Alternative Software System Representation in Three Dimensions

DIMITAR IVANOV*, MILENA LAZAROVA**, HARALAMBI HARALAMBIEV*, DELYAN LILOV*

*Applied Research and Development Center, Musala Soft, World Trade Center, 36, Dragan Tsankov blvd., 1303, Sofia
** Systems Department, Technical University of Sofia, 8 Kliment Ohridski blvd, 1756, Sofia

BULGARIA

*dimitar.ivanov@musala.com, haralambi.haralambiev@musala.com, delyan.lilov@musala.com, http://www.musala.com
** milaz@tu-sofia.bg, http://www.tu-sofia.bg

Abstract: All the operations, concerning the support of a software system require a good understanding of it. In order to provide enough information for fast orientation in given software system, the present paper proposes an alternative three dimensional visualization, which uses several publicly familiar metaphors, inspired from the space and the geography. Since the visualization is based on entity-relationship model of the system code, the representation uses several methods for better understanding of the graph drawing – layout algorithms, semantic coloring, edge bundling as well as an innovative approach for visual clustering – the bloom effect.

Key-Words: information visualization, software visualization, 3D, visual metaphor, software comprehension

1 Introduction

One of the most significant problems, concerning the dynamic work of software developers is the need of short-term rapid comprehension of unfamiliar software system. Panas et al. discuss the process of support, extension and reuse of industrial size software systems and present external statistics, claiming that 50% to 75% of the time and resources are invested in software comprehension as well as 47% to 62% of the time for actual enhancement and correction tasks is spent on comprehension activities. Another relative problem in the context of time-consuming software comprehension is the lack of adequate, well-structured documentation [10].

According to Rilling et al. there is close relationship between the process of reverse engineering and the software comprehension [13], which is a good premise for mutual usage of shared approaches in order to decrease the comprehension time interval. Different methods can be used to facilitate the understanding of given software system as much as possible. We believe that the most suitable method for orientation in considerable amount of information is to visualize it adequately. Moreover, such information representation will assist the communication between the system developers [17], which is also a great impediment for the working groups.

Every user of the tools for software visualization and comprehension takes a role, corresponding to the responsibilities taken in the software project. In order to modify and improve the system, the developers are interested in its functional and nonfunctional issues. On the other hand, the managers aim to identify the system components with global impact, which affect the maintenance, reliability or performance of the project [10]. Software architects strive for identifying the role of the system’s components, their mutual relations and the measurements of given software metrics, related with them [3]. The visualization, described in the present paper, aims to analyze such requirements and to develop the most commonly needed in order to assist all the users of a given software system.

One of the popular works, concerning the information visualization defines a simple rule - “overview first, zoom and filter, then details on demand” [14]. The proposed visualization adheres to these concepts and defines its own abstraction and metaphor. The abstraction is needed to overcome the details on displaying the source code and to improve the understanding of the software system [1]. The chosen metaphor is natural, publicly
familiar and close to the known universe – the elements are located in unlimited space and semantically clustered by using different approaches like semantic coloring, semantic positioning and semantic clustering. Every element of the software system has corresponding representation in the metaphor. The visualization uses an innovative approach for visual clustering, achieved by using natural way of grouping elements into visual clusters – the bloom effect [7].

There are vast amount of discussed examples, which illustrate the advantages and the disadvantages of representations in two or three dimensions ([9], [13], [17]). The work presented in this paper uses 3D representation as an environment for demonstrating different approaches for visualization of given software system. We believe that the balance between the visually attractive and information richer visualization can be achieved only in three dimensions.

To complete the task for visualizing the selected abstraction, metaphor and the bloom effect we need development tool with more complete application programming interface (API), which provides easy integration with different types of 3D widgets [11]. That is why, the current work develops software visualization system on OpenGL [15] as a multiplatform rendering framework for visualizations in 2D and 3D.

2 Related work
There are various approaches for software representation and visualization – dynamic, static, combined visualizations, etc. The following paragraphs summarize the related work on the software visualization approaches, representing static details of the system.

Balzer et al. present the landscape metaphor for visualization of the static software structure (Fig.1) by combining three-dimensional landscape elements, customized layouts and hemispheres, used to group semantically near elements [1].

Ploix describes metaphor, using solar system model (Fig.2) – solar systems represent functions, surrounded by its planets – the expressions of the program and their moons – the data, used by the expressions, where the lines shows the calling links [12].

2.1 Representation
The representation is considered as one of the most important aspects of the visualization and concerns the graphical representation of the software system

3 Visualizing software systems
The proposed software visualization adheres to six key areas of interest proposed by Young et al. [19], organized in the subsections below.

3.1 Representation
The representation is considered as one of the most important aspects of the visualization and concerns the graphical representation of the software system
components, including the information volume, encoded in every atomic visual element.

In order to provide alternative and understandable software system visualization, we experiment in constructing drawings of the system graph, using combination of metaphors borrowed from the nature and the surrounding world.

3.1.1 Space metaphors
The space metaphors, used in the proposed visualization are inspired by the widespread popular science materials, describing the known space universe. We made the parallel between the software systems and the universe - as the planets and their satellites form systems, the systems form galaxies, situated in the unlimited universe, so the statements form methods, the methods forms classes, which are part of components, combined in a software system.

One basic element of the visualization, similar to the space, is the concept of gravity – related elements are situated near to each other, following own orbits, according to their semantics (e.g. the private methods are nearer to the class representation, while the public methods are in the outer orbits - Fig.4). The positioning of the graph elements is very popular and complicated problem in the graph drawing area. The proposed visualization uses dynamic force-directed layout algorithm [8], developed to represent software system, following the concept of gravity.

3.1.2 Geographic metaphors
The comprehension of a software system is frequently related to discovering simple relationships between the components, e.g. finding the way between certain graph vertices. We found it similar to the navigation tasks between geographic objects and decided to express the software system in such manner (Fig.5). The whole graph, representing the software system, is laid out on the surface of adequately sized sphere. The elements of the system are situated following the same methods – related elements positions are near to each other. The result visualization resembles to geographic map.

3.1.3 Combined metaphors
The space and geographic metaphors achieve aesthetically appealing visualizations of software system, but the orientation in such three-dimensional structures appears to be a problem without coordinate system or another starting point. However, the use of classical xyz or spherical coordinate systems does not improve the orientation as well. In order to address such issue we experiment in two directions, by combining the approaches, described above.

3.1.3.1 System plane
As a slight modification to the geographic metaphor, we experimented placing the whole system graph in a single plane in the space (Fig.6). By using this approach we imitate geographic map with a 3D view.
Based on this fact, we propose software visualization, abstracting the information as a directed graph. The nodes of the graph represent the code elements of the software system – classes, methods, interfaces, etc. The edges of the graph represent the relations between the code elements – extends, implements, calls, etc. The nodes and the edges are related to additional information about their corresponding code elements, which could be accessed on demand.

### 3.3 Navigation

The volume of the represented information presumes corresponding size of the visualization. The navigation is a set of approaches for guiding the users through the visualization without getting them lost or disorientated.

#### 3.3.1 Semantic coloring

The color and the shape of a visual element are its most noticeable properties. In order to achieve the best semantic separation, every element color is dynamically calculated with an algorithm for generating distinct colors in HSV and converting them in RGB [16]. The principle of this automated generation is simple – every component defines its color interval, which is also split by the number of the owned elements and used to generate their color (Fig.8). Respectively, the visual distinction of the colors depends on the number of the components of the system.

![Fig.8 Distinct color generation](image)

This approach creates visual mapping between the responsibility of the element in the system and the color of its representation in the visualization. We believe that by using this approach, we significantly accelerate the process of full visual identification of an element by its representation.

#### 3.3.2 Bloom effect

As we discussed above, the semantic clustering of the elements in the proposed visualization is achieved by using the bloom effect [7]. Such type of approach for visual clustering fits perfectly in the described metaphors allowing unlimited level of drill-downs. In the current example, the distance...
from the camera (the viewer) to the objects explored determines the size of the bloom and naturally forms a kind of super-nodes, representing whole system components.

3.3.3 Edge bundling
The number of the relations between the software system elements turns the graph drawing in an incomprehensible tangle. The proposed visualization uses force-directed edge bundling algorithm [6] to reduce the overload and to provide clearer system visualization. After applying the bundling on the edges, the visualization looks better and is particularly useful in the general overview of the system when using the different metaphors (Fig.5, Fig.6, Fig.7). Moreover, when the edges of the graph are bundled in a plain, the representation looks very similar to geographic map (Fig.6).

3.4 Correlation
The correlation is the linkage between the visualization and the represented information store. Unfortunately, the used approach for building the entity-relationship model of the source code (see 3.2) is limited to building a single state of the code and therefore visualizing only a single state of the system. On the other hand, there is a linkage between the visualization elements and their corresponding parts of the system, including their code. Frequently, the source code of the system element is enough to identify where it is located and what is its purpose.

3.5 Automation
The automation defines how much of the visualization is generated automatically without need of user interaction in the entire process. The proposed visualization has fully automated process of construction and the user input is limited to two basic steps – selection of the source code directory, where the system’s source code is located and selection of the metaphor to be used – space, geographic, system plane or component plane. Actually, the user interaction in the process of visualization construction is decreased to minimum.

3.6 Interaction
Interaction concerns the quality and the quantity of the needed user interaction, in order to use the visualization sufficiently. Since the proposed visualization is targeted to software developers, architects and manager, its interaction relies on the basic interaction tools – the keyboard and the mouse. All the camera navigations such as panning, zooming and rotation are implemented to be fully controlled by the mouse. The camera navigation is also accessible through the keyboard, but more limited – for going forwards, backwards, left and right. We also experimented in using different interaction techniques (UniCam [20]) but it appeared to be not enough usable for the users.

4 Conclusions
Despite the presence of numerous tools for software comprehension and visualization, the developers, software architects and managers are still having problems in the analysis and comprehension of unfamiliar software system. The same problems can arise in the processes of bug fixing, refactoring, optimization, etc.

By focusing the user attention on different aspects of the data, according to the viewing distance, the proposed visualization provides a full control over the represented information. Some of the used approaches provide general overview of the system as well as more detailed view of its components and their structure. On the other hand, the rest of the approaches provide more intuitive methods of representing the information, by using widespread and popular metaphors.

The visualization also adds more advanced features such as bloom effect, edge bundling and custom graph layouts. These supplements provide better structuring of the represented information as well as more natural view of the used abstractions and metaphors. The semantic coloring of the elements also helps the orientation in the system representation.

The visualization, proposed in the current paper presents several alternative and different approaches for representation of software systems. The general description of these methods provides a mechanism for visualizing any data, described using entity-relationship model. Moreover, the details, presented in each level of the visualization can be used for identification of structural and behavioral problems in the visualized systems.

5 Future work
The characteristics of given software system depends on its purpose and basic functions. In order to visualize it adequately we plan a mechanism for
integration of different business intelligence rules. Such mechanism can extend the visualization qualities and facilitate the dynamic identification of different subsystems, according to their purpose.

Another possible challenge for the present visualization is the representation of several related systems in a single visualization.

6 Acknowledgement

This work was partially supported by the Bulgarian Science Fund through contract ДМУ 02/18 – 2009 “Fast Orientation in Complex Information Systems”.

References:


