# Lowering Visual Clutter of Clusters in Component Diagrams

Lukas Holy, Jaroslav Snajberk, and Premek Brada
Department of Computer Science and Engineering, Faculty of Applied Sciences,
University of West Bohemia, Pilsen, Czech Republic
{lholy, snajberk, brada}@kiv.zcu.cz

***Abstract*—Nowadays, component applications can easily consist of hundreds or thousands of components and it is thus difficult to understand their structure. Diagram visualization does not help much because of visual clutter caused by big amount of elements and connections. This paper describes an approach of removing a large part of connections from the diagram while preserving the information about component interconnections. It also describes a viewport technique for showing all information about interfaces for selected group of components right in the diagram area. After that it presents novel integration of above mentioned techniques which maps a group of components to the content of a viewport. These techniques are among other benefits useful in the reverse engineering process. The main idea of this technique can be used in a similar way to reduce the clutter in the node-link graphs. To show the effect of this technique, example reduction of lines is discussed. So the better understanding of a diagram is also shown on preliminary results.**

***Keywords-software visualization; component; visual clutter.***

## I. Introduction

Software applications become more and more complex and although there are lots of tools, which help the development process, they are still limited in helping human understanding of the application structure. Software components are one of the ways to handle this complexity as they encapsulate parts of functionality to unified components. Even with the usage of components, applications can easily consist of hundreds or thousands of them. It is therefore difficult to explore the structure of the application and create a mental model of the whole system.

One of the ways how to get an insight into a component application structure can be UML (Unified Modeling Language) component diagram. When the diagram becomes large there are many problems with exploring it. One is the contradictory need of providing enough details and showing the complete diagram (application structure) at the same time. Another question is how to reduce visual clutter caused by the large number of elements and connections between them. This visual clutter makes tracing of dependencies difficult and hinders orientation in the diagram. Current tools do not offer features designed for work with such large diagrams, as we have shown in our previous paper [1]. In this work, we present several techniques to reduce visual clutter in UML component diagrams and help user to form clusters of components.

### *A. Structure of the Paper*

In the following section, a problem of visual clutter is defined first. After that, a related work is described in Section III. Then, in Sections IV and V a SeCo (Separated Components) technique and its implementation is presented. This technique helps to reduce the visual clutter in large graphs. Also, another technique called viewport is shown in Section VI ,which helps to form component clusters. After that, the novel integration of SeCo and viewport techniques is proposed in Section VII. Finally, our contribution is discussed in Section VIII and summarized in Section IX.

## II. Problem Definition

Developers face multiple challenges in large diagrams visualization such as difficult orientation, limited amount of visible elements on the screen while showing its details, insufficient details when showing overview or the visual clutter [2].

This paper focuses on the problem with highly connected components and the clutter caused by their connection visualization as well as the component clusters visualization.

Very often, only a small amount of components is connected to a large number of other components. Such components are often, among developers, informally called "God Objects". It is difficult to trace the connections in the surrounding area of these objects. Another problem in visualization is forming and working with clusters of components, which usually represent one feature or logical unit of the system. These problems cause exhausting space, which is one of the essential resources in the visualization and can be used for easing the work with large component diagrams.

## III. Related Work

Visual clutter can be reduced by many techniques. The whole taxonomy of these techniques has been described by Ellis and Dix in [3]. A short description of those techniques related to our work is provided.

The clutter caused by the lines is often reduced by edge bundling [4]. Although this approach reduces the clutter, it can be difficult to trace the dependencies between connected nodes leading through the edge bundles.

The visual clutter can be also lowered by using node clustering, where one cluster usually represents multiple nodes. The overview of clustering algorithms can be found in [5].

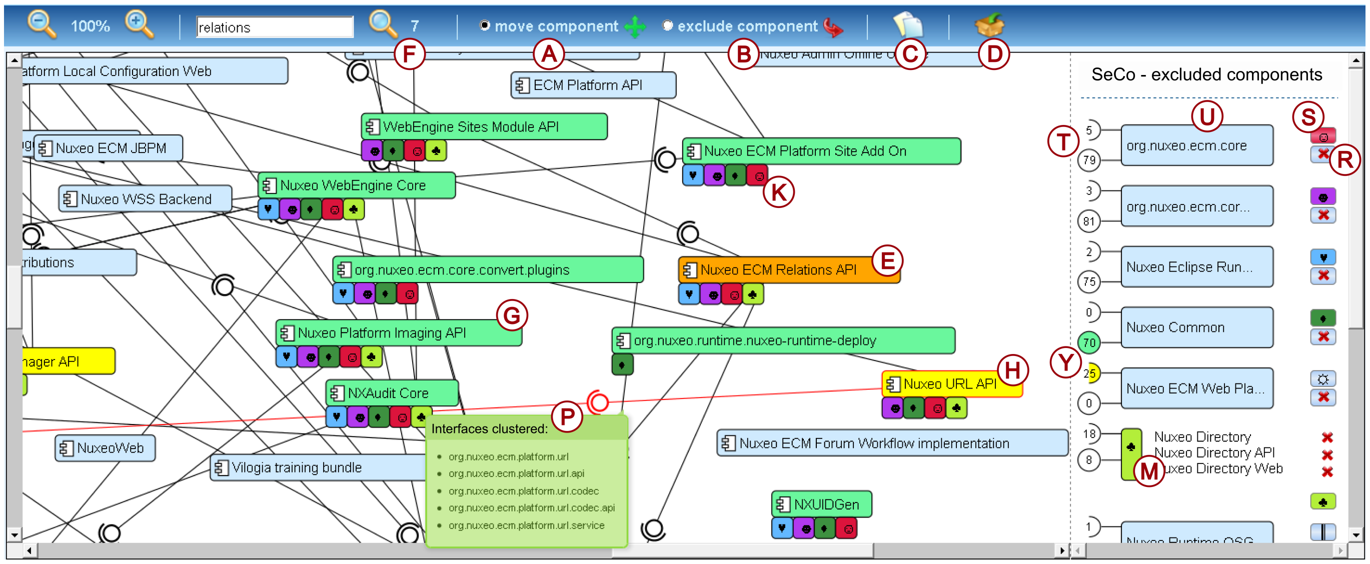

Fig. 1. Complex Component Application Explorer tool demonstration

Another influencing factor is the chosen layout algorithm, which can ease orientation in both clustered graphs [6] and a non-clustered ones [7]. In the following section, our visual clutter reduction approach is described.

## IV. The Separated Components Area Technique

The technique proposed in [8] reduces the visual clutter by removing the components with a large number of connections from the main diagram into a so called *separated components area* (abbreviated to SeCo) placed on the border of a window (right sidebar in Figure 1).

When a user moves components from the main diagram to this area, the lines between these components and remaining components are elided. SeCo consists of a list of items. Each item consists of clustered interfaces (indicated with mark (T) in Figure 1), components (U) and one corresponding symbol (S). Interfaces are clustered into two sets (T): all provided interfaces (displayed as “lollipops”) and all required interfaces (displayed as “sockets”). Numbers inside the clustered interfaces represent the number of elements clustered in the given symbol.

The purpose of symbols is to create clear and easily recognizable key, which uniquely identifies one item within SeCo. Then, these symbols can be used in the diagram area to represent connection between a given component and the corresponding item placed in SeCo. They are shown as small rectangles neighboring the displayed components (K) and containing the symbol, which corresponds to the connected item (S).

It is possible that a particular functionality of the system is implemented by several components. When this functionality is used by a large number of other components in the system, it is beneficial to represent them as a group in SeCo (M).

## V. The Separated Components Area Technique’s Implementation

SeCo technique implementation is called CoCA-Ex (Complex Component Applications Explorer). CoCA-Ex works on the ComAV platform (Component Application Visualizer) [9], so it could use ComAV’s reverse-engineering and management features. ComAV can automatically reverse-engineer the whole component-based application of a supported component model. Further component models can be easily added using an extension mechanism offered by RCP (Rich Client Platform). ComAV is also able to add other visualization styles. It means that CoCA-Ex is only one way how structures of applications can be visualized on ComAV platform.

The CoCA-Ex tool can be used via desktop application interface or web interface. In a desktop interface version, structure of all analyzed applications is saved for future visualization. Such structure is handled as a project by the ComAV – it is shown in a project view with other projects (structures), it can be renamed, deleted or updated. Such project oriented approach is known from Eclipse IDE (Integrated Development Environment). In a web interface version the user starts the visualization process by picking desired components from the local machine and uploading them to the server. The ComAV platform creates the model of the application and the CoCA-Ex shows the application diagram in the web page. The demonstration of the CoCA-Ex’s interface is shown in Figure 1.

CoCA-Ex use servlets from the JEE technology, as the back-end technology. Servlets are used mainly because of the Java implementation of the ComAV tool. HTML5, JavaScript, jQuery framework and CSS3 (Cascading Style Sheets) were used for the front-end. Canvas and SVG (Scalable Vector Graphics) elements from HTML5 (Hypertext Markup Language) are used to represent the nodes of the diagram. Although the HTML5 technology is still not fully supported by all main browsers, its current state is sufficient for CoCA-Ex purposes. Also desired features such as SVG support or Canvas are likely to be stable in the near future.

The tool provides standard features such as panning and zooming. There are two modes of manipulating the components with appropriate icons in the toolbar. First mode is for moving components (A) where the user can manually adjust the layout of the diagram. Second mode (B) serves for removing components from the diagram area to the SeCo area simply by clicking on the desired components, which should be removed. Last two icons in the toolbar serve for the automatic removal of a configured amount of components from the diagram to the SeCo area. The tool is currently configured to remove 15% of most connected components. The icon (C) is used for removing these components and adding them to SeCo area as individual items. The next icon (D) creates one group for all of them.

CoCA-Ex offers a fulltext search in components' names. In Figure 2, one can see the search for a word "relations". Seven components in the diagram contain this word as indicated by the number seven (F). Matching components are highlighted by orange color (E).

If one clicks on the provided interfaces of a component in SeCo, these interfaces and connected components become highlighted by green color. An example is shown on dependency between the *Nuxeo Common* component's provided interfaces (Y) and *Nuxeo Platform Imaging API* component (G). Similarly, for interfaces required by components in SeCo highlighting by yellow color is used. It is demonstrated on dependency between *Nuxeo URL API* component (H) and *Nuxeo ECM Web Platform UI* component's required interfaces (Y).

For several components from the SeCo area (those with symbols' background highlighted by different colors (S)) there are delegates shown in the diagram area, e.g., (K). For inspecting interfaces, the tool offers highlighting of a connection by a red color and showing the interfaces involved in the connection (P), as shown in the green tooltip. Each individual component shown in SeCo has its own button (R) to remove it back to its original position in the diagram area.

## VI. Viewport for Component Diagrams

The viewport technique shows the diagram zoomed-out to provide the appropriate overview of the complete architecture, with elements displayed without details. Besides that it shows selected components in detail inside a *viewport area* plus all their relations with other components in the diagram in an interactive border area (see gray area marked with (11.) in Figure 2). These relations are clustered into two sets for each component: all provided interfaces (displayed as "lollipops") and all required interfaces (displayed as "sockets").

These interfaces are then connected to clustered proxy components, visually represented as rectangles with rounded corners. Each rectangle represents one or more components. Numbers inside the clustered interfaces and proxy components represent the number of elements clustered in a given symbol.

## VII. Viewport for Groups of Components

This section presents the novel integration of viewport and SeCo technique. For showing groups of components (as described in Section IV) in the diagram area a viewport can be used. A group of components shown in the SeCo can be moved to the diagram area and shown as a viewport. Similarly the viewport and its content can be moved from the diagram area to the SeCo.

According to level of details of a viewport, it is possible to show:

1) a viewport as a symbol belonging to a group only,
2) a viewport with all details for all components and their relations in given group.

These possibilities are described in following sections.

### A. *Viewport with Details*

For moving a group from SeCo to the diagram area there is an icon (indicated with mark (1.) in Figure 2). After a user clicks on this icon the group will disappear from the SeCo and will be shown in the diagram area as a viewport.

Each viewport has its small toolbar, which contains a symbol representing a group (2.) and icons for important actions. The symbol has similar meaning as symbols used in SeCo. In Figure 2 there is the icon for canceling the viewport (4.). It releases the components from the viewport to the diagram area and deletes the viewport itself. Also there is the icon (3.) for moving a whole viewport to SeCo, which removes the viewport from diagram area and shows its contents in the SeCo as a group. Finally one can see icon (6.) for minimizing viewport to be represented as a icon only, which is described in following section.

### B. *Viewport as a Symbol*

One of the viewport's important features is its ability to be collapsed into an symbol (7.). It is very important part of visible elements reduction process as well as visual clutter reduction. Viewport symbol represents the whole viewport and its content. It means that components included in the viewport are not visible at the time the viewport is collapsed into the symbol. When a user hovers a mouse over the this symbol a small toolbar appears. There are icons for following actions:

- showing viewport in full details (8.), which shows viewport in a way described in previous section,
- moving viewport from diagram area into a SeCo (5.), which creates a group in SeCo from components contained in viewport,
- releasing components from given viewport and removing the viewport itself (9.).

## VIII. Discussion and Examples

In a lot of situations one can use the SeCo features to form groups of components. These groups can serve as named categories according to, which the user can classify the rest of the components in the diagram area and thus form a logical units of an investigated system.

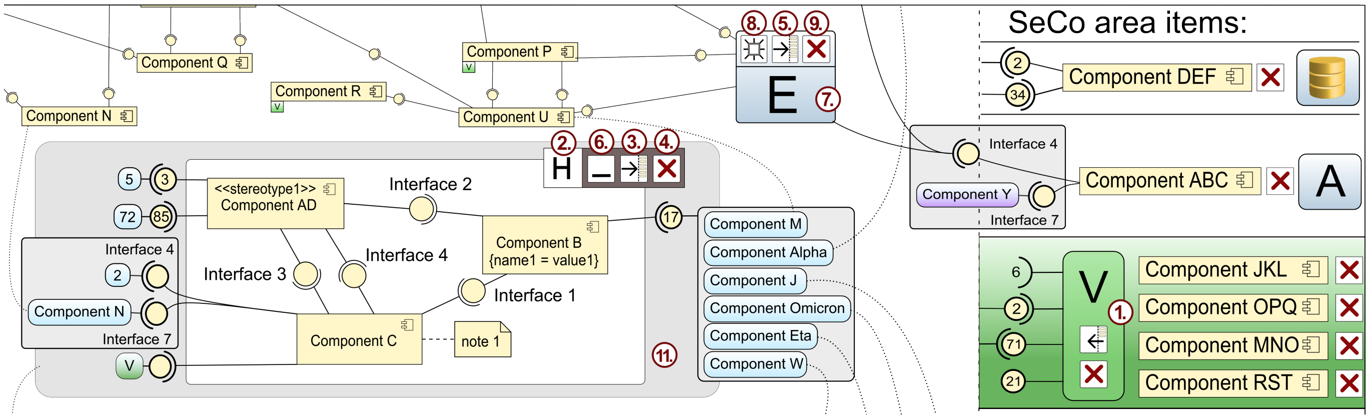

Fig. 2. Viewport with SeCo

A viewport gives an alternative to form groups in the diagram area. Its benefit is also the ability to be moved to the place in the diagram where a user forms the group. It should enable to explore and understand the dependencies in large diagrams by showing the context of a selected diagram subset. The proxy elements should reduce the need for the disorienting pan&zoom otherwise necessary while exploring dependencies and provide user relevant information in one place. The viewport is placed on a given position in the diagram, thus there can be more viewports in a diagram. At the moment when a group inside a viewport is not important, it can be collapsed into a viewport symbol. It gives a user a possibility of showing several groups and still have enough space in diagram area to work with rest of the components.

Several experiments using the proposed technique were performed. In one of them only 7 Nuxeo components have been removed from the diagram area into SeCo leading to 241 of 698 interface connection lines remaining in the graph. Therefore, the graph was reduced of about 65% of lines.

It shows that by using the proposed technique, significant visual clutter reduction may be achieved.

## IX. Conclusion and Future Work

In this paper, an advanced technique was described. This technique helps to reduce the amount of lines in UML component diagram of large applications, by removing the selected components from the diagram area. It uses SeCo where the selected components are shown, and symbolic delegates, which represent the connections instead of lines. A viewport technique was also described. This technique is used for showing all the information about interfaces for selected group of components right in the diagram area. The novel integration of above mentioned techniques was proposed. These techniques maps a group of components to the content of a viewport. Viewport symbols for graphical representation of groups were also described. These symbols saves a space in the diagram area. Appropriate interactions were proposed for all these techniques.

These techniques are, among other benefits, useful in the reverse engineering process when the user is interactively getting familiar with the whole diagram. It helps with creating the mental model of the application by easing the process of clusters creation. Which is the reason why these techniques will be part of a ComAV platform, that already supports reverse-engineering of applications of various component models.

Preliminary evaluation shows that the presented ideas are helpful in large graph visualization, where one suffers from visual clutter caused by the large number of connection lines.

Implementation of viewport technique is scheduled for integration into CoCA-Ex application to enable users to form relevant clusters comfortably and validate the ideas on concrete tasks. We also plan to evaluate above mentioned ideas by users or case study.

## Acknowledgment

The work was supported by the UWB grant SGS-2010-028 Advanced Computer and Information Systems. Authors would like to thank Jindra Pavlikova for the work on the implementation.

## References

[1] L. Holy, J. Snajberk, and P. Brada, “Evaluating component architecture visualization tools - criteria and case study,” 2012.
[2] R. Rosenholtz, Y. Li, and L. Nakano, “Measuring visual clutter,” *Journal of Vision*, vol. 7, no. 2, August 2007.
[3] G. Ellis and A. Dix, “A taxonomy of clutter reduction for information visualisation,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13, no. 6, pp. 1216 –1223, nov.-dec. 2007.
[4] D. Holten, “Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 741–748, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2006.147
[5] S. Schaeffer, “Graph clustering,” *Computer Science Review*, vol. 1, no. 1, pp. 27–64, 2007.
[6] Q. Feng, “Algorithms for drawing clustered graphs,” 1997.
[7] S. Hachul and M. Jnger, “Large-graph layout algorithms at work: An experimental study,” http://jgaa.info/ vol. 11, no. 2, pp. 345369, 2007.
[8] L. Holý, K. Ježek, J. Snajberk, and P. Brada, “Lowering visual clutter in large component diagrams,” in *16th International Conference Information Visualisation*, 2012.
[9] J. Snajberk, L. Holy, and P. Brada, “Comav - a component application visualisation tool,” in *Proceedings of International Conference on Information Visualization Theory and Applications*. SciTePress, 2012.