Quality Assessment of Graph Drawing Sequences Representing Software Systems Evolution

D. Ivanov, H. Haralambiev, M. Lazarova, S. Boychev

Key Words: *Graph drawing; software evolution visualization; graph layout; measures.*

Abstract. *As a widespread approach for representing data, graph drawings are used for displaying both software systems and their evolution. Every version in the process of software evolution is represented by a different graph representation. Consequently, a proper layout algorithm is needed in order to retain the connection between different drawings and ease their parallel observation. The selected algorithm should be able to provide drawing of graph sequences which is stable and mental map preserving. This paper introduces visual and numerical approaches for assessment, observation and analysis of software systems evolution. Certain set of measures is selected and used for this task. For visual and numerical assessment of software system graph layouts a software tool is developed and used for experimental observation and evaluation of the measures by their application to the graph drawings of certain software projects revisions.*

1. Introduction

The quality appraisal of graph drawing is a key problem, concerning the understandability of the represented model. Most of the graph representations are used to display the sequence of model states. Therefore, the mental connection between these states should be as clear as possible. To assess the understandability, clearness and mental map preservation of this sequence of graph drawings, an appropriate set of measures, concerning these qualities, should be defined and used.

This paper focuses on observation of the graph drawings, which represent software evolution. Measures, used to form the drawing assessment, are also introduced. Several definitions of software evolution are suggested in [18], but in the context of the paper the most appropriate is Lehman and Ramil's definition: *„All programming activity that is intended to generate a new software version from an earlier operational version*". According to the source code versions, the corresponding graph is frequently changed. In this context the quality and stability of the layout is vital for the system state perception.

Some of the graph representations have various limitations, concerning the layout quality and stability over a numerous versions of the drawing. The appropriate representation of the software evolution is another general problem.

During the research for existing problems in the graph drawing algorithms the following software tools, having this kind of problems could be presented.

In order to represent the Kiviat graph of module coupling dependencies clearly, RelVis [16] attempts to prevent layout problems by using standard graph drawing algorithms, such as hierarchical or spring layout. Unfortunately, the graph layout gets complex when visualizing large number of source code entities and needs post processing steps or pre-filtering of relations. The quality of the layout varies in the different graph drawings depending on various subjective factors. This fact indicates the need of a formal assessment of the quality - exact measure values for each pair of drawings.

The Building Block Communications System [17] introduces other types of graphs, representing the Distributed Feature Composition model in an understandable way according to strictly defined rules. The layout of the graphs aims at symmetry, mental map preservation and effectiveness of the drawing. This effect is achieved by using various types of virtual physical model algorithms for graph drawing. Nevertheless, a formal assessment of the graph drawing quality is not provided.

The Evolution Storyboard [1] uses an energy-based layout algorithm to provide dynamical views of the software's evolution structure. In order to help the user to understand the system structure better, an animation shows the movement of nodes between consecutive time stamps. This feature assumes that every state of the represented graph uses a mental map preserving layout. Such layout could be easily chosen according to the values of properly selected measure.

GraphAEL[5] creates, represents and animates different states of several graph types. It uses a force-directed graph drawing algorithm to achieve mental map preserving and aesthetically pleasing layouts for series of graph drawings. The authors discuss several improvements of existing force-directed graph drawing algorithm, but the progress of the improvement process is not appraised adequately by a certain graph drawing measure values.

The observation of the software evolution requires appropriate representation of the data. Gonzalez, Theron, Telea and Garcia [8] propose a multilevel timeline view, supporting a set of tasks, useful for the project managers. In order to inspect the changes made along the revisions, the authors represent a set of source code metrics in charts.

The representation of the software system and its evolution is often made to provide additional information about system artifacts as well as a source code metrics, added or modified elements and the relations between them. In this context, the paper considers graph as the most appropriate representation of a software system. Each graph drawing algorithm is aimed at providing an easy to use, good and mental map preserving layout of the displayed data. To clearly define such aesthetic and logical criteria, formal approaches for the graph layout assessment are put in practice.

The purpose of the paper is to evaluate software evolution

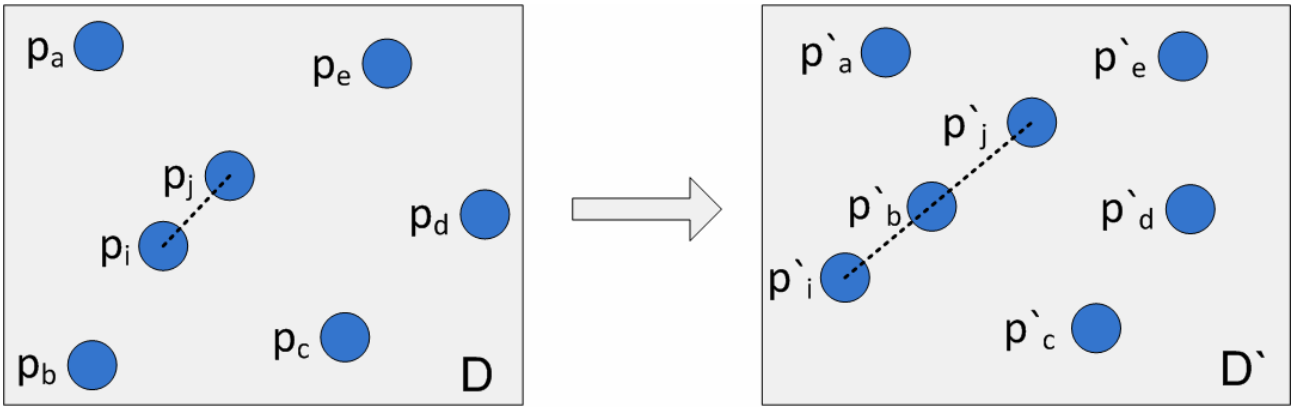

Figure 1. Nearest neighbor within

graph representations. To appraise the quality of a given graph layout, a certain set of measures is defined and used for software graph layout assessment. The selected measures are applicable for a pair of graph drawings. Experimental evaluation of the measures is made by their application to the graph layouts, which represent the software evolution of given software projects.

2. Graph Drawing Quality Measures

A set of graph layout measures is introduced in [3,13 and 2]. Four of these measures are selected for the purposes of numerical assessment of graph drawing sequences of software versions. The selection criterion for three of the measures is based on the empirical research, made in [2]. The expectance is only the ε-clustering which is considered appropriate measure for software graph drawings, despite the conclusions, given in [2]. All the measures selected are focused on measuring the quality and the stability of a given graph layout.

A graph G refers to a collection of vertices (or 'nodes') and a collection of edges that connect pairs of vertices. Following the terminology in [3], let define D and D' two drawings of the same graph G. Each object of G can be associated with two sets of coordinates, one describing the position in D and the other the position in D'. P denotes the set of pairs (p_i, p_i') where p_i and p_i' represent the location of the i-th in D and D', respectively. Let $d(p, q)$ be the Euclidean distance between points p and q in D as well as in D'.

2.1. Composite Distance to Neighbor

The composite distance to a neighbor measure calculates the overall impact of the weighted version of the nearest neighbor within measure for all the nodes of the graph.

The nearest neighbor within the measure, marked as *nnw* in [3], is a proximity measure. This measure type reflects the idea that points near each other in the first drawing should remain close to each other in the second drawing. The nnw measure is based on the reasoning that if p_i is the closest point to p_j in D, then p_i' should be the closest point to p_j' in D'. The weighted version of the nnw measure (*figure 1*) considers the points closer to p_i' and p_j' - if there are more points between p_i' and p_j', the visual linkage between p_i' and p_j' has been disrupted in a greater degree and the drawing looks more different (*figure 1*).

More formally, the nnw measure is described in the formulas below [3].

$$nnw(D, D') = \frac{1}{W|P|} \sum_{1 \le i \le |P|} closer\,(p_i', p_j')$$

where p_j is the closest point to p_i in D and

$$closer\,(p_i', p_j') = \left|\{k | d(p_i', p_k') < d(p_i', p_j')\}\right|, W = |P| - 2.$$

The distance is scaled by the number of points being considered. W is the maximum weight contributed by a single point, so that the measure's value is always in the range.

2.2. Ranking

Ranking [13] is example of a measure, which can be used for assessing the preservation of the mental map in graph drawings. The ranking measure considers the fact that the relative horizontal and vertical positions of a node should not vary too much according to the graph modifications (*figure 2*).

The formal description reads as follows: let *right* (p) and *above* (p) be the numbers of points to the right and above

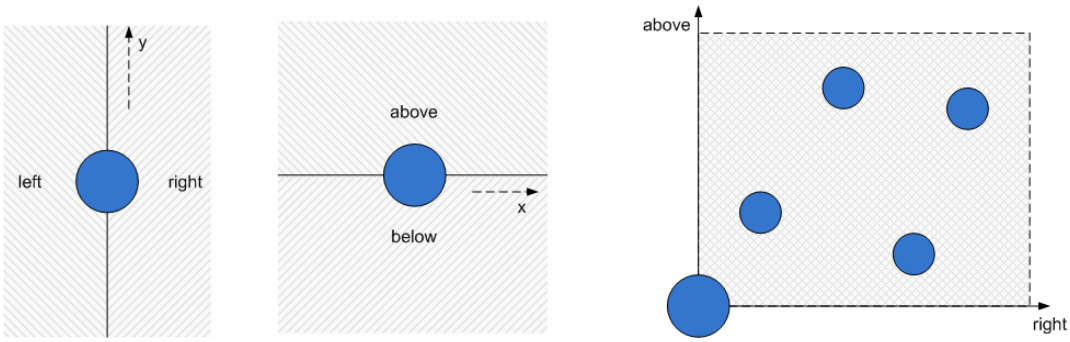

Figure 2. Ranking

of p, respectively (*figure 2*). λ is the normalizing factor. Then

$$rank\ (P, P') = \frac{1}{\lambda} \Sigma_{p \in P} \min\{|right\ (p) - right\ (p')| + |above\ (p) - above\ (p')|\ , \lambda\}$$

where $\lambda = 1.5\ (|P| - 1)$. The constant *1.5* is based on the experiments, reported in [3].

The value of the measure is inversely proportional to the preservation of the mental map, e.g. the lesser the ranking is - the more the mental map is preserved.

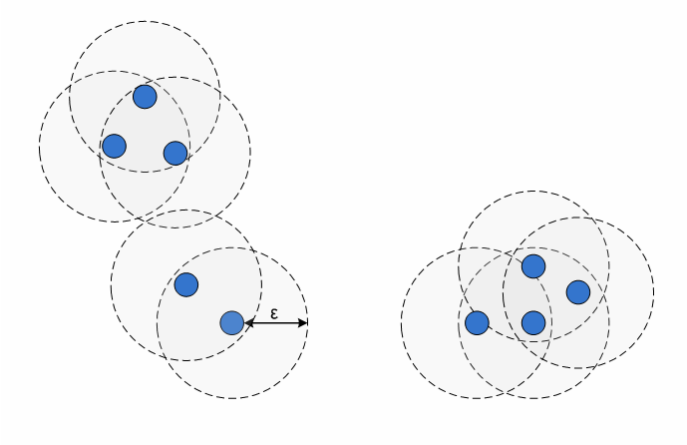

Figure 3. Epsilon clustering

2.3. ε - Clustering

Following [3] an ε-cluster for a point p is the set of points q such that $d(p,q) \leq \varepsilon$, where $\varepsilon = \max_p \min_{q \neq p} d\ (p,q)$. The idea of the ε-clustering measure (*figure 3*) is to compare the ε-cluster for p_i in D to that for p_i' in D'. Let E_{C_D} and $E_{C_{D'}}$ be the set of edges (i, j) if $d(p_i, p_j) \leq \varepsilon_D$ and $d(p_i', p_j') \leq \varepsilon_D$, respectively. The distance, considering all the removed and added edges, is thus:

$$distance\ (D, D') = \frac{|E_{C_D} \cup E_{C_{D'}}| - |E_{C_D} \cap E_{C_{D'}}|}{|E_{C_D} \cup E_{C_{D'}}|} = 1 - \frac{|E_{C_D} \cap E_{C_{D'}}|}{|E_{C_D} \cup E_{C_{D'}}|}$$

The distance is zero, if the edges between all vertices remain the same, and it is one, if there are no common edges.

2.4. Weighted Orthogonal Ordering

The orthogonal ordering measure appraises the mutual orientation of point pairs and focuses on assessing the preservation of their relative ordering. To clearly imagine the orientation, suppose every node has a compass rose on its center and the angles represent the directions [2] (*figure 4*).

Formally, if p_i is northeast of p_j in D then p_i' should remain northeast of p_j' in D' [3]. According to the level of mental map preservation precision, Bridgeman and Tamassia [3] suggest two weight functions - constant and linear. For more accurate layout assessment according to the angle diversion, below is described the orthogonal ordering measure with linear weight function.

$$order\ (D, D') = \frac{1}{W|P|} \sum_{1 \leq i,\ j \leq |P|} min(order\ (\theta_{ij}, \theta_{ij}'),\ order(\theta_{ij}', \theta_{ij}))$$

where:

1) θ_{ij} is the angle from the positive x axis to the vector $p_j - p_i$ (*figure 4*);
2) θ_{ij}' is the angle from the positive x axis to the vector $p_j' - p_i'$ (*figure 4*);
3) $order\ (\theta_{ij}, \theta_{ij}') = \int_{\theta_{ij}}^{\theta_{ij}'} weight\ (\theta)\ d\theta$;
4) $W = \min\left\{\int_0^{\pi} weight\ (\theta)\ d\theta,\ \int_{\pi}^{2\pi} weight\ (\theta)\ d\theta\right\}$;
5) $$weight(\theta) = \begin{cases} \dfrac{\frac{\pi}{2} - (\theta \bmod \frac{\pi}{2})}{\frac{\pi}{4}}, & if\ (\theta \bmod \frac{\pi}{2}) > \frac{\pi}{4} \\ \dfrac{(\theta \bmod \frac{\pi}{2})}{\frac{\pi}{4}}, & if\ (\theta \bmod \frac{\pi}{2}) \leq \frac{\pi}{4} \end{cases}.$$

3. Graph Drawing Sequences Assessment

The selected graph drawing measures are used for assessment of the graph drawing sequences. For this purpose a software tool called ReViewer is developed. ReViewer is used to visually appraise the benefit of the above described measures. By taking the XML descriptions of the graph drawings, ReViewer encapsulates the measures calculations and represents the result values in several views.

3.1. Creating Drawings Representation

ReViewer is developed to be a universal and cross-plat-

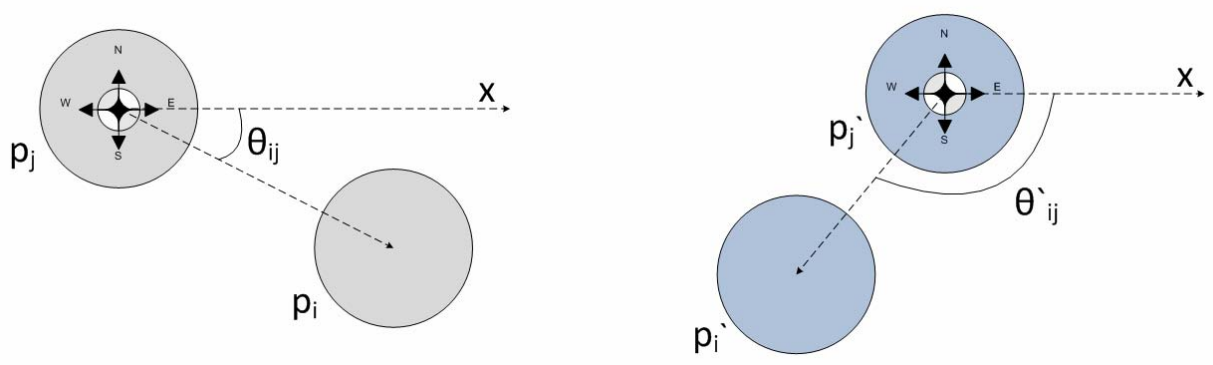

Figure 4. Orthogonal ordering

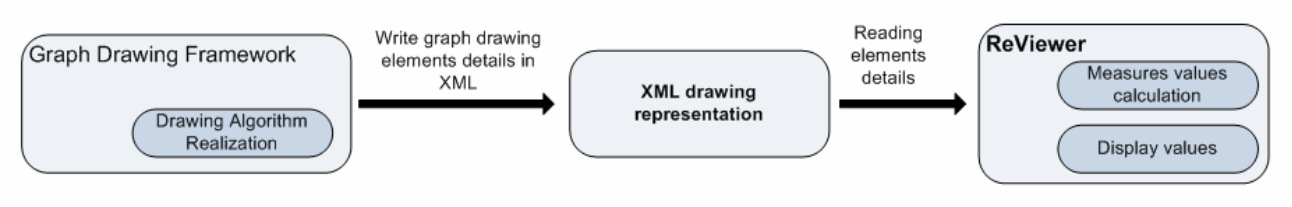

Figure 5. Data flow between the graph drawing framework and ReViewer

form toolkit (*figure 5*). It takes the graph drawing elements details described in an XML file as input. The stored XML data is used for the measure values calculations, which are then displayed in several views for graph drawing assessment, observation and analysis. Thus the ReViewer is independent from the used graph drawing framework, including the graph drawing algorithms, internally implemented in the graph drawing framework.

In order to represent the software evolution the source code from a number of revisions of several java open source projects has been used ([4,12,14]). The code for every revision is compiled and a Knowledge Discovery Model [11] is built. This kind of model represents the source code in an entity - relationship model (entities are packages, classes, methods, etc.; relationships - extends, implements, has type, etc.). Two graph drawing algorithms are used to represent the KDM models - Dynamic Fruchterman - Reingold [7,6] and Dynamic Force-Directed [9]. They are implemented in JUNG [15] and their drawings described in an XML files. These XML files are taken as an input for the ReViewer, and ReViewer only needs the XML files for calculating the measure values and displaying them (*figure 5*).

3.2. Displaying the Values of Measures

ReViewer uses several synchronized Eclipse views to clearly display the graph drawing measures values in an understandable way. By observing the represented data, more precise assessment of the layouts can be made. Since different practical problems require different features of the layout, appropriate graph drawing algorithm can be chosen based on the measures values observation.

3.2.1. Layouts Distribution View

The Layouts distribution view (*figure 6*) represents a chart for comparing given measure values for a single or multiple graph layouts. The revision numbers are shown on the x axis and the measure values appear on the y axis. Every layout has a seesaw line, which nodes represent the measure values for a given revision. In order to observe the software evolution, revisions are chronologically ordered. Since the measures are defined for two drawings of the same graph, every node in the chart represents the measure value for the drawings of the graph in the current and the previous revision.

This type of measure values representation provides the ability of exploring single or multiple layout measurements through the versions of the software evolution graph. In this context, the comparison of several layout measure values appears to be a possible approach for choosing the best layout, concerning certain practical problems.

3.2.2. Measure Values Details View

Measure Values Details view (*figure 7*) - consists of a couple of views, which display statistical information about a selected revision node - as a bar chart diagram and a numerical information.

3.2.2.1. Measure Values per Node

Most of the graph drawing quality measures focus on assessing certain criteria for each node of the laid out graph. For that reason a view of the particular node measure values is a useful tool for observing the quality of the entire layout. The *Measure values per node view* (*figure 7*, left) provide such information, sorted by the measure value. Every node measure value is represented by an adequately scaled bar. The node IDs are displayed on the x axis of the chart. Because of the large number of nodes, the IDs are not always visible. The measure values are shown on y axis.

This data representation shows the distribution of the measure values along the whole node set of the laid out graph and facilitates the node set data analysis. Moreover, precise observations of the variations in the measure values can be made in order to detect the optimal distribution for a graph representation.

3.2.2.2. Description View

The Description view (*figure 7*, right) provides statistical information about the modifications made in the currently compared couple of graph drawings. This information eases the process of analysis of the layout measure values displayed in the *Layouts distribution view* - the relationship between the graph drawings and their measure values is ensured by the views synchronization.

3.2.3. Layout Differences View

The Layout Differences view (*figure 8*) displays the visual representation of the graph in a selected state of the evolution of the software system. In order to facilitate nodes tracing among the revisions, the view provides an animation effect (*figure 10*) - smooth transition between the currently analyzed states of the system graph representations.

To follow the direction and the distance of the movement, every node leaves a visible track with proportionally calculated size and displacement step length (*figure 10*). The fine tracing is provided by step by step control, animations steps and delay adjustments (*figure 8*).

Basically following the concept of information consistency, the *Layout distribution view* highlights (*figure 9*) a node selected in the *Measure values per node* view. This approach allows an easier detection of the location of a single node or a group of nodes determined by their measure value or position in the common measure values distribution.

Three types of node states appear as a result of the comparison between the graph drawings - added, changed (modified), unchanged. To clearly separate this types of nodes, every type is displayed with different color (*figure 10*, picture 2)

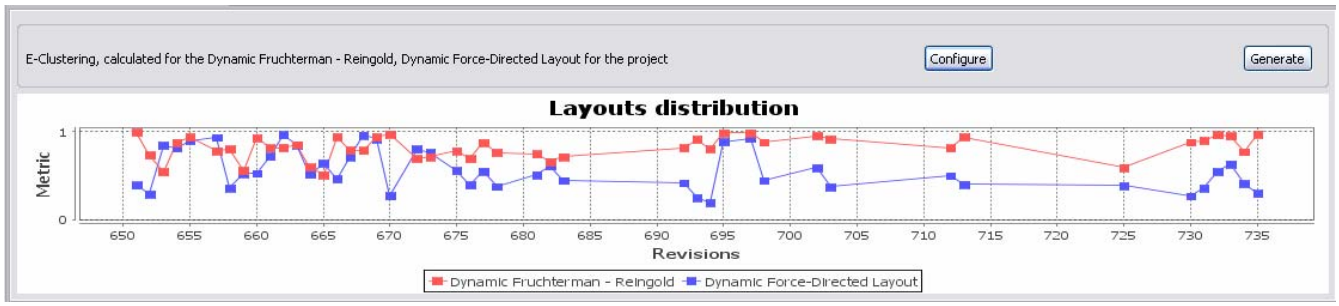

Figure 6. Layouts distribution view

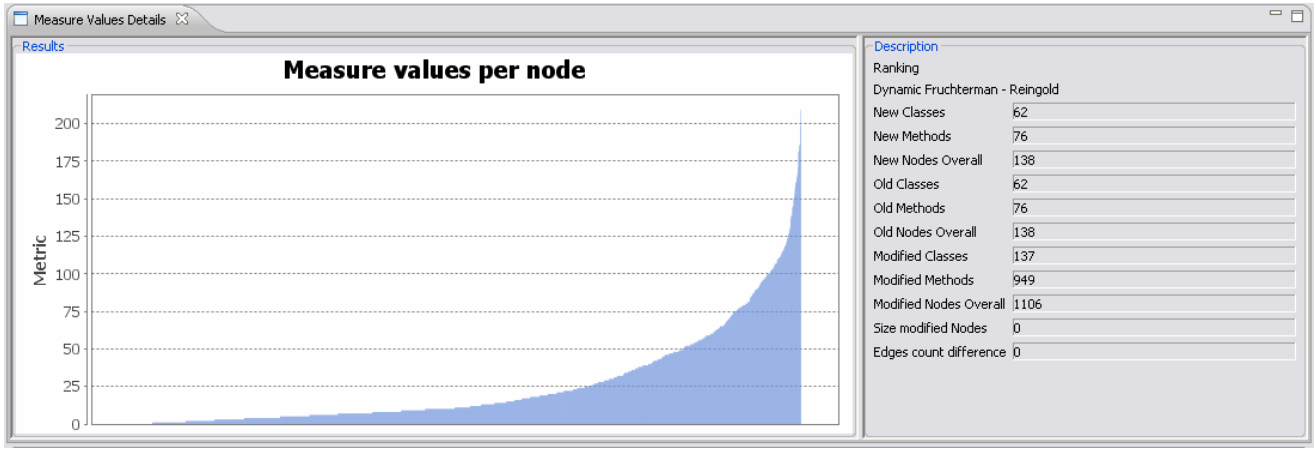

Figure 7. Measure values per node view

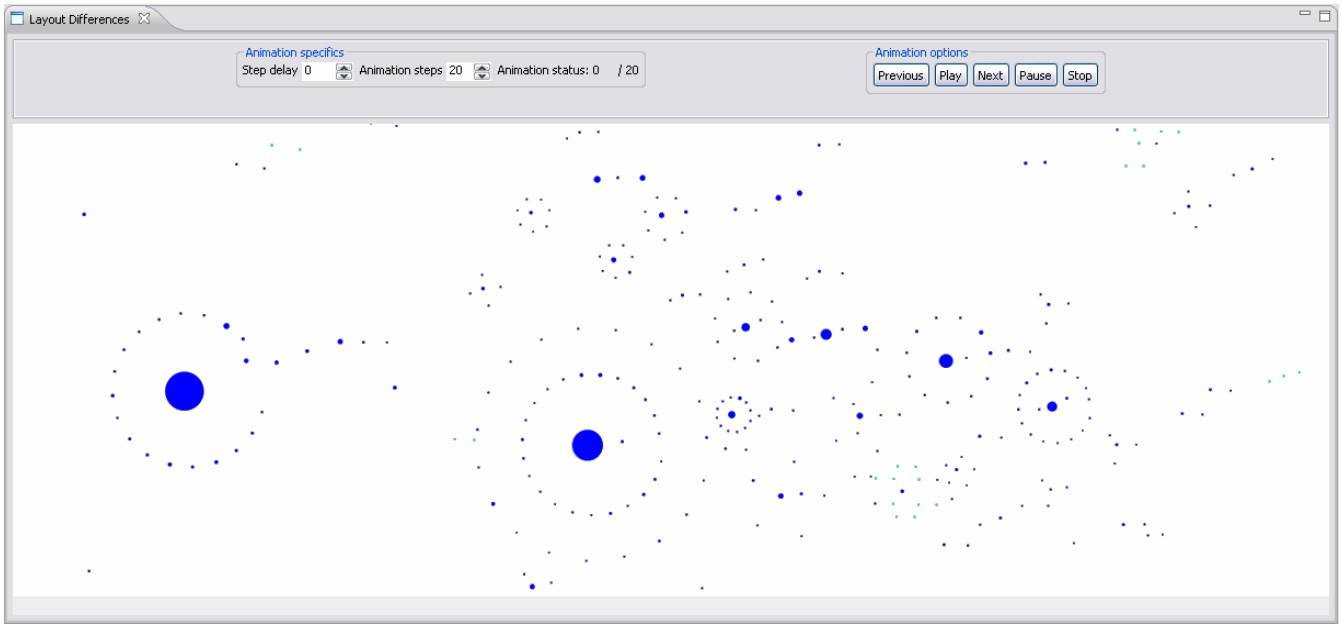

Figure 8. Layout Differences view

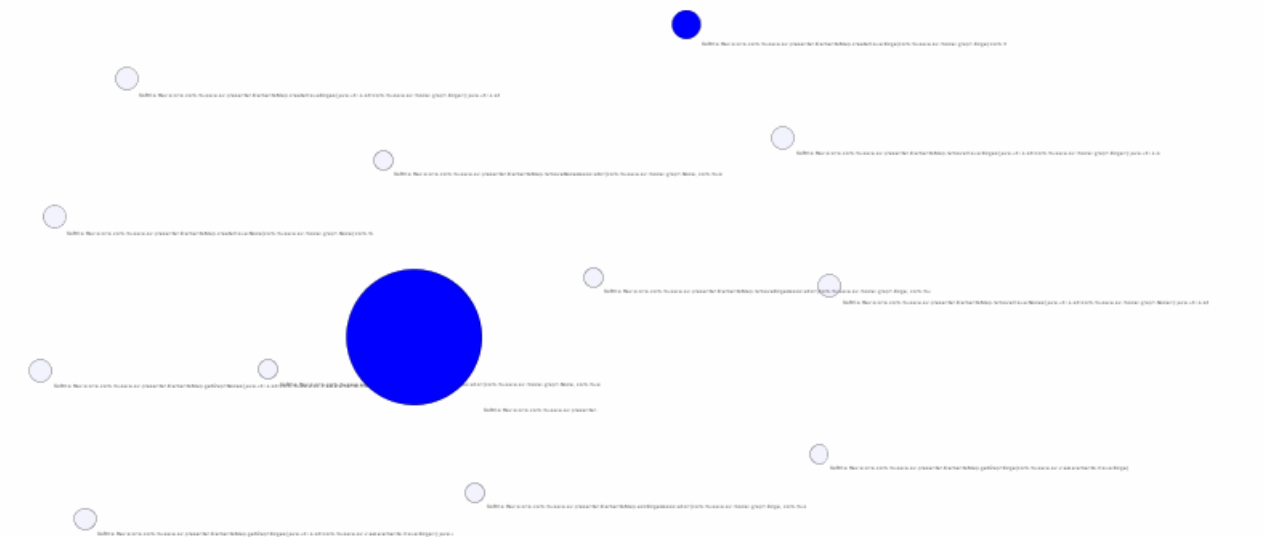

Figure 9. Selection of node in a Layout Differences view

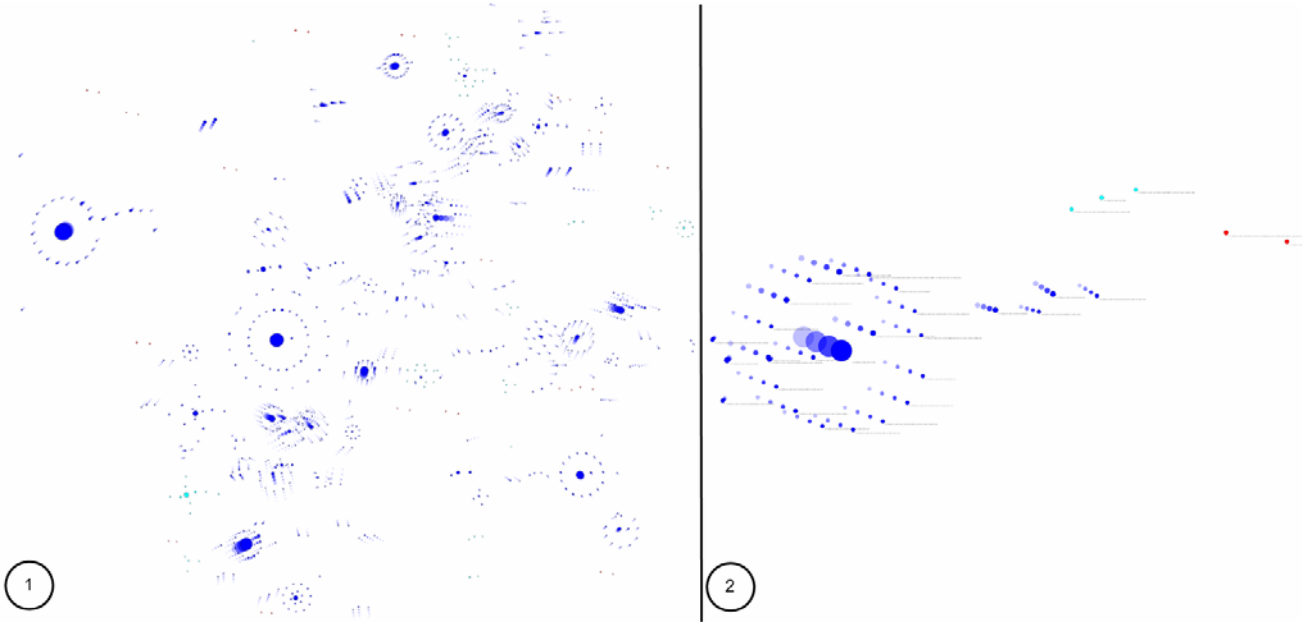

Figure 10. (1) Animation effect on a complete layout; (2) detailed view of the animation effect

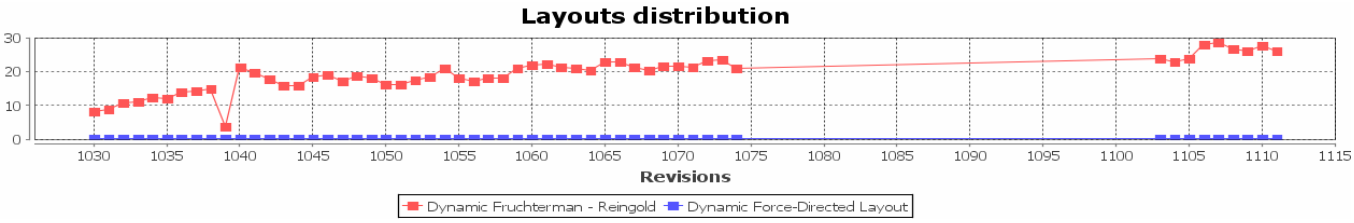

Figure 11. Composite Distance to Neighbor measure values in 54 revisions of JMemorize

- added nodes are red , changed (modified) nodes are blue and the unchanged nodes are shown in cyan color.

4. Analysis of the Graph Drawing Measures

To unify the evaluation analysis of the introduced measures, all the examples in the paper use two dynamic graph drawing algorithms - Dynamic Fruchterman - Reingold [7,6] and Dynamic Force-Directed [9].

To facilitate the analysis explanations the following term is defined:

- *Translation segment* - the segment between the center of the node in the first drawing and its center in the second drawing (*figure 12*, *figure 14*, *figure 16*, *figure 18* , *figure 20*, *figure 22*, *figure 24*).

The source code of open source projects ([4,12,14]) is used to analyze and prove the correctness of the complicated measure definitions and also for observation of the measure values in software evolution graphs. Some of the revisions are not compilable and therefore has no graph representation. The example illustrations display the translation between two drawings of the same graph in a pair of revisions, which are referred as [start_revision_number, end_revision_number]. Such pair of drawings is represented by an end_revision_number in the layout distribution chart.

4.1. Composite Distance to Neighbor Analysis

The measurements of the *Composite Distance to Neighbor* measure are given in *figure 11* for 54 revisions of the open source learning tool jMemorize [4].

In order to prove the correctness of the measure definition, the revisions with the highest difference in the measure values are examined (*figure 11*, revisions [1106, 1107], difference = 28,495). The measure indicates in what degree points positioned near each other in the first drawing remain close to each other in the second drawing. Therefore the most appropriate and comprehensible way to assess the measure definition correctness is to draw all the representation of the translation segments of the graph nodes and analyze their relative positions (*figure 12*, *figure 14*).

Almost all of the translation segments in *figure 12* are collinear and have similar lengths, which indicates the relative node position preservation. The lack of translation segments intersection shows that there are also no additional points in between. Therefore the measure values are valid, according to the measure definition. The *Measure values per node* view for the dynamic force - directed layout (*figure 13*) shows a border case distribution, only several nodes with moderate measure values.

The drawing in *figure 14* is completely different - most of the translation segments intersect and their lengths highly vary. The measure values per node for the dynamic Fruchterman - Reingold layout (*figure 13*) are uniformly distributed with extremely high values (compared with the values of the force - directed layout), which is a precondition for loss of the relative node position and mental map destruction.

The results from the analysis of JMemorize made in the ReViewer show that the dynamic force-directed layout preserves the relative node position better than the dynamic Fruchterman - Reingold algorithm. This makes it appropriate for JMemorize graph visualizations requiring mental map preservation.

4.2. Ranking Analysis

The measurements presented in *figure 15* show the ranking measure values for 45 revisions of the open source Visual Diff and Merge Tool JMeld [12]. To clearly point the drawing criteria differences, which the measure values address, the revision node with the highest alteration (93,867) is taken (revisions [534, 535]). The used approach for proving the measure definition is visualization of the translation segments.

Observing the dynamic Fruchterman - Reingold graph drawing (*figure 16*) transition between 534 and 535 revisions the following conclusions are made - most of the nodes are moved in variety of distances in all possible directions. This fact reflects the number of the nodes, situated right and above - the most significant aesthetic criteria, measured by the Ranking measure.

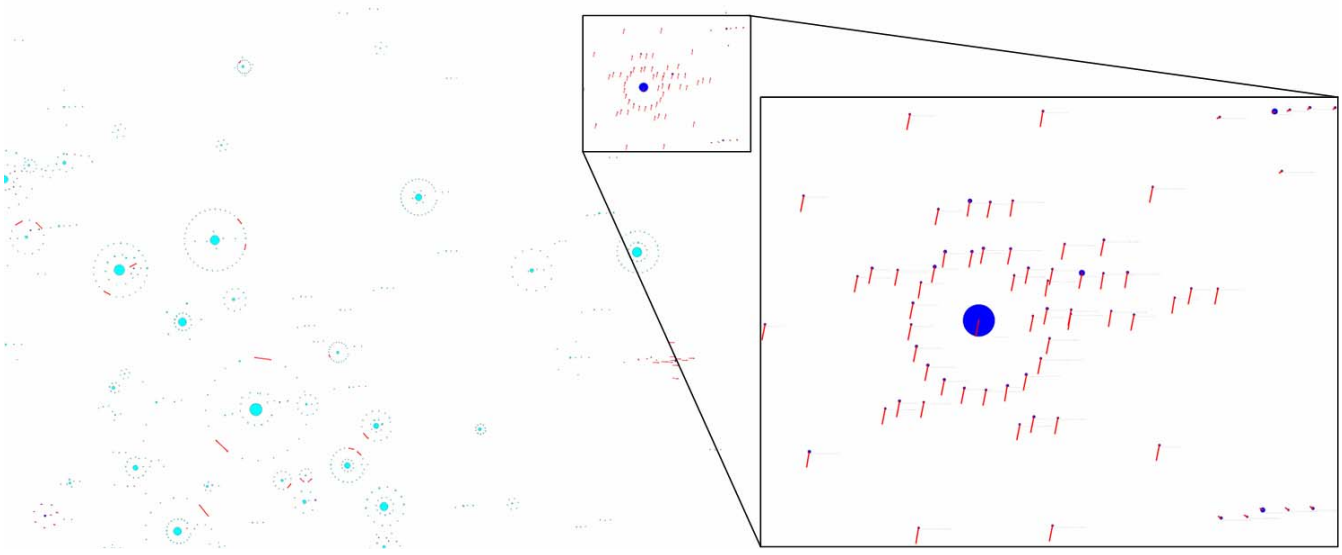

Figure 12. JMemorize, revisions [1106, 1107], Dynamic Force - Directed Layout

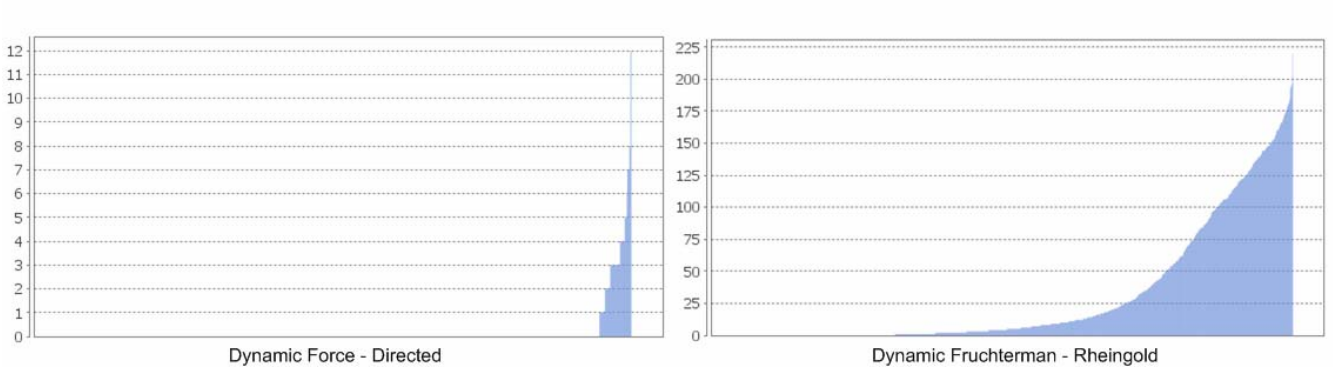

Figure 13. Measure values per node between revision [1106, 1107] revisions of JMemorize

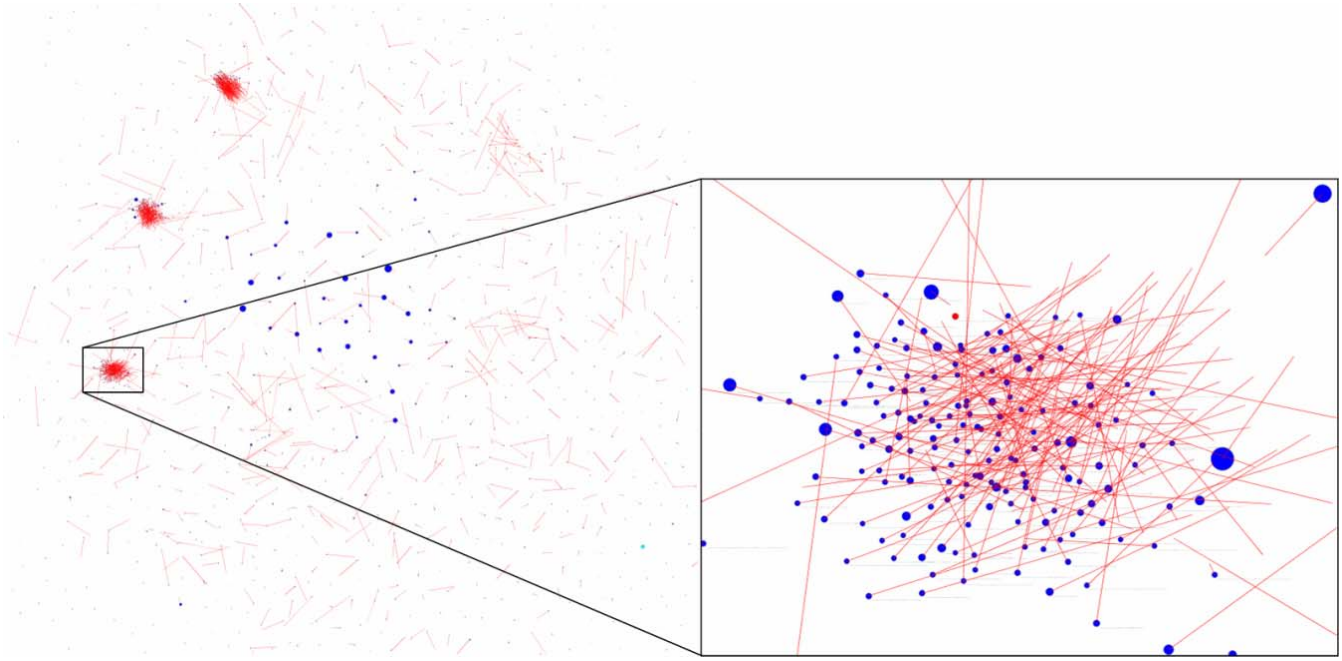

Figure 14. JMemorize, revisions [1106, 1107], Dynamic Fruchterman Reingold Layout

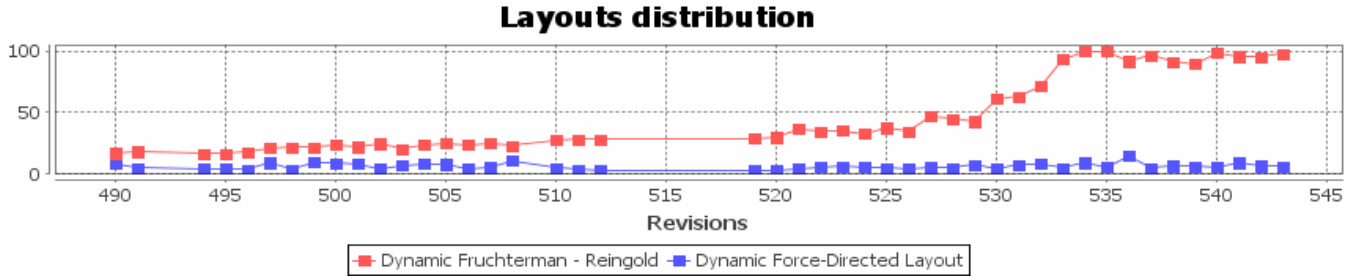

Figure 15. Ranking measure values in 45 revisions of JMeld

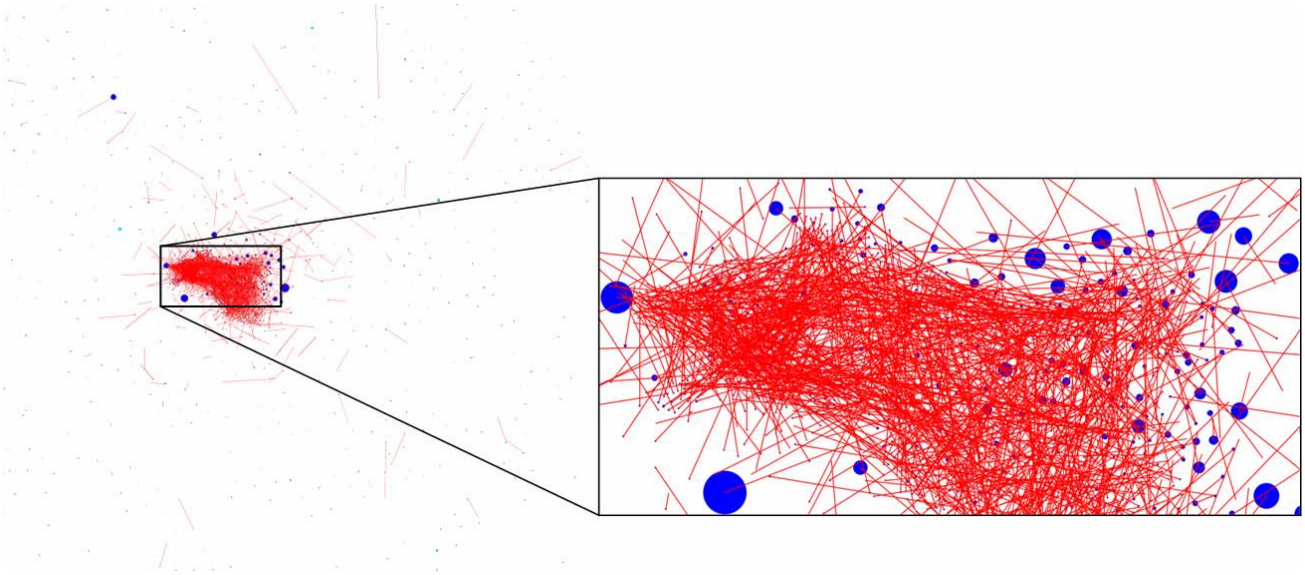

Figure 16. JMeld, revisions [534, 535], Dynamic Fruchterman - Reingold layout

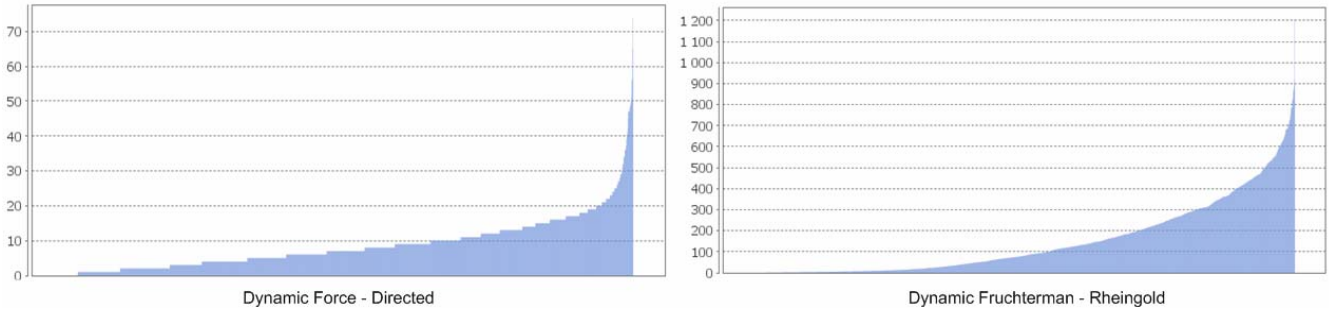

Figure 17. Ranking measure values per node between [534, 535] revisions of JMeld

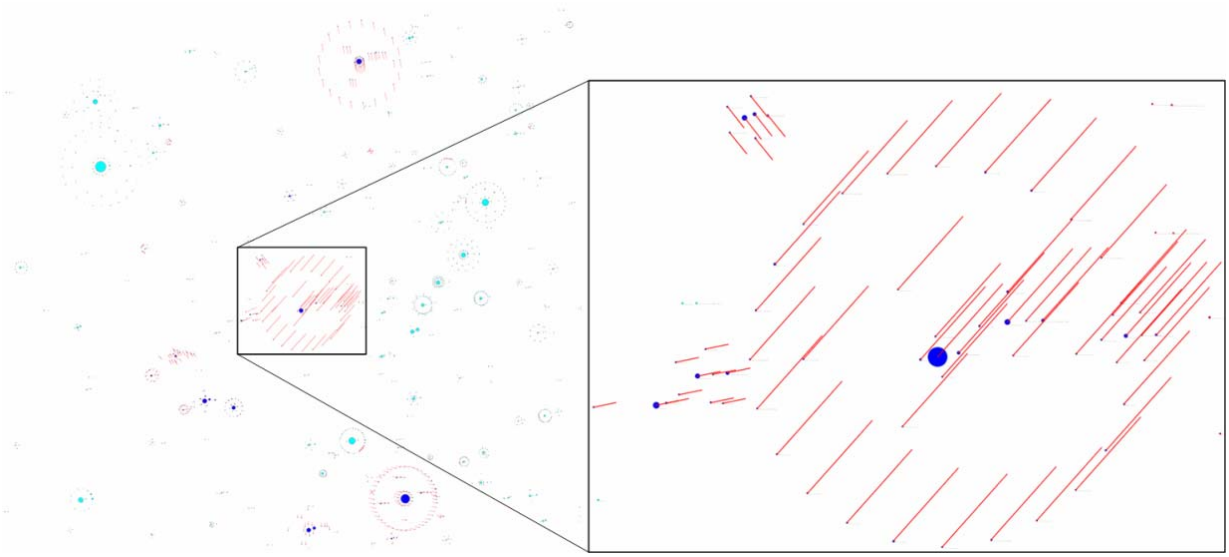

Figure 18. JMeld, revisions [534, 535], Dynamic Force - Directed layout

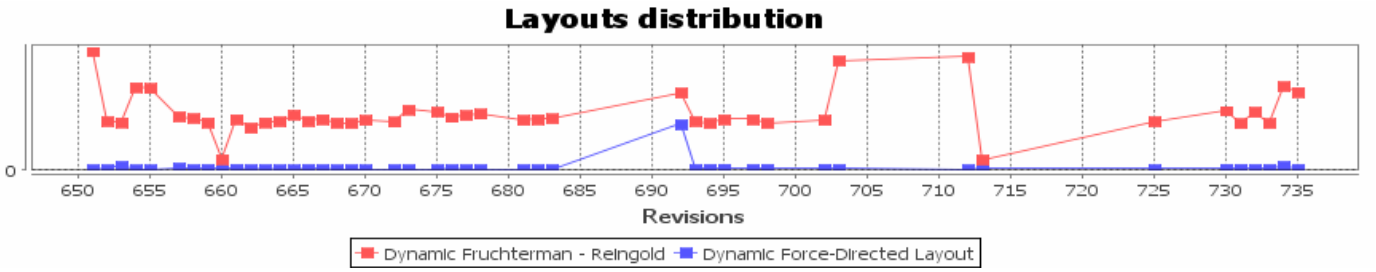

Figure 19. Epsilon clustering measure values in 45 revisions of JavaGit

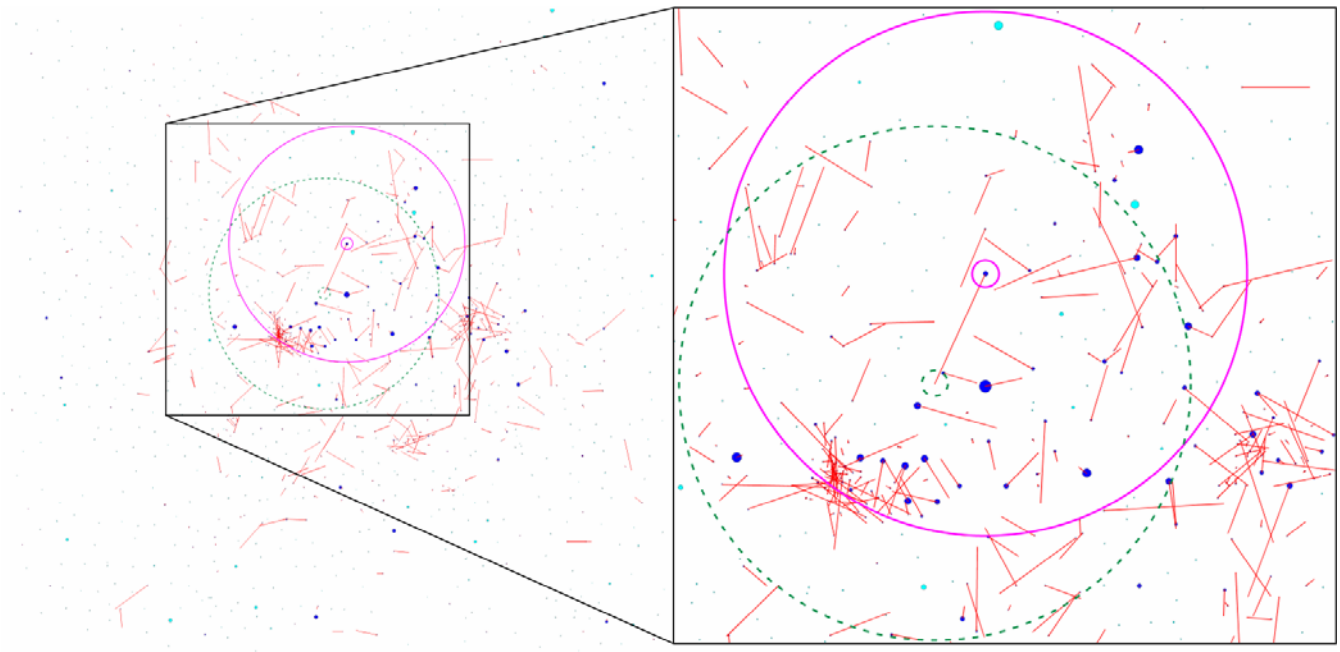

Figure 20. JavaGit, revisions [702, 703], Dynamic Fruchterman - Reingold layout

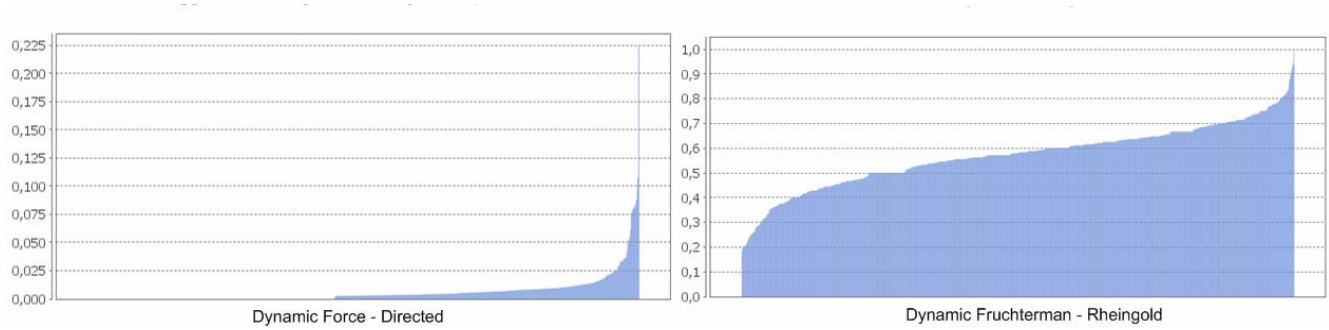

Figure 21. Epsilon clustering measure values per node between [702, 703] revisions of JavaGit

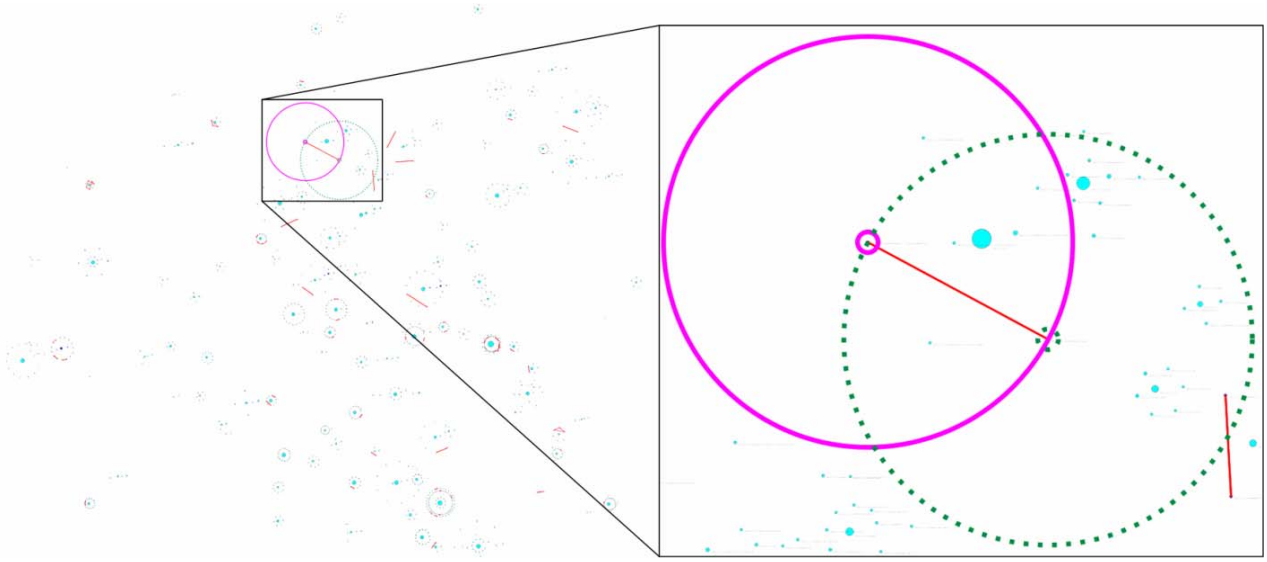

Figure 22. JavaGit, revisions [702, 703], Dynamic Force - Directed layout

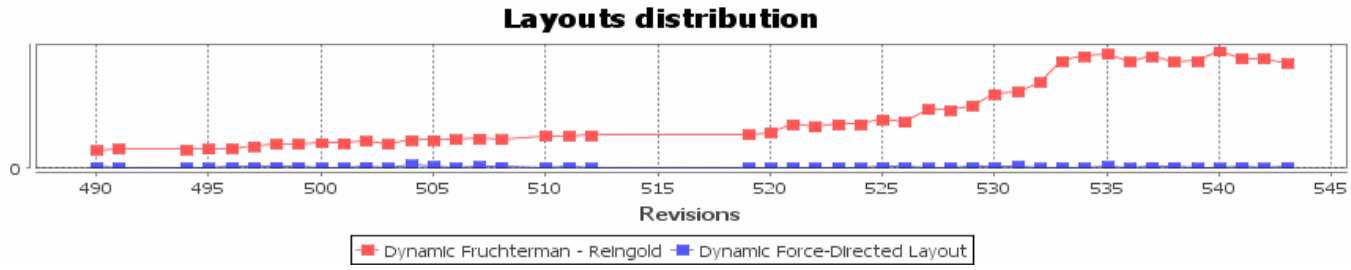

Figure 23. Weighted orthogonal ordering measure values in 45 revisions of JMeld

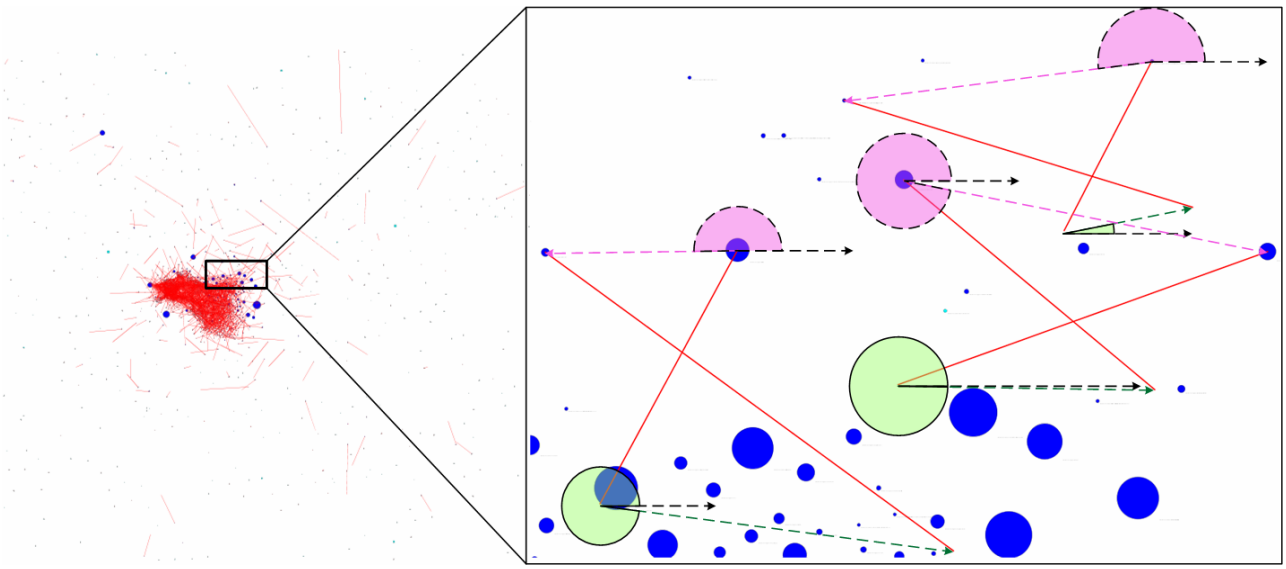

Figure 24. Measurements of orthogonal ordering direction angles, JMeld, revisions [534, 535], Dynamic Fruchterman - Reingold layout

In the dynamic force - directed layout between revisions 534 and 535 (*figure 18*) there are significantly less nodes, translated to large distance. Respectively, they hold their right and above situation towards their neighbors, which is reflected by the values of the measure.

Another aspect for graph layouts quality comparison are the values, represented in the *Measure values per node* view (*figure 17*). The measure values determine several equally distributed intervals, but the general value range is significantly different - up to 70 for the dynamic force - directed layout and up to 1200 for the dynamic Fruchterman - Reingold layout. These values indicate the destruction of the mental map in the drawing, generated by the dynamic Fruchterman - Reingold drawing algorithm.

By examining the measure values in the *Layouts distribution view* (*figure 15*) there is a considerable difference in the values of the ranking measure for the two drawing algorithms. Moreover, the values for the dynamic Fruchterman - Reingold algorithm increase in the process of software evolution, while the dynamic force-directed values vary close to smaller constant value.

Based on the above observations it is clear that the dynamic force-directed graph drawing algorithm is a better choice for the JMeld project if its graph drawing representation requires a mental map preserving layout.

4.3. ε-Clustering Analysis

The ε-clustering measure values for the java access API for Git repositories JavaGit [14] are represented in *figure 19*. In order to facilitate the ε-clusters identification, their borders and centers in the example illustrations are visualized with dashed green circle (representing the clusters for the first graph drawing) and a magenta solid circle (representing the clusters for the second graph drawing). The translation segments represent node movements between the graph drawings.

The revision node for revisions [702, 703] is taken for the demonstration of the ε-clustering definition as a representative of the revision nodes with the highest value alteration (0.543).

The higher epsilon value (0.549) in revision node [702, 703] is due to the clusters disintegration between the drawings, generated with the dynamic Fruchterman - Reingold graph drawing algorithm. The node with the highest measure value (0.946) is easily identified by the *Measure values per node* view and its clusters are drawn in *figure 20*. The illustration shows that more of the most nodes, contained in the green clusters, are not contained in the magenta clusters and also new nodes are added in the magenta cluster. These facts are respectively reflected in the metric measure. By observing the picture (*figure 20*) and the measure values per node (*figure 21*) for the dynamic Fruchterman - Reingold layout the conclusions above can be made for most of the nodes and their clusters.

The dynamic force - directed layout of the graph in revisions [702, 703] (*figure 22*) shows a significantly different situation - only several slightly modified nodes, reflected in the low drawing measure value (0,006). The clusters of the node with the highest measure value (0.224) are drawn showing the mutual position of the node and its neighbors. It can be seen that the number of the points in the clusters is decreased which directly influences the metric measure. As the nodes keep their positions between the revisions the measure values per node for the dynamic force - directed drawing algorithm are lower, compared to the values of the dynamic Fruchterman - Reingold layout (*figure 21*).

As examining the e-clustering measure values in the chart on *figure 19* a quality assessment of the graph drawing algorithms can be produced - the force - directed graph drawing algorithm turns out better layout quality in the context of the relative nodes position preservation when visualizing JavaGit.

4.4. Weighted Orthogonal Ordering Analysis

The values of the orthogonal ordering measure are shown for 45 revisions of JMeld[12] on figure *23*. The examples below use the drawings of the graph between revisions [534, 535], which are also used in the ranking measure analysis.

Since the orthogonal ordering measure calculations are strongly affected by the angles between the different states of the nodes, an additional displayed data visualizes them - the direction angle, representing the angle between the graph nodes in the first graph drawing is given in green sectors; the direction angle, representing the angle between the graph nodes in the second graph drawing - in magenta sector with dashed border (*figure 24*). The angles are drawn in a positive direction, as used in the measure calculation. Black arrows indicate x axis. The translation segments are also drawn to indicate the positions of the nodes in the first and second drawing. Since the weighted orthogonal ordering measure concerns the entire drawing, the measurements per element are not calculated.

The measure value (0.259) in revisions [534, 535] for the dynamic Fruchterman - Reingold layout is one of the highest in the software evolution interval, displayed on *figure 23*. The illustration on *figure 24* shows several direction angles between the nodes positions in the two revisions. The difference in the angles measurements in these consecutive drawings is also displayed by the measure values.

By examining *figure 18*, representing the translation segments for the dynamic force-directed layout it appears that the direction angles remains unchanged since the translation segments are collinear. This fact explains the low value of the measurement between revision 534 and 535 - only 0.004.

According to the measure values on *figure 23* it appears that the dynamic force-directed layout is a better choice for preserving the relative point pairs ordering in the JMeld project software evolution graph representation.

5. Contribution

The quality of data analysis is dependent to a large extent on the appropriate target data set representation as well as the ease and quick orientation within. Since graph drawing is a widely used approach for representing graphs, the quality of the drawing algorithm used is a general problem. By introducing and verifying different methods for layout quality data representation and analysis, the current paper clearly expresses the need of a tool for assessment, observation and analysis of graph drawing sequences. ReViewer is proposed as a possible solution of the defined need. Based on the specific graph drawing appraisal

measures, the ReViewer uses different types of views to represent example data in an intuitive manner from a various synchronized points of view.

6. Conclusion

Visual and numerical approaches for assessment, observation and analysis of graph drawing sequences based on several selected measures are presented in the paper. A software tool ReViewer is developed and used for experimental evaluation of a number of graph layouts by a set of defined measures. The experimental analysis over several revisions of sample software projects shows that the selected measures and the suggested visualization provides a suitable basis for software evolution understanding as well as a stable and mental map preserving drawing of graph sequences.

Future work will include realization of several more layout measures as well as aggregation of measures according to different graph drawing aesthetic criteria. The comparison between arbitrary (not only sequential) graph drawings, representing a concrete source code revisions, should be also provided. Detection of software erosion based on the software evolution system graph also could be done as a part of the graph drawing analysis.

7. Acknowledgment

This work was partially supported by the Bulgarian National Science Research Fund through contract □□□ 02/18 - 2009 „Fast Orientation in Complex Information Systems“.

References

1. Beyer, D., A. Hassan. Evolution Storyboards: Visualization of Software Structure Dynamics. Proc. of the 14th IEEE International Conference on Program Comprehension, ICPC '06, IEEE Computer Society Washington, DC, USA, 2006, 248-251.

2. Bridgeman, S., R. Tamassia. *A User Study in Similarity Measures for Graph Drawing*. Proc. of the 8th International Symposium on Graph Drawing Springer - Verlag London, UK, 2001, 19-30.

3. Bridgeman, S., R. Tamassia. Difference Metrics for Interactive Orthogonal Graph Drawing Algorithms. Proc. of the 6th International Symposium on Graph Drawing Springer - Verlag London, UK, 1998, 57-71.

4. Djemili, R. www.jmemorize.org

5. Erten, C., P. Harding, S. Kobourov, K. Wampler, G. Yee. GraphAEL: Graph Animations with Evolving Layouts. Proc. of the 11th Symposium on Graph Drawing, Perugia, Italy, 2003, 98-110.

6. Frishman, Y., A. Tal. Online Dynamic Graph Drawing. Proc. Eurographics/IEEE VGTC Symp. Visualization (EuroVis '07), 2007, 75-82.

7. Fruchterman, T., E. Reingold. Graph Drawing by Force-Directed Placement. Software Practice and Experience, John Wiley & Sons, Inc. New York, NY, USA, 21, 1991, 1129-1164.

8. Gonzalez, A., R. Theron, A. Telea, F. Garcia. Combined Visualization of Structural and Metric Information for Software Evolution Analysis. Proc. of IWPSE-Evol '09: Joint ACM Int. and Annual ERCIM Workshops on Principles of Software Evolution (IWPSE) and Software Evolution (Evol) Workshops, New York, NY, USA, 2009, 25-30.

9. Iliev, I., H. Haralambiev, M. Lazarova, S. Boychev. Dynamic Force-Directed Graph Layout for Software Visualization. To be published, ICEST 2011.

10. Kaufmann, M., D. Wagner (Eds.). Springer, 2001, Drawing Graphs - Methods and Models - (Lecture Notes in Computer Science; 2025).

11. KDM, http://www.omg.org/spec/KDM/1.0/

12. Kuip, K. http://keeskuip.home.xs4all.nl/jmeld/index.htm

13. Lee, Y., C. Lin, H. Yen. Mental Map Preserving Graph Drawing Using Simulated Annealing. Proc. of the 2006 Asia-Pacific Symposium on Information Visualisation - Volume 60, Australian Computer Society, Inc. Darlinghurst, Australia, Australia, 2006, 179-188.

14. Linder, J., G. Dhindsa, N. Singh, C. Bosley, R. Snyder, M. Artemenko, A. Jadhav. http://javagit.sourceforge.net/

15. O'Madadhain, J., D. Fisher, T. Nelson. Java Universal Network/Graph Framework http://jung.sourceforge.net/

16. Pinzger, M., H. Gall, M. Fischer, M. Lanza. Visualizing Multiple Evolution Metrics. Proc. of the ACM Symposium on Software Visualization (ACM SoftVis'05), New York, NY, USA, 2005, 67-75.

17. R. Gansner, E., J. Mocenigo, S. North. Visualizing Software for Telecommunication Services. Proc. of the 2003 ACM Symposium on Software Visualization, New York, NY, USA, 2003, 151-159.

18. Torchiano, M., F. Ricca, A. De Lucia. Empirical Studies in Software Maintenance and Evolution. IEEE Int. Conf. on Software Maintenance (ICSM'2007), 2007, 491-494.

Manuscript received on 7.06.2011

Dimitar Ivanov *was born in 1987 in Vidin, Bulgaria. He has graduated from Sofia University, Faculty of Mathematic and Informatics, specialty Informatics. He is working as a software engineer and researcher at Musala Soft. His research interests are in the field of software visualization and graph drawing metrics.*

Contacts:
Applied Research and Development Center at Musala Soft
36 Dragan Tsankov blvd
1057 Sofia, Bulgaria
Phone: +359 2 969 58 00
e-mail: dimitar.ivanov@musala.com

Haralambi Haralambiev *was born in 1985 in Burgas, Bulgaria. He has graduated from Sofia University, Faculty of Mathematics and Informatics, specialty Computer Science. He is working at the Applied Research and Development Center at Musala Soft. His research interests are in the field of software visualization, information visualization, software modernization.*

Contacts:
Applied Research and Development Center at Musala Soft
36 Dragan Tsankov blvd
1057 Sofia, Bulgaria
Phone: +359 2 969 58 00
e-mail: haralambi.haralambiev@musala.com

Stanimir Boychev *was born in 1973 in Stara Zagora, Bulgaria. He has graduated from Sofia University, Faculty of Mathematics and Informatics, specialty Mathematics. He is working at Musala Soft Ltd. His research interests are in the field of analysis and transformation of software systems.*

Contacts:
Applied Research and Development Center at Musala Soft
36 Dragan Tsankov blvd
1057 Sofia, Bulgaria
Phone: +359 2 969 58 00
e-mail: stanimir.boychev@musala.com

Assoc. Prof. ***Milena Lazarova****, Ph.D, M.Sc. Eng., has graduated from the Technical University of Sofia, Faculty Computer Systems and Control, specialty Computer Technologies. She is working in the Department „Computer Systems" at the Technical University of Sofia. Her research interests are in the field of computer graphics, image processing, pattern recognition, geographic information systems, parallel information processing, parallel algorithms, parallel programming.*

Contacts:
Systems Department at Technical University of Sofia
8 Kliment Ohridski blvd
1756 Sofia, Bulgaria
e-mail: milaz@tu-sofia.bg