A System for Big Attributed Hierarchical Graph Visualization

VICTOR KASYANOV, TIMUR ZOLOTUHIN
Institute of Informatics Systems
Novosibirsk State University
Novosibirsk, 630090
RUSSIA
kvn@iis.nsk.su

Abstract: - Information visualization is a process of transformation of large and complex abstract forms of information into visual form, strengthening cognitive abilities of users and allowing them to take the most optimal decisions. A graph is an abstract structure that is widely used to model complex information for its visualization. In this paper, we consider a system aimed at supporting of visualization of big amounts of complex information on the base of attributed hierarchical graphs.

Key-Words: - Attributed hierarchical graphs; big graphs; information visualization; visualization systems.

1 Introduction

The current state of computer science and programming can not be imagined without graph theory methods and algorithms. The wide applicability of graphs is due to the fact that they are a very natural means of explaining complex situations on an intuitive level. These advantages of representing complex structures and processes with graphs become even more tangible if there are good means of visualizing them. Therefore, it is no coincidence that in the world there is growing interest in methods and systems of visualization of structured information based on graph models [1], [2], [3], [4].

At present, visualization based on graph models is an integral element of processing complex information on the structure of objects, systems and processes in many applications in science and technology. The market is full of high-end software products that use information visualization methods based on graph models. Basically these are numerous specialized systems or embedded components of systems that are aimed to visualize certain subclasses of graph models and / or special applications, but there are also some universal systems (e.g., Higres [6], aiSee [7], yEd [8], Cytoscape [9]) which can be used for visualization of graph models of general type and wide use.

Since the information that it is desirable to visualize is constantly growing and becoming more complicated, more and more situations arise in which the classical graph models cease to be adequate. More powerful graph formalisms are required to represent information models that have a hierarchical structure. Hierarchy is the basis of numerous methods of analysis and synthesis of complex information models in various areas of application of computer systems. To represent a hierarchical kind of diagramming objects, some more powerful graph formalisms have been introduced, e.g. compound digraphs [9] and the clustered graphs [10]. The compound digraph is an extension of an undirected graph in which inclusion relation between nodes of directed graphs can be defined. A clustered graph is an extension of an undirected graph in which hierarchy can be represented by recursive partitioning of the graph into disjoint subgraphs. The compound and clustered graphs are relatively general graph formalisms that can handle many applications with hierarchical information, and are amenable to graph drawing.

The size of the graph model to view is a key issue in in information visualization based on graph models [2], [4]. Large graphs pose several difficult problems for visualization. As a rule, existing visualization systems are aimed to visualize relatively small graphs and can not be effectively used when it is required to work with graphs of large size of real applications. Another weak point is that usual systems for graph visualization do not have a support for many different attributes of graph elements. The standard situation for graph visualization systems is to have one text label for each node and, optionally, for each edge.

The hierarchical graphs are an extension of cluster and compound graphs and can be used in many areas where visualization of complex and

large amount information is needed [11], [12]. In this paper, the Visual Graph system intended for visualization of complex structured large amount of information on the basis of attributed hierarchical graph models is considered.

The rest of the paper is structured as follows. Section 2 gives a general description of application area of the Visual System. Section 3 describes its user interface and tools. In Section 4 some implementation features of the Visual System are outlined. Section 5 is our conclusion.

2 Application Area

The Visual Graph system has been developed to visualize the internal data structures typically found in compilers and other programming systems. Data structures that occur in these systems are usually represented as attributed graphs of big size. For example, the attributed syntax trees are used as the internal representations of translated programs in almost all compilers or interpreters. Optimizing and restructuring compilers require static analysis of control and data relationships in a program and their presentation in the form of a more general graph model of the program, for example, such as the control-flow graph and the data-flow graph.

It is assumed that the Visual Graph system is used as follows (Fig. 1). First, a compiler (or another programming system) itself or with an auxiliary program transforms a graph model arisen during compiling a source program from its internal representation into a file of one of the formats supported by the Visual Graph system, usually into the GraphML-file [13]. Then the Visual Graph system will be able to read this graph model from the file, to visualize it and to provide a user with different navigation tools for its visual exploration.

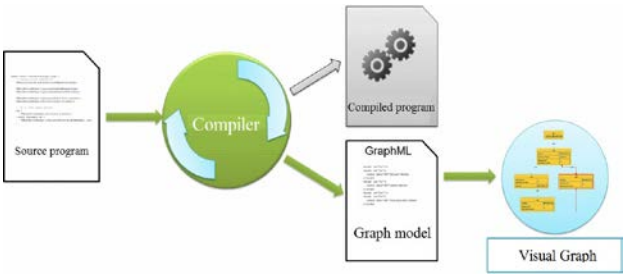

Fig. 1. Application of the Visual Graph system.

3 User Interface and Tools

The user interface of the Visual Graph system is shown in Fig. 2. It includes desktop, navigator, minimap and attribute panel.

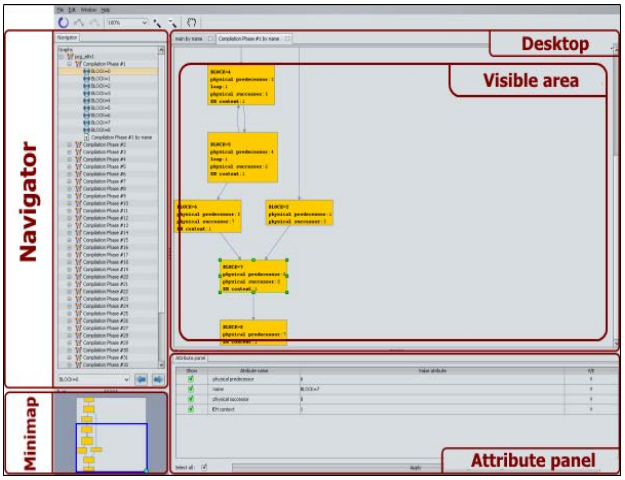

Fig. 2. User interface of the Visual Graph system.

3.1 Desktop

Desktop consists of a set of tabs that are opened by a user to visualize the selected portion of the graph model as its image on the plane. To improve the image automatically obtained, the user can change easily the shape of nodes and edges, the layout and its settings, the representation of attributes, the scale of the visible area and more.

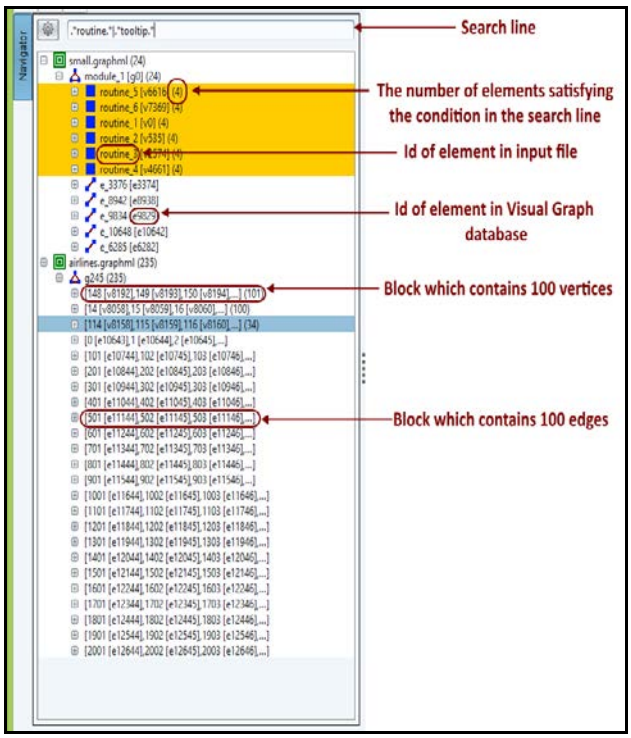

Fig. 3. Navigator of the Visual Graph system.

3.2 Navigator

Navigator is intended for visualization of all graphs with which a user is working as an image where nesting trees of fragments of these graphs are represented via indentations (Fig. 3).

To quickly search on the trees, a search line was implemented, which allows the user to use regular expressions. After search, the user can select the interesting elements and open them in a new tab. The user can also specify a set of attributes that will be visible for these elements. To work with big graphs navigator can download their necessary elements dynamically every time, when the user uncollapses an element containing inner elements. If the number of inner elements is too large, the elements can be combined in blocks. Each block contains no more than 100 elements.

3.3 Mini-map

Mini-map is a tool that allows a user to observe the entire graph shown in a current tab of the desktop, and also to move and to zoom its visible region, i.e. such a part of the graph which is visible in the current tab.

3.4 Attribute Panel

Attribute panel is a tool that allows a user to control the visualization of attributes for the selected nodes and edges in the current tab. For this purpose the user has to select from the graph of the current tab those nodes and edges that are of interest to his/her and then to note in the attribute panel those attributes that he/she wants to visualize for these elements.

3.5 Filter

Filter is a tool, which supports the search in the current tab all elements (nodes and edges) of the graph model according to some conditions. The conditions are described by the user using the attribute names and values (Fig. 4). User-specified conditions can be combined into expressions by means of logical operators and parentheses. The result of the filter work is the set of all those elements of the graph model that satisfy the specified conditions (Fig. 5).

3.6 Notepad

Notepad is a tool that allows a user to download additional information in the form of text files and link it with the graph model. The user can cross from the graph model elements to the associated additional text information and backwards. For example, notepad can be used for connection of a syntax tree with the source code.

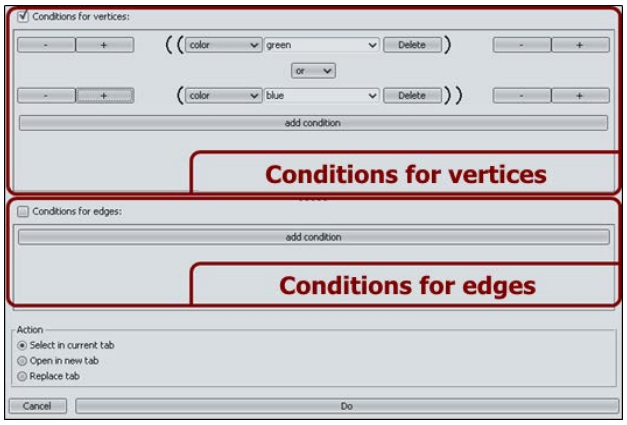

Fig. 4. Filter of the Visual Graph system.

3.7 Structural Analysis Tools

Structural Analysis Tools include a variety of algorithms for graph model, which help the user to select and visualize the information he needs in their images. These include tools for finding the shortest path between two given nodes, maximal strongly connected subgraphs containing a given node, all paths between two given nodes, immediate dominator for a given node, and largest common subgraph of two graphs.

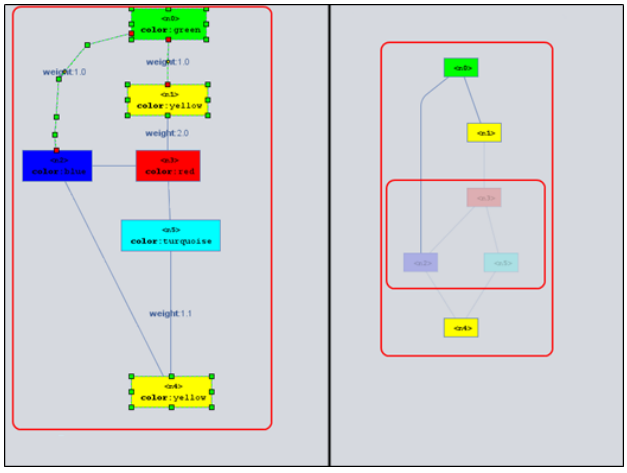

Fig. 5. Screenshot of the filter's result

4 Implementation Features

4.1 Storage of Big Graph Models within the System

File size with an input graph model can reach hundreds of megabytes, which prevents the use of the computer's memory to store graphs rendered in the system. It was therefore decided to use the caching data to the hard disk, using a relational database. The embedded SQLite database [14] has been chosen as a relational database. Unlike most popular relational database, the SQLite does not require installing a server, and the client-server architecture is reduced to work with files.

In the Visual Graph system a part of the data is contained in the database and another part is contained in the main memory, and thus high speed of working with large graph models is achieved.

4.2 Reduction of Layout Time

Visual Graph was designed to explore large graphs that consist of many hundreds of thousands of elements. However, our layout algorithms have nonlinear time, and so the layout of large graphs may require considerable time.

There are two main ways to speed up the layout algorithm in the Visual Graph system: multi-aspect layout of graph and control of layout algorithms.

The first way of time reduction is due to the fact that the Visual Graph system never tries to build an image of the entire graph, and is limited to building only drawings of those parts of it that are currently interesting to a user. Therefore, before the user starts to consider the graph, the system can do nothing, and thus get rid of the complex task of building the layout of the entire large graph, which is solved first in the usual graph visualization system. Later, when the user starts interacting with the system and shows interest in those or other parts of the graph, the layouts of the corresponding parts of the graph are constructed per demand. As a result of this interaction, a so-called multi-aspect layout of the graph takes place, which is a certain set of layouts of relatively small parts of the graph model. For presentation of multi-aspects layout a set of tabs which includes a separate tab for visualization of each considered part of the graph model is used.

Thus, during the interaction of the user with the system only when the user's desire arises to consider this or that part of a graph model, the application of the layout algorithm to the corresponding part of the graph model takes place. To form the interested part of the graph model the user can select its elements in the current tab or in the navigator. The user can also define some condition either in the filter or in the search line of navigator. Then this condition will be used for selection of all graph elements which will form the interested part of the graph mogel. The search of all graph elements satisfied to the condition can be performed either locally (in some subgraph, e.g. through a subgraph presented in the current tab) or globally (around the entire graph). As a rule, a multi-aspect layout of a graph model makes every visible part of the graph model smaller, thus enabling its layout to be calculated faster and the quality of its layout to be improved.

The other way of time reduction is connected with the possibility to control the layout algorithms by user. For example, some phases of the layout algorithm can be omitted by the user, or the maximum number of iterations of some phases of the layout algorithm can be limited. However, this way can usually decrease the quality of the graph layout. But it is possible to solve this problem in the Visual System, since the user can always attempt to improve the automatically obtained layout by hand, e.g. by moving of nodes or by changing of their sizes or forms.

4.3 Expandability of the System

All the features of the Visual Graph system, including possibilities for navigation, visualization and structural analysis, are implemented (and are available to users of the system) using a set of tools, which can be easily extended by both the developers of the Visual Graph system and any third-party developers.

To achieve a simple expandability of the Visual Graph system it was decided to use the Apache Felix product [15], which, in turn, is an implementation of the OSGi specification [16]. This solution is the de facto standard for this type of tasks and allows different developers to easily extend the system by writing new plug-ins.

5 Conclusion

The Visual Graph system intended for visualization of big amounts of complex information on the basis of attributed hierarchical graph models was considered.

Unlike its analogues (such as aiSee [5], yEd [6], Cytoscape [7] or Higres [17]) the Visual Graph system has the following important properties. It supports the processing of arbitrary attributed hierarchical graphs (including compound and cluster graphs) and the using for specification of the input (visualized) graph model the standard GraphML language. The system makes a multi-aspect layout of graph model that consists of separate drawings of only such its fragments which have been interested for user during graph model consideration and constructed on demand. The Visual Graph system provides also rich opportunities to navigate through big graph model, to make its structural analysis and to work with the attributes of its elements, as well as to extend and customize easily the system to specific needs of a concrete user.

At present the Visual Graph system is focused on the visualization of data structures arising in compilers, can simultaneously work with them both in graphical and in text forms, and provides smooth performing of the basic operations on graphs with up to 100000 elements (nodes and edges).

Its successful test use had been in the Intel Company.

It is also worth noting that using the Visual Graph system is not limited to the visualization of the internal data structures arising in compilers. It can be applied in other related fields which require graph visualization and navigation. For example, the system is used now as a base of visual debugging tool of a parallel programming system CSS which is under development for supporting cloud supercomputing on the base of the Cloud Sisal language [18]. The CSS system uses the attributed hierarchical graphs for internal representations of Cloud-Sisal-programs and provides means to write and debug Cloud-Sisal-programs on low-cost devices as well as to translate and execute them in clouds [19]. So, the system can open the world of parallel and functional programming to all students and scientists without requiring a large investment in new, top-end computer systems.

This work is supported in part by the Russian Foundation for Basic Research under grant RFBR 18-07-00024.

References:

[1] G. DiBattista, P. Eades, R. Tamassia, I.G. Tollis. *GraphDrawing: Algorithms for Vizualization of Graphs*, PrenticeHall, 1999.

[2] I. Herman, G. Melançon, M.S. Marshall. Graph visualization and navigation in information visualization: a survey, *IEEE Trans. on Visualization and Computer Graphics*, Vol. 6, 2000, pp. 24 - 43.

[3] V.N. Kasyanov, V.A. Evstigneev. *Graphs in Programming: Processing, Visualization and Application.* St. Petersburg, BHV-Petersburg, 2003. (In Russian).

[4] V.N. Kasyanov, E.V. Kasyanova. Information visualization on the base of graph models. *Scientific Visualization*, Vol. 6, No. 1, 2014, pp. 31–50 (in Russian)

[5] aiSee homepage, http://www.aisee.com

[6] yEd homepage, http://www.yworks.com

[7] Cytoscape homepage, http://www.cytoscape.org

[8] Higres homepage, http://pco.iis.nsk.su/higres/

[9] K. Sugiyama, K. Misue. Visualization of structured digraphs, *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 21, No. 4, 1999, pp. 876-892.

[10] Q.W. Feng, R.F. Cohen, P. Eades. Planarity for clustered graphs, *Lecture Notes in Computer Science*, Vol. 979, 1995, pp. 213 - 226.

[11] V.N. Kasyanov. Hierarchical graph models and information visualization, In: *Proceedings of the 2012 Third World Congress on Software Engineering (WCSE 2012)*, IEEE Computer Society, 2012, pp. 79-82.

[12] V.N. Kasyanov. Methods and tools for structural information visualization, *WSEAS Transactions on Computers*, Vol. 12, No. 7, 2013, pp. 349 - 359.

[13] U. Brandes, M. Eiglsperger, J. Lerner and C. Pich. Graph Markup Language (GraphML), In: *Handbook of Graph Drawing and Visualization*, CRC Press, 2013, pp. 517 - 541.

[14] SQLite homepage, http://www.sqlite.org

[15] Apache Felix homepage, http://felix.apache.org

[16] OSGi Alliance homepage, http://www.osgi.org/Main/HomePage

[17] I.A. Lisitsyn, V.N. Kasyanov. HIGRES - visualization system for clustered graphs and graph algorithms, *Lecture Notes in Computer Science*, Vol.1731, 1999, pp. 82 - 89.

[18] V.N. Kasyanov, E.V. Kasyanova. Methods and Tools of Parallel Programming. *CEUR Workshop Proceedings*, Vol. 1839, 2017, pp. 141-154.

[19] V.N. Kasyanov, E.V. Kasyanova. Graph- and cloud-based tools for computer science education, *Lecture Notes in Computer Science*, Vol. 9395, 2015, pp. 41 - 54.