Qualitative Visualization of Processes: Attributed Graph Layout and Focusing Techniques

Kent Wittenburg[1] and Louis Weitzman[2]

[1]Bell Communications Research, 445 South St., Morristown, NJ 07960 USA
kentw@bellcore.com

[2]Meta Media Design, 49 Melcher St., Boston, MA 02210 USA
weitzman@media.mit.edu

Abstract. Many techniques and algorithms have been proposed for layout and interactions with basic node and link graphs. Here we consider layout and focusing techniques for attributed graphs and show an application of these techniques in a Bellcore system for support of business process design and reengineering called ShowBiz. ShowBiz supports two primary visualization techniques for qualitative analysis of flowgraphs. The first involves the use of parsing to perform graph reduction for encapsulation and focusing. The second is an attributed graph layout technique we have called RB-layout in recognition of its genesis in Rummler-Brache methodology for business process reengineering [7]. The user selects attributes that partition the set of nodes in the graph and the system generates layouts in which the position of the nodes is determined by the value of the attribute in question.

1 Introduction

Despite the large literature on layout and focusing techniques for basic node and link graphs [4][6], there have been relatively few general techniques proposed specifically for attributed graphs. Standard practice treats attribute information as "adornment," i.e., drawing nodes and/or links with differing icons, color, or size. Here we present novel layout and focusing techniques specific to attributed graphs that have been motivated by an application in business process design and reengineering. We begin with an overview of the ShowBiz tool, followed by illustrations of (1) our use of grammars for support of flowgraph abstraction and focusing and (2) a layout technique that enables visualization and editing of attribute spaces. We then discuss these techniques in the context of other methods proposed for graph layout and focusing.

2 ShowBiz

The process flow modeling tool called ShowBiz has been created by Bellcore Applied Research for internal customers in the Bellcore Professional Services organization. The primary users of this tool consult with clients to improve the efficiency of existing business processes or else design new business workflows occasioned by the introduction of new technology or new forms of business. To date, work has focused on activities in the work centers of large telephony concerns such as the Regional Bell Operating Com-

panies and also on the publishing and fullfillment processes surrounding premier World Wide Web sites. An important function of such a modeling tool is to facilitate rapid qualitative understanding of complex processes. The typical life-cycle of a project for these consultants involves iterating with their customers on representations of current methods of operations. Once the basic representations are agreed upon, subsequent stages may require data gathering to support quantitative analysis and simulation. Then improved future methods of operations are designed and ultimately implemented, but not necessarily with the involvement of the same part of the Bellcore organization.

ShowBiz is designed to allow users quickly and easily to build customizable process models. It supports a transition from the early stages of a design problem to more structured uses of the information in which data can be exported to external tools for simulation, analysis, and/or code generation. It also includes one-step World Wide Web publishing capabilities: whatever view the user can construct can be exported for information sharing on the WWW in the form of an imagemap along with automatically generated and linked html files that contain attribute information for the underlying task objects.

3 Dynamic Aggregation of Subflows

Visualization plays an important role in supporting rapid qualitative understanding of workflows. It is important to be able to easily hide and reveal levels of detail and construct alternative views of complex operations. In order to create more highly structured and concise models, reusable workflow components should be identified. Just as well-structured programming languages facilitate understanding and design, well-structured information models do the same. Aggregation operations in ShowBiz are designed to facilitate the creation of well-structured models and views in support of qualitative understanding of workflows.

Existing commercial flowcharting and process modeling tools standardly support the feature of hierarchically structured flowgraphs, where a single node in a graph can be expanded into another window, in which more detail is shown. However, these hierarchical structures must be assembled by hand and, once created, they are not easily changed. A distinguishing feature of ShowBiz is that users can form alternative hierarchies quickly and easily for the purposes of encapsulation or viewing.

Figure 1 shows a screen dump of a "home view" for a ShowBiz flowgraph, in which users can enter process flow information. The nodes with dog-eared corners represent reduced subflows that can either be expanded into the current diagram or viewed in another window. As users iterate on their models or construct views to focus on aspects of the representation, they can freely form new graph aggregations by invoking a parsing operation. The parser expects there to be one or more nodes pre-selected. It then expands out from the selected items seeking a derivation that includes the selected items and that constitutes a procedural block as defined by the flowgraph grammar. See Wittenburg and Weitzman [12] for a detailed discussion of the flowgraph grammar used in the application. The parser terminates with the first such derivation found and selects the subgraph contained within. The user can then operate upon the selected subflow for viewing or data organization purposes, replacing the selected subgraph with one of the dog-eared nodes.

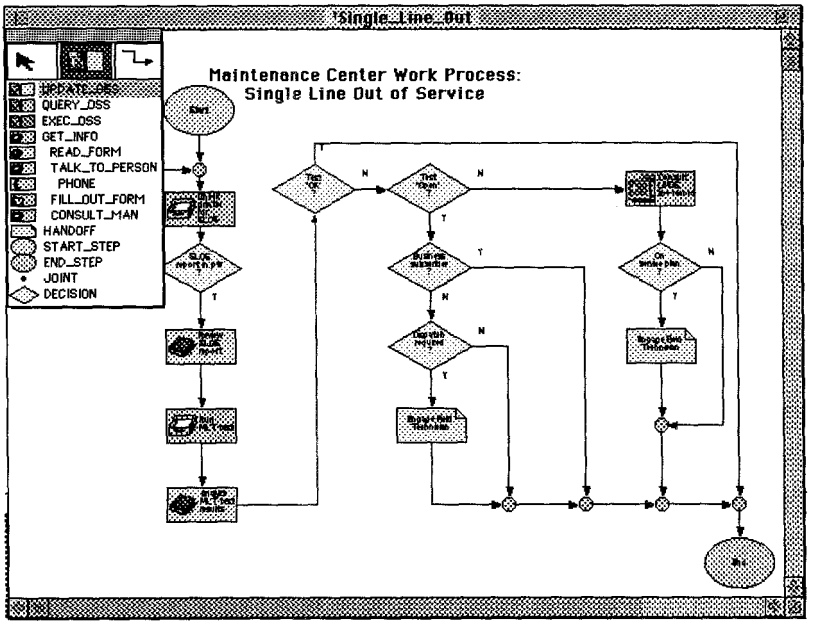

Fig. 1. A ShowBiz home view screen shot.

For example, let us presume that the user has selected the node labeled "Business Subscriber?" appearing near the center of Figure 1. A menu item labeled "Extend Selection to Subflow Group" would then be enabled, and, if chosen, would invoke a parsing operation. The result of such a parse in this case would be the group selection of the subflow nodes shown with heavy outlines in Figure 2. (The selection of arcs connecting a selected group of nodes is implicit.) It is guaranteed through the grammar that such a subflow meets all the requirements for encapsulation. The user may continue to invoke this grouping operation until a subgraph of the appropriate size is selected.

Once the larger group has been selected, other operations are enabled as shown in the menu in Figure 2. Users may reduce the subflow to a node in the same diagram or encapsulate the subflow for subsequent reuse by moving it to a separate file. This operation is similar to cut and paste, except that in addition it splices in an encapsulated subflow node in place of the subflow graph being removed and takes care of some bookkeeping operations such as adding *Start* and *End* nodes to the newly created subflow diagram.

Reduction of visual complexity through such aggregation operations is only part of the story towards creating effective visual presentations. The space freed up by reducing a subgraph should now be utilized by other graph elements still present. And of course the complementary insertion operations need to somehow make room for subgraphs that may require far more graphical real estate than the original encapsulated subflow node. While we have not achieved an automated solution to this space mangement problem when users are in the manual layout mode shown in Figures 1 and 2, reductions or expansions do trigger new layouts when users are in the automated layout

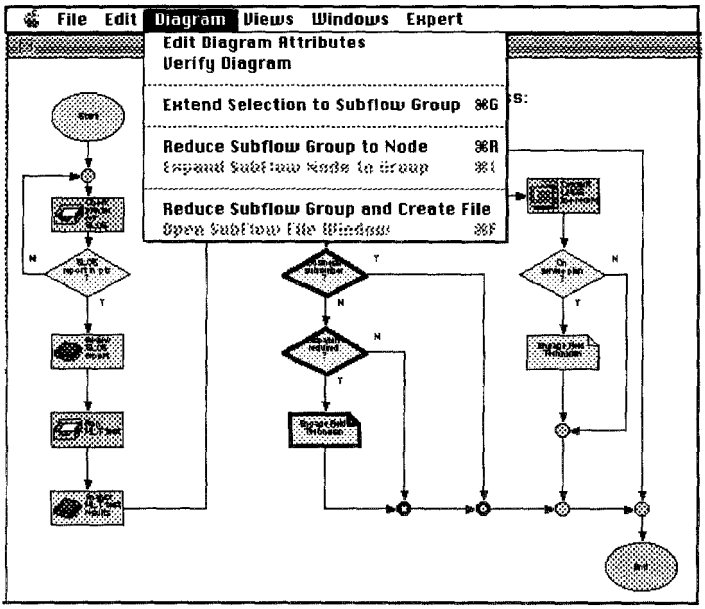

Fig. 2. Extending a selection to a subflow group

mode described in the next section. As has been noted by others who have experimented with sequences of user-directed operations that occasion new graph layouts (e.g., [5]), a remaining issue here is one of visual continuity. It is easy for users to become momentarily disoriented as an entire graph is erased and replaced. We expect to address this issue in future work.

4 Attribute-based Layout

In the views menu of ShowBiz, users are offered the option of specifying a "Rummler-Brache" layout, which places the system in automatic layout mode. In Figure 3 we show the sequence of dialogs for specifying such a layout. The user first has to choose an attribute that will partition the flowgraph's nodes. This choice list is determined by the attribute type definitions in the model. Next a dialog box pops up that lists the value choices for that attribute type, in this case, the type is *database*. Choices are preselected that currently are instantiated somewhere in the graph, but the user may override this default if, for example, editing actions are intended or some alternative focus is desired.

Figure 4 shows the layout that results from the specification in Figure 3. The left-hand pane lists the chosen values of the attribute, which determine the number and order of rows in the layout. The inital row is for "not applicable," used when either the node's class definition does not contain that attribute or else the attribute's value is unbound. These views project the process flow onto an attribute space that immediately reveals attribute values in the context of the overall control flow of the process. Such views can reveal important aspects of a process flow, such as all the points at which handoffs oc-

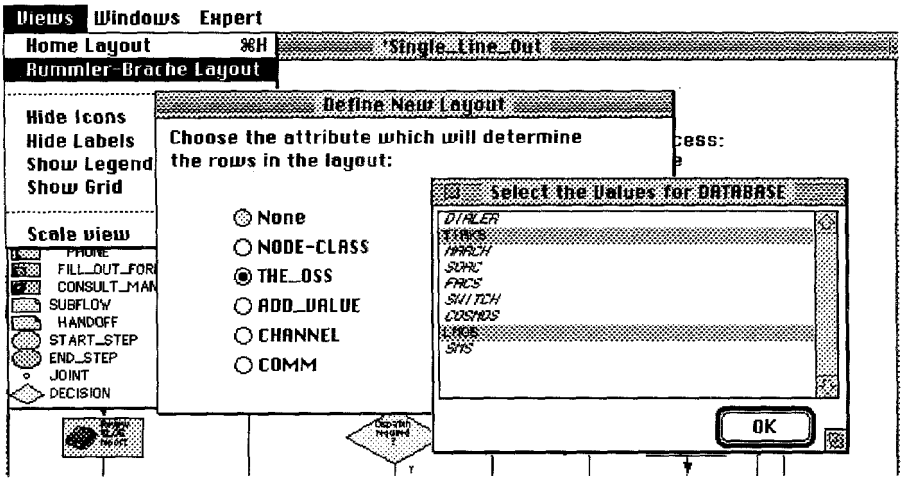

Fig. 3. Specifying an attribute-based layout.

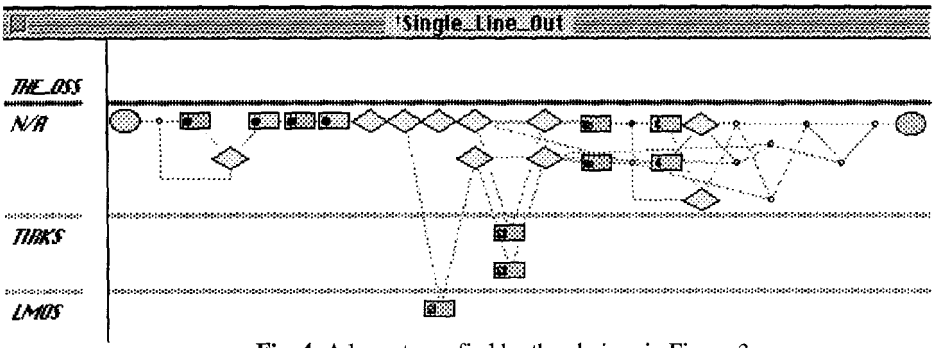

Fig. 4. A layout specfied by the choices in Figure 3.

cur across organizational boundaries or control is passed between systems. The Rummler-Brache methodology proposes creating such a process map to reveal organizational handoffs in a business process flow. Good design for business processes encourages minimizing such handoffs and maximizing accountability of a particular organization in the complete process.

In Figure 5 the user has chosen a different focus through a layout based on *node-class*. (Technically, node-class is not an attribute, but we can treat it as such in the layout.) Note the indentation in the value pane -- these values form an object-oriented hierarchy. The user could choose to consolidate values higher in the hierarchy, in which case determining row position for nodes in the main pane would make use of inheritance. The user has also simplified the diagram through the aggregation methods discussed in Section 3 -- thus we see fewer nodes than in Figure 4. The user is also in the middle of editing. Dragging a node from one lane to another is interpreted by the system as an editing action. In this case the user is changing the node class of the node under the cursor.

The algorithm employed for attribute-based layout is roughly as follows. A variant

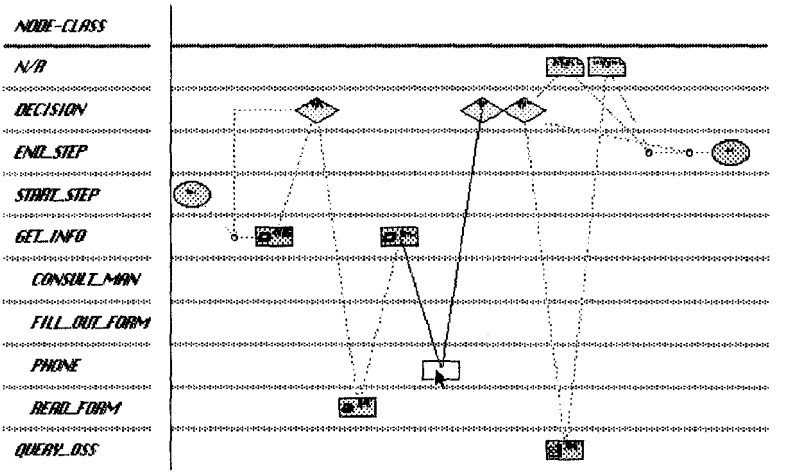

Fig. 5. Editing hierarchical values by drag and drop.

of the initial phase of the Sugiyama algorithm [10] determines the topological layers in the graph, which in our case sets the x-values of the node positions. We assign the x layers starting from the source of the graph rather than from the sinks in order for the node to appear at the least x value consistent with topology. An initial y-position for each of the attribute value rows is then computed and assigned to the nodes based on the node's value of the attribute. The initial y-value assigment is based on a simple linear ordering that derives from the type definition of the attribute. If the attribute's type is hierarchical, the order is arbitrary within a given level of the hierarchy tree. The nodes are then sorted first by y-value and then secondarily by x-value. We iterate through the sorted node list looking for ties of x/y values. When a tie is found, the y-value of each tying node is incremented, which has the effect of widening the swimming lane in the overall layout and incrementing the y of each succeeding attribue value in the ordering.

5 Discussion

Our approach to this interface is consistent with some of our earlier work in interfaces for design support [11]. We have attempted to allow the system to support the user in designing effective views of underlying data while allowing the user to interactively fine-tune the layout and override system decisions. Thus while a user drags a node or a node group, the system incrementally computes external arc positions within a certain distance from the changed nodes. The system constantly gives feedback to the user of what the result will be once the drag terminates. However, the system is not always successful (particularly since our algorithm adds at most two bends to each arc), and thus the interface adds a feature whereby the user can manually alter the position to which an arc connects to a node and then lock the arc to this position. A limitation of the current implementation is that the user cannot similarly override the system's decision of where to position arc bends. There is room for improvement as well in the arc layout methods that the system does use.

The use of parsing for aggregation is comparable to graph aggregation techniques found in many tools presented at previous Graph Drawing workshops such as VCG [8], daVinci [3], or Kimmelman et al. [5]. The difference is that we employ a declarative grammar-based description. Reducible clusters can be defined topologically but also by utilizing relations and predicates determined by attribute information. One of the user-interface issues that is not addressed by some of the other systems is how to define natural clusters that users can easily understand and specify. Our impression is that, for well-structured flowgraphs, users can easily work with the hierarchical decomposition that is a natural outcome of a context-free grammatical description of the visual language. Although graph grammars have been proposed elsewhere in the context of graph layout (e.g., Brandenburg [2]), we are not aware of any other work that employs higher-dimensional grammars in the service of aggregation for information modeling and focusing.

Alternative visual focusing methods, surveyed by Noik [6], include generalized zooming interfaces such as Pad++ [1] and fisheye transformations [9] as well as a variety of abstraction operations including node and arc hiding and/or ghosting [5]. There is a need for both geometric-based operations in support of visual focusing as well as for logically-based aggregation operations. While one may want to verify that a subflow has all the necessary structure to be encapsulated, one does not always need to do local rule-based aggregation for visual focusing. For example, one may want to de-emphasize all nodes of a certain type whether or not they are connected or form a logically coherent group. We see the logical and the geometric methods as largely complementary.

Note that the focusing techniques just discussed are confined to the adding, deletion, magnification, or demagnification of nodes and arcs. The attribute-based layout technique we presented, however, is another type of focusing mechanism. It allows a user to focus on attribute spaces that cut across all visible graph elements. The value of Rummler-Brache process maps is well-established as a pencil-and-paper method for visualizing organizational participation in business processes. We have automated the technique and applied it to attributed graphs more generally. The algorithm requires only that graphs be directed and that eligible attributes partition the nodes of the graph. It is useful not only for visualization but also for drag-and-drop editing.

Ongoing work includes application of these techniques to other domains such as World Wide Web management and information access. An open problem is the use of grammatical descriptions for incremental editing -- the aggregation methods mentioned here succeed only if the (sub)graphs conform to the grammatical descriptions, but satisfactory imcremental parsing methods are still needed when dealing with two-dimensional visual languages such as directed attributed graphs [13].

6 Acknowledgments

Contributers to the ShowBiz implementation include Cliff Behrens, Ron Dolin, Mark Rosenstein, Loren Shih, and Larry Stead. We are also indebted to Rodney Fuller, Catherine Hanson, Bob Root, and Laurie Spiegel. The research has been supported in part by ARPA grant N66001-94-C-6039.

References

1. Bederson, B., and J. Hollan. (1994) Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. Proceedings of ACM UIST `94 (Marina Del Rey, CA, USA), ACM Press, pp. 17-26.

2. Brandenburg, F. (1995) Designing Graph Drawings by Layout Graph Grammars. In R. Tamassia and I. G. Tollis, eds., Graph Drawing: DIMACS International Workshop, Princeton, NJ, USA, October 1994, Proceedings, Lecture Notes in Computer Science 894, Springer-Verlag, pp. 416-427.

3. Froelich, M., and M. Werner (1995) Demonstration of the Interactive Graph Visualization System da Vinci. In R. Tamassia and I. G. Tollis, eds., Graph Drawing: DIMACS International Workshop, Princeton, NJ, USA, October 1994, Proceedings, Lecture Notes in Computer Science 894, Springer-Verlag, pp. 266-269.

4. Di Battista, G., P. Eades, R. Tamassia, and I. Tollis (1994) Algorithms for Drawing Graphs: an Annotated Bibliography. Computational Geometry: Theory and Applications 4:235-282. [Updated version at http://www.cs.brown.edu/people/rt/gd-biblio.html.]

5. Kimelman, D., B. Leban, T. Roth, and D. Zernik. (1995) Reduction of Visual Complexity in Dynamic Graphs. In R. Tamassia and I. G. Tollis, eds., Graph Drawing: DIMACS International Workshop, Princeton, NJ, USA, October 1994, Proceedings, Lecture Notes in Computer Science 894, Springer-Verlag, pp. 218-225.

6. Noik, E.G. (1994) A Space of Presentation Emphasis Techniques for Visualizing Graphs. In Graphics Interface '94, pp. 225-234.

7. Rummler, G. A., and A. P. Brache (1990) Improving Performance: How to Manage the White Space on the Organization Chart, Jossey-Bass Publishers.

8. Sandar, G. (1995) Graph Layout Through the VCG Tool. In R. Tamassia and I. G. Tollis, eds., Graph Drawing: DIMACS International Workshop, Princeton, NJ, USA, October 1994, Proceedings, Lecture Notes in Computer Science 894, Springer-Verlag, pp. 194-205.

9. Sarkar, M., and M. Brown. (1992) Graphical Fisheye Views of Graphs. In Proceedings of CHI '92, Monterey, CA, May, 1992, pp. 83-91.

10. Sugiyama, K., S. Tagawa, and M. Toda (1981) Methods for Visual Understanding of Hierarchical System Structures. IEEE Transactions on System, Man, and Cybernetics 11:1047-1062.

11. Weitzman, L., and Wittenburg, K. (1993) Relational Grammars for Interactive Design. In Proceedings of IEEE Symposium on Visual Languages, August 24-27, Bergen, Norway, pp. 4-11.

12. Wittenburg, K., and L. Weitzman (1996) Relational Grammars: Theory and Practice in a Visual Language Interface for Process Modeling. In Proceedings of Workshop on Theory of Visual Languages, May 30, 1996, Gubbio, Italy. [http://www.cs.monash.edu.au/~berndm/TVL96/tvl96-home.html.]

13. Wittenburg, K. (1995) Visual Language Parsing: If I Had a Hammer... In Proceedings of CMC/95: International Conference on Cooperative Multimodal Communication, Theory and Applications, Eindhoven, The Netherlands, May 24-26, 1995, pp. 17-33.