Xcos Automatic Layout

References

Name: Chenfeng Zhu
Mentor: Mr. David Clément
Mentor: Mr. Paul Bignier

1 Mathlab

1.1 Automatic Layouting for Simulink

(http://de.mathworks.com/videos/improving-modeling-usability-automatic-layouting-for-simulink-93139.html)
Rearranging Blocks and Rerouting Lines are the main problems.

1.1.1 Method and Implementation

Automatic Layout Generation using ***an Improved Sugiyama Algorithm***
4-Step algorithm for layered graphs:

1) Feedback handling with user preferences
2) Hierarchisation with optional alignment
3) Barycenter crossing reduction with support for port constraints
4) Coordinate assignment with support for variable sized blocks and user preferences:
   - uses (integer) linear constraint solving to optimize block positions and sizes towards optimal edge straightness and minimal model size.
   - builds on a scaled, linearized block size / port position model.

Interfacing with Simulink: A Layout Server

1) External Layout Server
   Contains layout algorightms
   Processes layout requests
2) Simulink M Scripts
   Generate Layout Requests
   Decode Layout Results
   Update model
3) Layout Requests / Results
   XML encoded

Transferred via TCP socket
Contain old / new layout information

1.1.2 Use Cases and Usability Enhancements

1.1.2.1 Modeling Challenges - Building new Functionality

1) New Functionality is created within a subsystem:
   Blocks are added from the libraries.
   Connections are drawn.
2) Regular Layout Adjustments are required to establish and preserve the models descriptive quality.
   Block positions and sizes are adjusted
   Lines are rerouted
3) Rudimentary support is provided by recent improvements
   Block alignment and distribution

1.1.2.2 Modeling Challenges - Inserting Ports at BusCreator Blocks

1) A new Signal needs to be added to a Bus:
   Additional status information
   Functionality is to be added
2) Signals in buses are often grouped by their position in the bus:
   For flat hierarchies
   Relating signals are placed adjacent
3) Inserting signals is not supported natively:
   Adding at the last position is supported
   Moving the new port to the desired position is time-consuming
   Layout needs adjustments afterwards

1.1.2.3 Modeling Challenges - Breaking-up Subsystems

1) A models subsystem structure needs to be overhauled:
   Functional optimization
   Model refactoring
2) Existing subsystems need to be broken-up:
   Subsystem creation is supported with limited layout capabilities
   Breaking-up a subsystem requires manual cut, paste, connect, delete operations
3) A new layout needs to be generated afterwards.

1.1.3 Technical Issues and Solutions

1.1.3.1 Dynamic Context Menu

Restriction

The Simulink customization API for context menu extensions does not support arguments for action or container functions:

Requirement

Additional arguments are required for some functions, e.g. "Insert new port at position n".

Solution

Dynamically generated anonymous wrapper functions hide arugments from the API:

1.1.3.2 Simulink Challenges:

Nonlinear block size / port position relationship

Distance between ports is linear dependent on the block size

First port offset is not linear dependent on the block size

Layout and structure of a mode are not strictly separated

Line split points are not interchangeable, as lines are represented as connected segments

Line drawing and block positioning can lead to accidental connections

Other Issues

Interface for port positions returns imprecise positions

1.1.4 Features

1) One-click to beautify the model diagrams.
2) When building new functionalities, auto-arrange the blocks and lines (Block positions and sizes are adjusted and lines are re-routed).
3) When inserting new ports, relating signals are placed adjacent.
4) When breaking up the subsystems, it needs manual operations.

1.2 Smart Signal Routing

(http://blogs.mathworks.com/seth/2012/10/11/smart-signal-routing/)

This article is about the improvement to signal routing in Simulink.

1.2.1 Line routing with an optimal path

Smart signal routing means that when drawing a signal line, Simulink will automatically find the "optimal path" so that the new signal line is as short as possible, has minimal 90 degree turns, and doesn't overlap other blocks and text. Not only that, as drawing the signal line, Simulink lets users know exactly what the path is going to look like before we release the mouse button.
(http://blogs.mathworks.com/images/seth/2012Q3/smart_signal_routing_1.gif)

1.2.1.1 Features:

1) Find the path which is as short as possible.
2) If the line is not vertical or horizontal, use 90 degree turns to find a "shortest" path.
3) The path will NOT overlap other cells.
4) There is a preview for drawing a new line.
5) The lines will change simultaneously when moving the blocks (vertices).

1.2.2 Drawing custom paths

There is a new way to create our own custom signal line path too. Here is how it works: Draw a signal line like we normally would. Then release the mouse button when we want to create a bend in the line. We will see three blue arrows appear: one to keep going straight and two to make a left or right turn. Choose the arrow that we want and repeat until we have created our own signal path.
(http://blogs.mathworks.com/images/seth/2012Q3/smart_signal_routing_2.gif)

1.2.2.1 Features:

1) Draw the line as what we want to bend the line.
2) There are arrows to choose the direction.

I think this function has already been implemented in Xcos, but there are a few differences from it (the arrows for the direction).

1.2.3 Minimum disturbance

It will not get blown away if we move some blocks around as we continue building up the model. The signal lines that already exist will be disturbed as little as possible as we move blocks around.
(http://blogs.mathworks.com/images/seth/2012Q3/smart_signal_routing_3.gif)

1.2.3.1Features:

1) The lines will change as little as possible when moving the blocks (vertices). This is under the consideration.

2 Microsoft

2.1 Microsoft Automatic Graph Layout

(http://research.microsoft.com/en-us/projects/msagl/)

(https://github.com/Microsoft/automatic-graph-layout)

3 Others

3.1 Major Layout Algorithms

(http://docs.yworks.com/yfiles/doc/developers-guide/major_layouters.html)

3.2 Extending the Sugiyama algorithm for drawing UML class diagrams: Towards automatic layout of object-oriented software diagrams

()

3.3 A topology-shape-metrics approach for the automatic layout of UML class diagrams

()

3.4 Graphviz - Graph Visualization Software

(http://graphviz.org/)

4 Unused

4.1 Refactoring of Simulink Diagrams via Composition of Transformation Steps

4.1.1 Keywords

Simulink, Refactoring, Transformation

4.1.2 Introduction

It is a modular technique for refactoring Simulink diagrams based on the composition of predefined transformation steps.

4.1.3 Simulink Meta-Model for Refactoring

The criteria for meta-model:

1) All necessary structural properties of diagrams that are required by refactorings should be captured.
2) Support for incomplete diagrams.
3) Layout information must be captured.
4) Establish a degree of granularity that enables local structural changes.

4.1.4 Transformation Steps and Their Composition

4.1.4.1 The effect that a transformation step should be

4.1.4.2 Composite of transformation steps

4.1.5 Specification of Refactorings

4.1.5.1 Replace Goto/From With Explicit Signals

4.1.5.2 Merge Subsystems

4.1.6 Implementation