Validity of Metaphors and Views of Software Visualization for Parallel Computing

Vladimir L. Averbukh[1,2]
[1]IMM UrB RAS,
[2]IMCS UrFU
averbukh@imm.uran.ru

Mikhail O. Bakhterev[1,2]
[1]IMM UrB RAS,
[2]IMCS UrFU
m.bakhterev@imm.uran.ru

Dmitriy V. Manakov
IMM UrB RAS
manakov@imm.uran.ru

ABSTRACT

In this paper approaches to the evaluation of Software Visualization for Parallel Computing are considered on the examples of representation of call graphs and execution traces of parallel programs. The concept of visualization metaphor is described. The visualization metaphors using to depict call graphs and execution traces are surveyed. The validity of visualization techniques is considered on basis of analysis of metaphor properties, Shneiderman's scheme and other approaches to the evaluation of metaphors and views.

Keywords

Software Visualization, Parallel Computing, call graphs, execution traces, visualization metaphors, Shneiderman's scheme

1 INTRODUCTION

Software Visualization (SV) systems were actively developed as late as the 80th and the 90th years of the XX century. Much part of these systems is visual systems for performance tuning and program debugging in the field of parallel computing. But later it can be observed a certain recession in this domain. The reasons of the recession are connected with a number of problems in particular connected with perception, analysis, and interpretation of images depicted huge volumes of data. At the very beginning of Software Visualization evolution, when volumes of data were comparatively inconsiderable, designers used standard "Nodes and Arcs" approaches. However, as early as the 80s more sophisticated views were used. These views were based on one or another metaphors or they were built on the basis of the figurativeness of applications under consideration. In some cases Software Visualization systems were provided with tools of design and drawing so that their users-programmers themselves can develop views for their needs. Ideas of performance tuning are based on the representation of statistics of parallel program executions. For these purposes statistical graphics, first of all, Gantt charts and Kiviat diagrams and their modifications are used. Note that complex views using various modifications of statistical charts to this day are the main most in the "industrial" systems of performance tuning and performance debugging [1], despite the obvious limitations when dealing with real high-performance programs. It appears that in the 90th the designers of visual debugging systems focused on the problems of capture of data in the frameworks (and under restrictions) of the then existing parallel computers. However, other problems exist, for example, the problems associated with visualization – how to choose and how to show entities of parallel programs, as well as to analyze and interpret them. In the case of parallel computation, the very definition of the program entities associated with its "erroneous" states is the tricky problem. The set and the essence of the analyzed software objects strongly depend on used parallel programming paradigm. In the case of performance tuning there is also a lot of problems, because the entities appropriated for analysis are hard to choose. Further in 2000th and 2010th years visualization metaphors were actively used. However use of interesting metaphors was not always clear as users often need the simple picture representation which could be accurately interpreted. In this regard, the important issue is the choice and the evaluation of visualization techniques, the analysis of their applicability to those or other cases. Designers need to evaluate visualization techniques basing both on the qualitative analysis and some formalization. There are various parallel program entities that are subjects of study, analysis and visualization. But in this paper, only the program execution traces and call graphs are considered, although they are not the only software analysis techniques.

2 VISUALIZATION OF EXECUTION TRACES AND CALL GRAPHS

From the beginning of the development of Software Visualization the question of how to graphically represent program entities came up. System designers relied on the standard ("paper") approaches to software visualization (for example control-flow diagrams, etc.) or shifted the task to users-programmers, providing a graphical toolkit. There are serious problems of scal-

ing the views based on one or the other diagram and charts. The next step in Software Visualization may be associated with metaphor using. But using the original metaphor was not always justified because of the need of simple but clearly interpretable presentation methods. "Graph-based" metaphors have significant limitations in this regard also. Execution traces and call graphs are used in one form or another by many debugging systems to describe the dynamics of the programs. Execution traces (also the term "Event Traces" is used) map the dynamics of the certain program executions. Visualization and "replaying" of execution traces are an important element of debugging systems. The visual presentations of the call graph are widely used in the systems parallel program performance tuning systems. In debugging systems realized in the 80s-90s of XX century methods based on charts or diagrams were used. Accordingly, traces are visualized in the form of dynamic "jumping" on the diagrams/charts. There was also the use of "passage" (or "running") on the text of a program highlighting the current position. In the case of high-performance computing, such visualization techniques are hardly suitable. Call Graphs also were represented as very complex and convoluted schemes. Now visualization metaphors were actively used. The metaphor essence consists in interpretation and experience the phenomena of one sort in terms of the phenomena of other sort. Metaphorization is based on interaction structures of source and target domains. During process of metaphorization some objects of target domain are structured on an example of objects of target domain and there is a metaphorical mapping (projection) of one domain onto another. That is the metaphor can be understood as a map from source domain onto target domain, and this map is strongly structured. There are many interesting examples of metaphor using for visualization of execution traces call graphs in parallel programs. For example such metaphors are used as Building metaphor [2], City and Landscape metaphors [3, 4], Dynamic Systems metaphor, [5], Molecule metaphor [2] metaphor. Also less customary metaphors for example Brain Metaphor [6] and Circular Bundle metaphor [7, 8] were suggested.

Views are designed on the basis of the metaphors. A view includes a description of possible visualization objects, their relative positions on the screen, as well as the possible interaction with them. The consideration of specific tasks of debugging and analysis is needed during the phase of the view development. Views are designed on the basis of the metaphors. A view includes a description of possible visualization objects, their relative positions on the screen, as well as the possible interaction with them. The consideration of specific tasks of debugging and analysis is needed during the phase of the view development.

3 PROPERTIES OF VISUALIZATION METAPHORS

The success or failure of debugging and performance tuning systems depends on many factors. Of course, the important factors are the comprehension of correspondence to system specifications and the system reliability. However at the design stage, an important task is the choice of methods of visual representation of objects and entities to be considered during debugging. One approach to the evaluation of visualization involves the examination of properties of visualization metaphors. We analyze the properties to consider the possibility of metaphor using for specific applications of Software Visualization. It is important to understand what objects may be represented with one or another metaphor. We need to analyze the possibility of the visualization metaphors (more precisely – the views based on the visualization metaphors) to represent a large and huge volumes of data and details required to understanding the program's operations. The positive effects of a 3D display and virtual and augmented reality environments are possible in these cases. Therefore it is important to analyze possible applications of metaphors in the frameworks of visualization systems using modern computer graphics environment, in particular the virtual reality environment. For all this, we need to describe how to verify the suitability of metaphor for solving problems under consideration. Note on such metaphor properties as ***"ability to contain any objects inside itself", "restriction of a perception context", "closeness", "inclusion in structure", "presence a structure inside", "naturalness of a metaphor"***. In the cases of popular in Software Visualization systems City Metaphor and similar Landscape Metaphor one may consider the following properties as:

Unlimited context. The user context isn't artificially limited in City Metaphor and Landscape Metaphor. When visualization of large volumes of data is needed, unlimited context allows to have a look-see round the whole picture and to select the key places quickly.

Naturalness. It is known that naturalness of a metaphor reduces efforts on the resultant image interpretation. In the cases of City and Landscape metaphors not only naturalness of spatial orientation, but naturalness of navigation takes place also. In case of a city metaphor the method of navigation is defined by the metaphor itself.

Organization of inner structure. Metaphors suggest the existence of an inner structure. In case of a City metaphor this structure is dictated by the metaphor itself, and it is defined rather rigidly – there are buildings, quarters, streets, districts. In Landscape metaphor a structure choice is nondedicated. In this case one may say about landscape nesting.

Key elements. Metaphors suggest a representation of large volume of information, and in most cases this information is rather homogeneous in visual sense. Users need the key elements to interpret this information. If we want to use a metaphor to reveal specific features and/or exceptions (for example bugs in programs), these elements have to be depicted by easy distinguished image-keys. One may design some key elements in frameworks of City or Landscape metaphors. In these cases some forms of guidance signs or markers may be used as key elements.

Resistance to scaling. These metaphors are stable in the case of increase in information volumes. Moreover, applications of City and Landscape metaphors are reasonable only in the cases of large information volumes.

In the cases of City and Industrial Landscape metaphor *transport corridors* help to design software visualization systems. Transport corridors may be used as means to represent *control flows, data flows*, and other relations between program constructions or parts of program complex.

Note that unlike in the case of Landscape metaphor, the choice of City metaphor strongly limits the set of possible views.

Thus City and Landscape metaphors may form base to represent considerable volumes of the structured information with identifications of specific interest cases that is necessary in the systems for performance tuning and program debugging for parallel computing.

Additionally possibility to fly over a city/landscape creates prerequisites to easy navigation. Flight with changes of height allows to carry out scaling and zooming. Interpretation of the graphical displays based on these metaphors seems to be simple.

4 VIEW EVALUATIONS

A number of papers are devoted to evaluation of views used in Computer Visualization Most of these researches are linked with Information Visualization. The paper [9] contains the outline of visualization analysis based on Visual Information-Seeking Mantra (so called Shneiderman's Mantra). B. Shneiderman presents seven high level user needs that an information visualization application should support [10]. In [11] these needs were refined to evaluate views of three-dimensional information visualization. Let's cite the outline of visualization analysis following [9]: **Overview; Zoom; Filter; Details-on-demand; Relate; History; Extract**.

It is supposed that if the system supports this set of operations, it may be used for Information and Software Visualization. In [7, 8] the criterion based on an expanded Schneiderman's mantra is applied to the analysis of visualization of execution traces constructed on the basis of two synchronized views 1) a circular bundle view for displaying the structural elements and bundling their call relationships, and 2) a massive sequence view that provides an interactive overview. The summary table of how the two synchronized views satisfy each of these seven criteria is provided. In general these views correspond to the chosen criterion.

Let's analyze now from a perspective "Shneiderman's Mantra" the possibilities of visualizations based on City and Landscape metaphors.

Overview task may be realized by the flight over the city or landscape. **Zoom** task is implemented easily by lowering or lifting during the flight. The idea of the realization **Filter** task may be borrowed from cartography (and Information Visualization based on cartography techniques). There is the method of filtration on maps presented geographical data – to eliminate some types of information from the overall picture as for example human settlements may be eliminated from the map presented ground reliefs. The analogy of cartography shows that Landscape metaphor is preferred then City metaphor in the case of **Filter** task. **Details-on-demand** task, as well as **Relate** task may be implemented within the framework of the extended Room-Building-City metaphor by means "passing" down the street and "inputs" inside buildings and rooms. **History** and **Extract** tasks may realized naturally in frameworks of City and Landscape metaphors.

Schneiderman's scheme may be applied to evaluate Software Visualization in cases when corresponding systems are destined to be analyzed large volumes of abstract data similar to Information Visualization systems. In other cases these scheme is not applicable. Schneiderman's criterion is based on check of necessary, but not ampleness conditions of quality of Information Visualization. The use of the Schneiderman's scheme presupposes the existence of large structured data volumes. But in this case the resulted visualization has to be a manageable size. More importantly it is assumed that the user either knows what she/he searches or at least she/he is able to recognize it. In the case of "circular bundle" the new complicated abstract visualization technique (based on the new metaphor) is used. The user should always correlate the images with non-obvious representations of interesting data.

Similarly the visualization techniques based on using different charts and diagrams in many cases generate abstract and nontrivial representations. In the case of high-performance computing the methods of visualization for execution traces may be ineffective, due to the complexity of both the analysis of codes execution, and large data volumes. Such considerations can be applied for many new abstract methods (visualization metaphors) for complex data representation. The views using modification of statistical diagrams and

charts scale insufficiently. They can't map the execution of hundreds and thousands of parallel processes. Also let's state a remark about "natural" metaphors using. Interpretation of graphical displays implemented in the framework of the "natural" metaphors, for example, interesting "animation" Brain metaphor, often is not obvious. The naturalness of imagery in the cases of City and Landscape metaphors can sometimes distract users. Also, there are problems of perception and interpretation of large and huge data volumes. For example, the the flickering and blinking animation displays observation may be unusual and unpleasant. Users of systems based on virtual reality can have discomfort in the form of dizziness and so forth. There is the problem of selecting the objects to be visualized in the debugging process. In the case of parallel computation the definition of program objects associated with its "bug" states is a difficult task. Set and the essence of the analyzed program objects differ markedly in the various paradigms of parallelism. The execution trace is only one of possible entities subjected to analysis and, as a consequence, to visualization. In the case of performance tuning there is also no clarity with selection of entities which can help to improve performance.

5 CONCLUSION

In this paper approaches to the evaluation of Software Visualization for Parallel Computing are considered on the examples of execution traces and call graphs of parallel programs. The validity of visualization techniques was evaluated on basis of Shneiderman's scheme. Additionally cognitive approaches to the analysis of visualization and to the evaluation of the implementation efforts were considered. In the pre-design analysis the consideration of whole range of evaluations is necessary. Contradictions between representation opportunities and visualization perception, analysis and interpretation abilities of the users exist. Interesting metaphors can give pictures difficult to interpret or demand big efforts when developing. Scaling problem remains unsolved for many techniques of Software Visualization for Parallel Computing. This problem is related to fundamental limitations on placing "big pictures" on the screen, and the user perception and interpretation of "big data" generated by debugging and performance tuning systems.

6 REFERENCES

[1] Mohr, B., 2014. Scalable parallel performance measurement and analysis tools - state-of-the-art and future challenges. In Supercomputing frontiers and innovations. Volume 1. Number 2 (2014). Pp. 108-123.

[2] Averbukh, V.L., Baydalin, A.U., Ismagilov D.R.,, Kazantsev, A.U., Timoshpolskiy, S.P., 2004. Utilizing 3D Visualization Methophors. In Proceedings of 14 International Conference "Graphicon". 2004, Moscow, Russia. Pp. 295-298.

[3] Fittkau, F., Waller, J., Wulf, Ch., Hasselbring, W., 2013. Live trace visualization for comprehending large software landscapes: The ExplorViz approach. In Proceedings of the 1st Working Conference on Software Visualization (VISSOFT), IEEE Computer Society, 2013. 4 pp.

[4] Waller, J., Wulf, Ch., Fittkau, F., Dohring, Ph., Hasselbring, W., 2013. SynchroVis: 3D Visualization of Monitoring Traces in the City Metaphor for Analyzing Concurrency. In First IEEE Working Conference on Software Visualization (VISSOFT), 2013. 4 pp.

[5] Osawa, N., 1998. An Enhanced 3-D Animation Tool for Performance Tuning of Parallel Programs Based on Dynamic Models. In SPDP 98 Welches Or. USA. pp.72-80.

[6] Palepu, V.K., Jones, J.A., 2013. Visualizing constituent behaviors within executions. In Proceedings of the 1st Working Conference on Software Visualization (VISSOFT), IEEE Computer Society, 2013. 4 pp.

[7] Cornelissen, B., Holten, D., Zaidman, A., Moonen, L., van Wijk, J.J., van Deursen, A., 2007. Understanding execution traces using massive sequence and circular bundle views. In Proc. of the 15th IEEE Int. Conf. on Program Comprehension. IEEE, 2007, pp. 49-58.

[8] Cornelissen, B., Zaidman, A., Holten, D., Moonen, L., van Deursen, A., van Wijk, J.J., 2008. Execution Trace Analysis through Massive Sequence and Circular

[9] Maletic, J.I., Marcus, A., Collard, M.L., 2002. A task oriented view of software visualization. In International Workshop on Visualizing Software for Understanding and Analysis. 2002. Pp. 32-40.

[10] Shneiderman, B., 1996. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In Proceedings of the IEEE Conference on Visual Languages, September 3-6, 1996. Pp. 336-343.

[11] Wiss U., Carr, D., Jonsson, H., 1998. Evaluating Three-Dimensional Information Visualization Designs. A Case Study of Three Designs. In Proceedings of International Conference on Information Visualisation, London, England, July 29-31. 1998. Pp. 137-144.