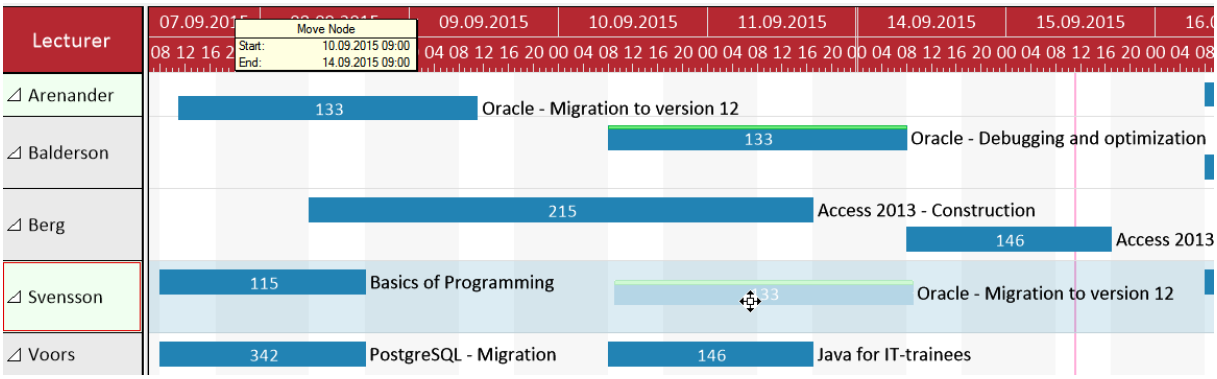
WHITEPAPER

Interaction Events: Context-Sensitive Decision Support During Drag & Drop Actions



NETRONIC

Contents

Introduction

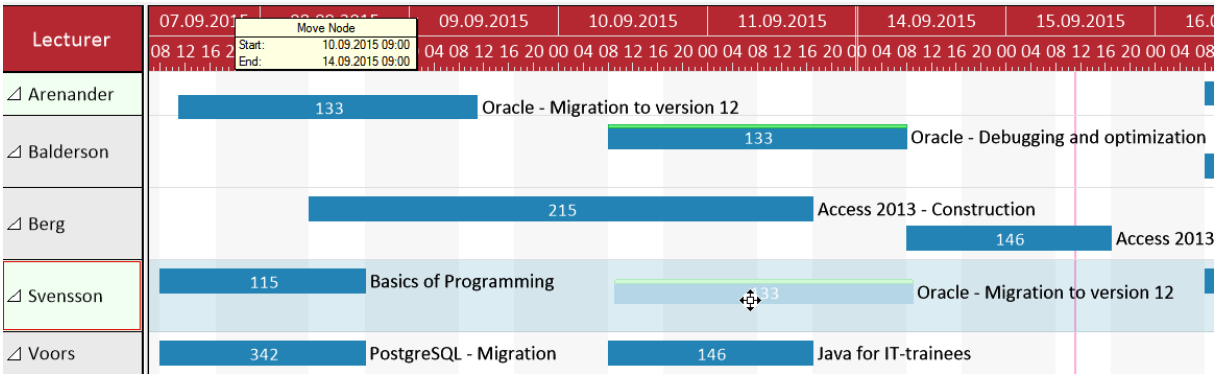
This whitepaper is about the new/enhanced events occurring during drag & drop interactions coming with VARCHART XGantt 5.0 and offering various options of receiving and processing information on the object during a mouse interaction while live update is enabled.

In this context, interaction does not only mean shifting an operation by drag & drop. It is meant in a sense that the Gantt diagram provides context-sensitive decision support information during the drag & drop interactions.

Target-oriented interaction

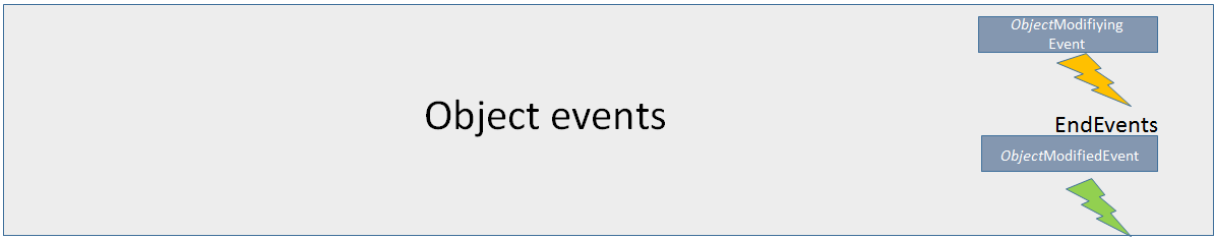
Interaction means changing of plan. An interaction starts with pressing a mouse key (typically the left) and finishes with releasing it. The crucial aspect is for the user to get all decision-relevant information on starting the interaction and during the mouse movement to have a sound basis for estimating the consequences of the modifications. So, if the planner has to change the existing plan, the interaction supports him in making the correct decision for the repositioning.

The following screenshot of a course booking plan shows an example of how the new functions help to effectively design interactions. Among others, the plan is designed in such a way that when a course (node) is being moved, the possible results in terms of which teachers are qualified for the theme and conflicts concerning the room occupancy are shown immediately.



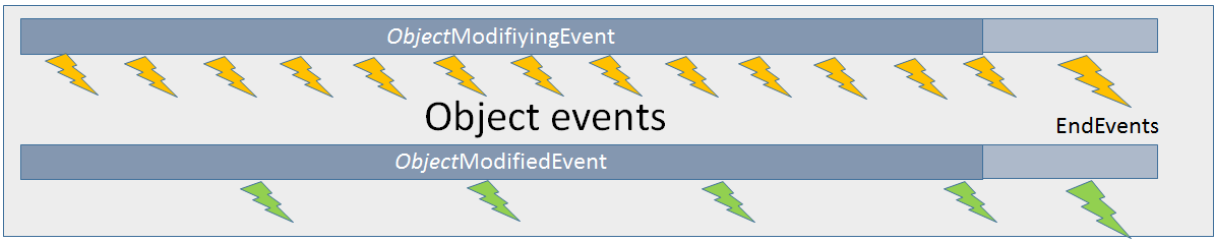
Default behavior up to now

In the default behavior, no feedback is given as to the status of the concerned object. Only when the mouse key is released, information on the old (before pressing the mouse key) and the new (after having released the mouse key) status is given by an ***Object*Modifying** event. In addition, an ***Object*Modified** event indicates that the operation is finished internally:



The new events

With VARCHART XGantt 5.0 some new events - the **Interaction** events - were added that not only accompany and describe the interaction. Moreover, the object events' time of calling and frequency were modified. Due to this modifications and enhancements, it is now possible to retrieve information on the changing object already during the interaction and not only after having finished it.



Interactions involved

We will explain events that describe the process of an interaction in VARCHART XGantt and the objects involved in greater detail, i.e. “Drag(Drop)” events during interactions that

- start with pressing the left mouse key at an object
- carry out movements with the mouse key being pressed
- end with releasing the left mouse key
- are treated in the course of “Live Update”

Terminology

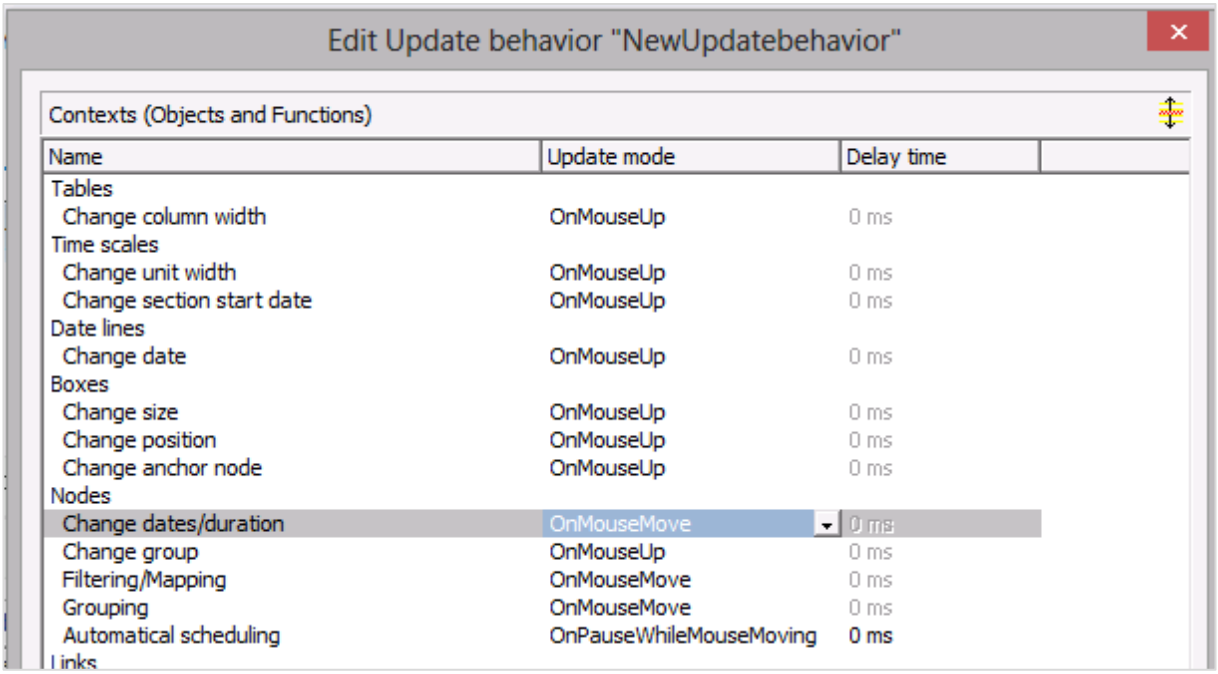
For a better understanding we’d like to further explain some terms that are used in the text.

Object Events

Object-Events, such as **VcDateLineModifying**, VcDateLineModified, **VcNode-Modifying**, **VcNodeModified** etc., are events, that, according to the practice already known up to now, are thrown at the end of an action during the addressed interactions.

Live Update

Live update means that a “Drag Drop” action causes a “What if the object was updated here?” scenario to be shown permanently, this resulting in processing different contexts, such as direct or dependent functionalities during an interaction, at different times. If, for instance, a node is being moved, this results in modifying various data and the node’s position, this in turn resulting in modifying the histogram curves or the summary bars, for instance. Depending on the settings in the **Live Update** dialog, the modifications will either come into effect at once or after hovering with the mouse a time span to be specified or at the end of the action on releasing the mouse key.



Example: *What does the updates look like if the update behavior „OnMouseMove" is selected for the moving of nodes?*

- Immediate effects on the node:
  - every date value of the node
  - filters are evaluated, thus causing other colors, e.g., to appear in the table area
  - summary bars
  - histogram curves

- Modifications after a waiting period (500 ms)
  - positioning the node in a group, for instance
  - optimization with corresponding layout of the node order

Only updates that are necessary and meaningful in the total context of the action should be carried out, because otherwise the chart would become too restless.

InInteraction Events

From VARCHART XGantt 5.0 SR3 onward, object events can be processed already while the interaction is running, this objects being called **InInteraction** events.

Important: *Be sure to enable the InInteraction events beforehand, either by the property **VcGantt.InInteractionEventsEnabled** = true or on the **General** property page.*

*Please note that when talking about interactions with nodes in the real mode, we will call the display object **Real (node)** and the data element in the chart **Chart node**. The chart node is not visible during the live interaction in the chart area because it will be replaced temporarily by the real node there, its presence, however, affecting the diagram in terms of ribbon height, optimization, colors in the table area etc.*

This way, according information on the normal objects are delivered during the interaction matching with the displayed phantom or real node.

When a node is moved, every snapping into place of the node (depending on its time unit and increment) causes a **VcNodeModifying** to be thrown (yellow lightnings). The real node shows the possible position and the possible layout and describes this status by the **VcNodeModifying** event. The node (e.Node) being passed in the event args, represents the real node's status.

Important: *This is why queries for properties of the chart nodes don't make sense or are not possible. Only the properties get/setDataField, AllData, ID can be retrieved or set.*

If, depending on the selected updating context, e.g. "On pause while mouse moving", the real object is updated, this will be indicated by the **Modified** event (green lightning). This can but doesn't have to happen at the same time as the **Modifying** events.

If a node is moved while the updating behavior "On mouse move" is selected, both events will appear at the same time.

To sum up the facts:

- If a node is moved, its modification, indicated by the real node, will be permanently described by the **VcNodeModifying** event**.**
- Modifications of the chart node are indicated by the **VcNodeModified** event.

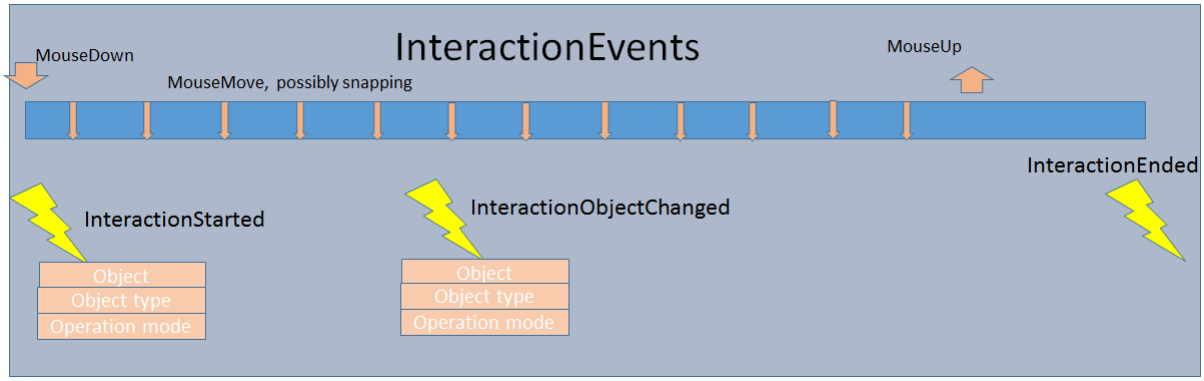
- When the interaction is finished, upon releasing the mouse key, the concluding event pair, consisting of the **VcNodeModifying** and the **VcNodeModified** event are provided.

The concerned objects in events that use real nodes are the real objects.

In the last **VcNodeModifying** event, the chart node (as opposed to the previous **VcNodeModifying** events) with the values that were last set during the interaction is provided, i.e. the status at the time of the last small green lightning. **e.OldNode** of the EventArgs describes the status at the beginning of the action. This way, the start and end status of the interaction can be compared.

As always, the chart node is available in the last **VcNodeModified** event and all internal processes are finished.

Interaction Events



As described above, the object events are now thrown during and at the end of an interaction. The signature of the event handler, e.g. of the **VcNodeModifying** event don't differ there. But how to recognize whether the event has been thrown during or at the end of an interaction?

This could be important, because not every modification resulting from a mouse movement, for instance, is to be stored to a data base: This would cause too much time-consuming effort. Of course, the data shall only be stored after the action was finished.

This problem can be solved now by some new events that accompany and describe the interaction and can be evaluated in the object events during the interaction.

As soon as the left mouse key is pressed, the **VcInteractionStarted** event delivers information on the object the mouse key is standing on (object and object type) and

on what is happening with the object. Everything that is needed for the interaction can be prepared.

***Tip**: The update behavior can also be switched object- and context-specific here. In an extreme case, one could have one node react completely dynamical and another one with a blue phantom frame. Moreover, an according setting (**InInteractionEventsEnabled**) allows for an individual decision about whether the object events are to come also during the interaction or not.*

Example: Node

By

- Object: NodeObject
- Type: vcObjTypeNodeInDiagram
- OperationMode: vcIIMMoveNode

upon pressing the left mouse key, the **VcInteractionStarted** event shows that the moving of a node in the chart has started.

Information or elements that ought to accompany the interaction can be initialized here.

Creating Objects

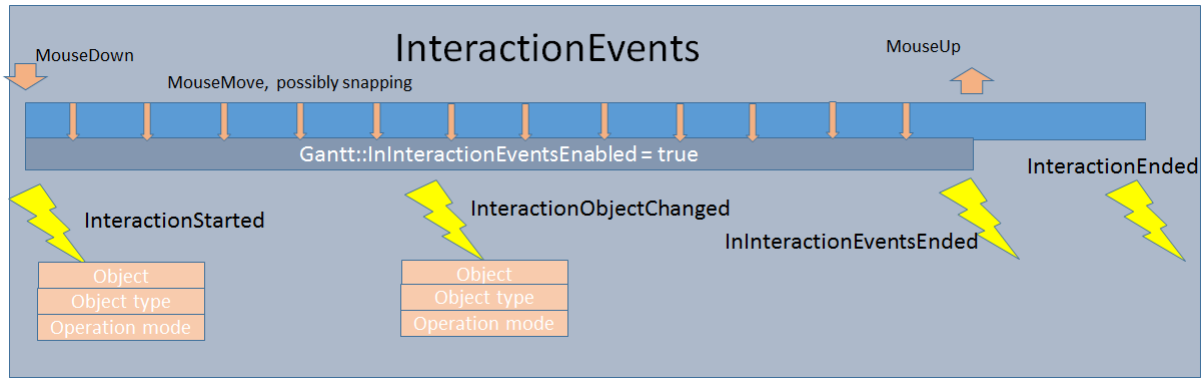
In some interactions, there's no object available initially, e.g. when creating nodes or boxes. In this case, the event **VcInteractionObjectChanged** comes as soon as the object was created internally, being the real chart node where nodes are concerned.

The end of the action is indicated by the **VcInteractionEnded** event. Every additional element having been used during the interaction can be removed here.

When new objects are created with Interaction events, the process is as follows:

- VcInteractionStarted
- VcInteractionObjectChanged
- Modifying/Modified Events, showing modifications when creating an element
- Creating und Created Events
- VcInteractionEnded

InInteraction Events activated during the interaction



When the **Interaction** events are also enabled during the interaction (**vcGantt.InInteractionEventsEnabled** = **true**), there will be an additional event indicating the end of these events upon releasing the mouse key: **VcInInteractionEventsEnded**

This makes it easy to differentiate the object events being thrown during the interaction from those that are thrown at the end of the interaction. If this event is thrown, the next object event will be the concluding event.

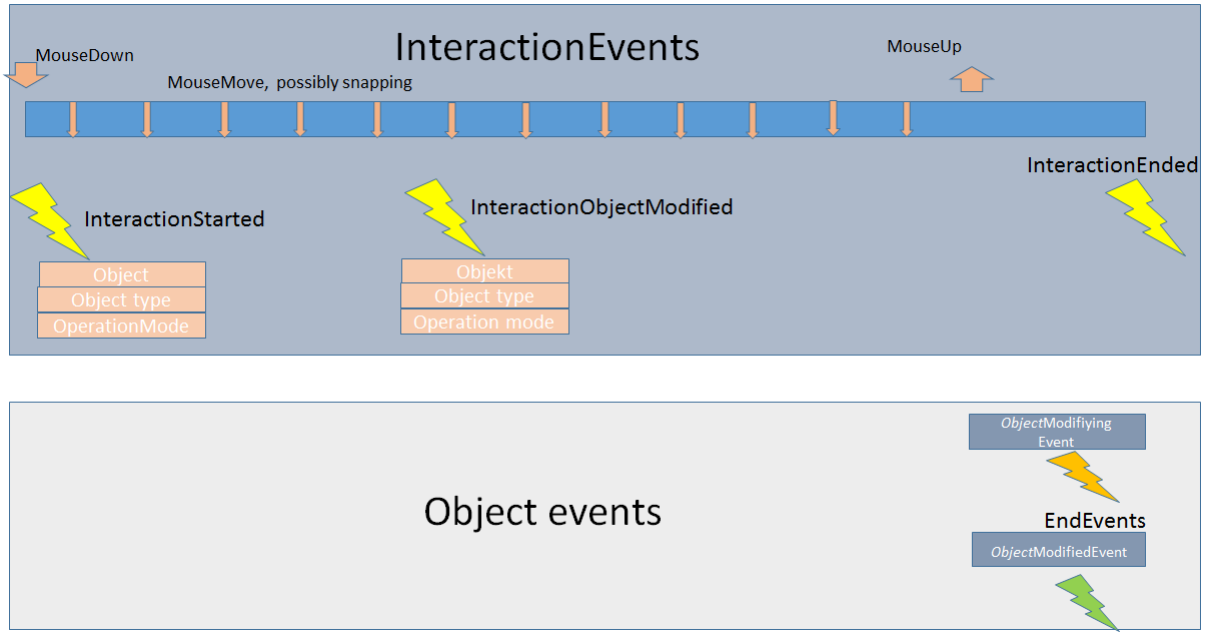
Possible scenarios

In other words, there are two possible conditions when using Interaction events.

Controlling an interaction with:

- **InInteraction** Events being switched off
- **InInteraction** Events being switched on

Cooperation with the events of the involved objects while the InInteraction events are **deactivated:**



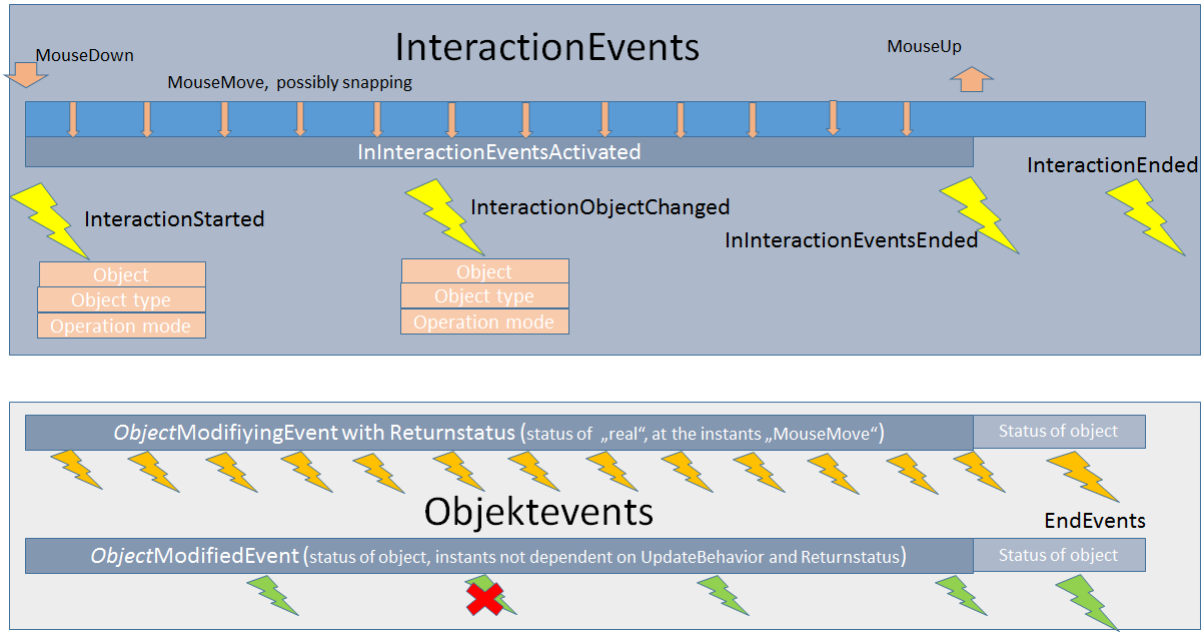
The screenshot shows how the Interaction (yellow lightnings) and the object events (ochre and green lightning) cooperate when InInteraction events are switched off (**vcGantt.InInteractionEventsEnabled = false**):

The interaction is started which is indicated by the **InteractionStarted** event.

When releasing the mouse key, the object events appear first, e.g. **VcNodeModifying** and **VcNodeModified** with a node. In other words this is the old behavior regarding object events so that existing code in the object events doesn't have to be modified if the InInteraction events are not used.

The end of the interaction is indicated by the **VcInteractionEnded** event.

Cooperation with the events of the involved objects while the InInteraction events are **activated**:



If the **InInteraction** events are used, the following events appear:

- **VcInteractionStarted** upon pressing the left mouse key
- **Modifying** and **Modified** events while the mouse is moved
- **VcInInteractionEventsEnded** and afterwards the finishing object events when the left mouse key is released
- **VcInteractionEnded** to indicate the end of the interaction.

Example: Moving a node:

The interaction starts when the left mouse key is pressed while the mouse cursor is at a node. The event **VcInteractionStarted** appears.

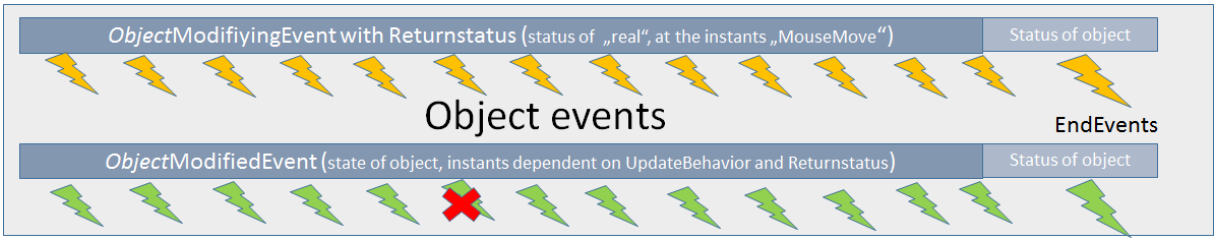
The events appearing upon moving the mouse indicate the status of the real node (**VcNodeModifying)** and while updating (**VcNodeModified**) the chart node.

When the mouse key is released, the **VcInInteractionEventsEnded** event appears

The object events **VcNodeModifying** and **VcNodeModified** indicate the status of the chart node at the end of the interaction.

The last to appear is the **VcInteractionEnded** event.

Example: Behavior of the object events when the node update behavior „On mouse move“ is set



Since the **VcNodeModifying** event allows for the EventReturnstatus (*e.ReturnStatus*) to be modified, this can now also be done during the interaction.

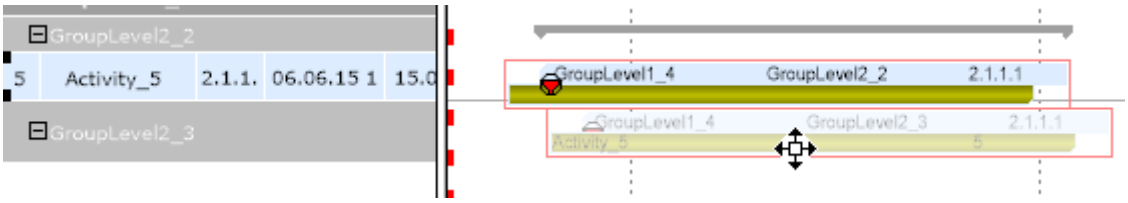
So, if *e.ReturnStatus = ReturnStatusFalse* indicates that the provided data are not “valid”, the object in the chart will not be refreshed with the next possible update and the according **VcNodeModified** event will not be thrown.

This is visualized by the object remaining at its old place and the current position being still indicated by the phantom.

The status of objects visualized by reals (currently only nodes and node boxes) is indicated as follows:

The current position is visualized by a brightened real, the values of which also still being provided in the events.

The last valid status, i.e. the last one not returning *ReturnStatusFalse* as **e.ReturnStatus**, is indicated by another real, that quasi “gets stuck” there; this way both pieces of information are being visualized.



At the node, the values of the last valid status, i.e. that of the stuck real, correspond to the *e.OldNode* in the **VcNodeModifying**-Event

If the last **VcNodeModifying** event before the **VcInInteractionEventsEnded** was finished with *ReturnStatusFalse,* the last valid state will be provided in the End events.

There it can be decided whether to accept this state or not. If in the End event *ReturnStatusFalse* is set, the original start status will be restored.

Practical Tip: We recommend to create an „accompanying **InteractionInfo**" object that provide the needed information on the interaction in the events and can be evaluated accordingly.

Conclusion

A well-designed Gantt chart should not be overloaded with information while at the same time it has to provide the planner with every crucial detail needed for decision-making.

The enhanced as well as the newly added events now enable the planner to master the respective demands to his plan by way of intelligently designed interactions, thus providing quick and effective decisions.

More resources

The ways of applying the interaction events are as manifold as is each Gantt application. Since this whitepaper can't deal with every possible option, further information is provided as shown below:

- Video showing three possible interaction variations:
  1. Visualize available time spans while the node is being dragged.
  2. Highlight time spans causing planning conflicts and show involved bars.
  3. Show time spans with planning conflicts; planning is allowed in spite of conflict.
- Blog on „Intelligent Interactions"
- Sample „Car Rental" of the sample collection that is included in the delivery of VARCHART XGantt.

Free trial version: Empower your planning application

Get a free trial of VARCHART XGantt and receive an impression of the control's capabilities.