C. M. Herr, N. Gu, S. Roudavski, M. A. Schnabel, *Circuit Bending, Breaking and Mending: Proceedings of the 16th International Conference on Computer-Aided Architectural Design Research in Asia,* 133-142. ©2011, Association for Computer-Aided Architectural Design Research in Asia (CAADRIA), Hong Kong

GRAPH VISUALISATION IN COMPUTER-AIDED DESIGN

An exploration of alternative representations for Generative-Components^{TM} Symbolic View

KARINE KOZLOVA,[1] ROHAM M. SHEIKHOLESLAMI,[2] LYN BARTRAM[3] and ROBERT F. WOODBURY[4]
Simon Fraser University, Surrey, Canada,
1. karine.kozlova@sfu.ca, 3. lyn@sfu.ca, 4. robw@sfu.ca
2. Burnaby, Canada, rohamsh@gmail.com

Abstract. In this paper we explore graph models used to illustrate the relationships between elements of designs in computer-aided design (CAD) systems. We discuss common limitations and ways to make such representations more usable and interactive. In order to study common problems of symbolic representations in CAD systems, we conducted a survey of a number of CAD applications that employ graph representations in their interface and provided comparative analysis of the properties of graph representations in these systems. As a case study we used Bentley GenerativeComponents^{TM} (GC) system - a parametric CAD application that uses graph ("symbolic") view to visualize the structure of design. We conducted series of interviews with expert GC users that revealed many limitations of the GC symbolic view. To address these limitations, we developed alternative representations of symbolic view that aim at enhancing user experience with the system and reviewed these with expert GC users. As a result of our study, we developed a set of interactive prototypes using SHriMP[1] visualization tool and Processing programming language. These provide improved ways of user interaction with symbolic representation, including better readability of the graph and, as a result, an improved support for design model analysis.

Keywords. Graph visualization; visual interfaces; CAD systems; visual interaction; node-link diagrams.

1. Introduction

Spatial thinking and graph representations are becoming more common in software applications. The use of graph representations varies from one domain to another, and the growing use of visualizations advocates its commonality among individuals in different fields. The complexity of parametric modelling makes the use of visual representations essential. Many graph or node-link representations have been widely used in these applications. In this paper we focus our study on the symbolic view of GenerativeComponents^{TM}(GC) software.

GC is an associative parametric design system that gives architects and engineers new ways to explore alternative building forms efficiently. GC captures and graphically presents both design components and abstract relationships between them. The symbolic model window displays all features, geometric and non-geometric, currently existing in the model. This paper looks at the GC symbolic view as an example of graph visualization in CAD applications.

The target audience of this paper are designers and researchers of CAD systems as well as researchers interested in application of visualization methods in real tools. This research could be beneficial for them in several ways. The case study described in the paper could provide more insights into ways people actually use CAD applications and the issues they have to deal with to complete their tasks. Our conclusions provide list of interactive features that could be applied across multiple CAD systems that use symbolic/schematic models in their design. This would help to improve comprehension of design model and promote more effective visual interaction with such systems.

2. Background and related work

Our work is based on findings from recent information visualization and cognitive science research. This includes issues in interpreting visual attributes (Irani and Ware, 2000), specifics of human information processing (Carpenter and Shah, 1998; Huang and Eades, 1995; Huang, 2007), use of colour (edge detection, emotional response to colours) (Ware, 2004), data representation (level of detail, visual comparisons, techniques for representing large datasets compactly) (Zhao et al, 2005; Herman et al, 2000; Herman, 2001) and Gestalt principles (Ware, 2004).

Research states that the size of the graph is one of the key issues in graph visualization: as it grows, its comprehension reduces (e.g. Herman et al, 2000). In this study we use research results from visualization of graphs, networks, node-and-link diagrams, hierarchical repositories and methods for reducing the complexity of these representations.

One of the approaches to reduce graph complexity described by Ware and Bobrow (2005) suggests that the use of interactive techniques, including static and motion highlighting, can facilitate the process of getting information from graphs. Hao et al (2000) discuss an invisible link technique for linking a large highly connected graph without cluttering the display. This way only the primary links are shown to user, which makes it easier to focus on the hierarchy of interest. Herman (2001) concentrates on the use of metrics, as a measure associated with a node or an edge to be used for large graphs. The metric values can be translated into visual attributes like color and saturation so that the emphasis is put on more important nodes and edges while hiding elements with low metric values.

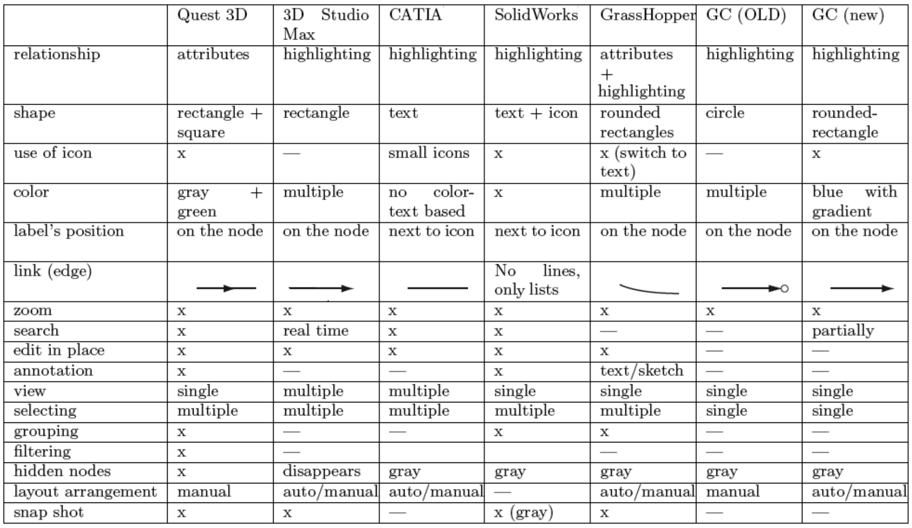
An important issue raised by Shneiderman and Aris (2006) is interactivity, and the ability to control node placement and link visibility in order to reduce the clutter and control the comprehensibility of data. Namata et al (2007) suggest the use of multiple coordinated views for network visualizations. This way graphs may be viewed and explored in parallel. In our research we propose methods to accommodate this functionality to GC symbolic view by suggesting the keymap (or an overview) of the graph or suggesting the use of two different graph layout representations: one for showing the sequence of creation, the other one for showing the dependencies. Other methods to reduce the complexity of the graph include elastic hierarchies (Zhao et al, 2005) as a way to combine different types of diagrams to reduce the space; algorithms for hierarchisation of complex, highly connected graphs (Mukherjea et al, 1995); and subdividing the graph into sub-graphs of manageable size (Herman, 2001).

The application of research studies mentioned above lays a foundation for our guidelines and interactive prototypes.

3. Software review

As a part of our research, we reviewed several software applications that have graph-based representations in their design including: Quest 3D™, 3D StudioMax™, CATIA™, GrassHopper™, and SolidWorks™. We compared graph interfaces of these tools to the symbolic model interface of GC. This review includes techniques and features that are provided in each system and the tasks that users accomplish using these features. Comparison of the tools is summarized in Figure 1.

During the time when this research was being conducted, the developers of the GC changed the symbolic model to a newer version (see Figure 2 for comparison). We were glad to see that in the new version some of our concerns were addressed. For example, search and auto layout features have been added to the new version. However, there are remaining issues, discussed below. For comparison we include both old and new versions of GC's graph representation.



	Quest 3D	3D Studio Max	CATIA	SolidWorks	GrassHopper	GC (OLD)	GC (new)
relationship	attributes	highlighting	highlighting	highlighting	attributes + highlighting	highlighting	highlighting
shape	rectangle + square	rectangle	text	text + icon	rounded rectangles	circle	rounded-rectangle
use of icon	x	—	small icons	x	x (switch to text)	—	x
color	gray + green	multiple	no color-text based	x	multiple	multiple	blue with gradient
label's position	on the node	on the node	next to icon	next to icon	on the node	on the node	on the node
link (edge)	─►─	──►	───	No lines, only lists	◟──	──►○	──►
zoom	x	x	x	x	x	x	x
search	x	real time	x	x	—	—	partially
edit in place	x	x	x	x	x	—	—
annotation	x	—	—	x	text/sketch	—	—
view	single	multiple	multiple	single	single	single	single
selecting	multiple	multiple	multiple	multiple	multiple	single	single
grouping	x	—	—	x	x	—	—
filtering	x	—			—	—	—
hidden nodes	x	disappears	gray	gray	gray	gray	gray
layout arrangement	manual	auto/manual	auto/manual	—	auto/manual	manual	auto/manual
snap shot	x	x	—	x (gray)	x	—	—

Figure 1. Comparison of graph models in CAD systems.

From one CAD application to another, the approaches to the same activity are different. Based on our analysis, the graph representation of the Quest 3D^{TM} is the most appealing among the applications that we reviewed. It provides user with multiple possibilities for graph interaction and exploration, such as edit-in-place, annotations, grouping, filtering and search (Figure 1). One feature of Quest 3D^{TM} that is worth mentioning, is the shortcut option that provides the ability to create a custom node anywhere in the graph that behaves like a quick link to the desired part of the graph. Another ability is the illustration of the attributes of each node in the graph. The attributes of each node are shown as black squares attached to the bottom of the node. This way the understanding of the relationship between each two nodes becomes more intuitive.

On the other hand, being able to select a node and move all the dependent nodes at the same time is a useful feature in 3D Studio Max^{TM} that helps the user to make a better sense of the graph.

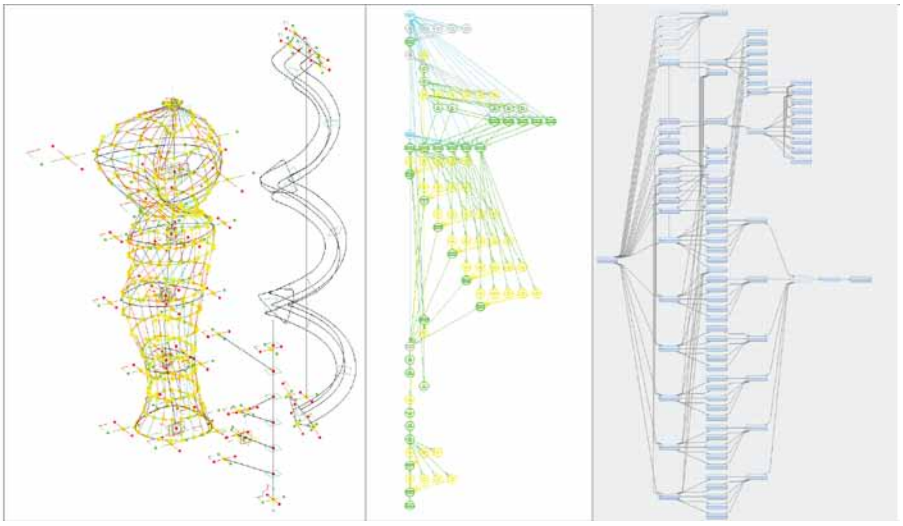


Figure 2. Example of GC Model view (left), old symbolic view (centre) and new symbolic view (right).

CATIA™ has different types of graphs that show the model. The main tree graph is based on the creation sequence of the features. Nevertheless, one can see the structural relationships in a graph as well as parent and children dependencies. Different expansion levels in the graph along with zooming of the selection are the highlights of CATIA™ graph representation. Highlighting the subsequent nodes of an object in the model by hovering the mouse on the parent node is a feature that users can benefit from in the SolidWorks™ graph representation. Ability to add annotations both in text format and by sketching is an interesting feature in GrassHopper™ that makes it more intuitive for users to work with.

4. User feedback

As a part of our research, we conducted semi-structured in-depth interviews with six expert GC users, and online interviews with three expert users. Through conducting these interviews we were trying to understand the goals and purposes that GC symbolic view was used for, and the nature of problems that users had to face during the interaction with the system. The questions addressed issues such as reasons for using symbolic view, desired functionality, ability to make sense of the model, preferred visualization capabilities, common problems, and layout issues.

We found that users worked with symbolic view primarily in order to understand the relationships and connections between nodes, for monitoring the process of model creation, searching for objects, and navigating the graph to get basic understanding of model's structure.

Problems that our participants expressed included: meaningless colors, no grouping functionality, no multiple selection for nodes, no search, no filtering options, no means for isolating parts of graph, no automatic layouts, and visual clutter that appeared with the increase of the amount of elements were the main problems GC users were dealing with. However, as we mentioned earlier, some of these problems have been addressed partially in the new version of the symbolic view. For example, the new symbolic view supports search feature in the graph. Four automatic layouts have also been added to the new symbolic view.

In the following sections we propose solutions to the above problems, and demonstrate them in interactive prototypes of GC symbolic view interface.

5. Investigation results

Based on our study, we identified and listed the features that need to be added to GC symbolic view to improve its comprehension. These features include following:

- Grouping of nodes to reduce their amount
- Multiple selection: selecting several nodes at a time
- A way to differentiate types/importance of nodes: colour, shape or size
- Better default algorithm to arrange the nodes
- Marking of important nodes
- A way to differentiate types/importance of links
- Search functionality
- Filtering: show/hide types of nodes or links
- Annotation or comments in the graph
- Edit in place: basic editing right in the graph
- Isolation of parts of the graph
- Levels of graph detail, key map
- Recording/history functionality (e.g. snapshots of previous design states)
- Layout arrangement algorithms
- Ability to move child nodes with parent nodes
- Highlighting of dependencies for the selected node

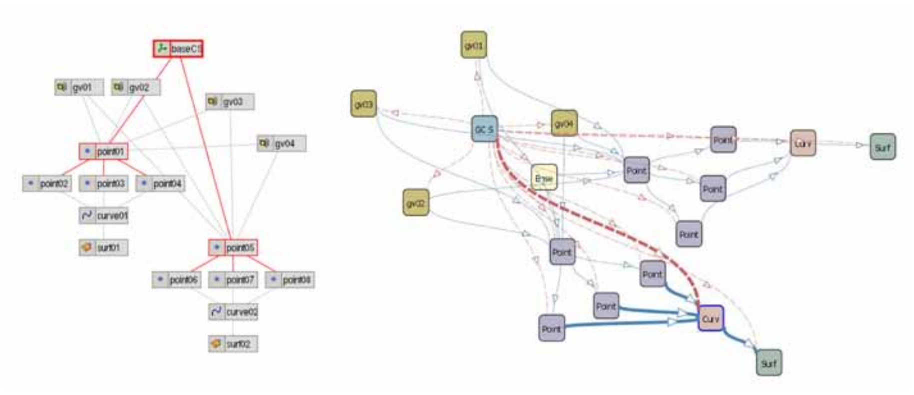


Figure 3. Interactive prototypes using Processing programming language (left) and SHriMP tool (right).

6. Prototypes

We created several paper prototypes and interactive software prototypes to demonstrate the ideas of possibilities for graph visualizations in GC. For our interactive prototypes, we tried two approaches. First, we tested our early ideas by creating a graph in Processing^{TM} (Figure 3, left). The prototype built in Processing uses icons for differentiating between types of nodes as well as interactive highlighting and node rearrangement techniques. In the second iteration, we demonstrate our findings using the SHriMP visualization tool.

The SHriMP interface includes search, filtering and highlighting options, customizing node and link colours and shapes, label styles, zoom modes, automatic layouts (e.g. radial, circular, spring, treemap), thumbnails, snapshots and manual layout rearrangement functionality. This means that all the graphs built with SHriMP are interactive, editable and allow exploration of the graph model to facilitate its comprehension. Figures 3 (right) and 4 show screen shots of SHriMP to express the desired features in the graph visualization. For example, Figure 3 (right) demonstrates the possibility of differentiating between types of edges and use of colour to differentiate between the types of nodes. Figure 4 (right) uses size of nodes to show importance levels and colour to show node types. It also uses curved edges, which increases user ability and speed of tracking paths. For better graph comprehension and reduction of clutter, the edge filtering option is included in this example.

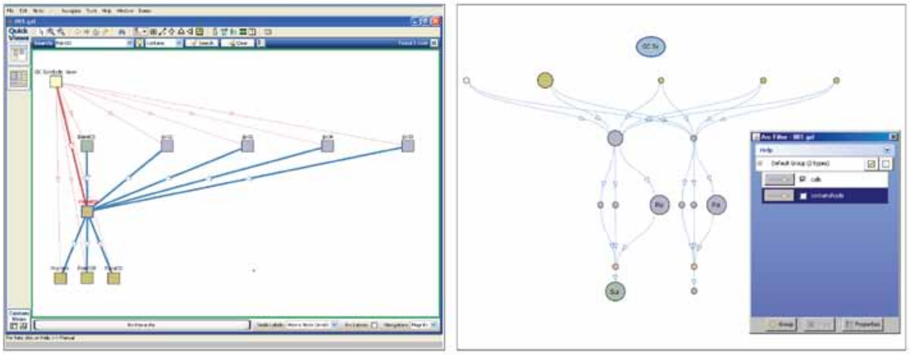


Figure 4. SHriMP interactive prototypes.

There are essential features that every graph representation should support in order for users to be able to interact with the graph. These features are: multiple selection, search and edit-in-place. Multiple selection eases graph arrangement. Further important features are: meaningful colour coding (type and relationship differentiation), which improves comprehension, and multiple views of the same graph, so one can study the graph in more detail and with different representations.

We discovered that both manual and automatic layouts have their advantages and disadvantages. For example, having system do the arrangement in a predefined layout is fast and reliable, however the ability to re-arrange the nodes in a way that makes more sense for the user is equally important. In this case, snapshots of the node arrangements may be useful, so users can record as many node arrangements as desired.

Grouping the nodes is a way to reduce the number of nodes in the graph. However, this encapsulation comes with the cost of losing or hiding some of the links and relationships. By grouping several nodes, one node that represents all the grouped nodes with links that show connections of each member inside the group with the rest of the graph, is created. This way, it becomes impossible to see which node in the group is connected to a certain node outside it unless that group is opened. In addition, the relationships among the group members become invisible.

From this study we conclude that the context of a specific visualization is the primary criterion by which the decision of whether to apply a specific method or not, should be made.

7. Discussion

Design considerations and interactive prototypes for symbolic view are initial steps in improvement of this important part of GC system. Additionally, the design principles, visualization techniques, and interaction features discussed in this paper can easily be generalized for designing and/or improving graph representations in other CAD applications. This work is an exploratory study, and we suggest the above analysis and suggestions to be further developed and validated with system expert users.

In software review we mainly focused on the CAD applications, which might not cover the complete area for the hierarchical graph representations. On the other hand, we tried to compare the appropriate applications that have similar usage as GC. By doing so, we realized that GC symbolic view lacks very basic visualization functionality and does not provide interactivity.

In this study we focused on the general visualization principles and perception issues rather than on specific user tasks. However symbolic representation may highly depend on the context of work the user is performing (Purchase et al, 2001). Undoubtedly, task-based user evaluation for the proposed solutions should be considered as the next phase of this project. Through this type of evaluation interactive prototypes should be tested on their ability to support users at various stages of the design process.

8. Conclusion

In this paper we presented a study that explored graph views used in CAD software and provided a set of features that can improve user interactions with this type of representations. Our results take into account the specifics of human visual information processing and perception, findings from visualization science and graph visualization research. We used the example of GC (a parametric CAD system) as a case study. Features that we consider important to be present in symbolic graph representations include search function, isolation of the part of the graph, interactive highlighting, filtering, grouping, keymap, multiple selection, types of zoom and layout algorithms. The primary goal of this research is to provide ways for improvement of current graph functionality in CAD systems by supporting interactivity, reducing clutter and promoting better and easier understanding of the model by taking advantage of various visualization approaches for nodes, links and layouts.

Endnotes

1. www.thechiselgroup.org

Acknowledgements

This work was partially supported through the SAP Academic Research Center Fellowship program, the Canadian Natural Science and Engineering Research Council Discovery Grants and Collaborative Research and Development Programs; Bentley Systems, Incorporated; and the MITACS Accelerate program.

References

Carpenter, P. A. and Shah, P.: 1998, A model of the perceptual and conceptual processes in graph comprehension, *Journal of Experimental Psychology: Applied*, **4**(2), 75 - 100.

Hao, M. C., Hsu M., Dayal, U., and Krug, A.: 2000, Web-Based visualization of large hierarchical graphs using invisible links in a hyperbolic space, 83 – 94.

Herman, I.: 2001, Graph visualization of complex information, *ERCIM News* **44**.

Herman, I.; Melancon, G. and Marshall, M.S.: 2000, Graph visualization and navigation in information visualization: A survey, *IEEE Transactions on Visualization and Computer Graphics*, 6(1), 24 – 43.

Huang, W.: 2007, *Using eye tracking to investigate graph layout effects*, 97 - 100.

Huang, W. and Eades, P.: 2005, How people read graphs, *AustralianComputer Society*, Sydney, Australia, 51 – 58.

Irani, P. and Ware, C.: 2000, Diagrams based on structural object perception, *ACM,* Palermo, Italy, 61 – 67.

Mukherjea, S, Foley, J. D. and Hudson, S.: 1995, Visualizing complex hypermedia networks through multiple hierarchical views, *ACM*, Denver, Colorado, United States, Press/Addison-Wesley Publishing Co., 331 – 337.

Namata, G. M., Staats, B., Getoor, L. and Shneiderman, B.: 2007, A dual-view approach to interactive network visualization, *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, *ACM*, Lisbon, Portugal, 939 -942.

Purchase, H. C., McGill, M., Colpoys, L. and Carrington, D.: 2001, Graph drawing aesthetics and the comprehension of UML class diagrams: an empirical study, *Australian Computer Society*, Sydney, Australia, 129 – 137.

Shneiderman, B. and Aris, A.: 2006, Network visualization by semantic substrates, *IEEE Transactions on Visualization and Computer Graphics*, **12**(5), 733 – 740.

Ware, C.: 2004, *Information Visualization: Perception for Design*, Morgan Kaufman, San Francisco, CA, 2nd ed edition.

Ware, C. and Bobrow. R.: 2005, Supporting visual queries on medium-sized node-link diagrams, *Information Visualization*, **4**(1), 49 - 58.

Zhao, S., McGuffin, M. J. and Chignell, M. H.: 2005, Elastic hierarchies: Combining treemaps and Node-Link diagrams, *Proceedings of the 2005 IEEE Symposium on Information Visualization*, October 23-25, 2005, 8.