Software Requirements Specification

for

Schedule Visualizer

Version 2.1
Author: Tomas Skalicky
28.4.2010

History of versions

Date	Changes / Notes	Version
3.7.2010	Short summary	0.9
22.3.2010	First full-value version	1.0
4.12.2010	Changes discussed with Mr. Sheahan (tasks vs. activities, filters of tasks in Resource View, banning of branches, etc.)	2.0
28.4.2010	Model input and model output specified	2.1

Content

1. Introduction

1.1 Purpose of this document

This document is a specification of the module called Schedule Visualizer. This module will be a part of FlowOpt project. Other modules of this project will be Workflow Editor, Optimizer and Schedule Analyzer.

1.2 Notation

For tasks or activities *TA1*, *TA2* and activity *A1*, we introduce the following notation:

Start(TA1) .....start time of *TA1*
Finish(TA1) .....finish time of *TA1*
Duration(TA1) .....duration of *TA1*
(*Duration(TA1) = Finish(TA1) – Start(TA1)*)

Temporal(TA1, TA2, X, Y, ES) .....temporal constraint between *TA1* and *TA2*.
It means: *Start(TA2)* ≥ *Finish(TA1)* + *X* and *Start(TA2)* ≤ *Finish(TA1)* + *Y* at the same time.
4 possible types:
- *SS* (start-to-start),
- *SE* (start-to-end),
- *ES* (end-to-start),
- *EE* (end-to-end).

Precedence(TA1, TA2) .....precedence constraint between *TA1* and *TA2*.
It means: *Start(TA2)* ≥ *Finish(TA1)*
~ *Temporal(TA1, TA2, 0, INFINITY, ES)*

Sync(TA1, TA2, SS) .....synchronization between *TA1* and *TA2*.
It means: *Start(TA1) = Start(TA2)*
~ *Temporal(TA1, TA2, 0, 0, SS)*
2 possible types:
- *SS* (start-to-start),
- *EE* (end-to-end).

Pin(A1) .....pin on *A1*.
~ *Temporal(A0, A1, X, X, ES)*, where:
- *A0* is an artificial activity which represents a start of the schedule → *Start(A0) = Finish(A0)* = 0, no allocated resources,
- *X = Start(A1) – Finish(A0) = Start(A1) – 0 = Start(A1)*.

Logical(TA1, TA2, ⇒) .....logical constraint between *TA1* and *TA2*.
It means: *TA1* ⇒ *TA2*
4 possible types:
- ⇔ (equivalence),
- ⇒ (implication),
- *OR*,
- *XOR* (exclusive *OR*).

1.3 Fundamental terms

Activity is an indivisible unit – cannot be expanded on subtasks or subactivities – which is determined by its start and finish time. It can allocate one or more resources.

Task is a container which contains at least one task or activity. Therefore, the task creates a tree hierarchy where internal nodes are tasks and leafs are activities. The task can be either collapsed or expanded. Status ***collapsed*** means that none of sons of the task is shown. Status ***expanded*** is the counterpart of the status collapsed – all sons are shown.
Start and finish time and duration of the task cannot be set or modified. These data is gained from subactivities, i. e. *Start(the task) = Start(its earliest subactivity)*. It is similar for the finish time.
The task cannot allocate a resource.

Both tasks and activities are titled.

Task-activity is used in a context when it does not matter whether it is a task or an activity.

If there are more possibilities of executing schedule, ***branch*** is a compact part of the schedule where these executions are different. ***Main branch*** is a branch which is involved in the schedule, selected by Optimizer as the most optimal schedule. On the other hand, ***alternative branches*** is a branch which is not in the result schedule, but which can be swapped by user with the main one. Branch consists of just one task-activity. This invariant is always satisfied, because only nested TNA (with additional synchronization and logical constraints) are allowed to be created in Workflow Editor.

All-types-of-constraints means these types of constraints:

- *Precedence(TA1, TA2)*,
- *Sync(TA1, TA2, SS)*, *Sync(TA1, TA2, EE)*,
- *Logical(TA1, TA2, ⇔)*, *Logical(TA1, TA2, ⇒)*, *Logical(TA1, TA2, OR)*, *Logical(TA1, TA2, XOR)*.

Just-unary-resources is a name of constraint which says that only unary resources are taken into account in Schedule Visualizer.

1.4 Integration to FlowOpt

Schedule Visualizer will be a part of a process of creating workflows and creating and optimizing a schedule. In Schedule Visualizer, user will be able to modify the schedule which he will get from Optimizer, run Optimizer to gain the better one, run Rescheduling algorithm to just correct broken constraints in the schedule and compare the current schedule with the previous ones using Schedule Analyzer.

Format of data acquired from Optimizer is described in „3. Model input and model output“. In short, the input data will contain information about tasks and activities, all-types-of-constraints, resources, resource allocations etc. However, there will be additional information about tasks, activities and branches in the data.

Each task-activity will be a part of just one workflow and for all of them this workflow will be known. This information will be propagated from Workflow Editor module to the input.

Both main and alternative branches will be in the input.

Schedule Visualizer will work just in ***compatible mode*** with workflows. It means that operations like adding and deleting a tasks-activities and all-types-of-constraints and adding a resource will not be available. The missing functionality is described in detail in „4. GUI“.

2. Non-functional requirements

ID	Name	Description
REQ001	User-friendly GUI and intuitive control	The module will be usable.
REQ002	Fast redrawing of charts	After user will have changed time date of an activity, charts will be redrawn under 1 second.
REQ003	Number of tasks-activities	The module will be able to work with at most 100.000 tasks-activities.
REQ004	Plugin extension	Schedule Visualizer will be a plugin of FlowOpt application.

3. Model input and model output

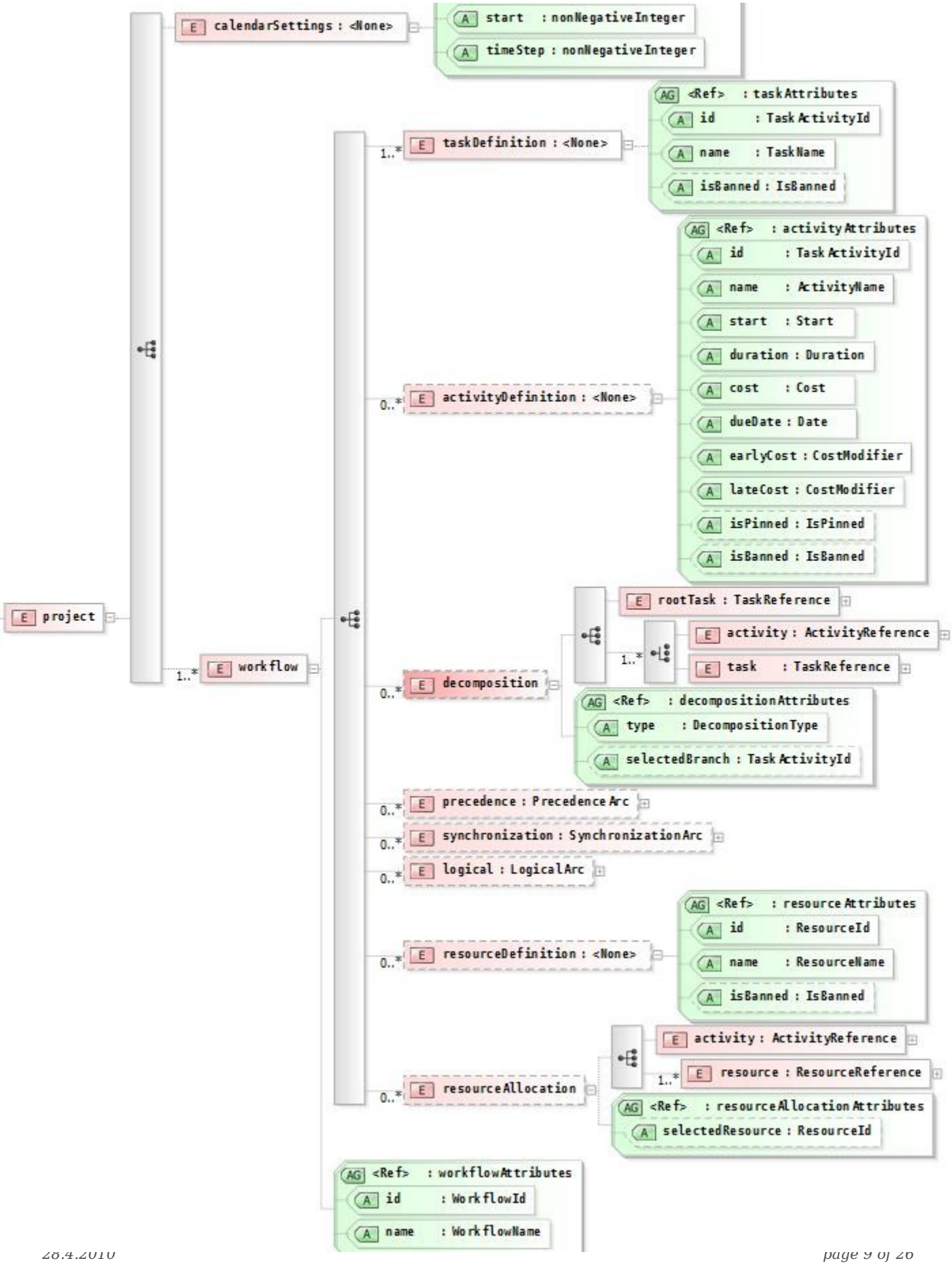
Schedule Visualizer will communicate on the data level only with an orchestration module. This communication will be implemented as sending and receiving XML files. Definition of schema of these files is specified in separated XSD file (sample XML is added too).

This schema (see figure below) is similar to the one which is used in Workflow Editor. There attributes and elements are added:

- ***activityDefinition[start]*** – start time of the activity
- ***activityDefinition[isPinned]*** – true whether the activity is pinned. Otherwise, false.
- ***taskDefinition[isBanned]*** – true whether the task is banned. Otherwise, false. For ***acitivityDefinition*** and ***resourceDefinition***, it is same.
- ***decomposition[selectedBranch]*** – ID of a task-activity which is selected by Optimizer or by user
- ***resourceAllocation[selectedResource]*** – ID of a resource which is selected by Optimizer or by user

In the schema, there are calendar settings as well. They contents:

- ***start*** – timestamp which represents time 0 in abstract time of the application
- ***timeStep*** – how much seconds equal 1 in this abstract time


E project
E calendarSettings : <None>
A start : nonNegativeInteger
A timeStep : nonNegativeInteger
1..* E workflow
AG <Ref> : taskAttributes
A id : TaskActivityId
A name : TaskName
A isBanned : IsBanned
1..* E taskDefinition : <None>
AG <Ref> : activityAttributes
A id : TaskActivityId
A name : ActivityName
A start : Start
A duration : Duration
A cost : Cost
A dueDate : Date
A earlyCost : CostModifier
A lateCost : CostModifier
A isPinned : IsPinned
A isBanned : IsBanned
0..* E activityDefinition : <None>
E rootTask : TaskReference
1..* E activity : ActivityReference
E task : TaskReference
0..* E decomposition
AG <Ref> : decompositionAttributes
A type : DecompositionType
A selectedBranch : TaskActivityId
0..* E precedence : PrecedenceArc
0..* E synchronization : SynchronizationArc
0..* E logical : LogicalArc
AG <Ref> : resourceAttributes
A id : ResourceId
A name : ResourceName
A isBanned : IsBanned
0..* E resourceDefinition : <None>
E activity : ActivityReference
1..* E resource : ResourceReference
AG <Ref> : resourceAllocationAttributes
A selectedResource : ResourceId
0..* E resourceAllocation
AG <Ref> : workflowAttributes
A id : WorkflowId
A name : WorkflowName

4. GUI

GUI will show the schedule in two views, in ***Task View*** and ***Resource View***. User will be able to switch between them. If user modifies the schedule in one view and then switches to the second one, the current schedule will be shown.

Both views will consist of two parts, the left one and the right one. The left part will be formed by a hierarchy of tasks and activities in Task View and by a table with data about resources in Resource View. In the right part, there will be a chart with a calendar bar on the top.

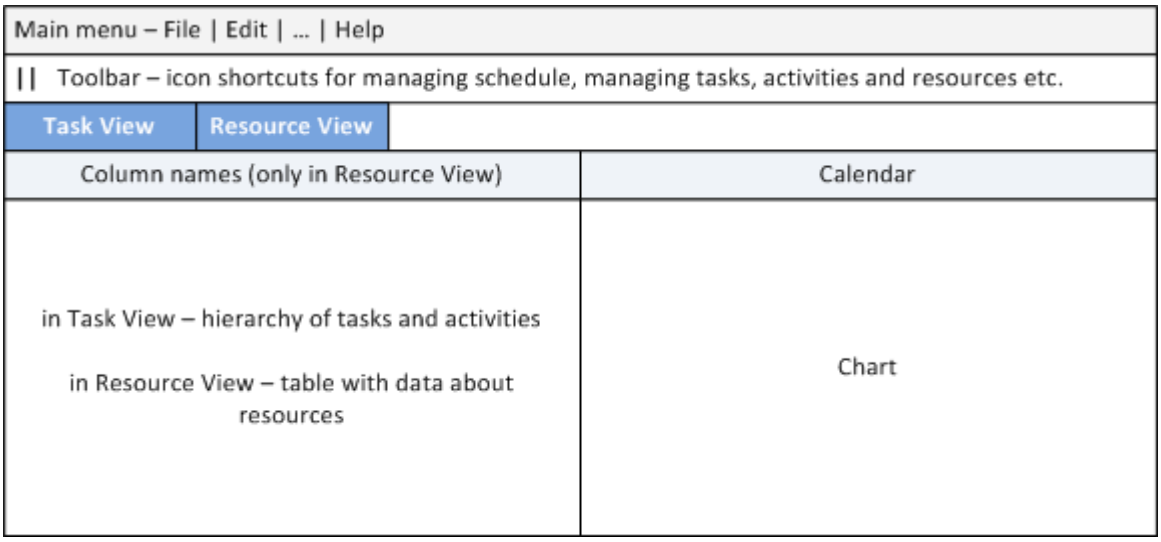


Figure 4.1: GUI appearance (draft)

User will be able to set whether he wants to **synchronize both views** or not (in default setting). If the synchronization will be turn on, it will work this way:

- If user is in Task View
  - If user focuses on an activity which allocates at least one resource and then switches to Resource View, the first resource which is allocated by the selected activity is focused on.
  - If user focuses on a task or an activity which does not allocated any resource and then switches to Resource View, it will be focused on the last resource which has been focused on in Resource View.
- If user is in Resource View
  - If user presses the right mouse button above a shape of an activity and selects an item called “Show in Task View” in a pop-up menu, the system will switch to Task View and focus the activity.

GUI will be user-friendly and have as intuitive control as possible. Therefore, method drag & drop will be often used.

4.1 Task View

Task View will consist of a hierarchy of tasks and activities (see figure 4.2) and a chart.

4.1.1 Schedule visualization

In the Gantt chart on the right, tasks, activities, branches and all-types-of-constraints will be visualized. In the system, there will also be 3 checkboxes (one for precedence constraints, one for synchronizations and one for logical constraints) which will indicate whether target constraints are hidden or visible. In default setting, all-types-of-constraints are visible.

Tasks will be represented by "***tables*** with 2 very short legs", activities by ***rectangles***. Sign "+" ("-") will be added to shapes of collapsed (expanded) tasks.

As mentioned in "6. Negative definition of scope", all-types-of-constraints will by binary.
Visualization of constraints between tasks-activities (see figure 4.2):

- *Precedence(TA1, TA2)* – ***arrow*** from the end of *TA1* to the beginning of *TA2*.
- *Sync(TA1, TA2, SS)* – ***dim line*** between the beginnings of *TA1* and *TA2*. For end-to-end synchronization, it is similar.
- *Logical(TA1, TA2, ⇔)* – ***dim line*** between the beginnings of TA1 and TA2. For *OR* and *XOR*, it is same.
- *Logical(TA1, TA2, ⇒)* – ***dim arrow*** from the beginning of TA1 to the beginning of TA2.

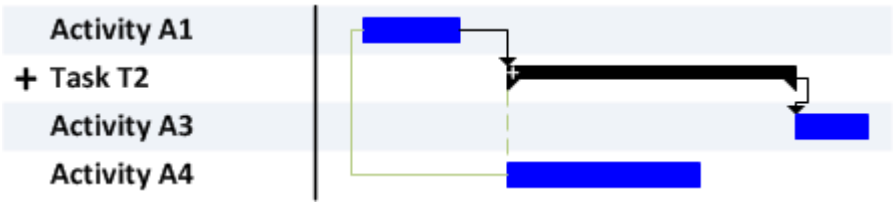


Figure 4.2: Schedule visualization
(*Precedence(A1, T2)*, *Precedence(T2, A3)*, *Sync(T2, A4, SS)*, *Logical(A1, A4, XOR)*)

4.1.2 Expanding and collapsing of tasks

User will be able to expand and collapse tasks. In default view, all tasks will be collapsed (see figure 4.2).

Mouse control:

1. Click on the sign "+" ("-") to expand (collapse) the selected task.

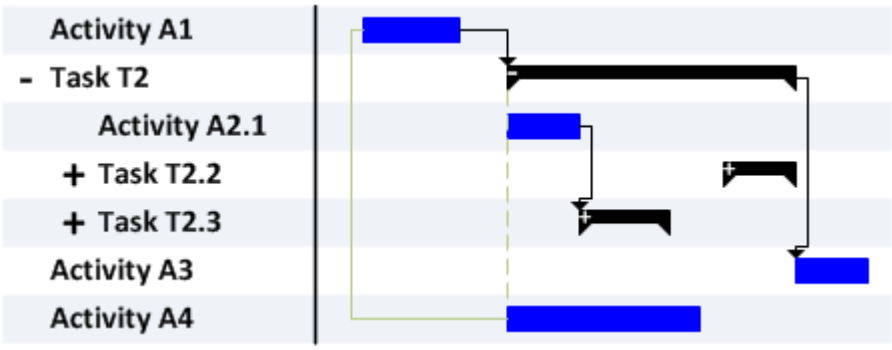


Figure 4.3: Expanded task *T2*

4.1.3 Adding and removing

It will not be possible to add or remove tasks-activities, any of all-types-of-constraints and branches. The reason is that if any of them was added or removed, schedule would not be compatible with its workflows.

4.1.3.1 Pins

It will be possible to ***pin*** or unpin just an activity. It will not be possible to pin a task.

Pin will be useful when user will want to say that it will not be possible to shift an activity anyhow during running Optimizer and Rescheduling algorithm ("5. Optimizer and Rescheduling algorithm"). "Anyhow" means it will not be possible to ***modify time data*** of the activity and shift the activity to ***alternative resources***. However, user will manipulate with pinned activities as ordinary ones, i. e. user will be able to move a pinned activity etc.

Feasibility of the schedule will be check just after these operations:

- pinning an activity,
- modification of time data of a pinned activity (except reducing a duration),
- shifting a pinned activity to an alternative resource.

As a result of this checking, the schedule will be either feasible or not. If not, user will choose whether the activity will be unpinned or the operation will be undone. If user swap branches and there are pinned activities in ex-main one, it does not matter, because those pinned activities will not be in the result schedule at all.

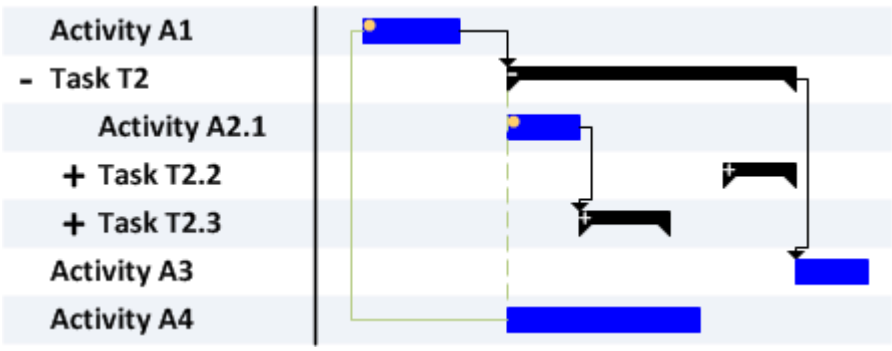


Figure 4.4: Pinned activities *A1* and *A2.1*

4.1.4 Editing of constraints

It will not be possible to edit any of all-types-of-constraints anyhow. The reason is the same as in "4.1.3 Adding and removing" – the compatibility with workflows.

4.1.5 Editing of time data of tasks and activities

User will be able to move tasks and edit time data of activities. Resizing is not allowed for tasks as a result of a concept of task (see "1.3 Fundamental terms").

During dragging and resizing (only for activities) of a shape of a task-activity using the mouse, the shape will be redrawn immediately. However, all constraints in a schedule will be hidden during that operation and they will be shown after releasing a mouse button (see figure 4.5).

Figure 4.5: Activity *A1* is being resizing. Its start time is being moved to the right.
(Dotted shape is the previous location of the activity.)

4.1.5.1 Moving tasks

User will be able to move a task in time.

All tasks-activities in a hierarchy (descendants) of the task will be moved too and distances among them will be the same (see figure 4.6). During dragging a shape of the task, no descendants will be redrawn because of a speed. That will be done after the releasing of a mouse button.

Mouse control:

1. Place the mouse above a shape of the selected task.
2. Click on the left button of the mouse and hold it.
3. Drag the shape where you want.
4. Release the mouse button.

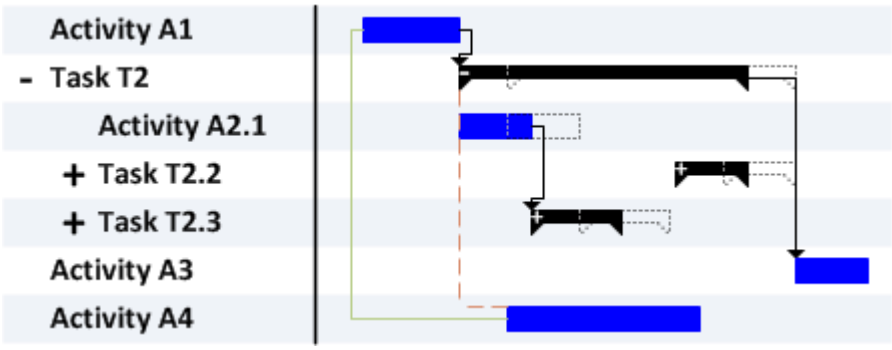


Figure 4.6: Task *T2* has been moved to the left and then a mouse button has been released.
(Dotted shapes are the previous location of tasks and activities. *Sync(T2, A4, SS)* has been broken.)

4.1.5.2 Moving activities

User will be able to move an activity in time.

Mouse control is same as in “4.1.5.1 Moving tasks”.

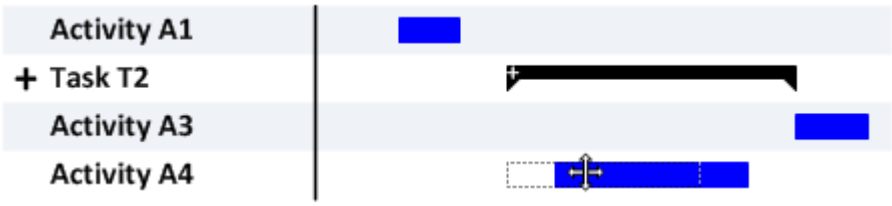


Figure 4.7: Activity *A4* is being moved to the right.
(Dotted shape is the previous location of the activity.)

4.1.5.3 Changing start time, finish time and duration of activities

User will be able to change a start time, finish time and duration of an activity (see figure 4.5).

Mouse control:

1. Place the mouse above the left (right) edge of a shape of the selected activity.
2. Click on the left button of the mouse and hold it.
3. Drag the beginning (end) of the shape where you want.
4. Release the mouse button.

4.1.6 Alternative branches

Data about branches will be in the input data. It will not be possible to add, remove or edit them anyhow. User will only be able to:

- show alternative branches of the selected main one,
- make selected branch the main one,
- ban selected branch.

4.1.6.1 Visualization

Alternative branches will be dim (see figure 4.8). In default view, they will not be visible.

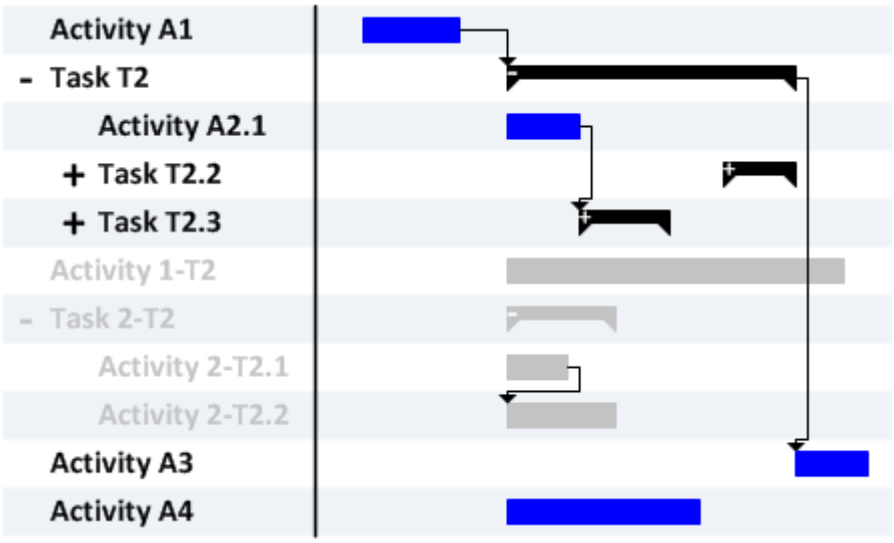


Figure 4.8: Shown alternative branches

In figure 4.8, all branches, i. e. tasks *T2* and *2-T2* and activity *1-T2*, start at the same time. The reason is that neither activity *1-T2* nor descendants of task *2-T2* are scheduled – none of their time data are set – because Optimizer cannot schedule alternative branches. For better transparency, shapes of alternative branches and all their descendants will start at the same time as the main branch. It will be possible that duration of branches will be different (see figure 4.8).

4.1.6.2 Changing main branches

User will be able to swap an alternative branch and the main one.

Mouse control:

1. Place the mouse above a shape of the selected alternative branch.
2. Click on the left button of the mouse and hold it.
3. Drag the shape above a shape of the main branch.
4. Release the mouse button.

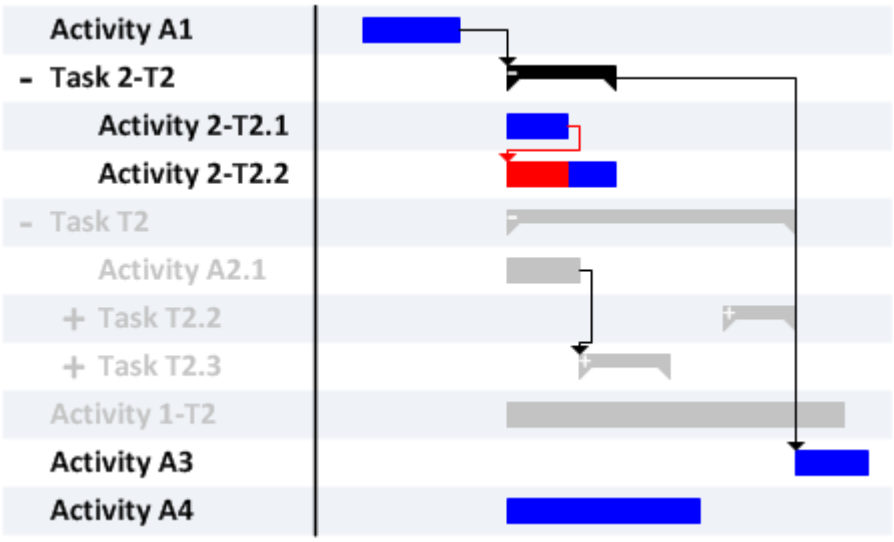


Figure 4.9: Branches formed by tasks *T2* and *2-T2* has been swapped.
Therefore, *Precedence(2-T2.1, 2-T2.2)* is broken.

If the main branch become the alternative one, start time of descendants will not be changed – not be moved to the start of the branch.

4.1.6.3 Breaking logical constraints

In schedule, there will be logical constraints. They could be broken just by swapping branches, not by any other allowed operation. Highlighting of them is mentioned in “4.1.7.2 Broken constraints”.

4.1.6.4 Banning of branches

User will be able to ban an alternative branch. This will be useful, when user will want to run Optimizer and want to avoid a result schedule which will contain the selected branch. As a result, if user wants to have a particular branch in the result schedule, he will have to banned all alternative ones.

Whole banned branches will darken.

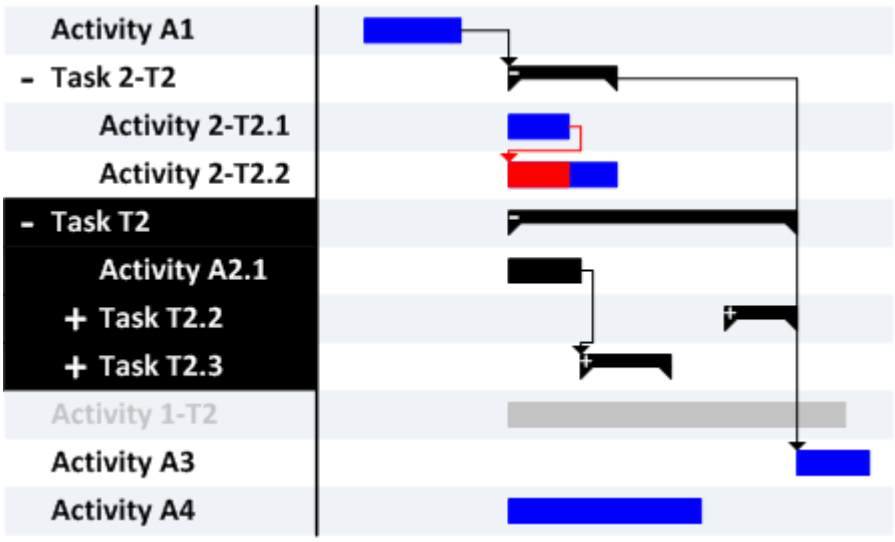


Figure 4.10: Branch formed by task *T2* is banned.

4.1.7 Highlighting

In Schedule Visualizer, highlighting will be implemented. Among others, workflows and broken constraints will be highlighted.

4.1.7.1 Workflows

If user wants to know which tasks-activities are in the same workflow as the selected one, he will click on a shape of the task-activity.

Mouse control:

1. Click on a shape of the selected task-activity.

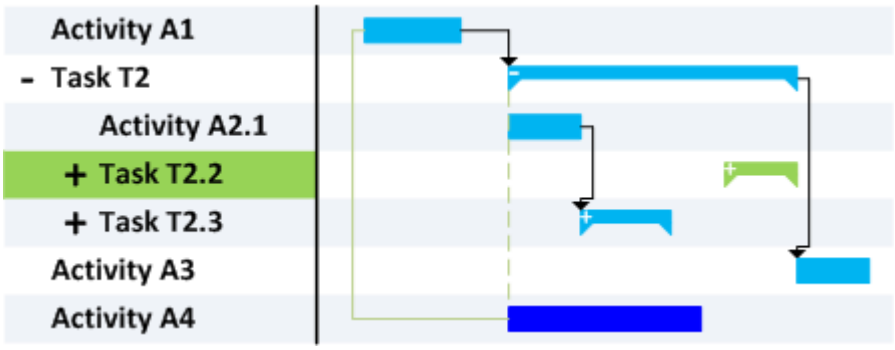


Figure 4.11: Task *T2.2* has been focused.

4.1.7.2 Broken constraints

All broken constraints will be highlighted whether both tasks-activities are in the result schedule.

Logical constraints are exceptions:

- *Logical(TA1, TA2, ⇔)* – highlighted whether *TA1* is in the result schedule and *TA2* not. And vice versa.
- *Logical(TA1, TA2, ⇒)* – highlighted whether *TA1* is in the result schedule and *TA2* not.
- *Logical(TA1, TA2, OR)* – never broken.
- *Logical(TA1, TA2, XOR)* – never broken.

Highlighted shapes according to a type of a constraint:

- *Precedence(TA1, TA2)* – arrow and overflowed part of the shape of TA2 (activity A3 in figure 4.12).
- *Sync(TA1, TA2, SS)* – line. For *EE* type, it is same.
- *Logical(TA1, TA2, ⇔)* – line.
- *Logical(TA1, TA2, ⇒)* – arrow.

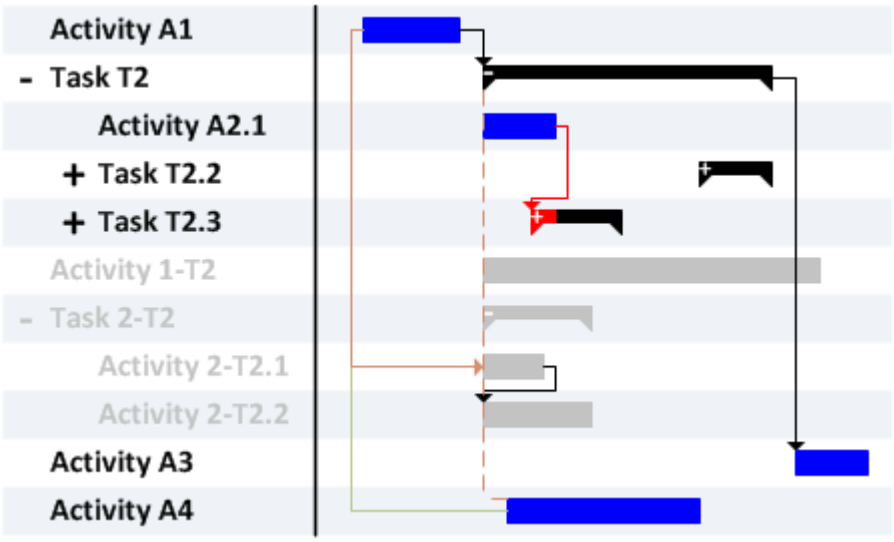


Figure 4.12: Broken constraints (*Precedence(A2.1, T2.3)*, *Sync(T2, A4, SS)*, *Logical(A1, 2-T2.1, ⇒)*). It is needed to make a branch *2-T2* the main one to correct the logical constraint.

4.2 Resource View

Resource View will consist of table with data about resources (e. g. resource name, resource type) and a chart.

4.2.1 Chart

In the chart, there will be shown just activities which will have allocated at least one resource. It will be possible that activities will allocate more than one resource (activity *A1* in figure 4.13).

Neither any constraints, nor alternative branches will be visualized in the chart.

Activities will be shown in a line one after another except those ones which will break just-unary-resources constraint (see figure 4.13).

Figure 4.13: Activities *A2* and *A3* are breaking just-unary-resources constraint.

4.2.1.1 Filter

In the schedule, there might be a lot of resources. Therefore, it could be quite uncomfortable to make

changes in the schedule, such as moving selected shape of an activity to an alternative resource.

As a result of this fact, it will be possible to filtered resources.

Two filters will be available:

- ***ALLOC*** – all resources allocated by the selected activity (see figure 4.14),
- ***ALTER*** – all alternative resources for the selected shape of activity (see figure 4.15).

In both filters, user will be able to do same thing as in default view.

Mouse control:

1. Place the mouse above a shape of the selected activity.
2. Click on the right mouse button.
3. For *ALLOC* (*ALTER*), select an item called "Filter – allocated resources" ("Filter – alternative resources") in a pop-up menu.

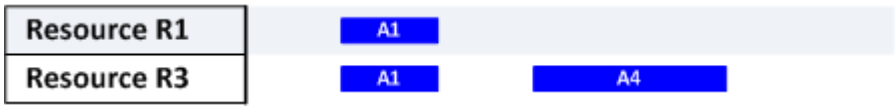


Figure 4.14: *ALLOC* used on activity *A1*

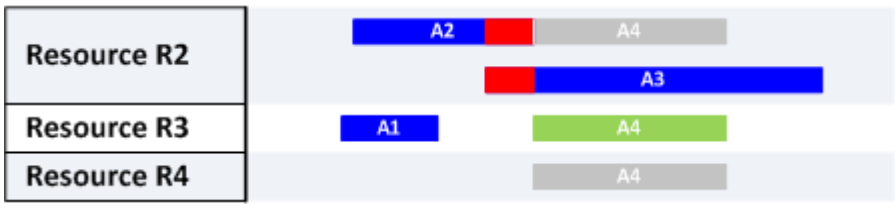


Figure 4.15: *ALTER* used on the highlighted shape of activity *A4*

4.2.2 Adding, removing and banning of resources

It will not be possible to add a resource. The reason is that if it was added, it would not carry out any activity, because alternative resources are determined in Optimizer (see "4.2.4 Alternative resources").

Removing of a resource will not be possible too. However, user will be able to banned a resource. Activities which will allocated that resource will darken (see figure 4.16).

Figure 4.16: Resource *R3* has been banned.

4.2.3 Editing of time data of activities

User will be able to edit time data of activities.

If modified activity and other activities on the same resource break just-unary-resources constraint during dragging or resizing of a shape of the modified activity, line of that resource will be enlarged and activities will be immediately redrawn in two or more levels and vice versa (see figure 4.17).

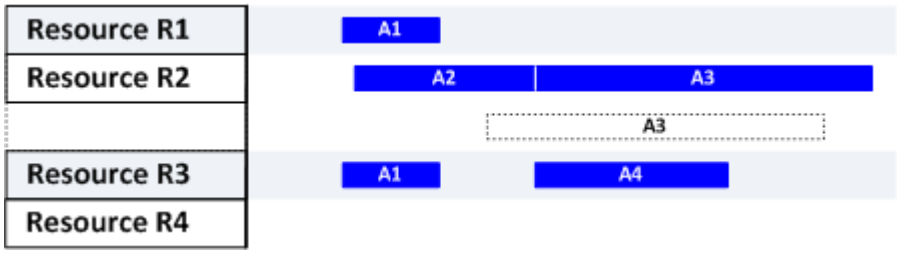


Figure 4.17: Just-unary-resources constraint between activities *A2* and *A3* has been corrected.
(Dotted shape is the previous location of a shape of activity *A3*.)

If activity allocates more than one resource, all shapes of this activity will be dragged (resized) during dragging (resizing) of the focused shape (see figure 4.18). After releasing a mouse button, all shapes of the activity will have the same time data.

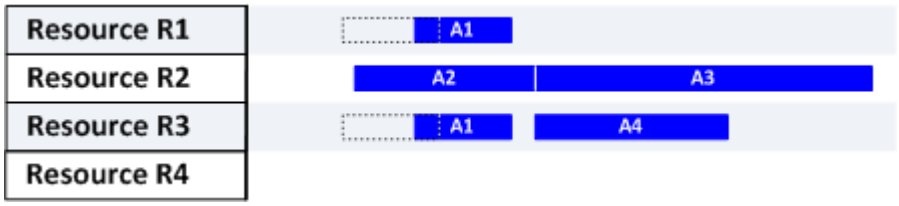


Figure 4.18: Activity *A1* is moving to the right.
(Dotted shapes are the previous location of shapes of activity *A1*.)

4.2.3.1 Moving activities

There will be differences between Task and Resource View only in mouse control.

It will not be possible to shift a shape only few pixels, because of "4.2.4.2 Moving activity to alternative resource". Both operations are based on dragging a shape:

- moving in time – horizontally,
- moving to an alternative resource – vertically.

Thus, if user want to just move an an activity to an alternative resource, not move in time, he will have to avoid any move horizontally. Therefore, it will be set a limit of move tolerance. It means that user will also be able to move an activity in time and move the activity to an alternative resource at the same time.

Mouse control:

1. Place the mouse above any shape of the selected activity.
2. Click on the left button of the mouse and hold it.

3. Drag the shape where you want.
4. Release the mouse button.

4.2.3.2 Changing start time, finish time and duration

Same as “4.1.5.3 Changing start time, finish time and duration”.

4.2.4 Alternative resources

In the input data, there will be activities with information about allocated resources. There will be also information about alternatives to each resource. User will be able to swap an allocated resource with one of its alternative of an activity.

4.2.4.1 Visualization

Alternative resources will be dim (see figure 4.15). They will be visible just during dragging a shape of an activity.

4.2.4.2 Moving activity to alternative resource

Details about dragging a shape of an activity are in “4.2.3.1 Moving activities”.

Mouse control:

1. Place the mouse above a shape of the selected activity.
2. Click on the left button of the mouse and hold it.
3. Drag the shape above an area of the alternative resource.
4. Release the mouse button.

4.2.5 Highlighting of broken constraints

There will be highlighted only shapes or parts of shapes of activities which will break just-unary-resources or precedence constraints (see figure 4.13).

4.3 Undo/Redo operations

Undo/Redo operations will be available. However, running Optimizer or Rescheduling algorithm will clear the buffer of changes and it will not be possible to use Undo/Redo operations on optimizing and rescheduling.

4.4 Calendar

The module will work with a calendar which will be a separated module.

In Schedule Visualizer, the internal representation of time data will be in an abstract unit and all calculations will be made in it. However, user will specify how large this unit is and what the start time is. The start time will be represented as a timestamp. The allowed smallest unit will be a second.

When time data has to be shown, they will be converted from abstract unit to the standard time unit.

4.5 Zooming

It will be possible to zoom in and zoom out both charts, in Task and Resource View. Zooming will be just horizontal.

Mouse control:

1. Place the mouse above the calendar bar.
2. Click on the left button of the mouse and hold it.
3. To zoom in (out), drag the mouse to the right (left) as far as you want.
4. Release the mouse button.

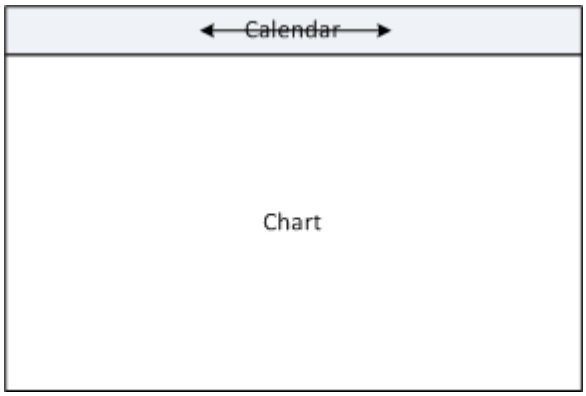


Figure 4.19: Zooming

During the dragging the mouse, units in the calendar bar will be updated immediately. It means that if the units are too small, they will be enlarged (e. g. days → weeks) and vice versa. In the chart, shapes of tasks-activities will take the current ratio scale of the chart into account.

The chart will be redrawn just after releasing the mouse button, not during dragging. It means that the chart will not be zoomed immediately.

5. Optimizer and Rescheduling algorithm

It will be possible to run Optimizer from Schedule Visualizer. Optimizer will acquire the current schedule as an input and optimize it. Optimizer will take pins into account.

In addition, there will be Rescheduling algorithm in Schedule Visualizer. Its goal will be to correct broken constraints in the current schedule. Thus, the result will be a feasible schedule. Rescheduling algorithm could only use operation of shifting activity, no other.

Among all feasible schedules, the algorithm should select the one which is the most similar to the original schedule. However, that is NP-complete problem. Therefore, only approximate solution will be find.

6. Negative definition of scope

There is a negative definition of functionality which will be implemented in the module. It will help with limiting scope of the module.

Tasks and Activities	Tasks-activities will be sons of at most one task
	Branch will consist of just one task-activity
	No loop among precedence constraints
	No common temporal constraint
	No temporal constraint with **minimum** and **maximum** distance between tasks-activities
	No activity **setup time** (= minimum distance between two activities – depends on those activities)
	No **time window** (= no calendar constraints)
	Just binary precedence constraints
	Just binary synchronizations (*SS*, *EE*)
	Just binary logical constraints (⇔, ⇒, *OR*, *XOR*)
	No adding and removing task-activity
Resources	Just **unary** resources
	No adding resource