving . In our opinion, besides powerful general-purpose visualization systems, specialized systems oriented at specific users and their problems are necessary.

Evaluation of adequacy in visualization rests on analysis of user preferences in choosing language means for problem solving of the given application domain. That is why new techniques based on results of psychological studies are needed for evaluating the quality of visualization systems.

New approaches to studying adequacy in visualization would help solving another important problem, namely, that of generating visualization metaphors and corresponding visual languages that support problem solving in the given application domain.

A vital issue is finding a body of mathematics providing proper formalization for visualization metaphors.

To conclude, let us outline a scheme of a visualization system. Development may proceed in the following order:

- analysis of the application problem and construction of a model of the application domain;

- analysis of specificity of potential system users and construction of a user model;

- selection or formalized generation of the visualization metaphor and language with properties specified;

- formal evaluation of the quality of the visualization metaphor and language from the viewpoint of the application domain and the user;

- construction of a prototype visualization system, experimental check of hypotheses for the quality of the language;

- refinement of the visualization language.

ACKNOWLEDGMENTS

I thank A.V. Konovalov and S.V. Sharf for interesting and seminal ideas, which gave the author and his students effectual help in studying visualization languages and metaphors, as well as techniques for their evaluation. Among the students of the Mathematics and Mechanics Faculty, Ural State University, taking an active part in the works on visualization, I would like to distinguish I.V. Tarskikh and V.V. Vorzopov, who played an important role in the research and development.

This study was supported in part by the Russian Foundation for Basic Research, project no. 99-07-90078.

REFERENCES

1. Averbukh V.L., Vorzopov V.V., Konovalov A.V. Evaluations of Visual Languages and Metaphors for Parallel Programming Visualization Systems // Yekaterinbourg, Institute for Mathematics and Mechanics Urals Branch of Russian Academy of Science (In Russian)

2. Samofalov V.V. Scharf S.V. The Visual Process: Design, Development and Application in Parallel Programming // Algoritmy i programmnye sredstva parallel’nyh vychislenii. (Ed. A. Kukoushkin and S. Scharf). Yekaterinbourg. IMM UrO Of RAN. 1995. pp. 170-181. (In Russian)

3. Hoare C.A.R. Communicating Sequential Processes. Printice-Hall. Englewood Cliffs. 1985.

4. Averbukh V.L., Konovalov A.V., Tarskikh I.V., Vorzopov V.V. Analyses of Visual Metaphors and Languages. Toward Prototyping of Software Visualizing Systems // "East--West". International Conference on Human-Computer Interaction EWHCI’94. St.-Petersburg, Russia, 2--6 August, 1994". Proceedings. ICSTI. Moscow. 1994. V.I, pp. 244--254.

5. Averbukh V.L. Toward Formal Definition of Conception "Adequacy in Visualization" // Proc. 1997 IEEE Symposium on Visual Languages, Sept. 23-26, 1997. Isle of Capri, Italy. - S.l.: IEEE Comput. Soc., 1997. p.46-47.

6. Averbukh V.L., Konovalov A.V., Vorzopov V.V. An Approach to Evaluations of Software Visualization // Human Factors in Computing Systems. CHI 97 Extended Abstracts. Atlanta, Georgia USA, 22-27 March 1997. ACM, 1997, p. 42.

7. Averbukh V.L., Kumkov S.S., Shilov E.A., Yurtaev D.A., Zenkov A.I. Specialized scientific visualization systems for optimal control application // Nonsmooth and Discontinuous Problems of Control and Optimozation: Proc. IFAC Workshop Chelyabinsk, Russia, 17-20 June, 1998.- New York etc.: Pergamon, 1999.- P.71-76.

8. Averbukh V.L., Kumkov S.S., Patsko V.S., Pykhteev O.A., Yurtaev D.A. Specialized Visualization Systems for Differential Games // Progress in Simulation, Modeling, Analysis and Synthesis of Modern Electrical and Electronic Devices and Systems / N.E.Mastorakis (ed). S.L.: WSES Press, 1999. P.301-306.

9. Blackwell A. Green T.R.G. Does Metaphor Increase Visual Language Usability? // 1999 IEEE Symposium on Visual Languages VL’99, Tokyo, Japan, September 1999. http://www.cl.cam.ac.uk/~afb21/publications/VL99.pdf

10. Chang S.-K., Tauber M.J., Yu B., Yu J.-S. A Visual Language Compiler // IEEE Transactions on Software Engineering. Vol. 15 No 5 (May 1989). pp. 506-525.

11. Christensen H. A Software Development Environment based on a Geographic Space Metaphor http://www.daimi.aau.dk/~hbc/Ragnarok/gsm.ps.gz

12. Douglas S., Hundhausen Ch., McKeon D. Toward Empirically-Based Software Visualization Languages // Proceeding of VL’95. http://www.computer.org/conferen/vl95/talks/T11.html

13. Eisenstadt M., Domingue J., Rajan T., Motta E. Visual Knoledge Engeneering // IEEE Transactions on Software Engineering. Vol. 16 No 10 (October 1990). pp. 1164-1177.

14. Erwig M. Abstract Syntax and Semantics of Visual Languages // Journal of Visual Languages and Computing (1998) 9, Pp. 461-483.

15. Fleet D., Ware C. An Environment that Integrates Flying and Fish Tank Metaphors // Human Factors in Computing Systems. CHI 97 Extended Abstracts. Atlanta, Georgia USA, 22-27 March 1997. ACM, 1997, pp. 8-9.

16. Glinert E.P. Out of Flatland: Toward 3D Visual Programming // Explor. Technol.: Today and Tomorrow. Fall Joint Comput. Conf. Dallas. Tex. Oct. 25-29. 1987. Proc." Washington D.C. 1987. Pp. 292-299.

17. Johnson G.J. Of Metaphor and the Difficulty of Computer Discourse // Communication of the ACM. Vol. 37, No 12 (Dec. 1994). Pp. 97-102.

18. Kiper J.D., Howard E., Ames Ch. Criteria for Evaluation of Visual Programming Languages // Journal of Visual Languages and Computing (1997) 8, Pp. 175-192.

19. Koike H., Takada T., Masui T. VisuaLinda: A Framework for Visualizing Parallel Linda Programs // Proceeding 1997 IEEE Symposium on Visual Languages. September 23-26, 1997 Isle of Capri, Italy. Los Alamitos, Ca. IEEE Computer Society. 1997. pp. 174-178.

20. Koike H., Chu Hui-Chu. How Does 3-D Visualization Work in Software Engineering?: Emperical Study of a 3-D Version/Module Visualization System // Proceedings of the 20th International Conference on Software Engineering. 19 - 25 April 1998. Kyoto, Japan. http://computer.org/proceedings/icse/8368/8368toc.htm

21. Leung Y. K., Apperley M. D. A Review and Taxanomy of Distortion-Oriented Presentation Techniques // ACM Transaction on Computer-Human Interaction. Vol. 1, No 2. June 1994. pp.126-160.

22. Madsen K.H. A Guide to Metaphorical Design // Communication of the ACM. Vol. 37, No 12 (Dec. 1994). Pp. 57-62.

23. McCormick, B.H., T.A. DeFanti, M.D. Brown (ed), Visualization in Scientific Computing // ACM Computer Graphics Vol. 21, No. 6, (Nov. 1987).

24. McDonald N.H. Video-Graphic Query Facility for Database Retrieval // Computer Graphics. Visual Technology and Art. Proceeding of Computer Graphics Tokio’85. Tokio. Springer-Verlag. 1985. Pp. 229-243.

25. Musil S. Monitoring Parallel Programs with INHOUSE http://www.ani.univie.ac.at/ani/research/Monit.html

26. Nardi B.A., Zarmer C.L. Beyond Models and Metaphors: Visual Formalism in User Interface Design // Journal of Visual Languages and Computing (1993) No 4, pp.5--33.

27. Petre M. Why looking isn’t always seeing: readership skills and graphical programming // Communications of the ACM. Volume 38, No. 6 (June 1995) pp. 33-44

28. Petre M., Price B.A. "Why a Computer Interface is Not Like a Painting: the user as a deliberate reader"; East-West HCI’92: The St. Petersburg International Workshop on Human-Computer Interaction, Juri Gornostaev, Ed., 1992, pp 217-224.

29. Price B.A., Baecker R.M. The Automatic Animation of Concurrent Programs// "East-West". International Conference on Human-Computer Interaction. Proceedings of the EWHCI’91 held in Moscow, Russia, 5-9 August, 1991." ICSTI. Moscow. 1991. pp. 128-137

30. Repenning A. Agensheets: A Tool for Building Domain-Oriented, Dynamics, Visual

Environment // Ph.D. Dissertation. University of Colorado at Boulder. Department of Computer Science. CU-CS-693-93. Dec. 1993.