Architecture design

File	Modified

No files shared here yet.

Drag and drop to upload or browse for files

Application Documentation

Most of the implementation of the editor is in javascript\examples\grapheditor\www\js folder. Other folders are the mxGraph API implementation.

You can see the API Specification by clicking here. This user manual may also be useful in understanding mxGraph. Here is an even more detailed manual if needed.

All documentation written are additional functions that we have implemented. Other functions can be referenced by looking at the documentation.

Actions.js

This allows for function calls from menus and toolbars. For example, we can call dialog boxes.

this.addAction(actionID, function, null, null, tooltip) may be helpful for adding shortcut actions.
actionID - name of the action (used to reference it)
function - function to call when executing action
Keep the two as null
tooltip - shows the tooltip (toString() method)

this.addAction('zoomIn', function(evt) { graph.zoomIn(); }, null, null, "zoom in");
This allows for zooming in and out for example

Dialog.js

Handles all the UI dialog boxes that appear when accessing menus or toolbar items.

EditDiagramDialog(editorUI)	Creates a dialog box that allows from inputlist input.

Graph.js

Controls the entirety of the graph and canvas. This includes everything about creating shapes, moving shapes, deleting shapes etc.

Helper.js

Most of the addtional functions implemented is here. This includes modifying the underlying XML to display a graph.

Function	
generateXMLAndRefreshCanvas(textareaString, editorUI)	Takes a string from the input list and refreshes the canvas
readInputList(editorUI)	Reads the string from the input list text area
updateNodeList(InputList,editorUi)	Reads XML from editor and converts into readable input list
rightShiftvertical(rootXML,childMap,node,shift)	Calculate the distance from the node of a sub goal vertial tree

GoThroughTree(rootXML,node,shiftx,shifty,childMap)	Shifting all the children under node by shiftx and shifty
shifting(rootXML,node,shiftx,shifty)	shifting node by shiftx and shifty
edgeshifting(rootXML,edge,shiftx,shifty)	shifting the turning point of the subgoal by shiftx and shifty
newRepositionTree(editorUi)	Repositions the tree so that non-functional goals and functional goals are not overlapping.
childinTotal(rootXML,childMap,node,height)	return the total height+shift before node in a subgoal tree
Addpoints(cellgeo,x,y)	add the turning point of each subgoal
String.prototype.width(font)	Takes a string and calculates the pixel size according to the font
isVertical(childrenArray,node,childMap,first)	checking if node is able to show its children in vertical subgoal tree
parallelogram(id, value, x, y, inputString,isvert)	Constucts an xml document for a parallelogram
heart(id, value, x, y, parent, inputString)	Constucts an xml document for a heart
actor(id, value, x, y, parent, inputString)	Constucts an xml document for an actor
spade(id, value, x, y, parent, inputString)	Constucts an xml document for a spade
cloud(id, value, x, y, parent, inputString)	Constucts an xml document for a cloud
arrow(id,fromid,toid,isvert)	Constructs an xml document for connecting two shapes with an arrow
actordot(id, x, y, parent, inputString)	Constucts an xml document for dot textbox
inializeXML(rootXML)	Generates the inital XML that mxGraph needs to display in the canvas
addChildren(rootXML, parentInput, childrenArray,childMap)	Takes a parallelogram and its children, then generates the XML document that contains the shapes and connections
findByID(rootXML, nodeID)	Finds the XML document using the ID
checkError(inputString)	Takes the input and returns a string of all the errors
hideOnTypeHelper(inputList, editorUi, types)	Set elements in inputList to invisible based on element type
showOnTypeHelper(inputList, editorUi, types)	Set elements in inputList to visible based on element type
showHideToggle(ui, types)	Toggle visibility of elements in inputList based on element type
textConverter(cellStyle, textValue)	Modify input text in canvas by remove and too many lines
canvasVisibilityHelper(editorUi, visible, ids)	Toggles visibility of elements in inputList based on element type
singleElementErrorCheck(text, inGraphEditing)	check if the input value from user is valid (can only contain alpha, numbers, space and/or newline)

Shapes.js

This is where we create new shapes. The way shapes are created is very convoluted. To get a grasp of how this is done, search up the function

UmlActorShape() in the file. The function creates a stick figure shape. Shapes can also be defined as XMLs
inthe \javascript\examples\grapheditor\www\stencils folder.

To add icons to XMLs here, use svg2xml which converts the more popular svg format to the xmGraph xml implementation.

Follow the UI prompts to import SVG and export XML.

Once XML is exported, copy to any shape.xml file in \javascript\examples\grapheditor\www\stencils.

Reference the shape-name when creating the shape (Look at Jgraph documentation).

Dependencies

Java JDK 8+

Install JDK

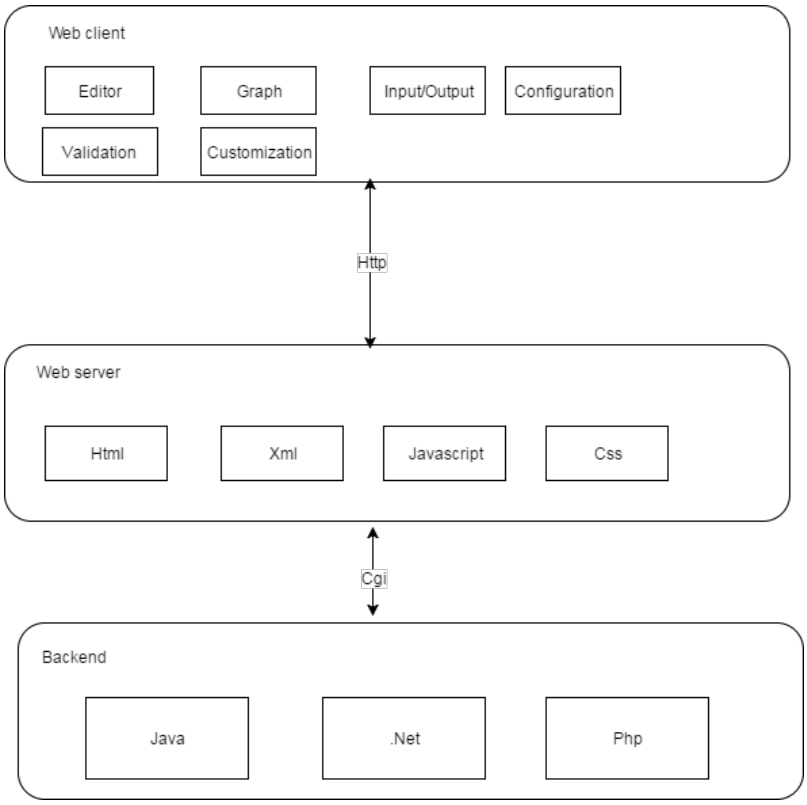
All systems must have JDK installed to build the application. Follow the **link.**

Sidebar.js

This handles the display of the left sidebar.

Sidebar.prototype.init()	initialise sidebar, called when page up.
Sidebar.prototype.showTooltip(elt, cells, w, h, title, showLabel)	Show the tooltip of current item
Sidebar.prototype.hideTooltip	Hides the current tooltip.
Sidebar.prototype.addCreatePalette(expand)	Adds the create palette to the sidebar.
reloadSidebar(editorUi)	Reaload Sidebar from XML, called when import xml
updateSidebarTree(parentStrings, childMap, editorUi)	rebuild the jstree in sidebar to display input list
dataValueAddHelper(dataValue, iconString, id, parentID, text)	help jstree add non-functional subgoals from input list
updateHelper(text, editorUi, refreshTree)	update text box and refresh canvas based on new input list
checkCountHelper(parentID, iconString, limit, dataValue)	check whether number of goal within limit
jstreeCreateHelper(inputText, parentID,iconString, typeSymbol, dataValue,editorUi)	help jstree create non-functional subgoals
jstreeHideHelper(currentID, typeSymbols, editorUi)	help jstree hide non-functional subgoals
jstreeShowHelper(currentID, typeSymbols, editorUi)	help jstree show non-functional subgoals
updateSidebarSingle(id, cellStyle,newText)	Updates a single node in jstree based on id and new text, called when editing in graph
Sidebar.prototype.createTitle(label)	Creates and returns the given title element.

Architecture overview

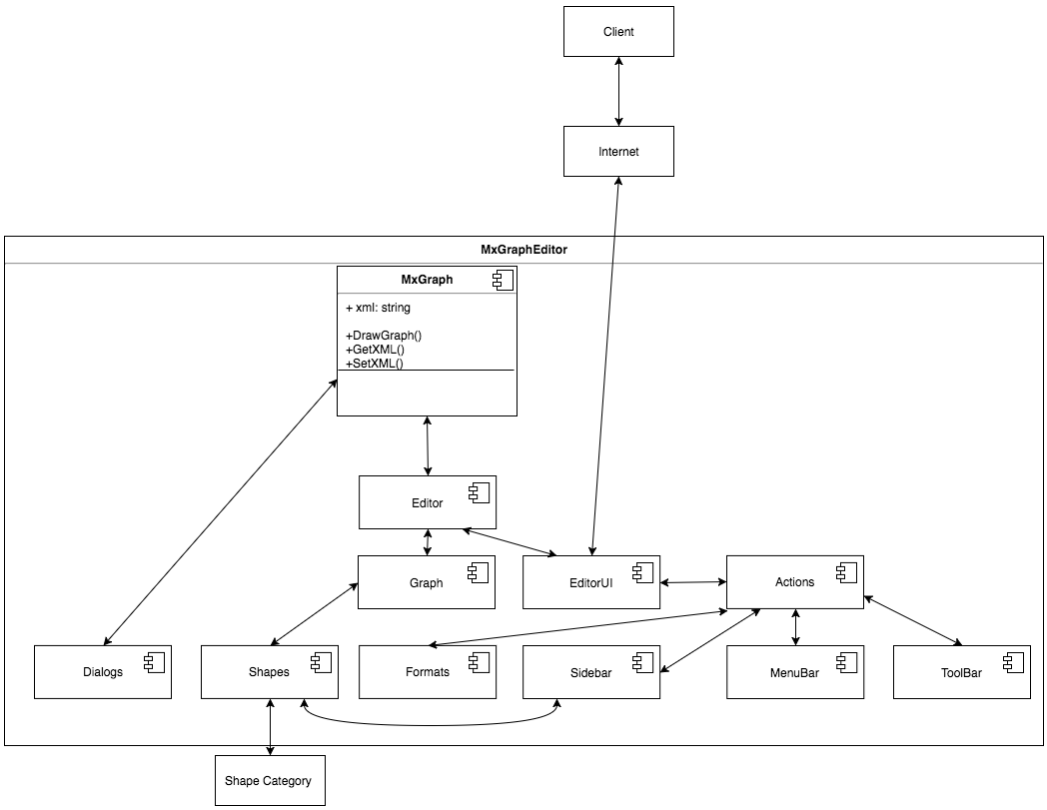


Java - server side

javascript - client functionality

Architecture overview.xml

Web Server Layer



the XML for the graph is here: Goal_Quoll_Archit.xml

Build / installation

The code can be compiled to run on multiple platforms including MACOSX, Linux Ubuntu and PCs with MS-Windows and JDK installed (see Apendix).

The mxGraph library requires Google Chrome 30 or later, Firefox 31 or later, Microsoft Internet Explorer 9.0 or later, Safari 6.2 or later, default Android browser in Android version 5.0 or later and default iOS browser in iOS 8.0 or later.

Installation

In Terminal

- Open terminal window
- cd into the directory containing the source files for the application
- cd into the 'java' directory which contains the build.xml file
- enter command in terminal window 'ant grapheditor'. (See Appendix for installing ant)
- Wait a few seconds.
- Open a browser and enter address http://localhost:8080/index.html
    - If running on a server ensure that port 8080 is open. You can then access the application on http://[SERVERIP]:8080/index.html
    - Note: If you want to remove 8080 (use the default html port), you must setup a proxy server. (Read appendix)

In Eclipse

- Open Eclipse (See Appendix for installing Eclipse)
- Right click the file com.mxgraph.examples.web.GraphEditor under the folder javascript/examples/grapheditor/java/src
- Click on 'run'
- Open a browser and enter the addresss http://localhost:8080/index.html

Appendix

Install Proxy Server

https://techteam.wordpress.com/2009/10/22/use-port-80-instead-of-port-8080-on-web-application/

Install JDK

All systems must have JDK installed to build the application. Follow the **link.**

Install Ant

Windows

https://www.mkyong.com/ant/how-to-install-apache-ant-on-windows/

MAC

https://www.mkyong.com/ant/how-to-apache-ant-on-mac-os-x/

Linux

http://dita-ot.sourceforge.net/doc/ot-userguide13/xhtml/installing/linux_installingant.html

Install Eclipse

Eclipse IDE for Java EE Developers is required follow this link

http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/heliossr2

Known Issues / Browser Compatability

- When non-functional goals get to +4, looks strange.
- Highlighting diagram, cannot unhighlight
- Limited edit functionalities on diagram
- Firefox: Right click sidebar tree & Input list line numbers not working