On Comments in Visual Languages

Christoph Daniel Schulze, Christina Plöger, and Reinhard von Hanxleden

Deptartment of Computer Science, Kiel University, Kiel, Germany
{cds,cpl,rvh}@informatik.uni-kiel.de

Abstract. Visual languages based on node-link diagrams can be used to develop software and usually offer the possibility to write explanatory comments. Which node a comment refers to is usually not made explicit, but is implicitly clear to readers through placement and content. While automatic layout algorithms can make working with diagrams more productive, they tend to destroy such implicit clues because they are not aware of them and thus do not preserve the spatial relationship between diagram elements. Implicit clues thus need to be inferred and made explicit to be taken into account by layout algorithms.
In this paper, we improve upon a previous paper on the subject [9], introducing further heuristics that aim to describe relations between comments and nodes. These heuristics mainly help to reduce the number of attachments of comments that should not be attached to anything. We also derive propositions on how developers of visual languages should integrate comments.

1 Introduction

Visual languages are in widespread use for developing software, either in addition to or at the expense of more traditional text-based languages. Languages such as ASCET (ETAS Group) or LabVIEW (National Instruments) allow developers to define software systems using *node-link diagrams* such as the one in Figure 1: *nodes* (or *actors*) are entities that consume and produce data, which are transmitted between nodes through the *links* connecting them. Most visual languages support comments, usually in the form of special kinds of nodes that display text. However, finding out which node a comment refers to can be a challenge because of the two-dimensional nature of positioning them. Some languages solve this problem by allowing developers to explicitly attach comments to diagram elements and visualizing the attachment with a line. But not every language supports this feature, and not every developer uses it if it does. Developers often seem to rely on other, more implicit clues instead, such as the distance between comments and nodes. This falls into the category of secondary notation [5].

For diagrams to be understandable in the first place, their elements have to be carefully placed on the drawing area. Layout algorithms can reduce this effort by positioning nodes and routing edges automatically. However, unless a comment is explicitly attached to the node it refers to, the layout algorithm has

no knowledge of their relation and they may end up in vastly different places in the final layout. This wreaks havoc with the implicit clues that would have allowed a viewer to understand which node a comment refers to. This problem mainly applies to *layout creation algorithms*, which calculate new layout from scratch, as opposed to *layout adjustment algorithms*, which try to clean up an existing layout while preserving spatial relationships [4]. However, many layout algorithms used in practice fall into the former category. To use them in spite of their problems with comments, it is necessary to infer attachments between comments and nodes to make them explicit for the layout algorithm. This is what we introduced as the *comment attachment problem* in previous work [9].

Contributions. In previous work, we evaluated comment attachment based on the distance between comments and nodes [9]. This resulted in a lot of comments that should have been left unattached, but were attached to nodes by the presented algorithm, which is what we call *spurious attachments*. Here, we introduce a number of additional heuristics that ultimately serve to reduce the number of spurious attachments. We evaluate them on a set of Ptolemy diagrams and draw conclusions on how to properly integrate comments into visual languages. Note that for this paper, we limit outselves to attachments between comments and nodes and leave attachments between comments and other elements, such as edges and ports, for future work. More details are available in a technical report [7].

Use Case. Ptolemy is a visual language based on node-link diagrams developed at UC Berkeley [6] that supports comments in the form of nodes that contain text. The KIELER Ptolemy Browser[1] allows users to browse through a Ptolemy model along with its submodels in a single window by dynamically expanding or collapsing nodes that contain further models, which requires automatic layout algorithms. The layout algorithm we use [8] is a layout creation algorithm based on the hierarchical layout method first introduced by Sugiyama et al. [10]; we therefore need comment attachment to keep comments close to the nodes they implicitly refer to.

Ptolemy ships with a set of demo models intended to showcase different models of computation, actors, and development techniques. 348 of them, created by 40 different developers, will serve as our main data set throughout this paper. Overall, the models averaged 21.4 nodes as well as 3.1 comments per model, of which 182 (about 17%) refer to a specific node.

Related Work. We are not aware of any studies on how developers use comments in visual languages. However, the usage of documentation systems such as Javadoc have been studied, for example by Kramer [3], but the results do not seem to be applicable to our domain: Javadoc has clear rules on what comments refer to, which visual languages usually lack.

[1] `http://rtsys.informatik.uni-kiel.de/kieler`

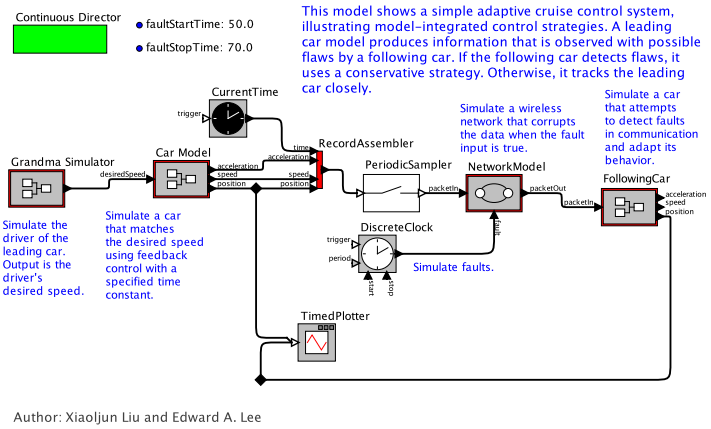

Figure 1. A small node-link diagram as laid out manually using the Ptolemy language.

To the best of our knowledge, our previous paper is still the only one on the subject of inferring comment attachments in visual languages [9]. Eichelberger recognizes that comments can relate to different elements (or none at all) in UML class diagrams [2]. Other work based on textual languages, for example by Buse and Weimer on automatically augmenting Javadoc comments [1], also requires knowledge about relations between comments and code. However, the attachment rules for documentation in textual languages are usually clearly defined, not as ambiguous as in visual languages. With comment attachment, applications such as automatic handling of documentation may become viable for visual languages as well.

Outline. In the next section, we will introduce and discuss our comment attachment heuristics. We will then evaluate them and discuss the results in Section 3, trying to derive suggestions for developers of visual languages. We conclude the paper in Section 4 with open topics for future research.

2 Heuristics

We can distinguish two categories of comments: *node comments* refer to a specific node while *non-node comments* do not. Non-node comments can be further divided: *title comments* contain the title of a diagram, *author comments* contain the names of a diagram's authors, and *general comments* contain general information about a diagram not specific to any one node. The goal of any *automatic attachment algorithm* is to attach every node comment to the node it refers to while leaving non-node comments unattached.

In the following, we will introduce the basic idea of each heuristic. There are two kinds of heuristics: *filters* aim to recognize different types of non-node comments to prevent them from being attached to anything, and *regular heuristics* try to recognize node comments and attach them to the node they refer to.

Detailed evaluations of how well each heuristic describes the usage of comments in out main data set can be found in the accompanying technical report [7].

Font Size Filter Text documents usually start with a title set in a larger font size than the rest of the text. One may well hypothesize title comments in diagrams to be set in a larger font size as well. The filter thus finds the set of comments with the largest font size. If the set only contains a single comment, it selects that as the title comment and thus filters it out, provided that it is not the only comment on the uppermost hierarchy level and that its font size exceeds the default font size.

Text Prefix Filter Visual languages usually do not support special comment types for non-node comments, but it seems reasonable to hypothesize that they will often start with similar phrases. In our data set, these are phrases such as "This model" (general comments) and "Author:" (author comments). The text prefix filter thus filters out comments that start with such prefixes.

Area Filter It seems reasonable to assume that general descriptions of what a program does will often be longer—and therefore larger—than more specific comments. The area filter thus filters out a comment if its area exceeds a certain threshold. We were, however, unable to find a good threshold value above which all comments can be considered non-node comments.

Node References Heuristic If the name of a node appears in a comment, we consider this to be a *node reference*. If a comment contains such a node reference, it seems sensible to assume that it should be attached to that node. This ceases to be true once further references occur in the comment: since our use case only allows comments to be attached to a single node, we consider such comments to be general comments. If a node name appears exactly as is in the text of a comment, the heuristic attaches the two unless the comment contains the names of other nodes as well.

Distance Heuristic The distance between a comment and a node may be the most obvious heuristic and was already examined in our first paper on the subject [9]. The hypothesis here is that the node a comment refers to is the one closest to the comment. The heuristic thus finds the node closest to a given comment and attaches the two unless their distance exceeds a predefined threshold.

Alignment Heuristic In graphic design, alignment between elements is used as a means to establish a relationship between them. It seems reasonable to assume that comments are aligned to the node they should be attached to. For a given comment, the heuristic thus finds the node best aligned to it, possibly restricted to nodes within a certain maximum distance, and attaches the two unless the alignment exceeds a predefined threshold. As it turns out, though, alignment is actually not a very good predictor for attachments.

2.1 Discussion

Based on analyses we performed, it seems that established conventions such as a big font size or how the list of authors is to be included in a diagram work best

for comment attachment. Other heuristics work to an extent (node references, distance) or have little predictive value (area, alignment). A considerable share of the information that helps link comments to nodes still seems to be in a comment's text, which is a lot harder to analyze.

There are two limiting factors to this analysis. First, the data set is smaller than we would like it to be. The number of diagrams is comparatively low (348), as is the number of authors that produced the diagrams (40). Also, the number of comments actually attached in our manual attachment (182 out of 1078, 17%) is not that high. The second and more severe problem is that all diagrams were created as demonstration models for the Ptolemy tool to help explain how certain actors or models of computations are used and how to develop using Ptolemy; the heuristics that work well for this particular set of diagrams are not guaranteed to work well for another set. In fact, from looking at diagrams produced with other languages and by other developers it seems that we may not find a universally applicable set of rules for comment attachment. We feel confident, however, that our heuristics are a good starting point to analyse the usage of comments in visual languages.

3 Evaluation

To evaluate our heuristics, we compared the automatic attachments computed for our data set against a manual attachment. That attachment was produced by inspecting each comment and determining which node, if any, it refers to. If that was not clear, the model was removed from our data set.

During the evaluation, we look at which node each comment is attached to in both attachments. There are four cases:

Correct A comment has the same attachment in both attachments.
Changed A comment is attached to different nodes.
Lost A comment is attached to a node in the manual attachment, but is not attached to anything in the automatic attachment.
Spurious A comment is not attached to any node in the manual attachment, but is attached to a node in the automatic attachment.

Attaching comments based only on the distance heuristic (see Figure 2a) yields results similar to previous results [9]: as the distance threshold is increased, the overall error rate increases mainly because the number of spurious attachments increases. The number of lost comments decreases as more comments are attached to nodes.

To reduce the number of spurious attachments, Figure 2b shows the results of applying the two most unproblematic filters (the font size and text prefix filters) as well as the node reference and distance heuristics: if the node reference heuristic finds an attachment, that attachment is applied; otherwise, the distance heuristic is invoked. This significantly reduces the number of spurious attachments as the distance threshold is increased. More importantly, however, the node reference heuristic causes fewer lost attachments, at the expense of

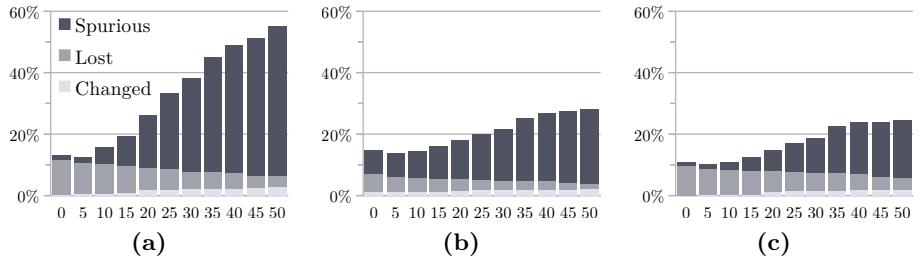

Figure 2. Results of comparing automatic attachments to the manual attachment, all involving the distance heuristic subject to different threshold values. (a) Distance heuristic only [9]. (b) Font size filter, text prefix filter, node reference heuristic, and distance heuristic. (c) As (b), but with a maximum distance threshold of 30 imposed on the node reference heuristic and with the area filter with a conservative setting.

more spurious attachments in the lower threshold areas. We think that this is a worthwhile tradeoff, since a node attached to a comment by the node reference heuristic is at least mentioned in the comment.

In an attempt to further reduce the number of spurious attachments caused by the node reference heuristic, Figure 2c shows the results of applying a distance threshold of 30 to the node reference heuristic, and of engaging the area filter with a very conservative setting. This decreases the amount of spurious attachments and the error rate overall to about 10% at best, at the expense of fewer found correct attachments as the number of lost attachments increases.

3.1 Discussion

The effectiveness of comment attachment largely depends on a good configuration of the heuristics. These results suggest that comment attachment should be replaced by proper support for explicit attachments in visual languages. However, comment attachment stays relevant for browsing scenarios similar to our use case, for languages that do not provide explicit attachments, or when users do not make use of them.

The latter problem seems most relevant to integrating comments into visual languages. A lack of proper attachments can prevent tool developers from making more advanced features available, such as good automatic layout, semantic reasoning, or even generating documentation. The best solution may be twofold. First, provide different kinds of comments, such as general comments, author comments, and node comments. Dragging a node comment onto the drawing area could then include displaying "attachment lines" that indicate which node the tool will interpret the comment to refer to, thus forcing explicit attachments.

The addition of such features to existing tools offers another area of application for comment attachment. Opening diagrams that do not make use of explicit attachments would trigger comment attachment and present the user with an automatically inferred attachment that they can then modify.

4 Conclusion

Building on our previous work on the subject, we introduced more heuristics to describe comment usage that varied in how well they perform. We used those heuristics to improve upon previous attachment results, mainly by keeping more non-node comments from being attached. Use cases include automatic layout as well as semantic reasoning about programs written in visual languages, as has already been explored in the context of textual languages.

Regarding future work, it first seems necessary to analyse the usage of comments in more visual languages and to compare the results. It also seems worthwhile to survey users of visual languages as to whether they use any deliberate conventions when writing and placing comments. Second, the attachment framework as well as the heuristics will have to be extended to support comments attached to diagram elements other than nodes. And third, comments sometimes describe a whole group of nodes. It seems extremely hard to infer such group attachments, but this intuition needs confirmation.

References

1. R. P. Buse and W. R. Weimer. Automatic documentation inference for exceptions. In *Proceedings of the 2008 International Symposium on Software Testing and Analysis*, ISSTA '08, pages 273–282. ACM, 2008.
2. H. Eichelberger. *Aesthetics and Automatic Layout of UML Class Diagrams*. PhD thesis, Bayerische Julius-Maximilians-Universität Würzburg, 2005.
3. D. Kramer. API documentation from source code comments: A case study of Javadoc. In *Proceedings of the 17th Annual International Conference on Computer Documentation*, SIGDOC '99, pages 147–153. ACM, 1999.
4. K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages & Computing*, 6(2):183–210, June 1995.
5. M. Petre. Cognitive dimensions beyond the notation. *Journal of Visual Languages & Computing*, 17(4):292 – 301, 2006.
6. C. Ptolemaeus, editor. *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org, 2014.
7. C. D. Schulze, C. Plöger, and R. von Hanxleden. On comments in visual languages. Technical Report 1602, Christian-Albrechts-Universität zu Kiel, Department of Computer Science, Apr. 2016. ISSN 2192-6247.
8. C. D. Schulze, M. Spönemann, and R. von Hanxleden. Drawing layered graphs with port constraints. *Journal of Visual Languages and Computing, Special Issue on Diagram Aesthetics and Layout*, 25(2):89–106, 2014.
9. C. D. Schulze and R. von Hanxleden. Automatic layout in the face of unattached comments. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'14)*, Melbourne, Australia, July 2014.
10. K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109–125, Feb. 1981.