API requirements for the Workflow editor module

This documents lists some specific items that the Make API should provide for the workflow editor module of the FlowOpt project. They are by no means all necessary – the API doesn't have to directly include all of the listed items, however it should provide sufficient means to make implementation of those items possible.

Data transfer

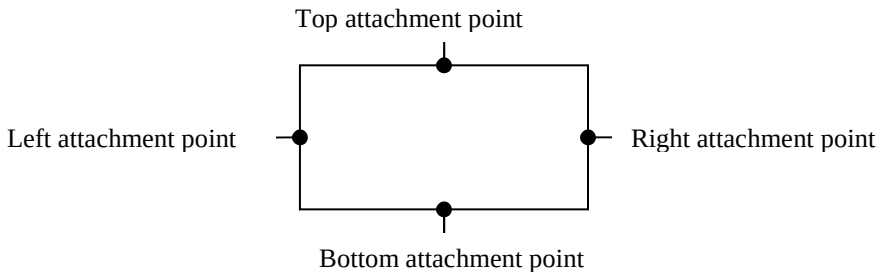
- Using the specified XML files.
- Save / load a workflow from the DB, list all defined workflows, activities, resources in the DB.

Settings

- Save and load application settings to / from the database.
- Preferred for me: a raw further unspecified XML.

Activities

- Similar to Make's activities – rectangular with labels
- Draw an activity on a specific position on the screen.
- Get/set activity's width, height, color (fill + border), transparency
- Activity's name – rendered in the center of the activity, possible to specify font, size and color.
- Change the activity's border thickness and border style (line, dash, custom – pattern created by repeating given bitmap glyph).
- Activity has four attachment points. These are the points where arcs may connect to the activity.



Arcs

- Arcs similar to Make – oriented lines between activities and tasks
- Draw an arc from one activity to another, i.e. start and end point of an arc should be specified either as an activity object + attachment point on the activity (i.e. top, down, left right), or an absolute position on the screen. Example API (constructors)

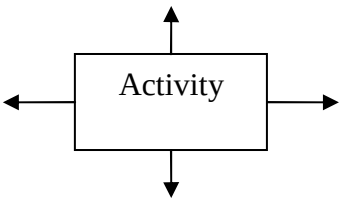
```
Arc.Arc(Point from, Point to)
OR
struct ArcDefinition {
      Activity fromActivity;
      Activity.AttachmentPoint fromAttachmentPoint;

      Activity toActivity;
      Activity.AttachmentPoint toAttachmentPoint;
}
Arc.Arc(ArcDefinition)
```

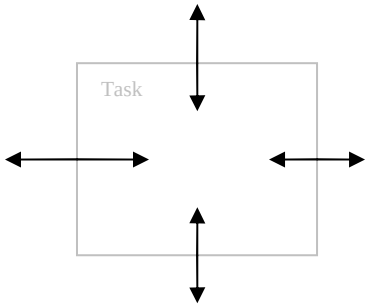
- Change an arc's color, thickness and transparency.
- Change an arc's start and end points (arrow / circle / diamond / custom shape – provided bitmap glyph).
- Way to set "breakpoints" on the arc, i.e. create curves / corners on the arc, so it isn't just a straight line.
- Arcs may have a text label assigned to them, rendered with the arc, possible to set font, font size and color
- Algorithms on leading the arcs on the screen so they don't intersect with other objects would be welcome but not necessary

Tasks

- Collapsed tasks can be rendered using the activity rendering API.
- Expanded tasks are rendered as a rectangular border around a nest of activities. „Border" means that it must be possible to render inside structure (preferably without the need to adjust z-order, i.e. border should have no fill).
- Render this border on specified position on the screen.
- Change the border's color, thickness, transparency, style (provided bitmap glyph).
- Get/set the border's width, height.
- Task's name – rendered in the top left corner of the task, possible to set font, size and color.
- Border has the same 4 distinct points as an activity, again they serve as attachment points for arcs, however in this case the arc may also lead „inside" the border, unlike activities where the arcs only lead „outside" (only relevant if there is some algorithm on leading the arc between objects so they don't intersect):



For activities, arcs only lead from the attachment points to the outside



For tasks, arcs can lead both inside and outside

Activity list

- Get all activities defined in the database.
- Get all activity groups defined in the database, together with information on which group does which activity belong to.
- Component to visualize activity list and activity groups and allow the user to pick activities.
- Some support for drag/drop behavior on this component – for ex. an event when the user grabs an activity from the list + an event when the user lets go of the activity over the canvas.

Misc

- A “canvas” like component to render onto – all absolute positions are within this component, an option to show or hide grid (set color of the grid).
- Set the z-order for all visible objects (including arcs).
- Standard events on this canvas – mouse clicking, double clicking, moving, dragging, key type, key press. Broadcasted by objects (activities, tasks, arcs).
- Standard windows GUI elements – label, button, textbox, menu, listbox etc…