Front-end component for workflow system

Introduction

We are seeking a component to plug into a system currently under development. The system that is being built is using Vue JS, however this component is not REQUIRED to be written in Vue but would need to be compatible with it. The CSS framework being used is Tailwind CSS, again this component doesn't required the use of this but would be advantageous. This component will have the following functions:

(a) To draw a workflow chart in accordance with the specifications outlined in this document based on a set of interconnections provided by the back end
(b) To allow the addition of new "steps" or "branches" by dragging them from a palette onto the workflow
(c) To be able to pan and zoom around the work flow chart.

This document outlines a view as to how this work may be completed. If you have a view as to how this can be done better, please communicate that.

Drawing the workflow chart

The elements

There are four elements to draw: Start, Stop, the "step" and the "branch" as shown in Figures 1 and 2 below. The "step" is a block rectangle of width B and height H1. It has the following features

(i) a coloured strip to the left-hand side
(ii) an icon or image
(iii) a name/title
(iv) a short description

The back end will pass

(a) The type of the bloc, be it a step or a branch, or other
(b) The colour to use for the coloured strip,
(c) The icon, (TBD if passing an address for a remote image or a component)
(d) The name/title, including properties such as the font and size
(e) The description, including properties such as the font and size
(f) The border colour, thickness, highlight colour, highlight thickness
(g) A flag that indicates whether the line from the bottom of the block should be drawn (indicating the end of the flowchart)
(h) The hierarchical position/address of the block in the form number,number….number e.g. "-4,1,1,2"
(i) (optional) additional description, including properties such as the font and size

The “branch” element has the same dimensions as the “step”, but with curved ends and always has the static contents (not dynamic)

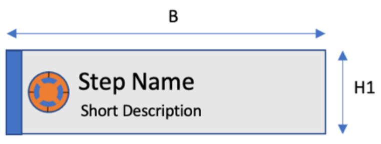


Figure 2: Step

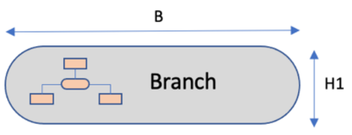


Figure 1: Branch

Under certain circumstances we may choose to add an “additional information” section to the step. In this configuration the step will be shown with a different height (as shown in Figure 3). In this case, where a branch exists on the same grid line as this extended height step, the branch will still be displayed in the same grid but will be vertically aligned with the step (as in Figure 4).

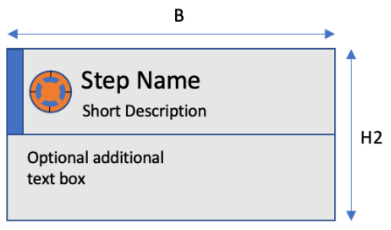


Figure 3: Step with height H2

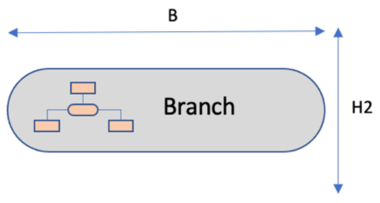


Figure 4: Branch vertically aligned when height is H2

In the flowchart layout there is one final element: the “plus” (as shown in Figure 5). This is drawn on a flowchart in a location where a “step” or a “branch” can be added/dragged on to.

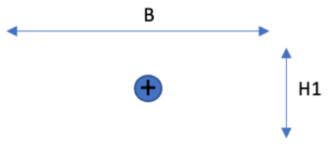


Figure 5: The "plus"

The layout

The layout is defined by a grid of Blocks as shown in Figure 6. The dimension B and H are defined by the back end. Where the back end requires the increased height of H2, H will equal H2. Otherwise H will equal H1. An indexing for locations on the grid is also shown below.

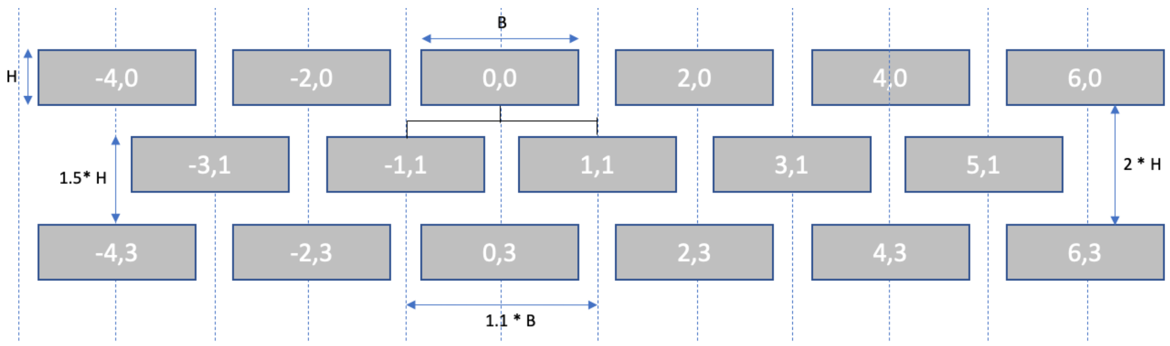


Figure 6: The layout grid

Vertical lines defining the interconnections between “steps” and “branches” will lie along the dotted lines shown in Figure 6. Horizontal lines will be centered at 0.25 * H above and below the row of cells. Other than the lines, all “steps”, “branches” and “pluses” are placed on the grid. Figure 7 displays how “pluses” are used. They represent a location where a new “step” or “branch” can be dragged on to or clicked to add a new item (done by dispatching a custom HTML event).

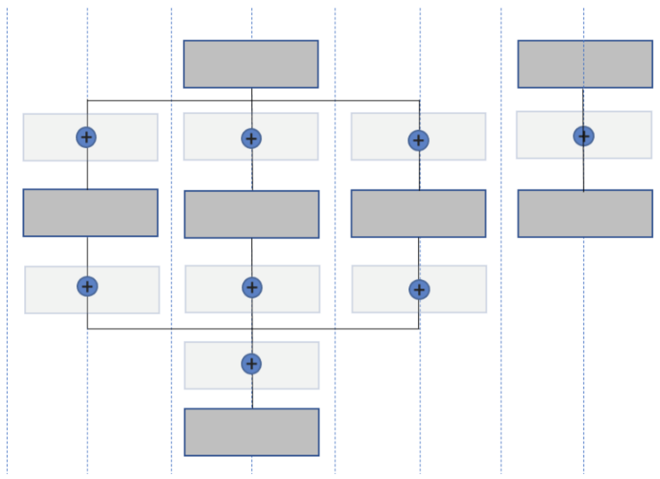


Figure 7: Displaying the use of the "plus"

In Figure 7, we see two alternate paths (just for illustration the final product will only have 1 path per diagram). On the right, there is a simple “step” leading to another “step”. A “plus” is placed in the middle, allowing an additional “step” to be added between the existing ones. On the left there is a branch leading to three distinct paths (note the difference from the example in Figure 6 where the branch leads to *two* paths). There are “plusses” added in all the locations where a user may wish to drag a “step” or a “branch”. Note that the gray boxes around the plusses are not drawn in practice and are only shown for positional clarity, though we may in the future decide to have a greyed out box or bordered box when hovering over a “plus” area to indicate when a new item can be placed.

As soon as a “step” or “branch” has been dragged onto a plus and released, the entire workflow chart is redrawn. See below for a more detailed description as to how this works.

We don’t expect this component to actually calculate the required changes when adding/moving elements around, merely to pass the information required to an “external” (at least to this component) entity to process this change and update the data, and for this component to react to the change or be told about the change and redraw itself.

The initial position

Any workflow starts with a position as shown in Figure 8. It has a “start”, an “end” and a “plus” that allows other types to be added.

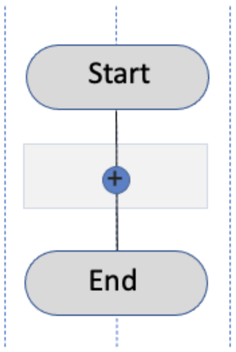


Figure 8: Flowchart initial position of start and end.

Workflows are built then by adding “steps” and “branch” elements that are dragged onto the “plus” and any further “plusses” that are drawn.

Changes when a “step” is added

Figure 9 shows the change in display when a “step” is dragged onto a “plus”. The entire chart is redrawn on the release of the button.

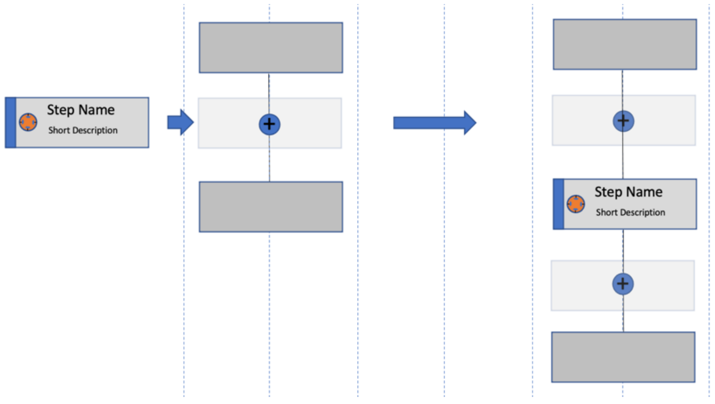


Figure 9: Addition of a "step"

Changes when a “branch is added”

Figure 10 shows the change in display when a “step” is dragged onto a “plus”. The entire chart is redrawn on the release of the button. Note that the gray boxes around the plusses are not drawn in practice and are only shown for positional clarity. But as stated before we may want to add this feature in the future.

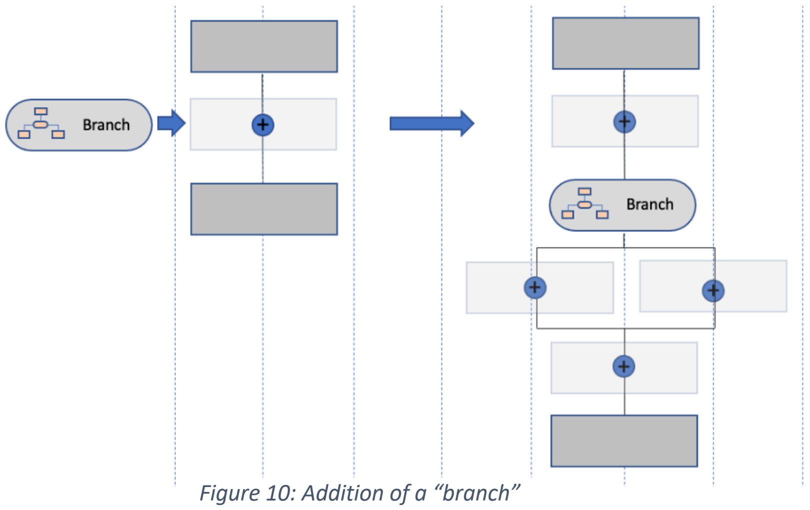


Figure 10: Addition of a “branch”

Rules for more complex workflows

In the back end, workflows are described in terms of a parent-child relationship. “Steps” may only have one child. “Branches” may have *n* children, where *n > 1*. Branch details are provided as an ordered set of children. The must be displayed in order from left to right. More than one parent is only possible as part of the branching

The workflow should be laid out on the grid, as shown in the figures above, making space either to the right or left as is required to display the necessary Blocks. As additional Blocks are added, they may be added on a central branch, rather than an end one. In this scenario, all the Blocks either to the left (if the branch being expanded is to the left of centre) or the right (if the branch is to the right of centre) must be moved to make room.

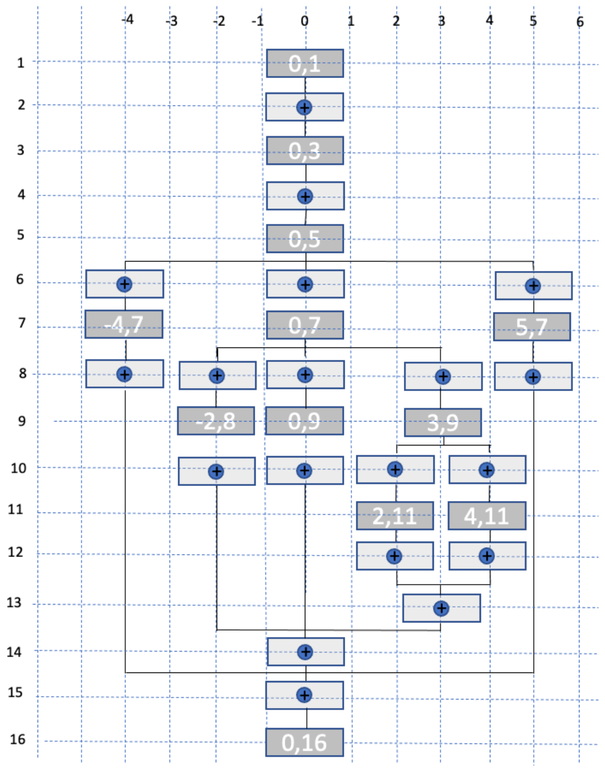


Figure 11: A more complex chart in Edit Mode

Figure 11 shows a more complex chart. You will note that “plusses” are placed in the grid block directly beneath the “step” and that additional white space is always after the “plus”.

Selecting a “step” or “branch”

When a user clicks on a step or branch, the border of the Block changes colour and thickness according to the specifications as passed from the back end.

When selected the front end should dispatch a custom html event to indicate which Block has been selected.

Edit and Display Mode

Until now, all descriptions have been made of Edit Mode. In "edit" mode, new "steps" or "branches" may be added through the GUI.

In Display Mode, there is no ability to add new steps and the blocks containing "plusses" are *not* displayed. The standard pan and zoom functions still operate in display mode. Blocks may still be selected and data in them edited. Redraws may be triggered from the back end.

Figure 12 shows the same flowchart as Figure 11, but in display mode. Note that the absence of the need for "plusses" means that the chart is much more compressed, both horizontally and vertically. The only significant change to the drawing rules is that in Display mode, the horizontal that join the elements of a branch at the bottom may be placed at the normal 50% point of the grid, but also at the 0% point of the grid. This is seen below at row 7.

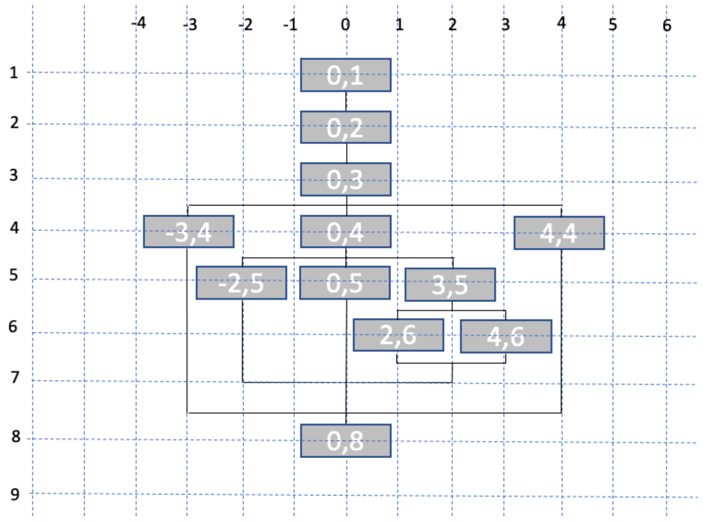


Figure 12: A more complex workflow in Display mode

Describing the hierarchy

Figure 13 shows the flowchart in Figure 12, with a hierarchical numbering system. This is the form that blocks are stored in in the back end and will be passed to the front end for drawing. As shown, numbers increase as the flowchart goes through steps. Blocks 1 and 2 are steps. Block 3, however is a branch, and all of the steps (until we get to Block 4 at the bottom of the Figure) are carried out in the context of that branch. Similarly, the blocks notated with 3.2.1, 3.2.2 and 3.2.3 are all subordinate to the branch at Block 3.2.

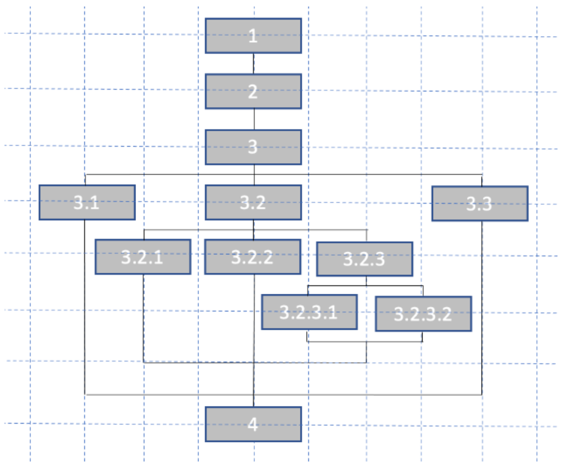
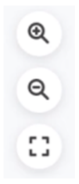


Figure 13: More complex workflow with hierarchical labelling

Pan, Zoom and Show all

The display should include buttons for zooming in and out. The screen should pan by clicking and dragging at any point. The "show all" button will zoom out so that the entire flowchart is displayed in the window. A bar such as shown below (either in horizontal, or vertical as shown) can be used. The display should also include a toggle between "Edit" and "Display".



Changing the workflow chart

Changes can be brought about to the workflow chart through

(a) dragging an element onto a "plus" within the existing chart
(b) clicking on a plus (triggering an "external to the component" add of an item at a certain point.
(c) a change elsewhere in the system

In each case, a redraw is initiated in the front end.

The process of dragging an element into a workflow operates is as follows (refer also to Figures 14 and 15, below)

1. One of either a "step" or a "branch" is selected from a palette at the LHS of the window
2. Once a block has been selected, it can be dragged from the palette into the main working part of the window.

3. The palette item selected becomes greyed out whilst the dragging takes place
4. When the cursor enters the vicinity of a “plus” as defined by the boundaries of the surrounding block (as seen in Figure 15), the block is shown in shaded outline
5. The release of the cursor when a block is shown in shaded outline will result in the addition of that type of block in that position into the workflow. This will result in
   a. Communication to the back end that this has occurred
   b. The triggering of a redraw of the workflow when it is updated as described in Figure 9 or 10.

Note that as the cursor can only be over one block at a time, only one will be active for dropping.

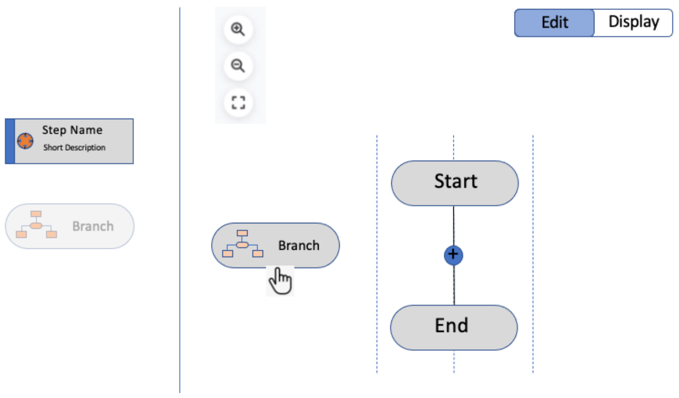


Figure 14: Dragging a block

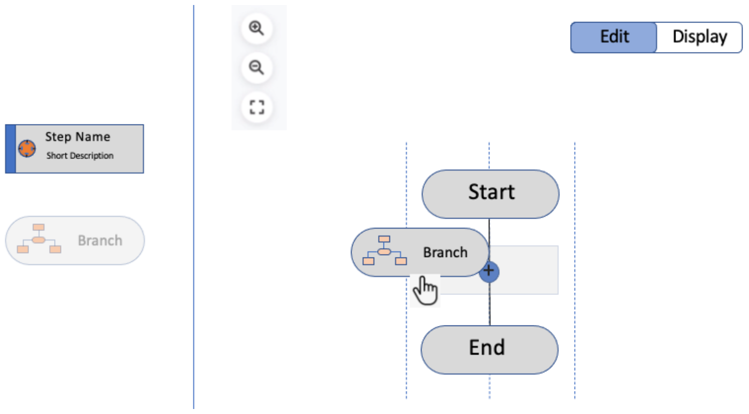


Figure 15: Dragging a block: now ready for release