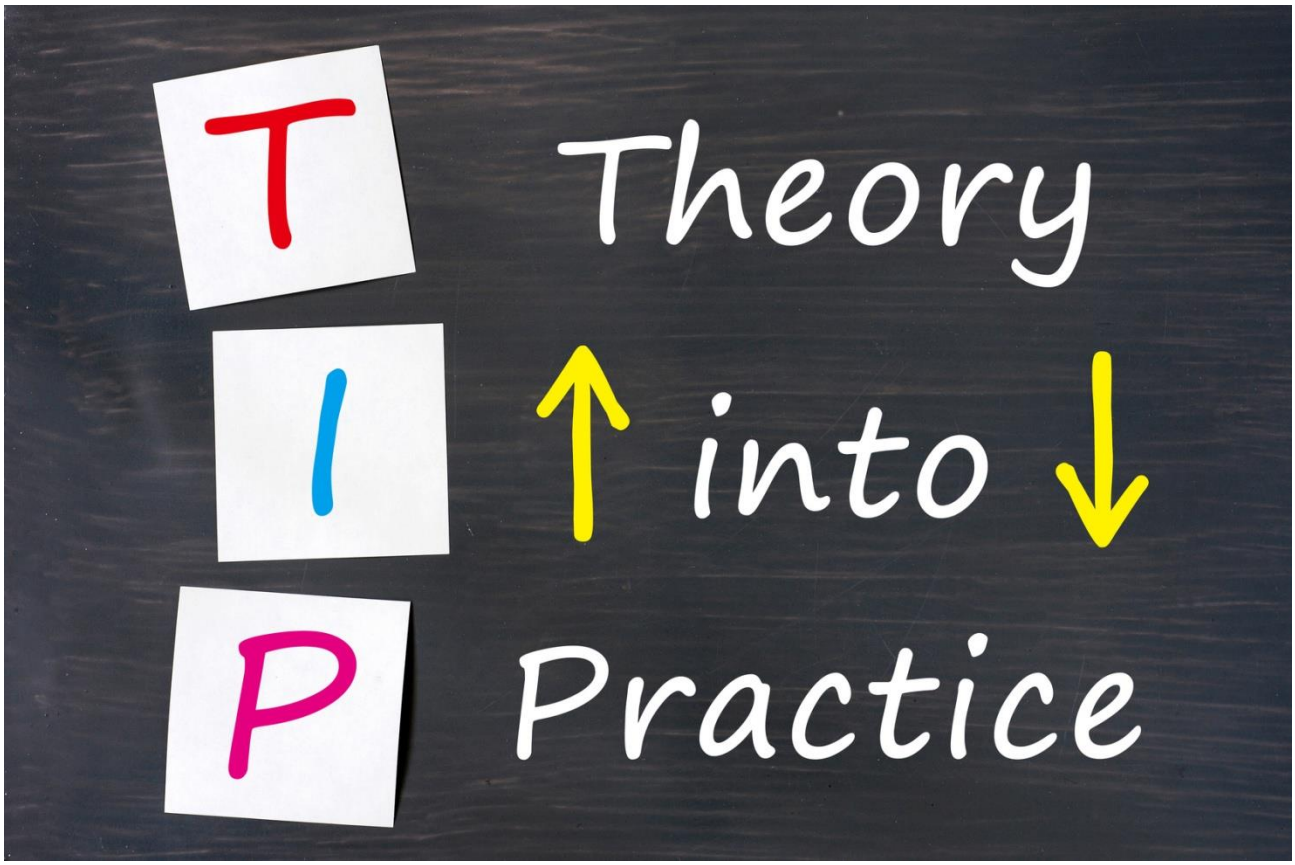
EBOOK

11 Best Practice Tips for Advanced Gantt Visualization



NETRONIC

Table of Contents

Why Tips & Tricks?

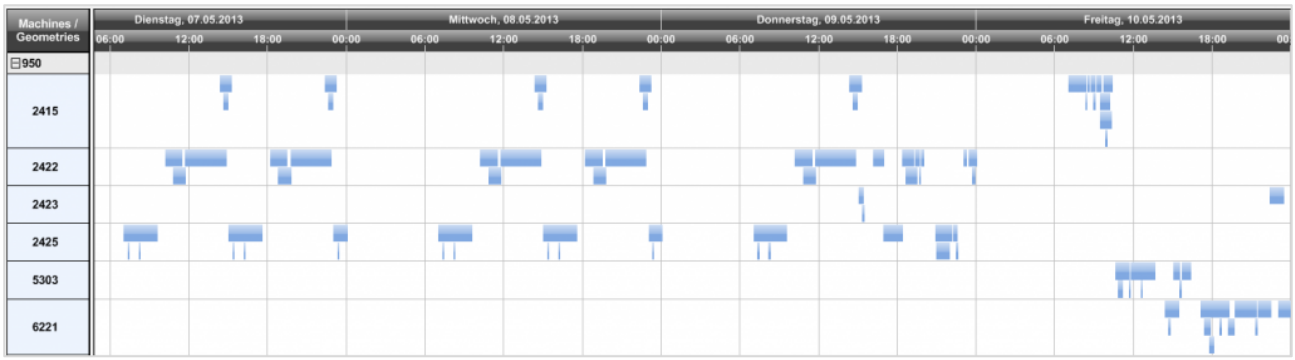
Our Gantt chart software controls in the NET, ASP.NET and ActiveX Editions already offer a combination of profound functionality and flexibility: They comprise every function that is expected of a powerful up-to-date Gantt chart. At the same time, due to its highly flexible configuring options, VARCHART XGantt integrates seamlessly into your application.

So you may ask yourself: Why do I need Tips & Tricks?

Did you ever get the feeling that in spite of the sophisticated features VARCHART XGantt offers, there's still one little detail missing to make your chart perfect? This is why we present you our collection of Tips & Tricks, offering valuable suggestions for working with your VARCHART XGantt application. We're sure you will detect more useful details than only the one you were looking for.

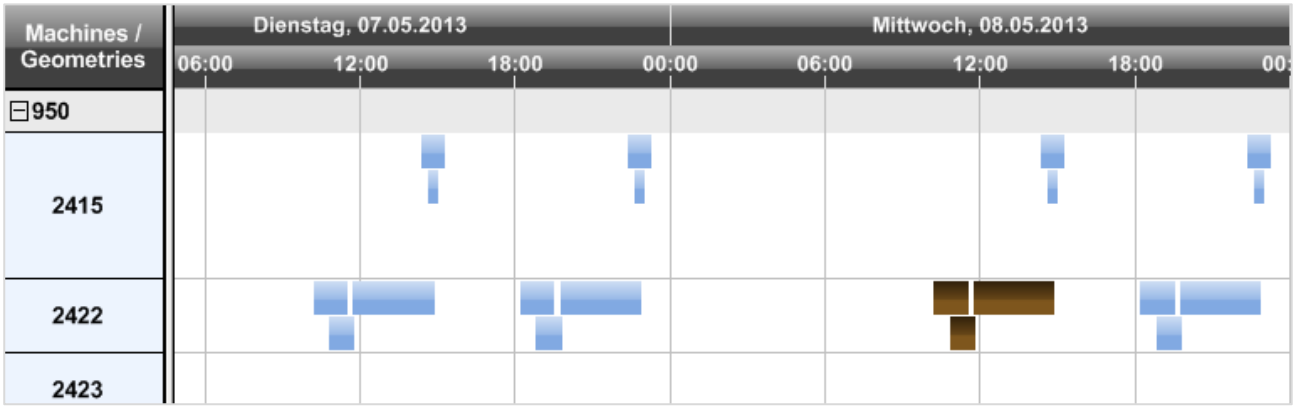
1. Visualize Tasks With Short Duration

Especially when production scheduling is concerned, Gantt charts are a proven tool to manage resources with time-related tasks as efficiently as possible. In many production environments, these tasks tend to have only short processing times so that the Gantt chart has to visualize many of them one after another – showing as much details as possible. At the same time, the planner needs a high-level overview of the manufacturing process. In other words: a proper balance between detail and abstraction has to be found and since the bars lack space to provide task information, meeting these demands by a Gantt chart is tricky.

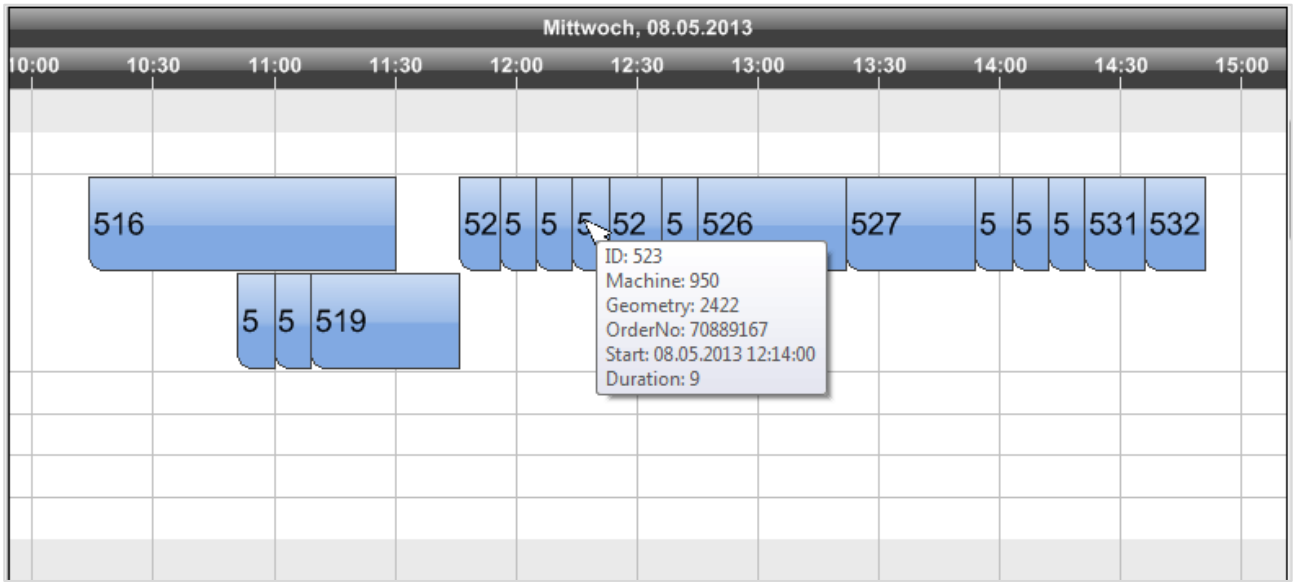


Of course one could select small units like hours in the calendar settings, but this would not provide an overview of the next days' machine capacity load that is essential for the scheduling process. However, an intelligent Gantt chart offers a way to solve this problem by letting the planner zoom into selected tasks.

The example diagram being used here was created with the component VARCHART XGantt .NET. It first provides the high-level overview and shows all operations of one production order as one joint and comprehensive bar. Hence, all the details of any (micro) operation are not seen when starting this sample application. In addition to this, the Gantt chart works with "speaking colors" and applies layers arranged one above the other to quickly show that there are conflicts on some machines regarding the scheduled tasks. Tasks having the same start date are not arranged next to each other here but one below the other. Thus, the scheduler can identify bottlenecks much faster, and hence can take corrective actions sooner.



By a simple mouse click he selects the area he wants to have a closer look at (colored in brown), thus zooming into the selected area and viewing the single tasks of this period being displayed individually. Further information on the individual tasks is provided by the tooltip text containing all planning-relevant details.

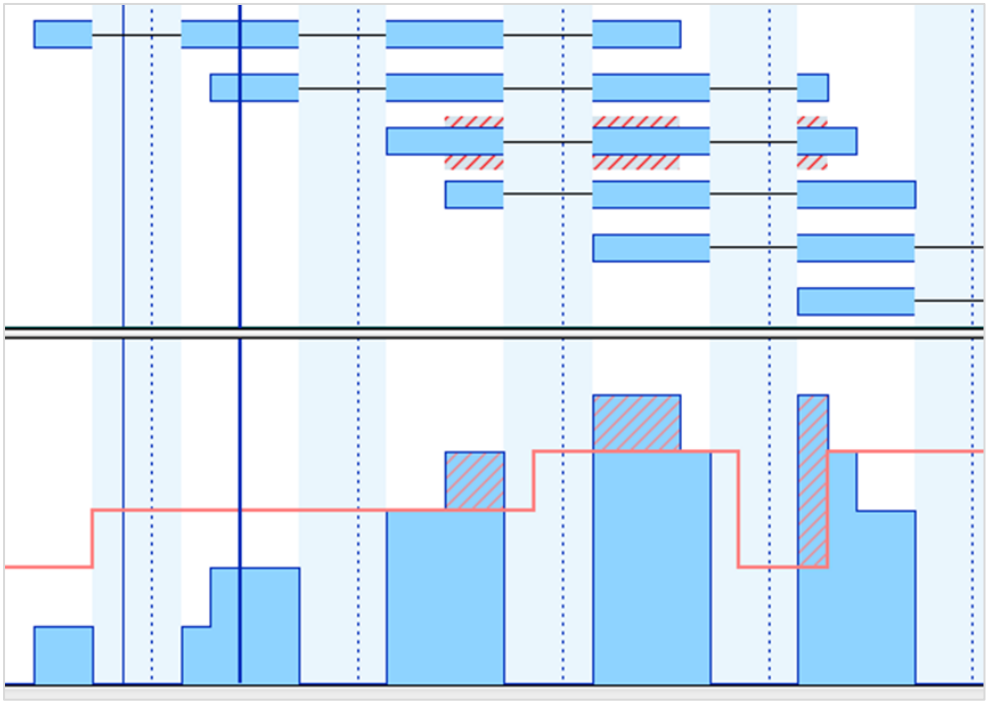


This way of visualizing allows the planner to keep control of the tasks, to increase the machines' efficiency and to improve the delivery precision.

2. Visualize Resource Bottlenecks in Gantt Chart Row

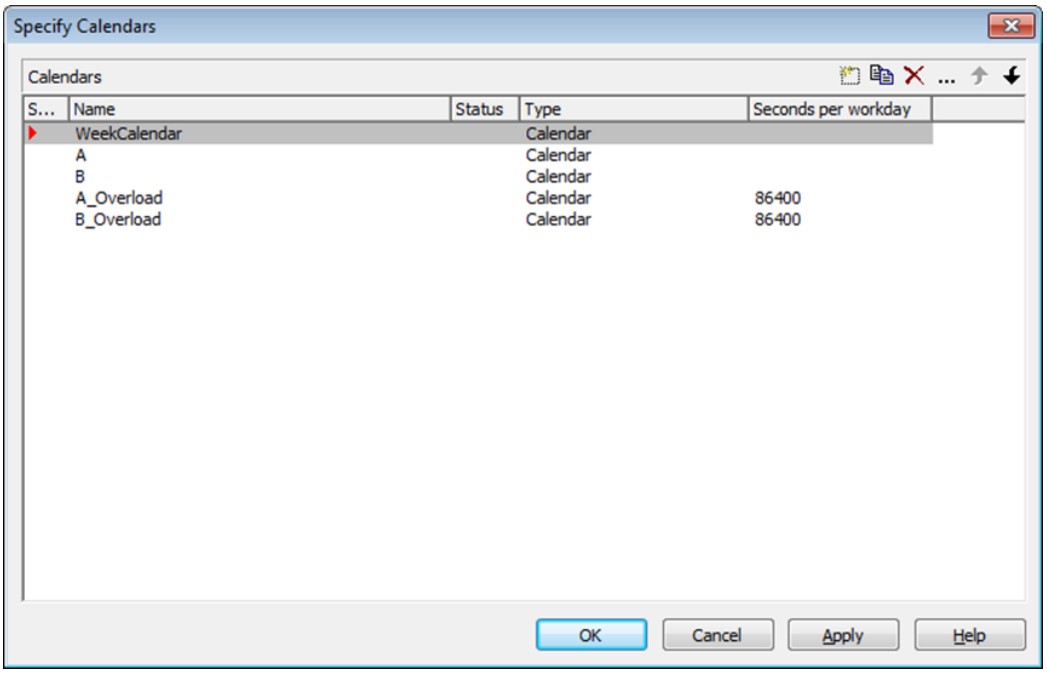
Gantt charts are frequently used to visualize and manage resource utilization and capacity load in production scheduling applications. Ideally, this is achieved by complementing the Gantt diagram with a histogram. However, when moving a bar the user always has to observe two moving elements on the screen in parallel: the bar and the histogram. Sometimes, this feels as difficult as determining an offside position when playing football. Beginning with version 5 of VARCHART XGantt , we provide .NET Gantt chart developers with the capability to also show resource overloads in the respective diagram row.

We want to achieve that overload ranges in a Gantt chart grouped by resources, each with its capacity load being shown in a histogram of its own, are not only displayed in the histogram but also directly at the node while pointing at it with the mouse cursor.

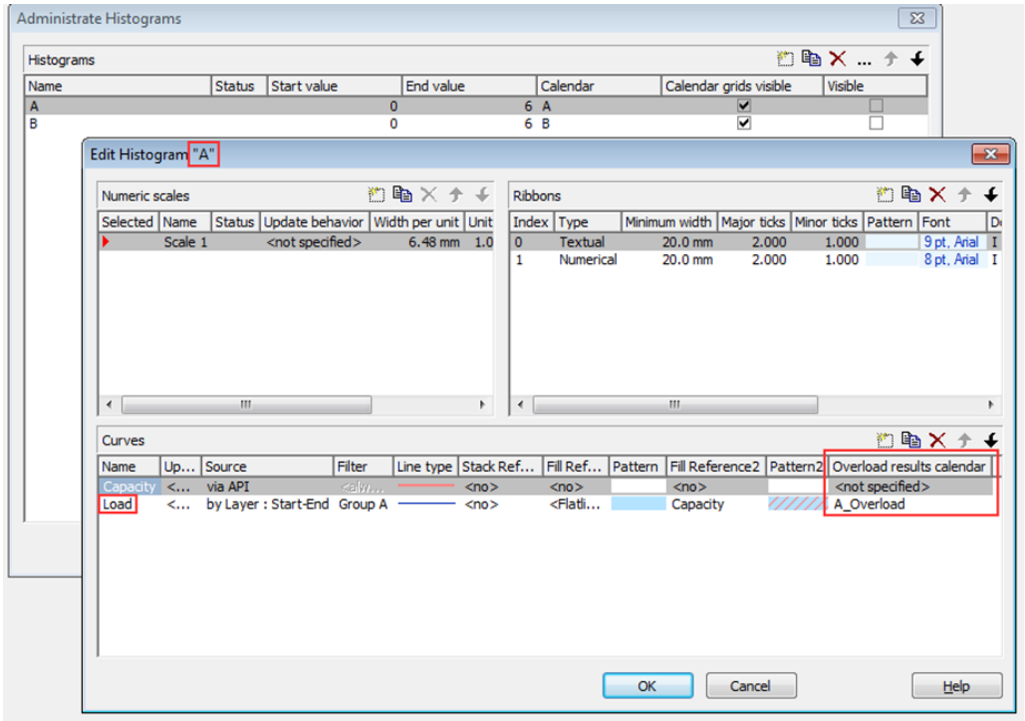


The following settings are needed to achieve this:

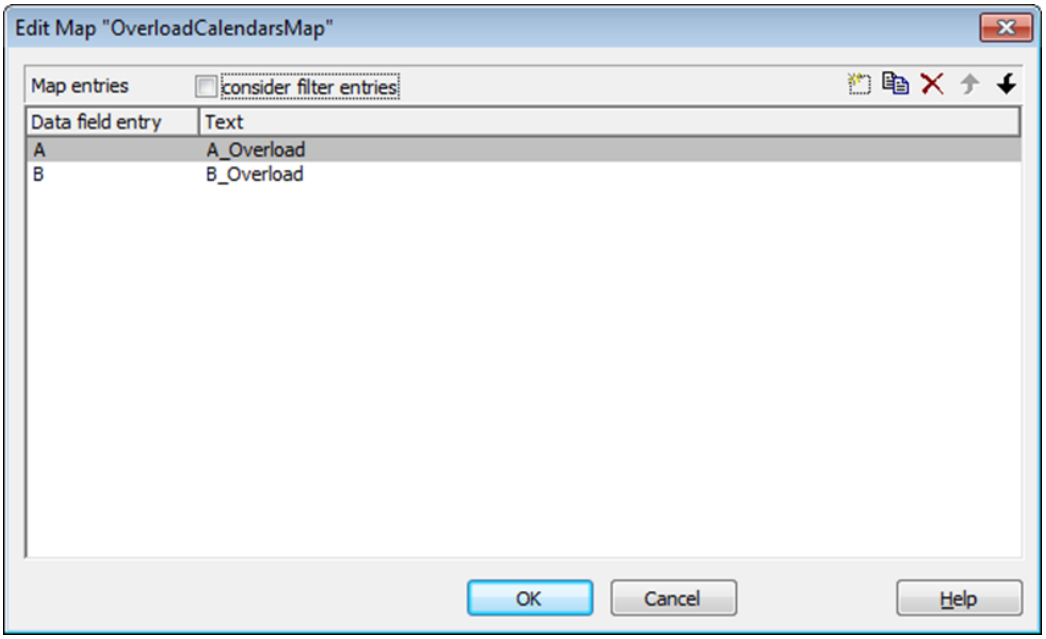
1. The present example features the resources A and B with their respective group calendars both having the same name. Create an overload calendar for each calendar and name them A_Overload and B_Overload accordingly. The calendars have to be empty, which means that they must not contain intervals or profiles.



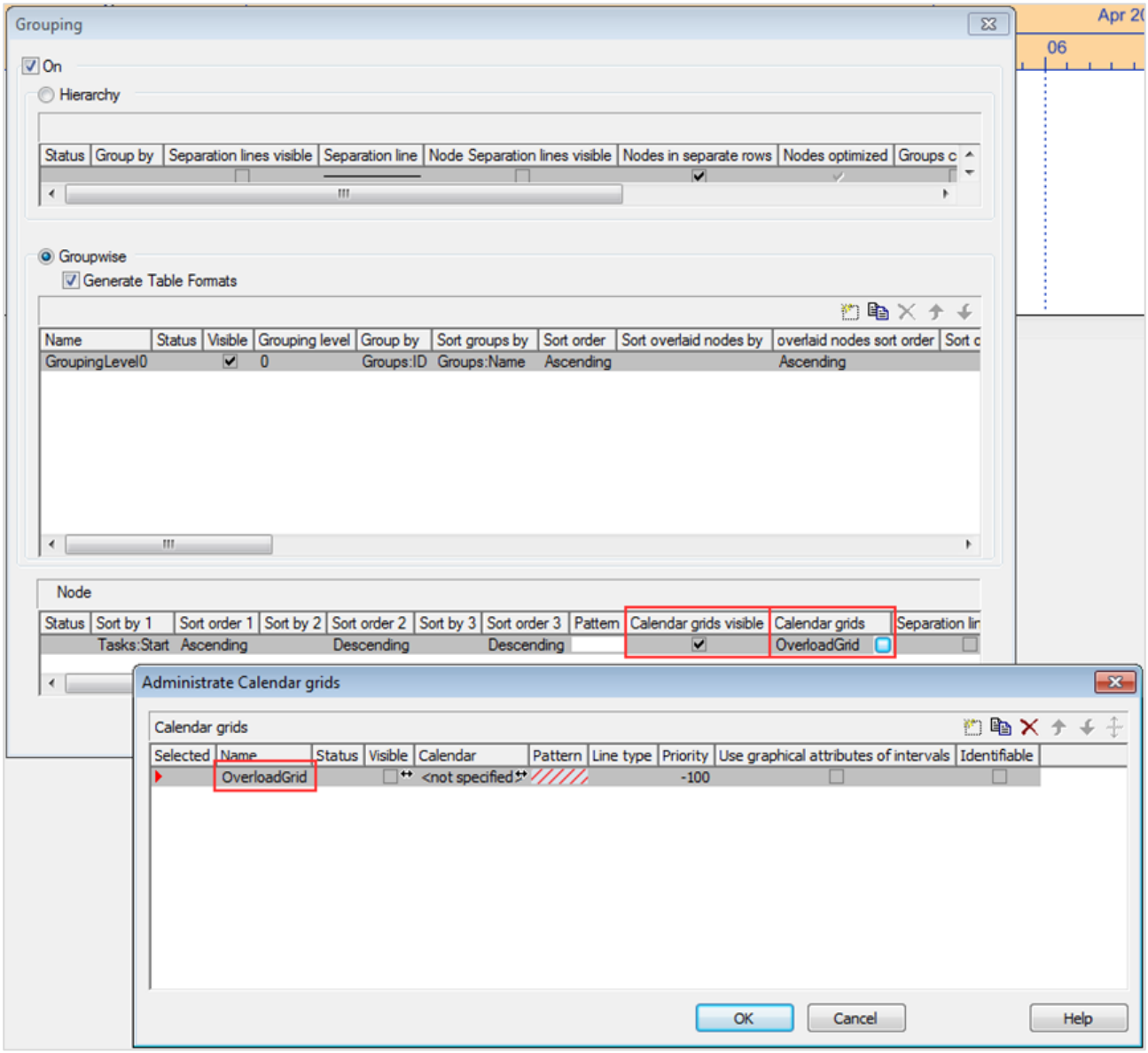
2. Specify these calendars as **Overload Results Calendar** in the capacity curve of each histogram.



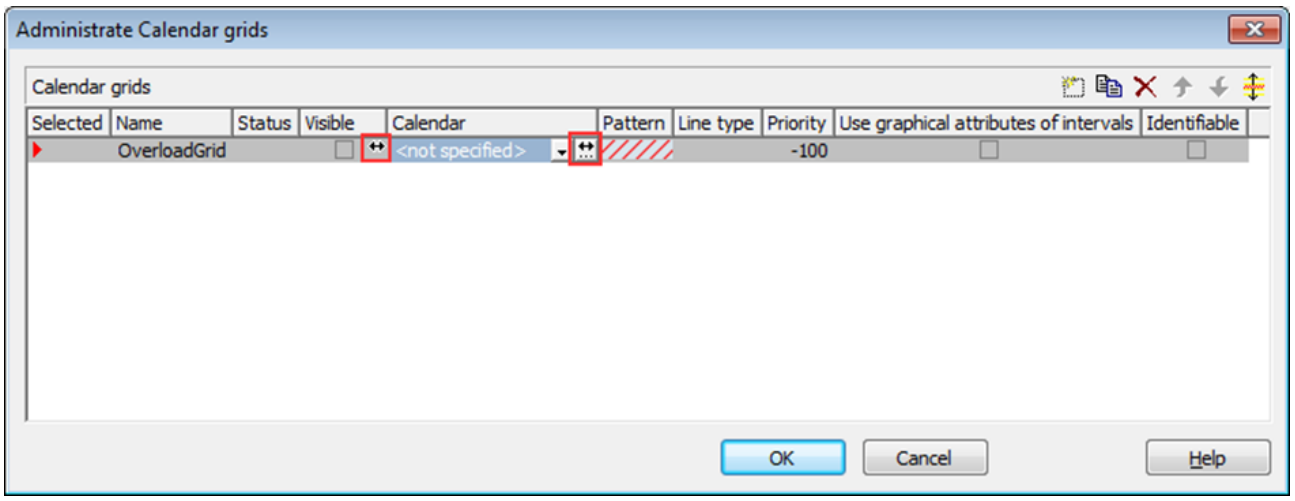
3. Create a mapping table to assign the overload calendars to the resource names (i.e. the contents of the grouping data field).



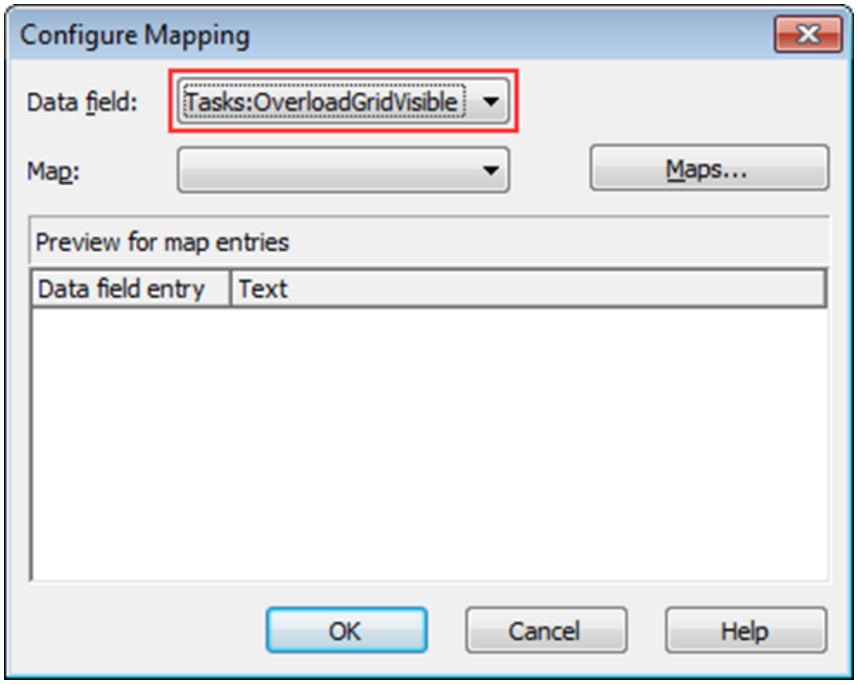
4. In the **Node** area of the **Grouping** dialog create the new calendar grid **OverloadGrid,** and tick the **Calendar grids visible** box.



5. In this calendar grid, the visibility and the calendar being used have to be mapped:



The **Visible** value is taken from the data field **OverloadGridVisible** that is set to 0 or 1, depending on whether the overload grid is to be displayed for the respective node or not.

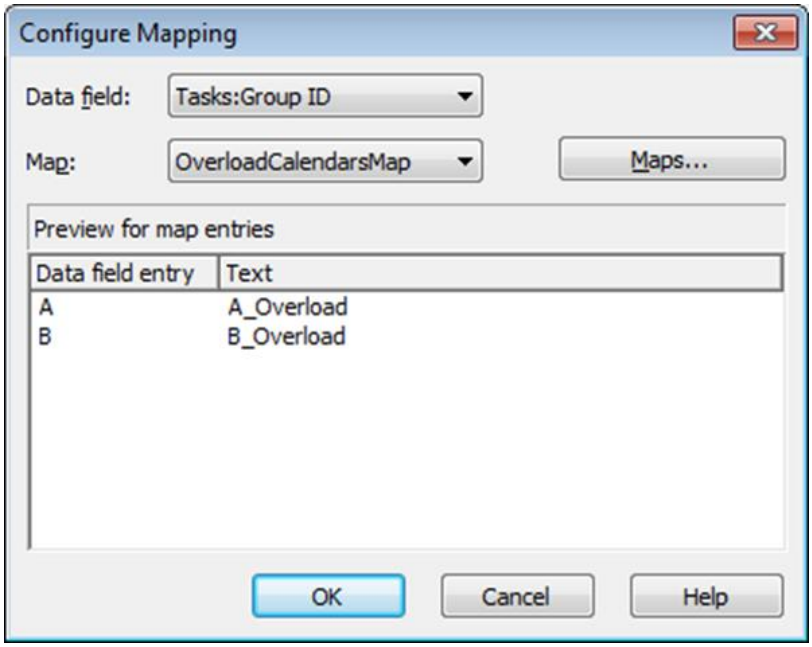


The values are assigned to **OverloadGridVisible** in the code as follows:

```
private void vcGantt1_MouseMove(object sender, MouseEventArgs e)
{
   string histogramName = string.Empty;
   object identObj = null;
   VcObjectType identObjType = VcObjectType.vcObjTypeNone;

   if (MouseButtons == MouseButtons.None
   {
      vcGantt1.IdentifyObjectAt(e.X, e.Y, ref identObj, ref identObjType);
      ShowOverloadsInNodeRow(identObj, identObjType);
   }
}

private void ShowOverloadsInNodeRow(object identObj, VcObjectType
identObjType)
{
   VcNode node;
   VcNode oldNode = vcGantt1.GetNodeByID(_oldNodeID);

   switch (identObjType
   {
      case VcObjectType.vcObjTypeNodeInDiagram:
      node = (VcNode)identObj;
      node.set_DataField(eTasks.OverloadGridVisible, "1");
      node.Update();
      _oldNodeID = node.ID;
      break;
```

```
        default:
        if (oldNode != null)
        {
        oldNode.set_DataField(eTasks.OverloadGridVisible, "0");
        oldNode.Update();
        }
        break;
    }
}
```

6. The calendar to be used is mapped as described below:



Data field designates the field being used for grouping.

Please make sure to select the calendar grid, which will be indicated by the red triangle in the **Selected** column.

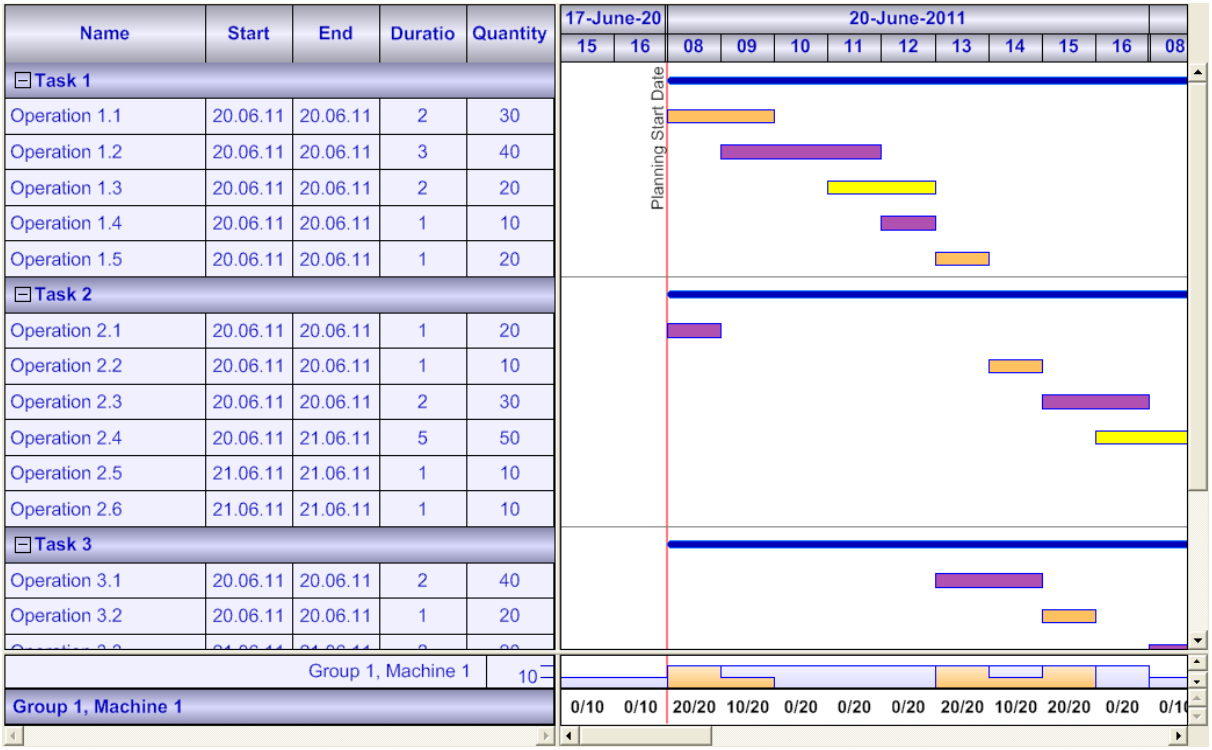
3. Use Values to Label a Curve

It is detailed planning in particular that would benefit from representing resources not only by a histogram to allow a quick recognition of capacity bottlenecks but also displaying the exact values of a resource's utilization in addition.

A vivid example of use would be production planning with the exact number of pieces of a machine being controlled. Due to the high complexity of one or more histograms containing stacked curves with many values over time that in many cases result in a lack of space and thus prevent the label from being readable, curves cannot be labelled in VARCHART XGantt. If only few and seldom changing values are displayed in the application, thus leaving enough room for a label in X-direction, a curve label can be obtained by a second XGantt instance that is placed below the histogram.

In our sample the Gantt chart is created by VARCHART XGantt ActiveX. The capacity of a machine, measured by the number of pieces, is displayed together with the scheduled number of pieces (the first number always denoting the utilization, the second number the maximum availability of the resource):

Production planning displaying the utilization in the histogram



The highlighted order stipulates a production of 30 pieces. The histogram shows the resource scheduler's calculated distribution of 20 pieces at 8 o'clock and 10 pieces at 9 o'clock. The available capacity of 10 pieces at 9 o'clock can thus be easily recognized.

The horizontal scroll bars of the upper XGantt have to be disabled to make the two XGantt entities look like one diagram. Moreover, the settings of the zoom factor, the table width, the horizontal scroll offset and the unit width for the diagram have to be the same. This is achieved by the handling of several events such as OnTableWidth, OnTimeScaleSectionRescale, OnZoomFactorModifyComplete and **On(Pre)ScrollDiagramHor**. The sample **ControlCenter**, being included in the download of VARCHART XGantt and showing two XGantt entities which are arranged one below the other illustrates how the implementation is accomplished.

To position the numbers, a layer designed for containing the label, is created in the second XGantt at design time. This layer may be borderless and transparent so that the label doesn't look like a common layer. At runtime, a group and a node per label are created, the label being placed in that period of the curve that is to be labeled. When the group layout is switched to "All nodes in one row" and "Nodes optimized", the labels will all be drawn next to each other at the correct X-position! Use the following code to create the label (VB98):

```
Private Sub loadCurveValues(histogramName As String)
   Dim currentDate As Date
   Dim histogram As VcHistogram
   Dim capacityCurve As VcCurve
   Dim loadCurve As VcCurve
   Dim leftDate As Date
   Dim rightDate As Date
   Dim leftValue1 As Long
   Dim leftValue2 As Long
   Dim rightValue As Long
   Dim cal As VcCalendar

   'Reset second XGantt (only necessary if switching between several curves is
     possible)
   VcGantt2.Reset vcRemoveNodes

   'Calculate necessary objects
   Set cal = VcGantt1.CalendarCollection.Active
   Set histogram = VcGantt1.HistogramCollection.HistogramByName _
                                                    (histogramName)
   Set capacityCurve = histogram.CurveCollection.CurveByName(histogramName)
   Set loadCurve = histogram.CurveCollection.CurveByName _
                                              ("Load_" + histogramName)

   'Create nodes in a loop over the time displayed
```

```
   currentDate = VcGantt1.TimeScaleStart
   Do While currentDate < VcGantt2.TimeScaleEnd
      'Read curves
      Call capacityCurve.GetValues(d, leftDate, leftValue1, _
                                   rightDate, rightValue)
      Call loadCurve.GetValues(d, leftDate, leftValue2, _
                               rightDate, rightValue)

      'create node for second XGantt with Node-ID, 2 values, group name =
      'histogram name, start and end date
      VcGantt2.InsertNodeRecord CStr(d) + ";" + _
                                CStr(leftValue2) + "/" + _
                                CStr(leftValue2) + "/" + _
                                histogramName + ";" + _
                                normDat(d) + ";" + _
                                normDat(DateAdd("h", 1, d))

      'calculate next start date (at the end of a working day moving forward to
      'the beginning of the next working day)
      currentDate = cal.AddDuration(currentDate, 1)
      If Hour(currentDate) = 17 Then
         currentDate = cal.AddDuration(cal.AddDuration(currentDate, 1), -1)
      End If
   Loop

   'import all nodes to the Gantt diagram
   VcGantt2.EndLoading

   'Perform grouping again
   VcGantt2.GroupNodes True
End Sub
```

4. Optimized Arrangement: Clarity Without Loss of Detail

The task of displaying large amounts of data in a Gantt chart very often leads to the question of how to gain or maintain clarity. One way of solving this problem would be to group the data records according to suitable criteria and then start by collapsing these groups, overlapping tasks being highlighted by overlap layers. Then, in order to see further details, individual groups can be expanded interactively.

This approach, however, leaves the user only with two options: view either space-saving, collapsed groups on the one hand or expanded groups with each data record being displayed in a separate row on the other hand.

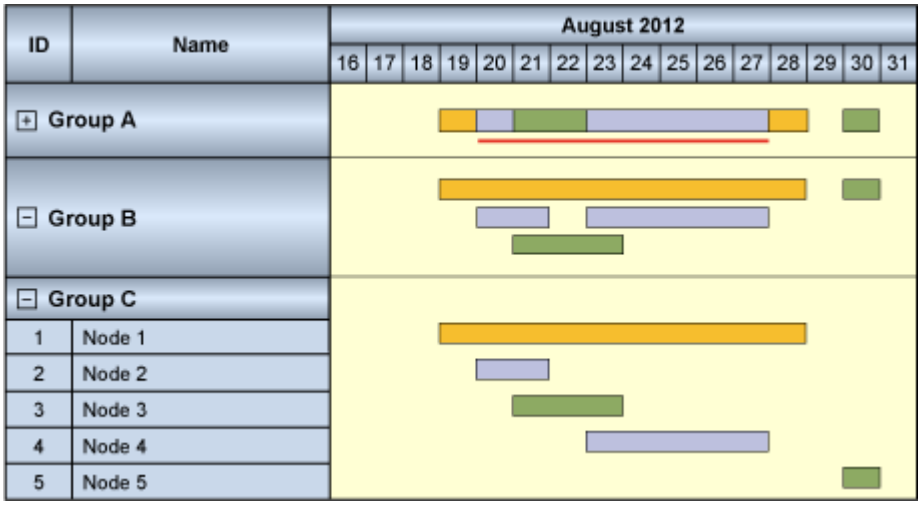
In some cases a visualization option combining these two extremes would be more useful (and desirable). For these cases NETRONIC offers a great layout feature: **Optimized arrangement.**

By using this feature, the user can place the data records of a group side by side in one row instead of in separate rows - as far as possible without overlapping. If bars overlap they will be displayed below the bars that have already been placed.

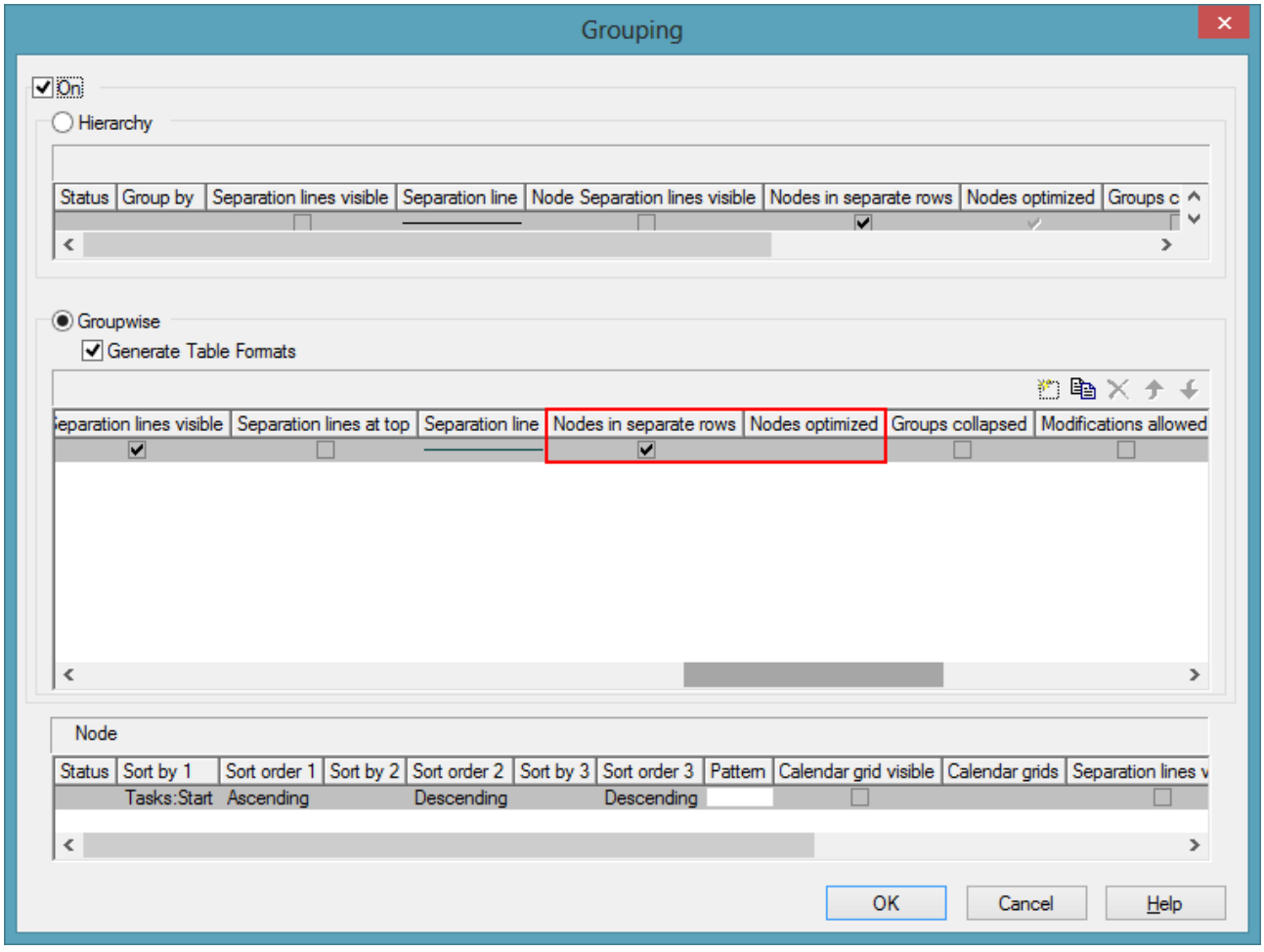
As a result, the feature provides a compact representation that at the same time gives an overview of the groups' details.

The following figure compares the options:

Group A: collapsed
Group B: optimized
Group C: expanded



The **Grouping** dialog offers all settings needed for this feature:



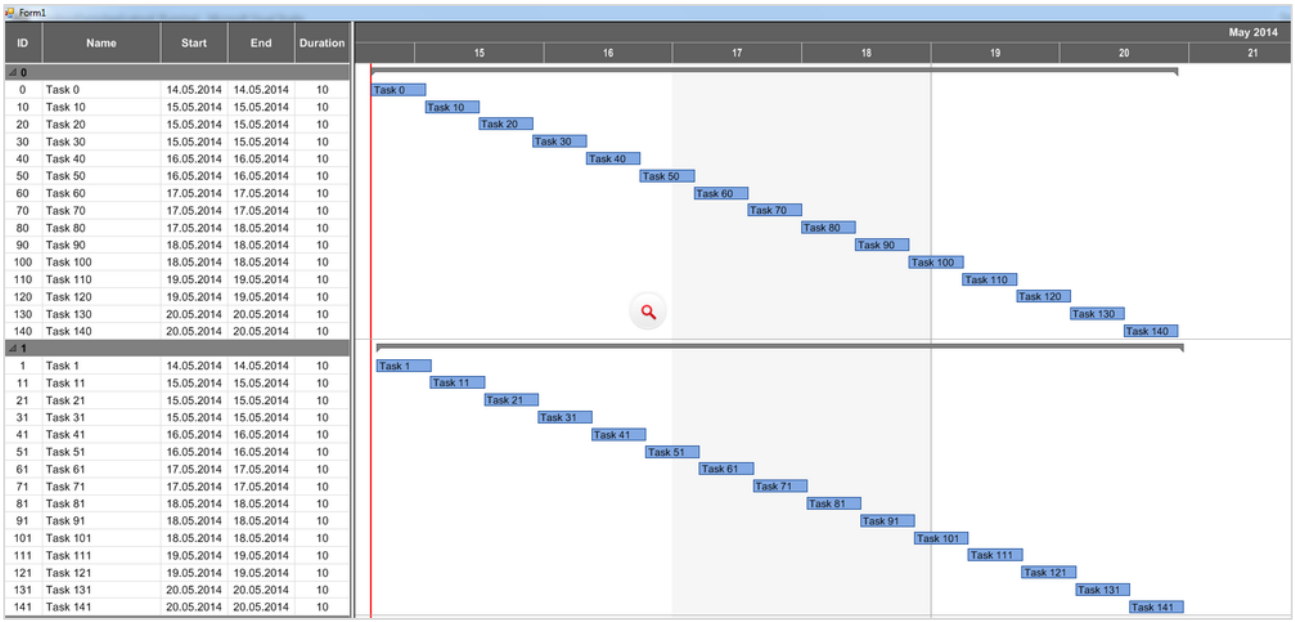
5. Extend Node Sorting Criteria in VARCHART XGantt

Let’s assume you visualize a list of tasks in a Gantt chart. In addition you need to sort these tasks in descending order to get a better overview of what task you should work on next. VARCHART XGantt supports up to three criteria to sort tasks. We show you an approach how to expand this number of criteria

How to sort nodes in VARCHART XGantt

Suppose we have 15 groups all filled with nodes and all of them are already sorted like this:

1. Criterion: Start-date in ascending order
2. Criterion: Priority in descending order
3. Criteron: Id - so that even if Start-date and Priority are the same the plan will have a deterministic order



ID	Name	Start	End	Duration
0				
0	Task 0	14.05.2014	14.05.2014	10
10	Task 10	15.05.2014	15.05.2014	10
20	Task 20	15.05.2014	15.05.2014	10
30	Task 30	15.05.2014	15.05.2014	10
40	Task 40	16.05.2014	16.05.2014	10
50	Task 50	16.05.2014	16.05.2014	10
60	Task 60	17.05.2014	17.05.2014	10
70	Task 70	17.05.2014	17.05.2014	10
80	Task 80	17.05.2014	18.05.2014	10
90	Task 90	18.05.2014	18.05.2014	10
100	Task 100	18.05.2014	18.05.2014	10
110	Task 110	19.05.2014	19.05.2014	10
120	Task 120	19.05.2014	19.05.2014	10
130	Task 130	20.05.2014	20.05.2014	10
140	Task 140	20.05.2014	20.05.2014	10
1				
1	Task 1	14.05.2014	14.05.2014	10
11	Task 11	15.05.2014	15.05.2014	10
21	Task 21	15.05.2014	15.05.2014	10
31	Task 31	15.05.2014	15.05.2014	10
41	Task 41	16.05.2014	16.05.2014	10
51	Task 51	16.05.2014	16.05.2014	10
61	Task 61	17.05.2014	17.05.2014	10
71	Task 71	17.05.2014	17.05.2014	10
81	Task 81	18.05.2014	18.05.2014	10
91	Task 91	18.05.2014	18.05.2014	10
101	Task 101	18.05.2014	18.05.2014	10
111	Task 111	19.05.2014	19.05.2014	10
121	Task 121	19.05.2014	19.05.2014	10
131	Task 131	20.05.2014	20.05.2014	10
141	Task 141	20.05.2014	20.05.2014	10

Every node represents a task. Now let us assume that one of the groups contains the unassigned tasks and the other 14 groups represent machines with assigned tasks.

Now we want to see tasks with a certain status at the top of the "unassigned tasks" group.

So how can we achieve this?

We could create another grouping-level to differentiate the states, but we would only need that extra level in the "unassigned tasks" group and not the other 14. This is not possible for only one group.

We could recreate the grouping structure as a hierarchy, but that would cause some considerable amount of work changing existing code and functionality just to show some tasks at the top of a group. This is not feasible.

So what can we do?

Simply use a 4th sorting criteria!

Extend the sorting criteria

We combine the values that would be in the new first and second criteria, save this combination into a newly added field in the node and change the first node sorting criteria to this new field.

This way we effectively use 4 sorting criteria while only using all 3 available criteria in XGantt.

To stick to our example we would sort the nodes like this:

1. Criterion: Status ascending is an Integer
2. Criterion Start-date ascending is a DateTime
3. Criterion: Priority descending is an Integer as well
4. Criterion: Id is a String

Now we have to combine the 1st and 2nd criteria and store the combined value to a new field.

This new field will be a String, because this is the only type that would let us combine an Integer with a DateTime.

Please keep in mind that by using a String field the values will be sorted alphanumerically. That means Integers in a String field will be sorted not by their value but by their leading characters.

The numbers 1 – 12, for example, would look like this:

1
10
11
12
2
3
...

To have them sorted by value we have to fill these number up with leading "0"s until the length of the longest number is reached:

01
02
03
04
…

With that in mind we use the sortable format for DateTimes, i.e. "yyyyMMddhhmmss", as well and we are good to go. Now we just have to combine those two values as a string and save it into the new field.

When this is finished for all nodes we just have to call **SortNodes()** on the XGantt instance and the rest will be handled by XGantt..

6. How to Best Show Nodes in Grouped View

In Gantt charts activities are often displayed in groups. This is done, for instance, in a "machine Gantt" which visualizes the machine load, and which differentiates machine groups and the respective machines and it is called double-staged grouping.

Groups in Gantt charts imply expanding and collapsing, depending on whether one wants to view the planning data in a broader perspective or in detail.

And this is exactly where conventional Gantt controls face issues: Normally, nodes are only displayed on one group level. How is it possible then to display nodes in a Gantt chart group either collapsed at the top level or expanded at the lowest level?

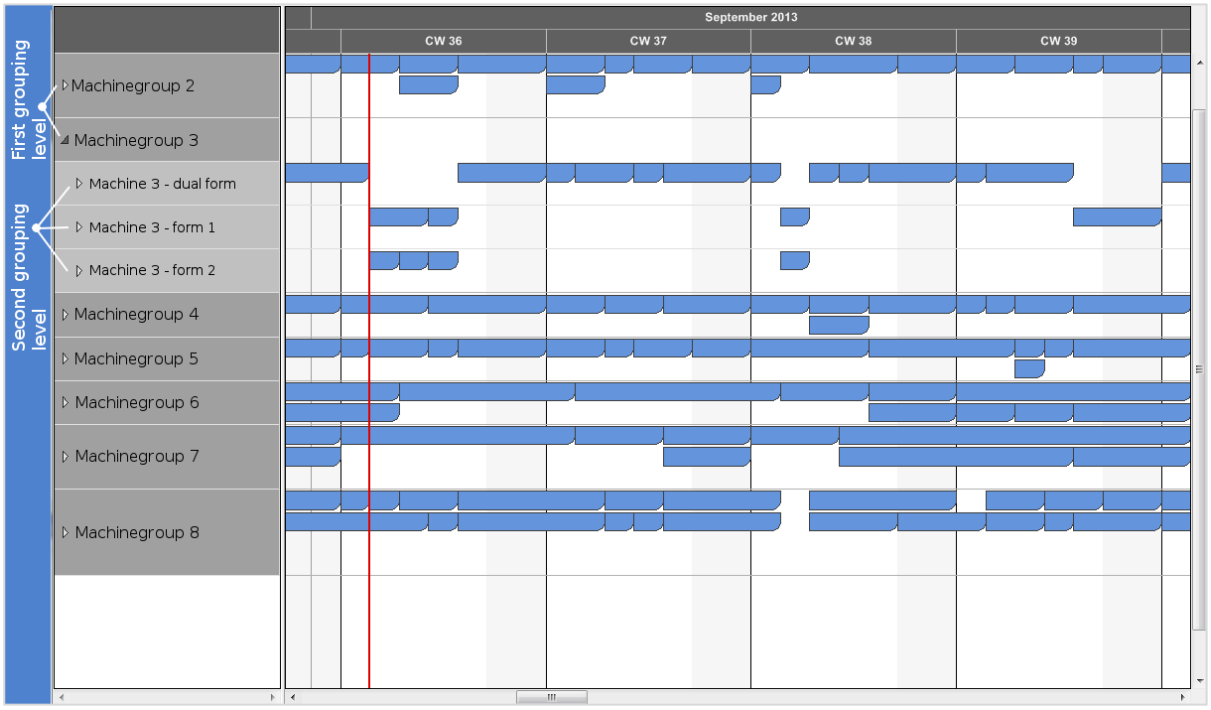
Read below how you can visualize the nodes on the group level you want and how an individual group based arrangement of nodes enhances the information value of the Gantt chart representing your planning data.

The VARCHART XGantt “shifting” trick

VARCHART XGantt is a very flexible .NET Gantt control that can be tailored to nearly every visualization requirement with respect to time-oriented planning data by only small programming additions. We proceed as follows to gain a simultaneous view of activities at different grouping levels:

The machine Gantt chart appears collapsed when it is started. The below sample hence displays the machine groups, whereas the single machines are not shown initially. Thus all nodes are displayed at the top level.

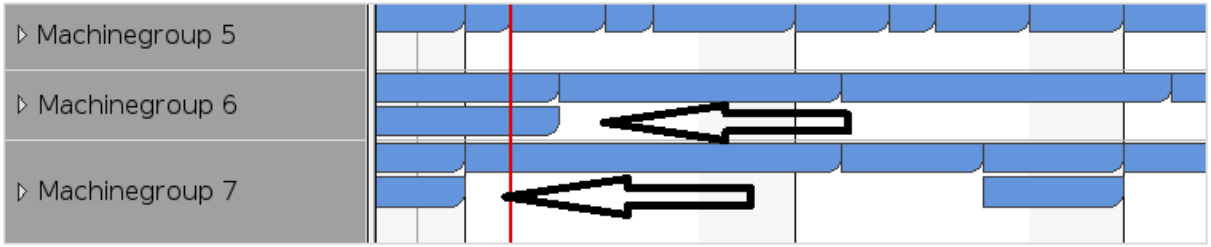
After having expanded the machine group, the second level - the individual machines - appear. Now, we visually shift the nodes (see 'how to instruction' at the end of this chapter) from the machine group to the machines. When collapsing the first level again, we simply undo the shifting and the nodes again appear on the first level.



Our Tip: Optimized arrangement of nodes in one row

To arrange the nodes at level 1 and 2 we have selected the VARCHART XGantt mode “all nodes in one row” simultaneously with the “nodes optimized” option. With this, the following is achieved:

a) All nodes are shown in one row next to each other so that you can see the machine utilization at one glance.
b) If nodes overlap, an additional row is shown and the overlapping nodes appear in the new row.

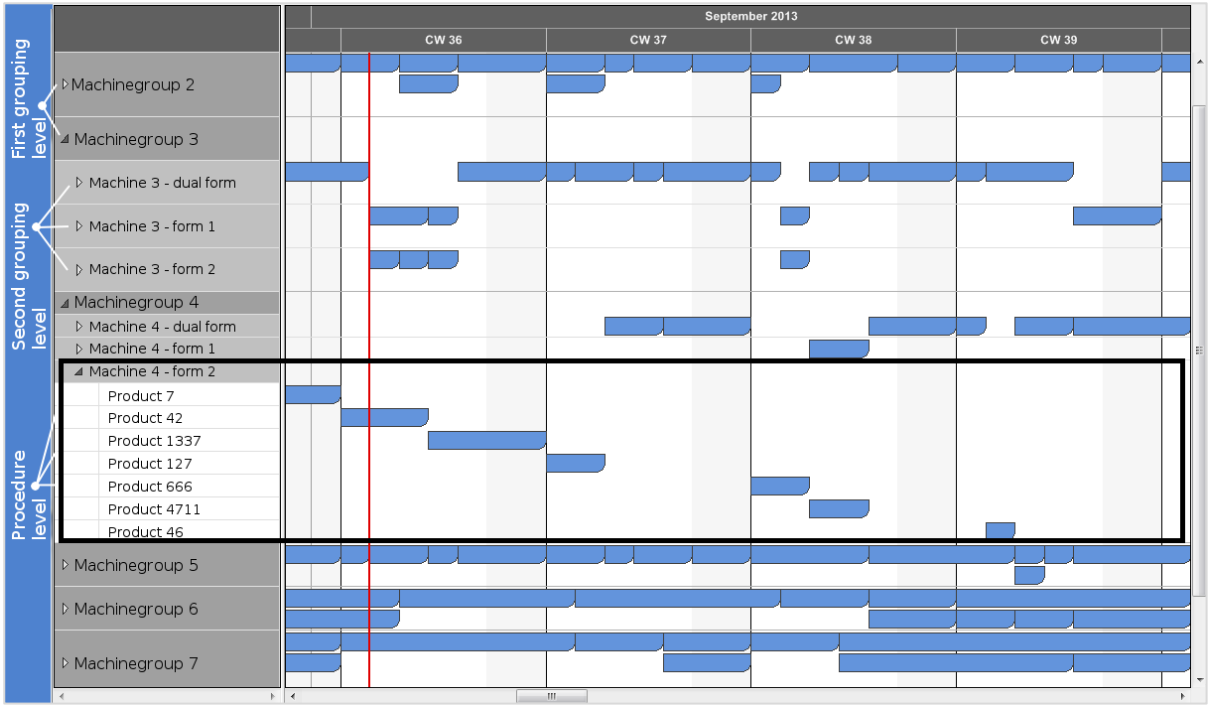


Change the node arrangement when the 2nd level is expanded

As we want to provide the planner with the option to get a view that is as detailed as possible, we have changed the “all nodes in one row” to “nodes in separate rows “ by ticking the respective check box for the second grouping level. On expanding the machine

view, every node will be clearly arranged in a row of its own so that the planner recognizes the exact timing of the activities.

In addition the activities are sorted by start date so the planner can easily find the very next planned activity immediately (see also our other tricks on how to sort data in VARCHART XGantt).



„How to“ for this node arrangement in the grouping view with VARCHART XGantt

Step 1

Please note: We don’t use extended data tables in this example. In our traditional data structures groups are created automatically by the grouping criteria of the nodes.

Settings on the VARCHART XGantt property pages:

We need at least three grouping fields in the data definition of the nodes.

1. Group name for the first grouping level
2. Group name for the second grouping level
3. Dummy grouping data field

The “Shifting” trick: If you leave the second grouping field empty, e.g. by an empty string, VARCHART XGantt sorts the respective node to the group at the next highest level. Don’t simply replace the grouping name of the second level by an empty string because then the value will be definitely deleted and can’t be assigned any more. This is why we need the dummy data field that either an empty string or the grouping name of the second level will be copied into when needed. Initially, the dummy data field will be left empty.

Step 2

Next, if not yet specified, we create a GroupLevelLayout for each of the two grouping levels.

Settings needed:

- Grouping by: Your grouping field resp. dummy data field
- Nodes in separate rows :false
- Nodes optimized: true
- Groups collapsed: true

Step 3

Now the VARCHART XGantt configuration file has to be exported, edited with a text editor and then imported again.

The exported INI file shows the "CreateGroupForEmptyEntry" key for each grouping level ([GroupingLevel_X]). To simplify matters, this key has to be set from “YES” to “NO” for all existing grouping levels. Don’t forget to set this key again for all grouping levels being added later.

Step 4

Needed source code:

As first step, we need to check which grouping level triggered the Modified event.

- If the first level (0) is collapsed or expanded, we have to create the second level and shift the nodes by a foreach loop.
- If the second level (1) is collapsed or expanded, we have to switch between single-line or multi-line view.

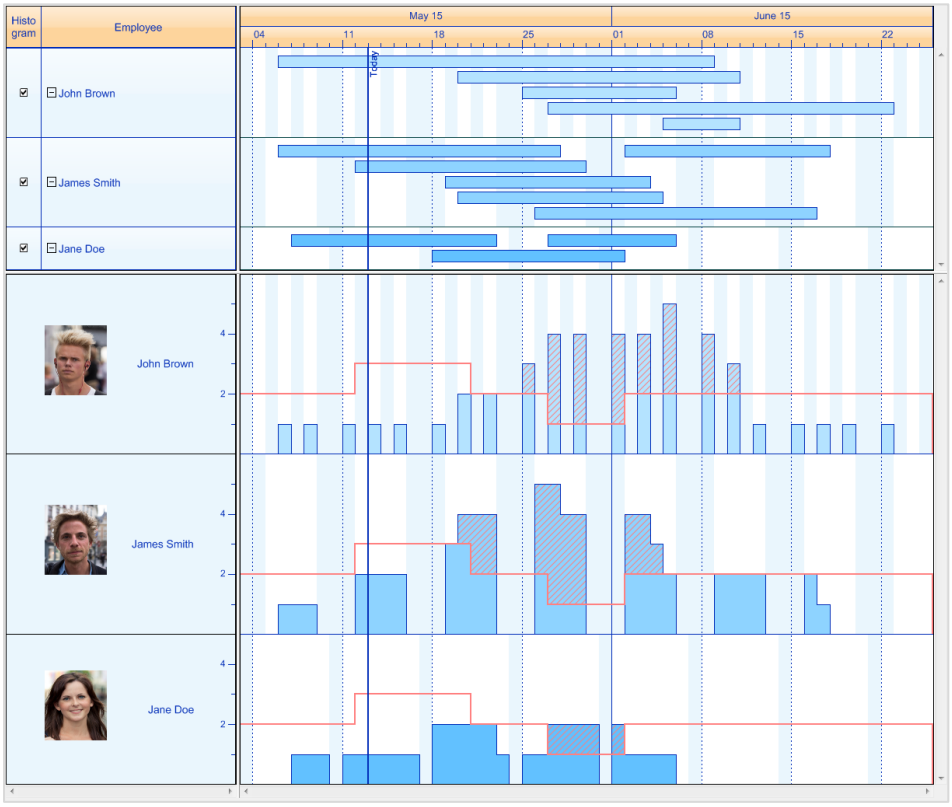
```
private void vcGantt1_VcGroupModified(object sender, VcGroupModifiedEventArgs e)
    {
    vcGantt1.SuspendUpdate(true);
    if (e.Group.GroupingLevel == 0)
        {
        if (e.Group.Collapsed)
            {
            foreach (VcNode node in e.Group.NodeCollection)
                {
                node.set_DataField(DT.Maindata_GroupLvl2FieldIndex, "");
                node.Update();
                e.Group.NodesOptimized = true;
                e.Group.Update();
                }
            }
        else
            {
            foreach (VcNode node in e.Group.NodeCollection)
                {
                node.set_DataField(DT.Maindata_GroupLvl2FieldIndex,
                node.get_DataField(DT.Maindata_GroupLvl2DummyFieldIndex));
                node.Update();
                }
            }
        }
if (e.Group.GroupingLevel == 1)
    {
    if (e.Group.Collapsed)
        {
        e.Group.NodesArrangedInOneRow = true;
        e.Group.NodesOptimized = true;
        e.Group.Update();
        }
    else
        {
        e.Group.NodesArrangedInOneRow = false;
        e.Group.Update();
        }
    }
    vcGantt1.SuspendUpdate(false);
}
```

7. Add Visual Value to Your Gantt Charts By Portrait Photographs

Imagine your Gantt chart being grouped by employees, showing a resource load chart (histogram) for every staff member. In that case, showing the employees' pictures next to the histogram would add great visual value to your application.

How to show images in the histogram of a .NET Gantt chart

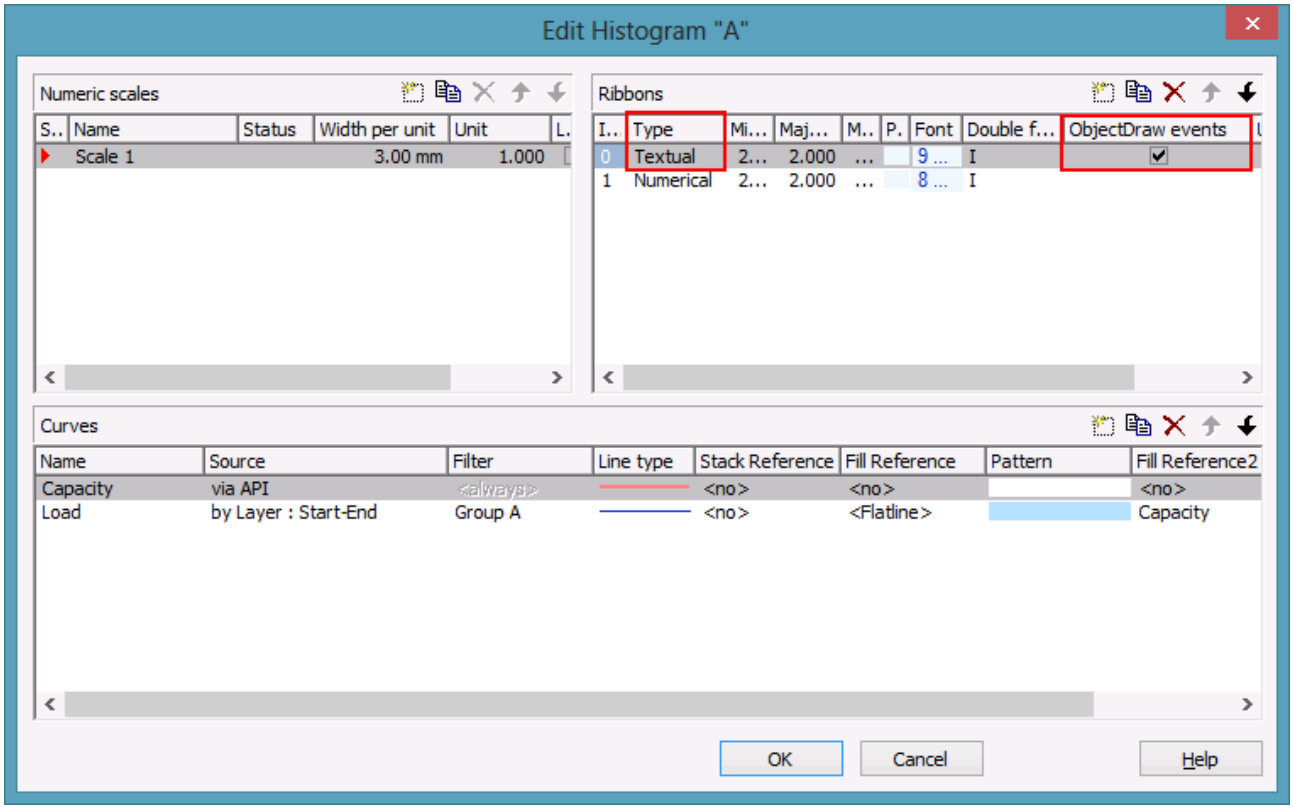
The below screenshot gives you an idea about what we want to achieve: put John's, James' and Jane's photos next to the histogram so that you get an immediate (visual) understanding of their workload.



Use code to add photographs

The event **VcObjectDrawn** lets you enrich objects being drawn by our .NET Gantt control VARCHART XGantt in terms of design with your own program code. To draw a picture, tick

the **ObjectDraw events** box of the desired ribbon in the **Ribbons** area of the **Edit Histogram** dialog at design time. As this option is only available for the ribbon type "Textual", you might have to add another ribbon.



Execute the following code in the VcObjectDrawn event:

```
private void vcGantt1_VcObjectDrawn(object sender, VcObjectDrawnEventArgs e)

 {
   Graphics g = e.Graphics;
   Bitmap myBM = null;

   switch (e.ObjectType)
   {
       case VcObjectType.vcObjTypeNumericScale:
         //Get the coordinates of a point in the numeric scale
         int x = e.UpdateRect.Right - 5;
         int y = e.UpdateRect.Top + 5;

         //Identify in which histogram we are
         object identObj = null;
         VcObjectType identObjType = VcObjectType.vcObjTypeNone;
         vcGantt1.IdentifyObjectAt(x, y, ref identObj, ref identObjType);
         VcHistogram histo = (VcHistogram)identObj;

         switch (histo.Name)
         {
           case "A":
             myBM = new Bitmap(Application.StartupPath
```

```
+ @"\..\..\Bitmaps\Image1.png");
              //photo credit: kinojam via photopin cc
              break;
            case "B":
              myBM = new Bitmap(Application.StartupPath
+ @"\..\..\Bitmaps\Image2.png");
              //photo credit: Enthuan via photopin cc
              break;
            case "C":
              myBM = new Bitmap(Application.StartupPath
+ @"\..\..\Bitmaps\Image3.png");
              //photo credit: chris zerbes via photopin cc
              break;
            default:
              break;
          }
//Draw the bitmaps in the numeric scales of the histograms
          x = e.CompleteRect.Left + 50;
          y = e.CompleteRect.Top + e.CompleteRect.Height / 2 - 50;
          g.DrawImage(myBM, x, y, 80, 90);
          break;
        }
}
```

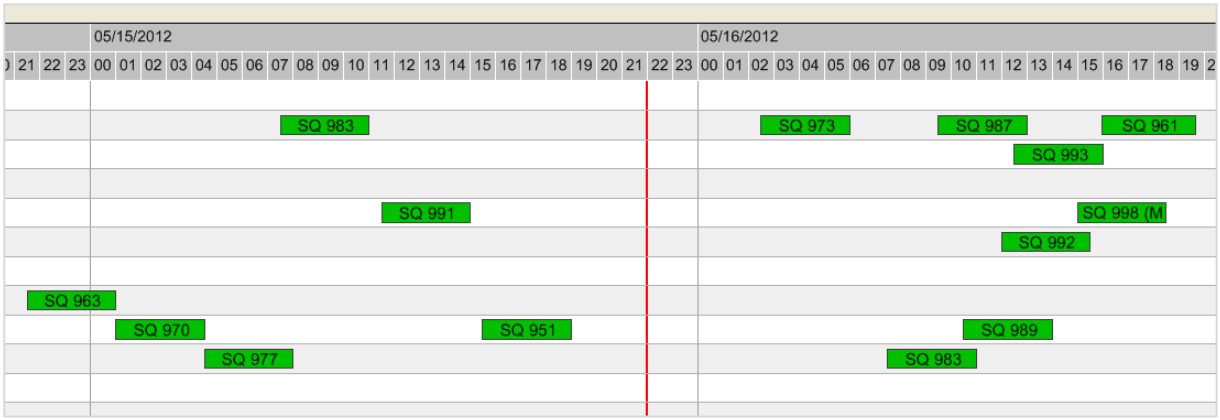
8. Scrolling Time Scale below Fixed Date Line – the Rolling Gantt

With VARCHART XGantt you can create Gantt charts allowing to shift the time scale and the bars ***automatically*** from right to left - similar to an animated banner. This is done by displaying a scrolling time scale below a fixed date line.

Create a **System.Windows.Forms.Timer object** that executes the following code on each **Tick** event*:*

```
DateTime leftDate = DateTime.MinValue;
DateTime rightDate = DateTime.MinValue;

VcDateLine dl1 = vcGantt1.DateLineCollection.DateLineByName("DateLine_1-1");

vcGantt1.GetCurrentViewDates(ref leftDate, ref rightDate);

if (vcGantt1.TimeScaleEnd <= rightDate.AddMinutes(2))
{
    vcGantt1.TimeScaleEnd = vcGantt1.TimeScaleEnd.AddHours(1);
}

DateTime newLeftDate = leftDate.AddHours(1);
DateTime newDateLineDate = dl1.Date.AddHours(1);

dl1.Visible = false;

vcGantt1.ScrollToDate(newLeftDate, VcHorizontalAlignment.vcLeftAligned, 0);

dl1.Date = newDateLineDate;

dl1.Visible = true;
```

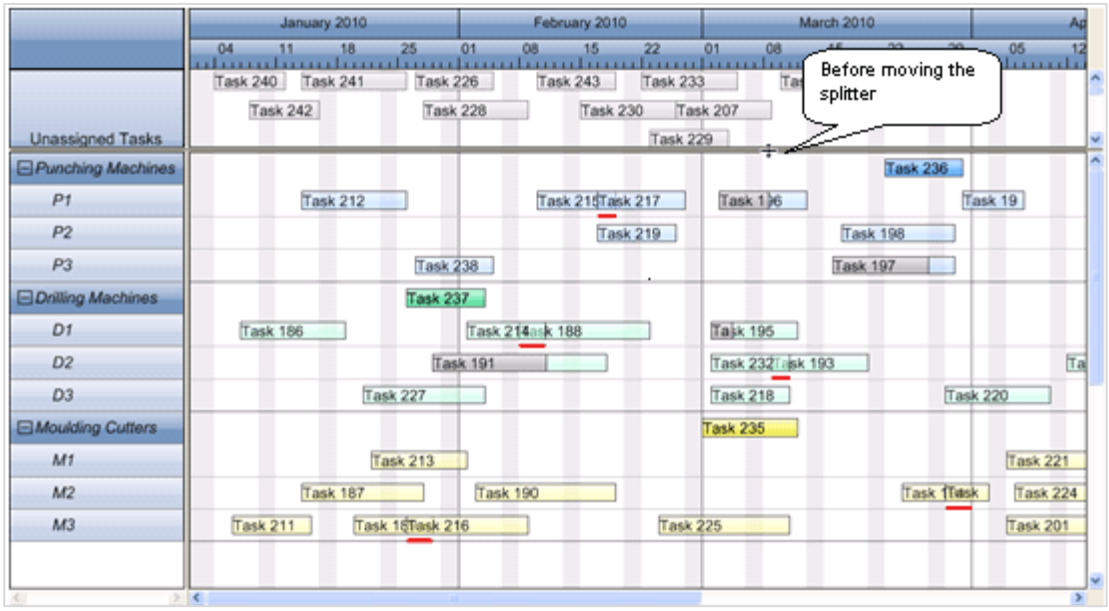
This code will cause the time scale to shift to the next hour on each **Tick** event. The temporal position of the date line can be set at design time on the XGantt property pages or with the **VcDateLine.Date** property when starting the program.

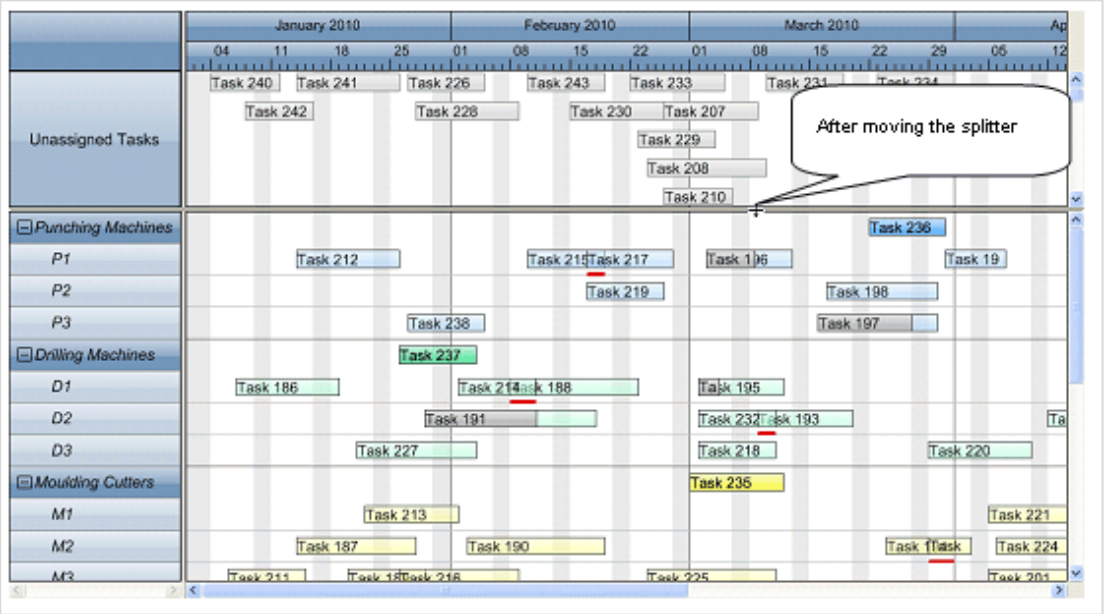

05/15/2012
05/16/2012
SQ 983
SQ 973
SQ 987
SQ 961
SQ 993
SQ 991
SQ 998 (M
SQ 992
SQ 963
SQ 970
SQ 951
SQ 989
SQ 977
SQ 983

9. Gantt Below Gantt – Two Entities of VARCHART XGantt On One Form

If you want to display two entities of XGantt on one form, you should use a SplitContainer. A SplitContainer is a control that consists of two panels, the proportions of which to one another can be modified interactively at runtime by a splitter between them. Using a splitter makes sense if, for instance, many unscheduled tasks are displayed one below the other.

If the two entities of XGantt are mounted on the two panels of the SplitContainer, the entities' width to height ratio to one another can be modified interactively.





To create a neat picture, the settings of the table width and the time scale should be the same for both XGantts. In addition, you should disable the horizontal scroll bar of the upper chart and the time scale of the lower chart so that there will be more room left for displaying the data. As the horizontal scrolling of the upper chart is then no longer possible, you need the following code to ensure the simultaneous scrolling of the upper chart in case the scroll bar of the lower chart will be moved (vcGantt1 being the upper, vcGantt2 the lower chart):

```
private void vcGantt2_VcDiagramHorizontalScrolled(object sender,
VcDiagramHorizontalScrolledEventArgs e)
{
vcGantt1.FitRangeIntoView(e.CurStartDate, e.CurEndDate, 0);
}

The interactive modification of the time scale solution (unit width) in the
upper chart requires a reaction as well (if this interaction has not been
disabled in general):


private void vcGantt1_VcTimeScaleSectionRescaling(object sender,
VcTimeScaleSectionRescalingEventArgs e)
{

DateTime leftDate = new DateTime(1,1,1);
DateTime rightDate = new DateTime(1,1,1);
int minBasicUnitWidth = 75;          //May have to be adjusted
if (e.NewBasicUnitWidth < 75);
{
e.TimeScale.get_Section((short)e.SectionIndex).UnitWidth = minBasicUnitWidth;
```

```
vcGantt2.TimeScaleCollection.Active.get_Section((short)e.SectionIndex).UnitWidth
= minBasicUnitWidth;
e.ReturnStatus = VcReturnStatus.vcRetStatFalse;

}
else
vcGantt2.TimeScaleCollection.Active.get_Section((short)e.SectionIndex).UnitWidth
= e.NewBasicUnitWidth;
vcGantt1.GetCurrentViewDates(ref leftDate, ref rightDate);
.ScrollToDate(leftDate, VcHorizontalAlignment.vcLeftAligned, 0);
}
```

The following code is needed to respond to the interactive moving of the splitter between the table and the diagram of the two charts:

```
private void vcGantt1_VcTableWidthChanging(object sender,
VcTableWidthChangingEventArgs e)
{
vcGantt2.LeftTableDiagramWidthRatio = (short)e.TableDiagramWidthRatio;
}
private void vcGantt2_VcTableWidthChanging(object sender,
VcTableWidthChangingEventArgs e)
{
vcGantt1.LeftTableDiagramWidthRatio = (short)e.TableDiagramWidthRatio;
}
```

10. Performance Increase by Partial Load

Learn about one more highlight of VARCHCHART XGantt: The property **VcGantt.PartialLoadThreshold** that under certain conditions significantly increases the data reloading performance. This mainly concerns transaction-intense applications that are characterized by the repeated reloading of only small data amounts into complex Gantt scheduling applications at runtime.

With the conventional and proven VARCHART XGantt approach, the default loading cycle starts with the first **InsertNodeRecord** (**DataRecordCollection.Add**) and ends with **Endloading**. When new data are added to the application, this methodology first removes all existing internal structures like groupings, sorting or summary bar calculations that are based on the previous data set. In the second step, these internal structures are rebuilt on the basis of the enhanced data. Meeting the demands of many NETRONIC customers, this methodology was especially optimized for loading large data amounts into complex Gantt structures such as loading data into an empty Gantt chart for the first time.

Decision cycles get shorter and planning and scheduling applications increasingly depend on (intraday) actual data. Hence the question of low latency becomes more and more important for our customers. In such volatile environments, the removal and rebuilding of the complete Gantt structures may not always be the most suitable way. Thus we decided to complement the conventional data loading methodology with an approach that is optimized towards rapidly feeding in only a few incremental data.

The VARCHART XGantt property **PartialLoadThreshold** lets the user define a threshold for the number of incremental data up to which the VARCHART XGantt will be created in performance optimized partial updates. When new data are added to the Gantt chart, the system automatically performs either an optimized partial update for smaller quantities or a full update for larger quantities. The optimal threshold value depends on the customer-specific Gantt chart implementation, the complexity of the Gantt functionality, and the number of nodes already loaded and still to be added. For further information see the XGantt manual or look up the **Tips&Tricks** on our website.

The property lets VARCHART XGantt cope even with the most challenging requirements in terms of performance: On the one hand, the control allows for the quick loading of huge data volumes when the Gantt chart is initialized, at the same time, however, it can also handle rapid data feeds and updates.

11. Undo/Redo in XGantt Applications

In some occasions you might want to reverse certain actions you have carried out in your plan. For this purpose you can integrate Undo/Redo in your Gantt application.

Preliminaries:

First download this class.

The code is written in C# and could be converted into any language with little effort.

After having embedded the class in your project, the UndoHandler will be available. All public methods of the class are static so that it won't be necessary to create an object of the class or to initialize anything.

You'll have to declare the new Namespace in your project or change the namespace of the new class to that of your project to gain access.

- AddAction()
- UndoLastAction()
- RedoNextAction()
- Clear()
- IsUndoActionLeft()
- IsRedoActionLeft()

Capture Modifications:

All modifications of the project data have to be passed to the UndoHandler. Those modifications would be modifications by user interactions in the XGantt or by API calls of your application. The interactive modifications can be handled in the events below:

- VcNodeDeleting
  Example: UnDoHandler.AddAction(UnDoAction.ActionType.NodeDelete, e.Node);
  The passed node (e.Node) is the one to be deleted

- VcNodeModifying
  Example: UnDoHandler.AddAction(UnDoAction.ActionType.NodeUpdate, e.OldNode);
- VcNodeCreated
  Example: UnDoHandler.AddAction(UnDoAction.ActionType.NodeInsert, e.Node);
  The passed node is the one newly created

Undo/Redo

- UndoLastAction()
  Example: UnDoHandler.UnDoLastAction(vcGantt1);
  Undoes the last modification of the passed XGantt

- RedoNextAction()
  Example: UnDoHandler.ReDoNextAction(vcGantt1);
  Actions undone can be restored by Redo

Status:

Undo/Redo buttons should only be activated in applications if this makes sense. If no actions are possible, the buttons should be greyed out again. To do so, you have to know the current status of the Undohandler.

The methods IsUndoActionLeft() und IsRedoActionLeft()provide information about the status. Both return a Boolean value.

Restrictions:

The class UndoHandler was deliberately kept as small, clear and general as possible, this involving, of course, some restrictions in terms of functionality. You could, however, expand or modify this class as suits you best.

- Restrictions to one Gantt entity only
  Solution: Remove static declaration and create an UndoHandler entity for each XGantt
  Links are not being considered

Functionality

- Modification of data will create an object of the UndoAction class.This object includes all old node information as well as the kind of modification. There are exactly three ways of changing a node:
  1. Modification of data
  2. Deleting of nodes
  3. Creation of a new node
- The static class UndoHandler collects all single UnDoActions on a stack and administrates it. In addition, the UndoHandler is the interface to the application developer, meaning you

The internal events of the Undo depend on the ActionType.:

- If a node has been modified, the old status is “copied unto” the new one.
- If a node was deleted, a new one containing the data of the old one will be created.
- If a node has been created, it will be deleted again

Free Trial: Empower Your Scheduling Application

Did our best practice tips whet your appetite for more of XGantt or would you like to try yourself and benefit from our vivid samples?

Download our free trial version of VARCHART XGantt and have a look at the included sample collection.