Automatic Layout

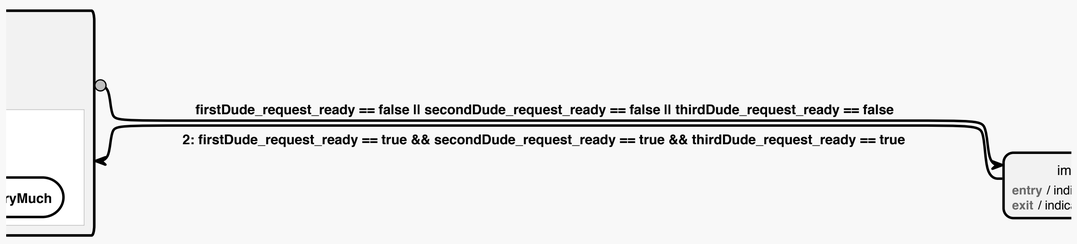
CAU
Kiel University
Faculty of Engineering
Department of Computer Science

An Opportunity

Label Management

The Problem

Visual programming languages usually cannot get away without textual labels. Depending on the visual language, these labels can grow rather large, as in this example taken from an SCChart:



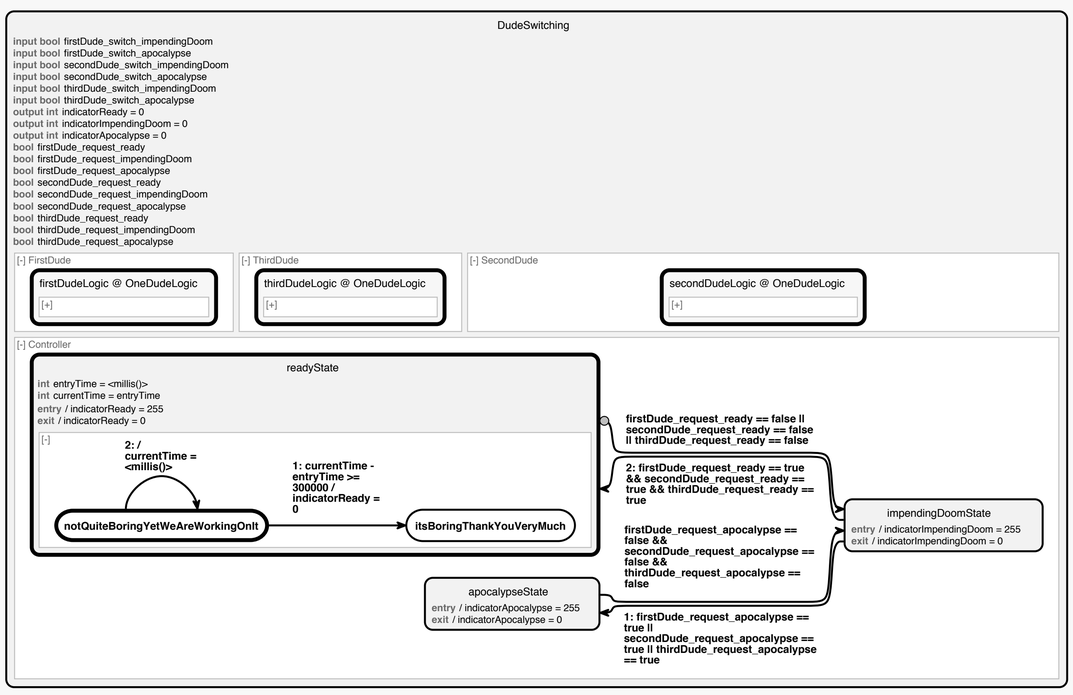
This in turn causes the diagrams to grow very large. Today's modeling tools usually only offer zooming as a solution.

Existing Solutions

Zooming basically gives users the choice between two different ways to try and cope with large diagrams. First, they can zoom into the diagram until they can read everything, as in the example above. This can make it hard to keep the context in mind. Second, they can lower the zoom level until the diagram fits the screen completely. However, long labels usually cause awkwardly wide aspect ratios, requiring the zoom level to be reduced below the point of legibility. Other solutions, such as fisheye lenses [1, 2], can result in distortion or text too small to be read.

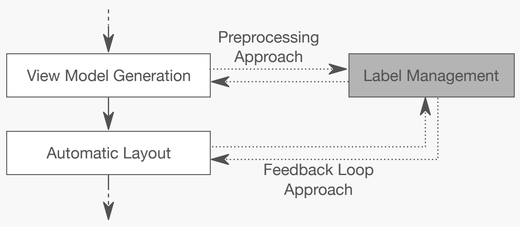
Label Management

Label management attacks the problem from a model-view-controller perspective: the text the view displays for a label can differ from the label's original text, thereby changing the amount of space required by the label. Exploiting this to change the size of the whole diagram requires to change its layout. Having to do this manually is tedious and time-consuming, but that changes when using automatic layout. This scenario is thus an opportunity of using automatic layout algorithms.



There are different ways to change a label's text, as there are different ways of integrating label management into the view generation process.

View Generation



Label management can be integrated into the view generation process in two ways. The *preprocessing approach* invokes label management when generating the view model. Label management decisions can be based on options set by the user or the mode the tool is currently operating in. In contrast, the *feedback loop* approach sets up label management when generating the view model, but delegates its invocation to the layout algorithm. This opens up the additional opportunity to shorten only those labels that actually cause the diagram to grow large while leaving other labels unchanged.

Shortening Strategies

Original Label Text
```
(not SignalA) xor (not SignalB) / SignalC(counter)
```
Syntactical Abbreviation
```
(not SignalA) xor (...
```
Semantical Abbreviation
```
SignalA, SignalB / SignalC
```
Syntactical Hard Wrapping
```
(not SignalA) xor (not
SignalB) / SignalC(cou
nter)
```
Syntactical Soft Wrapping
```
(not SignalA) xor (not
SignalB) / SignalC(
counter)
```
Semantical Wrapping
```
(not SignalA) xor
(not SignalB) /
SignalC(counter)
```
Target Shortening Width

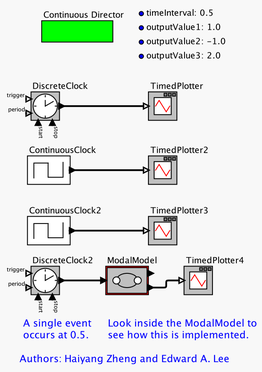
A Challenge

Comment Attachment

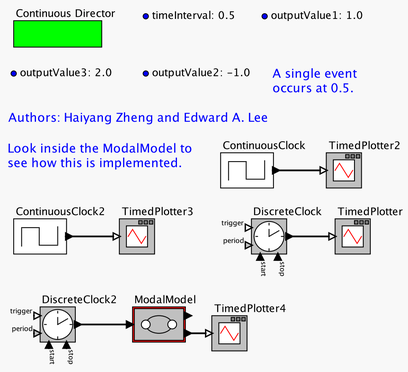
The Problem

Seemingly small problems can keep users from employing automatic layout. These are challenges that must be met for automatic layout to be successful. Many such challenges seem to be related to secondary notation, of which the placement of comments is one example. In manual layouts, the relation between comments and the diagram elements they refer to usually becomes clear through placement. Many automatic layout algorithms do not preserve the relative placement between diagram elements and thus mess up these relations, as in the following example from UC Berkeley's Ptolemy tool:.

Manual Placement



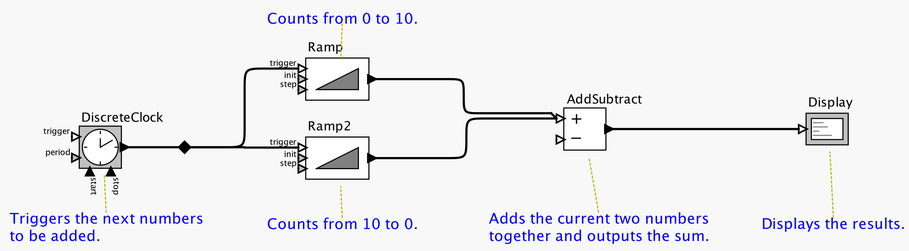
Automatic Layout



Existing Solutions

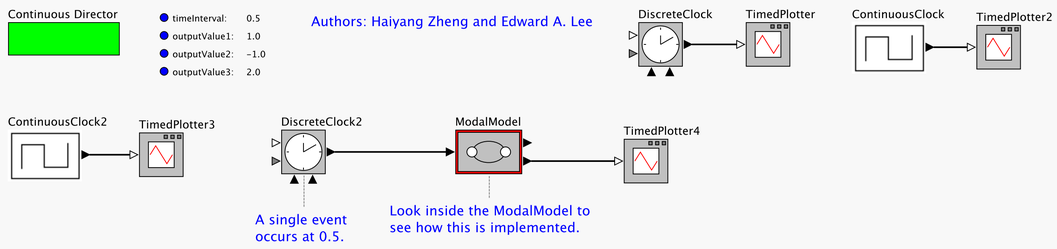
One solution to this problem is to only use *layout adjustment algorithms* [3]. As opposed to *layout creation algorithms*, layout adjustment algorithms do not compute a new layout from scratch, but base their placement decisions on the diagram's original layout. There may, however, not be a layout adjustment algorithm available for the given scenario. Worse, there may not even be an original layout in the first place.

Some visual languages allow users to specify explicitly which element a comment refers to [4]. This is usually indicated by a line connecting the two, as in the Ptolemy diagram below. However, neither do all languages support this feature, nor would all developers use it.



Comment Attachment

Comment attachment infers the implicit relations between comments and the diagram elements they refer to and makes them explicit. With the relations now available, layout creation algorithms can place comments and their diagram elements in close proximity:



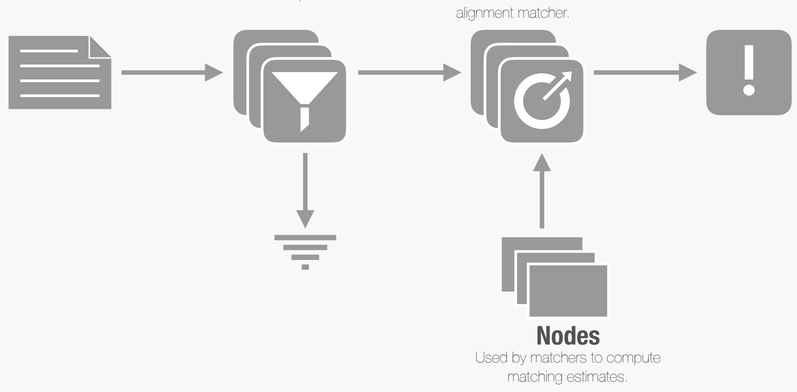
To infer attachments, comments are put through a pipeline that outputs an estimation of how likely it is for a comment to refer to different diagram elements. This result is then used to decide which element, if any, the comment will be attached to.

Comment
Enters the pipeline.

Filters
Discard comments that do not relate to specific elements. Available filters are the font size filter and the text prefix filter.

Matchers
Estimate how likely it is that a comment refers to a given node. Available matchers are the text reference, the distance, and the alignment matcher.

Decider
Decides which element, if any, to attach a comment to based on the matching results.



Nodes
Used by matchers to compute matching estimates.

Contact Person
Christoph Daniel Schulze
Department of Computer Science, Kiel University
Olshausenstr. 40, 24098 Kiel, Germany
Phone: +49 (0) 431 880-7297
Fax: +49 (0) 431 880-7615
cds@informatik.uni-kiel.de
http://www.informatik.uni-kiel.de/rtsys

Literature

[1] Y. K. Leung, M. D. Apperley (1994).
A review and taxonomy of distortion-oriented presentation techniques.
ACM Transactions on Computer-Human Interaction, 1(2), 126–160.
[2] M. Sarkar, M. H. Brown (1992).
Graphical fisheye views of graphs.
Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 83–91.
[3] K. Misue, P. Eades, W. Lai, K. Sugiyama (1995).
Layout adjustment and the mental map.
Journal of Visual Languages & Computing, 6(2), 183–210.
[4] H. Eichelberger (2005).
Aesthetics and Automatic Layout of UML Class Diagrams.
PhD thesis, Bayerische Julius-Maximilians-Universität Würzburg.

Project Information

ELK ECLIPSE LAYOUT KERNEL
https://eclipse.org/elk

Poster presented at the ACM Student Research Competition at the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS) 2016,