Constraint-Based Layout in Visual Program Design*

Winfried H. Graf Stefan Neurohr

German Research Center for Artificial Intelligence (DFKI) GmbH
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
Email: {graf,neurohr}@dfki.uni-sb.de

1 Introduction

In the expanding field of visual programming, layout and design tasks are crucial points. Particularly, knowledge-based editing facilities can be seen as integral parts of the next generation's visual programming tools [2, 3]. One essential drawback of current visual programming environments is their lack of providing adequate tools for maintaining graphical style, legibility, transparency, adaptivity, and consistency of graphical and multimedia presentations in interactive dynamic settings.

By the example of *InLay*, the constraint-based graphical editor of the multimedia layout manager *LayLab* [5], we address the incorporation of AI aspects in the visual software design process and the automated layout of informational graphical schemata. Following our previous work on constraint-based graphical layout [6], we show how advanced constraint processing techniques, such as prioritizing constraints and dynamic constraint satisfaction, can be widely applied to visual programming and program visualization environments in order to maintain graphical style and consistency for adaptive layouts.

2 Visual Program Layout

Layout of large and complicated visual presentations is laborious to produce by hand. Here, it is unacceptable to spend more time on the visualization of the application information to be presented than the application algorithm itself. For example, in a program visualization context, the user does not want to waste time with layout specifications.

Numerous visualization systems for producing diagrams automatically from program code and generating static graphical displays of data structures as well as algorithm animation have been delevoped [7]. Most dedicated traditional algorithms for drawing graphs exploit heuristics to overcome the complexity problem [1], but they are not flexible enough and do not allow to add user preferences. Some of them propose less effective canonical displays which are difficult to create for linked and nested structures. Other approaches such as *Gelo* [8] include predefined data views and allow the graphical specification of topological constraints by the user, or address a declarative formalism for the definition of graphical layout in the context of visualizing object-oriented programming systems [4]. Essential shortcomings of most systems are that the displays are not aesthetically pleasing and cannot be tailored to the user's needs. Apparently, no work has been done on applying graphic design expertise to visualizing data structures.

*The research reported in this paper has been carried out in the $\mathcal{PPP}$ project which is supported by the German Ministry for Research and Technology under contract ITW 9400.

3 The Constraint-based Editor InLay

The constraint-based graphical editor *InLay* supports interactive graphical and multimedia layout and design, pre/post-editing and beautification tasks as well as the intelligent use of innovative interaction and visualization techniques. Therefore, we have extended *LayLab*'s dedicated constraint solver model [6], that is mainly based on the *DeltaBlue* [9] constraint hierarchy solver, by mechanisms to handle user interaction and dynamic input by indirect reference constraints.

Since software visualization systems are frequently limited in the breath and extensibility of their displays as they do not allow user-defined displays, we have made all decision processes sensitive to design parameters such as user's layout preferences, presentation type, presentation intent, output medium, and resource limitations.

Moreover, since layout in visual programming interfaces frequently requires a direct manipulation responsiveness, an incremental update of the generated graphical material is necessary. As the user browses or edits a dynamic graphical presentation, the system updates visibility changes smoothly to avoid visual discontinuities. All views of a flow chart are redesigned simultaneously in realtime to maintain a set of visibility constraints automatically as the viewing specification changes.

In order to perform all these tasks, *InLay* has at its disposal a large knowledge base comprising presentation structures, graphic-design heuristics, commonsense knowledge about layout, interaction and visualization techniques as well as a user-/domain-specific knowledge.

4 A Small Example

Let's have a short look at some snapshots taken from our visual programming testbed *InLay* (Fig. 1)

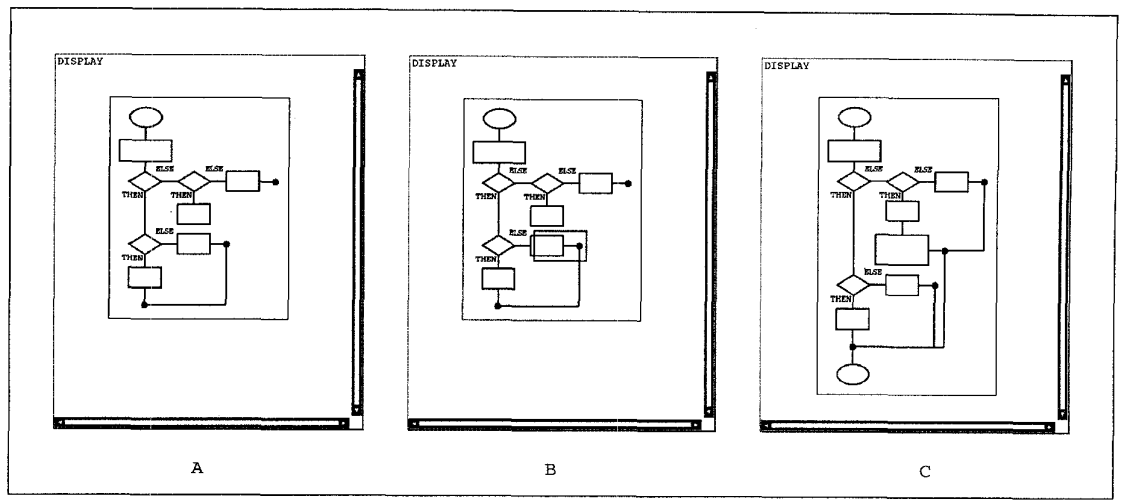

Figure 1: Constraint-based visual programming with InLay

showing the insertion of a new visual item in a program flowchart. Here *InLay* supports the layout of the visual programs by automatically beautifying the display after adding and connecting operations.

5 Implementation and Applications

A prototype of *InLay* has been developed as a major component of the overall layout manager *LayLab* [5] that is available on Sun SPARCstations and Silicon Graphics machines under UNIX implemented using Allegro Common Lisp/CLOS and CLIM. *LayLab* has already been approved for automatic layout control in the multimedia presentation system *WIP* [10] and is currently going to be extended for dynamic interactive settings. Beside visual programming environments there arise numerous potential application domains that suffer from visual design and consistency problems, such as different kinds of network diagrams (e.g., workflow charts), the broad field of intelligent multimedia interfaces, CASE tools, CSCW, and virtual realities.

6 Conclusion and Future Work

Visual program layout and program visualization make a number of useful contributions for all kinds of graphical editing tasks. We have described the constraint-based graphical editor *InLay* that automatically handles aspects of display layout in visual programming environments. Furthermore, innovative visualization and interaction techniques for dynamic displays can generate new insights and can be seen as a new quality of communication media. In current work, we concentrate on the intelligent use of visualization techniques: animated layout of chart diagrams, abstracting presentation parts, focussing of active display objects, editing graphical histories, and the visualization of hierarchical information structures.

References

[1] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Algorithms for Drawing Graphs: an Annotated Bibliography*, 1993.

[2] E. P. Glinert, Ed., *Visual Programmimg Environments: Applications and Issues*, IEEE Computer Science Press, Los Alamitos, CA, 1990.

[3] E. P. Glinert, Ed., *Visual Programmimg Environments: Paradigms and Systems*, IEEE Computer Science Press, Los Alamitos, CA, 1990.

[4] V. Haarslev and R. Möller, "Visualization and graphical layout in object-oriented systems", *Journal of Visual Languages and Computing*, no. 3, pp. 1–23, 1992.

[5] W. H. Graf, "LayLab: A constraint-based layout manager for multimedia presentations", in *Human-Computer Interaction: Software and Hardware Interfaces*, G. Salvendy and M. J. Smith, Eds., pp. 446–451. Elsevier, Amsterdam, 1993.

[6] W. H. Graf, "Constraint-based graphical layout of multimodal presentations", in *Advanced Visual Interfaces*, T. Catarci, M. F. Costabile, and S. Levialdi, Eds., World Scientific Series in Computer Science - Vol. 36, pp. 365–385. World Scientific Press, Singapore, 1992.

[7] M. H. Brown, *Algorithm Animation*, ACM Distinguished Dissertations. MIT Press, Cambridge, MA, 1988.

[8] C. Duby, S. Meyer, and S. P. Reiss, "Using GELO to visualize software systems", in *Proceedings of the UIST'89 (ACM SIGGRAPH Symp. on User Interface Software and Technology)*, Williamsburg, VA, 1989, pp. 149–157.

[9] B. Freeman-Benson, J. Maloney, and A. Borning, "An incremental constraint solver", *Communications of the ACM*, vol. 33, no. 1, pp. 54–63, 1990.

[10] W. Wahlster, E. André, W. Finkler, H.-J. Profitlich, and T. Rist, "Plan-based integration of natural language and graphics generation", *Artificial Intelligence, Special Issue on Natural Language Processing*, vol. 63, 1993.