Jaret timebars component (1.49)

Jaret timebars component (1.49)

Published 2013-09-17
Copyright © 2013 Peter Kliem

Table of Contents

List of Figures

List of Tables

About this document

This document is part of the documentation of the jaret timebars component. Other useful information is contained in the javadoc documentation of the component itself. Also consider examining the example code.

The current version of the software can always be found at *jaret.de* [http://www.jaret.de/timebars]. For any questions, inquiries and suggestions contact <peter.kliem@jaret.de>.

Date	Status	Author	Description
2007-02-18	Released	P. Kliem	Initial release of the document
2007-05-01	Update	P. Kliem	Update for 1.02 including first draft information about the timebars.addon package.
2007-06-05	Update	P. Kliem	Update for 1.10 (vertical orientation, calendar example)
2007-08-15	Update	P. Kliem	Update for 1.11/1.12 (variable row heights/ widths)
2007-09-16	Update	P. Kliem	Update for 1.13 (FancyExample, extended drawing area for intervals, licensing)
2007-11-03	Update	P. Kliem	Update for 1.15 (timeline example, known bugs, externalize overlap strategy)
2008-03-22	Update	P. Kliem	Update for 1.19
2008-04-06	Update	P. Kliem	Update for 1.21
2008-05-04	Update	P. Kliem	Update to 1.25
2009-01-01	Update	P. Kliem	Update to 1.30
2009-10-08	Update	P. Kliem	Update to 1.38 (incomplete)
2013-09-17	Update	P. Kliem	Update to 1.49 (incomplete)

Table 1. Document history

Chapter 1. About jaret timebars and it's concepts

The jaret timebars component is a widget/component for displaying a Gantt chart. It is designed to work with the SWT toolkit and Swing. Both versions share most of the code but the SWT version is the driving force and the one better supported. The Swing version may not support all features the SWT version does.

1.1. Overview

There are two viewer components (Swing and SWT) implementing an interface that is used by a delegate (one for both gui toolkits) to do the specialized gui handling and the interfacing to the configured renderers. The viewers make extensive use of the delegate that contains the core controller logic. In fact most of the viewer methods just delegate to the delegate.

Rendering (of nearly everything) is always done by renderers called from the viewer component. Renderers can support printing so that printing becomes possible (SWT only, renderers are aware of printing so that an optimized or adapted rendering is possible). Customizing the renderers for special needs (needs that can not be met by the supplied default renderers) and implementing the model (or extending the default models) is the main work to be done when using the component.

The model displayed is either a flat list of rows containing intervals or a tree structure containing the rows of intervals. All model elements are observables so modifications of the model will be reflected by the component instantly without further refresh actions need to be taken. When implementing a model the abstract base classes can easily be extended. The viewers support transparent sorting and filtering on the model. The model is clean in the sense that the viewstate is not part of the model (i.e. expanded/collapsed information when using a hierarchical model).

1.2. Features

- Follows the MVC pattern: the component visualizes and modifies all elements according to a well defined model interface. Visualization does not change the model for sorting or filtering.
- Rendering is done by renderers allowing a complete customization without modifying the source code of the component itself. Default renderes are supplied for ease of use and as a basis for customization. Renderers may support printing (SWT only).
- Supports flat and hierarchical models.
- Supports visual editing and interactions (dragging of intervals, resizing of intervals, resizing the header areas). For supporting drag & drop operations it is possible to draw "ghosted" intervals.
- Supports relations between intervals.
- Additional components: marker (vertical marking of a point in time), synchronizer to synchronize multiple viewers
- ViewstateSaver for saving the viewstate when using the timebar viewer in an eclipse RCP.
- Region selection that can be used for zooming etc.

1.3. Compatibility, platforms and status

1.3.1. Compatibility

The component requires Java 5 allowing a fully typed API. It is currently developed using SWT 3.4. However no special features of SWT 3.3 or SWT 3.4 are used, so that the component should work with these earlier versions.

1.3.2. Platforms

As for the Swing version there is no concern what platforms are supported. For the SWT version the main development has been done using Windows XP[1] . Other platforms that have been successfully tested are Linux/GTK2 and MAC OS X 10.4.5/Intel (however these arte not tested on a regular basis).

The component is completely custom drawn, so it does not adhere to the platform it runs on.

1.3.3. Status

This documentation accompanies the version 1.0 (or higher) of the timebars component. The component has not been built in a day: it's history goes back to 2004. This means it has reached a certain level of maturity and is used at some places (commercially and in open projects). The future development will try to be as interface stable as possible. However some changes to the interface will be necessary to support new features.

Note

If you encounter any bugs or oddities, please let me know: <peter.kliem@jaret.de>.

1.3.4. Future plans

The 1.0 version marks a stable version. Future extensions will be integrated as smooth as possible. If a future feature requires really large modifications in the API this will result in a second package or will include compatibility classes/interfaces.

Possible future features are

- Some more utilities (i.e. relations) UPDATE: partly realized with 1.19/1.25
- SWT: do a deep test on Linux/GTK and OSX
- Some more support for DnD

1.4. License / Dual licensing

The jaret timebars component is generally licensed under the GNU Public License (GPL). If you do not want to be constrained by the GPL (rendering your product to be under the GPL when using the component) you can obtain a commercial license from

Marcus Thyssen Softwareentwicklung
marsys.de [http://marsys.de]

You can find the two possible licenses in the appendix. However if you are not sure or if you are in need of a different license contact <peter.kliem@jaret.de>.

As of version 1.13 an exception to the GPL allows linking the timebars component to other freely available (source code) software. See the appendix for the concrete exception text.

[1]All trademarks used are property of their respective owners.

Chapter 2. Using the timebars

This chapter covers the main topics of using the jaret timebars component. It is an addition to the javadocs which are a useful and up to date source for most of the API functions. So this documentation will *not* cover all of the API. Most of the descriptions put the SWT version first.

2.1. The quickest start

The best way for a quick start is looking at the examples.

Tip

Download the source distribution and set up a project (either a normal java project or a plugin project) to reference it from the project you are using. In many cases it will be very handy to have the source at hand.

2.2. General notes

A component as complex as the timebars component naturally gets a huge interface. So you will notice a mass of methods that you might never need. Some of the complexity has been moved to the model and the renderers, rendering those more complex. And even with this huge API the semantics of an interactive Gantt chart are not universally defined.

So the timebars component tries to match common needs and common use cases. However: the source of the component is quite readable ... and since it is not getting any easier to use when getters/setters are introduced for *everything* there are some useful constants at the top of the implementations that can be examined first if something should be configured that is not accessible by getters/setters.

The delegate should not be used directly outside the timebar viewers. For debugging and experimental features it can be retrieved by `getData("delegate")` in the SWT world an by `getClientProperty("delegate")` with the swing vesion. This will be possible in all future versions. For some purposes (i.e. designing a custom overlap strategy) the delgate has to be used. For this reason a simple getter has been added with version 1.19. Please keep in mind that the interface of the delegate need not to be stable (in fact it is quite stable).

2.2.1. Dependencies

The timebars component relies on the jaret util package (downloadable from the website). This package contains some date/time functionality heavily used by the timebars component.

Caution

The `JaretDate` is a wrapper for convenient date/time operations. The wrapper has the drawback of being a mutable type thus the type should be handled with care and will mostly be copied by it's copy method before further usage.

The timebars component depends on jface (AKA the MVC tier in the SWT GUI world) because it contains actions to use with jface and is an implementation of a `ISelectionProvider` so it can be used as a selection provider in an eclipse RCP application.

The viewer interfaces and implementations in jface are about to be extended with 33M5. It might be possible that this will have an impact on future versions to get closer to some of the jface concepts.

2.3. The model

The timebar viewer is centered around the main data model: the `TimeBarModel`. The timebar model is a flat list of `TimeBarRowModels` containing `Intervals`(de.jaret.util.date) that make up the elements of the gantt

chart. The hierarchical model `HierarchicalTimeBarModel` contains one `TimeBarNode` as the root node of the tree. The nodes are an extension of the ordinary row with an additional list of children.

Intervals can be implementations of `IIRelationalInterval` allowing `IIntervalRelations` to be added that can be rendered by an `IIRelationRenderer` and that can be selected.

The things you have to know about the models:

- The rows and the complete model as the summary of the rows communicate a min and a max date of the time span contained in the model. As long as the default for the viewer is not changed the min/max of the displayed area is derived from the min/max of the model.
- The strategy for determing overlap information can rely on the sorting order of the intervals in the row (see javadoc).
- Usually you will find an abstract implementation and a default implementation. The abstract implementation contains the usual fire* methods for informing registered listeners about any changes. The default implementation can be used out of the box but are in no way very performant or elegant. In most cases the default models will be quite useful - when it comes to a special use case you might consider to use an optimized model coded for the special use case.

2.4. The renderers

Most of the visual appearance is generated by the renderers. Renderers are used to actually paint elements of the viewer. The SWT renderers generally have a draw method that gets the appropriate parameters do to the rendering. Swing renderers are designed along the lines of the Swing TableRenderer: they are asked for a parameterized JComponent that than is used as a stencil to draw.

Caution

Swing renderers must not create a new JComponent for every paint request!

Caution

SWT renderes should take care to implement the dispose method to dispose any resources (i.e. colors) they have acquired. Renderes ill be disposed when the component is disposed.

The things you have to know about renderers in general:

- Renderers are called frequently. So try to get them optimized.
- Some renderers have some other methods for acquiring information that only the renderer can provide, like the tooltip text or selection areas.
- There are default renderers available. However these should be adapted to your needs.
- SWT: The `RendererBase` supplies a basic functionality.
- The region a renderer can paint on is usually secured by the clipping rect. However you can modify the clipping rect to draw outside of the renderers bounds (do use carefully). As of version 1.13 you can extend the drawing area "legally" (SWT). See below.
- SWT: the GC is used for all rendering operations. So make sure that the basic properties that might have changed during rendering restored when rendering finished (Foreground, Background etc.).

2.4.1. Printing (SWT only)

The SWT version supports printing the gantt chart. Printing is done using the `TimeBarPrinter`. When print-

ing the setup of a TimeBarViewer including the configured renderers is copied. The renderers for printing have to be provided by the renderers itself. For that purpose every renderer interface contains a factory method for creating a renderer that is capable of printing.

The renderers are aware when a paint request is processed for printing. So it becomes possible to use a smaller font or thicker lines to improve the print and use the resolution of the printer.

Note

Printing does not support variable x scales and printing does not support millisecond accuracy.

SWT printing support is being added to the GTK variant of SWT with version 3.3. A quick check with 33M5 shows, that printing on linux/GTK is not working properly.

Printing on the Mac with 32M5 did not work. However this is a quite outdated version.

Printing is not yet adapted for vertical orientation.

2.4.2. TimeBarRenderer

The `TimeBarRenderer` should have been called "IntervalRenderer". This is the renderer responsible for rendering the intervals itself. The default implementation just draws a gray box. The SWT version is capable of dispatching to several renderers depending on the class of the interval beeing rendered. This mechanism is superfluous (but remains for 1.0 since it does not hurt) with the possibility to register renderers for different classes directly with the timebar viewer (added with the 1.0 version).

The drawing area for the interval is calculated by the begin and end timestamps of the rendered interval. Usually the clipping of the graphics context is set to match this area, thus renderes can not draw outside the interval bounds. With version 1.13 renderes might implement the TimeBarRenderer2 interface (SWT), that allows informing the time bar viewer about a preferred drawing area. The clipping will then be extended to the preferred area, allowing drawing outside the interval bounds. The extended bounds will be taken into account when deciding which portion of the viewer has to be redrawn if `StrictClipTimeCheck` is false (default is `false`). The delegate holds the look ahead/back time in minutes that are used when determing which intervals have to be drawn.

Note

The extended drawing has a small impact on the performance of the timebar viewer component since more intervals have to be examined when drawing. Setting `StrictClipTimeCheck` to true will disable it.

2.4.3. TimeBarGapRenderer

The `TimeBarGapRenderer` is used to render elements between intervals in a row. For SWT there is a default implementation available that is usually not set to the viewer. For Swing an implementation can be found in the PDi example.

2.4.4. TimeScaleRenderer

The `TimeScaleRenderer` is responsible for drawing the time scale on the x axis. Care has to be taken if the scaling of the viewer is set to be variable since it can quickly happen that thousands of marks are painted by a time scale renderer that does expect a certain range of the scaling. The `DefaultTimeScaleRenderer` addresses this using the helper `TickScaler` that defines several tick intervals for different ranges of scale.

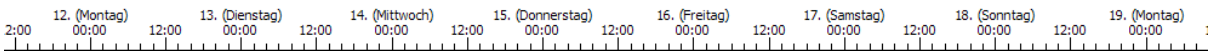

Figure 2.1. DefaultTimeScaleRenderer (german locale)

A second general purpose time scale renderer is included: the `BoxTimeScaleRenderer`. This renderer uses a number of DateIterators (jaretutil) to determine what strips of boxes to paint. The BoxTimeScaleRenderer is clearer than the default renderer.

The timescale can be rendered above the diagram or below the diagram. The position is set using `setTimeScalePosition`. A universal renderer has to take care of this possibility.

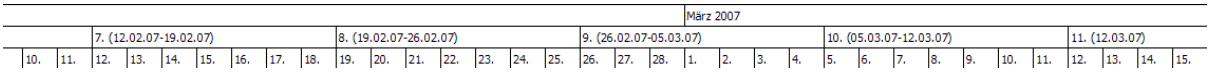

Figure 2.2. BoxTimeScaleRenderer (german locale)

2.4.5. HeaderRenderer

The header renderer is used to render normal header information (on the y axis). The default renderer simply renders the label provided by the `RowHeader`.

2.4.6. HierarchyRenderer

The hierarchy renderer renders the hierarchy part for a row when a hierarchical model is used. The DefaultHierarchyRenderer for SWT supports drawing custom (bitmap) symbols for collapsed/expanded nodes and leaves. The label and an optional icon can be supplied by an `ILabelProvider`. The swing version is quite plain.

2.4.7. GridRenderer

The `GridRenderer` is used to render the background of the diagram. The default implementation uses the `TickScaler` to determine it's steps for rendering unless a ITickScaleProvider has been configured. GridRenderers have to accept an ITickScalProvider to achieve a coupling between time scale renderers and the grid rendeering. The default grid renderer will use the supplied tick scale provider. The BoxTimeScaleRenderer and the DefaultTimeScaleRenderes do implement the ITickProvider interfaces.

The default grid renderer (SWT) can be configured to use a holiday enumerator for coloring special and holidays. It is also capable of coloring the weekend days.

As of version 1.24 the SWT GridRenderer is also responsible for drawing the row highlighting and row selection in the diagram. For a quick fix reserving the current behaviour just extend `AbstractGridRenderer` when you have implemented your own. The grid renderer interface will be a bit more extended in future versions to support some other things that have not been easy customizable. The getters/setters for highlight color etc. in the timebarviewer have been deprecated but will still work with every extension of the abstract grid renderer. The `DefaultGridRenderer` now uses alpha blending for drawing the row selection.

2.4.8. TitleRenderer

The `TitleRenderer` is a simple renderer for rendering the area that is defined by the timescale height and the header/hierarchy width. The default implementation just renders the title of the viewer (set by `setTitle`) in a bigger font centered. It can be configured to use an image as the background.

2.4.9. GlobalAssistantRenderer

The `GlobalAssistantRenderer` provides the possibility to to some rendering not bound to an element in the first place. The interface contains two methods that can be used for additional painting. One method will be called just before the intervals are painted, the other after the intervals have been painted. An example of the usage can be found in the `FancyExample` rendering marks on the diagram area.

2.4.10. IRelationRenderer

The `IRelationRenderer` is used to render relations between intervals. It is called after the grid has been

painted before the intervals are drawn. The relation renderer has to manage most things on its own since relation rendering can be done quite different. The relation renderer is responsible for supplying a method for checking whether a position in the diagram is occupied by a relation allowing the selection of relations.

2.4.11. IMiscRenderer

The `IMiscRenderer` collects some rendering tasks for various elements. The first things it draws are the selection and the region rect. In Swing it is also responsible for rendering row selections (SWT: done in the GridRenderer).

2.4.12. TimeBarMarkerRenderer

The `TimeBarMarkeRenderer` is used for rendering markers. The default implementations just renders a vertical line.

2.5. Performance

The performance is mainly driven by the speed of the renderers and the number of intervals displayed. Interactive operations on intervals use the look up methods in the model to determine if an interval is in a certain region of time. So if these methods are implemented imperformant that can have an impact.

The performance of painting with SWT is obviously best using Windows. On other platforms the paint methods are slower.

Version 1.02 introduced scrolling optimizations for SWT and Swing, i.e. graphical content is copied and only parts that are newly exposed will be drawn. The optimizations are enabled by default, since version 1.10. If you experience problems using SWT and the optimization under Linux/GTK and OSX/Intel disable optimzed scrolling. The optimized scrolling can not be used together with a variable xscale.

Note

The performnce will degrade when using non overlapping drawing with lots of intervals in a row, since the algorithm for determing the overlap information needs some time. As of version 1.19 the default overlap strategy has been improved (thanks to Mathias Kurth). If you know your overlap patterns you can supply an optimized overlap startegy by implementing `IOverlapStrategy`. An example of a custom overlap strategy can be found in the TimeLineExample.

2.6. Interactivity

2.6.1. Built in manipulation of intervals

The timebar viewer supports resizing and dragging of intervals as built in functions. Which intervals can be modified is controlled by possibly more than one `IntervalModificator`. This interface defines controlling methods to allow/disallow resizing and dragging of intervals. Additionally it provides the possibility to define a grid snap for the internal modifications. To generally allow modifications the `DefaultIntervalModificator` can be used which is a simple implementation allowing all modifications.

Resizing is done when an interval is dragged on an edge. The selection delta for determining whether an edge is being hit can be configured using `setSeletionDelta(int delta)`. This selection delta is also used when dragging markers and the limiters of the header and hierarchy area.

Manipulation of intervals using the built in interactivity features directly manipulate the model objects. This is very appropriate if you have other elements reacting on changes in the model. However if you have other observers registered to your model that do operations that take some time it can become quite unresponsive depending on the other observers. If this is the case you might consider layering the model or use drag&drop for manipulations, doing the real model change only in case of a drop.

2.6.2. Drag&Drop

SWT: The built in manipulation is a direct concurrency to the drag&drop functionality of the underlying OS. In the SWT version there is no built in functionality for handling dnd. The decision not to include functionality was made because there are uncountable variants of using dnd with a component like the timebar viewer. However to support dnd the possibility to render "ghosted" intervals and rows has been added. The remaining functionality has to be implemented outside the viewer component. An exmaple showing how to use the ghosted elements together with dnd can be found in the `SwtOverlapExample`.

Swing: Drag&Drop can be adapted. See the EventMonitorExample for a simple implementation.

2.6.3. Markers & co

`TimebarMarkers` are vertically rendered points in time for marking. They can be dragged in the area of the time scale. The selection is controlled by the selection delta. Markers are rendered by the `TimeBarMarkerRenderer`.

The y axis of the chart is composed of the hierarchy area (when using a hierarchical model) and the header area (or y axis area) for normal row headers. The widths of the areas can be set using the appropriate setters. If allowed (`setLineDarggingAllowed`) the limiting lines can be dragged by the user allowing adaption to the use case without further programming.

2.7. Other aspects and specials

This section collects some other aspects well worth beeing mentioned.

2.7.1. Keyboard control

The SWT version defines some keybindings. The bindings are documented in the javadoc of the TimeBarViewer. If there is the need to change the key bindings currently the only way is to edit the source. Keyboard control for the Swing version is pending.

2.7.2. Filtering & sorting

The timebar viewer supports sorting of the rows by setting a `TimeBarRowSorter`. The sorting is done only for the view. The model remains untouched. Filtering of the rows in the timebar viewer can be achieved by setting a `TimeBarRowFilter`. The filtering does not modify the model. When applying a row filter an/or sorter to a hierarchical model be aware that filtering/sorting will not take the hierarchy into account unless you code the implementations to do so.

Intervals can be filtered as well. To do so a `TimeBarIntervalFilter` can be set.

2.7.3. Listeners

Besides the listeners that can be registered with the model there are some more listeners that can be attached to the tiebar viewer:

- FocussedIntervalListener: invoked whenever the focussed interval changed
- ISelectionRectListener: invoked whenever a selection rect (ultiple interval selection) changes/closes or a region selection occurs.
- ITimeBarChangeListener: invoked whenever changes on intervals are ongoing.

2.7.4. TimeBarViewerSynchronizer

The `TimeBarViewerSynchronizer` can be used to synchronize two or more viewers concerning scale, rowheight and/or start date. This can be quite useful when showing different models on the same time scale.

2.7.5. Internationalisation

The timebar viewer itself does not contain any parts that have to be localized. The default renderers do adapt to the default locale. In the examples there are some localizations done for demonstrating purposes (mainly GridRenderer and TimeScaleRenderer getting a localized HolidayEnumerator for rendering holidays in different colors and show an appropriate tooltip. The actions in the util package have no support for localization; if needed they have to be modified. In some of the early examples you will stumble upon german comments and class names. This will hopefully not hurt.

2.7.6. Millisecond accuracy

The targeted use of the timebar viewer have always been intervals in a range from seconds up to weeks. So the horizontal scrollbar operates on seconds giving the possibility to scroll through roughly 68 years if the underlying platform supports full integer values for the scroll bar. If you set the viewer to millisecond accuracy (`setMilliAccuracy(true)`) the scroll bar will operate on milliseconds, resulting in a possible range of roughly 24 days.

2.7.7. Variable time scale

The timebar viewer has been designed with the idea of a fixed timescale. However if there is the need to display events on a timescale where the length of the intervals is very small compared to the distance between them it can be quite handy to change the scale on the x axis for regions. This will also allow a magnify function in the viewer. The solution for this is the possibility to change the scaling by adding special intervals holding a pixel per second value to the viewer. The mechanism reuses the concept of the timebar row to hold the special intervals making it possible to easily display them in an timebar viewer. The timebar viewer has to be explicitly set to the variable x scale mode with `setVariableXScale(boolean state)`.

The time scale can be broken by marking PPS intervals as breaks and specifying a pixel width that should be used for rendering.

An example of usage can be found in the swt hierarchy example. It has to be activated by modifying the source code (constant at the beginning of the source). Another example is in the MilliExample. In this example the pps intervals are displayed (and can be manipulated) in a second timebar viewer.

Caution

When using different scales make sure the timescale and grid renderers can handle the different scales. Otherwise it can easily happen, that the renderers that depend on the scale do a lot of unnecessary painting making the viewer slow. In the MilliExample the grid and the scale just stop painting when a special pps value is set.

Note

When the PPS intervals displayed in the timebar viewer that they manipulate and are changed there by dragging or resizing some operations are doomed to panic since all the basic values change while being used for calculations.

2.7.8. Context menus

SWT/Swing: The timebar viewer supports setting context menus on the different areas. For rows and intervals a delegate can decide to show different menus for different implementations of the model elements.

2.7.9. SWT: actions

The SWT version contains some actions that can be used with the timebar viewer. These are quite simple and can be found in the util.actions package of the distribution.

Note

These actions might be moved to a separate addon plugin in the future. They have no support for localization; texts are in english.

2.7.10. ISelectionProvider

The SWT version of the timebar viewer implements the `ISelectionProvider` from the JFace viewer package to allow easy integration in eclipse RCP environments.

Future versions might adopt the currently changing viewer API in JFace.

2.7.11. Region selection

The viewer supports a selected region (opposed to the rectangle used to select many intervals). It has to be enabled (`setRegionRectEnable`). A region is selected by shift-click-drag and remains in plave unless it is cleared (clearRegionRect). The painting is done by the IMiscRenderer.

2.8. Vertical orientation

The normal orientation for a gantt chart is horizontal. The timebars component does support vertical orientation for special cases where a vertical orientation might be useful. The orientation can be swichted using `setOrientation`.

The API of the timebars component has been designed for horizontal operation. This results in some methods having names that does not really match the orientation, e.g. `setXAxisWidth` has to be interpreted as "setYAxisHeight" when using vertical orientation. See the javadoc comments on methods if you are not sure.

Note

Renderers have to support vertical orientation. Most of the default renderers do support vertical orientation.

For an example that uses vertical orientation see the calendar example.

2.9. Variable row heights/widths

Since version 1.12 the timebar viewer supports variable sized rows (or columns if oriented vertical). This behaviour has to be activated (getTimeBarViewstate().setUseVariableRowHeights(flag)). If it is the row height/width can be set on the viewstate. The row heights can also be calculated by a strategy (implement `IRowHeightStaretgy`, set it on the viewstate). This makes it possible to resize rows of a certain type or those with a lot of overlapping intervals.

Interactive resizing of rows/columns is supported. It can be activated on the viewers: setRowHeightDraggingAllowed. Dragging of reights is possible in he x axis area and the hierarchy area.

Note

Using variable row heights/widths has an impact on performance if very large numbers of rows are used in the model.

Variable row heights/widths do not work together with row scaling (scale rows so that a fixed number or rows is always displayed).

Chapter 3. Addon package

This chapter comments on the timebars.addon package. The package contains some utilities that should be kept seperate because they introduce more dependencies on the timebars component. Please consult the javadoc of the classes for detailed information.

3.1. The ViewStateSaver

The viewstate saver is a useful tool for saving viewstate information in an `IMemento` when using the timebars component in an Eclipse RCP application. It saves most of the viewstate infomation like collapsed/expanded information on hierarchical models the scroll position and so on.

The viewstate saver requires a `IHierarchyIdService` for identifying the nodes.

3.2. The TableTimeBarSynchronizer

The timebarviewer itself does not support displaying tabular data beside the gantt display. It is possible to do a custom rendered, passive table using a customized header renderer. This will not be sufficient for more complex tabular data.

The synchronizer synchronizes a jaret table and a timebarviewer on several aspects. The vertical scrolling and the rowheights will be synchronized so that the widgets can be placed beside one another. The table header height will be adapted to match the height of the x axis of the timebar viewer.

The model for the table will be created by wrapping the table of the timebarviewer. There is support for both flat and hierarchical models. Selection will be synchronized on row level. For flat (non hierarchical models) filterig and sorting of the timebar rows will be synchronized from the table towards the timebar viewer.

Note

The synchronizer does not support vertical orientation of the timebar viewer.

Chapter 4. Some solutions and details

This chapter lists some solutions for common and not so common requirements that may or may not have been covered in the previous chapters.

4.1. Time resolution

The usual time resolution the timebar viewer operates in is seconds. This will work for most use cases and leads to a second accurate scrollable time of approximately 68 years (the scrollbar value is used directly by the viewer and is 32 bit signed). If the model range exceeds the directly scrollable range the range is extended by a factor, leading to lower scrolling resolution but allows for huge ranges. However for some scientific visualization a higher resolution is required. The viewer supports millisecond accuracy. This can easily be enabled (`setMilliAccuracy(boolean milliAccuracy)`). The millisecond accurate scrolling range will be approximately 24 days. If the range is exceeded the scrolling is extended by reducing the resolution.

A higher time resolution can be by doing a custom implementation of the `Interval` interface, adding getters and setters for your time format (supposing you have a special time object since the java.util.Date does not support more than milliseconds). You than do the scaling to milliseconds in the implementation of the interval. The only thing to be tweaked with time bar viewer is the rendering of the intervals and the rendering of the timescale.

Due to the limitations of the underlying java.util.Date dates before 1.1.1970 will not work out of the box. To enable the timebars for dates before 1970 the workaround would be a projection of the dates on dates after 1970. Maybe a simple shift would do. Than the timebars would work perfectly ... just the renderers need to do the backwards projection.

4.2. Daylight saving time (DST)

The viewer operates on the basis of a java.util.Date wrapped in a JaretDate. In the basis (milliseconds since 1/1/1970) this is not affected by DST. DST is done by the TimeZone and will be seen when a date is rendered readable by means of date formatting.

In the timebars DST mainly affects the time scale rendering. The default renderer and the BoxTimeScaleRenderer can be configured to do a DST correction. This will not be perfect as the ticks are shifted around the DST switch.

Chapter 5. Examples

This chapter comments on some of the examples that are supplied in the the source download of the jaret timebars. Other comments on the examples can be found on the website.

5.1. Hierarchy

There are examples demonstrating the use of a hierarchical models for swing and SWT.

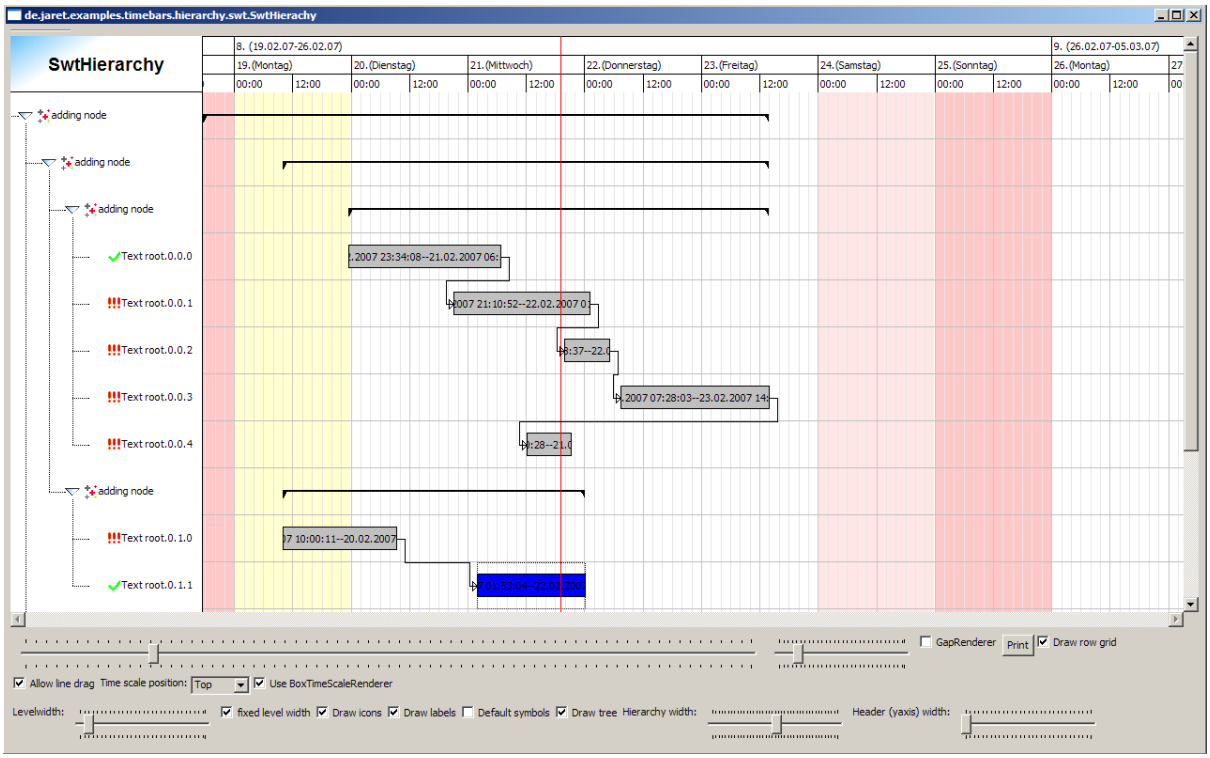

Figure 5.1. SWT timebars (1.0) showing a hierarchy

Simple hierarchical view (SWT version). Scaling, manipulating the intervals, tree structure. The sum interval is rendered by a specialized sum renderer. The arrows between the intervals are a demonstration of the use of relations. Context menus on scale and hierarchy. Hierarchy using label provider including icons. A marker has been added. The y axis width is set to zero leaving only the hierarchy area visible.

Can be switched (check the source code) to demonstrate variable xscale scaling. If enabled a row containing intervals determining regions of different scaling is shown as the last row. The pps intervals can be manipulated.

5.2. Overlapping intervals and DND

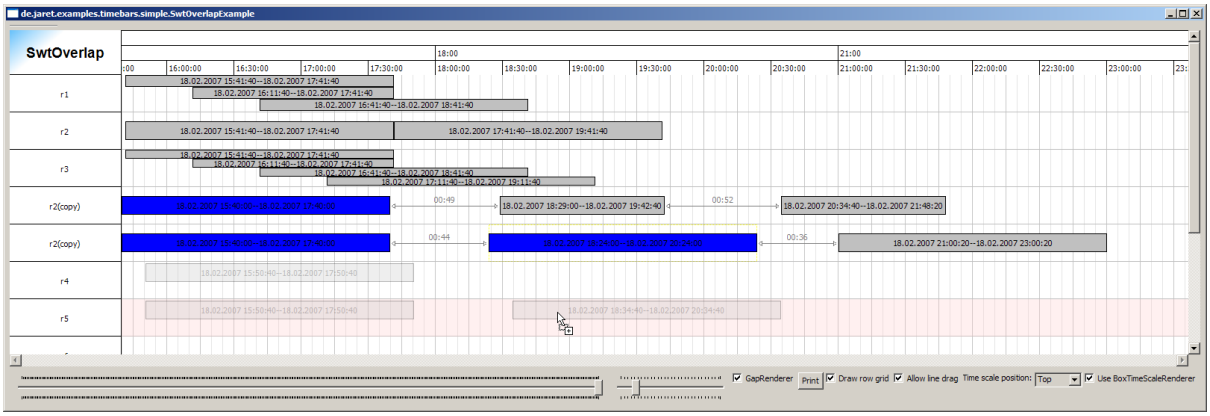

Figure 5.2. Non overlapping with drag in progress

This example demonstrates the handling of overlapping intervals if drawOverlapping is false. Every interval will only get a fraction of the row space for rendering. Renderer is the simple default renderer. Possibility to change the intervals (dragging, resizing). Possibility to change the scale. Printing possibility. Other renderers: default.

The example also contains an example of defining a drag source and a drop target on the timebar viewer. The majority of the functionality is implemented outside the timebar viewer. This takes into account, that there are many possibilities to define the semantics of drag and drop.

5.3. calendar example

The calendar exmple (de.jaret.examples.timebars.calendar) uses the vertical orientation to display an outlook-like day calendar where each day is presented in a vertical timebar row. The intervals used in the model project all appointments to the same day thus they are drawn right beside each other if they overlap in time.

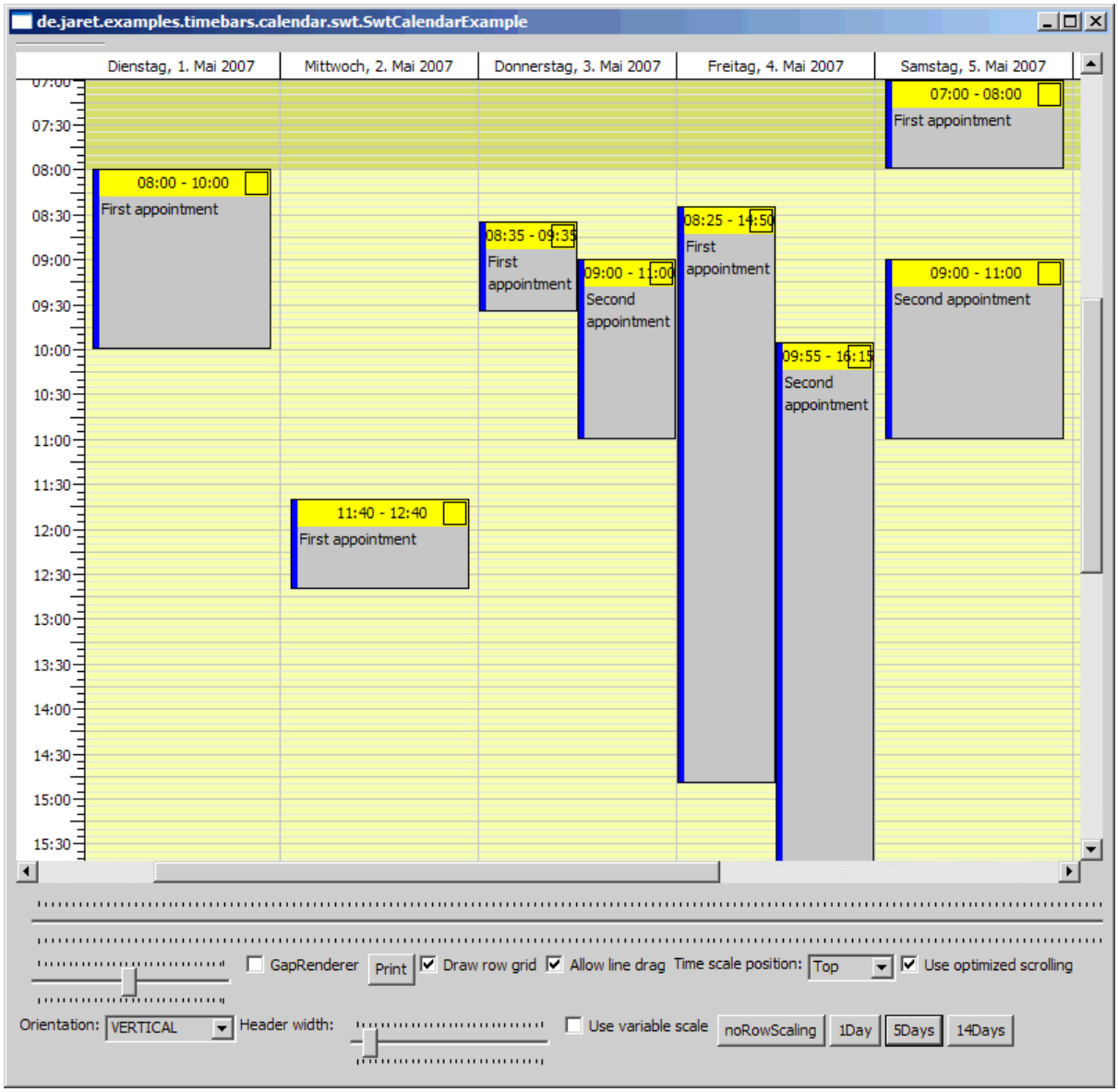

Figure 5.3. Calendar example (SWT version)

The variable time scale feature can be used to compress non working time in the calendar display. An interval modificator makes sure that intervals can not be extended over day boundaries.

Note

The calendar example is the base for the jaret calendar plugin for eclipse that connects to the google calendar. The jaret calendar will be published (soon) on xpmp.de.

5.4. fancy example

The FancyExample (SWT only) demonstrates some techniques for advanced drawing. This includes drawing outside the bounds of the rectangle given by the begin and end timestamps of an interval.

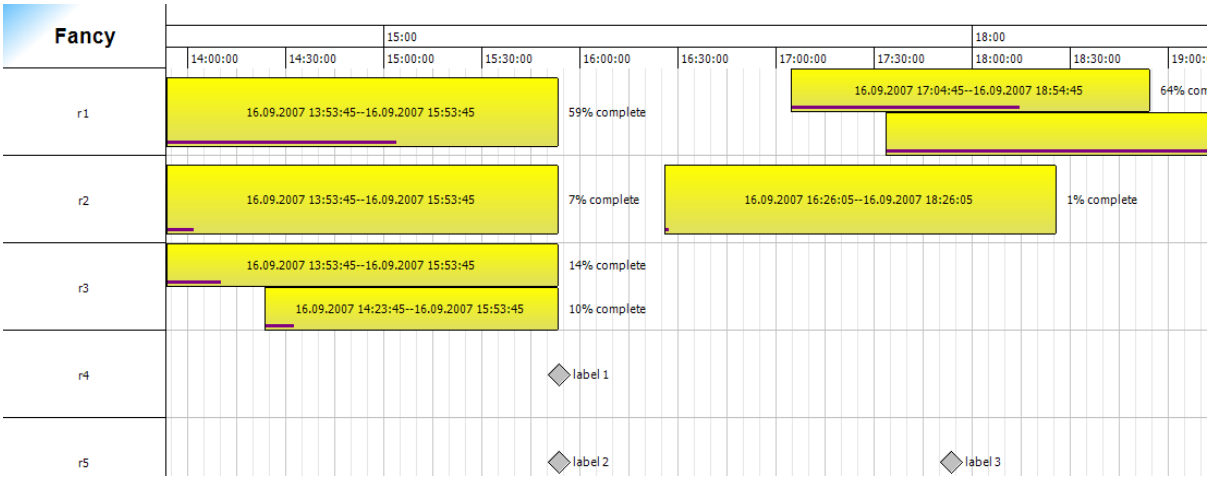

Figure 5.4. FancyExample (SWT, Version 1.13, no effects)

The examples demonstrates the use of the TimeBarRenderer2 interface that adds a method to calculate the preferred size for rendering to the old renderer interface. This open the opportunity for the renderer to request for more unclipped space when rendering. The FancyRenderer just adds label right to the rendered interval area. The FancyEventRenderer draws a single point in time as a diamond plus an additional label.

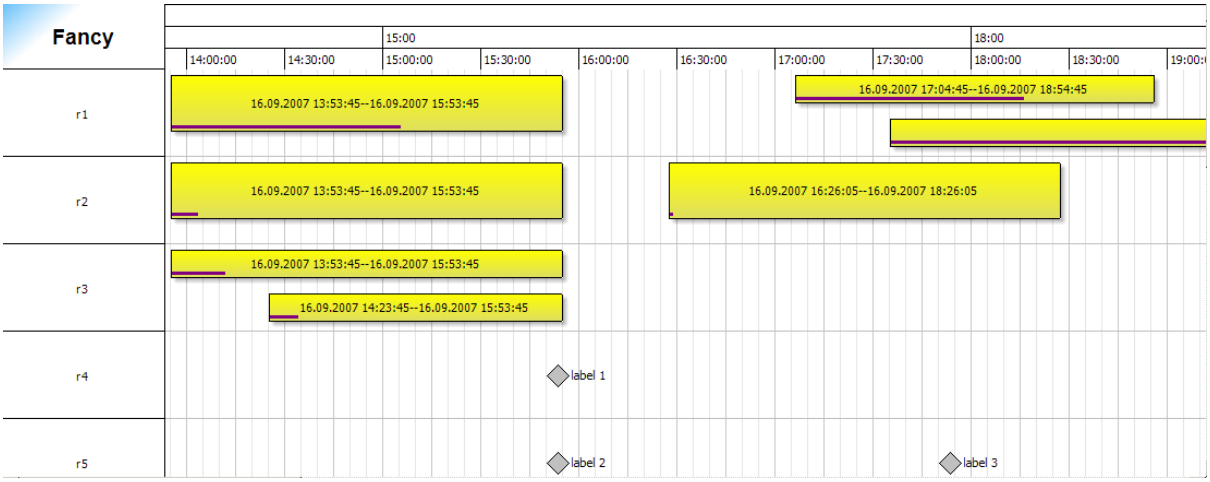

Figure 5.5. FancyExample (SWT, Version 1.13, drop shadow)

The FancyIntervalRenderer can be parameterized to draw a drop shadow for the intervals. The method to draw the drop shadow has been taken from the article by Nicholas Rajendram, IBM Canada, see *http://www.eclipse.org/articles/article.php?file=Article-SimpleImageEffectsForSWT/index.html* [http://www.eclipse.org/articles/article.php?file=Article-SimpleImageEffectsForSWT/index.html].

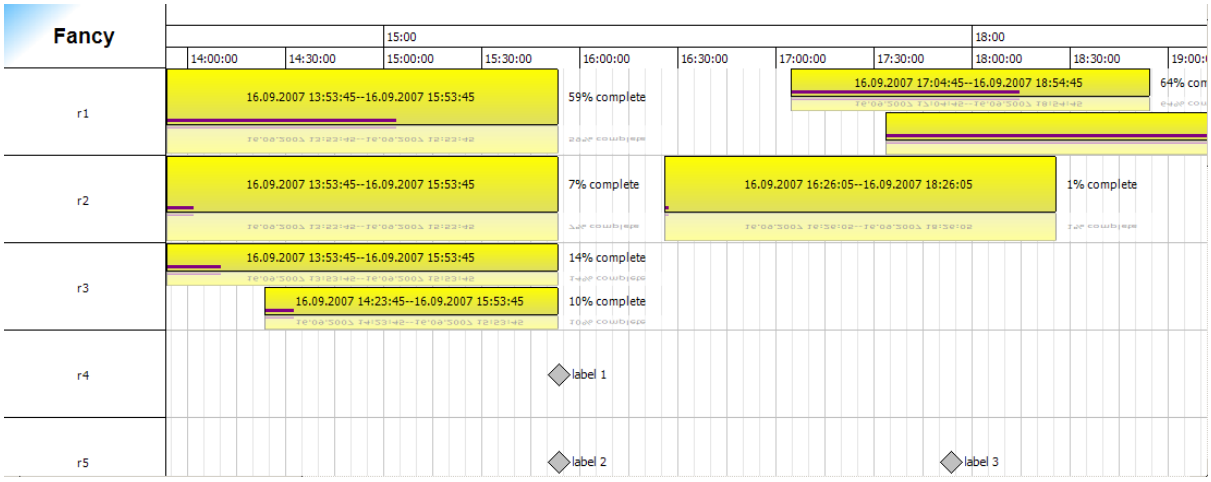

Figure 5.6. FancyExample (SWT, Version 1.13, reflection)

The FancyIntervalRenderer can be parameterized to draw a reflection of the rendered interval. The idea was taken from Daniel Spiewak: see *http://www.eclipsezone.com/eclipse/forums/t91013.html?start=0* [http://www.eclipsezone.com/eclipse/forums/t91013.html?start=0].

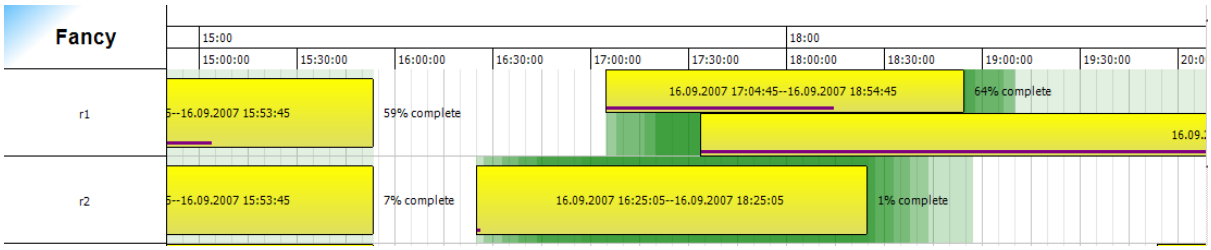

Figure 5.7. FancyExample (SWT, Version 1.13): global rendering before interval rendering

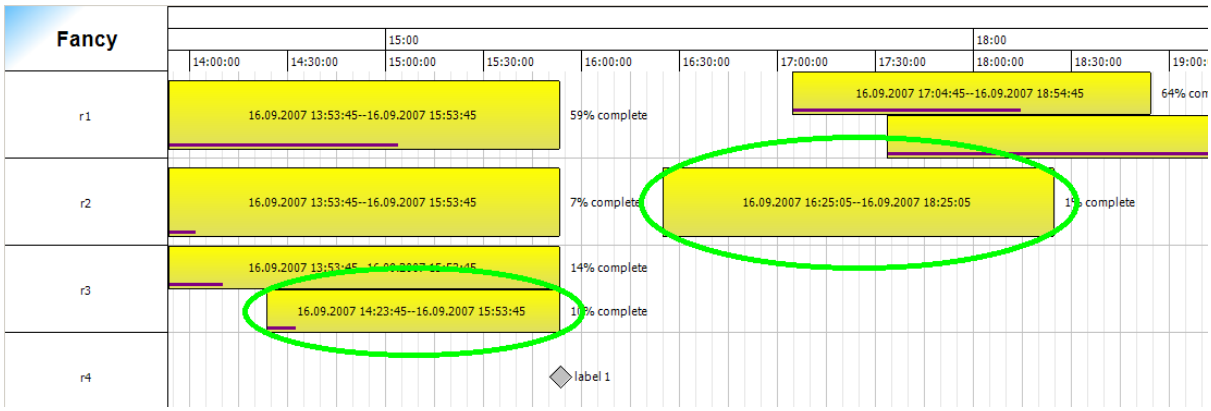

Figure 5.8. FancyExample (SWT, Version 1.13): global rendering after interval rendering

The example also demonstrates the use of a GloablAssistentRenderer to render a history of moved fancy intervals by drawing an alpha shaded background before the intervals are painted and by drawing a special selection mark after the intervals have been painted.

Note

The additional effects are not performance optimized and might cause quite a bit of cpu time consumption.

5.5. timeline example

The timeline example is inspired by the SIMILE timeline (http://simile.mit.edu/timeline/). It features two time bar viewers rendering the same model in different scales. The viewers are coupled so that scrolling is reflected in the other viewer. Scrolling can be done by dragging in the viewer and the time scale. Rendering has been adapted for the different scales.

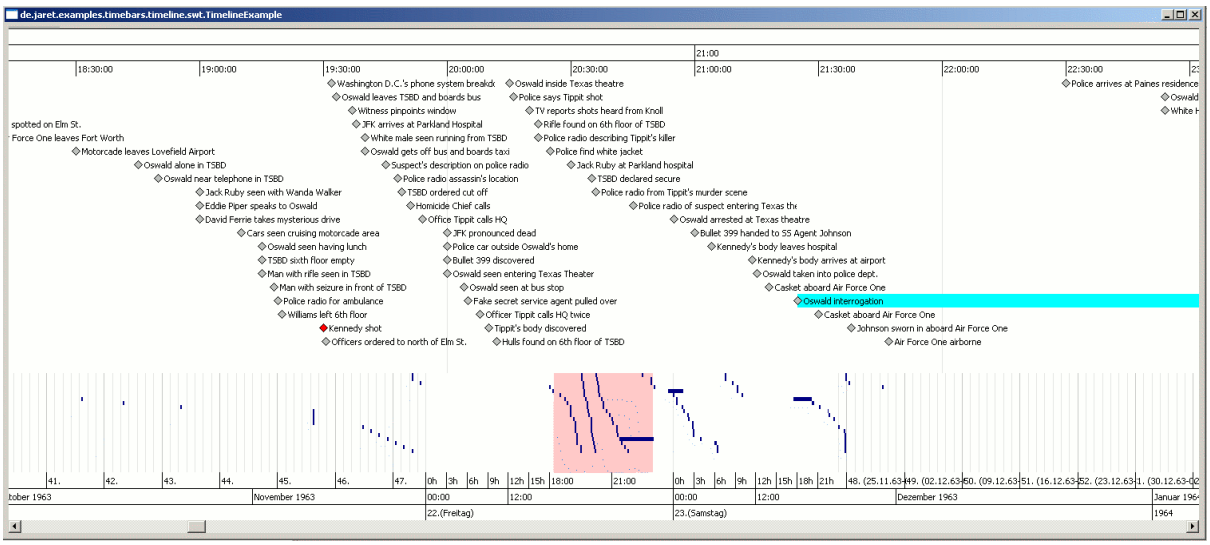

Figure 5.9. Timeline example

The timline example makes use of an externalized overlap strategy that simply distributes the events that are all stored in a single time bar row over a number of virtual rows without further checks. It also uses a row height strategy to keep the one row always as high as the diagram area available.

5.6. EventMonitor example

The event monitor is showing immutable events in several streams.

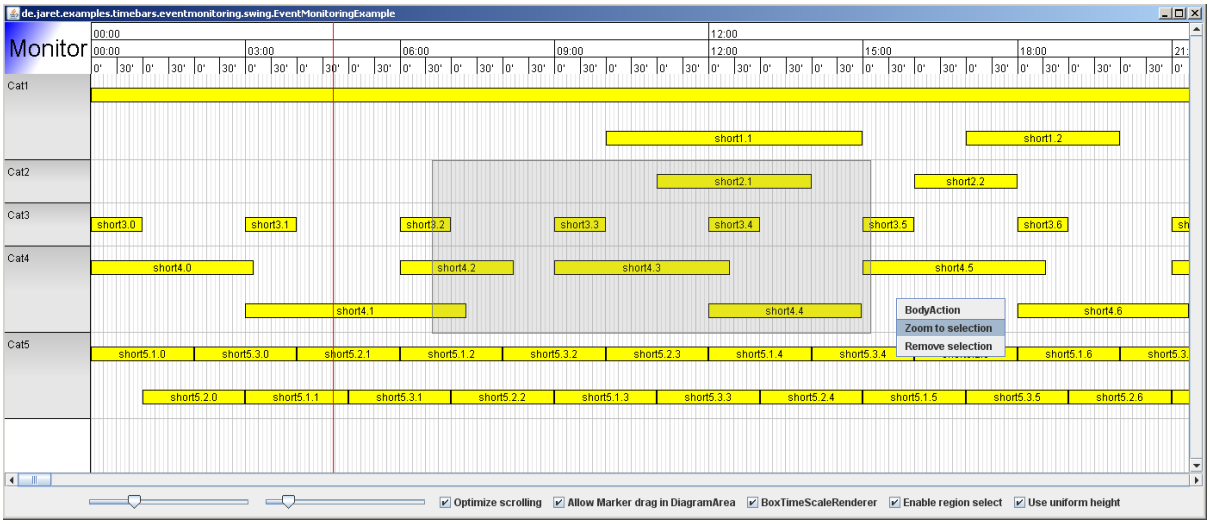

Figure 5.10. EventMonitor example

The event monitor example is a swing example showing severl techniques:

- Region selection and atached zoom action
- row height startegy sizing the rows height to the number of overlapping intervals
- uniform height for all intervals in a row with overlapping intervals
- custom (nicer) header renderer for Swing
- Swing BoxTimeScaleRenderer
- simple DND application for Swing (Drag intervals/rows out of the renderer)

5.7. Linechart example

Since all rndering is based on the timescale it is quite easy to use the timebars to visualize other time related data like a linechart. The rendering is done inside one interval, so it can mixed with a gantt chart and can be synchronized to a gantt chart using a synchronizer.

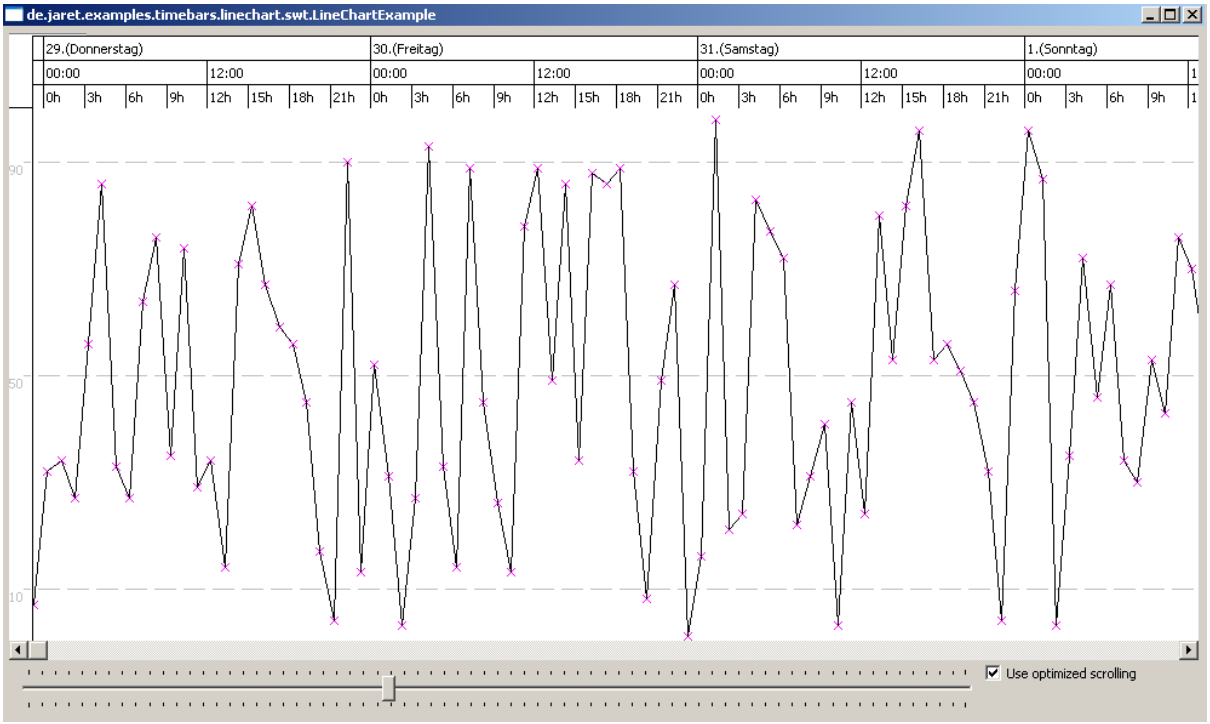

Figure 5.11. LineChart example

Chapter 6. Appendix

6.1. Known bugs and limitations

This section covers some known problems on different platforms.

- When using models that span a large time optimized scrolling fails. Simple workaround: setOptimizedScrolling(false). The cause is an integer overflow.

6.1.1. Linux/GTK

- For some weird reason ghost intervals can not be painted if a highlighted row is painted using transparency. Solution: The highlighting of a row will be done with alpha=100 when ghosted intervals or rows are present and the platform is gtk.

6.2. Changelog since version 1.0

This section contains the changelog of the timebars component since version 1.0 for reference.

6.2.1. 2013-09-17 version 1.49

- Switched the build to maven 3 (looks good so far)
- Removed some forgotten (ouch!) System.out.println statements (Sorry!)
- Swing: corrected clipping width/height for gap rendering (Thx Chris!)
- Swing+SWT: Corrected gap rendering (Thx Chris!)
- Swing: DefaultHeaderRenderer now features configurable width (Chris)
- Swing: The inner JComponent used for rendering is now accessible (_tbv._diagram) - this allows adding listeners directly to the component getting the events (Hi Jordan!)
- Swing: configurable mousewheel scrolling: see setAllowMouseWheelXAxisScrolling, setAllowMouseWheelYAxisScrolling
- When using millisecond accuracy there is now a scaling for the time scrollbar in place thus the limit that has been imposed on scrolling with millisecond accuracy is gone (Limit has been imposed by the int value of the scrollbar handle).

6.2.2. 2012-02-09 version 1.48

- Swing: Fixed another clipping issue that caused out of bounds drawing of the row header (Thx William)

6.2.3. 2012-02-08 version 1.47

- Swing: Fixed a clipping issue that caused out of bounds drawing of intervals (Thx William)

- Swing: fixed title renderer paint problem (Thx Chris)

6.2.4. 2011-11-06 version 1.46

- Fixed NullPointerException when setting row height before setting a model (Thx Patrick)
- SWT: fixed call to super.dispose() in onDipose (thx Chris)

6.2.5. 2011-10-04 version 1.45

- IMPORTANT:
  Since Eclipse 3.7 introduced getOrientation in the component I had to refactor the old set/getOrientation methods to set/get
  TB
  Orientation This change has to be reflected in the timebars addon -> check the update if you need the addons.
- Swing: improved scroll units (Thx Chris)
- Swing: included a fix (Thx Chris) making the distribution of intervals better when using uniform height.
- Fixed an issue that caused an incorrect behavior when applying a filter to a scrolled viewer using variable row heights (Exception) (Thx Chris)
- Fixed exception when using dragging of row heights/lines and no rows present (Thx Chris)
- Swing: Fixed a drawing bug that caused drawing of the last row on top of the first row (variable row heights only) (Thx Chris)
- Swing: Fixed a rendering bug for the title renderer component when using other layout mangers than border layout or box layout (Thx Chris)
- Fixed a quite obvious bug in delegate.getRowBounds that caused the height of the returned rectangle to be to large when using horizontal orientation (Thx Chris)
- Fixed Delegate.yForRow to handle variableHeights+no rows displayed correctly (Thx Chris)
- Swing: fixed Pop-Up handling on Linux (Thx Patrick!)
- Swing: Fixed a clipping bug that caused huge intervals to be rendered outside the viewer (OSX, Linux)
- Swing: Fixed a clipping bug that caused caused a minor glitch when scrolling "under" the header (Thx Chris)
- SWT: New combining TimeScaleRenderer that allows the use of two time scale renderers. New simple DateStripRenderer that can be used to maintain a steady view of the starting date. TimeScaleRenderer interface has a new simple to implement method: supportsOptimizedScrolling - check the JavaDoc.
- Swing: (Thx David): Fixed SWT import
- Exposed scrollXXXtoVisible methods in the viewers (check the delegate if something is missing)

6.2.6. 2010-08-18 version 1.44

- Fixed a glitch in edge dragging: coordniates used for drag detection differed from the coordinates th euser pressed the button. This lead to strange behaviour: cursor indicates edge drag, press, nothing happens.

- Swing: (Thx Chris): Fix a problem where rows would sometimes not be drawn when scrolling
- Right click/PopUp behaviour: if interval right clicked on is not already selected use this as the only selected one (-> Context menu works properly) (Thx Chris)

6.2.7. 2010-06-14 version 1.43

- Fixed a printing bug: grid has only be printed for the range currently displayed. There might still exist a problem with printing when the timescale renderer is not the ITickProvider used (not very probable).
- Fixed a bug that occurred when resizing the component without rows in the model (thx Chris)
- Fixed popup (right click behaviour) (thx Chris)
- Swing: added the possibility to use a title renderer directly as a component (not just painting; see scheduling example)

6.2.8. 2009-12-31 version 1.42

- Added a new Swing example: the SchedulingExample.
- Internal, unpublished

6.2.9. 2009-11-13 version 1.41

- Did some small changes to setLastRow and the component resize behaviour since there have been problems using the Swing version. If you rely on this (i.e. vertical autoscroll) please check carefully.
- Removed two forgotten System.outs (Ughh!)
- Fixed a possible NPE in Swing(Headertooltip)

6.2.10. 2009-11-02 version 1.40

- Fixed resizing behavior when the component exposes space below the last row the space will be used rendering more rows from above
- Did another small fix correcting setLastRow
- Fixed a bug that sets the minimum gridsnap for dragging and moving intervals to 1 second preventing clean operation at millisecond accuracy. This fix may surprise you if you have not setup an IIntervalModificator and you are using the DefaultIntervalModificator. You may need to set the gridsnap to 1 second explicitly.
- Fixed some problems that may arise if non default TimeScaleRenderers are used without non default GridRenderers (Swing and SWT): Since the DefaultGridRenderers will use the the ticks of the TimeScaleRenderes if the the TimeScaleRenderer implements the ITickProvider interface to synchronize the ticks, it had to be informed properly when another TimeScaleRenderer has been set that does not supply ticks or when the time scale should not be shown.

6.2.11. 2009-10-08 version 1.39

- Added the possibility to do a DST correction in the time scale renderers (Default and Box: btsr.setCorrectDST). This is a fix for shifted ticks in the time scale around the DST switches in spring and autumn.
- Added the method `getPopUpInformation()` that supplies the row and date corresponding to the coordinate a popup menu will be displayed.

6.2.12. 2009-09-23 version 1.38

- Fixed: NPE in Swing version if a no context menu has been defined for an interval type
- Fixed: Bug in scrollRowToVisible when using variable row heights (prevented the vertical autoscroll sample code to work with variable row heights)
- Fixed: Swing context menus using mac OS X (weird: "In Mac OS X, the pop-up trigger is set on MOUSE_PRESSED. In Windows it is set on MOUSE_RELEASED. For portability, both cases should be considered.")
- Swing: added support for drawing outside the interval bounds for the TimeBarRenderer (see getPreferredDrawingBounds). This has been around for SWT before ... an example for Swing is the SwingEventExample (TBD for SWT, the feature is implemented for SWT: see the FancyExample). The feature is still in development (as is the example). The changed interface will cause Swing users the inconvenience to implement the method ... which is trivial as can be seen in the DefaultTimeBarRenderer.
- Swing: the scroll bars are now placed on panels that are accessible by getters (getHorizontalScrollPanel, getVerticalScrollPanel). This allows special extension to be placed in the scroll bar area.
- The new interface IIntervalModificator provides a small extension over the existing IntervalModificator allowing control of the grid snap for single intervals.

6.2.13. 2009-09-03 version 1.37

- Fixed: setSecondsDisplayed will now work with vertical orientation
- Fixed: vertical autoscroll sample code in SwtOverlapExample: +-1 problem -> could not scroll to the last row

6.2.14. 2009-08-17 version 1.36

- Swing: removed renderer delegation from the DefaultRenderer (not necessary - register your renderers with the viewer directly)
- Added better control for the displayed rows: setLastRow. This is in conjunction with a bugfix in scrollRowToVisible and a possible vertical autoscroll solution for drag+drop (can be found in the swt overlap example)
- When dragging an interval the bounds are now set in a sequence that ensures that end will always be > begin
- setSecondsDisplayed is now safe to be used even when the viewer has not been painted. The calculations will be deferred until the component is painted. Additional parameters (centering, centerDate) will be ignored in this case, since the calculations are not possible.
- Fixed a weird NPE occurring with Swing/GTK during construction of the timebar viewer

6.2.15. 2009-06-23 version 1.35

- SWT: Fixed a small bug in the RelationRenderer (NullPointer under rare conditions)
- Added synchronization for the y axis width in the synchronizer

6.2.16. 2009-05-20 version 1.34

- SWT: Fixed a small bug that could happen when displaying a context menu

6.2.17. 2009-05-07 version 1.33

- Added a feature for setting the initial displayRange when the viewer is first displayed and the width of the widget is available. See `setInitialDisplayRange`
- Fixed a bug in the DefaultTimeBarRowModel that prevented the correct update of the max date if an interval would change min AND max date.
- Fixed a bug in the Swing DefaultTimeScaleRenderer (refactoring artefact, thanx Ravi).

6.2.18. 2009-02-22 version 1.32

- BoxTimeScaleRenderer now scales down from years to minutes/seconds/milliseconds
- Swing and SWT: The DefaultTimeScaleRenderer has been replaced by a new one that scales from years to milliseconds. If your application needs the old default renderer you can set the old renderer explicitly (it is still in the distribution: OldDefaultTimeScaleRenderer).
- Internal coordinate calculation will now always use milliseconds as the basis. This should not cause any problems. However, if you encounter problems, please report the bug.
- Fixed a small bug in tooltip handling.
- Swing: added the IMarkerRenderer for better control of how markers are painted.
- Swing: added the IGlobalAssistantRenderer for misc rendering.
- New methods for scaling: setSecondsDisplayed(...); the scaling can be centered around a date
- GridRenderers now have to accept an ITickScalProvider to achieve a coupling between time scale renderers and the grid rendeering. The default grid renderer will use the supplied tick scale provider. The BoxTimeScaleRenderer and the DefaultTimeScaleRenderes do implement the ITickProvider interfaces.
- Extended ITimeBarChangeListener: marker drags will be reported.
- The variable x scale feature now supports "breaking" the timescale by marking pps intervals as breaks and defining a pixel width for the display. See the milli example for details (documentation follows).

6.2.19. 2009-01-11 version 1.31

- Fixed a bug in the overlap calculations that lead to intervals not properly rendered (rendered above the row) in some data constellations. (Thanks Daran!)

- Clipping on Linux GTK seems to be implemented using only 16 bit integers. This results in random drawings if a high zoom factor is used in combination with long intervals or relations (the drawings are not really random, since they are just caused by overflowing 16 bit integers). Added a workaround in the relation renderer and the sum renderer of the swt hierarchy example.

6.2.20. 2009-01-01 version 1.30

- SWING: Did a lot of corrections and additions for the swing component so that it nearly catches up to the SWT variant:
  - Ported the BoxTimeScaleRenderer to Swing
  - New Swing example: EventMonitorExample demonstrating some of the new features mentioned below.
  - Added the title renderer for swing and supplied a simple default title renderer.
  - Added relation rendering in the swing world (ported over the relation renderer from SWT).
  - Added the context menu support.
  - Fixed several minor bugs.
- Introduced findbugs in the maven build (surprisingly low number of issues) and fixed some more important issues reported (nothing really serious).
- Introduced the region selection: a selection that is tied to time and rows that stays present and can be used for several purposes. The functionality has to be enabled (`setRegionRectEnable`). Region selections can be done with shift+click+drag. The region selection will persist until it is cleared (`clearRegionRect`) or replaced by the next selection. As an example a simple zoom action has been implemented in the EventMonitorExample (Swing) and the SwtHierarchy example.
- Introduced a new listener for listening to the selection rect/region selection allowing online information about ongoing rectangle selections: (`ISelectionRectListener`)
- Introduced the IMiscRenderer (Swing and SWT) that will collect rendering routines for some elements/parts in the timebar viewers (to be extended over time).
- Added the possibility to hide the root node when using a hierarchical model (`setHideRoot`).
- Added the possibility to draw intervals overlapping in selected rows (`ITimeBarViewState.setDrawOverlapping(row)`).
- Added the option to allow dragging of markers not only in the time scale area (`setMarkerDraggingInDiagramArea`.
- Fixed: when using vertical orientation, optimized scrolling and timescaleposition = bottom the scale would not be scrolled.
- Attention:(SWT) Removed the deprecated stuff (registering further renderers) from the default renderer. You can easily register specialized renderes with the TimeBarViewer directly.
- Added the possibility to enforce uniform heights/widths for intervals drawn non overlapping in the same row (see `setUseUniformHeight`).

6.2.21. 2008-11-08 version 1.29

- New methods to easy scrolling to elements: `void setFirstRow(TimeBarRow row)` scrolls to a row without requiring its index. `void scrollIntervalToVisible(Interval interval, double horizontalRatio, double verticalRatio)` scrolls an interval to a position in the viewable

area.

- Fixed a bug in AbstractGridRenderer (static reference to a Color that got disposed); Thanks Thomas

6.2.22. 2008-09-20 version 1.28

- Behaviour on focussing/selecting an interval: was scroll begin to visible: this is now refined to scroll only when no part of the interval is visible. Scrolling on focus change can be disabled completely (setScrollOnfocus).
- improved printing when using linux: swt printer devices running on linux seem to always report 72x72dpi which resulted in bad output when using the standard renderers. The RendererBase now contains the method `getDefaultLineWidth` to get a corrected line width that will work with low resolutions.

6.2.23. 2008-08-19 version 1.27 (not published)

- Behaviour on focussing/selecting an interval: was scroll begin to visible; now configurable if scrolling should occur (setScrollOnfocus). Maybe this will be replaced by a strategy if necessary.

6.2.24. 2008-06-22 version 1.26

- Fixed: Before the first paint (sometimes) NullPointerExceptions could happen because of uninitialized layout
- Fixed: NullPointer in getTooltipText (sometimes)
- Fixed: Relation rendering: arrows when using end_end.

6.2.25. 2008-05-29 version 1.251

- Just added a new example: The linechart example: check the screenshots.

6.2.26. 2008-05-04 version 1.25

- introduced relations between intervals and relation rendering and selecting. (IRelationalInterval, IIntervalRelation, IRelationRenderer, implementations thereof). There is a default implementation of the IRelationRenderer that is not setup by default (RelationRenderer). Changes have been made to the selection model (can now contain selected IIntervalRelations) and the selection provider (can now have relations in the structured selection). Unless you use relations you will not have to make changes to existing code. The default relation renderer does not support vertical orientation. The usage can be studied in the SwtHierarchyExample. The relations are observables but have not been added to be observed by the viewer (this would have introduced major API changes), so for now a redraw has to be triggered when the relation itself changes (this is most probably a rare case).
- fixed a small bug in the box time scale renderer (week label included first day of the next week). Bug has been fixed in the jaret utils package (Thanx Thomas)

6.2.27. 2008-04-25 version 1.24

- added some changes to the row selection behaviour (shift-click range select, toggle mode for row selections that can be set on the selection model for simulating check box selections). Check box selections and the toggle mode can be seen in the swt overlap example (activate in the control panel).
- the rendering of row selection and highlighting has been moved to the grid renderer (so it is better customizable). This obviously breaks the API. For a quick fix reserving the current behaviour just extend AbstractGridRenderer when you have your implemented own. The grid renderer interface will be a bit more extended in future versions to support some other things that have not been easy customizable. The getters/setters for highlight color etc. in the timebarviewer have been deprecated but will still work with the every extension of the abstract grid renderer.
- the grid rederer now supports an alpha setting for the row selection

6.2.28. 2008-04-17 version 1.23

- made the timebarviewer friendlier to GUI-Designers (will hopefully not throw exceptions without a model)

6.2.29. 2008-04-12 version 1.22

- Removed a dependency on Java 6 in the DefaultOverlapStrategy (Thanks Maarten)

6.2.30. 2008-04-06 version 1.21

- Fixed a bug that have been introduced with the autoscroll improvements: When using the grid snap dragging an interval has not been working.
- Th egrid snap provided by the IntervalModificators is now applied specifically: the first modificator that claims to be applicable will provide the grid snap. That makes it possible to use different grid snaps for different types of intervals.

6.2.31. 2008-03-30 version 1.20

- Improved the resize detection/beviour: increasing the size of an interval has sometimes behaved unexpected, since the cursor had to be inside the interval.
- Autoscrolling has been quite fast (since it was accellerating with the scrolling). This has been fixed. It is now possible to set the autoscrollDelta (in pixel) to ensure a controllable drag.
- The timebar viewer will now drag all selected intervals if an interval is dragged and the option is enabled. Autoscrolling is focussed on the actual interval that is beeing dragged. If one of the intervals is not allowed to move to the new position, the whole drag is stopped. This behaviour can be enabled by setting setDragAllSelectedIntervals(true).

6.2.32. 2008-03-22 version 1.19

- Improved calendar example to show whole day appointments in header area.
- Thanks go to Mathias Kurth for supplying an improved overlap calculation drastically decreasing the time needed. This is especially helpful when handling huge quantities of intervals. (Added a link to his work on the usage page)

- Added a method to access the delegate in the viewers. Please note that the direct use of the delegate ist not ecouraged unless it is absolutely necessary.
- Added the RelationRenderer to render intervals having a relation (IRelationalInterval). The relation rendering is work in progress.

6.2.33. 12/27/2007 version 1.18

- Internal version distributed with the jare calendar plugin
- Bugfix: Interval modification failed in some special cases when using more than one IntervalModificator.

6.2.34. 12/22/2007 version 1.17

- Bugfix: (non variable time scale) Scroll bounds have not been correct (small difference) corresponding to the model
- Bugfix: just another small scrolling bug (using page steps with optimized scrolling)
- Small change in the IntervalModificator interface (Sorry for that!) allowing better differentiation when using heterogenous models. (new Method isApplicable(...) to check whether the modificator is responsible -> just return true to preserve current behaviour.)

6.2.35. 11/20/2007 version 1.16

- Bugfix: Scrolling with the time scroll bar using page steps failed when using optimized scrolling.
- Bugfix: Single selections have been disabled by the selection behaviour if other intervals have been selected ... corrected

6.2.36. 11/03/2007 version 1.15

- Small improvements for scroll bars (right sized from the first appearance, max data check corrected)
- Fixed several bugs when adding/removing nodes in a hierarchical model. (For test purposes the SWT Hierarchical example can now be switched to support drag and drop. Check the source code for the flag)
- The default hierarchy renderer can now handle different label providers for different row/node implementations
- Improved selection behaviour: when doing a multiselection the next single selecting click did reset the multi selection. This has been changed to do the single select on mouse release when intervals are selected. Thanks to Martin Schmidt for pointing that out.
- Added a new example showing events in a timeline fashion (s. screenshots). This has been inspired by the simile timeline for html pages (http://simile.mit.edu/timeline/).
- Externalized the strategy for calculating the overlap information. This allows implementing strategies that can support lareg numbers of massively overlapping intervals without beeing too slow. See the timeline example for a sample of an alternative implementation.
- BoxTimeScaleRenderer improved to handle variable scaling by adapting the scale

- Improved TimeScaleDragSupport to allow listening on diagram drags for scrolling
- Some small API extensions (non breaking)

6.2.37. 09/22/2007 version 1.14

- Fixed "row selection not working"-Bug
- Fixed some Tooltip-Bugs (vertical).

6.2.38. 09/16/2007 version 1.13

- Fixed a small bug that lead to a paint error for partly painted intervals when using variable scale
- The BoxTimeScaleRenderer now supports vertical orientation (might be time consuming since vertical oriented text is)
- SWT: rendering outside the bounds given by the interval is now supported. See the extended renderer interface TimeBarRenderer2 for details on how to do this from your renederer. This has a little impact on rendering optimizations so it can be turend off. See TimeBarViewer#setStrictClipTimeCheck for details.
- SWT: Added an example doing some more fancy drawing. Demonstrating the new feature of drawing outside the core interval space (FancyExample).
- Added a linking exception to the GPL license that allows linking to other freely available software (source code freely available)

6.2.39. 08/19/2007 version 1.12

- Added variable row heights/widths including the possibility to change the height/width of a row by dragging or the use of a calculation strategy (see ITimeBarViewstate).
- fixed some ends concerning vertical orientation (still the interface might be quite irritating when thinking vertical)
- SWT: printing fixed for windows, horizontal orientation, including support for variable row heights

6.2.40. 07/18/2007 version 1.11

- Fixed a bug that prevented the SWT version from drawing overlapped intervals without height correction.
- Added a listener that is informed while intervals are modified by the users (ITimeBarChangeListener).
- Began a list of known usages of the timebars component (see the jaret website)

6.2.41. 06/05/2007 version 1.10

- *This version incorporates some bigger changes. Please regard it as a snapshot! Bug reports welcome.*
- introduced vertical orientation. Modifications have been done carefully, the semantics of some API calls changed organically (rows now can be columns depending on the orientation ...). Support for vertical render-

ing has been added to most of the default renderers (Swing and SWT). (Some examples allow switching orientation; a special example has been added: the calendar example, see screenshots).

Note: Hierachy renderers have not yet been enhanced to support vertical orientation --> so dont use a hierarchical viewer with vertical orientation by now.

- added support for internal row (column) scaling: if set the viewer adapts the row height to always display a fixed number of rows (columns) (see setAutoRowScaling)
- Scrolling optimizations fixed for SWT (tested Win, Linux/GTK, OSX/Intel). Scrolling optimizations are now *ENABLED* by default. If you experience any artefacts when scrolling, disable scroll optimizations (setOptimizedScrolling(false).)
- By mistake and stupidness the TimeBarViewerDelegate referenced SWT classes rendering it unusable for Swing. Fixed that (Thanks Tim!)

6.2.42. 05/01/2007 version 1.02

- SWT&Swing: Introduced scrolling optimizations (copying previously drawn areas). The behaviour is switchable (setOptimizedScrolling) and defaults to false, since I noticed problems using SWT on Linux/GTK and OSX/Intel. However the performance gain on Windows XP is quite big if a lot or fancy diagrams are drawn. For Swing the optimizations do work on all of the platforms. The scroling optimizations can not be used together with a variable x scale.
- Added a method to the DefaultTimeBarRowModel that allows addition of more than one interval at a time. (Pls. note that the default implementation is an implementation that can easyly - and sometimes should - be tailored for the needs of the cocrete application). Also added methods for removal of more than one interval at a time.
- Fixed issues with non overlapped drawing and the dynamic use of an interval filter.
- Enhanced non overlapped drawing: in some cases the the result wasted some place.
- Fine tuned interaction between model and selection: Intervals removed from the mdoel will be removed from the selection. (This is not implemented for removeing lots of intervals at once, sice it would sacrify the peformance gained by removing all intervals at once)
- Fixed a small bug that occured when an interval was selected (apparent only on large scales, caused scrolling the viewer without need).
- Fixed updating of the vertical scroll bar (to short in some situations)
- introduced the timebars.addon package

6.2.43. 03/12/2007 version 1.01

- small correction to the default hierarchy renderer and the title renderer so that it accepts image descriptors instead of ressource paths
- a bug in the DefaultTimeScaleRenderer fixed (drawing was not complete)
- fixed a bug that prevented all intervals from beeing drawn when showing a long period of time (small pps)

6.3. Licenses

The jaret timebars component is dual licensed. Contact `<peter.kliem@jaret.de>` for any inquiries, ques-

tions or suggestions concerning the license. For the use of the component under the GPL within other free projects an exception has been designed (see below) that allows the usage of the timebars component together with other licenses as long as the other software is freely available in source from.

6.3.1. GPL

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those

sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License. Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs. If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library. If this is what you want to do, use the GNU Library General
Public License instead of this License.

6.3.2. Linking exception to the GPL

Linking this library statically or dynamically with other modules is making a
combined work based on this library. Thus, the terms and conditions of the
GNU General Public License cover the whole combination.

Nothing in this Exception gives you or anyone else the right to change the
licensing terms of the jaret timebars component.

As a special exception, the copyright holders of this library give you permission
to link this library with independent modules, if
- all linked code is available in source code form free of charge
- and if all linked code is licensend under one of the following licenses:
Eclipse Public License 1.0 (EPL), Common Public License 1.0 (CPL),
GNU Library or "Lesser" General Public License (LGPL) 2.0/2.1,
Apache License 2.0, Common Development and Distribution License (CDDL)

6.3.3. Jaret commercial license

Warning

The following commercial license is *only* applicable if purchased!

Commercial License agreement for the "Jaret Timebars" GUI Component (ONLY APPLICABLE IF PURCHASED) 07/2006

This license agreement covers the commercial use of the GUI Component "Jaret Timebars" (only applicable if purchased; GPL has to be applied otherwise) furthermore referenced as the SOFTWARE. This is a legal agreement between you (the individual or corporation who purchased a commercial license and who operates the software) and Marcus Thyssen Softwareentwicklung.

The SOFTWARE is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE is licensed, not sold.

1. GRANT OF RIGHTS You are granted the following rights - Incorporate the software in applications you build. - Modify the source code of the component for use in applications you build. - Sell applications royalty-free including the software (may include the source code or the modified source code of the component). Your customers do not get the right to include the software in applications they build. They may however modify the source code in the context of the application sold by you and they may sell the applications built by you containing the software.

There is no limitation on the number of applications you build incorporating the component. Installation and usage by an unlimited number of developers of your company is allowed. If you introduce a third party in the development process the third party may use the software in the context of the application you build.

2. RESTRICTIONS Neither you nor your customers may sell the software as a component or as part of a component library or as part of a framework.

3. TRANSFER OF THE LICENSE You are entitled to transfer this license to another individual or corporation if and only if you have not build and sold an application including the component.

4. WARRANTY DISCLAIMER THE SOFTWARE IS PROVIDED ON AN "AS IS" BASIS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. YOU ARE RESPONSIBLE FOR THE ENTIRE RISK WITH RESPECT TO THE QUALITY AND PERFORMANCE OF THE PROGRAM. Marcus Thyssen Softwareentwicklung DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE PROGRAM WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATIONS OF THE PROGRAM WILL BE UNINTERRUPTED OR ERROR FREE, OR THAT DEFECTS IN THE SOFTWARE WILL BE CORRECTED. FURTHERMORE, Marcus Thyssen Softwareentwicklung DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGAR USE OR THE RESULTS OF THE USE OF THE SOFTWARE, INCLUDING BUT NOT LIMITED TO THEIR CORRECTNESS, ACCURACY, RELIABILITY OR EFFECTIVENESS. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY Marcus Thyssen Softwareentwicklung SHALL CREATE A WARRANTY OR IN ANY WAY INCREASE SCOPE OF THIS WARRANTY. SHOULD THE SOFTWARE PROVE DEFECTIVE, YOU (AND NOT Peter Kliem) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

SPECIFIC DISCLAIMER FOR HIGH-RISK ACTIVITIES. The SOFTWARE is not designed or intended for use in high-risk activities including, without restricting the generality of the foregoing, on-line control of aircraft, air traffic, aircraft navigation or aircraft communications; or in the design, construction, operation or maintenance of any nuclear facility. Marcus Thyssen Softwareentwicklung and its suppliers specifically disclaim any express or implied warranty of fitness for such purposes or any other purposes.

5. LIMITATION OF LIABILITY. IN NO EVENT WILL Marcus Thyssen Softwareentwicklung BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS,

BUSINESS INTERRUPTION, LOSS OF BUSINESS DATA, OR ANY OTHER INDIRECT, SPECIAL,
INCIDENTAL OR CONSEQUENTIAL DAMAGES) ARISING OUT OF THE USE OR INABILITY TO USE THE
SOFTWARE, EVEN WHERE Marcus Thyssen Softwareentwicklung HAS BEEN ADVISED OF THE POSSIBILITY OF SU
DAMAGES. IN NO EVENT SHALL THE LIABILITY OF Marcus Thyssen Softwareentwicklung OR DISTRIBUTORS, FO
CLAIM ARISING FROM THE USE OF THE SOFTWARE OR THIS AGREEMENT EXCEED THE AMOUNT OF
FEES RECEIVED BY Marcus Thyssen Softwareentwicklung, IF ANY, IN RESPECT OF YOUR USE OF THE SOFTWARE