Metaphorical Visualizations of Graph Structures

Ľuboš Ukrop Martin Jakubéci Peter Kapec
Faculty of Informatics and Information Technologies
Slovak University of Technology
Ilkovičova 3, 842 16, Bratislava, Slovakia
{lukrop, matko.jaka}@gmail.com, kapec@fiit.stuba.sk

ABSTRACT
Data visualization of large abstract data sets and complicated relations is a complex research area with different problems and constraints. Often simple shapes and structures are not very eye-pleasing. Visualization metaphors, which create a mapping between a well-known problem domain and a new complicated problem domain, can produce interesting visualizations. In this paper we propose two metaphorical visualizations of graphs and multidimensional data: we propose a metaphor of soap bubble clusters to visualize graphs and a nebulae (sky) metaphor, which uses nebulae and stars to visualize graphs and multidimensional data.

Keywords
soap bubbles, nebulae, visualization metaphor, graph visualization, hypergraph

1. INTRODUCTION
Complex data structures in their raw or pure textual form are unnatural for human perception system and can be very difficult to understand. To enhance comprehensibility, such data are often presented in visual form, represented by abstract geometrical shapes. This idea is further extended by metaphoric visualizations that use the analogy with real world objects. Visualization is often divided into scientific and information visualization, however there is a potential overlap, especially when considering metaphorical visualizations. In this paper we present two metaphorical visualizations of graph structures and multidimensional data.

In Section 2 we outline basic information from the data visualization fields. Section 3 describes proposed technique of graph visualization using soap bubbles metaphor and Section 4 introduces a metaphoric visualization that uses visual syntax of stars and nebulae to create unconventional presentations of graphs and multidimensional data. The proposed visualization metaphors are illustrated by visualizations of real data sets. Section 5 discusses related works and is followed by conclusions.

2. DATA VISUALIZATION
Data is the main object of interest in the visualization. The goal of information visualization is to display abstract entities and relations to provide better insight and easier understanding of information. Important data types that are used in many areas are graphs and multidimensional data. Both require specific approaches for visualization.

Graph visualization
Data, which are decomposable to elements and relations between them, can naturally be represented by graphs. Human understanding of graphs is enhanced by its visualizations. Graph visualization deals with creating, presenting and navigating through graphical representations of graphs. Numerous graph visualization algorithms have been developed (see [Her00] for survey in this area). Most of them are specialized according to the type of input graph – there are algorithms for visualization of trees, oriented graphs, hypergraphs etc. Problems and issues of graph visualization are mostly related to the size of input graph. While trying to visualize large graphs, we have to deal not only with the limits of displaying platform, but also with the limits of human perception system.

First step toward successful visualization is layouting of graph elements. Graph nodes and edges are usually placed in two dimensions, though the usage of 3D or even non-Euclidean geometry is becoming also common. To produce comprehensible output, the layout process has to follow certain aesthetic criteria [Pur00]. Popular methods to solve this task are force directed layout algorithms that employ a physical model built according to the graph structure. Starting from random initial node placement, simulated forces

acting between graph's nodes transforms the layout toward a state with minimal energy. This method was originally proposed by Eades [Ead84], but there are also many other popular variations, e.g. by Fruchterman and Reingold [Fru91]. After the layout is determined, nodes and edges are replaced with their graphical representations usually with the respect to the conventions accepted in this area.

Visualization of multidimensional data

Visualizing multidimensional data (data with more than three dimensions) is complicated, because the human eye is only able to recognize three dimensions. The best examples of multidimensional data are database tables that may contain millions of rows (records) and hundreds of columns (dimensions).

Many different methods to visualize multidimensional data in two or three dimensions have been developed, overview can be found in [Spe99]. Most approaches use some type of mapping attributes to visual and/or structural properties e.g. size, length, color, angle, shape etc. [Sii07]. Very popular are various projection methods that use different kinds of projections from higher dimensions into lower, e.g. Grand tour [Weg92] or SOM [Lat07].

Relatively new approaches use different high-dimensional clustering methods [Ber02]. An interesting approach, related to graphs, is based on clustering using a hypergraph model. During the first step, a weighted hypergraph is constructed to represent the relations among different items, and during the second step, a hypergraph partitioning algorithm is used to find *k* partitions such that the items in each partition are highly related [Han97].

Visualization metaphors

The world of computers is full of associations and descriptions that are based on appearance similarity, which help us to understand the nature of the problems. Metaphor is a tool, which enables us to do this. Metaphors help us to understand one problem area using another problem area. In general we can say that a metaphor is a projection between the source problem domain and the destination problem domain and we want to understand the destination area by comparison to the source area [Ave08].

Visualization metaphors use this metaphorical projection to visualize data. There are different examples of visualization metaphors: a solar system metaphor that uses stars and planets [Gra04], a city metaphor with buildings [Chi05], desktop metaphor [Lar09] or molecule metaphor, which is typical in graph visualization [Ave08]. Others can be also mentioned, e.g. dashboard metaphor, address metaphor etc.

3. SOAP BUBBLES METAPHOR

Soap bubbles are spherical structures made of thin soap fluid film that encloses certain volume of air. Their visual attractiveness and clustering dispositions led us to the idea to utilize them as a visual syntax used for visualization of simple graphs. In this visualization metaphor the graphs are presented as soap bubble clusters with their nodes represented by individual bubbles and edges displayed as cross connections of bubbles that represent the incident nodes. Since the structure of the cluster could be quite dense, connection between not adjacent bubbles will likely occur. Therefore we decided to enhance this visual representation with conventional graphical links used to indicate edges. Textual data attached to graph nodes can be displayed inside the bubbles.

To realize proposed visualization technique three main problems have to be solved: how to layout graph elements, how to represent soap bubble cluster geometry and how to simulate its optical properties to achieve realistic appearance.

Graph layout

To ensure that layout process will produce 3D structures similar to soap bubble clusters, couple of requirements have to be met. For a pair of adjacent nodes (i.e. bubbles), distance of their centers has to be approximately equal to the radius of the larger bubble. On the other hand, a pair of not adjacent bubbles has to be in maximal correlative distance, while preserving connections with adjacent nodes.

For this purpose, we utilized a custom iterative force directed layout algorithm. Each iteration starts with force accumulation. Force acting is defined between each pair of nodes. Let *i* and *j* be two different nodes, *d* distance between their centers and *dir* unit direction vector from *i* to *j*. Then the force acting between them is calculated as follows:

```
force = 0
IF i and j are adjacent THEN
    delta = d – max (i.radius, j.radius)
    force = dir * springStiff * delta
    IF delta > 0.0 THEN
        force = force * springCoef
ELSE:
    delta = d – (i.radius + j.radius)
    IF delta <= repelThreshold THEN
        force = – dir * repelCoef * exp(– delta)
i.force = i.force + force
j.force = j.force – force
```

Simplified interpretation of this is that adjacent nodes are connected by springs with natural length equal to node's larger radius and repulsive forces act between not adjacent nodes. Default values of parameters of the simulated spring (*springStiff*, *springCoef)*, and repulsing forces (*repelThreshold, repelCoef)* were determined experimentally based on observed behavior. With the mass of nodes assigned proportionally to their radius, accumulated forces are

applied[1], resulting in transition to next spatial configuration. To bring in variability of bubble sizes, which is typical for real clusters, radius is defined proportionally to the node degree using the arcus tangent function.

Cluster geometry

For the purpose of layout it was sufficient to characterize each bubble only by its position and radius. However, the rendering process requires more detailed description of the bubble surface. Isolated bubbles tend to have regular spherical shape. With bubbles included inside a cluster, situation is more complicated. Their surface is deformed according to cluster structure. In our solution, the structure of bubble cluster is a direct consequence of graph layout. Since the graph layout may change very frequently, we needed a solution that is able to adapt geometrical representation dynamically in shortest possible time. The approach presented by Sunkel et al. [Sun04] is very suitable for dynamic changes.

Bubbles are initially represented by spherical polygonal meshes. In each rendered frame their shape is modified in two steps:

1. For each bubble, bubbles with which it collides are identified. Based on that, list of intersection planes is built. By intersection plane we mean a plane that separates a part of the bubble surface, which is exceeding into another bubble.
2. Vertex shader in GPU accepts the list of intersection planes that belongs to currently rendered bubble. Before each vertex is transformed, it is tested against each intersection plane. In case it is an exceeding vertex, it is projected onto the intersection plane along the normal. This process is illustrated in Fig.1 - vertex P and all other exceeding vertices are shifted to form the junctions.

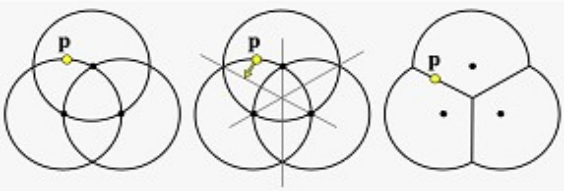

Figure 1. Bubble collisions resolving.

Soap bubble clusters produced with this method will not embrace all the geometric properties of a real cluster. However, such realism was not even our intention.

Visual properties

When the light hits the surface of a soap bubble, several optical effects are observable. Characteristic rainbow-like color toning is caused by interfering light waves. The most distinguished interference occurs between the wave reflected from outer soap film boundary and wave once reflected from inner boundary leaving the film with the same incidence angle. Resulting light intensity I_r is given by following equation:

$$I_r = 4\mathrm{I}_i R(\theta)\sin^2\left(\frac{2\pi}{\lambda} w\eta\cos(\theta_t)\right) \qquad (1)$$

Where I_i is incoming light intensity, θ is incidence angle, θ_t is transmitted angle, $R(\theta)$ is reflectance, λ is wave length, w is film thickness and η stands for soap water index of refraction [Gla00]. Because of Snell's law, the refraction of light also occurs. But since the film thickness is extremely small, it is significant only at the bubble boundaries [Küc02]. Fresnel effect causes, that with declining light incidence angle transparency of the surface raises and reflectivity becomes less obvious.

Photo-realistic rendering could be achieved by precise simulation of all the mentioned optical effects and properties with the use of ray-tracing algorithm. However, since our visualization technique was intended to function on conventional hardware in real-time, we utilized less computationally expensive solutions. Our approach is based mainly on the works of Glassner [Gla00] and Iwasaki et al. [Iwa04]. Using Equation 1 with constant light color (in spectral representation), and similarly to Iwasaki we precompute the interference effect and save it into a 2D texture (see Fig.2a) with vertical coordinate interpreted as the film thickness and horizontal as cosine of incidence angle. Alpha channel contains Fresnel reflectivity used by alpha blending transparency implementation. Film thickness is preserved in a grayscale texture that is mapped directly on the bubble surface (see Fig.2b).

Final soap film color calculation takes place in a fragment shader and involves texels from interference texture (which are computed based on film thickness texture and cosine of incident light angle), texels from environment cube map and diffuse material color. Reflections of dynamic objects (i.e. bubbles) and refraction are omitted.

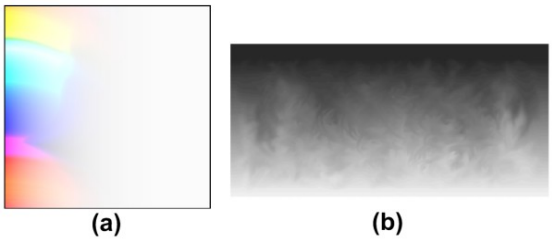

Figure 2. Interference (a) and thickness (b) texture.

Using the proposed visualization technique we were able to interactively visualize smaller graphs (approx. up to 100 nodes). Fig.3 shows visualization of

[1] We utilized Bullet physical engine for this purpose - http://bulletphysics.org.

a graph that exposes relations between scientific disciplines.

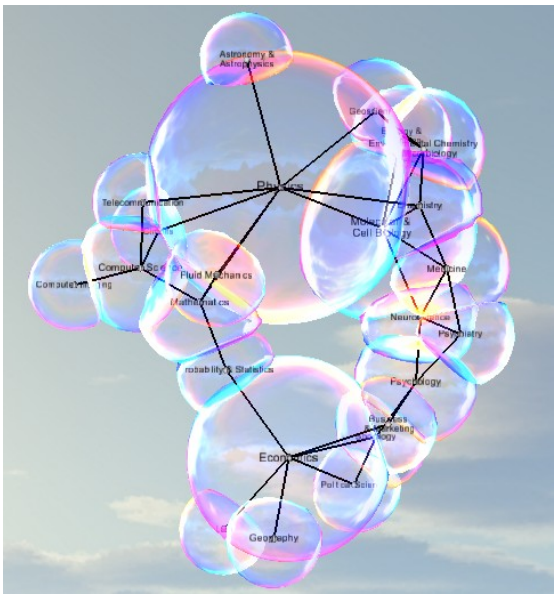

Figure 3. Sample graph visualized by soap bubbles metaphor.

The soap bubbles metaphor increased visual attractiveness of visualization, especially comparing to traditional node-link drawings, though the readability (especially when presented by static image) was reduced. To enable user to directly change the shape of a cluster, mechanism of space constraints was introduced. Space constraints act as enclosed barriers that the bubbles do not manage to cross. Effect of visualization constrained by flatten box is shown in Fig.4.

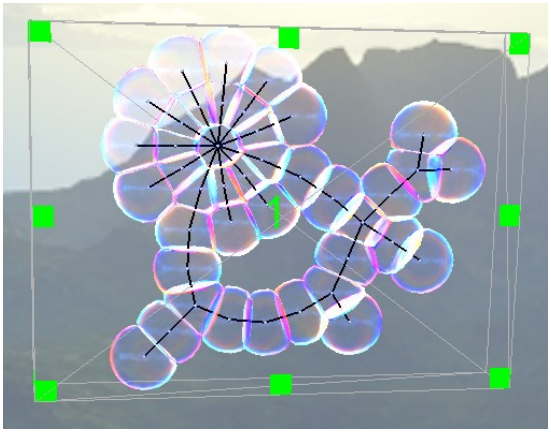

Figure 4. Visualization with space constraint.

Fig. 11 demonstrates visualization of concrete data. It is a part of graph obtained by extraction of software artifacts from source code of real application[2]. It contains all the extracted functions (left part of the cluster) with one of them represented in detail, with its parameters and associated comments (right part of the cluster).

[2] LuaDist - http://www.luadist.org

4. NEBULA METAPHOR

Nebulae (or sky) metaphor is used to visualize data using objects from night sky (from the universe). Typical objects from the universe are stars, nebulae, galaxies etc. The main advantage of this metaphor is that these objects are well known by all people, including small children.

In this approach, stars are used to represent entities (nodes of a hypergraph) and nebulae are used to represent relations (hyperedges of a hypergraph). The stars can be drawn as 3D shapes, for example spheres or textured rectangles, called billboards. Nebulae can be drawn using volumetric rendering or using particle system. Volumetric rendering gives very realistic results [Nad00], but its computational complexity makes it unusable in the case of data visualization. The reason is that real-time drawing and modification of many nebulae is needed. Particle systems are a better solution, but still hard to control in real-time and to draw hundreds of nebulae. A modification of particle systems is used, with pre-rendered cloud particles on a texture, which is mapped on rectangle, so the nebulae are drawn using billboards as well.

The next problem is how to place the billboards in space. A hyperedge visits many nodes and the nebula has to be placed between these nodes. In our approach, a center of the hyperedge is computed and the middle of a billboard is placed on the middle of a line segment, which connects the center of the hyperedge with a node. Billboards are generated for every node a hyperedge visits. Fig. 5 shows a hyperedge with three nodes (circles) and its center (triangle) and how the corresponding three billboards are placed (red, green and pink rectangles).

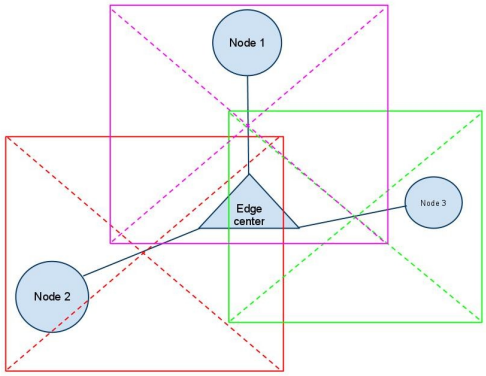

Figure 5. Nebulae billboards placement.

To distinguish between different hyperedges, the color of the nebulae has to be modified. This is achieved by setting the color of the rectangle to a color with alpha channel and then the color of the rectangle is added to the cloud texture using alpha blending. The colors of nebulae seen in space are quite specific, so a set of colors shown in Fig. 6. was chosen. Also other colors-sets can be used, however their selection, probably related to data, can produce

familiar e.g. clouds/smoke-like visualizations or completely unfamiliar visualizations.

Figure 6. Nebulae colors [Fad05].

Now that we prepared a metaphorical drawing method, we have to prepare a hypergraph layout. The standard way to do this is to convert hypergraph into a bipartite graph and then use a graph layout algorithm. In our approach, the hypergraph is layouted and drawn in its pure form. We use a modified Fruchterman-Reingold algorithm [Fru91], where the nodes of a hypergraph are attracted to the center of the hyperedge. After the attractive and repulsive forces are computed and applied, the center of every hyperedge is recomputed as a center of mass, by computing the average position of all node positions, which are connected by the hyperedge.

Fig. 7 shows a hyperedge, which represents calling of a method in software artifacts data. This hyperedge connects eight nodes, which represent different parameters. The hyperedge is visualized using a snebula with eight billboards with cloud textures and the nodes are visualized using billboards with a star texture.

Figure 7. Hyperedge visualized using nebulae metaphor.

Fig. 8 shows a hypergraph with several hyperedges, which were obtained from a larger hypergraph (Fig. 12) by filtering. Hyperedges are visualized using nebulae and stars, but with lines and captions disabled.

To demonstrate the metaphorical visualization a real data set was used. Fig. 12. presents a visualization of software artifacts from a real software system, which was implemented using Lua scripting language. The data includes different types of methods, classes, documentation relations etc. It consists of 1233 nodes and 459 hyperedges.

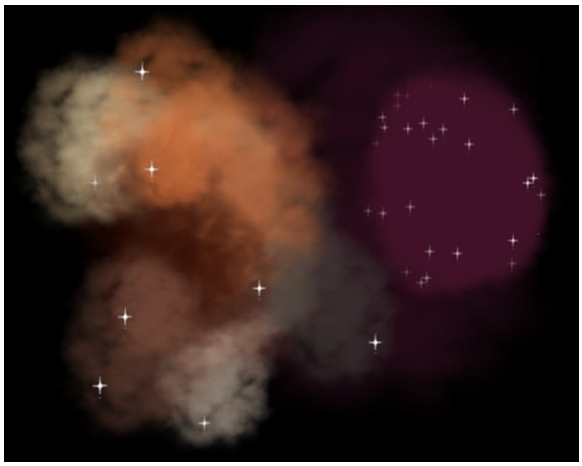

Figure 8. Hypergraph visualized using nebulae metaphor.

Hypergraph based multidimensional clustering

To visualize multidimensional data a simple clustering method inspired by the work [Han97] was implemented. This method utilizes the hypergraph representation and layout. The multidimensional data is transformed into hypergraph structure and then the clusters are automatically created by the layout algorithm. Transforming multidimensional data (rows and columns) can be done in different ways; we used classification of numerical values into ranges or intervals. The clusterisation process is done in these steps:

1. Create an empty hypergraph.
2. Create a node for every row.
3. Create intervals for every column.
4. Create a relation between every value of a row and the corresponding interval.
5. Create a hyperedge for every interval of every column, which is connected by a relation from step 4 to at least two rows. This hyperedge consists of nodes that were created in step 2 and are connected by a relation from step 4.
6. Apply the layout algorithm to the hypergraph.

Fig. 9 shows, how it is done on a sample data set, with six rows and two columns. One interval is created for every column (red interval for first column, green interval for second column); the other values are too different, so no other intervals are needed. Graphical representation shows the generated hypergraph, blue circles are rows, green and red circle are hyperedges, which represent intervals.

Steps 3, 4 and 5 of the process are a little bit complicated. Intervals in step 3 can be created by

different methods, we use a simple approach, where values of the columns are iterated and the minimum and maximum values are found. Then the range between these two values is divided into 10 equal intervals.

Name	C1	C2
R1	24	2
R2	45	3
R3	145	2
R4	2	1
R5	3	1567
R6	5397	538

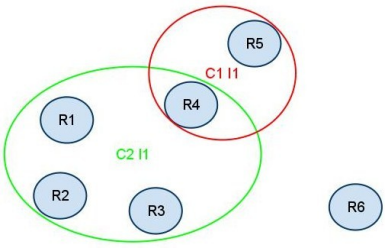

Figure 9. Clustering sample.

After that, we have to find out, to which interval every value of the columns fits. This is done in step 4. Then we just create a model of this situation by representing the relation between a value and its corresponding interval, by creating hyperedges. So rows with similar values of a column (values in the same interval) are connected together by a hyperedge and attracted by applying the layout algorithm. Fig. 10 shows a clustered dataset with different information about proteins. It consists of 1484 records and 8 columns (dimensions) and in the clustering process 1484 nodes and 80 hyperedges (intervals) were created[3].

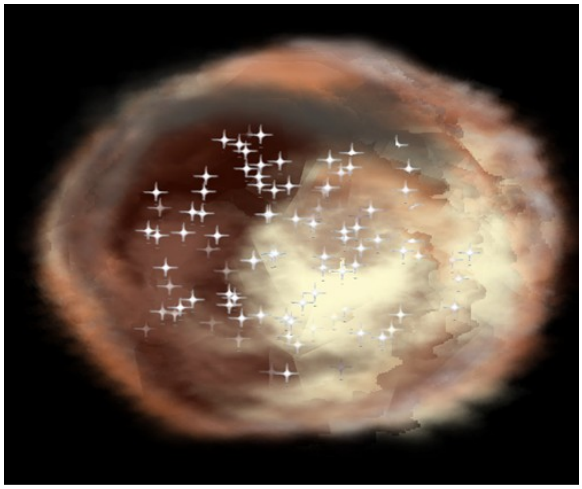

Figure 10. Clustered protein data visualized using nebulae metaphor.

[3] Data-set from http://archive.ics.uci.edu/ml/datasets.html Machine learning repository of University of Carolina

5. RELATED WORK

There were numerous attempts of computer simulation and rendering of soap bubble clusters. Glassner [Gla00] uses a sequence of CSG operations to create a cluster model considering geometric properties of real soap bubble clusters. With precise optical calculations implemented in a fragment shader he was able to achieve high level of realism, but his cluster consists of only three bubbles (analytical solutions exist merely up to this number) and relatively high computational complexity makes his solution inappropriate for real-time rendering. Ďurikovič targeted mainly dynamics of soap bubble clusters [Ďur05]. He represents each bubble as a system of particles connected by springs, taking all significant forces into account, thus simulating bubble creation, coalescence and bubble-plane collisions. Ďurikovič did not describe optical properties. Iwasaki et al. [Iwa04] combined mentioned works to simulate and render a small number of bubbles in real-time on conventional hardware. They reduced computational complexity mainly using a precomputed interference effect. Kück et al. [Küc02] developed rendering and simulation technique for large liquid foam structures. The foam is represented by set of spherical polygonal meshes connected by virtual springs. Junctions between colliding foam bubbles are computed directly inside the shader for ray tracing based renderer. Sunkel et al. [Sun04] came with real-time simplified simulation of large liquid foams: they are creating approximated planar bubble junctions in vertex shader by shifting overlapping vertices to the plane of intersection. Mentioned works were done only with an intention to handle this natural phenomenon by the means of computer science, without presenting concrete practical applications of their results. We applied simulation of soap bubbles in the area of data visualization to create a novel graph visualization technique, while using existing approaches from both fields (mainly [Gla00], [Iw04] and [Sun04]).

Few works use visualization metaphors similar to the nebula metaphor. A sky metaphor was used to visualize self-organizing maps [Lat07]. It displays data records as stars and the clustering process creates star clusters. A visual-analytic tool called IN-SPIRE and its predecessor SPIRE use a galactic metaphor [Won04]. The visualization is using stars and star clusters to help in analyzing of large data. Info-Vis visual explorer is used to interactively explore large collections of documents, which are displayed as stars and collections in the hierarchy are visualized as bounding polygons [Gra07]. All of these works are using just stars to show data records and mostly use star clusters to demonstrate similarity. Our approach is using a similar method when showing clustered multidimensional data, but is also able to show complicated relations displayed as nebulae. It is the only approach, which utilizes 3D visualization as well.

6. CONCLUSIONS

Metaphoric visualizations use analogy with real world objects to enhance user understanding of data and to enrich graphical presentations. In this paper we presented two visualization metaphors applicable on structured data. Soap bubbles metaphor was utilized to create an experimental technique of graph visualization using 3D soap bubble clusters. Based on existing methods and approaches for computer simulation of soap bubbles and custom force directed layout, it enabled us to interactively visualize smaller graphs in real-time. Second presented technique uses nebulae metaphor for visualization of hypergraphs in 3D. With visual syntax of stars and colored nebulae, it is able to visualize also multidimensional data, which can be transformed to hypergraph representations. Both proposed metaphors offer an interesting and unconventional data presentation. Future work will be dedicated to verifying practical usability by the means of user testing, quantitative evaluation and comparison with standard visualization techniques. Both presented metaphoric visualization are suitable for further experiments, e.g. adding smoke into bubbles or applying solar winds to nebulae.

7. ACKNOWLEDGMENTS

This work was supported by the grant KEGA 244-022STU-4/2010: Support for Parallel and Distributed Computing Education.

8. REFERENCES

[Ave08] Averbukh V.L. et al. Searching and Analysis of Interface and Visualization Metaphors. Human-Computer Interaction, New Developments, Vienna, pp. 49-84, 2008.

[Ber02] Berkhin, P. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, 2002.

[Chi05] Chiu, P. et al. MediaMetro: browsing multimedia document collections with a 3D city metaphor. In: Proc. of the 13th ACM international conference on Multimedia, pp. 213-214, 2005.

[Ďur05] Ďurikovič, R. Animation of soap bubble dynamics, cluster formation and collision. Journal of the Applied Mathematics, Statistics and Informatics, vol. 1, no. 2, pp. 33-48, 2005.

[Ead84] Eades, P. A heuristic for graph drawing. Congressus Numerantium, vol. 42, no. 1, pp. 149-160, 1984.

[Fad05] Fadai, K.: Painting a Nebula. Artistic tutorial, 2005.

[Fru91] Fruchterman, T., Reingold, E. Graph drawing by force-directed placement. Software-Practice & Experience, vol. 21, no. 11, pp. 1129-1164, 1991.

[Gla00] Glassner, A. Soap Bubbles: Part2. IEEE Computer Graphics and Applications, vol. 20, no. 6, pp. 99-109, 2000.

[Gra04] Graham, Y.H. et al. A solar system metaphor for 3D visualization of object oriented software metrics, in Proc. of the Australasian Symposium on Information Visualisation, Australian Computer Society, Inc., pp. 53–59, 2004.

[Gra07] Granitzer, Michael et al. InfoSky. InfoVis Wiki.

[Han97] Han, S. et al. Clustering In A High-Dimensional Space Using Hypergraph Models. Technical report, Department of computer science, University of Minnesota, 1997.

[Her00] Herman, I. et al. Graph Visualization and Navigation in Information Visualization: A Survey. IEEE Transactions on Visualization and Computer Graphics, vol. 6, no. 1, pp. 24-43, 2000.

[Iwa04] Iwasaki, K. et al. Real-time rendering of soap bubbles taking into account light interference. In: Proc. of the Computer Graphics International (CGI'04), pp. 344-348, 2004.

[Küc02] Kück, H. et al. Simulation and Rendering of Liquid Foams. In: Proceedings of Graphics Interface, pp. 81-88, 2002.

[Lar09] Lardinois, F.: Bumptop Launches: Make Your Physical Desktop Virtual. Blog, ReadWriteWeb, 2009.

[Lat07] Latif, K. and Mayer, R. Sky-Metaphor Visualisation for Self-Organising Maps. Journal of Universal Computer Science, Proc. of 7th International Conference on Knowledge Management, pp. 400-407, 2007.

[Nad00] Nadeau, D. R. et al. Visualizing Stars and Emission Nebulas, In: Proc. of Eurographics, Eurographics Association, 2000.

[Pur00] Purchase, H. C. A study of graph drawing aesthetics and algorithms. Interacting with Computers, vol. 13, no. 2, pp. 147-162, 2000.

[Sii07] Siirtola, H. Interactive visualization of multidimensional data. PhD dissertation, University of Tampere, 2007.

[Spe99] Spears, W.M.: An Overview of Multidimensional Visualization Techniques, in Evolutionary Computation, Morgan Kaufmann, pp. 104-105, 1999.

[Sun04] Sunkel, M. et al. Rendering and simulation of liquid foams. In: Proc. of the Vision, Modeling and Visualization, pp. 285-294, 2004.

[Weg92] Wegman, E.J. The Grand Tour in k-Dimensions. Computing Science and Statistics, In: Proc. of the 22nd Symposium on the Interface, Springer-Verlag, pp. 127-136, 1992.

[Won04] Wong, P.C. et al. IN-SPIRE InfoVis 2004 Contest Entry. In: Proc. of the IEEE Symposium on Information Visualization, pp. 216–217, 2004.

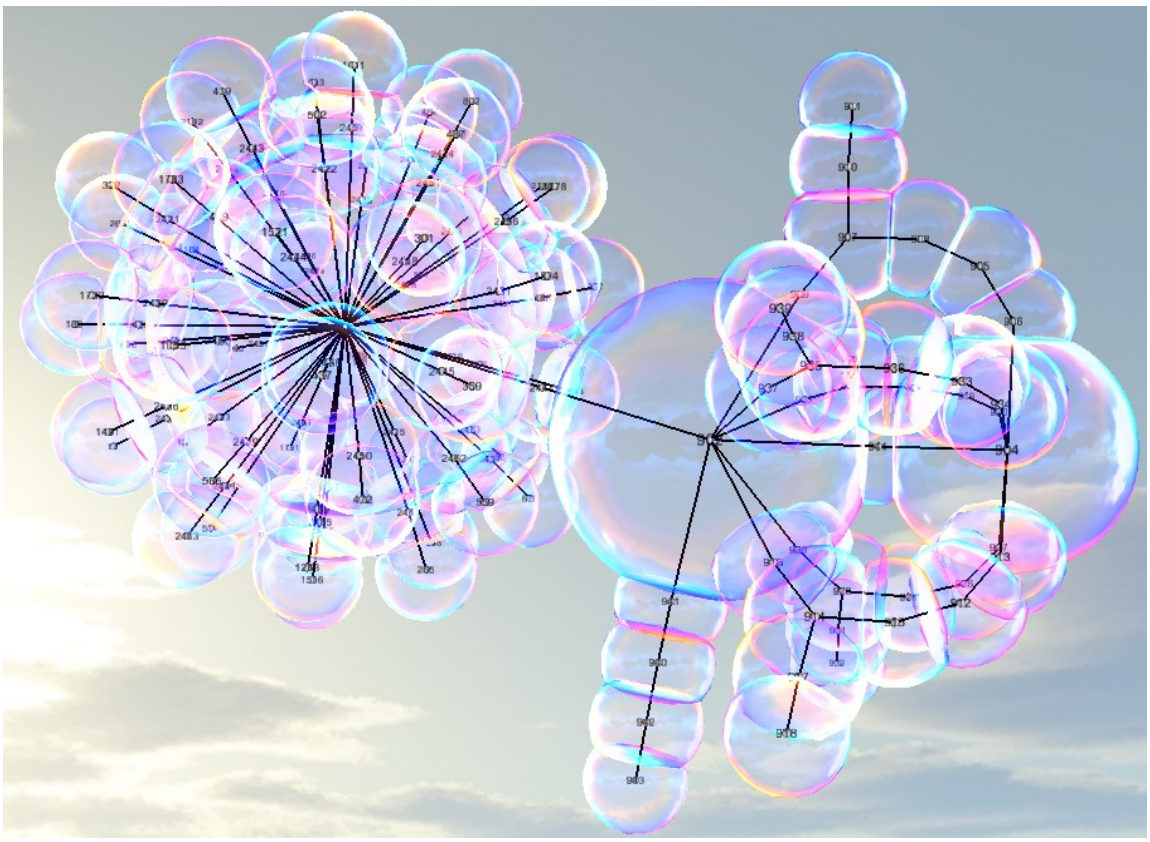

Figure 11. Software artifacts graph visualized by the soap bubbles metaphor.

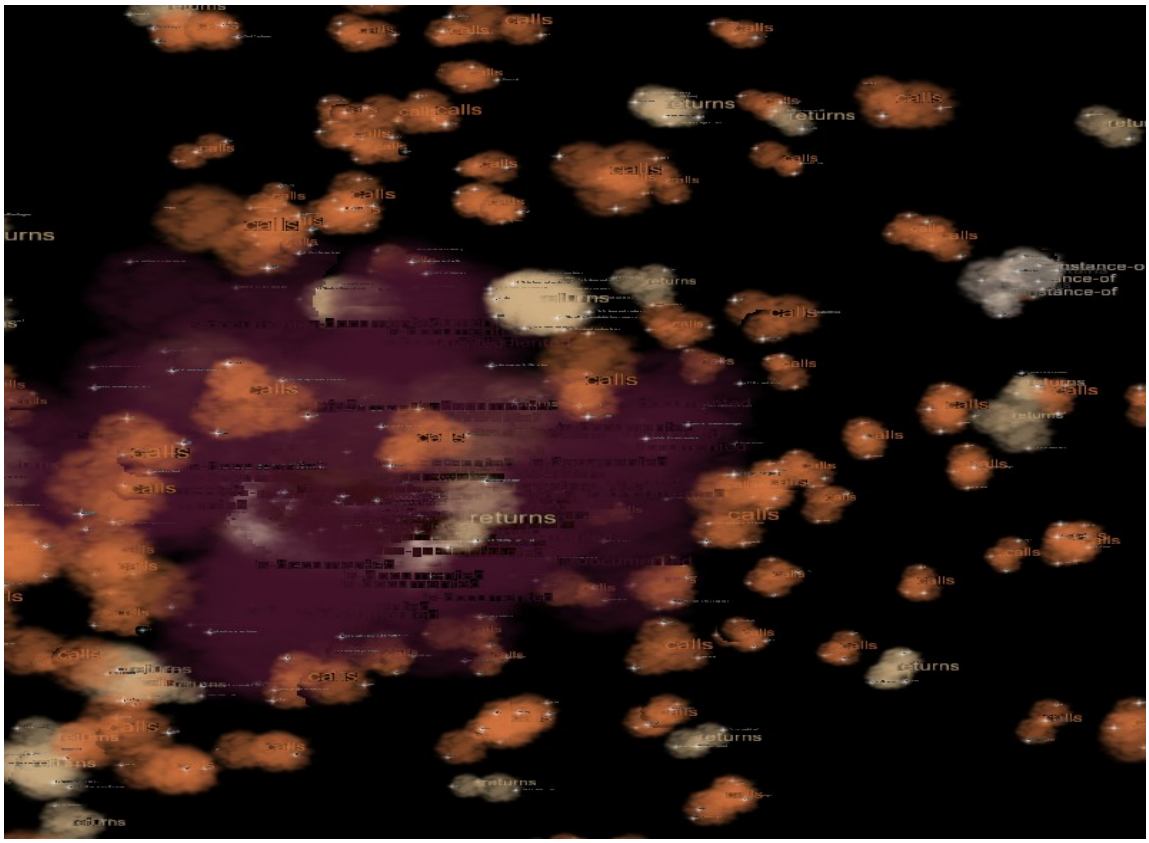

Figure 12. Software artifacts hypergraph visualized by the nebula metaphor.