**SPECS:**

1. Type display   
     
   If a user presses the button on a CyberComposition model, we want to show the data type of each signal and parameter port below the current point in the design. So - collect the ports and traverse all subsystems, collecting ports. Once you have the whole set, put them in a table with the port path (from the starting point) in the left-hand column, and the data type in the right-hand column.   
     
   For ports at the top level, you can get the data types by reading the class attributes of all the port objects.   
     
   For Simulink wrapper models, you can get the data types by navigating inside each port and looking to see whether it has a TypeRef object. If not, then it has no type. If so, then follow the reference to see what type it is.   
     
   For SignalFlow wrapper models, you can get the data types by checking an attribute on each port.   
     
   2. Compatibility   
     
   It would be great if you could check connections to see if the ports are compatible. We'll need to come up with a table that describes compatibility between Modelica class types and SignalFlow/Simulink types. If connected ports have incompatible types, then make their entries in the table have a different color so the user can quickly isolate problems.

**UI Ideas:**

1. A Vertical list with multiple rows. Each row containing a tuple of the Port Path( from RootFolder) and its type. In case of mismatch of 2 port types color the corresponding tuples with a specific color. For every pair of mismatched ports the color to represent the mismatch will be different.
   1. Pros-
      1. A linear list of the ports with their types adjacent to them in their corresponding tuples.
      2. Convenience in modifying the port type or name using the UI.
      3. Convenient to locate a port using a Search Feature in the UI.
   2. Cons-
      1. The 2 ports within a pair of mismatched ports may be located at a huge gap in the presented vertical list of ports. Tracing the connected port from either of them becomes tedious.
      2. For multiple pairs of mismatched ports there should be different colors. Similarly toned different colors will be a mess to the user’s sight.
      3. Multiple colored pairs of mismatched ports will be hard to locate among so many colored entries.
      4. There is no way to mention the connection between two ports in the UI. In case the user wants to delete or add a connection using the tool’s UI then he can NOT. He needs to go to the GME model and create or delete connections between ports. SOLUTION: Add the connected ports and the connection types for a port inside a tuple in the UI list. But then a tuple becomes too big for display and there is a lot of cluttering.
2. A graph with the ports as nodes and connections between them as graph edges. For mismatched ports the respective nodes and the links are colored ONLY IN RED.
   1. Pros-
      1. Graphical realization.
      2. No need to use confusing multiple colors as in the previous approach.
      3. Convenient to change port types.
      4. Convenient to change names of the ports.
      5. Convenient to add or delete connections.
   2. Cons-
      1. Cluttering in the graph in case of multiple ports might cause problem.
      2. If some user is fine with GUI then he can do modifications even in the GME GUI.
      3. May be more implementation time.
3. **Adjacency Matrix for the Ports. An existing connection between two ports is represented by blackened spot in the matrix. A red spot for the connection between two mismatched ports. [CHOSEN]**
   1. **Pros-**
      1. **Easy location of mismatched port types by looking the red spots in the matrix and corresponding ports in the row and column headings.**
      2. **Types of ports can be manipulated in the row or column headings.**
      3. **Easy creation or deletion of the connections between ports by locating the correct row and column headings (port names).**
   2. **Cons-**
      1. **For large number of ports lot of scrolling might be needed within the UI.**

**REQUIRED WORKFLOW for TOOL:**

* Start from the current MODELICA COMPONENT and traverse down its contained subsytems/ SignalFlows/ StateMachines etc.
* While traversing down the hierarchy, add a new heading in UI for every unique port found. Also fille the matrix cell for the connection accordingly in the UI.
* Show name and type of port over the heading block in a tag.
* Show the META type of the connection over the matrix cell in a tag.
* Push the matrix cell button to highlight the ROW and Column Heading with dashed lines originating at the matrix cell.
* Re-Push the matrix cell button and the dashed lines disappear and the boldness of the headings also disappear.
* Within the tool search for a port. Locate the port in the matrix on the UI. – **NOT NEEDED BECAUSE:**
  + **IRRELEVANT BECAUSE –**
  + **COLOR COMBINATIONS IN THE GRID WILL SHOW THE FAULTY PORTS AND LINKS**
  + **FREEZE GRID IS USED.**
* Modify the type or name of a port. Check the compatibility with the connected port. If compatible allow the modification and save the changes in the GME model.
* Create/delete a connection between two existing ports. Check port compatibility for this connection. If correct then save the changes in the GME model.

**OOPS DESIGN for ADJACENCY MATRIX:**

::Required Objects::

* **UI Window:**
  + **UIComponents**: Having certain actions associated with them that are presented in a menu
    - **Scrollable Components**: having 2D coordinates associated with them, inside a scroll window
      * **Matrix Headings:** Row Headings(Src Ports), Column Headings(Dst Ports)
        + **Port Name:**
        + **Port Type:**
      * **Matrix Cells:**  Representing connections between ports in rows and columns
    - **Unscrollable Components:**
      * **Message Console Strip:** To print the messages after some computation for the user
      * **Text Edit Console Strip**: To change the text for port names, types etc.
      * **Port Search Box:** To find the port/ports in the UI Window

::Object Properties::

**UIComponents:**

Has a menu associated with them which appears on the right click. A pure Virtual method to be implemented by inheriting classes as per their requirement.

Have a Tag associated with them. Tag appears whenever the mouse is over a component for more than a second.

Can have blinking borders to attract user attention. The borders blink only for a small while. Maybe a second.

**Text Edit Console Strip**:

Contains Text Field and the DONE button.

Available for use only when Edit action in the menu for any UI Component is invoked.

Till the editing is not complete the User can not have access to any of the other parts of the UI.

When user presses DONE, the corresponding scrollable component's respective contents must be changed.

When Edit action is invoked on a component the Strip must blink.

When Edit action is invoked on a component the Strip must be filled with the current text content of the UI Component

**Scrollable Components:**

Have X and Y coordinates recorded.

**Matrix Headings:**

Static Size of Heading Block Area.

Information for Ports.

Tags appearing when Pointers over.

Tags have Name and type separated by colon.

Hiding Small window with Port Name and Port Types. Appears on left click. No X or Y coordinates recorded for the hidden window.

Action Menu has Edit Port Name and Edit Type. These edits should reflect in the GME model also. The GME model modification related information should be printed in the Message Console of tool's UI

The name on the heading is the abbreviated name for the port constructed by only couple of initial letters.

**Matrix Cells:**

Have associated color with them - balck, white, red.

Black or Red colored cell has Tag has the current MetaConnection Type as its text.

White colored cell has Tag with empty text.

Right Click menu option is to delete connection when RED or BLACK. Or to Create a connection when WHITE.

Upon Creation of a new connection the ports are checked for types. If connection is possible then only change the color to black otherwise print Error/Warning on the Message Console. If connection is possible then only update GME model.

Upon deletion Update the GME model and print Success log message in the message console.

Right Click Menu Option is to Locate the Row Heading or Column Heading.

**Port Name & Port Type:**

Text based info for the ports with complete text.

Only right click menu action provided is to edit the text.

**Port Search Box:**

Text Field for entering the port name.

Search Button.

Locate the searched Matrix Heading in the scrollable UI using x,y coordinates. Locate either in the ROW Headings or the Column Headings.

**Message Console Strip:**

Prints a single line of log at a time.

-----------------

**Key:**

Top priority: no editing features, just display the flattened port graph as a matrix.

Medium priority: search and some helpful display features

Lower priority: editing

------------------

**::QUESTIONS::**

1. Can there be multiple source or destination connections between two ports?

Multiple source connections are possible, but not multiple destination connections.

1. Do we want to allow the user to in anyway add new ports using the UI tool?

No.

1. Only the Matrix headings need to be searched for ports using the Port Search Box. Do we still need a 2D coordinate system? We can probably avoid recording the UI location of the Matrix Cells?

Yes.

1. Should the message console only have a single line? or Multiple Lines?

Single line should work.