# SRS - Workflow editor module of the Flow Opt project

This document specifies software requirements on the workflow editor module of the FlowOpt project.

## *Used terms and remarks*

This section describes the terminology used throughout this document, font conventions and other metainformation relating to this document as a whole.

Throughout this document, I sometimes refer to information contained in other parts of the FlowOpt project specification (for example the declarative format for workflows). I therefore assume that the reader is familiar with the rest of the FlowOpt project specification, or at least its parts relevant to this module.

**Font conventions**
In the text, whenever I mention some term that will be referred to later in the document, it is written in *italics*. Fragments of declarative format code will be written in `courier`. In some cases, I'm not yet certain on the solution of the specific problem – these cases are marked by (TBD) mark and will be resolved later (advice on such problems is especially welcome).

**Nested TNA model description**
Our workflow model is based on the Nested TNA workflow model introduced in http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.69.5684&rep=rep1&type=pdf. Any information on this model not provided by this document can be found there.

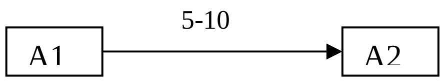
**Declarative format**
By declarative format, I mean the data format used to pass a workflow from this module to the scheduling engine module. Its complete description is in the project specification. In this document, I use n-ary `Activity` predicate instead of unary simply to save space, therefore instead of:

```
Activity(A1, Duration of A1, Cost of A1, …)
Activity(A2, Duration of A2, Cost of A2, …)
Activity(A3, Duration of A3, Cost of A3, …)
```

I only write

```
Activity(A1,A2,A3)
```

Also, for every arc there will be a predicate with its temporal constraint like in following case:



Produces a predicate like
`Temporal(A1,A2,5,10,ES)`

I don't explicitly write these temporal predicates, again to save space. All temporal predicates that are explicitly written are independent of individual arcs (they are a part of specific pattern transformation).

**Various remarks**
For the sake of this document, clicking refers to left clicking; dragging refers to dragging the mouse while holding the left mouse button.

## *Application overview*

This application will be a part of the Flow Opt project – specifically it will be a graphical workflow editor module, which will allow the user to create or import a workflow, edit it, save it and use it for scheduling by other Flow Opt modules. Emphasis will be on user friendliness, usability and completeness with regard to this specification.

### Hardware and software requirements

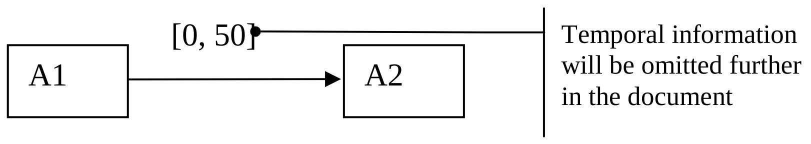
The application will be written in C# language and therefore, the user will be required to have .NET framework 3.5 installed to run this application. As for hardware requirements, a common up to date PC (2GHZ CPU, 1GB RAM) should be sufficient to run the application smoothly.  Since the FlowOpt project's modules will communicate via network, the user's computer will also have to have network connectivity in order to use other FlowOpt modules remotely. Since network traffic should be minimal, any current connection (256 Kbit/s and faster) should be sufficient.
Obviously in order to access functions provided by other FlowOpt modules locally, the user will have to have those installed as well and therefore meet the HW and SW requirements for these modules. Those can be found in FlowOpt specification.

## *Workflow model*

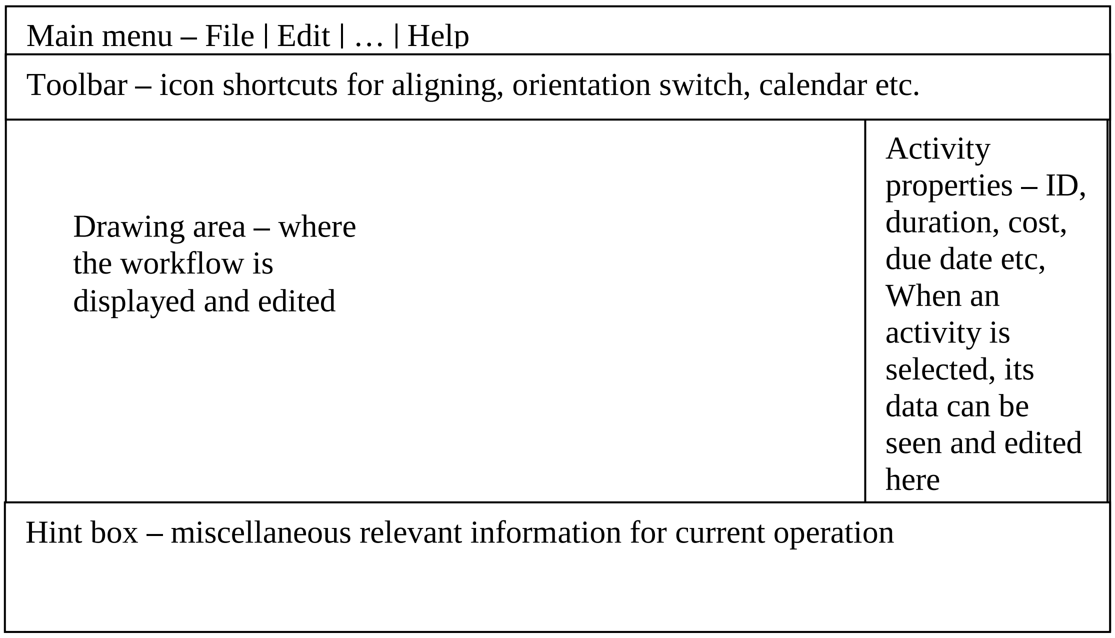
Our workflow model is based on the Nested TNA model, but it offers more. The user will be able to perform all operations relevant to the Nested TNA model (parallel/alternative/serial decomposition), plus he will be able to arbitrarily specify temporal and logical constraints. This means that the user can choose any two activities and specify a temporal or logical condition for them. All this will be thoroughly specified later in this document.

## *User interface*

The workflow is visualized by drawing activities as rectangles with their IDs and arcs connecting them as arrows with temporal information, as in the following figure. The user will be able to either display the temporal information at all times or only show it upon mouse-over.
In the rest of the figures in this document, temporal information will be omitted for the sake of simplicity.


[0, 50]
A1
A2
Temporal information will be omitted further in the document

Below is the basic overview of the main form.


Main menu – File | Edit | … | Help
Toolbar – icon shortcuts for aligning, orientation switch, calendar etc.
Drawing area – where the workflow is displayed and edited
Activity properties – ID, duration, cost, due date etc, When an activity is selected, its data can be seen and edited here
Hint box – miscellaneous relevant information for current operation

## Zooming

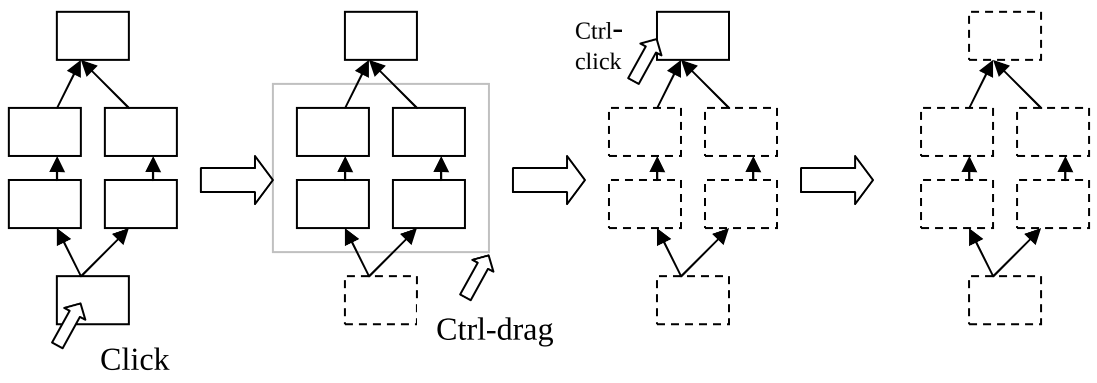
The user will be able to zoom in and out. There won't be a magnifying glass functionality (i.e. the possibility to enlarge just a selected part of the screen, without the rest changing size).

## Undo

The user will be able to perform an undo operation for all main operations. Its depth will not be unlimited. In some cases where undo operation is not well defined, it may not be available (TBD).

## Selection

The user will be able to select a single activity by left clicking on it. By doing so, its properties will be displayed in the activity property form, where they can be viewed and edited.
By ctrl-clicking on activities, they are added/removed to/from current selection. By dragging the mouse when nothing is selected, a rectangle appears indicating selection – upon releasing the mouse button, all activities within the rectangle will be selected. By ctrl-dragging the mouse, multiple activities can be added to selection. When multiple activities are selected, their properties cannot be edited; the activity property form is only active when a single activity is selected. Selected activities are highlighted (dashed line). Same applies for arc selection.



## Activity and arc properties

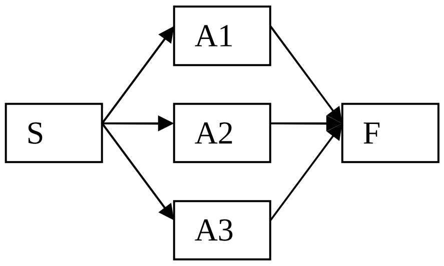
Both activities and arcs will have certain properties, which the user will be able to view and edit. In case of activities, these properties include for example ID, duration, cost, due date, late cost, resource requirements etc. There will be a special panel for viewing and editing these properties (see main form overview). Upon selecting a single activity, its properties are loaded into this form where the user may view and edit them.
As for arcs, they have a single property – a temporal constraint in the form of a minimal and maximal time distance between the activities that the arc connects. The user may input temporal information for any arc by double clicking on it, which will cause an input field to appear, allowing user to enter the desired time interval.

**Resources**
One of activity properties to take special note of are its resource requirements. These will be specified in the form of necessary attributes the resource must have in order for it to be able to perform the activity. Specifically, the resource editor (another FlowOpt module) will allow the user to define a set of resource attributes. In this (workflow editor) module, the user will be able to assign requirements on those attributes to any activity.
These requirements will be entered as a percentage (1 to 100%). For example, in resource editor the user might define resource attributes strength, sharpness, durability and precision. For an activity “Saw lumber” the user may specify a requirement of 75% sharpness, 50% durability and 25% precision. For an activity “Carry planks to warehouse” the user may specify 100% strength attribute requirement. Finally, for activity like “Make a chair” the user may enter 30% durability and 80% precision requirement.
There will be special dialogue in the activity property form to allow the user to enter the above described data.

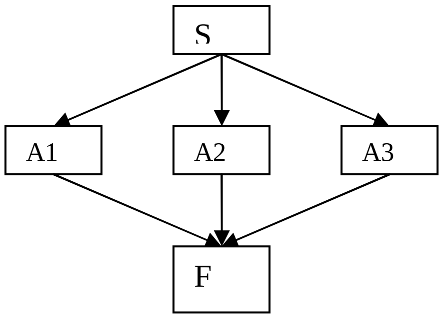
## Choice of orientation

In any moment, the user will be able to switch between two orientations – horizontal (left to right) and vertical (top to bottom). The following figure illustrates both cases:

Horizontal orientation:



Vertical orientation:



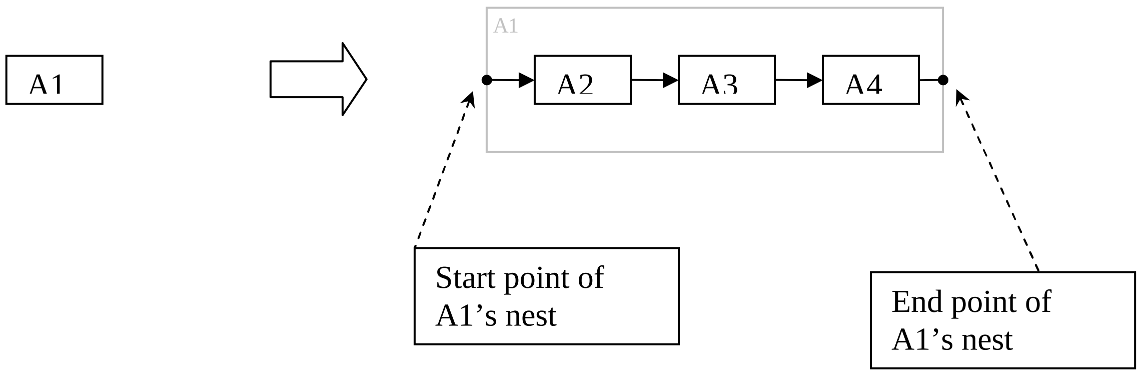
## Editing

In the beginning, the user starts with a workflow consisting of a single activity (vertex)[1]. For sake of this specification, this activity will be called the *initial activity* and workflow consisting of this single activity will be referred to as *initial workflow*. On this elementary workflow, the user can apply several editing actions.
To change the workflow structure, the user must perform decomposition – one activity is decomposed into several new activities, which are to be executed in a certain way in place of the original activity. Upon decomposition, an activity becomes a *composite* activity. All activities created by the decomposition belong to a *nest* of the decomposed activity. Activities that are not composite, i.e. they have not been decomposed, will be referred to as *elementary* activities.
Every nest has two special points to take note of – the *starting point* and the *end poin*t. The former represents the point at which this nest starts; the latter represents the point at which the nest ends. These points are marked by a black dot on edges of the nest (see figures below). These points serve to connect activities within the nest to the parent composite activity – the start point corresponds to start of the parent activity and the end point corresponds to end of the parent activity. Therefore, it is possible to specify constraints like for ex. time distance of a nested activity from the start of the parent activity or to the end of the parent activity.
Besides decomposition, the user can also arbitrarily specify binary temporal and logical constraints. This means that there will be two kinds of arcs in our workflow – those carrying temporal constraint and those carrying logical constraint. The former will be referred to as *temporal arcs* while the latter will be referred to as *logical arcs*.

All editing options are further described here:

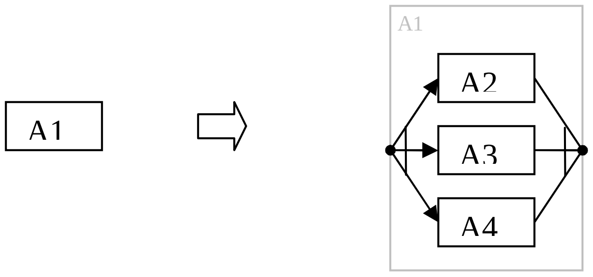
- **Serial decomposition** – the user can split a single activity into multiple activities, which will be executed one after another in place of the original activity.
  Here A1 is decomposed into activities A2, A3 and A4. A1 thus becomes a composite activity, with A2, A3 and A4 in its nest. The editor will visualize the nest like in the figure – as a grey rectangle over all the nested activities, with a name of the composite activity. It will be possible to turn this visualization off and only display the activities.

[1] In the formal Nested TNA model, the user starts with two activities connected by an arc, however unlike the formal model, we decompose the vertices rather than arcs, hence the slight change of model.



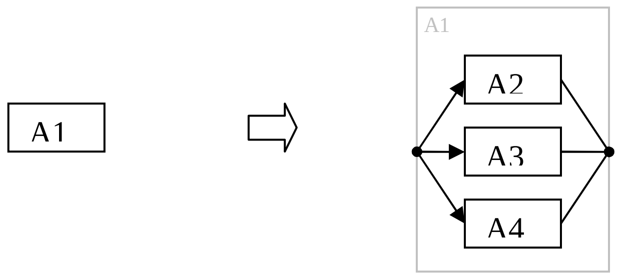
**Declarative format transcription:**
```
Activity(A1)
=>
Activity(A1,A2,A3,A4)
Temporal(A1,A2,0,inf,SS)          //start of A1’s nest to start of A2
Temporal(A2,A3,0,inf,ES)          //temporal arcs between created
Temporal(A3,A4,0,inf,ES)             activities
Temporal(A4,A1,0,inf,EE)          //end of A4 to end of A1’s nest
Decomposition(A1,[A2,A3,A4],AND)  //nest creation
```

- **Parallel decomposition** – the user can split a single activity (vertex) into multiple activities, which will be executed in parallel in place of the original activity (AND split / join). In this example, A1 is decomposed into A2, A3 and A4. Arcs leading to and from these activities are marked with a line crossing them – this indicates the parallel (AND) relation.



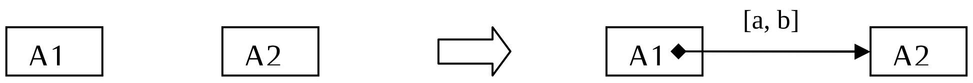
**Declarative format transcription:**
```
Activity(A1)
=>
Activity(A1,A2,A3,A4)
Decomposition(A1,[A2,A3,A4],AND)
Temporal(A1,A2,0,inf,SS)     Temporal(A2,A1,0,inf,EE)
Temporal(A1,A3,0,inf,SS)     Temporal(A3,A1,0,inf,EE)
Temporal(A1,A4,0,inf,SS)     Temporal(A4,A1,0,inf,EE)
```

- **Alternative decomposition** – the user can split a single activity into multiple activities, out of which a single activity will be executed in place of the original activity (XOR split / join). In the following figure, A1 is decomposed into A2, A3 and A4. Notice that unlike in the previous case, there is no line crossing arcs incident with A2, A3 and A4 – this means alternative decomposition.



**Declarative format transcription:**
```
Activity(A1)
=>
Activity(A1,A2,A3,A4)
Decomposition(A1,[A2,A3,A4],XOR)
Temporal(A1,A2,0,inf,SS)      Temporal(A2,A1,0,inf,EE)
Temporal(A1,A3,0,inf,SS)      Temporal(A3,A1,0,inf,EE)
Temporal(A1,A4,0,inf,SS)      Temporal(A4,A1,0,inf,EE)
```

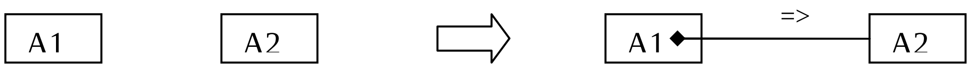
- **Adding a temporal constraint between two activities** – the user can specify a time interval [a, b] for a couple of activities A1 and A2 meaning that there has to be at least a and at most b time units between A1 and A2. In this example we specify a time constraint for activities A1 and A2.
  The user will also be able to specify the exact type of the newly created temporal constraint – it can be a start-start, start-end, end-start or end-end constraint.
  Note that the arc's starting point is different from the temporal arcs which are created through decomposition and it is also marked (diamond) to indicate that this temporal arc was created arbitrarily, not through decomposition.



**Declarative format transcription:**
```
Activity(A1,A2)
=>
Activity(A1,A2)
Temporal(A1, A2, a, b, ES)
```

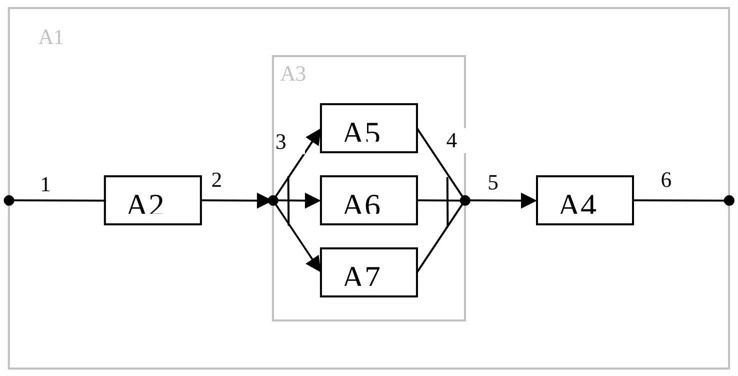
- **Adding a logical constraint between two activities** – the user can also specify a logical constraint for a couple of activities A1 and A2. Namely, the user can use implication (if A1 is executed, then A2 must also be executed), equivalence (A1 is executed if and only if A2 is executed), exclusive or (if A1 is executed, A2 is not and vice versa) and inclusive or (A1, A2 or both are executed).
  These constraints will be visualized similarly to the temporal constraints – as arcs. These arcs will not be directed and may be drawn with different color. To prevent confusion, the user will be able to only show one kind of arcs (temporal/logical), show

one and draw the other in the background grayed out, or show both kinds of arcs normally.
In this example, we specify an implication constraint for activities A1 and A2. As in the case of arbitrary temporal arcs, arbitrary logical arcs also start in a different point and their start is also marked (diamond) in order to distinguish them from arcs created by decomposition.



**Declarative format transcription:**
```
Activity(A1,A2)
=>
Activity(A1,A2)
Logical(A1, A2, =>)
```

**Transcribing arcs into declarative format**
There are many cases where arc constraint transcription into declarative format may not be obvious. In case an arc directly connects two activities, transcription simply produces a single predicate (either `Temporal` or `Logical` depending on arc type).
However, some arcs also connect activities with starting or end points of a parent activity, as in following figure (arcs that will be referred to below have been numbered):



There are basically four special cases that may happen (I assume all temporal constrains in the picture are [0, inf] since exact values don't matter here):

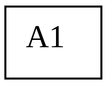
- Arcs leading from the starting point of a parent activity (arcs 1, 3 in the figure) – those produce a temporal constraint of the SS type (start – start) like so:
  `Temporal(A1,A2,0,inf,SS)`
  `Temporal(A3,A5,0,inf,SS)`
- Arcs leading to an ending point of a parent activity (arcs 4, 6 in the figure) – those produce a temporal constraint of the EE type (end – end) like so:
  `Temporal(A5,A3,0,inf,EE)`
  `Temporal(A4,A1,0,inf,EE)`
- Arcs leading to starting point of a composite activity (2 in the figure) – those simply produce a temporal constraint of the ES type (end – start). Notice that these arcs were already described, since they actually do connect two activities, for example arc 2 connects A2 and A3 and produces following constraint:
  `Temporal(A2,A3,0,inf,ES)`

- Arcs leading from an ending point of a composite activity (5 in the figure) – same thing as in previous case, these arcs again just connect two activities. For example arc 5 connects activities A3 and A4, and therefore simple ES type temporal constraint is sufficient, like so:
  `Temporal(A3,A4,0,inf,ES)`

## Description of editing process

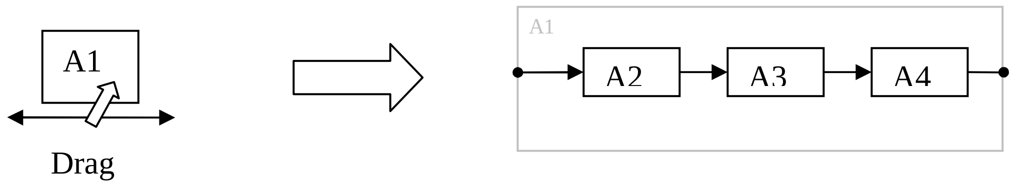
Here is a step-by-step example of creation of a simple workflow to illustrate the process. It assumes the horizontal orientation (for vertical orientation, the left-right dragging is replaced by up-down dragging and vice versa). Temporal predicates for arcs leading to/from a starting/end point have been omitted for the sake of simplicity.

1) The user starts with a single activity (the initial activity), which represents the entire workflow.



**Declarative format transcription:**
```
Activity(A1)
```

2) Serial decomposition application – the user clicks on the activity, which he wants to decompose (in this case A1) and drags the mouse while holding the left mouse button. Dragging to the right adds more activities to the decomposition, dragging to the left removes activities from the decomposition (in this case, the user splits the A1 activity into three serially executed activities A2,A3 and A4):

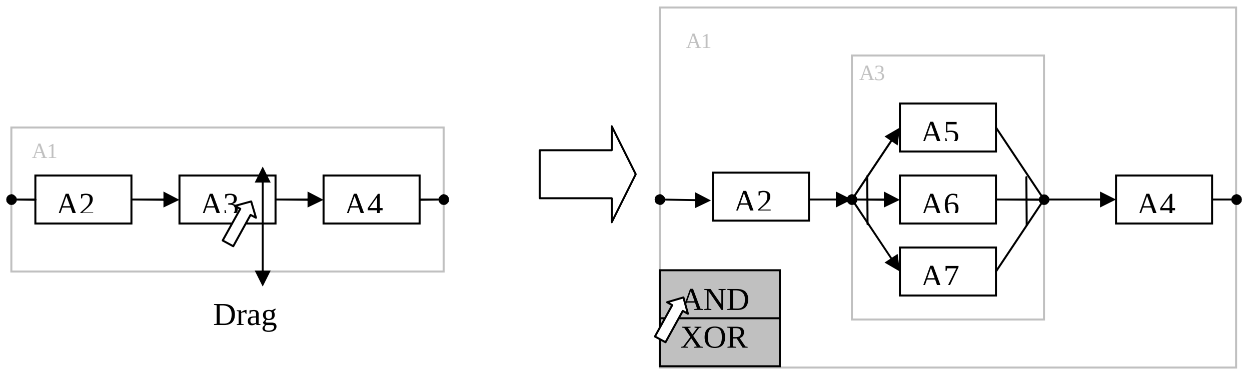


Declarative format transcription:
```
Activity(A1,A2,A3,A4)[2]
Decomposition(A1,[A2,A3,A4],AND)
Temporal(A2,A3,0,inf,ES)
Temporal(A3,A4,0,inf,ES)
```

3) Parallel decomposition application – The user clicks on the activity, which should be decomposed (in this case A3) and by holding the left mouse button and dragging the mouse he specifies the number of activities in the decomposition similarly as in the serial decomposition case – dragging up adds more activities into the decomposition, dragging down removes activities from the decomposition.
   Here the user decomposed the A3 activity into three activities A5, A6 and A7, which should be executed in parallel in place of the original A3 activity. When the user releases the mouse button, a pop-up menu automatically appears with the choice of

[2] To save space, I use n-ary predicate instead of unary for specifying activities.

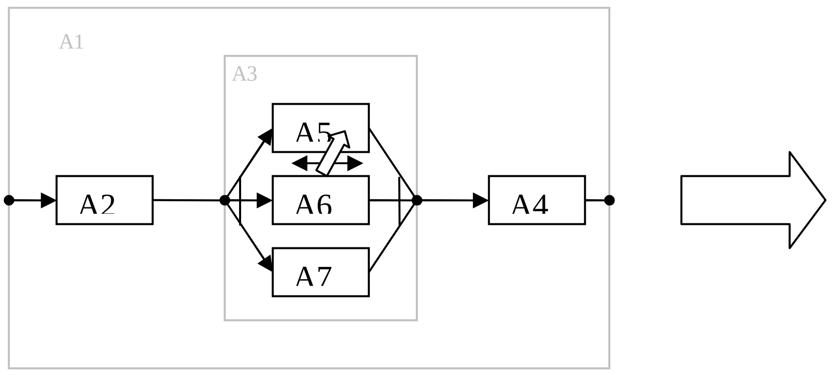
either AND split (parallel decomposition) or a XOR split (alternative decomposition) – when the user wishes to perform an alternative decomposition, the process is the same as with parallel, except for selecting XOR split in the pop-up menu.

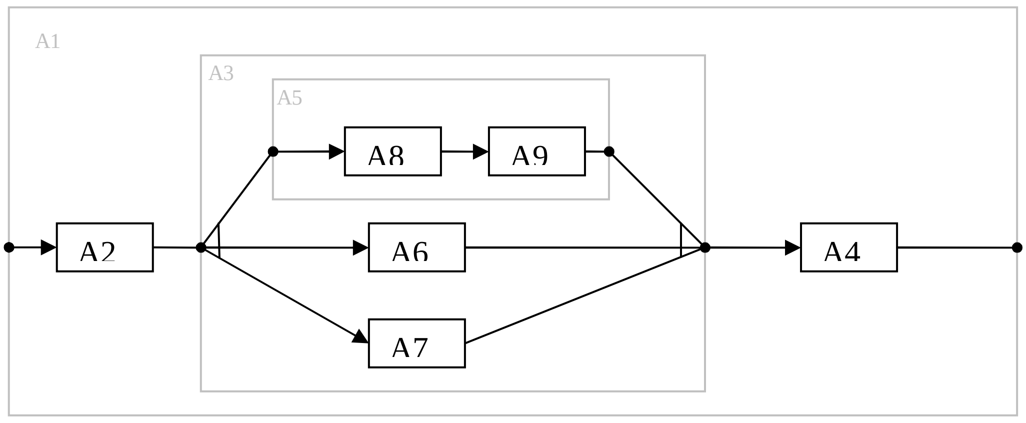


Declarative format transcription:

```
Activity(A1,A2,A3,A4,A5,A6,A7)
Decomposition(A1,[A2,A3,A4],AND)
Temporal(A2,A3,0,inf,ES)
Temporal(A3,A4,0,inf,ES)
Decomposition(A3,[A5,A6,A7],AND)
```

4) Further serial decomposition. This time the user chooses to decompose A5 into two activities A8 and A9:

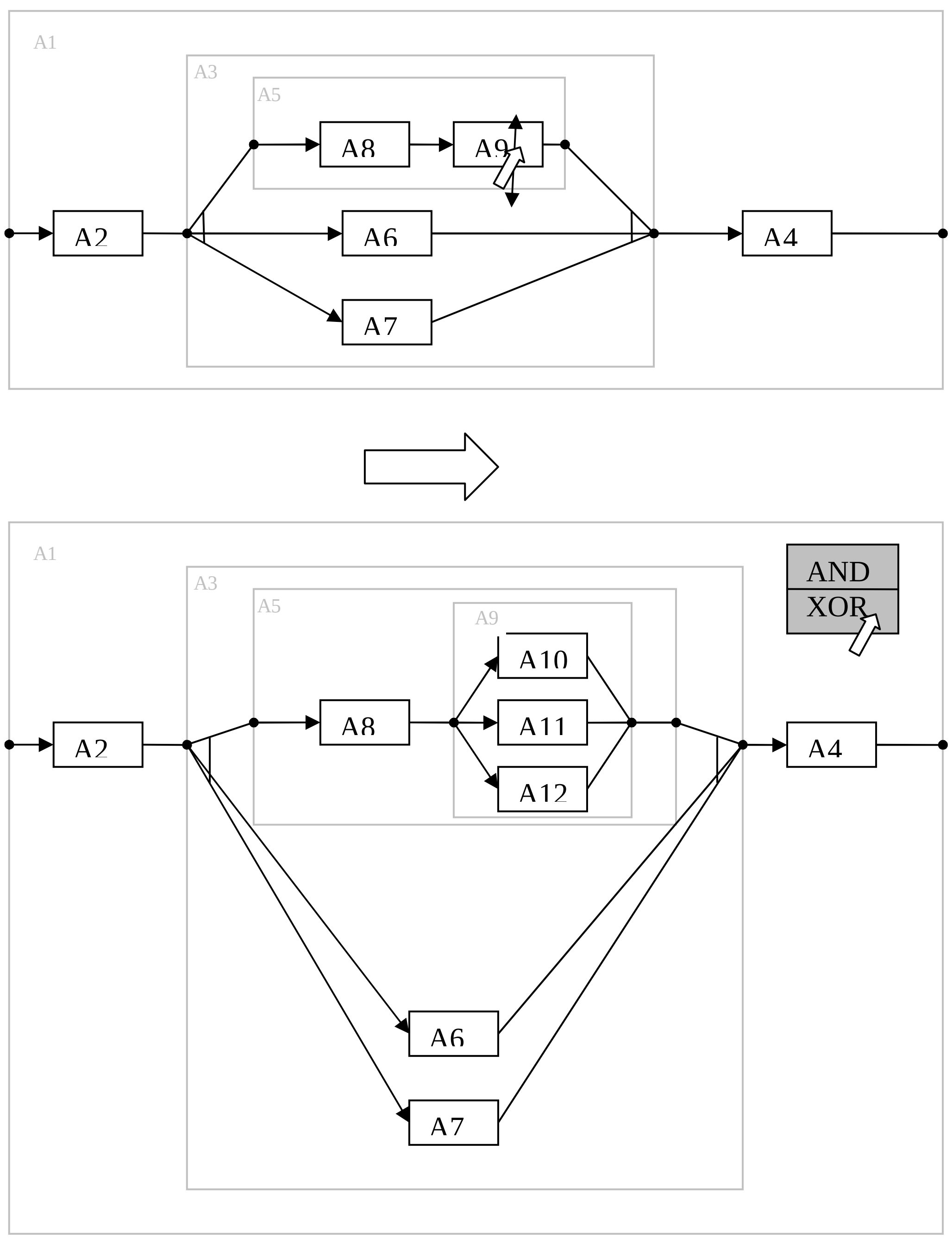




Declarative format transcription:

```
Activity(A1,A2,A3,A4,A5,A6,A7,A8,A9)
Decomposition(A1,[A2,A3,A4],AND)
Temporal(A2,A3,0,inf,ES)
Temporal(A3,A4,0,inf,ES)
Decomposition(A3,[A5,A6,A7],AND)
Decomposition(A5,[A8,A9],AND)
Temporal(A8,A9,0,inf,ES)
```

5) Alternative decomposition – As stated earlier, the process is similar to that of parallel decomposition, except in the pop-up menu XOR is selected indicating that the user wishes for an alternative decomposition, not parallel. Here the user chose to decompose the A9 activity into three activities A10, A11 and A12. When the A9 activity should be performed in the original workflow, exactly one of the activities A10, A11, A12 will be executed in the new workflow:

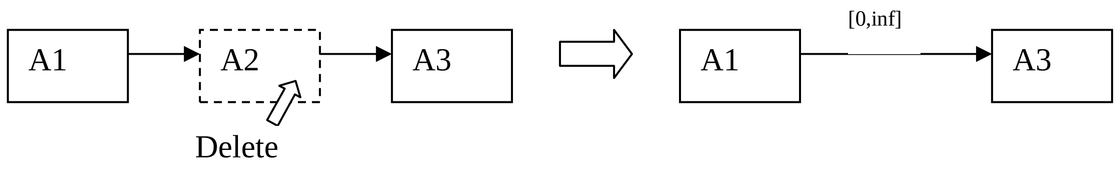


Declarative format transcription:

```
Activity(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12)
Decomposition(A1,[A2,A3,A4],AND)
Temporal(A2,A3,0,inf,ES)
Temporal(A3,A4,0,inf,ES)
Decomposition(A3,[A5,A6,A7],AND)
Decomposition(A6,[A8,A9],AND)
Temporal(A8,A9,0,inf,ES)
Decomposition(A9,[A10,A11,A12],XOR)
```

**Another way to do editing actions**
An alternative way to perform decomposition will be through right clicking on an activity the user wishes to decompose and selecting “Decompose this activity” option from the pop-up menu that appears. This will produce a dialog allowing the user to enter the number of activities that should be in the nest and type of decomposition. This option is mainly intended for large-scale decomposition, where dragging would not be convenient for the user.
For the same reason, there will also be a special dialog for creating temporal and logical constraints.

**Removal of activities**
The user can remove the selected activities within one nest by pressing the delete key or by selecting the right option from right-click pop-up menu. For every deleted activity, all incident arcs are automatically deleted too (can be disabled in settings). If a user removes all activities from the nest, it is the same as removing parent composite activity.
In case the user removes an activity within a sequence, it is also removed together with (two) incident arcs. If the removed activity had two neighbors, a new arc with temporal constraint [0, inf] is created to connect them as in the following figure.

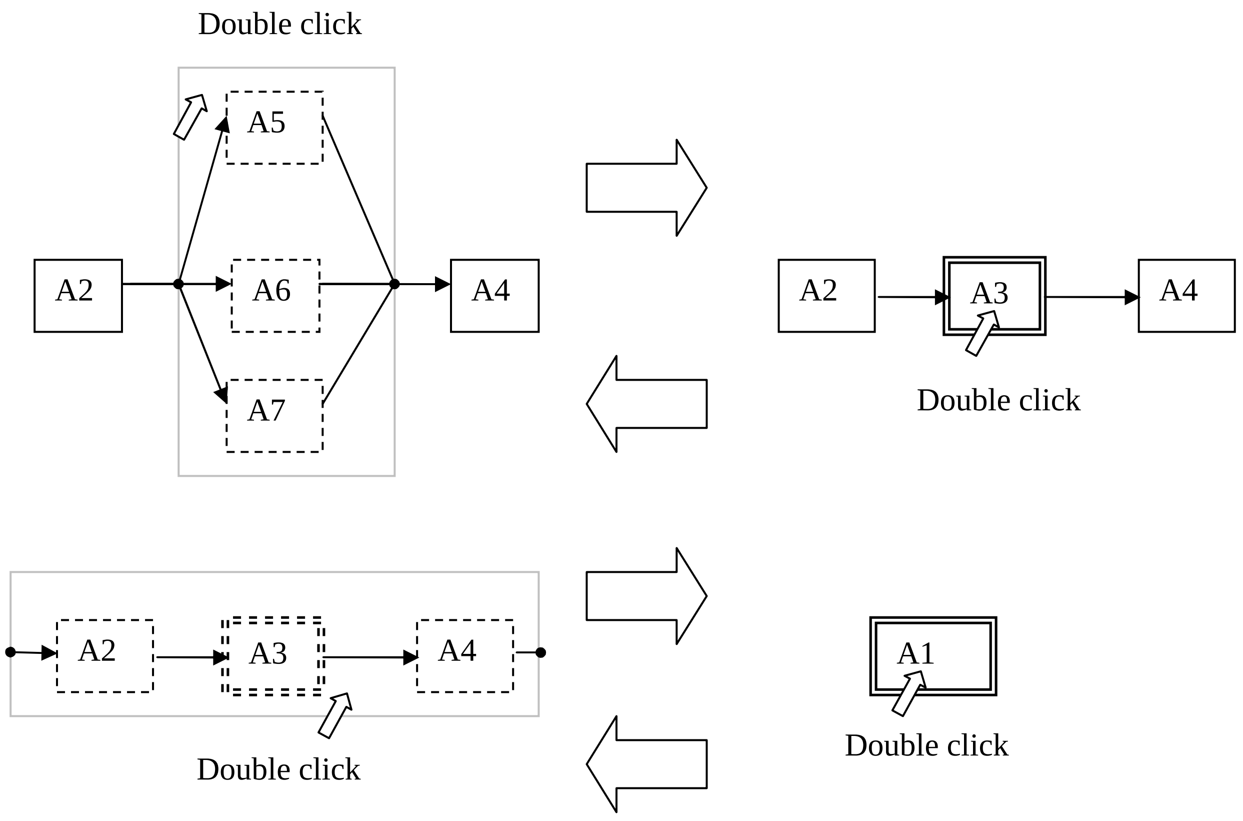


## Hierarchy

The user will be able to work with the natural hierarchy induced by the Nested TNA model. Since every decomposition creates a nest within decomposed activity, we can see a tree-like hierarchy with the initial activity in its root and elementary activities in its leaves. The user will be able to show or hide any subset of the nests he created, as well as expand a single nest into new window as a separate workflow, edit it and save it either into the original workflow in place of the expanded nest, or as a completely new workflow.

**Process description**
When the user moves mouse pointer over any activity, all other activities belonging into the same nest are highlighted automatically. Upon double clicking on the activity, the nest is collapsed into a single activity (the original activity), marked in a special way indicating that it is not an elementary activity, but a composite one containing a nest of activities. Upon double clicking on a composite activity, it is once again replaced by the nest of activities within it. This process can be applied arbitrary number of times, giving two extremes: fully expanded workflow, where every elementary activity is visible and fully collapsed workflow, where only one composite activity corresponding to the initial activity (the first one in the workflow, representing the entire process) is displayed.
The user can also right click anywhere in a nest (or on a composite activity) and select “Expand in a new window” option to open a separate window containing just the selected nest.
Note that any properties of an activity remain intact if the activity is decomposed into a nest of activities, they are simply related to the whole composite activity.

**Example**
In this example case, the user created a network consisting of 5 elementary activities A2, A4, A5, A6 and A7. Activities A5, A6 and A7 form a nest decomposed from activity A3, so upon right clicking on any of them, this nest is collapsed as seen in the first figure (arrow to the right). On the other hand, when A3 is collapsed, it can be expanded again by double clicking on it.
The second figure illustrates the same thing one hierarchy level higher – since A2, A3 and A4 are also a nest created from A1 (the initial activity), the user can collapse this nest too. Upon right clicking on A2, A3 or A4, the nest is collapsed and only the initial (now also composite) activity A1 is displayed. Double clicking on it can now expand this activity.



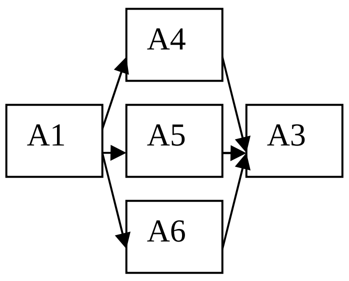
## Manipulating activities

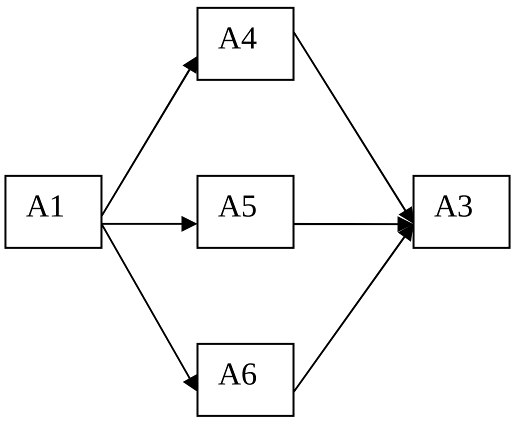
It will also be possible to do the following:

- Add more activities to a nest – right click + “Add more activities to this nest”. A dialogue appears where the user may enter a number of activities to be added.
- Remove activities from a nest – select activity and press the delete key or right click + “Remove selected activity”. All incident arcs will also be deleted.
- Change the order of activities in a nest (“swap” activities) – changes the execution order in case of sequence, for parallel / alternative decomposition it is only for user’s convenience. It will probably be done by using a special “cursor mode” for swapping activities within the same nest.

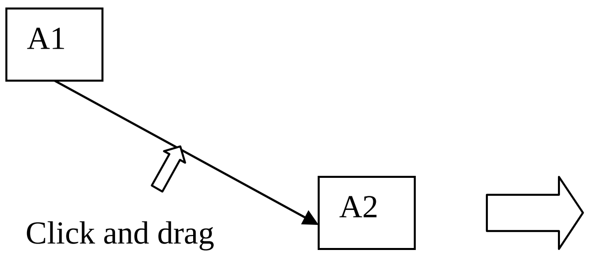
**Padding**
The user will be able to set the horizontal and vertical padding between activities, i.e. how much space is there between activities.

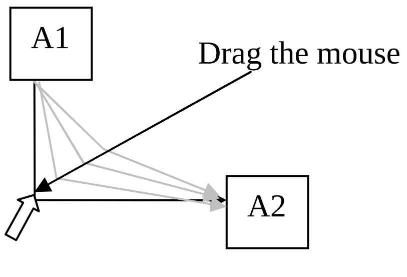
Example:
Low padding Higher padding

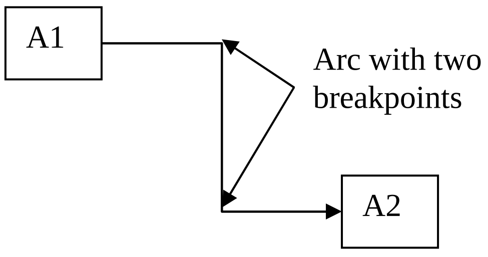




**Transforming arcs**
The user will be able to transform (only) temporal and logical arcs which were not added through decomposition by adding active breakpoints on them like so:







## Workflow analysis

If the user chooses to use workflow created in freelance mode for scheduling by other modules in FlowOpt, the application will automatically check for basic correctness of the workflow. The user can also invoke this check explicitly.
Basic correctness means, that the workflow is a graph with no cycles given by temporal arcs (TBD). If this check fails, the workflow cannot be used for scheduling.

## Calendar

The application will also provide a simple calendar as a means to specify when an activity can be scheduled (for ex. the user will be able to specify that activity A1 can only be scheduled on Mondays from 10:00 to 12:00).

## Import/Export of the workflow

The user will be able to import/export a workflow from/to the YAWL format (http://www.yawlfoundation.org/). Since the YAWL format is significantly more complex than our workflow format, there will be some modifications on the workflow begin imported from YAWL – the user will be notified of these modifications and, in case it is needed, the users feedback will be requested on how should the YAWL workflow be transformed to fit into our workflow format. The main goal here is to create a workflow visually resembling the one being imported from YAWL, not to create an identical workflow in our format (that is impossible due to said differences between our format and YAWL).

## Table of Contents