ERGOSIGN

WPF CUSTOM CONTROL DEVELOPMENT UNCHAINED

André Lanninger
UI Developer

lanninger@ergosign.de

Ergosign GmbH

André Lanninger
UIDeveloper

Datagraph 11

ChangeStyle

Load

62

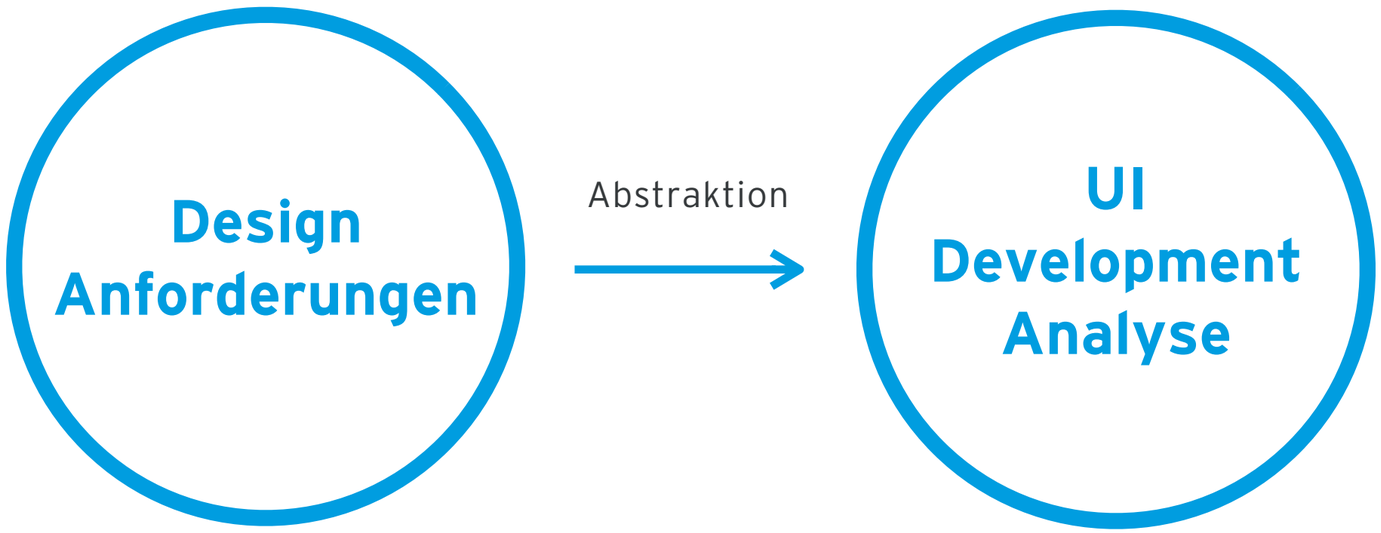
- Bubble 18
  0, 20
- Bubble 19
  10, 15
- Bubble 20
  20, **62**
- Bubble 21
  30, 10
- Bubble 22
  40, 27
- Bubble 23
  50, 77
- Bubble 24
  60, 73
- Bubble 25
  70, 90
- Bubble 26
  80, 33

Auf der Suche nach Custom Controls ;)

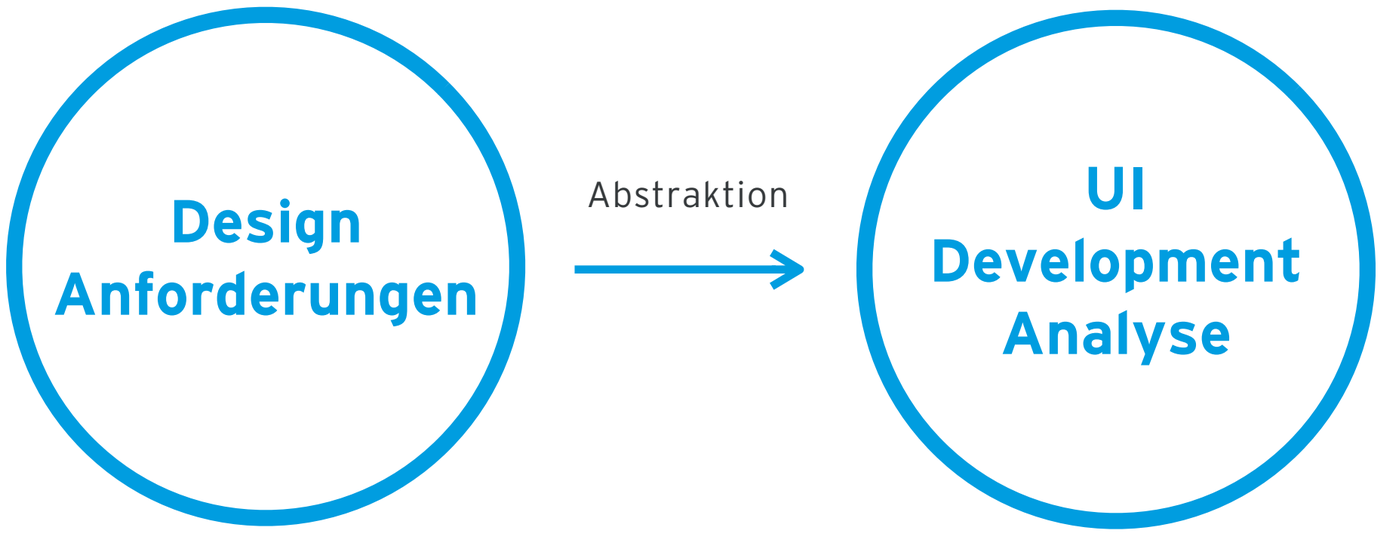
Abstraktes Vorgehen

Abstraktes Vorgehen

Abstraktes Vorgehen

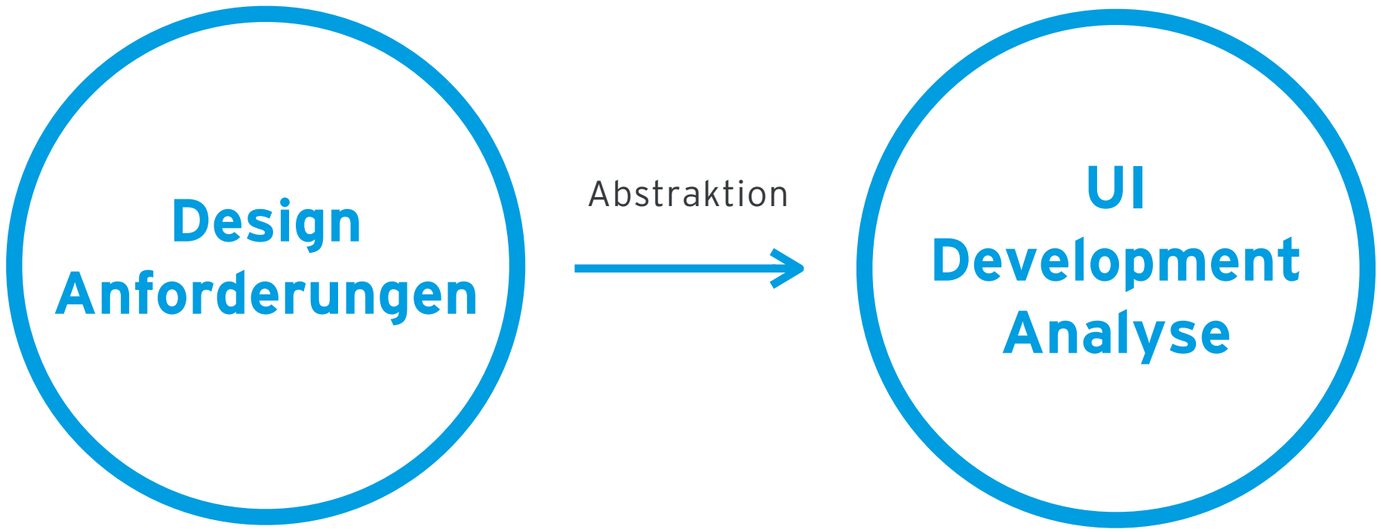


Abstraktes Vorgehen



Identifikation

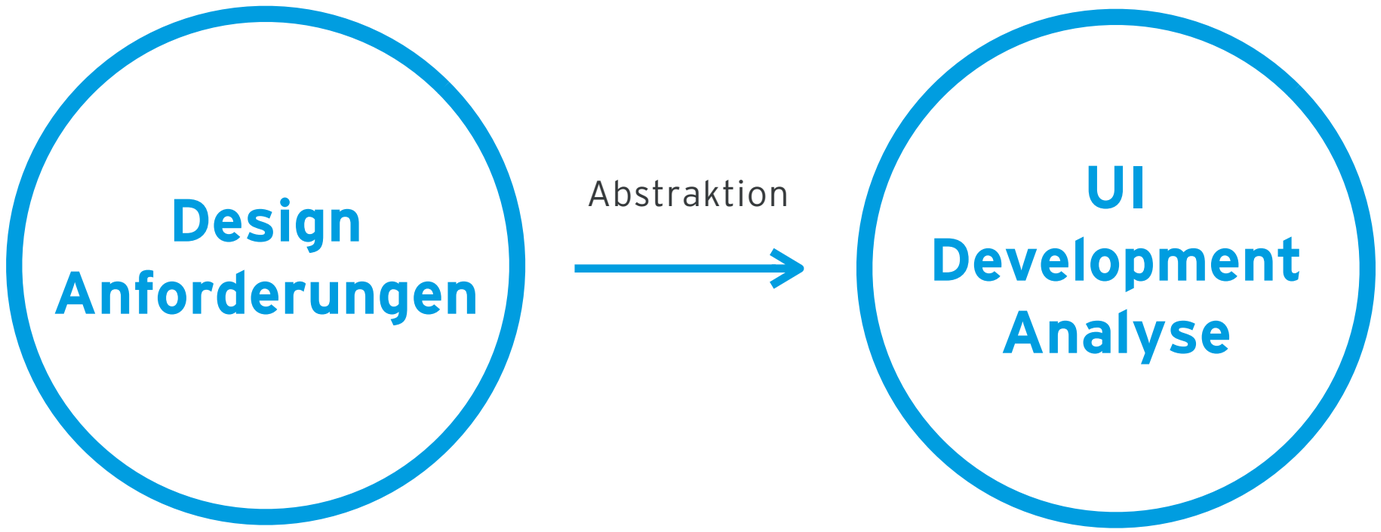
Abstraktes Vorgehen



Identifikation

- Standard UI Elemente

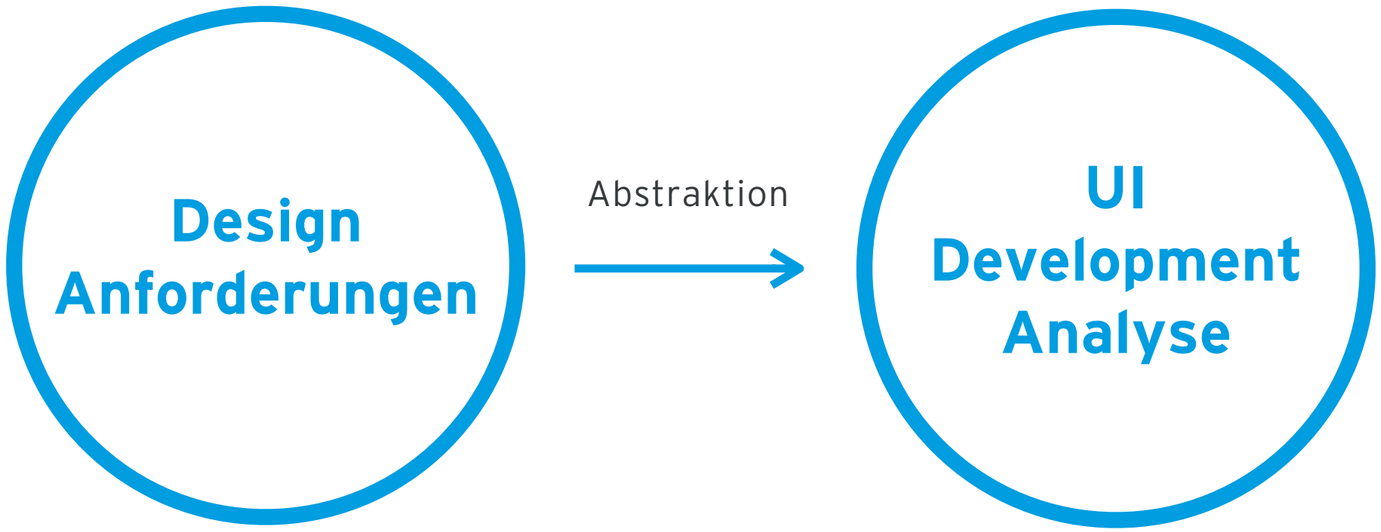
Abstraktes Vorgehen



Identifikation

- Standard UI Elemente
- Custom Controls

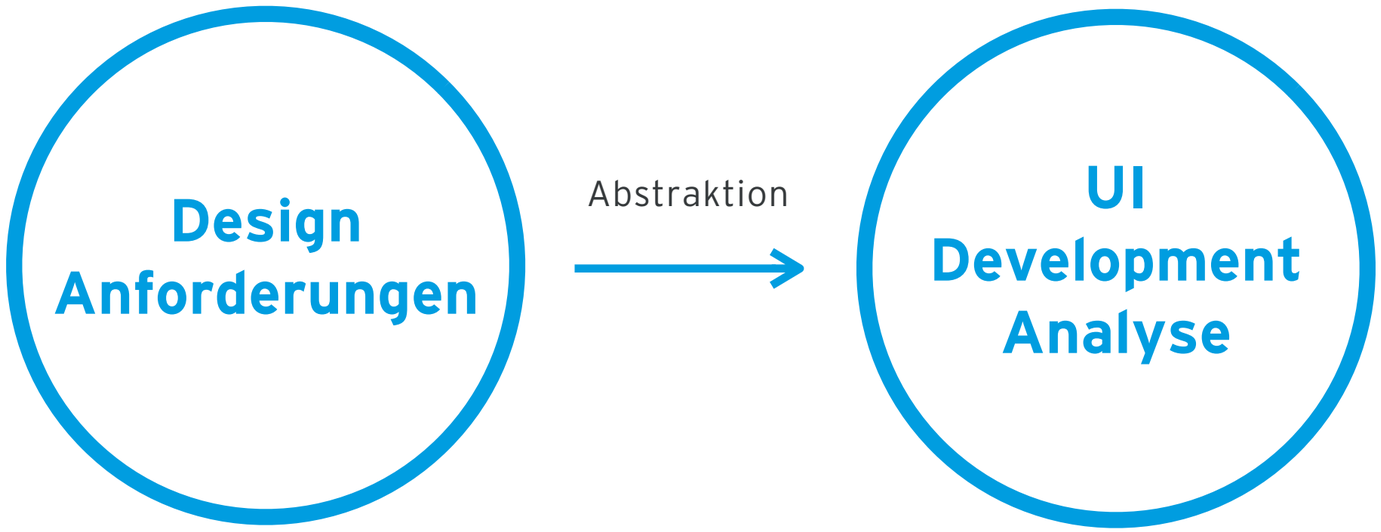
Abstraktes Vorgehen



Identifikation

- Standard UI Elemente
- Custom Controls
  - Anforderungsanalyse pro Custom Control

Abstraktes Vorgehen



Identifikation

- Standard UI Elemente
- Custom Controls
  - Anforderungsanalyse pro Custom Control
  - Funktionale Anforderungen, Properties, Events, ...

Abstraktes Vorgehen

Identifikation

- Standard UI Elemente
- Custom Controls
  - Anforderungsanalyse pro Custom Control
  - Funktionale Anforderungen, Properties, Events, ...

CUSTOM CONTROLS?

Eigentlich “nur”

Eigentlich “nur”

- Ableitung von einer konkreten **Klasse != UserControl**

Eigentlich “nur”

- Ableitung von einer konkreten **Klasse != UserControl**
- Styling- und Template- Möglichkeiten

Eigentlich “nur”

- Ableitung von einer konkreten **Klasse != UserControl**
- Styling- und Template- Möglichkeiten
- Visueller Aufbau im Control Template

Eigentlich “nur”

- Ableitung von einer konkreten **Klasse != UserControl**
- Styling- und Template- Möglichkeiten
- Visueller Aufbau im Control Template
- Default Style möglich Generic.xaml

Eigentlich “nur”

- Ableitung von einer konkreten **Klasse != UserControl**
- Styling- und Template- Möglichkeiten
- Visueller Aufbau im Control Template
- Default Style möglich Generic.xaml
- Zusammenfassung in einer Control Library möglich

Custom Control Library

Custom Control Library

- Besteht i.d.R aus Klasse, Style, Template

Custom Control Library

- Besteht i.d.R aus Klasse, Style, Template
- Verwaltung in Custom Control Library (VS Projektvorlage)

Custom Control Library

- Besteht i.d.R aus Klasse, Style, Template
- Verwaltung in Custom Control Library (VS Projektvorlage)
  - Kann beliebe Custom Controls & Ressourcen enthalten

Custom Control Library

- Besteht i.d.R aus Klasse, Style, Template
- Verwaltung in Custom Control Library (VS Projektvorlage)
  - Kann beliebe Custom Controls & Ressourcen enthalten
  - DefaultStyle wird nach Konvention in “Themes/Generic.xaml” deklariert

Custom Control Library

- Besteht i.d.R aus Klasse, Style, Template
- Verwaltung in Custom Control Library (VS Projektvorlage)
  - Kann beliebe Custom Controls & Ressourcen enthalten
  - DefaultStyle wird nach Konvention in “Themes/Generic.xaml” deklariert
    - Mapping im statischen Konstruktor des Controls (*DefaultStyleKeyProperty.OverrideMetadata*)

SQL TOOLS TEST INFRAGISTICS ANALYZE DEVEXPRESS WINDOW HELP

Quick Launch (Ctrl+Q)

Solution Explorer

Search Solution Explorer (Ctrl+ü)

Solution 'WPFApplication' (1 project)

WPFApplication

Properties

References

App.xaml

MainWindow.xaml

- Build Solution F6
- Rebuild Solution
- Clean Solution
- Run Code Analysis on Solution Alt+F11
- Batch Build...
- Configuration Manager...
- Manage NuGet Packages for Solution...
- Enable NuGet Package Restore
- New Solution Explorer View
- Calculate Code Metrics
- Add DevExpress Item

- New Project...
- Existing Project...
- New Web Site...
- Existing Web Site...
- New Item... Ctrl+Shift+A
- Existing Item... Shift+Alt+A
- New Solution Folder

- Add
- Set StartUp Projects...
- Add Solution to Subversion...
- Add Selected Projects to Subversion...
- Subversion
- Paste Ctrl+V
- Rename
- Open Folder in File Explorer
- Properties Alt+Enter

n Explorer | Team Explorer | Class View

ties

pplication Solution Properties

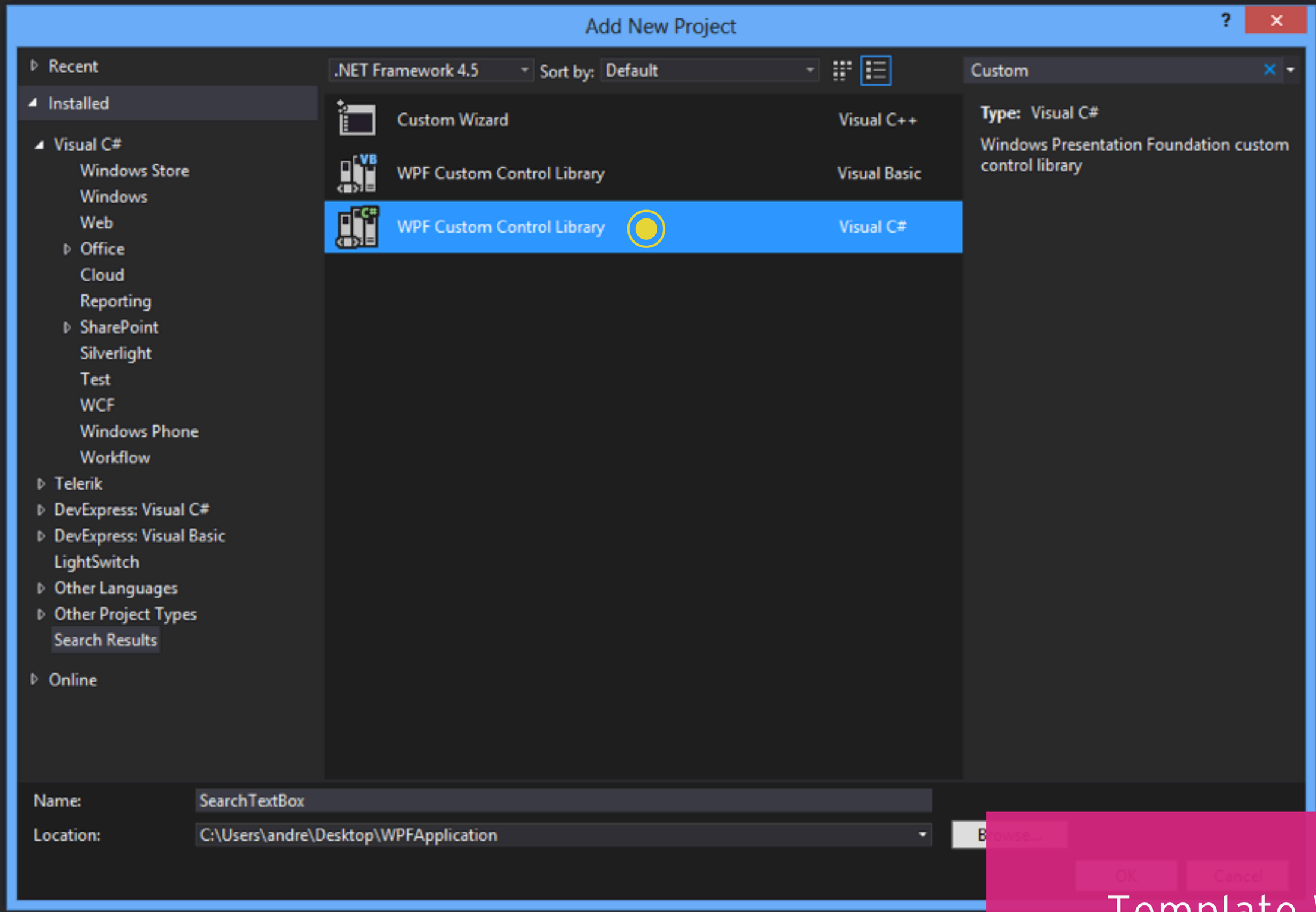
(Name) WPFApplication

Active config Debug|x86

Description

Path C:\Users\Sandra\Desktop\WPF

Neues Projekt erstellen



Template WPF Custom Control Library wählen

CustomControl1()

```
ML file.

e current project.
ile where it is


different project.
ile where it is


extBox"

e the XAML file lives
```

```
new FrameworkPropertyMetadata(typeof(CustomControl1)));
```

Solution Explorer

Search Solution Explorer (Ctrl+ü)

- Solution 'WPFApplication' (2 projects)
  - SearchTextBox
    - Properties
    - References
    - Themes
      - Generic.xaml
    - CustomControl1.cs
  - **WPFApplication**
    - Properties
    - References
    - App.xaml
    - MainWindow.xaml

Solution Explorer | Team Explorer | Class View

Properties

SearchTextBox Project Properties

Project File SearchTextBox.csproj

Project Folder C:\Users\Andrea\Desktop\WPFAp

Das wars schon ;)

CustomControl1.cs

```
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace SearchTextBox
{
    /// <summary>
    /// Follow steps 1a or 1b and then 2 to use this custom control in a XAML file.
    ///
    /// Step 1a) Using this custom control in a XAML file that exists in the current projec
    /// Add this XmlNamespace attribute to the root element of the markup file where it is
    /// to be used:
    ///
    ///     xmlns:MyNamespace="clr-namespace:SearchTextBox"
    ///
    ///
    /// Step 1b) Using this custom control in a XAML file that exists in a different project.
    /// Add this XmlNamespace attribute to the root element of the markup file where it is
    /// to be used:
    ///
    ///     xmlns:MyNamespace="clr-namespace:SearchTextBox;assembly=SearchTextBox"
    ///
    /// You will also need to add a project reference from the project where the XAML file lives
    /// to this project and Rebuild to avoid compilation errors:
    ///
    ///     Right click on the target project in the Solution Explorer and
    ///     "Add Reference"->"Projects"->[Select this project]
    ///
    ///
    /// Step 2)
    /// Go ahead and use your control in the XAML file.
    ///
    ///     <MyNamespace:CustomControl1/>
    ///
    /// </summary>
    public class CustomControl1 : Control
    {
        static CustomControl1()
        {
            DefaultStyleKeyProperty.OverrideMetadata(typeof(CustomControl1), new FrameworkPropertyMetadata(typeof(CustomControl1)));
        }
    }
}
```

Generic.xaml

```
<ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:local="clr-namespace:SearchTextBox">
    <Style TargetType="{x:Type local:CustomControl1}">
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="{x:Type local:CustomControl1}">
                    <Border Background="{TemplateBinding Background}"
                            BorderBrush="{TemplateBinding BorderBrush}"
                            BorderThickness="{TemplateBinding BorderThickness}">

                    </Border>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>
</ResourceDictionary>
```

```
public class SearchTextBox : TextBox
{
    Members

    DependencyProperties

    PropertyChangedCallbacks

    Events

    Commands

    Handlers

    #region Ctor

    /// <summary> ...
    static SearchTextBox()
    {
        DefaultStyleKeyProperty.OverrideMetadata(
            typeof(SearchTextBox),
            new FrameworkPropertyMetadata(typeof(SearchTextBox)));

        CommandManager.RegisterClassCommandBinding(
            typeof(SearchTextBox),
            new CommandBinding(ClearTextCommand, ExecuteClearTextCommand, CanExecuteClearCommand));
    }

    #endregion

    Init/Cleanup

    #region OverrideMethods

    /// <summary> ...
    protected override void OnTextChanged(TextChangedEventArgs e)...

    /// <summary> ...
    protected override void OnKeyDown(KeyEventArgs e)...
```

SearchTextBox.cs

```
<Style x:Key="{ComponentResourceKey TypeInTargetAssembly={x:Type local:SearchTextBox}, ResourceId=SearchTextBox
```

```
<Style TargetType="{x:Type local:SearchTextBox}">
    <Setter Property="Foreground" Value="Black" />
    <Setter Property="Background" Value="White" />
    <Setter Property="BorderBrush" Value="Black" />
    <Setter Property="BorderThickness" Value="1" />
    <Setter Property="Padding" Value="1" />
    <Setter Property="AllowDrop" Value="true" />
    <Setter Property="ScrollViewer.PanningMode" Value="VerticalFirst" />
    <Setter Property="Stylus.IsFlicksEnabled" Value="False" />
    <Setter Property="WatermarkFontStyle" Value="Italic" />
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type local:SearchTextBox}">
                <Border x:Name="Bd"
                        Background="{TemplateBinding Background}"
                        BorderBrush="{TemplateBinding BorderBrush}"
                        BorderThickness="{TemplateBinding BorderThickness}"
                        SnapsToDevicePixels="true">
                    <Grid>
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition />
                            <ColumnDefinition Width="auto" />
                        </Grid.ColumnDefinitions>
                        <Grid x:Name="TextContainer">
                            <TextBlock x:Name="Watermark"
                                       Margin="{TemplateBinding Padding}"
                                       FontStyle="{TemplateBinding WatermarkFontStyle}"
                                       Opacity=".5"
                                       Text="{TemplateBinding Watermark}"
                                       Visibility="Hidden" />
                            <ScrollViewer x:Name="PART_ContentHost" SnapsToDevicePixels="{TemplateBinding Snaps
                        </Grid>
                        <Button Grid.Column="1"
```

Generic.xaml

WANN IST EIN CUSTOM CONTROL SINNVOLL?

WPF Standard Control?

WPF Standard Control?

WPF Standard Controls gruppieren?

WPF Standard Control?

WPF Standard Controls gruppieren?

Styling & Templating?

WPF Standard Control?

WPF Standard Controls gruppieren?

Styling & Templating?

ValueConverter/Markup Extensions?

WPF Standard Control?

WPF Standard Controls gruppieren?

Styling & Templating?

ValueConverter/Markup Extensions?

Attached Properties/Behaviors?

WPF Standard Control?

WPF Standard Controls gruppieren?

Styling & Templating?

ValueConverter/Markup Extensions?

Attached Properties/Behaviors?

Drittanbieter?

WPF Standard Control?

WPF Standard Controls gruppieren?

Styling & Templating?

ValueConverter/Markup Extensions?

Attached Properties/Behaviors?

Drittanbieter?

vs.

My Custom Control

Allgemeine Überlegungen

Allgemeine Überlegungen

- Wartbarkeit

Allgemeine Überlegungen

- Wartbarkeit
- Kommunikation

Allgemeine Überlegungen

- Wartbarkeit
- Kommunikation
- Wiederverwendbarkeit

Allgemeine Überlegungen

- Wartbarkeit
- Kommunikation
- Wiederverwendbarkeit
- Produktivität bei großen Projekten

Allgemeine Überlegungen

- Wartbarkeit
- Kommunikation
- Wiederverwendbarkeit
- Produktivität bei großen Projekten
- Konsistenz in User Interface

Allgemeine Überlegungen

- Wartbarkeit
- Kommunikation
- Wiederverwendbarkeit
- Produktivität bei großen Projekten
- Konsistenz in User Interface
- “Optimale” Lösung für das Problem, nicht mehr und nicht weniger ;)

Allgemeine Überlegungen

- Wartbarkeit
- Kommunikation
- Wiederverwendbarkeit
- Produktivität bei großen Projekten
- Konsistenz in User Interface
- “Optimale” Lösung für das Problem, nicht mehr und nicht weniger ;)

- Jedoch ist es **nicht** immer notwendig ein CustomControl zu erstellen (siehe Alternativen & Prototyp vs. Produktiv Code)

André Lanninger
UIDeveloper

Datagraph 11

ChangeStyle

Load

62

- Bubble 18
  0, 20
- Bubble 19
  10, 15
- Bubble 20
  20, **62**
- Bubble 21
  30, 10
- Bubble 22
  40, 27
- Bubble 23
  50, 77
- Bubble 24
  60, 73
- Bubble 25
  70, 90
- Bubble 26
  80, 33

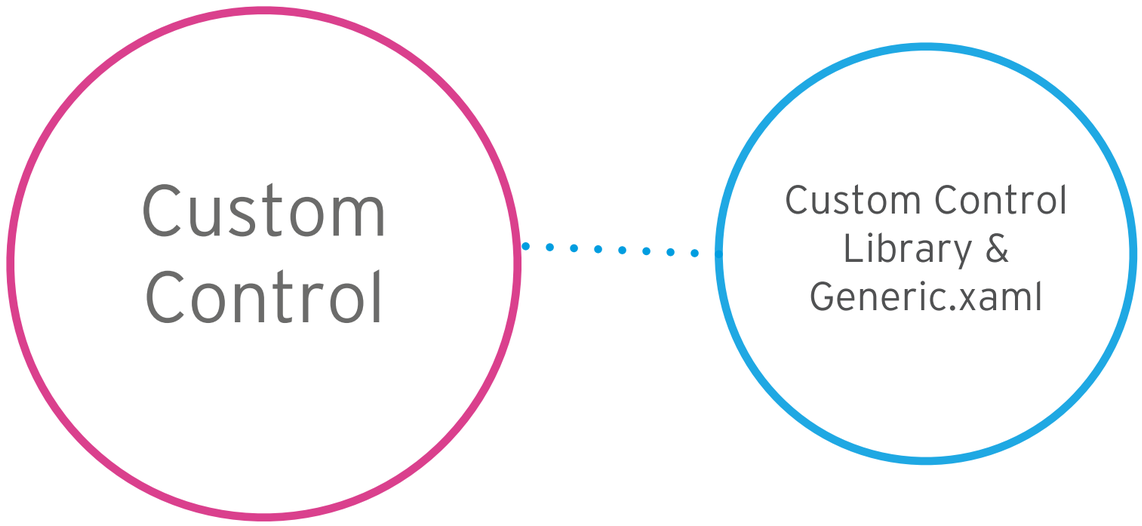
PointChart

SearchTextBox

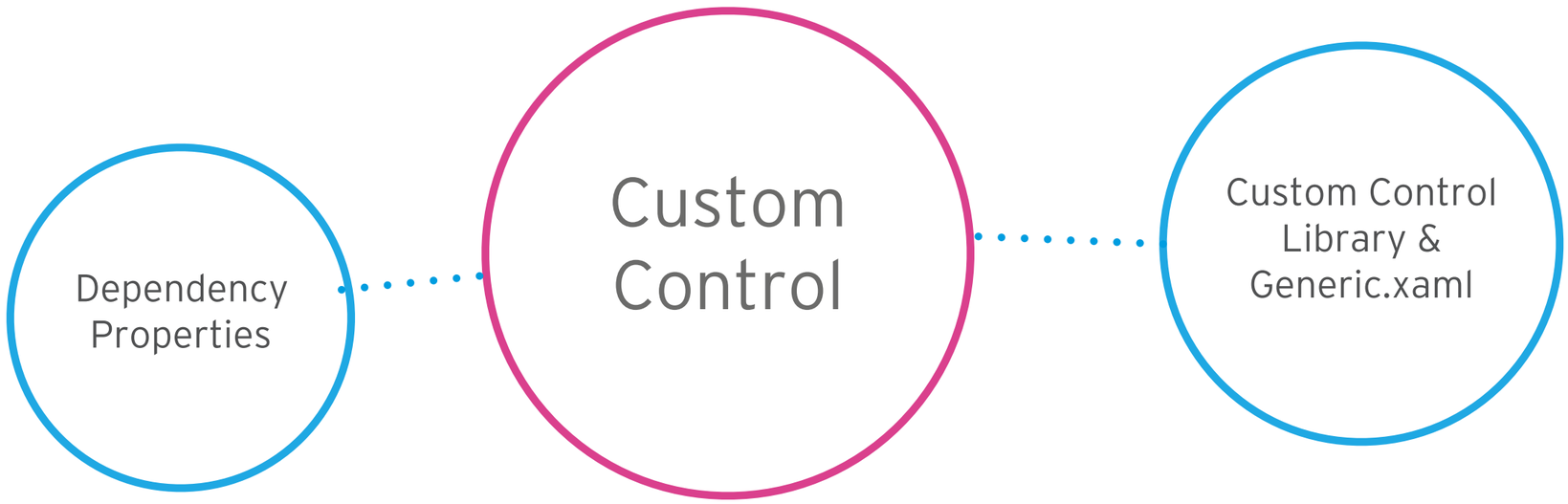
Die Ausrüstung!

Die Ausrüstung!

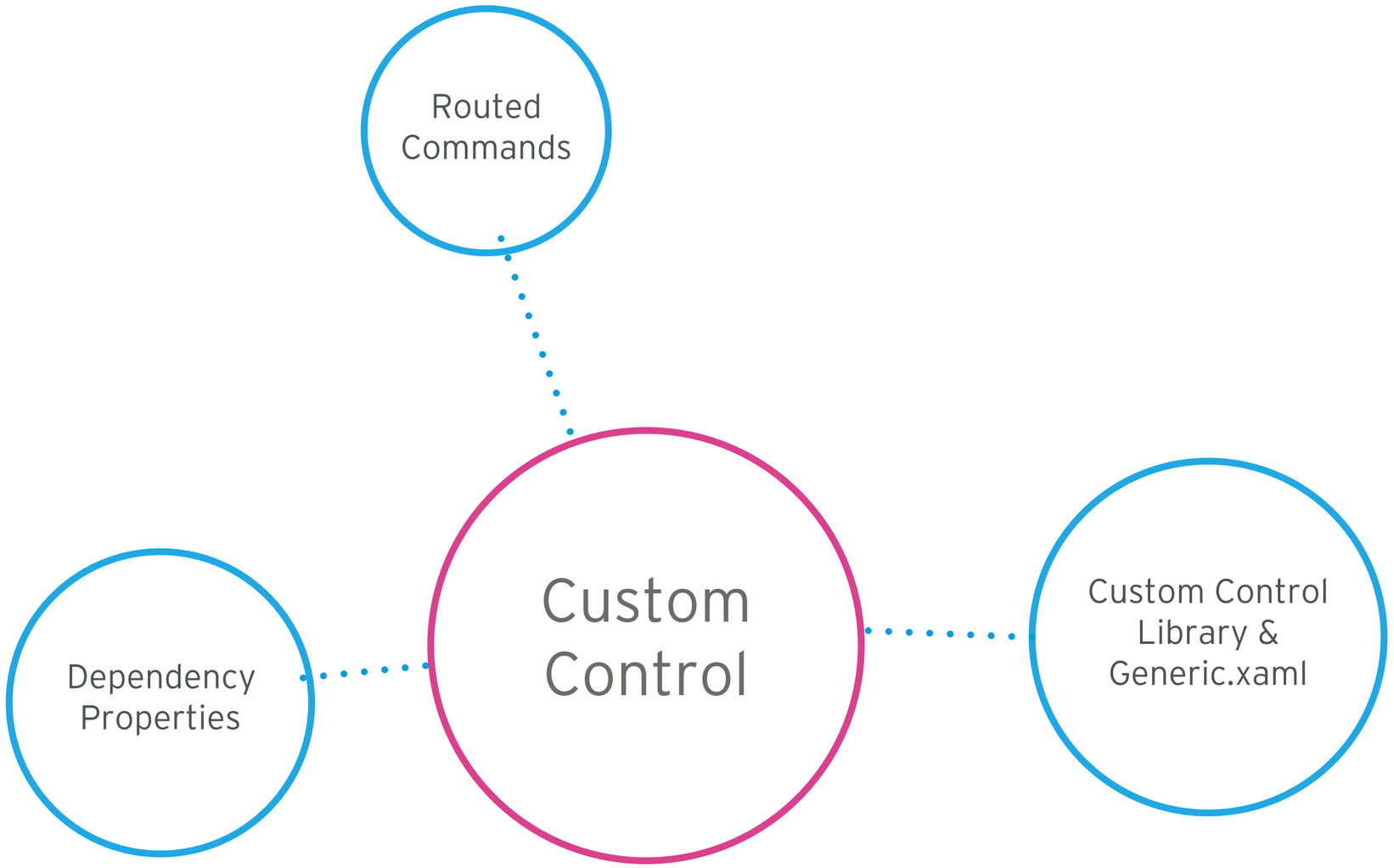
Die Ausrüstung!



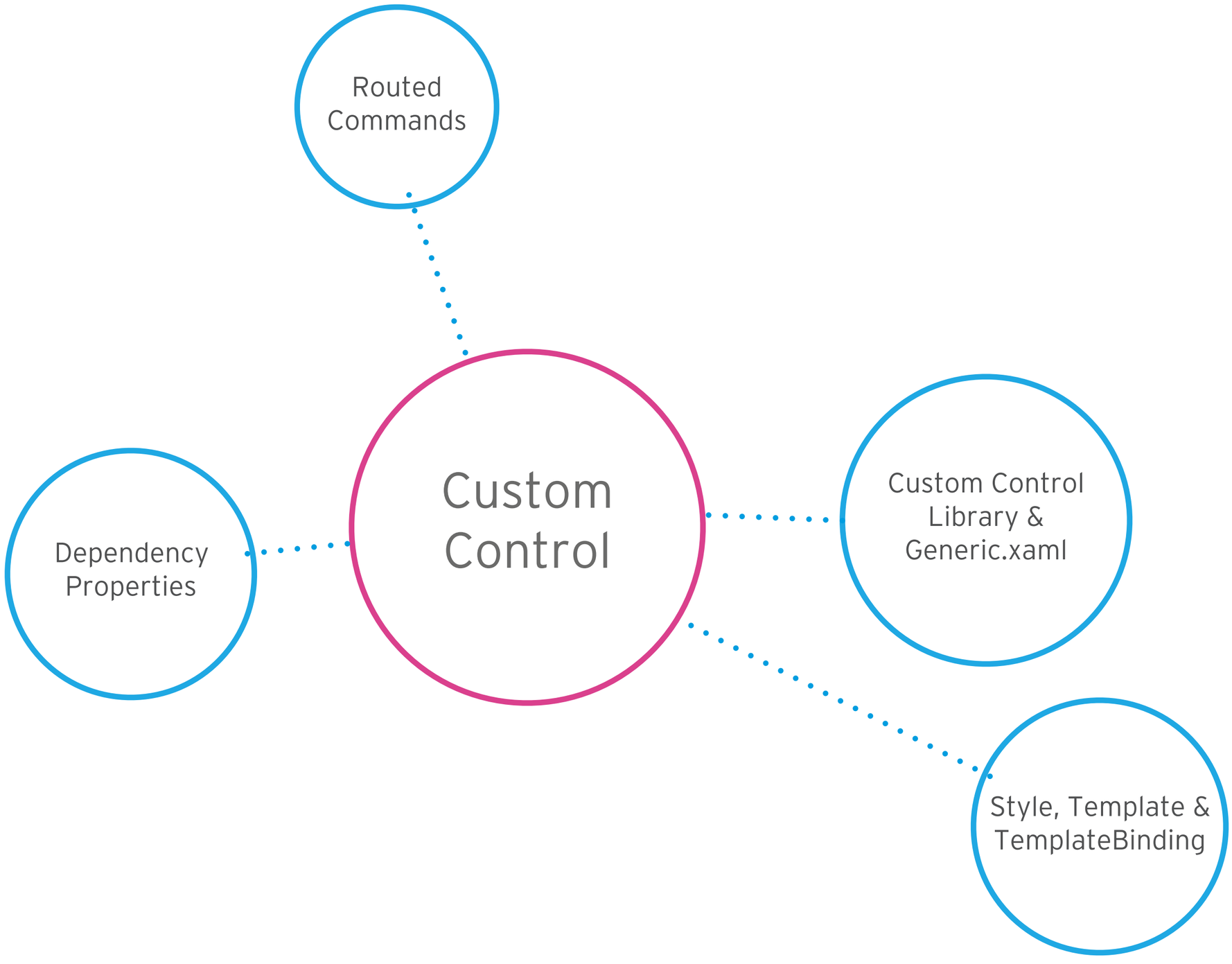
Die Ausrüstung!



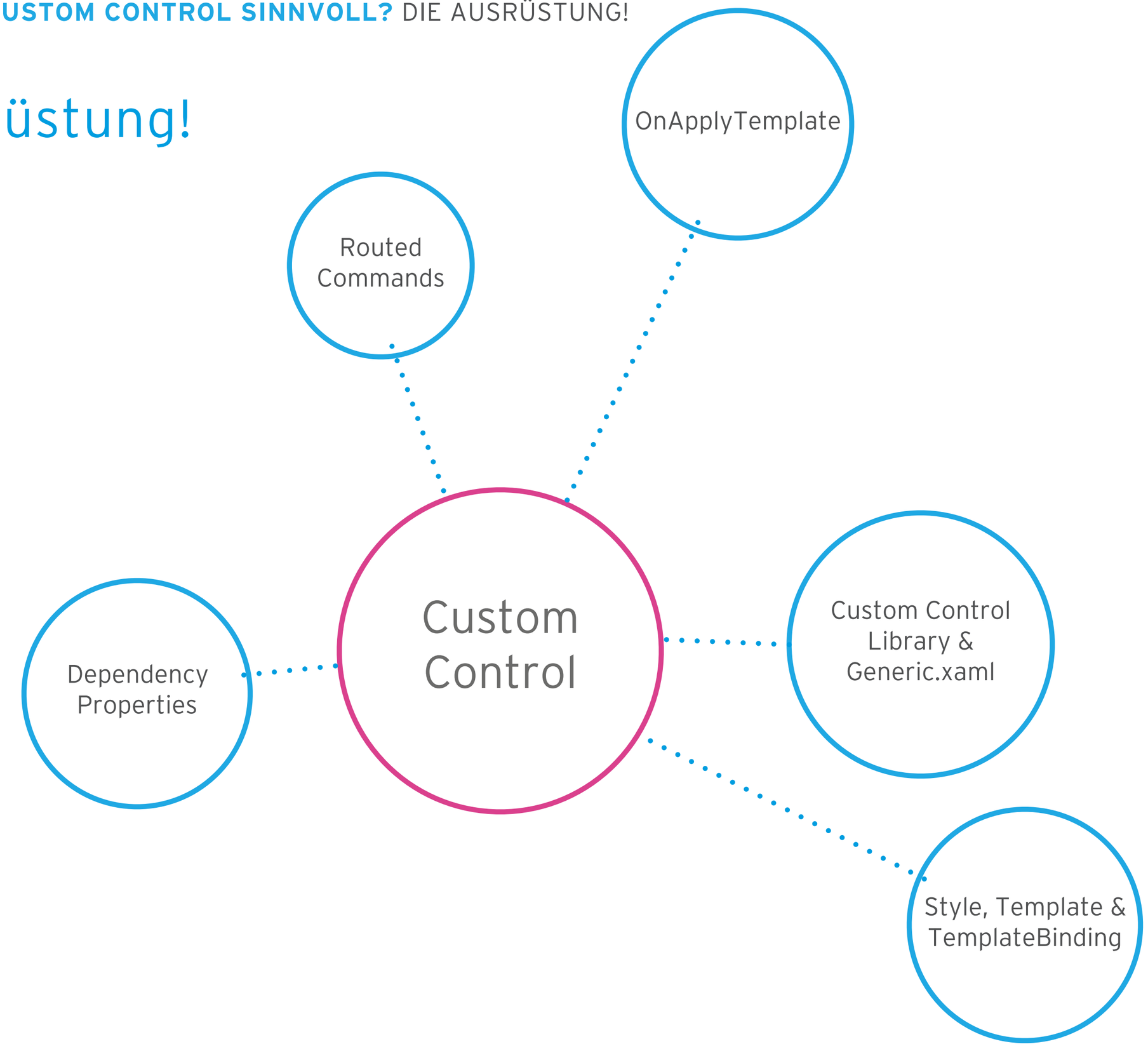
Die Ausrüstung!



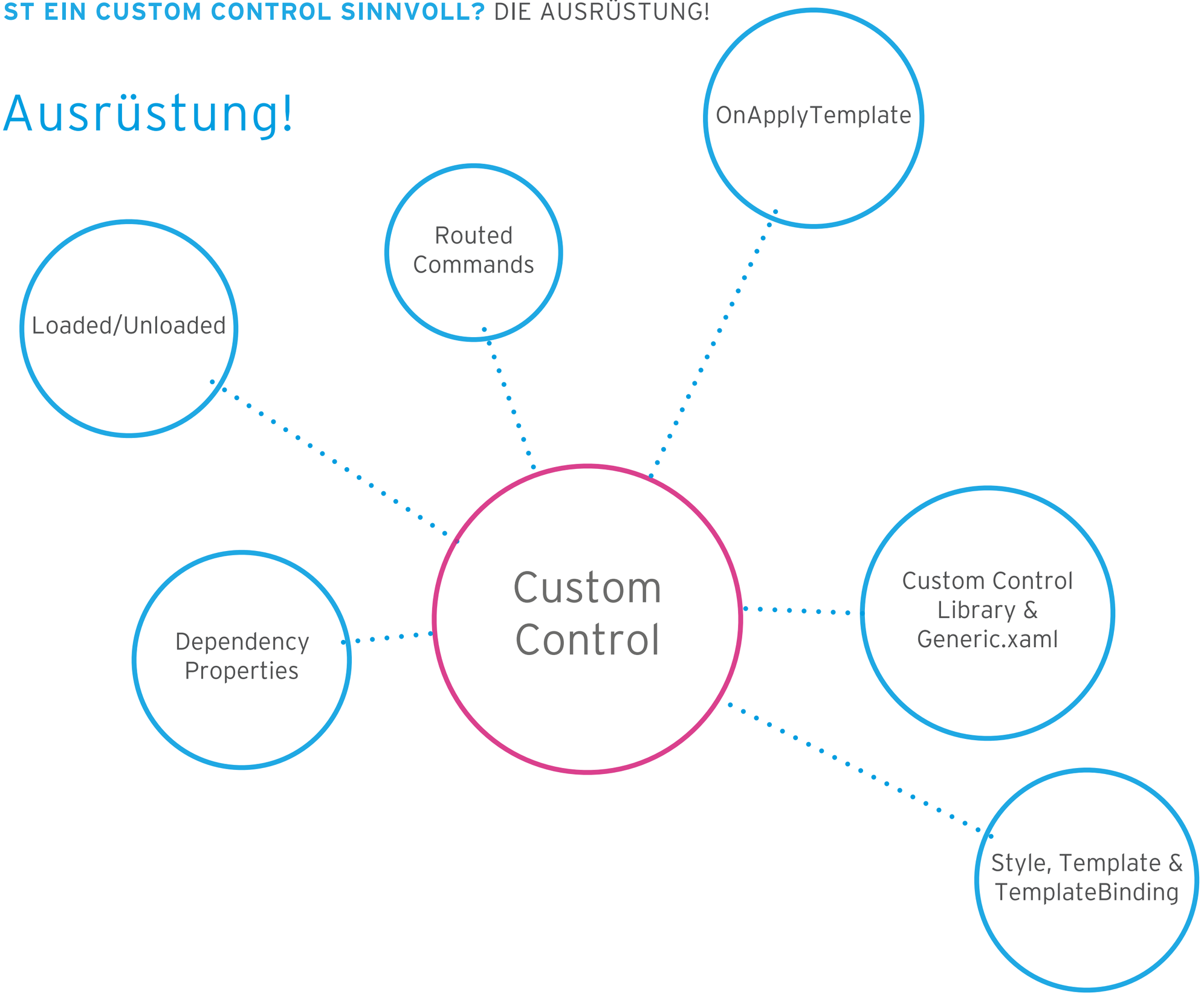
Die Ausrüstung!



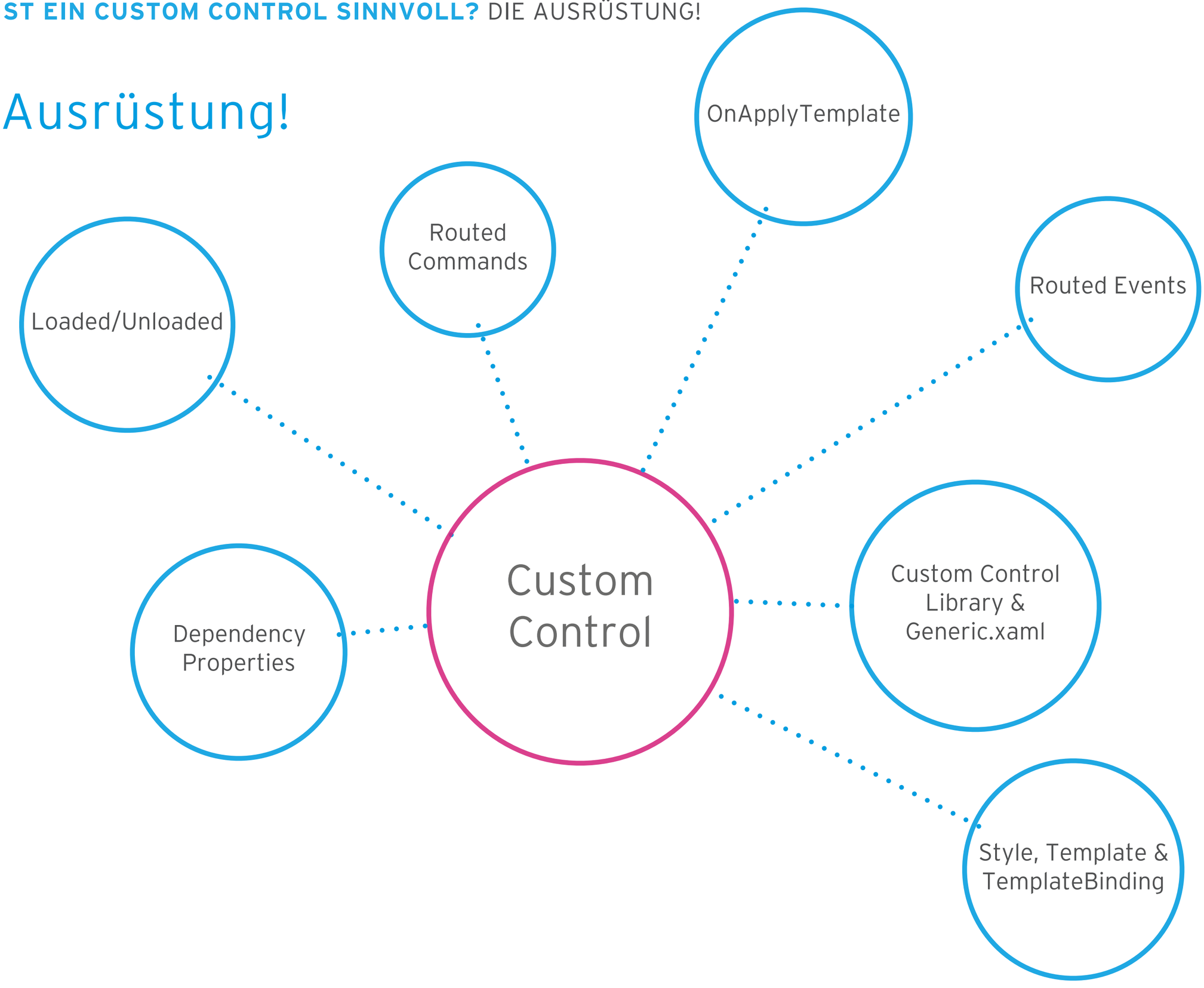
Die Ausrüstung!



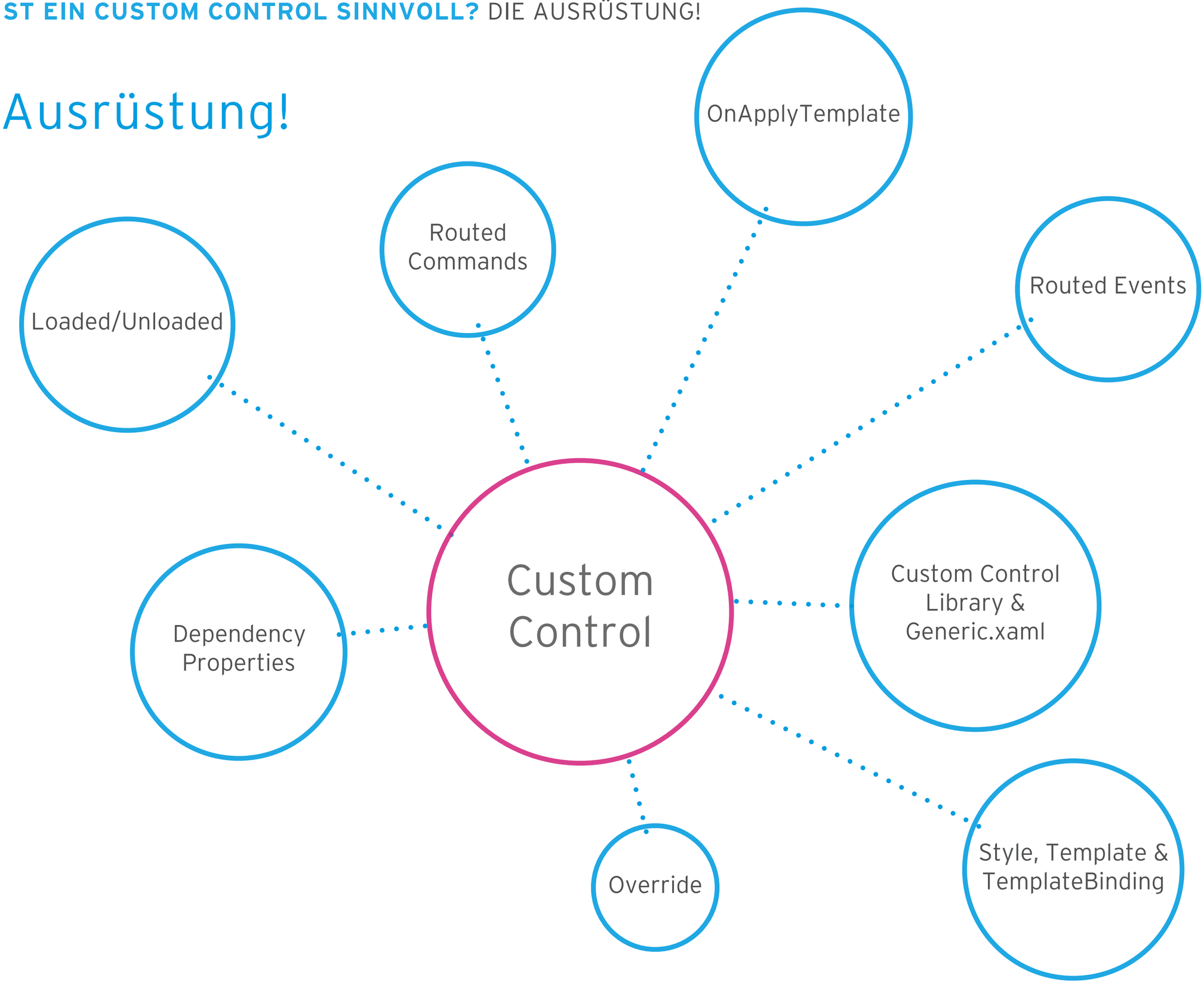
Die Ausrüstung!



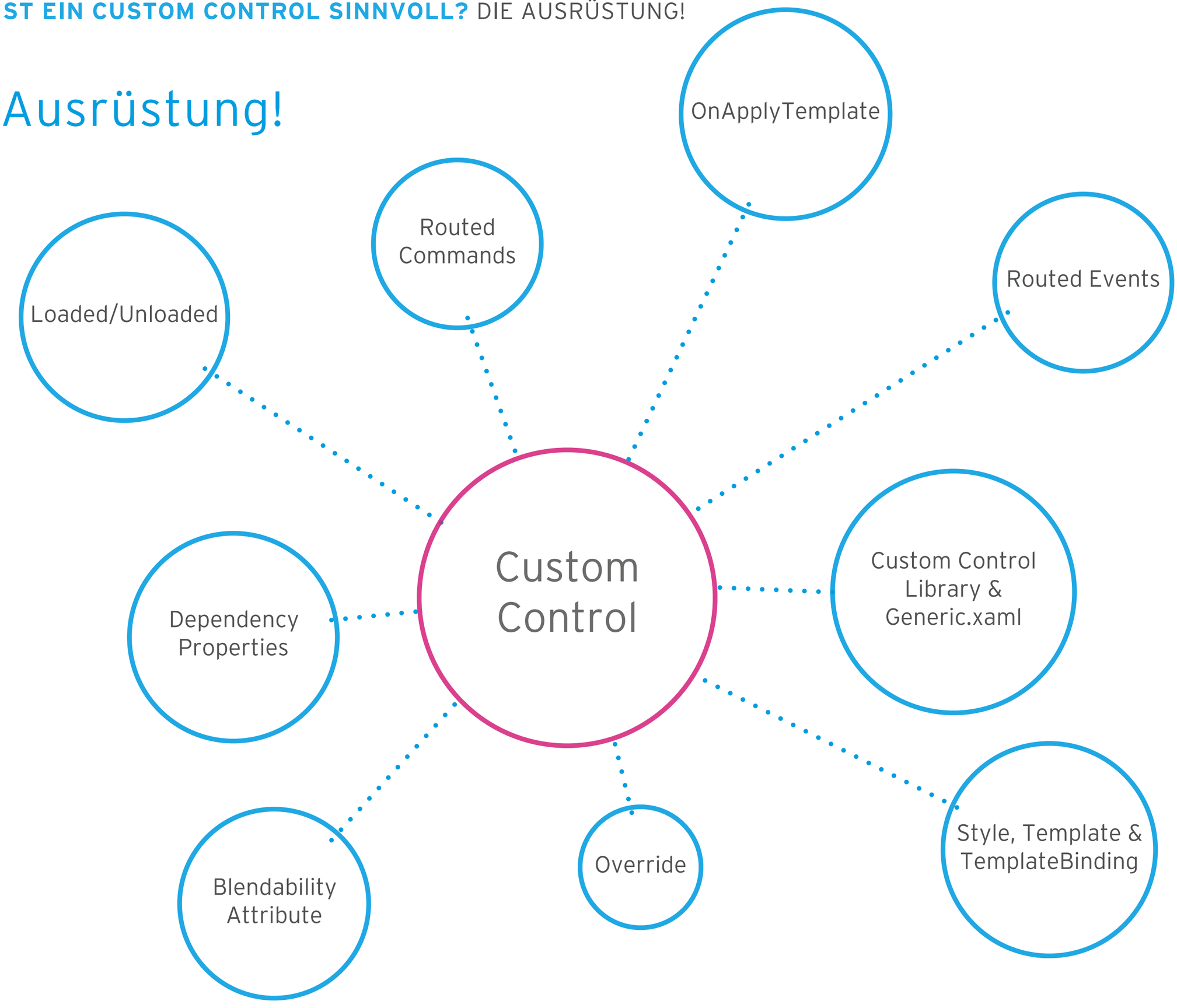
Die Ausrüstung!



Die Ausrüstung!



Die Ausrüstung!



SEARCHTEXTBOX

André Lanninger
UIDeveloper

Datagraph 11

ChangeStyle

Load

62

- Bubble 18
  0, 20
- Bubble 19
  10, 15
- Bubble 20
  20, **62**
- Bubble 21
  30, 10
- Bubble 22
  40, 27
- Bubble 23
  50, 77
- Bubble 24
  60, 73
- Bubble 25
  70, 90
- Bubble 26
  80, 33

SearchTextBox

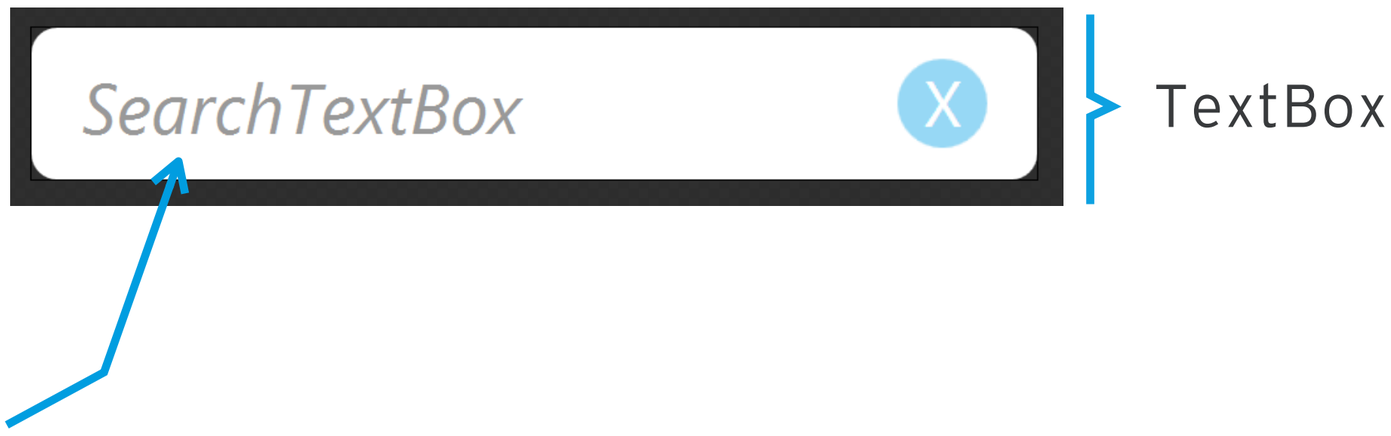
ANALYSE

View Model

View Model

View Model

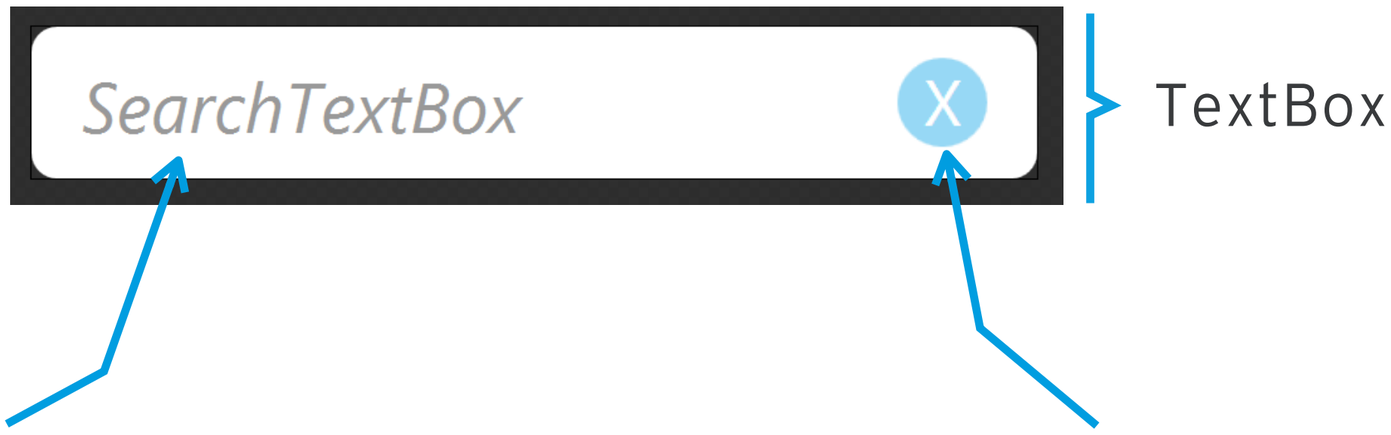
View Model



View Model

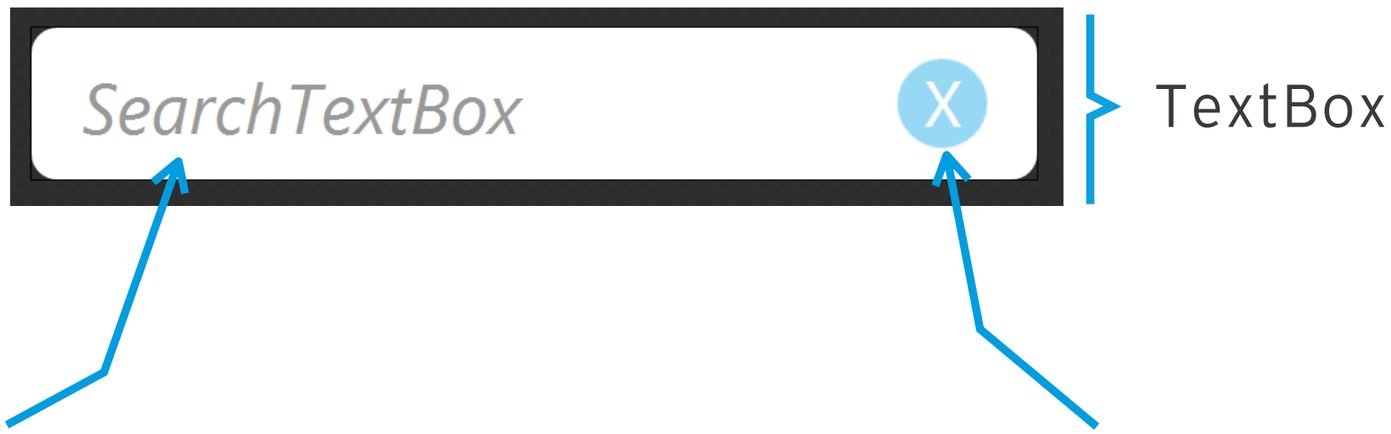
- Watermark
- WatermarkFontStyle
- WatermarkFontWeight
- WatermarkForeground

View Model



- Watermark
- WatermarkFontStyle
- WatermarkFontWeight
- WatermarkForeground

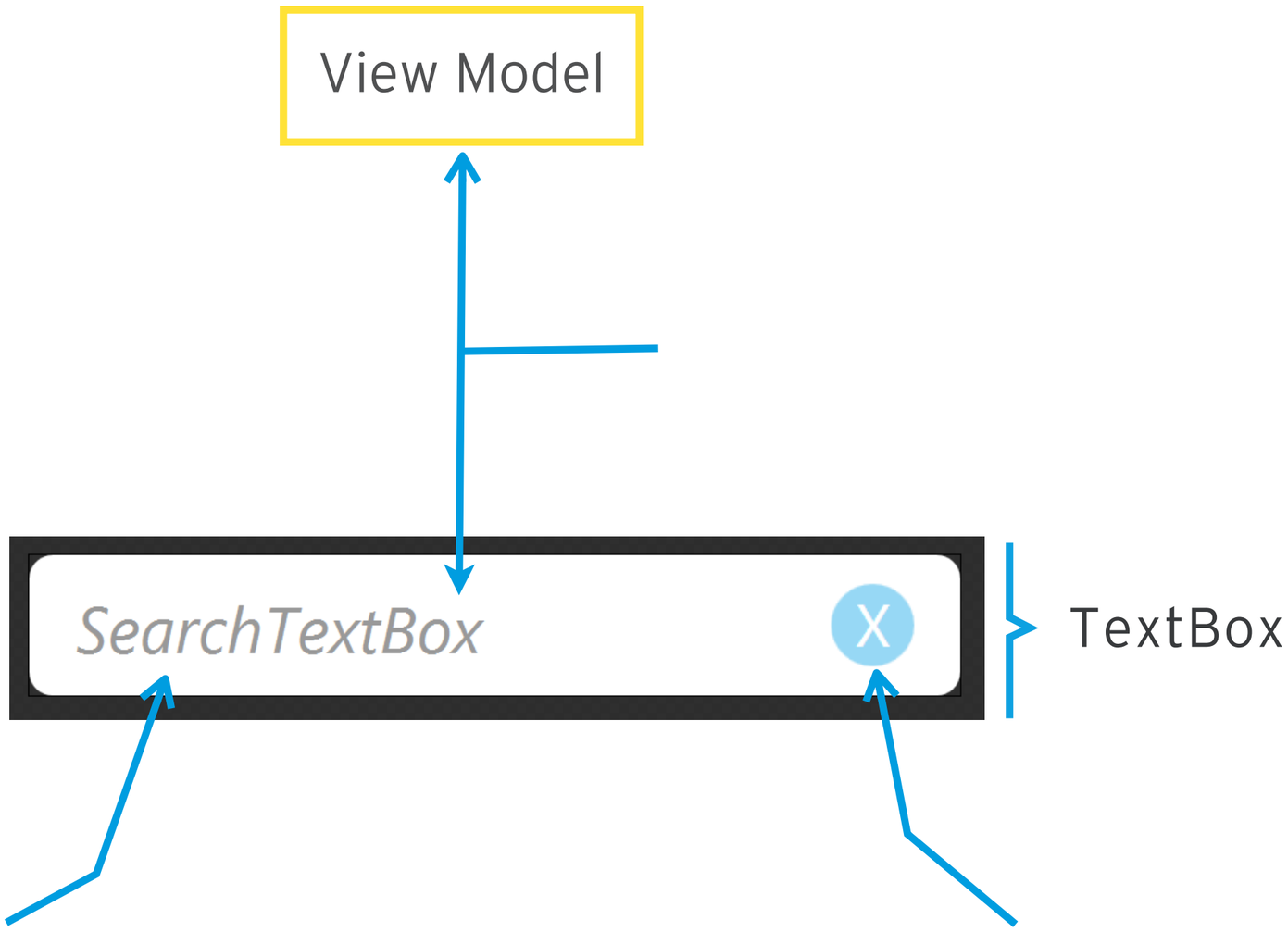
View Model



- Watermark
- WatermarkFontStyle
- WatermarkFontWeight
- WatermarkForeground

ClearCommand

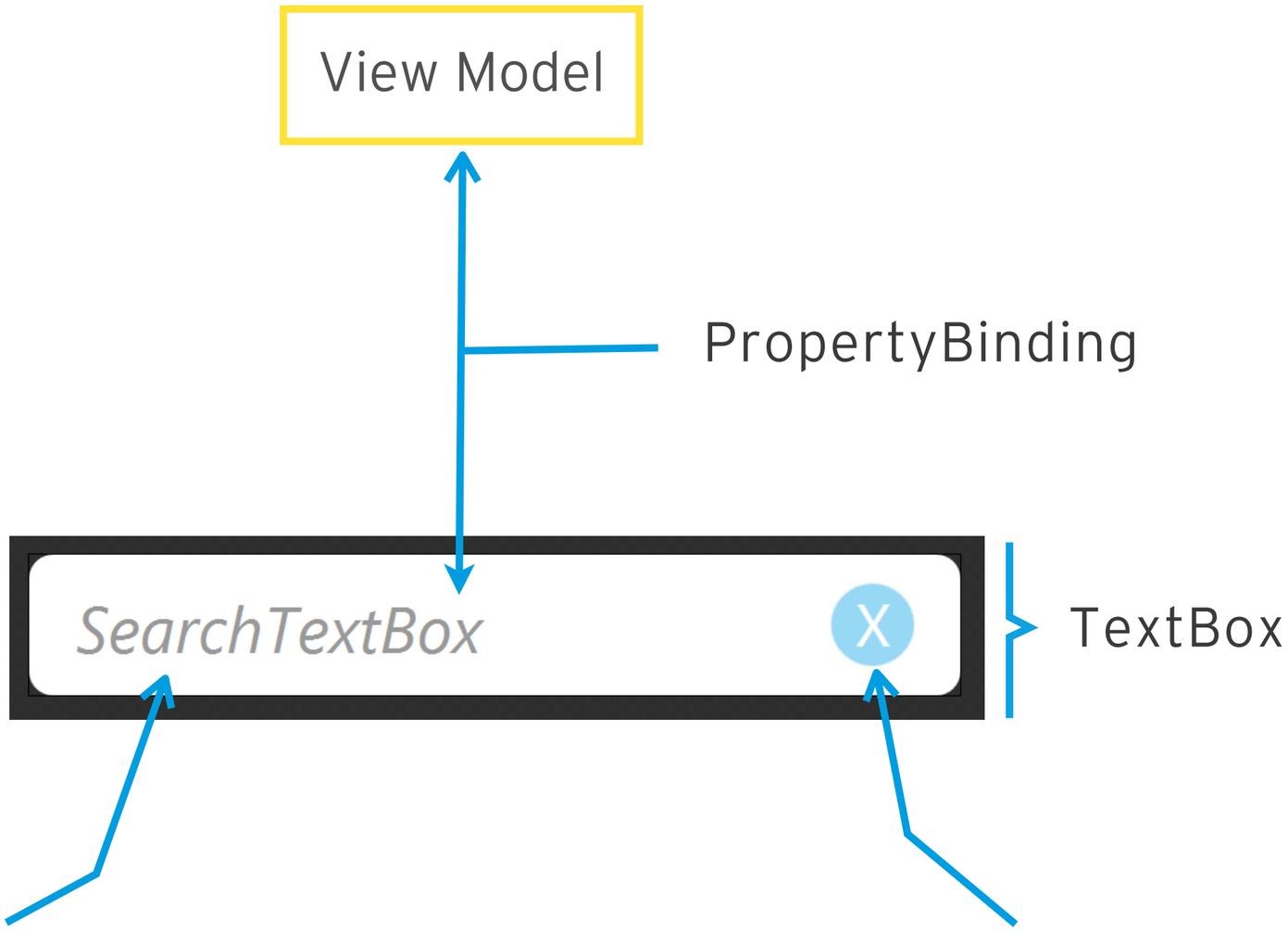
(Rucksetzen des Texts)



- Watermark
- WatermarkFontStyle
- WatermarkFontWeight
- WatermarkForeground

ClearCommand

(Rucksetzen des Texts)

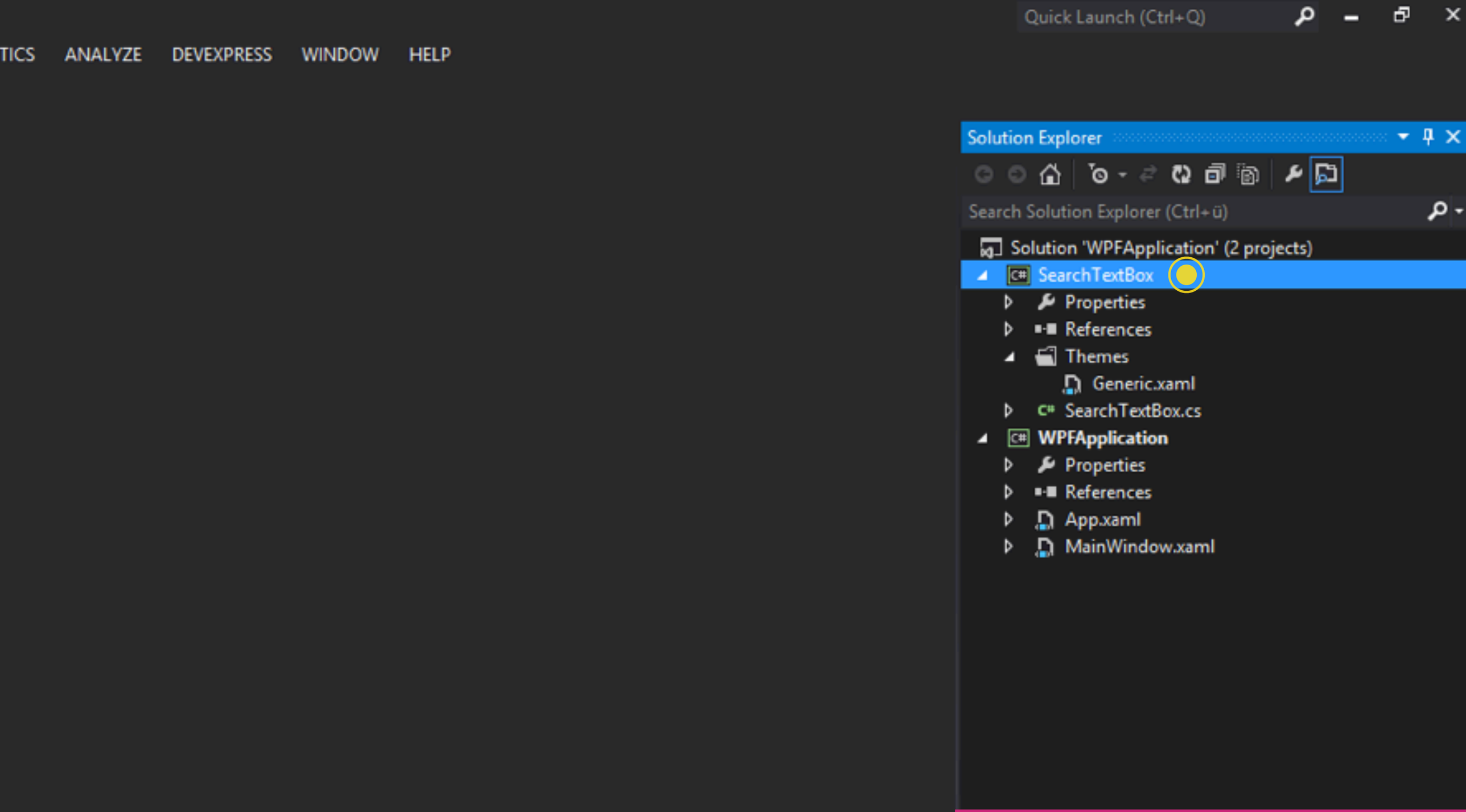


- Watermark
- WatermarkFontStyle
- WatermarkFontWeight
- WatermarkForeground

ClearCommand

(Rucksetzen des Texts)

CUSTOM CONTROL LIBRARY



Es kann losgehen!

```
/// <summary> ...
[ContentPropertyAttribute("MyDefaultContentProperty")]
[Description("The class SearchTextBox represents a textbox with watermark and clear t
public class SearchTextBox : TextBox
{
    Members

    DependencyProperties

    PropertyChangedCallbacks

    Events

    Commands

    Handlers

    Ctor

    Init/Cleanup

    OverrideMethods

    Methods

    Properties
}
```

Basisklasse bestimmen

(SearchTextBox.cs)

STYLE & TEMPLATE

SEARCHTEXTBOX STYLE & TEMPLATE

http://msdn.microsoft.com/en-us/library/aa970773(v=vs.110).aspx

TextBox Styles and Templates

.NET Framework 4.5 | Other Versions | 1 out of 2 rated this helpful

This topic describes the styles and templates for the TextBox control. You can modify the default ControlTemplate to give the control a unique appearance. For more information, see Customizing the Appearance of an Existing Control by Creating a ControlTemplate.

TextBox Parts

The following table lists the named parts for the TextBox control.

Part	Type	Description
PART_ContentHost	FrameworkElement	A visual element that can contain a FrameworkElement. The text of the TextBox is displayed in this element.

```
<Style TargetType="{x:Type TextBox}">
  <Setter Property="SnapsToDevicePixels"
          Value="True" />
  <Setter Property="OverridesDefaultStyle"
          Value="True" />
  <Setter Property="KeyboardNavigation.TabNavigation"
          Value="None" />
  <Setter Property="FocusVisualStyle"
          Value="{x:Null}" />
  <Setter Property="MinWidth"
          Value="120" />
  <Setter Property="MinHeight"
          Value="20" />
  <Setter Property="AllowDrop"
          Value="true" />
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="{x:Type TextBoxBase}">
        <Border Name="Border"
                CornerRadius="2"
                Padding="2"
                BorderThickness="1">
          <ScrollViewer Margin="0"
                        x:Name="PART_ContentHost" />
        </Border>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>
```

ERGOSIGN

```
<Style TargetType="{x:Type local:SearchTextBox}">
    <Setter Property="Foreground" Value="Black" />
    <Setter Property="Background" Value="White" />
    <Setter Property="BorderBrush" Value="Black" />
    <Setter Property="BorderThickness" Value="1" />
    <Setter Property="Padding" Value="1" />
    <Setter Property="AllowDrop" Value="true" />
    <Setter Property="ScrollViewer.PanningMode" Value="VerticalFirst" />
    <Setter Property="Stylus.IsFlicksEnabled" Value="False" />
    <Setter Property="WatermarkFontStyle" Value="Italic" />
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type local:SearchTextBox}">
                <Border x:Name="Bd"
                        Background="{TemplateBinding Background}"
                        BorderBrush="{TemplateBinding BorderBrush}"
                        BorderThickness="{TemplateBinding BorderThickness}"
                        SnapsToDevicePixels="true">
                    <Grid>
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition />
                            <ColumnDefinition Width="auto" />
                        </Grid.ColumnDefinitions>
                        <Grid x:Name="TextContainer">
                            <TextBlock x:Name="Watermark"
                                       Margin="{TemplateBinding Padding}"
                                       FontStyle="{TemplateBinding WatermarkFontStyle}"
                                       Opacity=".5"
                                       Text="{TemplateBinding Watermark}"
                                       Visibility="Hidden" />
                            <ScrollViewer x:Name="PART_ContentHost" SnapsToDevicePixels="{TemplateBinding SnapsToDevicePixels}" />
                        </Grid>
                        <Button Grid.Column="1"
                                Width="15"
                                Height="15"
                                Margin="0 0 5 0"
                                Padding="0 -2 0 0"
                                Command="{x:Static local:SearchTextBox.ClearTextCommand}"
                                Content="X"
                                Focusable="False"
                                Style="{DynamicResource {ComponentResourceKey TypeInTargetAssembly={x:Type local:SearchTextBox},
                                                                              ResourceId=SearchTextBoxButtonStyle}}" />
                    </Grid>
                </Border>
                <ControlTemplate.Triggers>
                    <MultiTrigger>
                        <MultiTrigger.Conditions>
                            <Condition Property="IsKeyboardFocused" Value="false" />
                            <Condition Property="Text" Value="" />
```

Style & Template

(Generic.xaml)

DEPENDENCY PROPERTIES BASICS

Dependency Properties - Basics

Dependency Properties - Basics

- Oberflächlich wie .NET Properties (Get, Set). Unterschied liegt in der Implementierung. Ausschließliche Verwendung bei Controls

Dependency Properties - Basics

- Oberflächlich wie .NET Properties (Get, Set). Unterschied liegt in der Implementierung. Ausschließliche Verwendung bei Controls
- Metadaten und Speicherperformance, Änderungsbenachrichtigungen Grundlage für Trigger, Data Binding, Animationen etc.

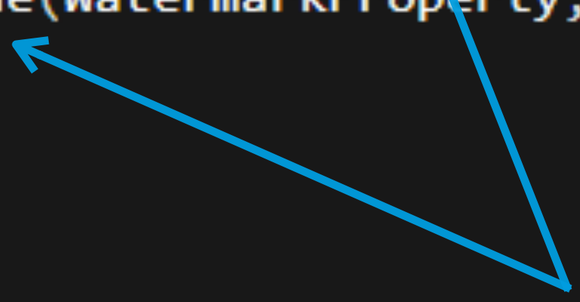
```
/// <summary>
/// WatermarkProperty - Gets or sets the watermark.
/// </summary>
public static readonly DependencyProperty WatermarkProperty = DependencyProperty.Register(
    "Watermark",
    typeof(string),
    typeof(SearchTextBox),
    new FrameworkPropertyMetadata(string.Empty));
```

- Registrieren über Statische Funktion
  - Name
  - Typ
  - Control Typ bzw. Klasse
  - Metadaten

Properties anlegen 1/2

(SearchTextBox.cs)

```
/// <summary>
/// Gets or sets the watermark.
/// </summary>
[Description("Gets or sets the watermark."), Category(PROPERTY_CATEGORY)]
public string Watermark
{
    get { return (string)GetValue(WatermarkProperty); }
    set { SetValue(WatermarkProperty, value); }
}
```



› Verwendung der Methoden GetValue(), SetValue()

Properties anlegen 2/2

(SearchTextBox.cs)

ROUTED COMMANDS

RoutedCommands

RoutedCommands

- RoutedCommand Klasse implementiert ICommand

RoutedCommands

- RoutedCommand Klasse implementiert ICommand
- Löst eine Aktion innerhalb eines Controls aus und muss abonniert werden

RoutedCommands

- RoutedCommand Klasse implementiert ICommand
- Löst eine Aktion innerhalb eines Controls aus und muss abonniert werden
- Verknüpfung mit Methode oder Funktion der Control Klasse per CommandBinding

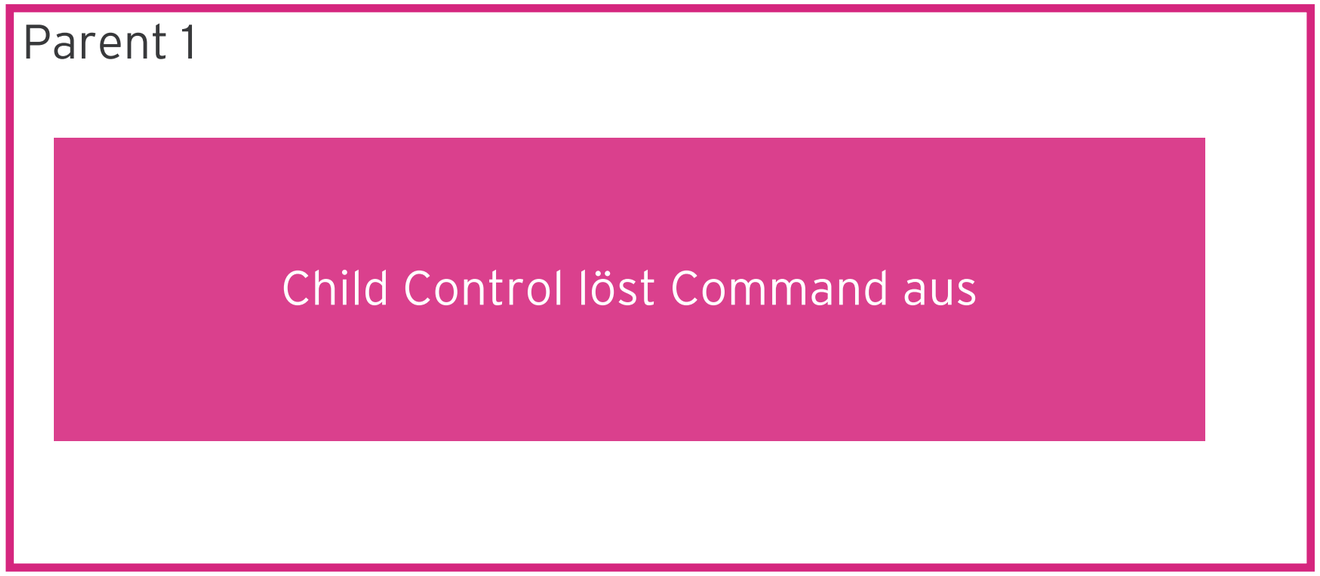
RoutedCommands

- RoutedCommand Klasse implementiert ICommand
- Löst eine Aktion innerhalb eines Controls aus und muss abonniert werden
- Verknüpfung mit Methode oder Funktion der Control Klasse per CommandBinding
- Kann über Command Property ausgelöst werden

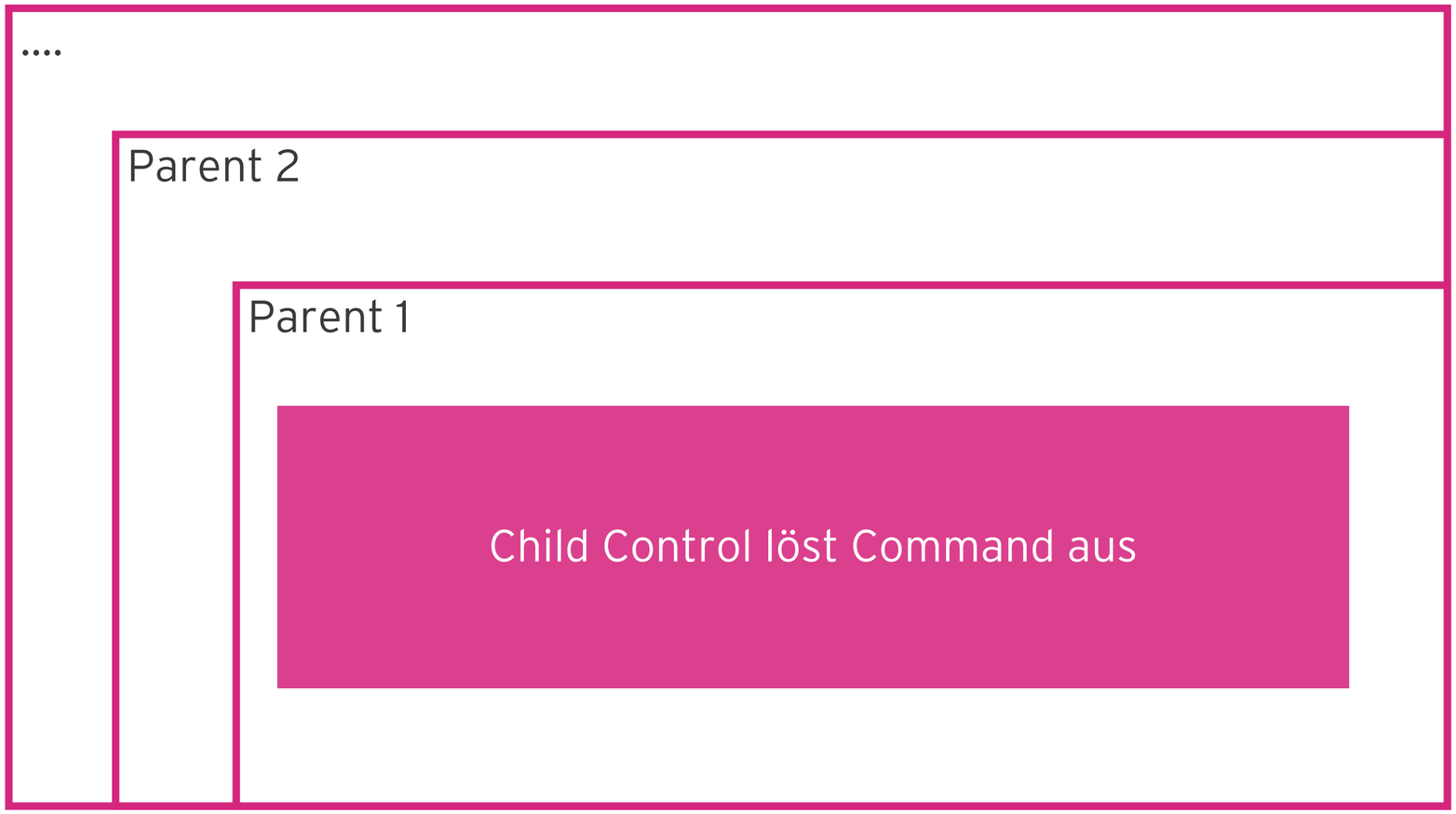
SEARCHTEXTBOX ROUTEDCOMMANDS

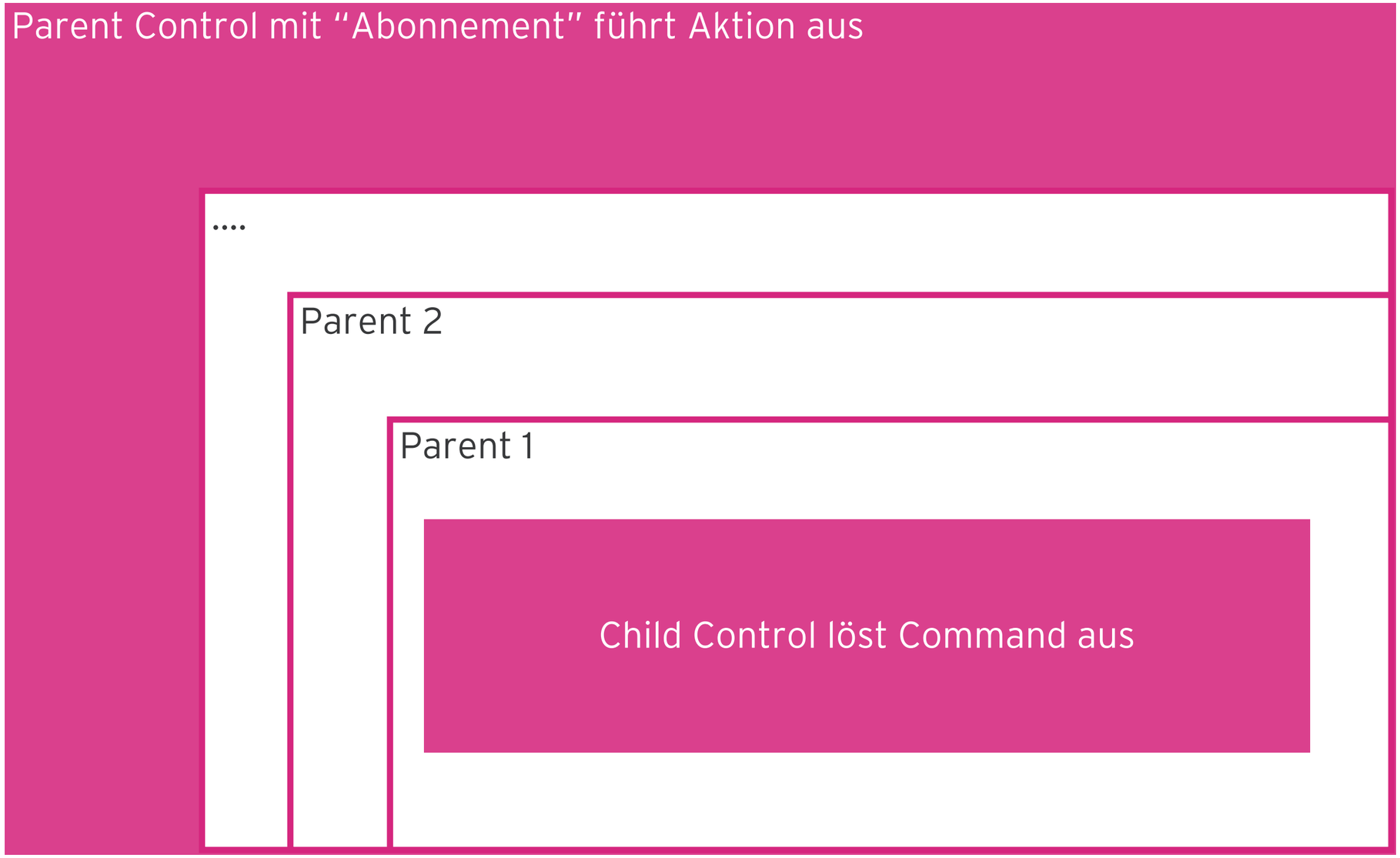
Child Control löst Command aus

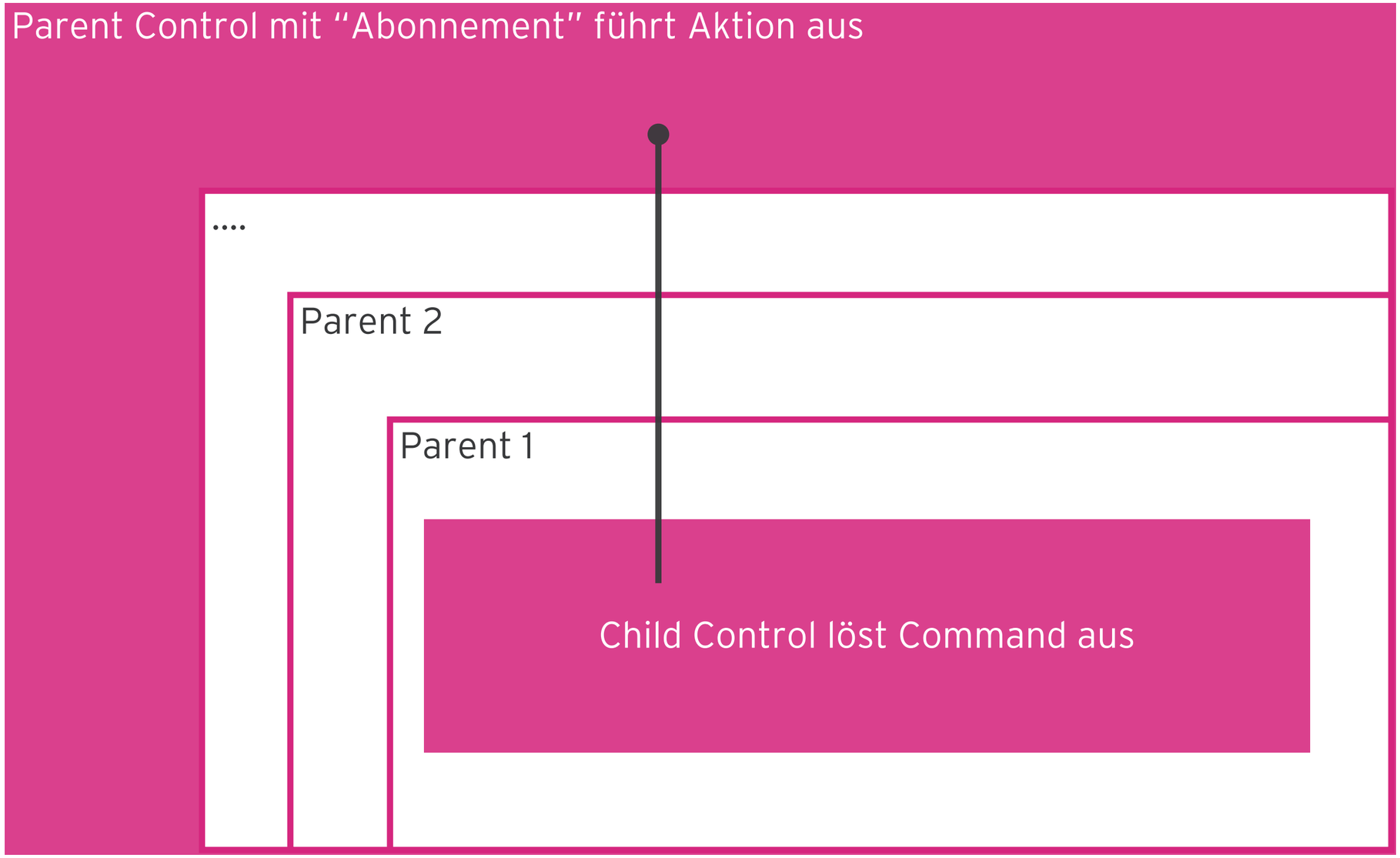
SEARCHTEXTBOX ROUTEDCOMMANDS




Parent 2
Parent 1
Child Control löst Command aus


....
Parent 2
Parent 1
Child Control löst Command aus

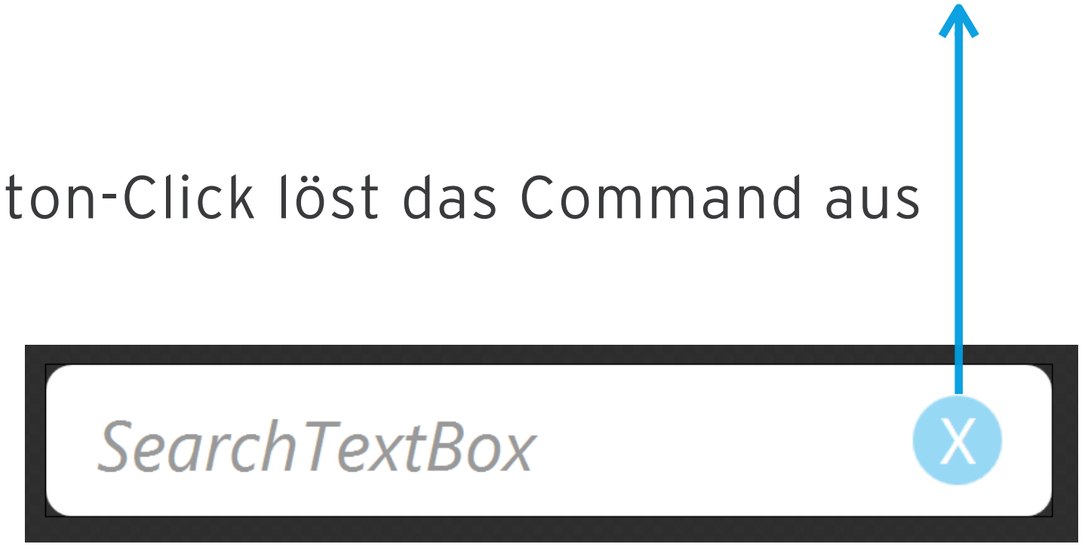

Parent Control mit “Abonnement” führt Aktion aus
....
Parent 2
Parent 1
Child Control löst Command aus


Parent Control mit “Abonnement” führt Aktion aus
....
Parent 2
Parent 1
Child Control löst Command aus

SEARCHTEXTBOX ROUTEDCOMMANDS

SEARCHTEXTBOX ROUTEDCOMMANDS

1.Button-Click löst das Command aus



2. SearchTextBox mit

“Abonnement” führt Aktion aus

1.Button-Click löst das Command aus

3. Command stoppt

2. SearchTextBox mit

“Abonnement” führt Aktion aus

1.Button-Click löst das Command aus

- Routed Command anlegen
  - Name
  - Control Typ bzw. Klasse

```
/// <summary>
/// ClearTextCommand - This command clears the text property.
/// </summary>
public static readonly RoutedCommand ClearTextCommand = new RoutedCommand("ClearTextCommand", typeof(SearchTextBox));

/// <summary> ...
private static void ExecuteClearTextCommand(object sender, ExecutedRoutedEventArgs e)
{
    SearchTextBox self = sender as SearchTextBox;
    self.Clear();
}

/// <summary> ...
private static void CanExecuteClearCommand(object sender, CanExecuteRoutedEventArgs e)
{
    SearchTextBox self = sender as SearchTextBox;
    e.CanExecute = self.CanClearText();
}
```

- Execute Aktion
- CanExecute Abfrage

Routed Command 1/3

(SearchTextBox.cs)

```
/// <summary>
/// static ctor
/// </summary>
static SearchTextBox()
{
    DefaultStyleKeyProperty.OverrideMetadata(
        typeof(SearchTextBox),
        new FrameworkPropertyMetadata(typeof(SearchTextBox)));

    CommandManager.RegisterClassCommandBinding(
        typeof(SearchTextBox),
        new CommandBinding(ClearTextCommand, ExecuteClearTextCommand, CanExecuteClearCommand));
}
```

- Command Binding erzeugen
  - Command
  - Execute
  - CanExecute

Routed Command 2/3

(SearchTextBox.cs)

```
<Button Grid.Column="1"
        Command="{x:Static local:SearchTextBox.ClearTextCommand}"/>
```

- Auslösen über Command Property / x:Static Verweis

Routed Command 3/3

(Generic.xaml)

VIEWMODEL KOMMUNIKATION

ViewModel Kommunikation - Basics

ViewModel Kommunikation - Basics

- Binding (Implementierung *INotifyPropertyChanged*)

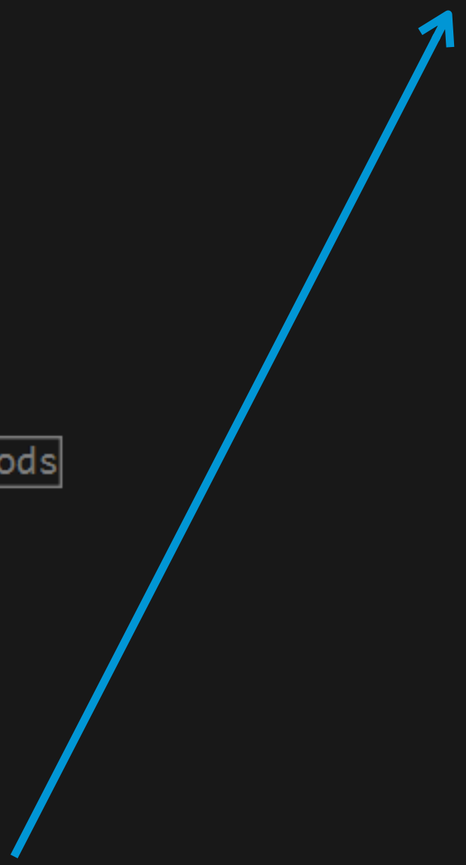
```
<SearchTextBox x:Name="SearchTextBox"
               Margin="0 0 0 10"
               Text="{Binding SearchString,
                      UpdateSourceTrigger=PropertyChanged}"
Watermark="Search Points" />
```

- Binding an ViewModel Property

ViewModel Kommunikation 1/3

(MainView.xaml)

```
public class SampleViewModel : INotifyPropertyChanged
{
    Members

    Commands

    Ctor

    Init/Cleanup

    OverrideMethods

    Methods

    Properties
}
```



- Implementieren von *INotifyPropertyChanged*

ViewModel Kommunikation 2/3

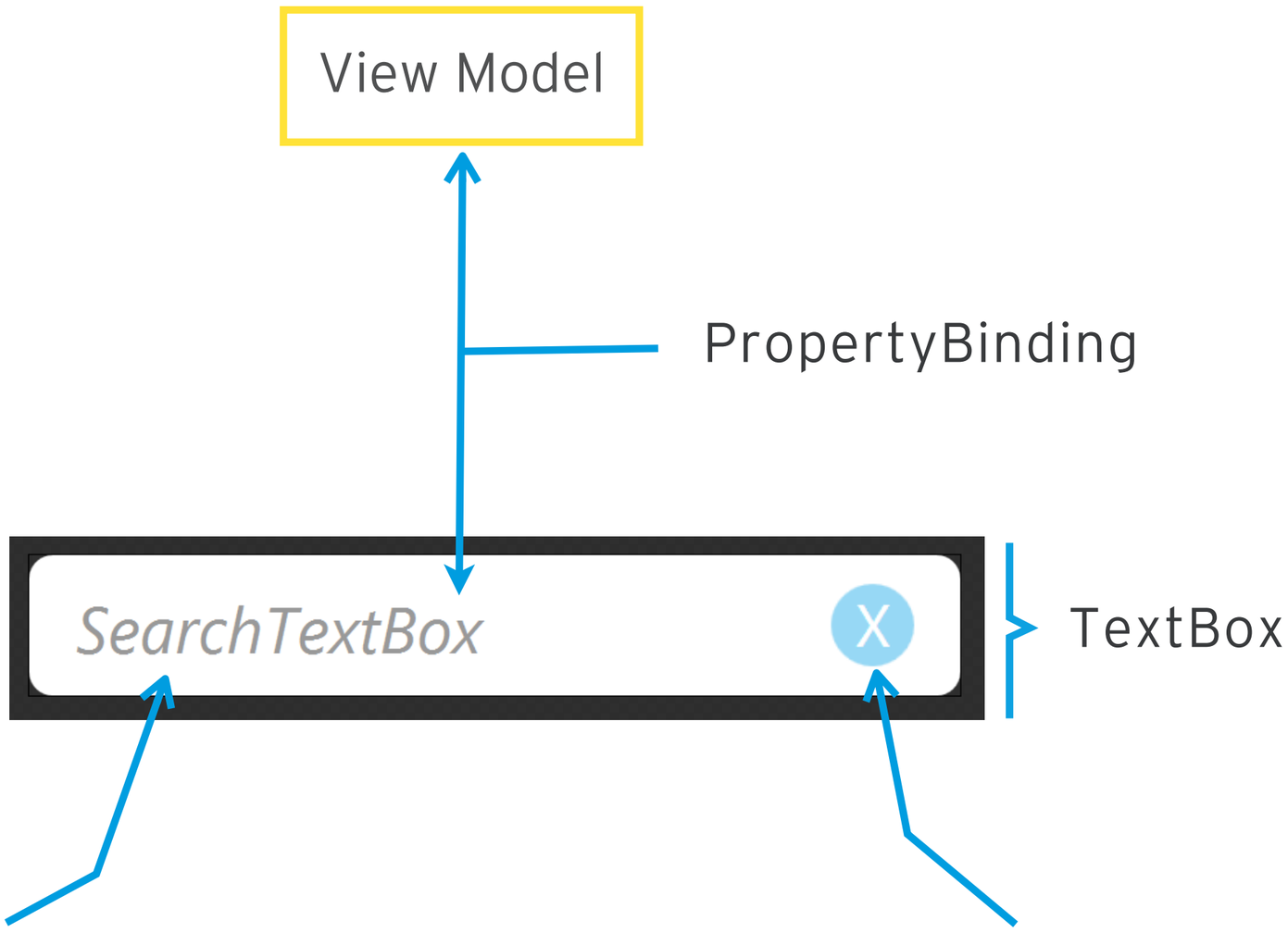
(SampleViewModel.cs)

```
/// <summary> ...
private string searchString;

/// <summary> ...
public string SearchString
{
    get
    {
        return searchString;
    }

    set
    {
        if (value != this.searchString)
        {
            this.searchString = value;
            UpdateHighlighting();
            OnPropertyChanged("SearchString");
        }
    }
}
```

- Aufruf der Update Logik
- Aufrufen von OnPropertyChanged (Update des Bindings)

ViewModel Kommunikation 3/3

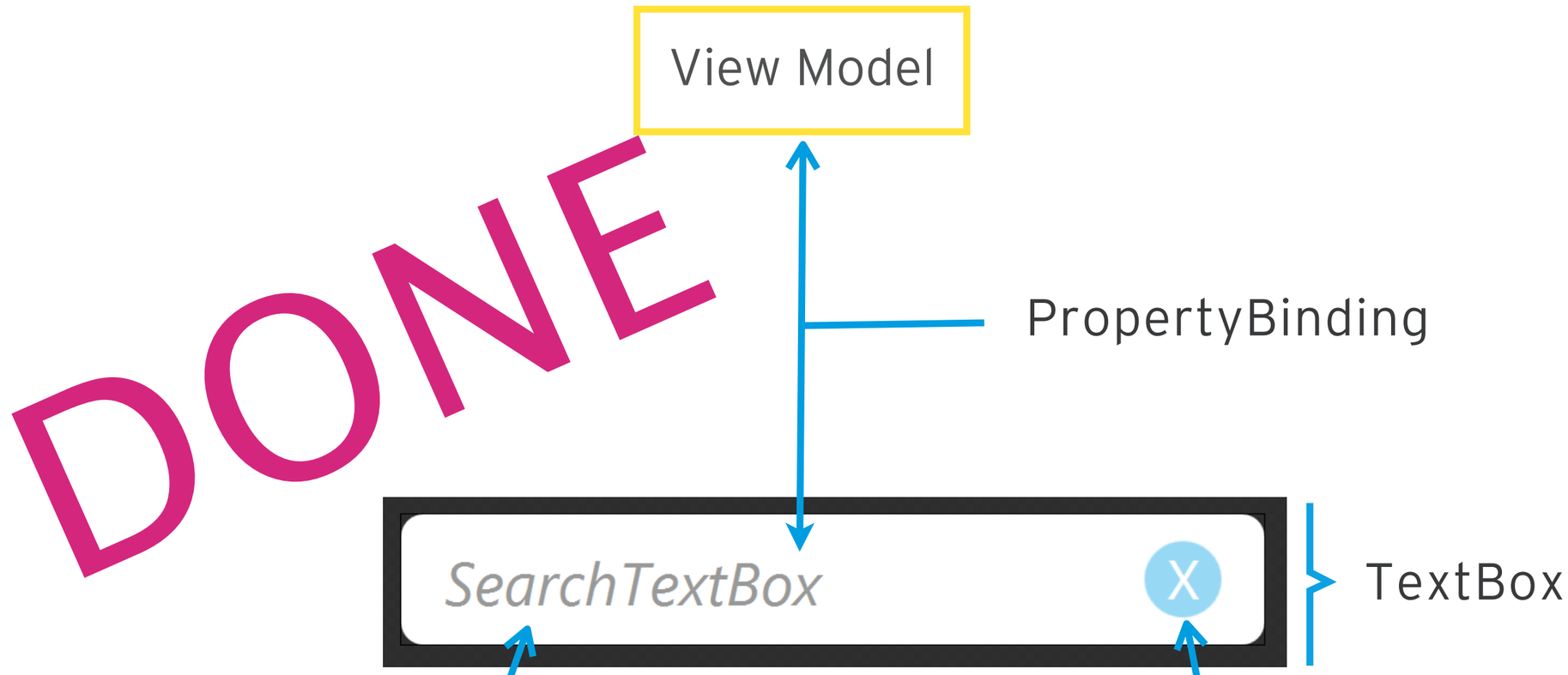
(SampleViewModel.cs)



- Watermark
- WatermarkFontStyle
- WatermarkFontWeight
- WatermarkForeground

ClearCommand

(Rucksetzen des Texts)



- Watermark
- WatermarkFontStyle
- WatermarkFontWeight
- WatermarkForeground

ClearCommand

(Rucksetzen des Texts)

POINTCHART

André Lanninger
UIDeveloper

Datagraph 11

ChangeStyle

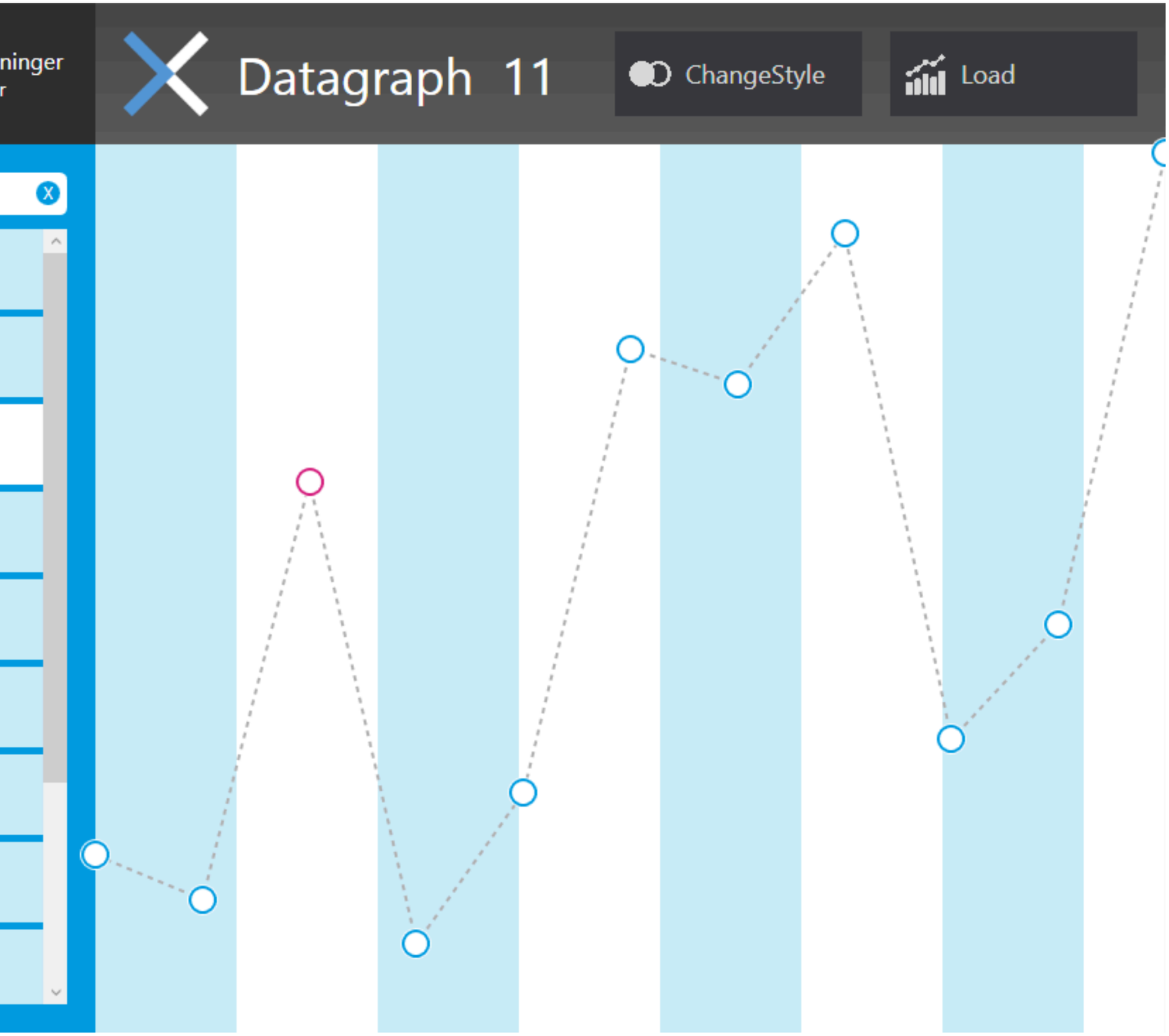
Load

62

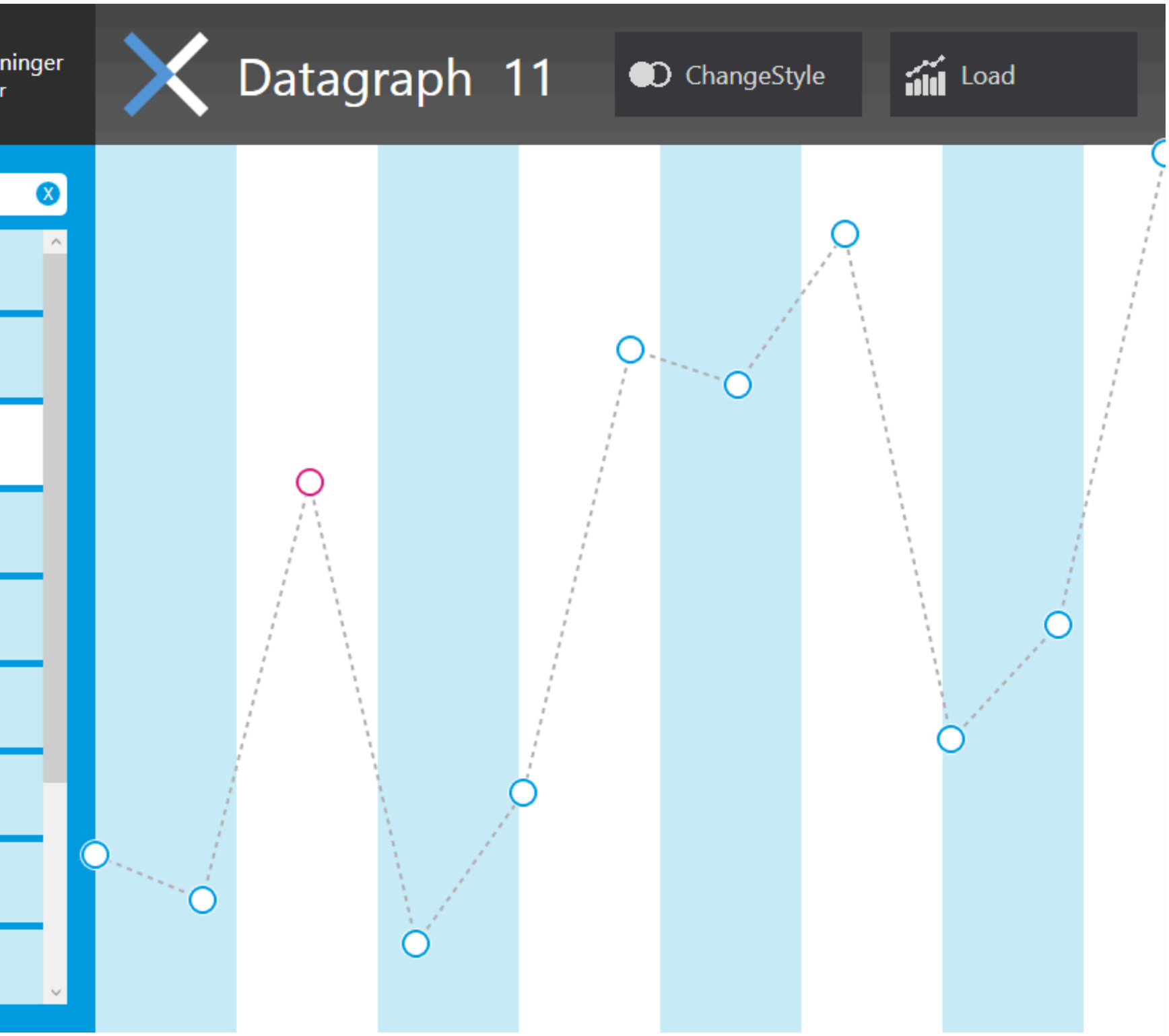
- Bubble 18
  0, 20
- Bubble 19
  10, 15
- Bubble 20
  20, **62**
- Bubble 21
  30, 10
- Bubble 22
  40, 27
- Bubble 23
  50, 77
- Bubble 24
  60, 73
- Bubble 25
  70, 90
- Bubble 26
  80, 33

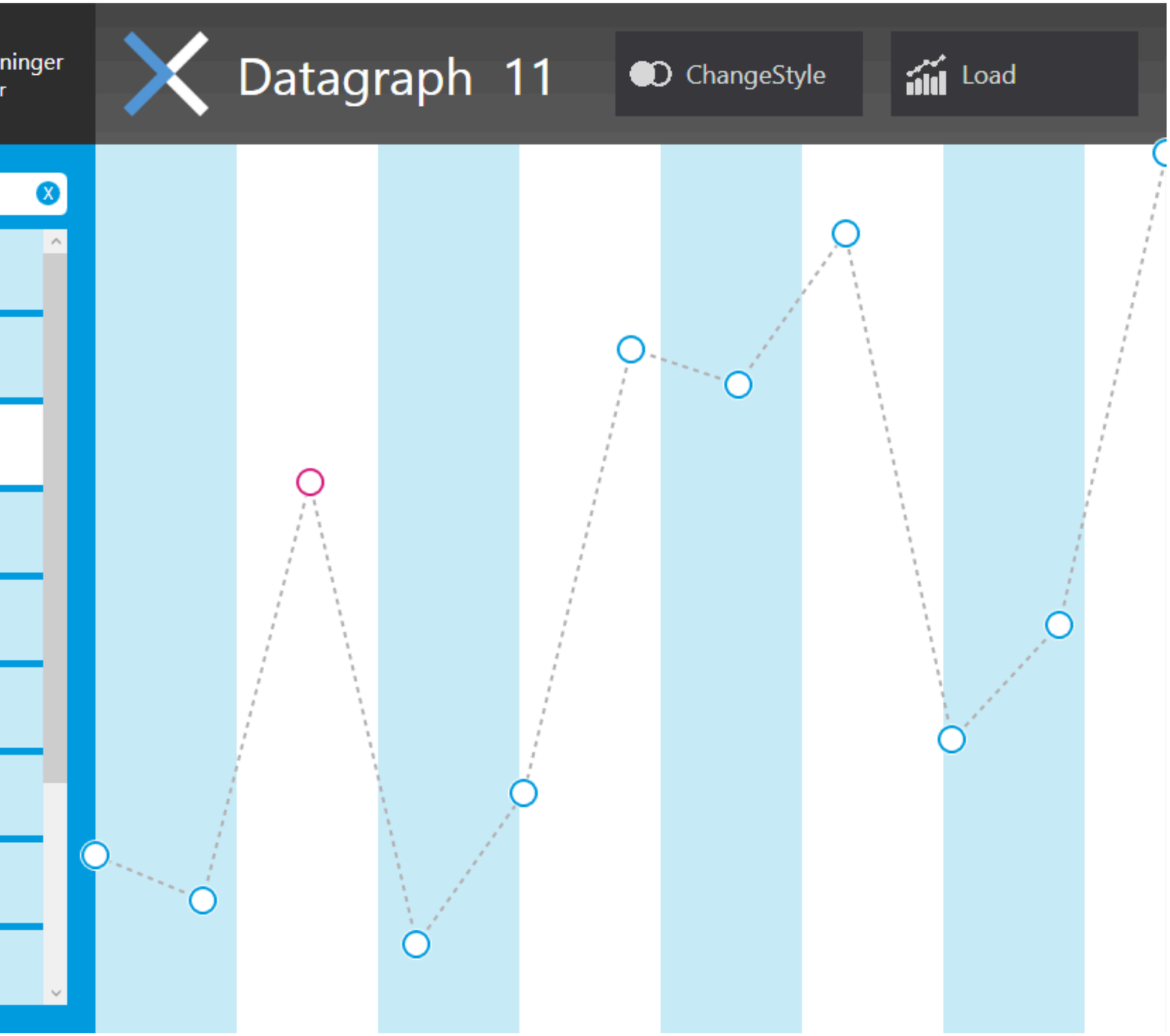
PointChart

ANALYSE


Datagraph 11
ChangeStyle
Load

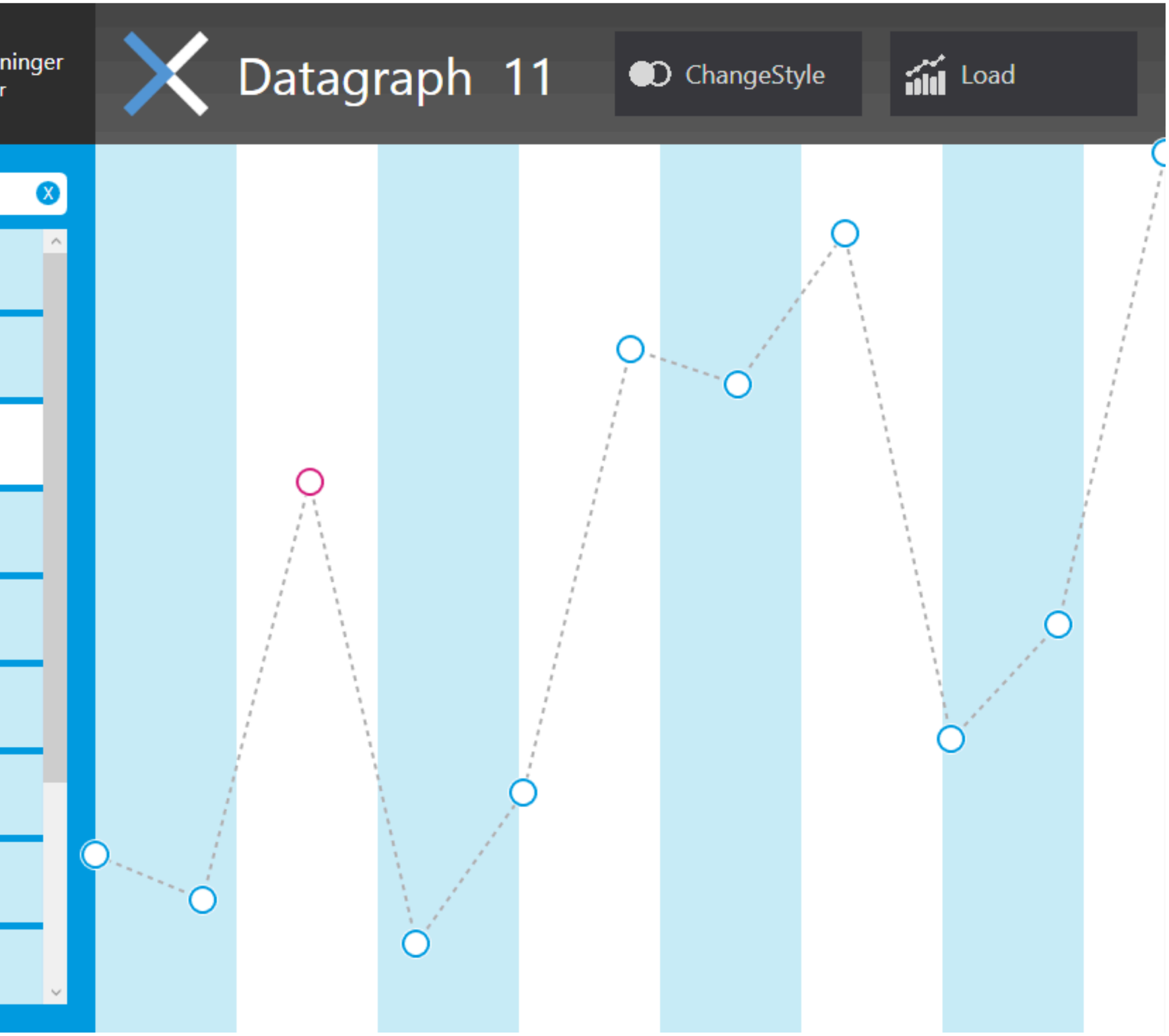
PointChartViewModel

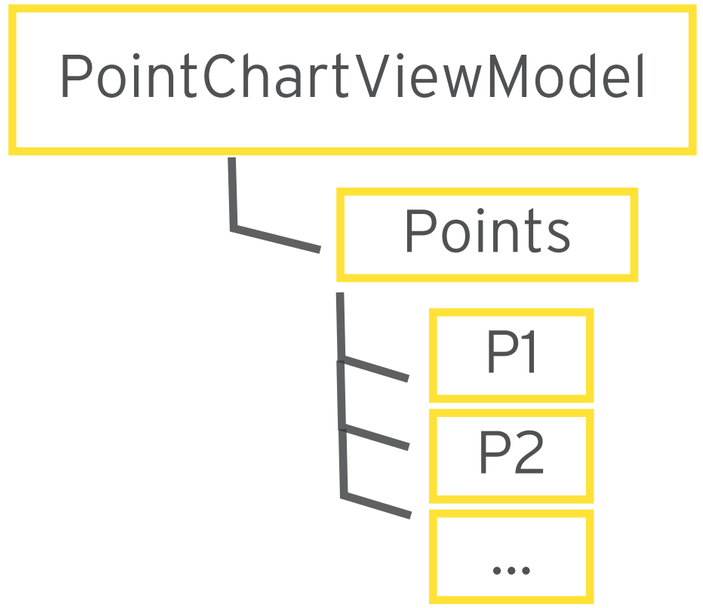


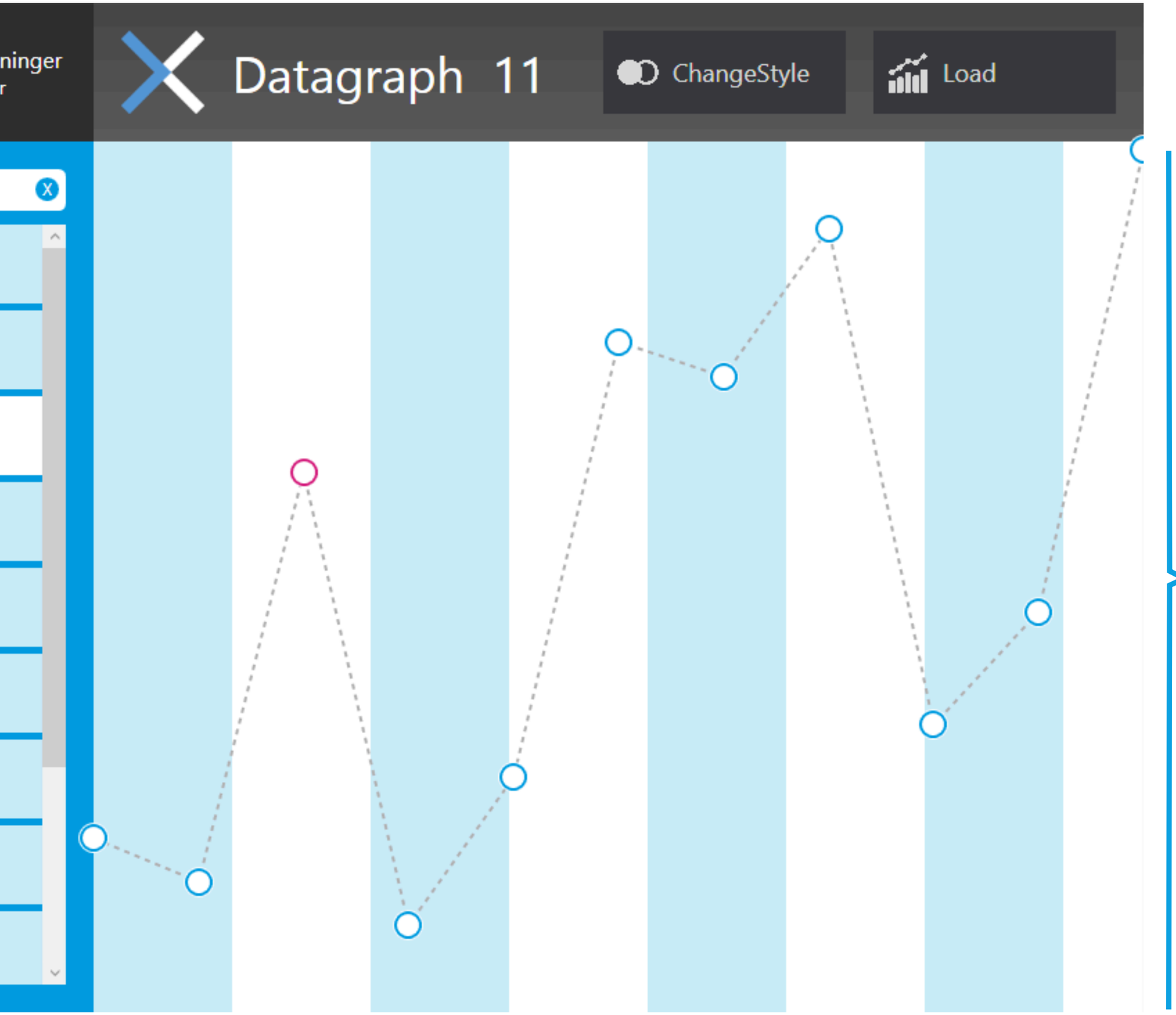


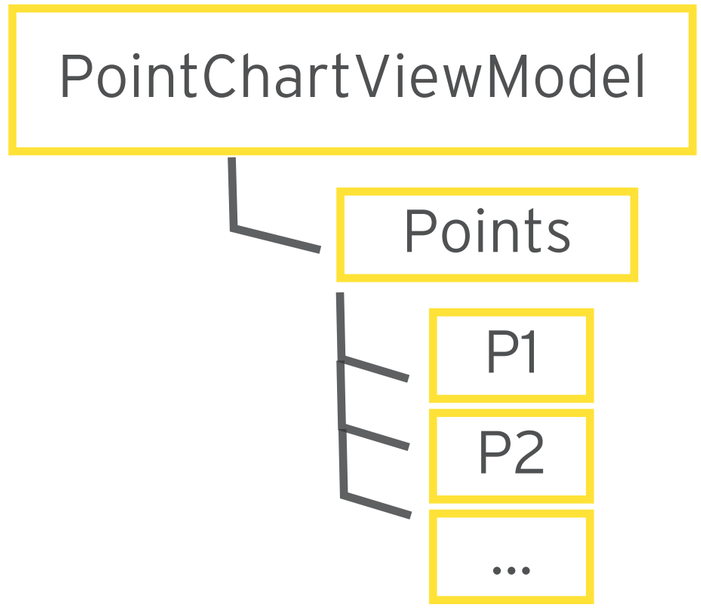
PointChartViewModel

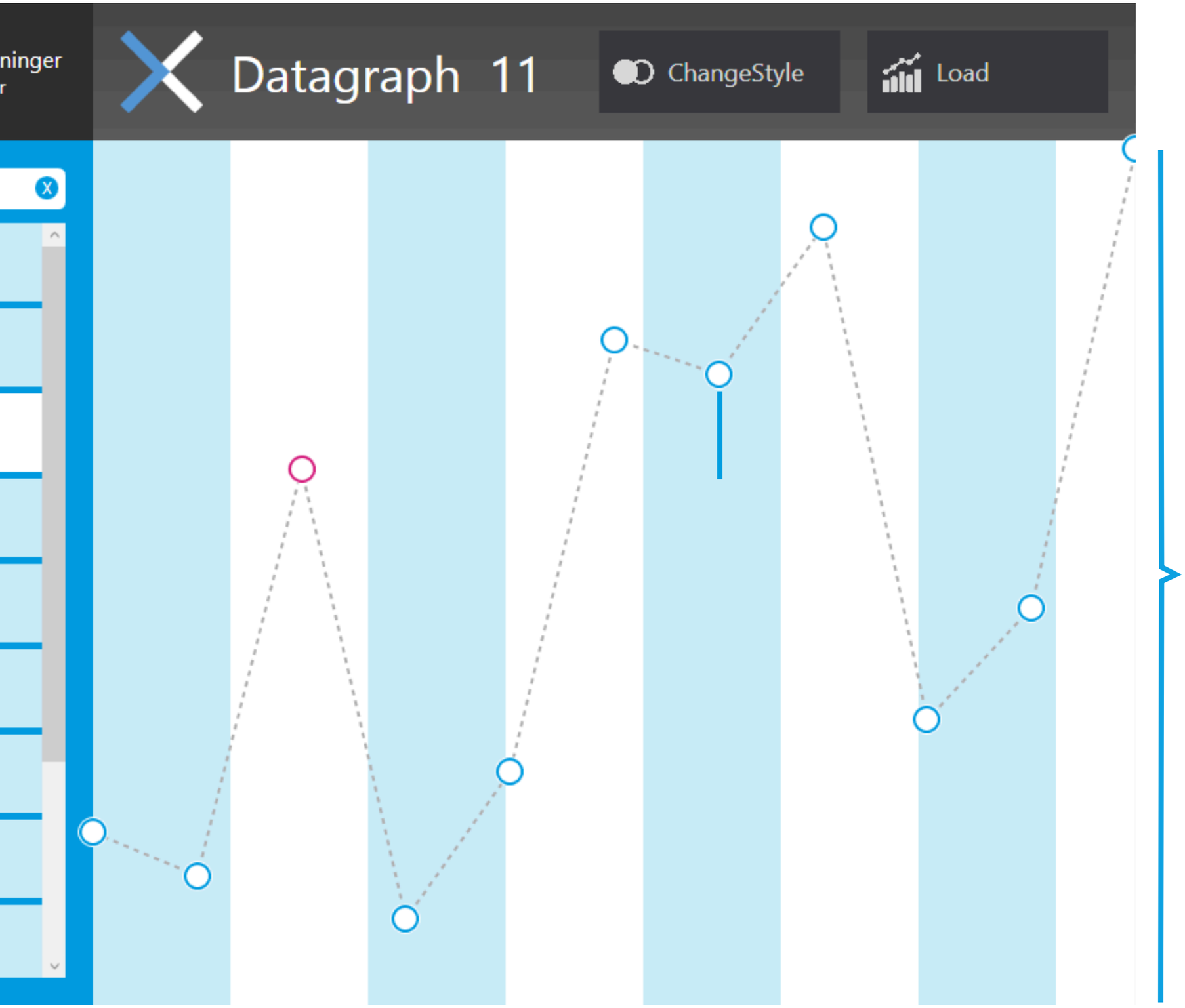
Points

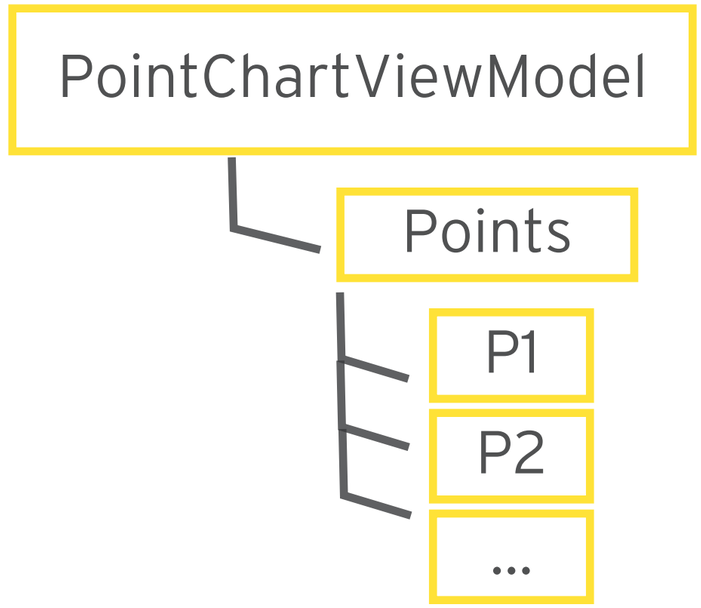

ninger
r
Datagraph 11
ChangeStyle
Load

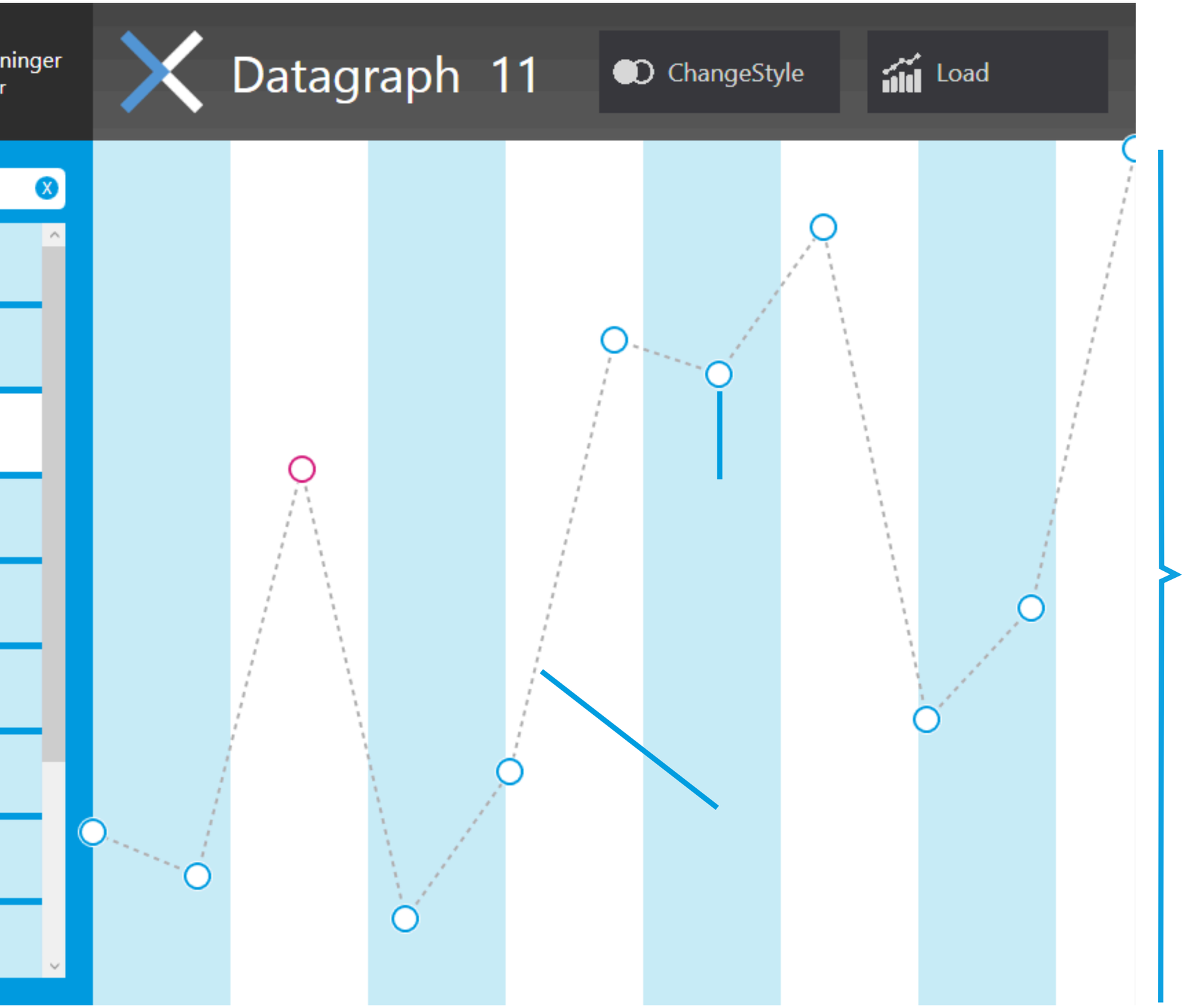

PointChartViewModel
Points
P1
P2
...

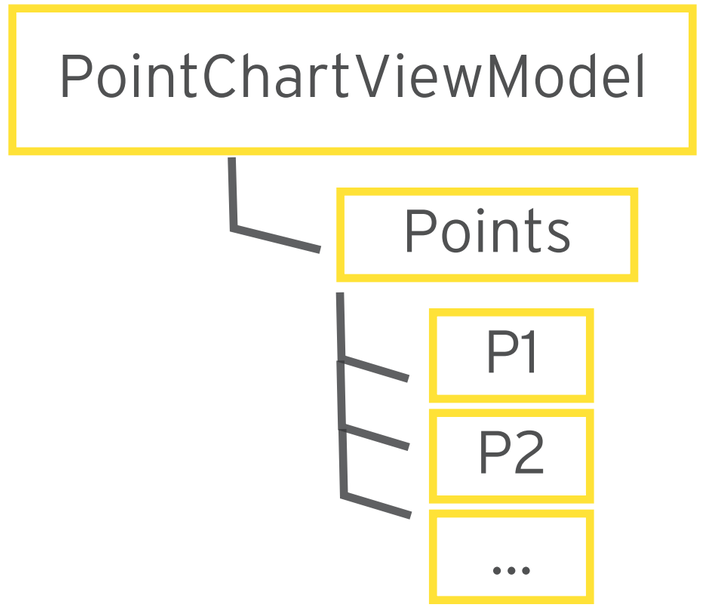

ninger
r
Datagraph 11
ChangeStyle
Load

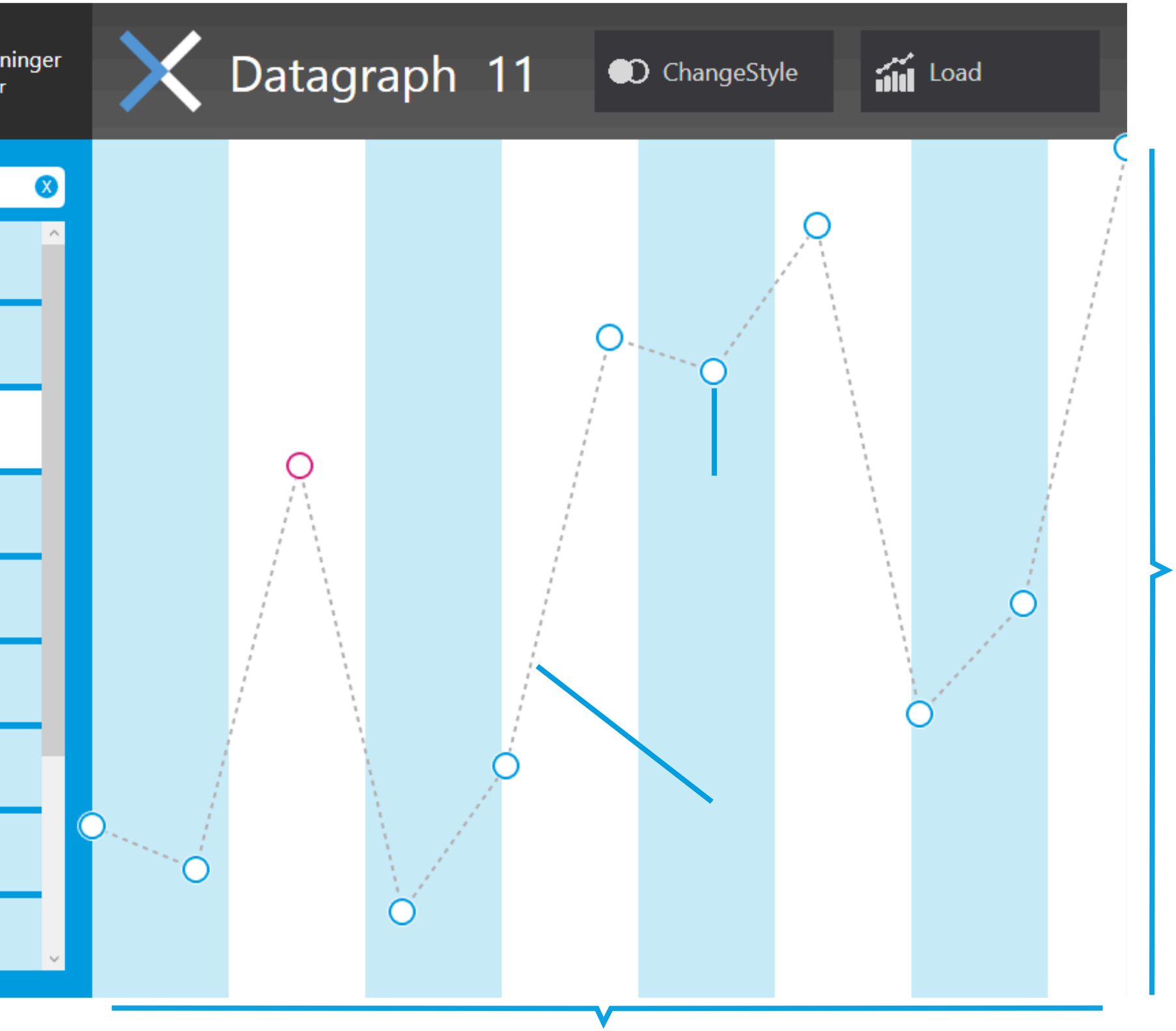

PointChartViewModel
Points
P1
P2
...

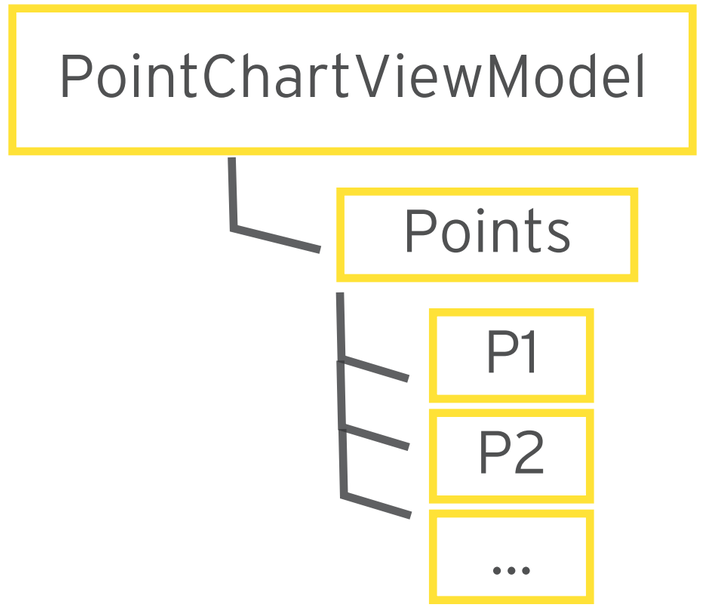

ninger
r
Datagraph 11
ChangeStyle
Load

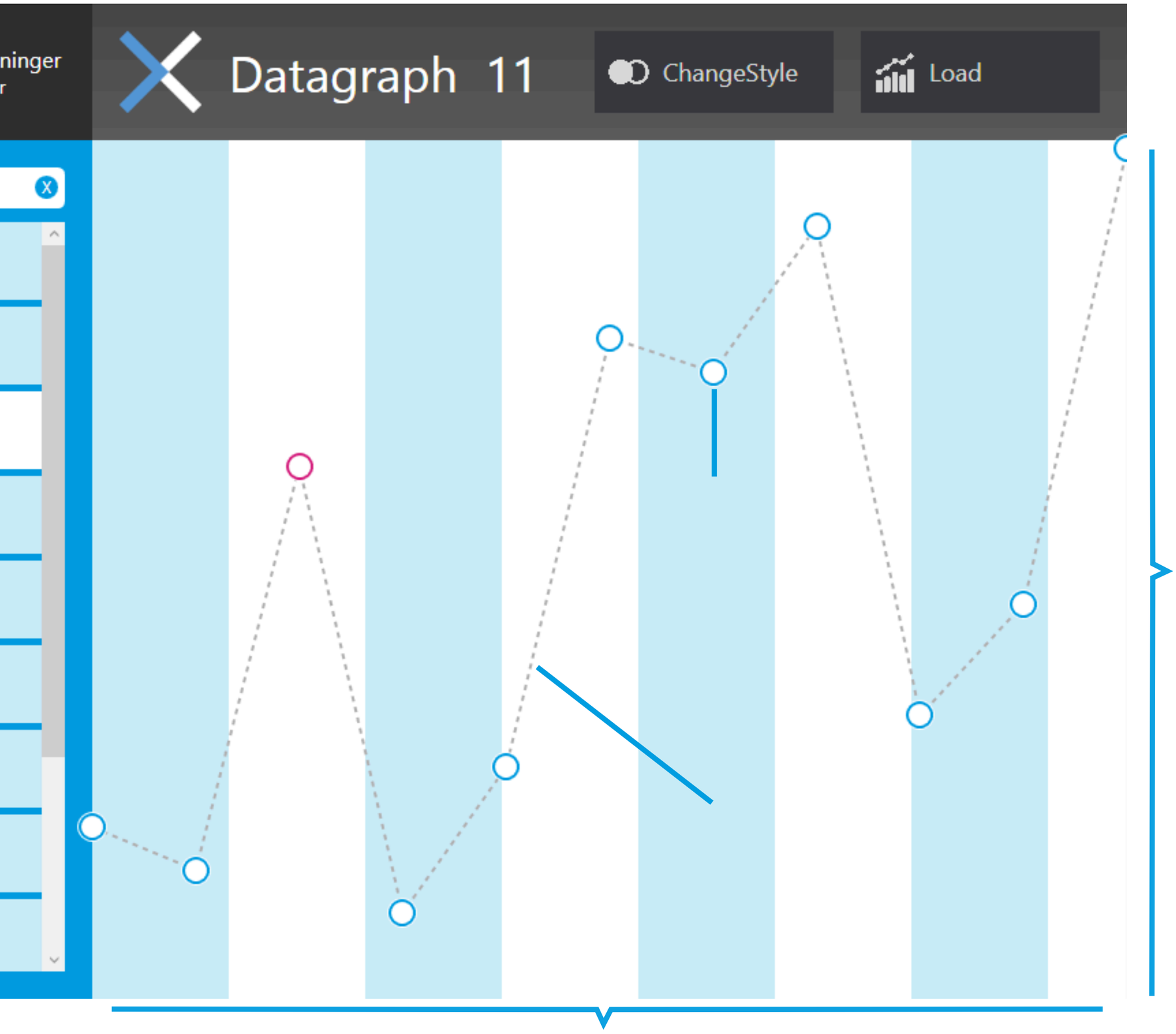

PointChartViewModel
Points
P1
P2
...


ninger
r
Datagraph 11
ChangeStyle
Load


PointChartViewModel
Points
P1
P2
...


ninger
r
Datagraph 11
ChangeStyle
Load


PointChartViewModel
Points
P1
P2
...



PointChartViewModel

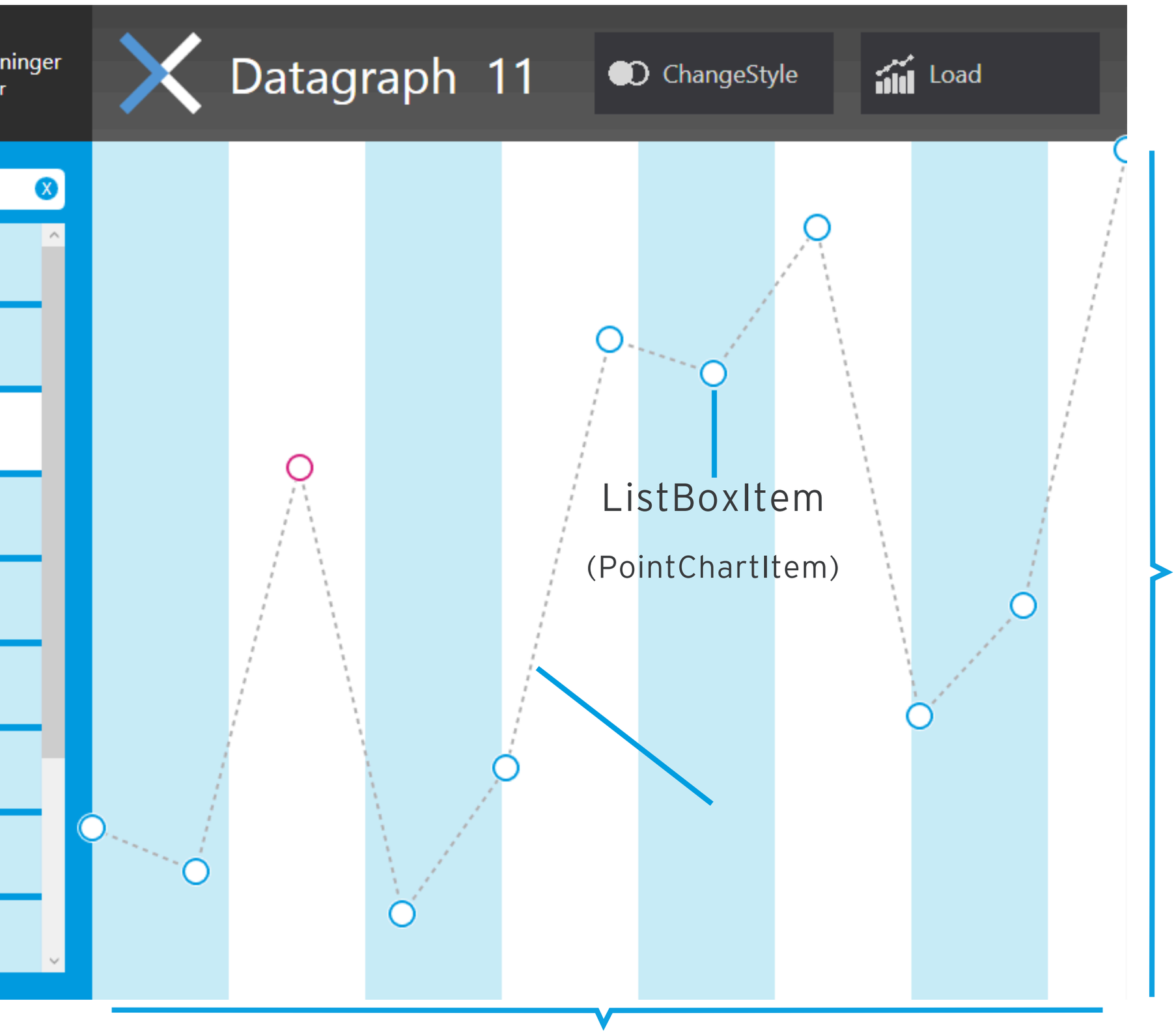
Points

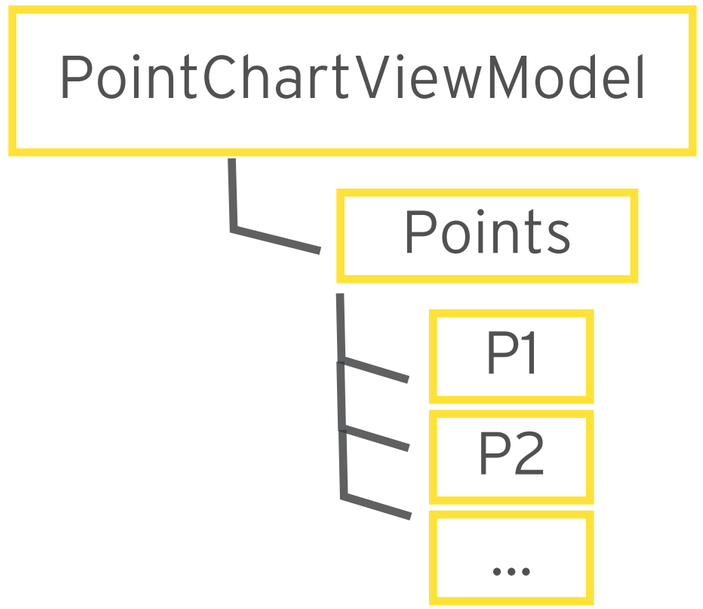
P1

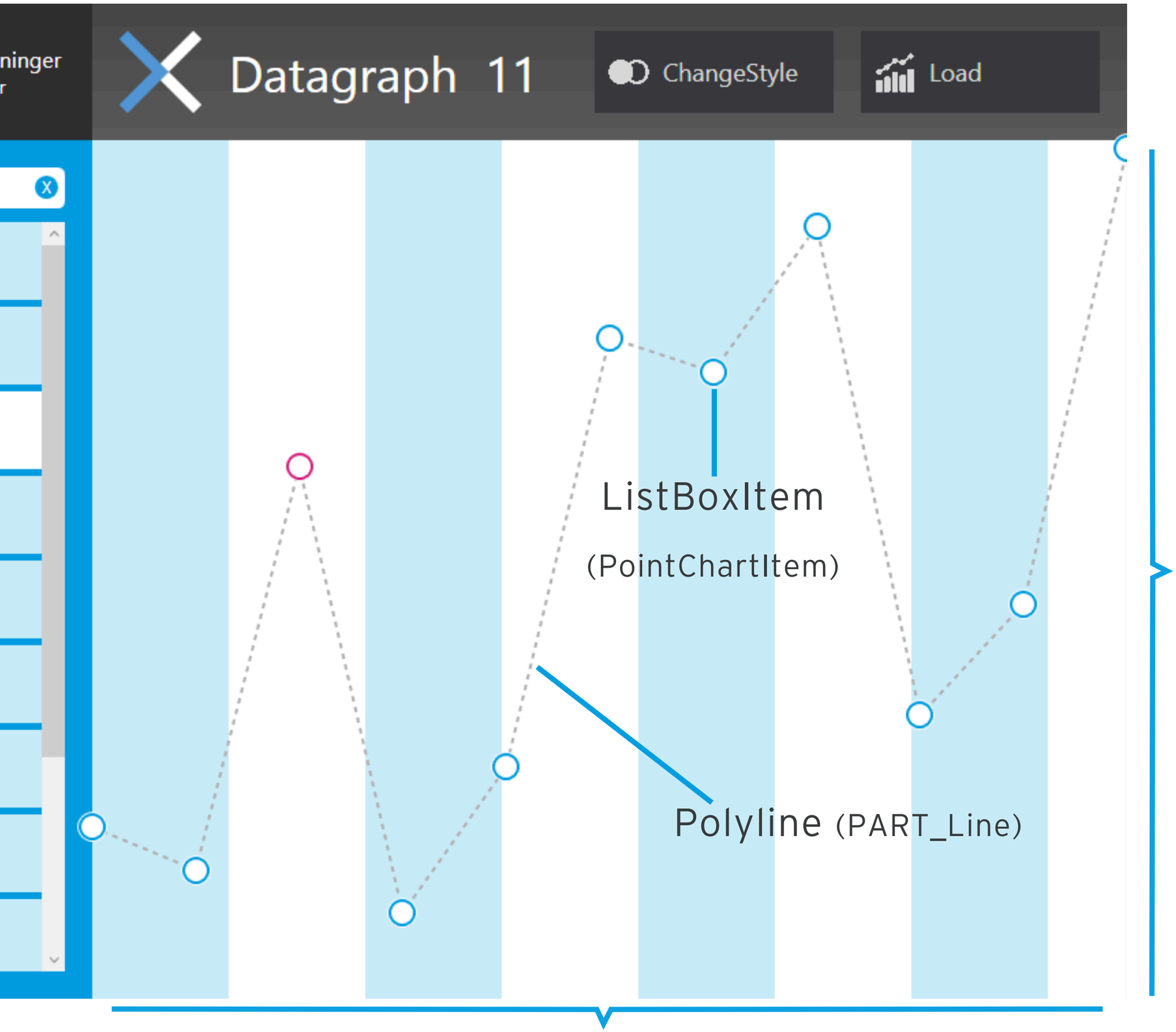
P2

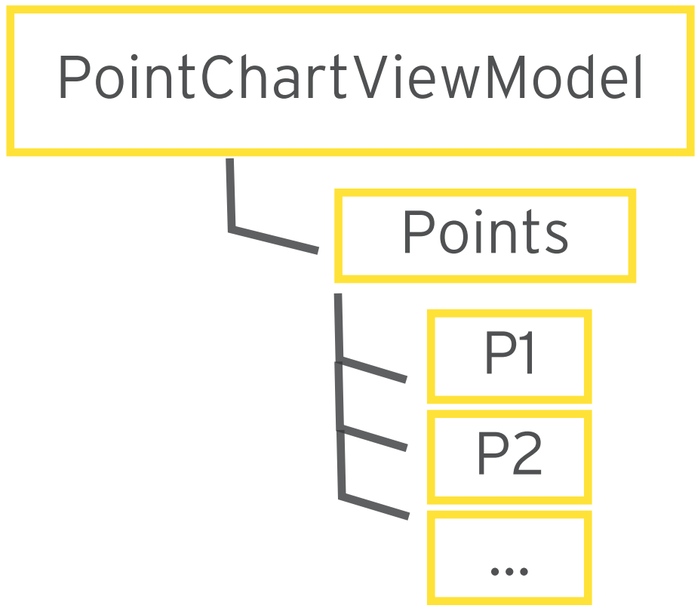
...

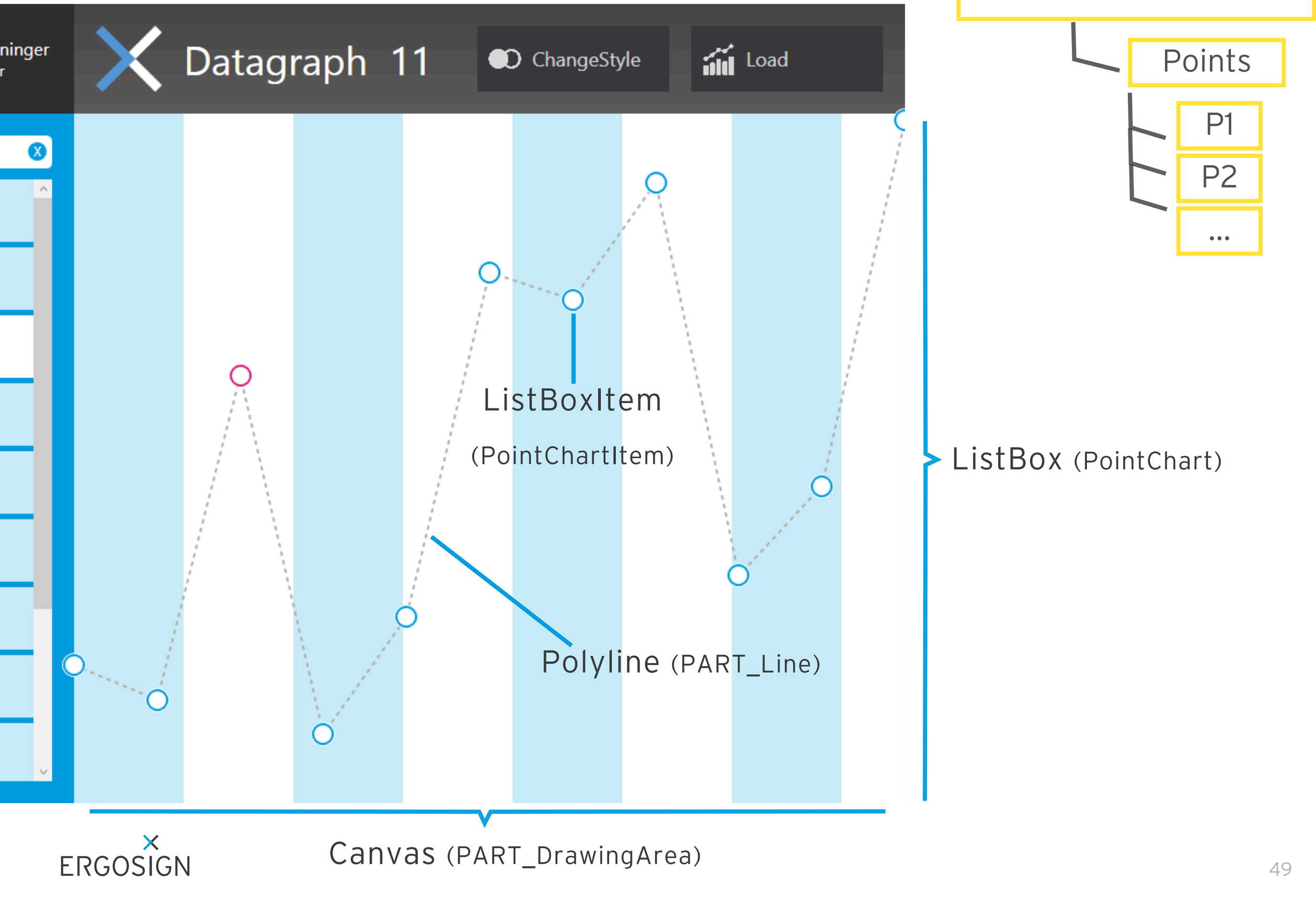
ListBox (PointChart)


ninger
r
Datagraph 11
ChangeStyle
Load
ListBoxItem
(PointChartItem)
ListBox (PointChart)

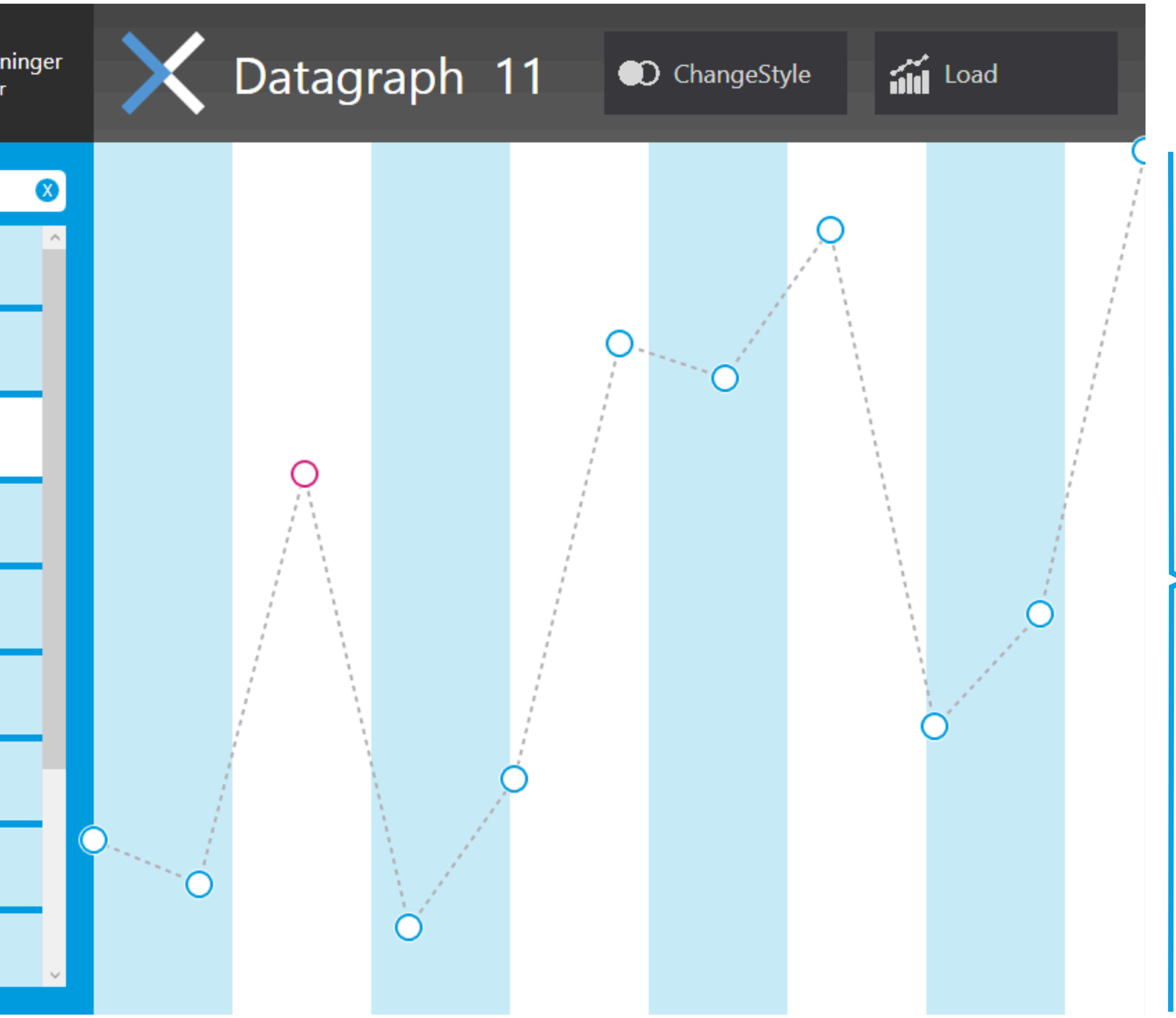

PointChartViewModel
Points
P1
P2
...


ninger
r
Datagraph 11
ChangeStyle
Load
ListBoxItem
(PointChartItem)
Polyline (PART_Line)
ListBox (PointChart)


PointChartViewModel
Points
P1
P2
...


Datagraph 11
ChangeStyle
Load
ListBoxItem
(PointChartItem)
Polyline (PART_Line)
ListBox (PointChart)
Canvas (PART_DrawingArea)

PointChartViewModel
Points
P1
P2
...



PointChartViewModel

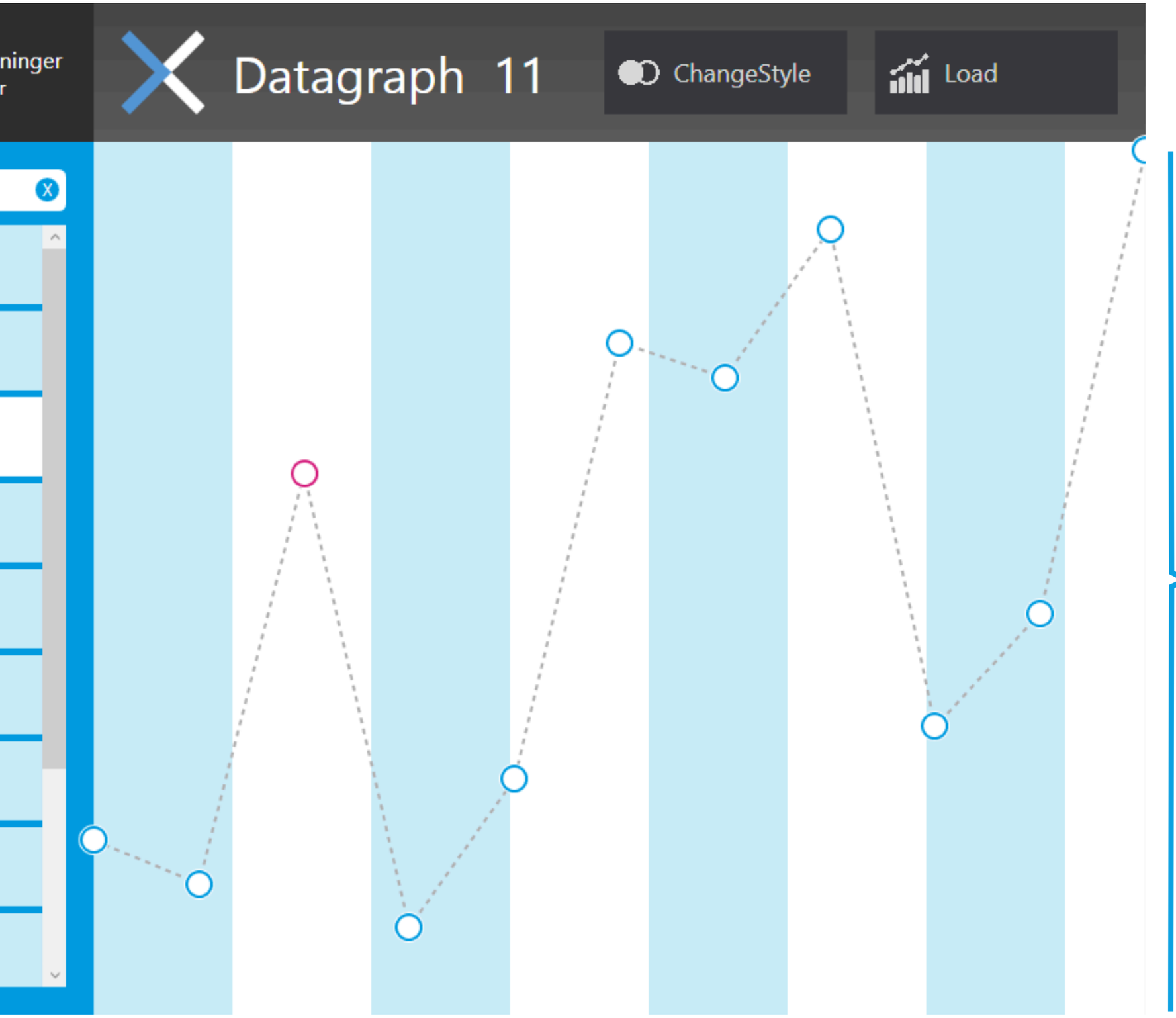
Points

P1

P2

...

ListBox (PointChart)

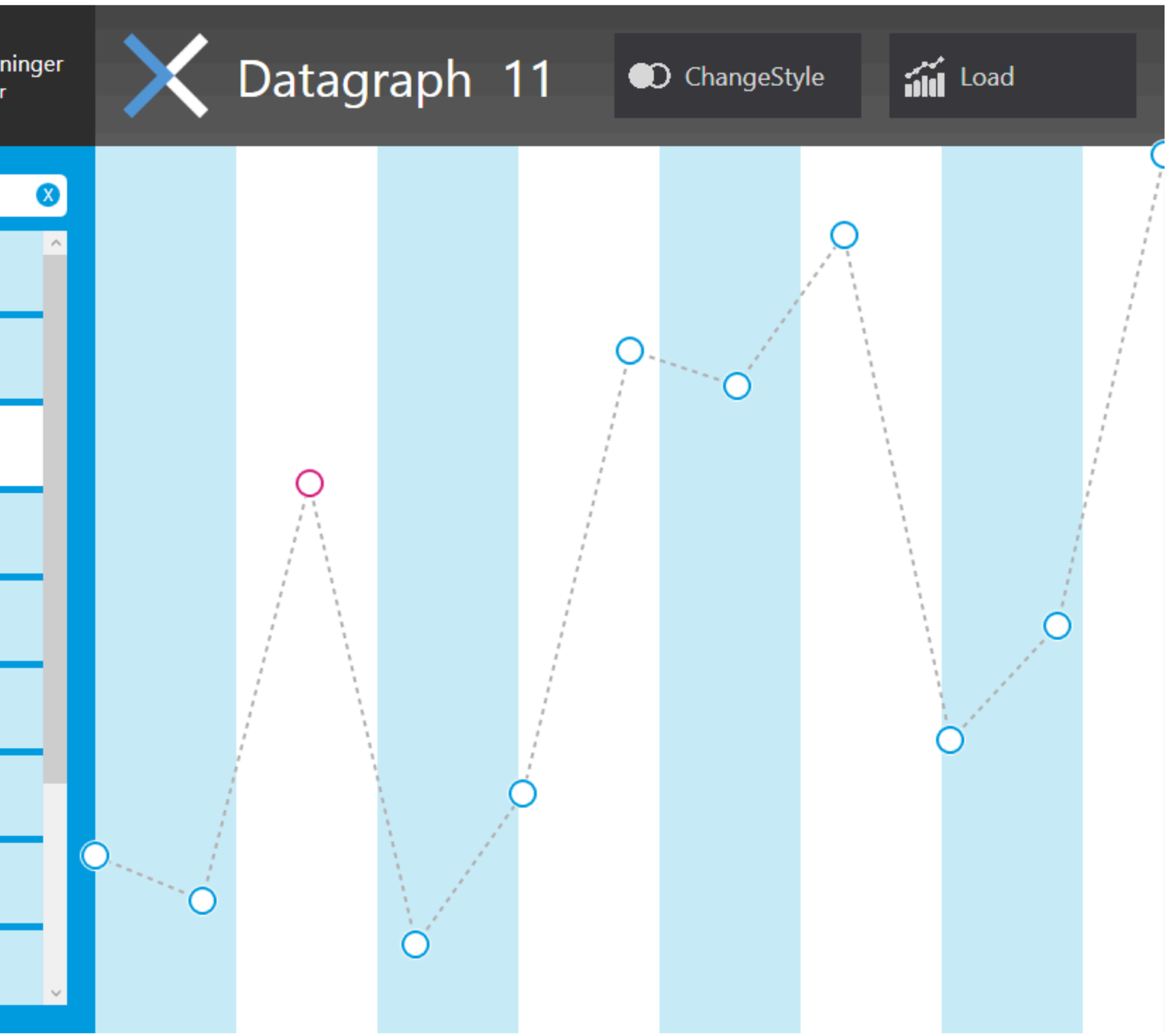


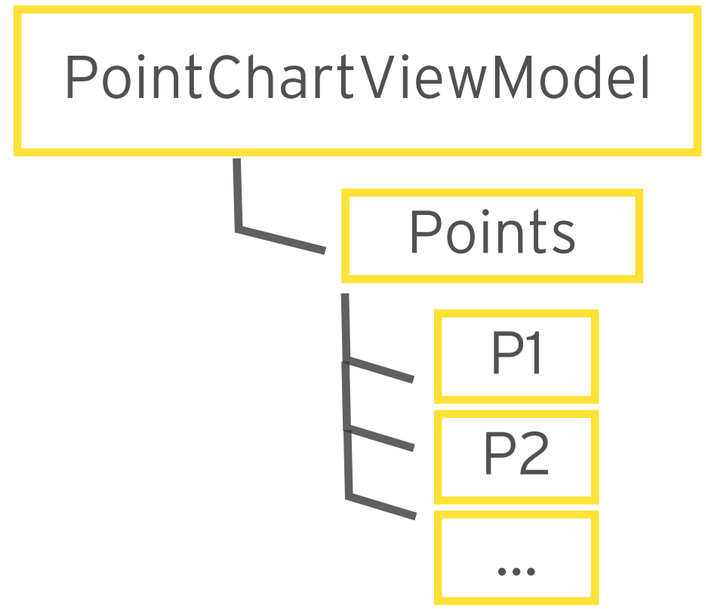
PointChartViewModel

- Points
  - P1
  - P2
  - ...

ListBox (PointChart)

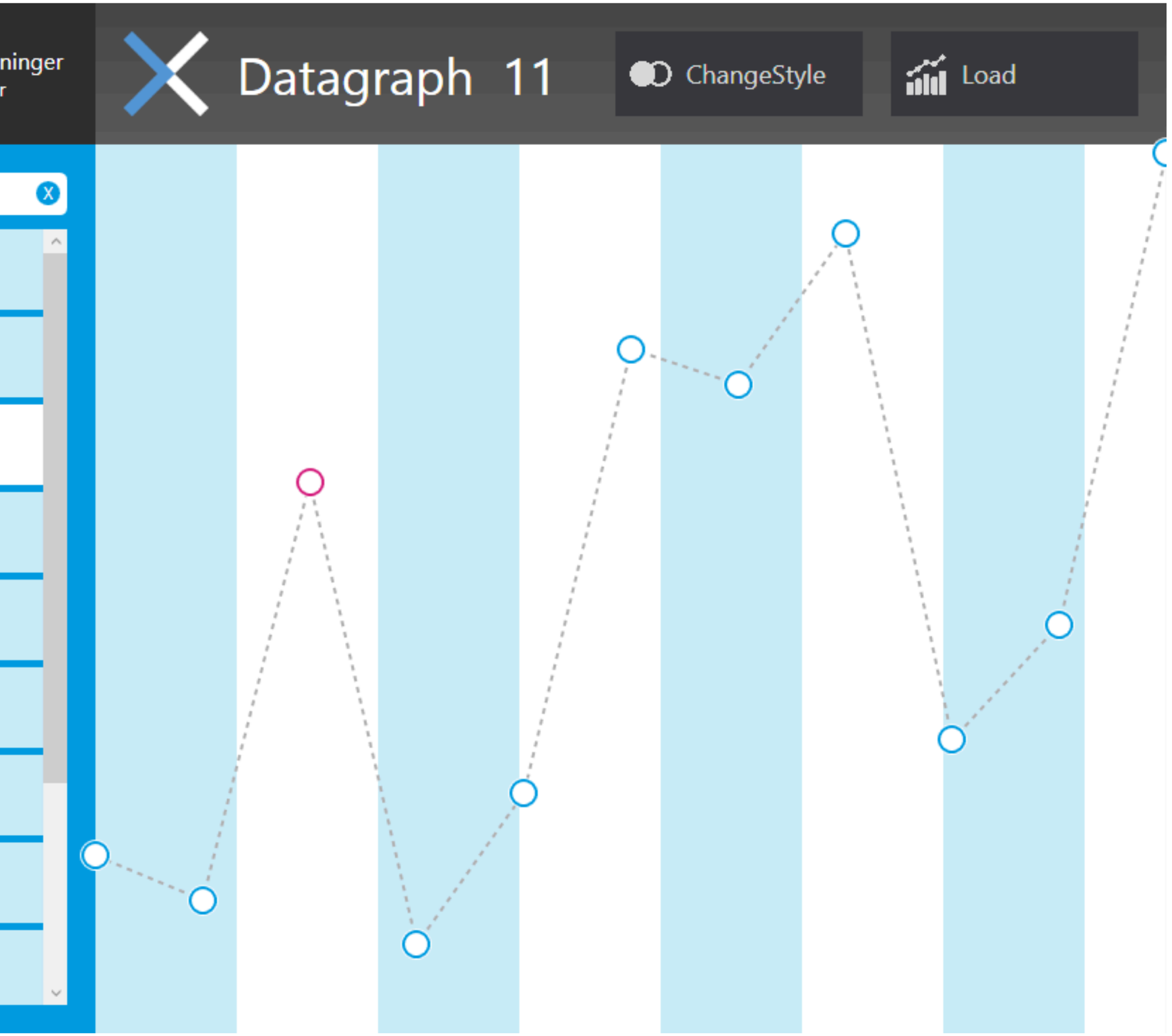
› X/YMinimum





ListBox (PointChart)

- X/YMinimum
- X/YMaximum

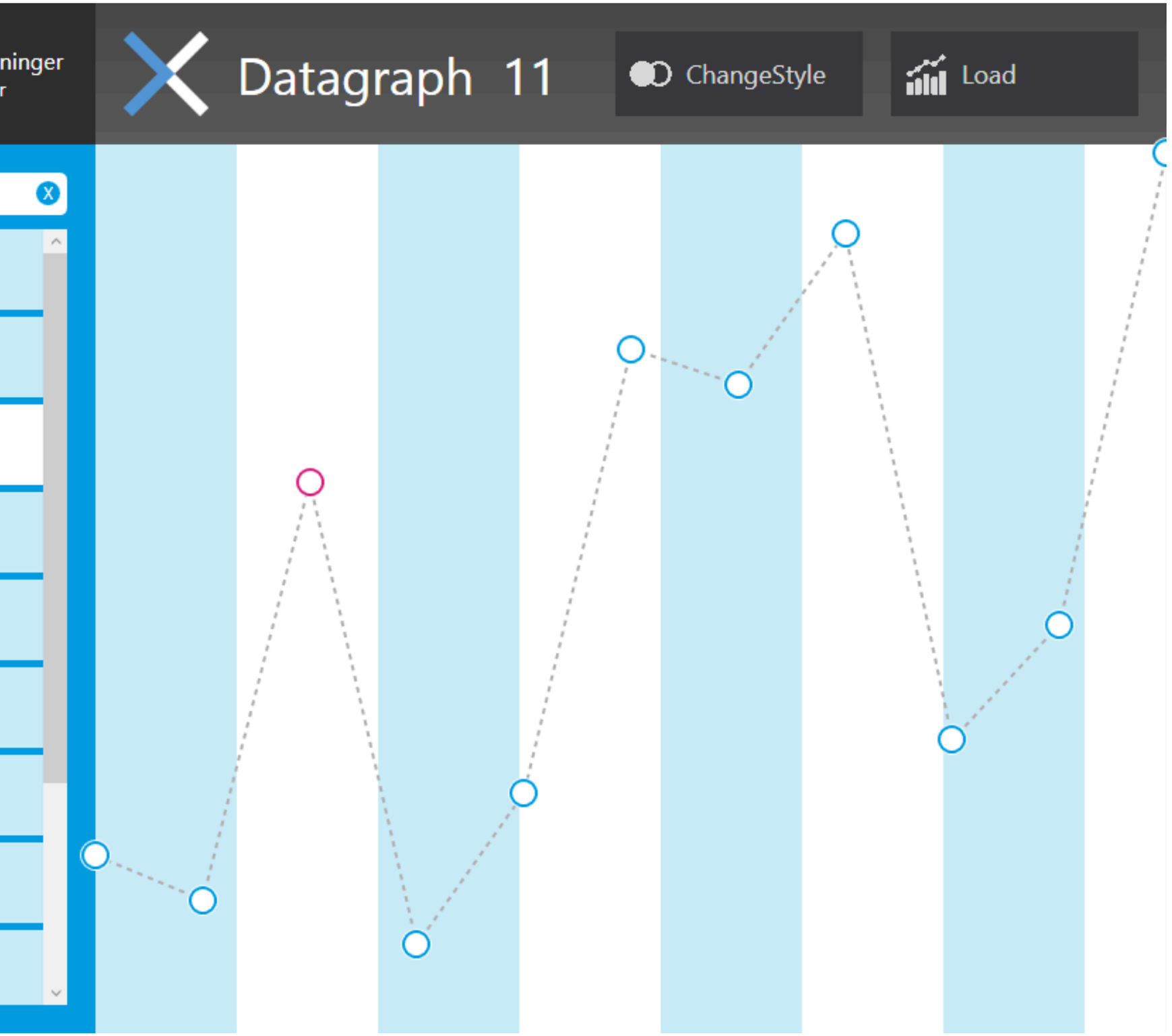


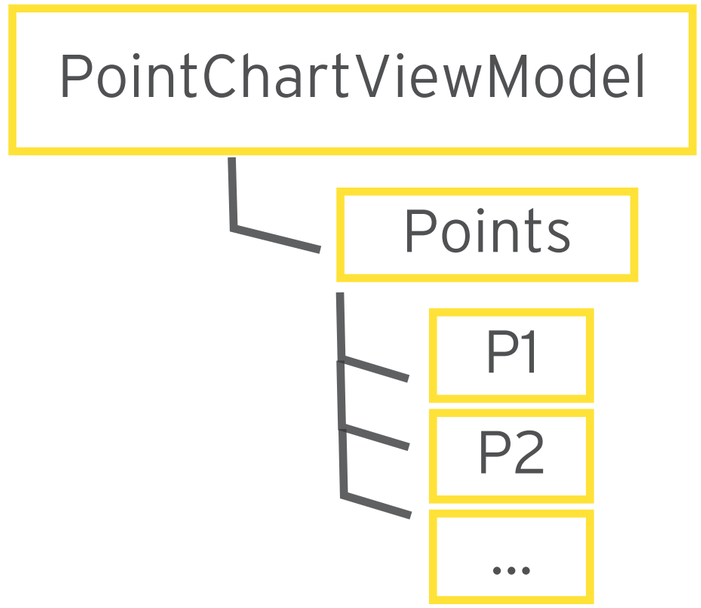
PointChartViewModel

- Points
  - P1
  - P2
  - ...

ListBox (PointChart)

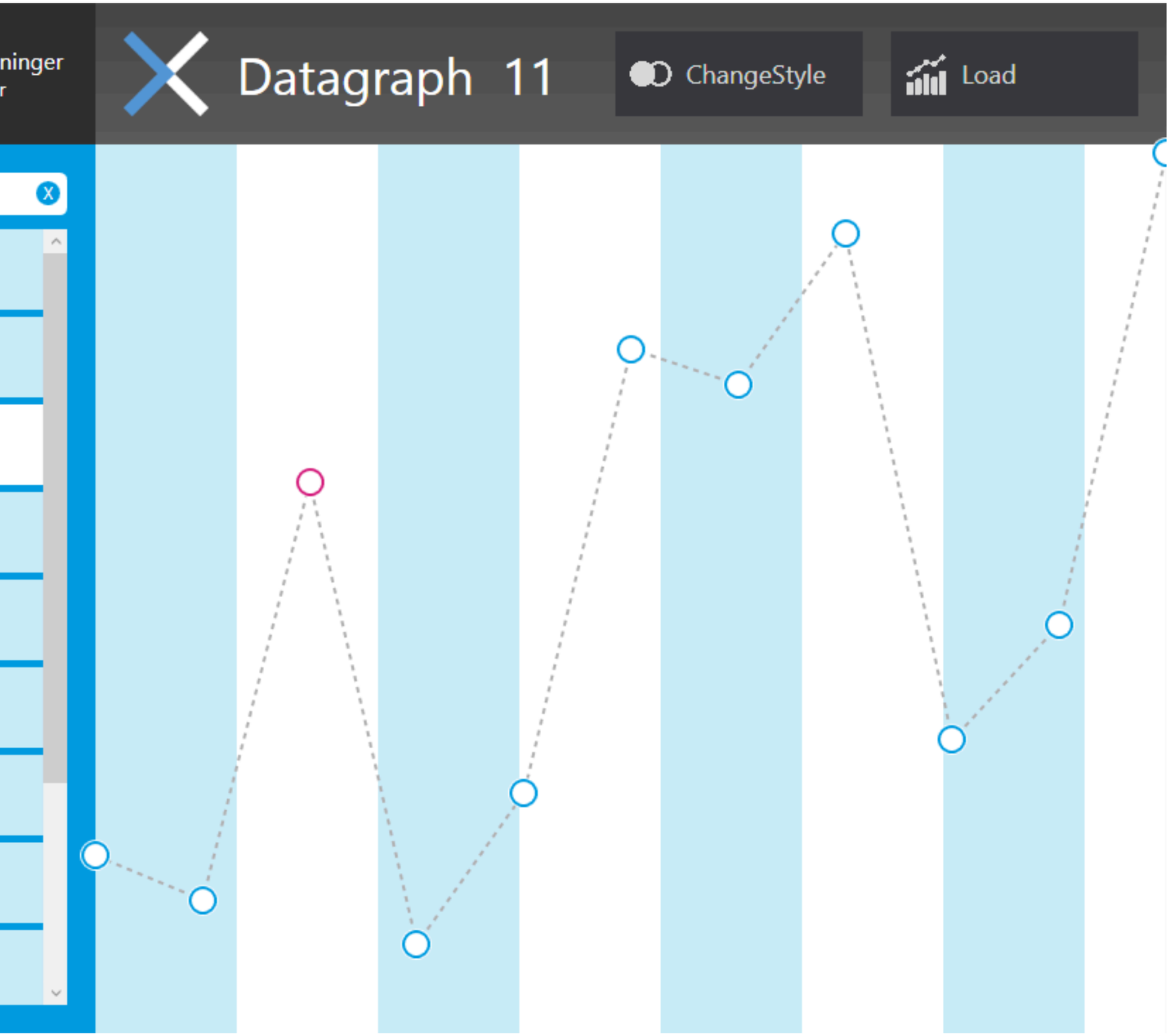
- X/YMinimum
- X/YMaximum
- LineThickness





ListBox (PointChart)

- X/YMinimum
- X/YMaximum
- LineThickness
- LineBrush



PointChartViewModel

Points

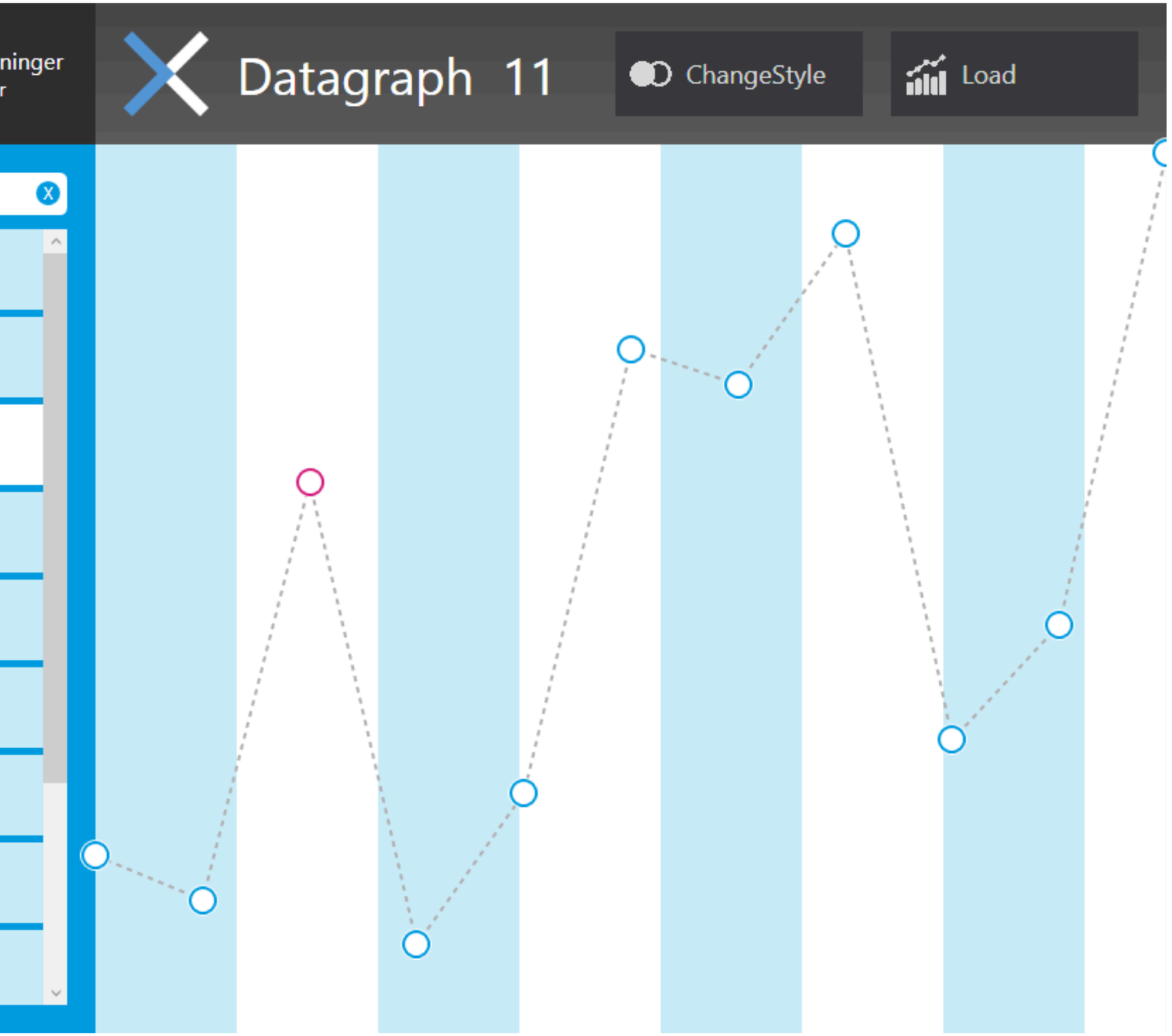
P1

P2

...

ListBox (PointChart)

- X/YMinimum
- X/YMaximum
- LineThickness
- LineBrush
- ...



PointChartViewModel

Points

P1

P2

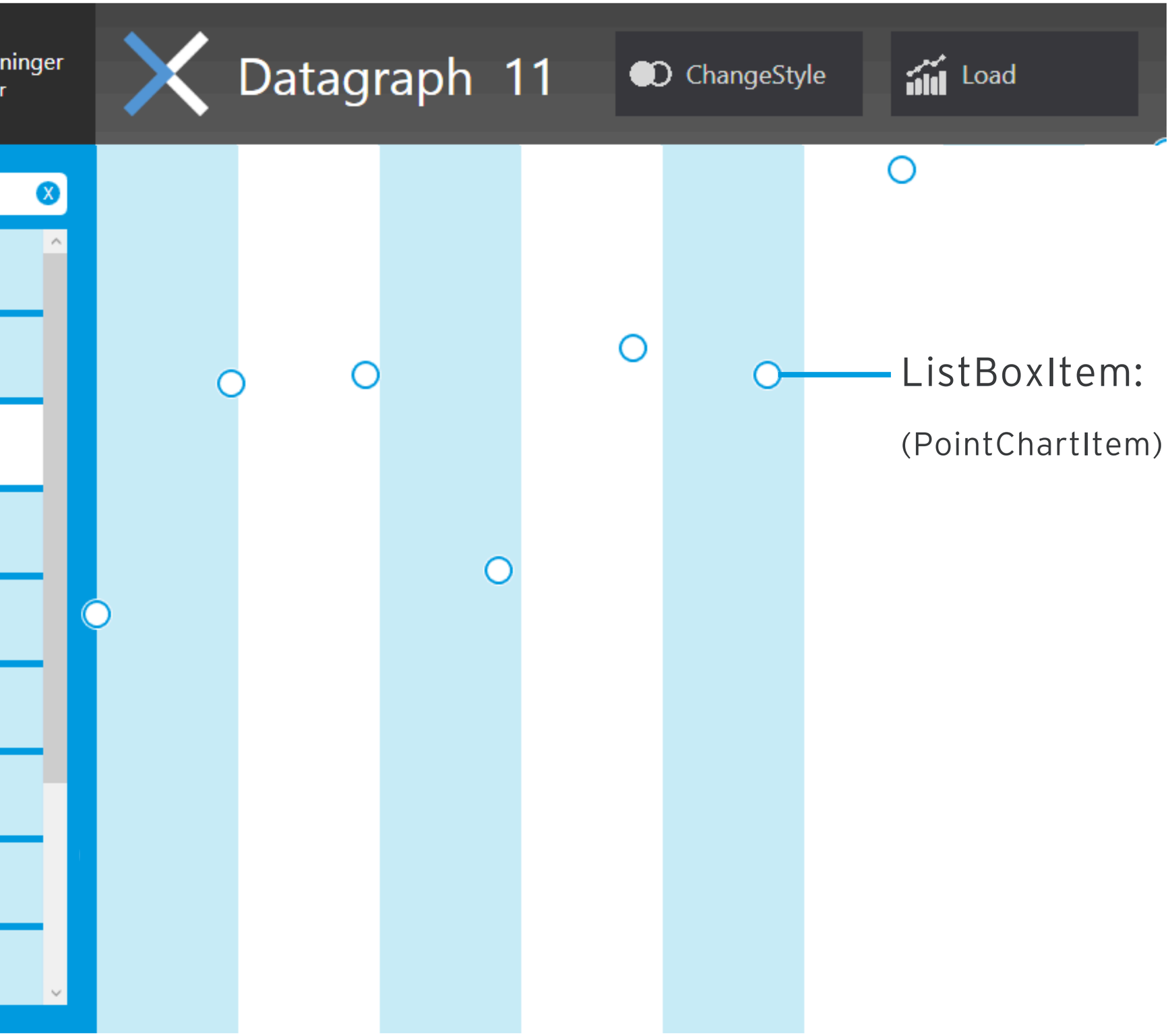
...

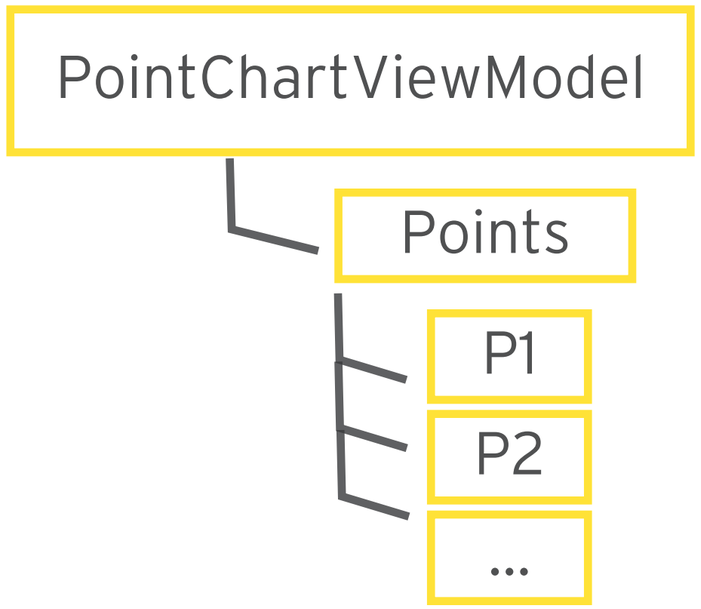
Property-Binding

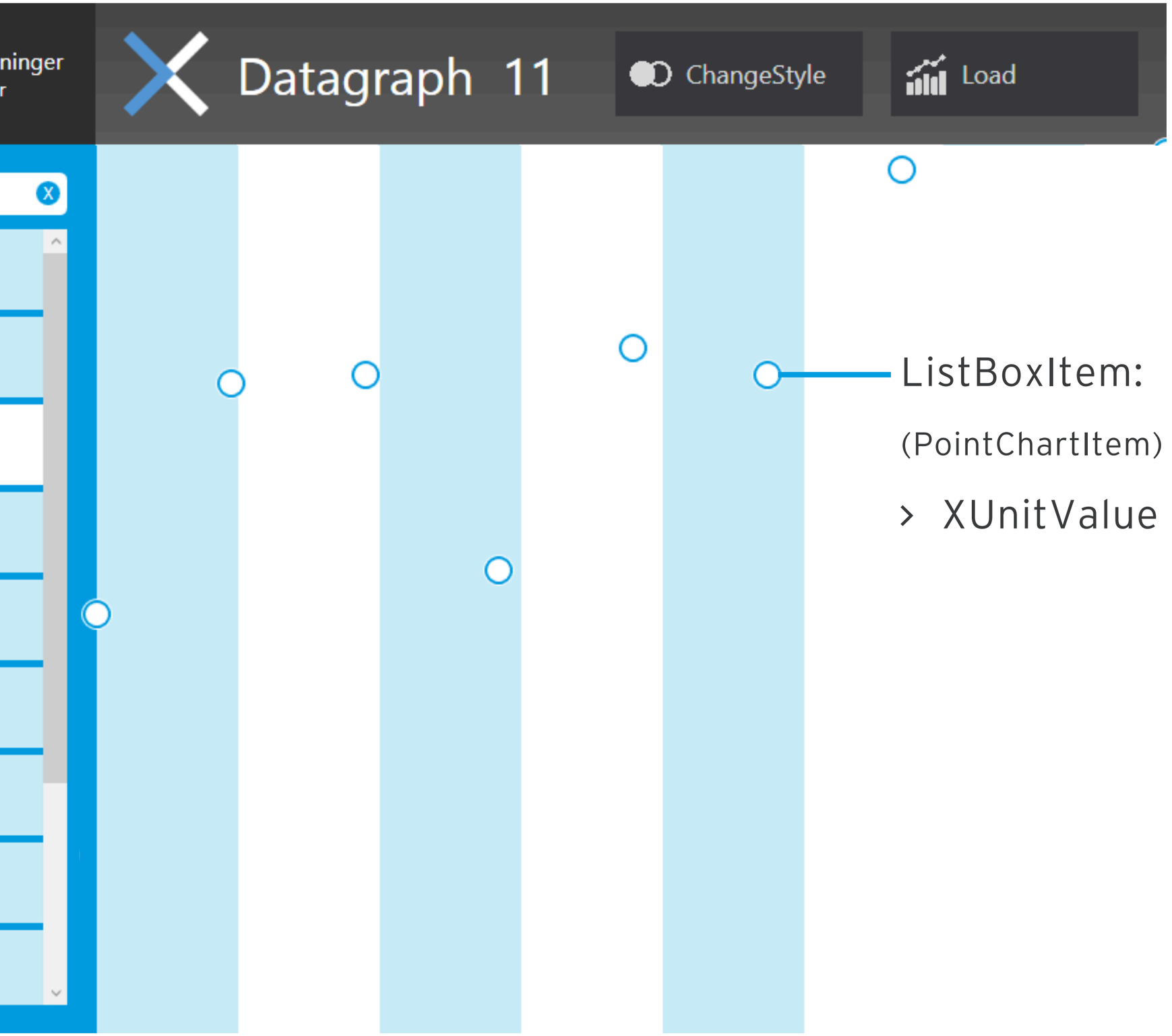
(ItemsSource)

ListBox (PointChart)

- X/YMinimum
- X/YMaximum
- LineThickness
- LineBrush
- ...


ninger
r
Datagraph 11
ChangeStyle
Load
ListBoxItem:
(PointChartItem)

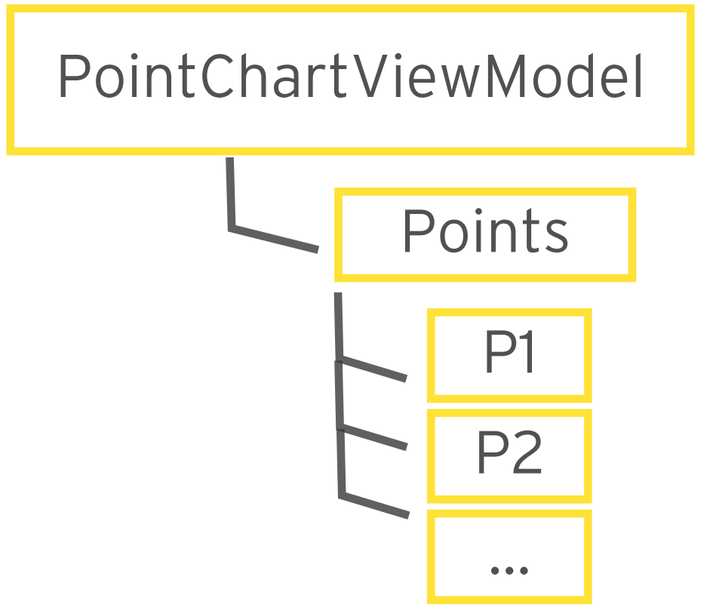

PointChartViewModel
Points
P1
P2
...

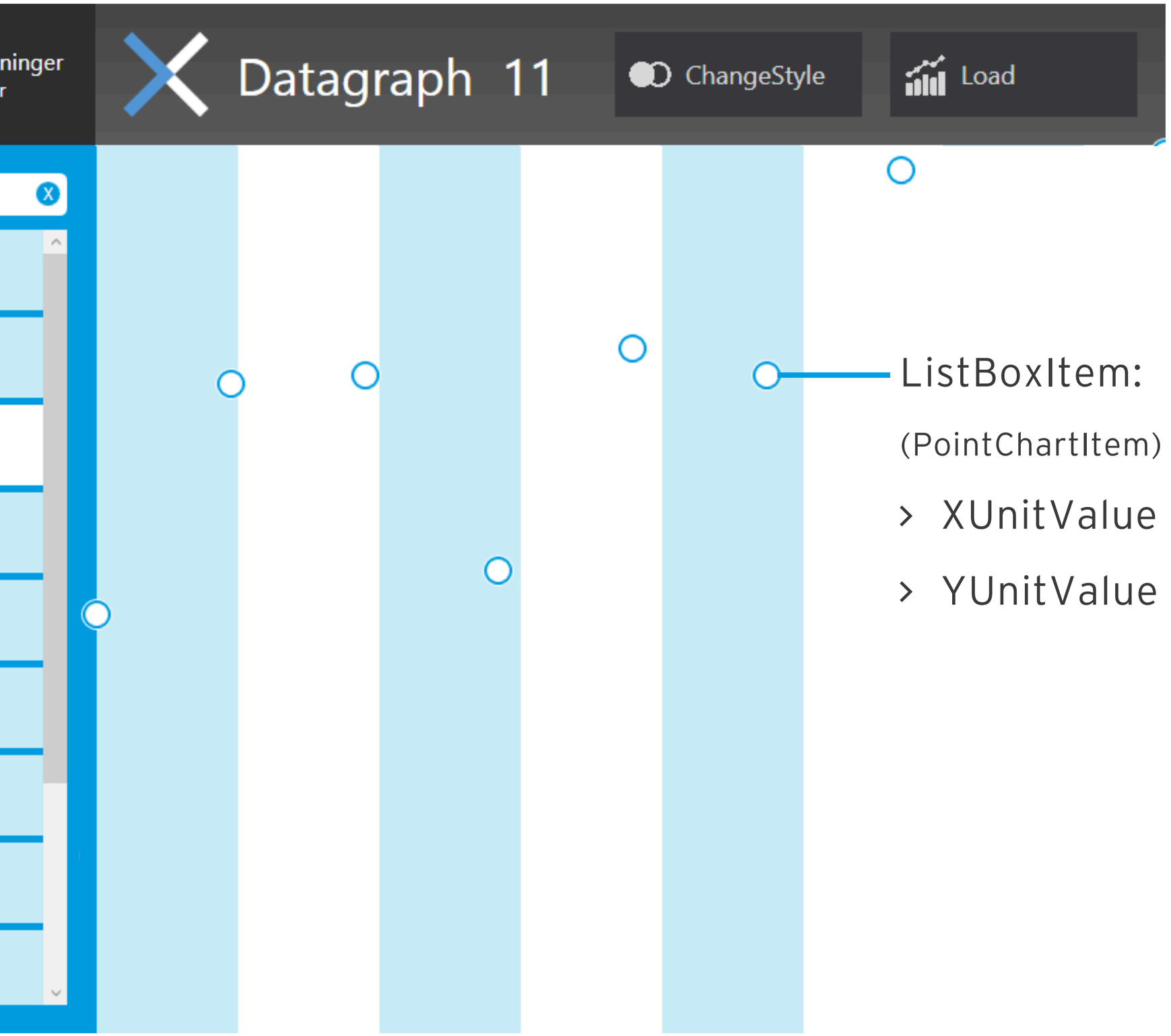


ListBoxItem:

(PointChartItem)

› XUnitValue



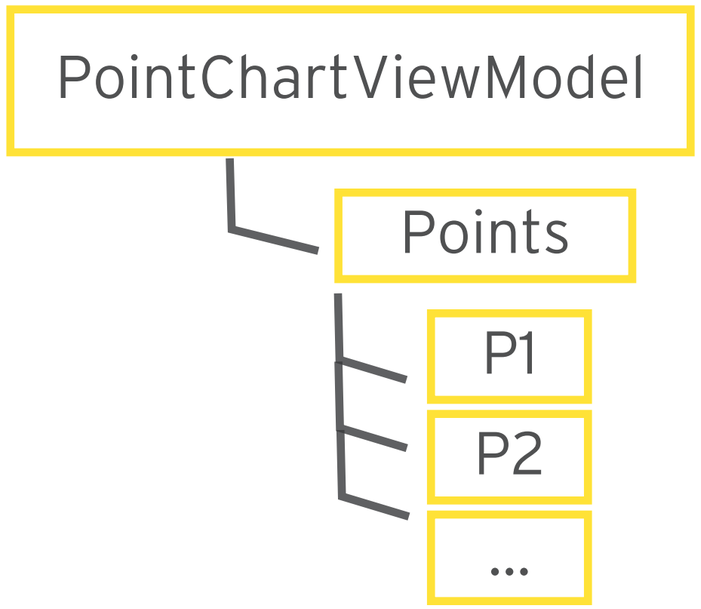


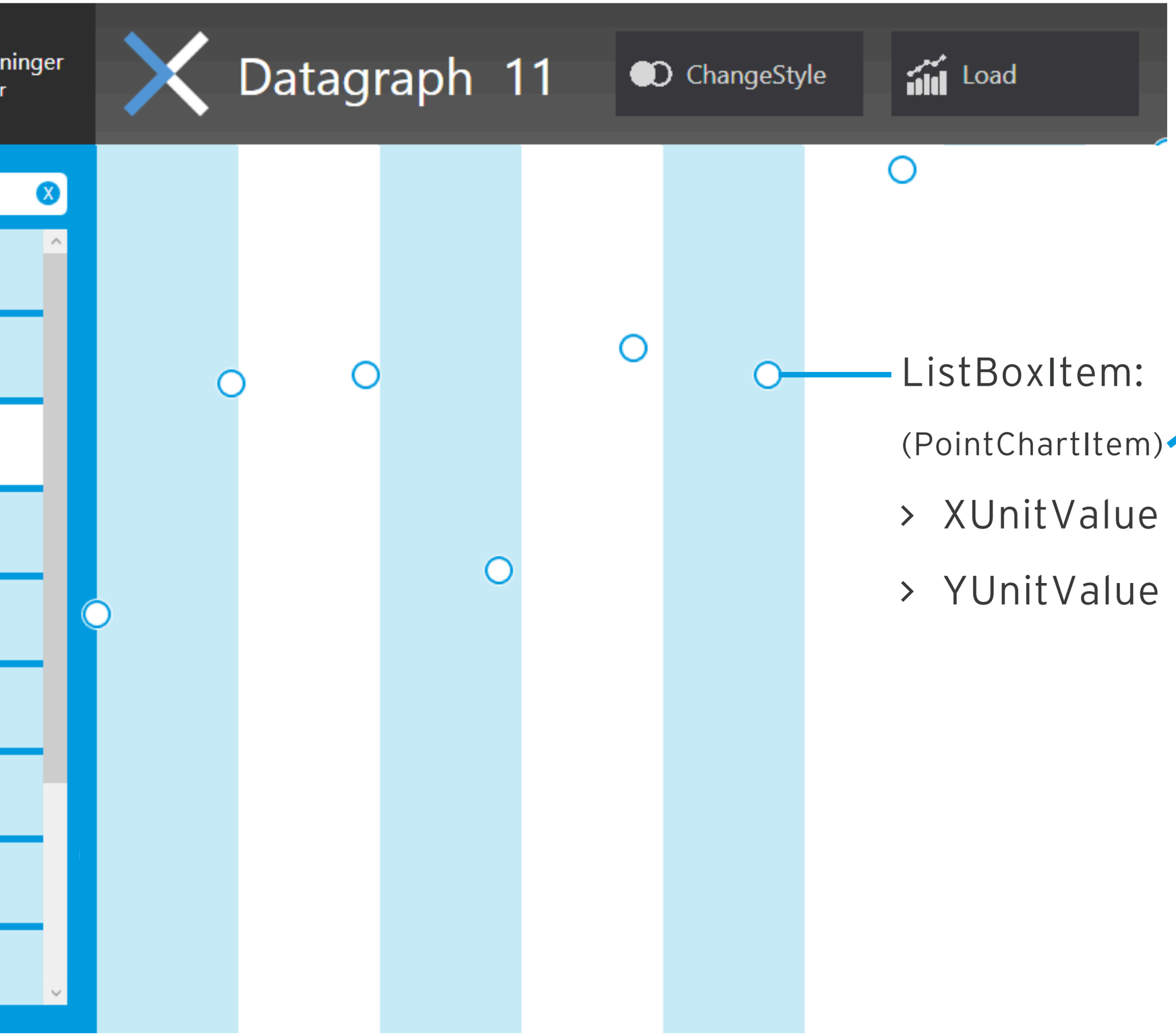
ListBoxItem:

(PointChartItem)

› XUnitValue

› YUnitValue





ListBoxItem:

(PointChartItem)

› XUnitValue

› YUnitValue

PointChartViewModel

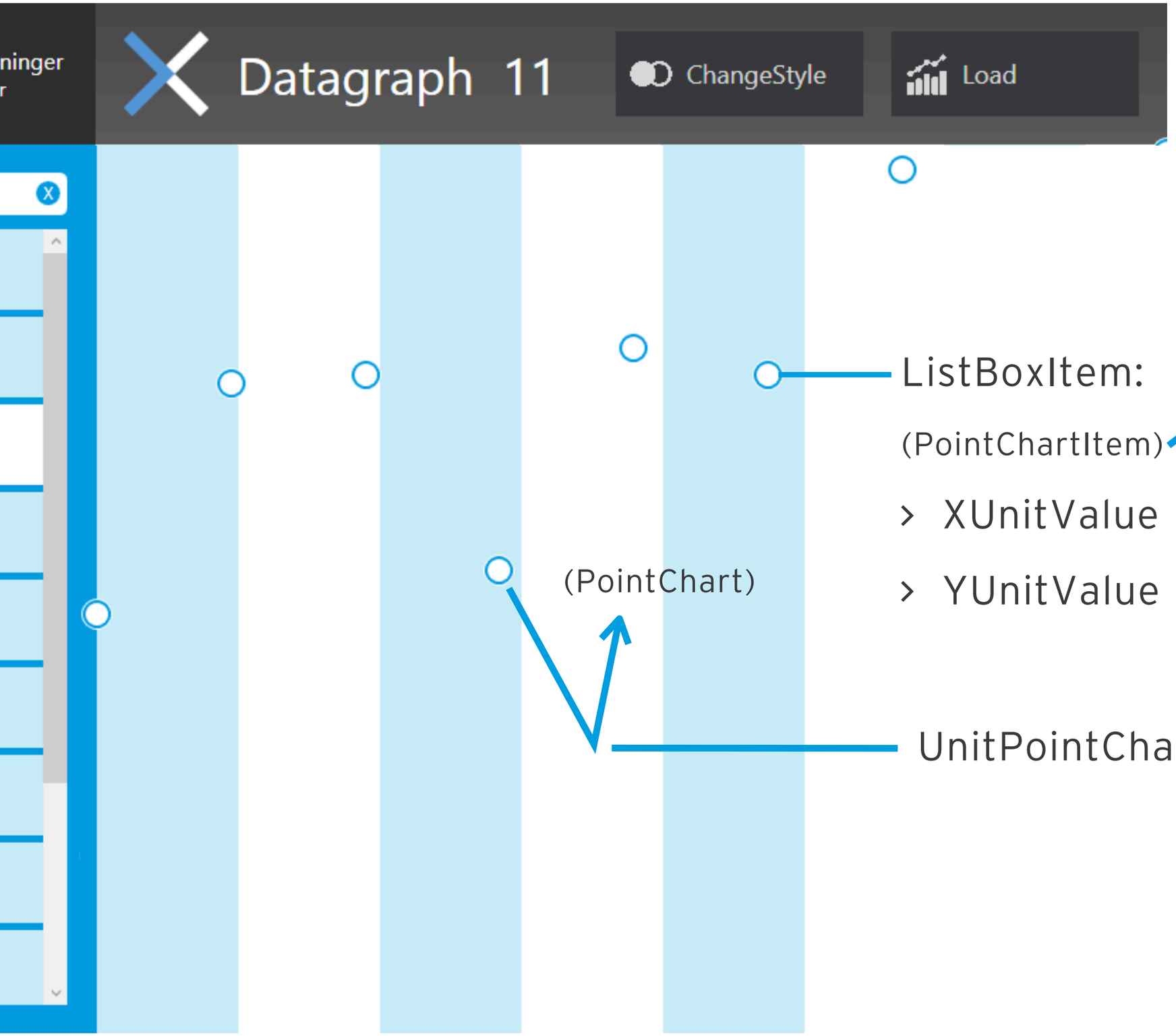
Points

P1

P2

...

Property Binding



PointChartViewModel

Points

P1

P2

...

ListBoxItem:

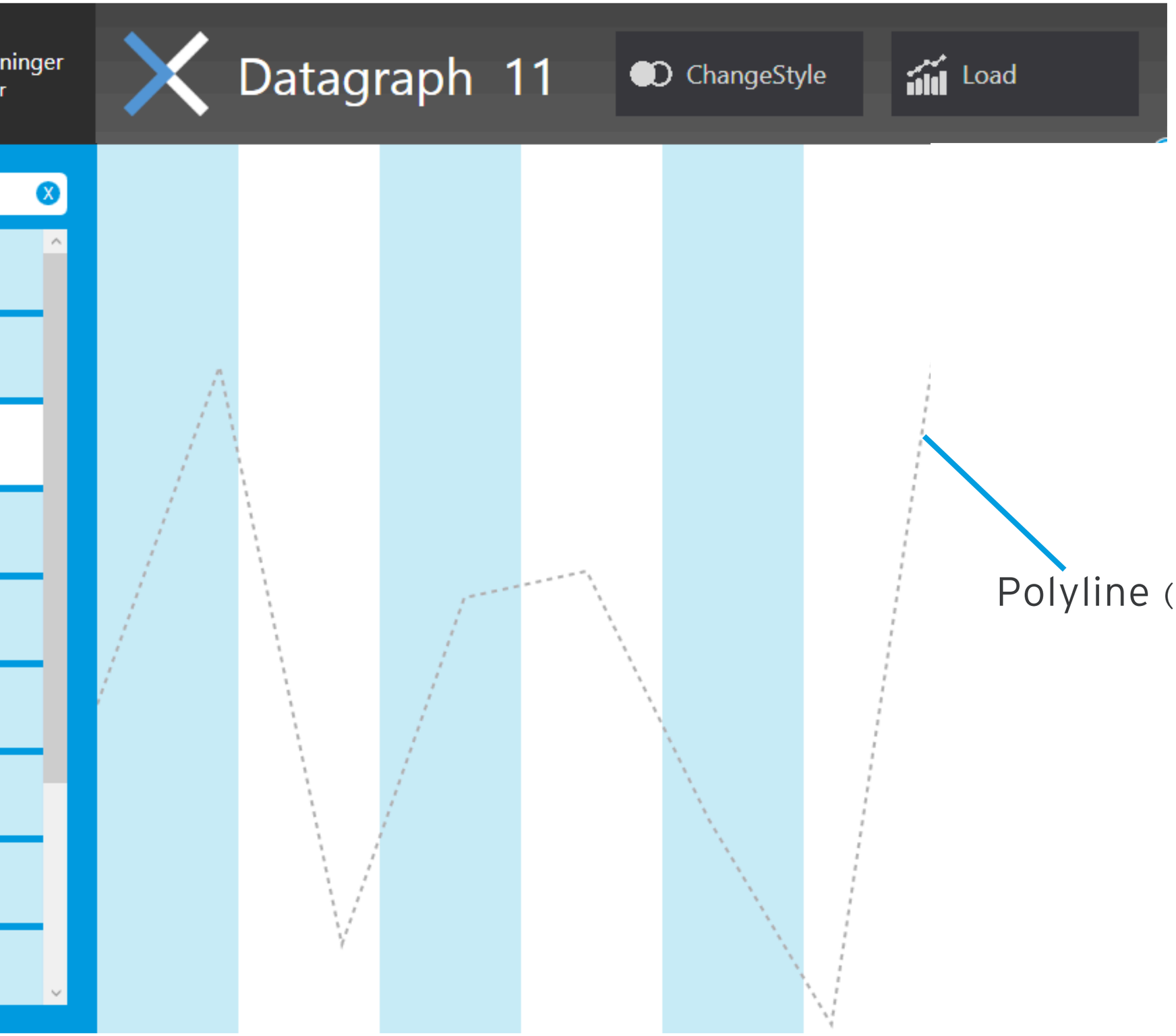
(PointChartItem)

› XUnitValue

› YUnitValue

Property Binding

UnitPointChanged-RoutedEvent



PointChartViewModel

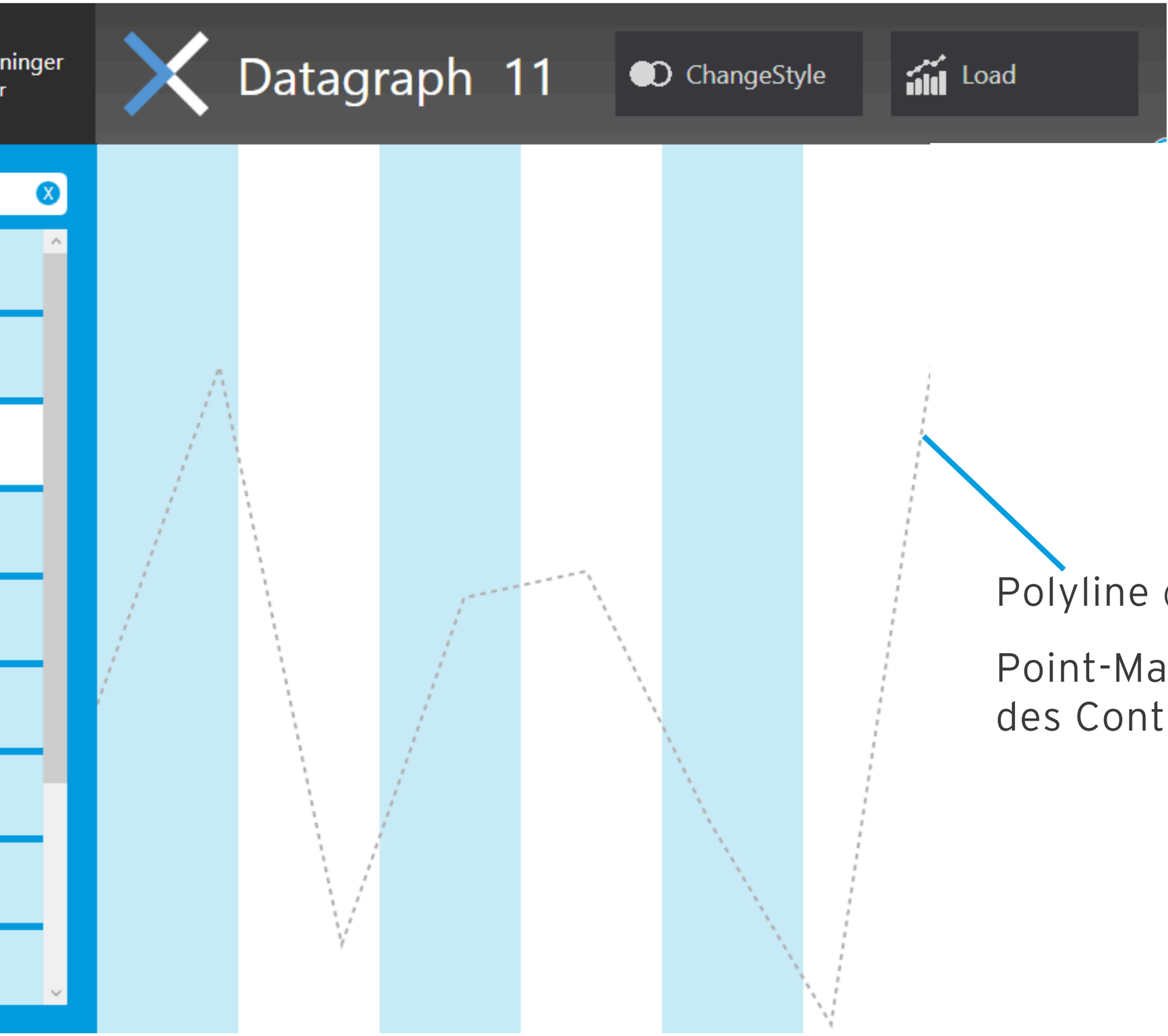
Points

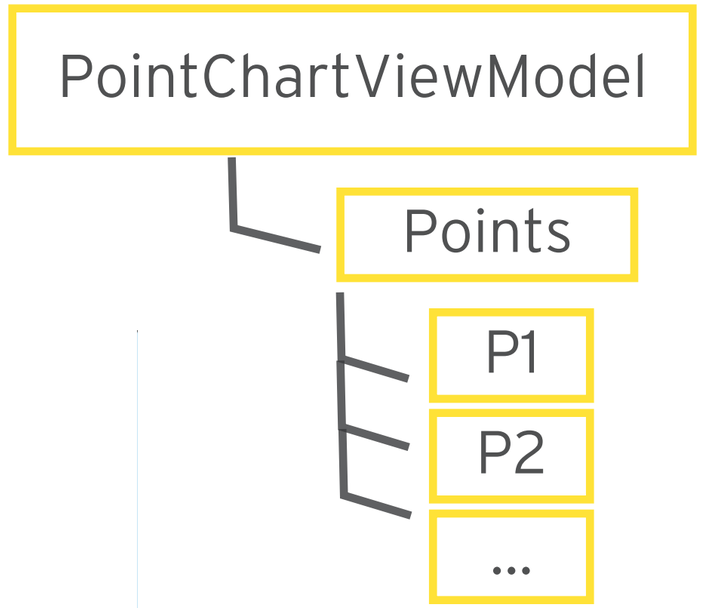
P1

P2

...

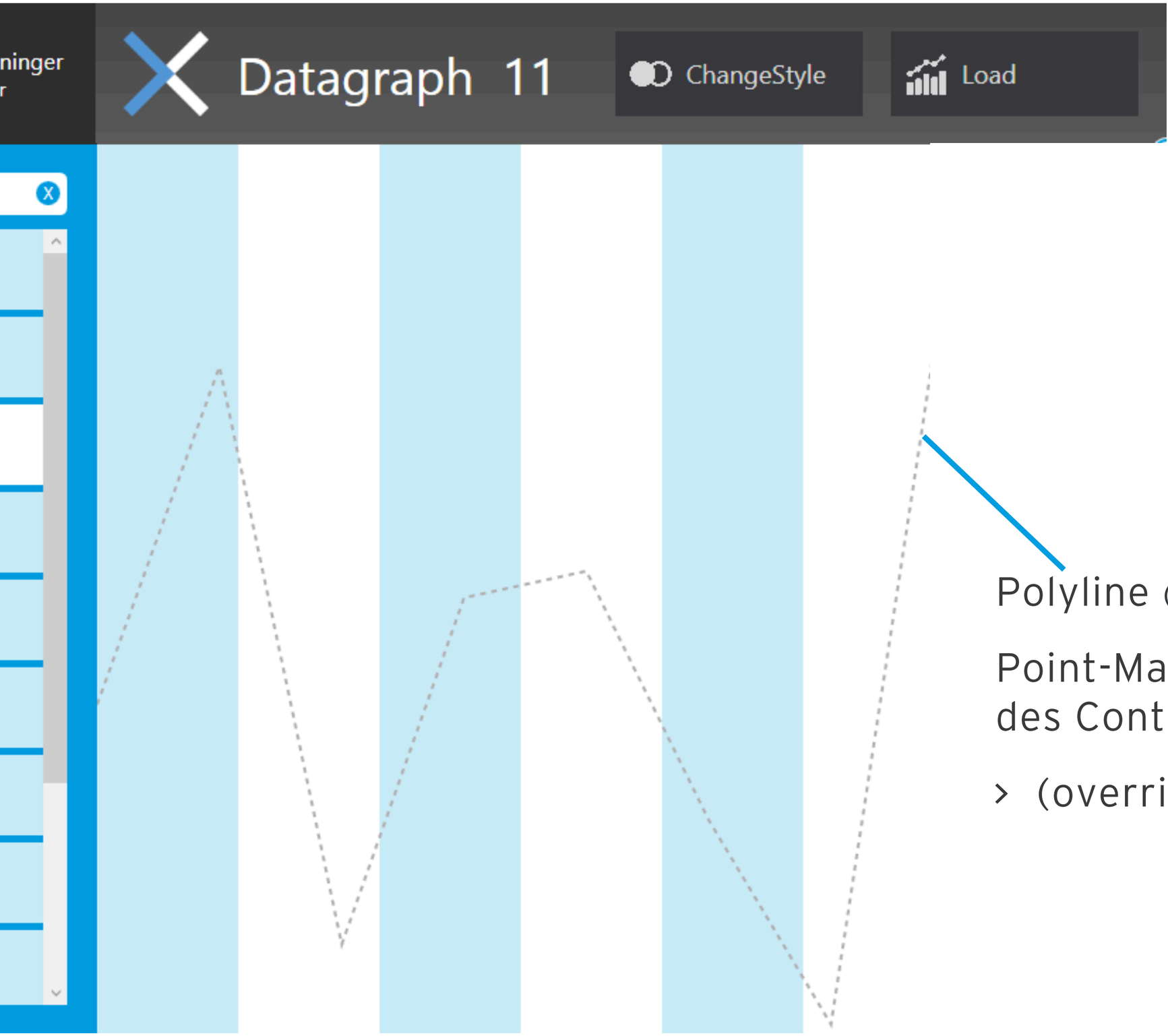
Polyline (PART_Line)

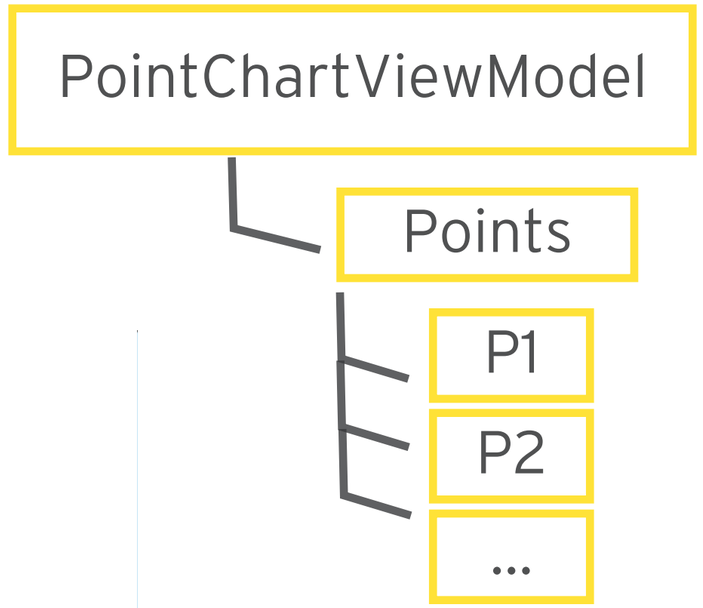




Polyline (PART_Line)

Point-Management innerhalb des Controls

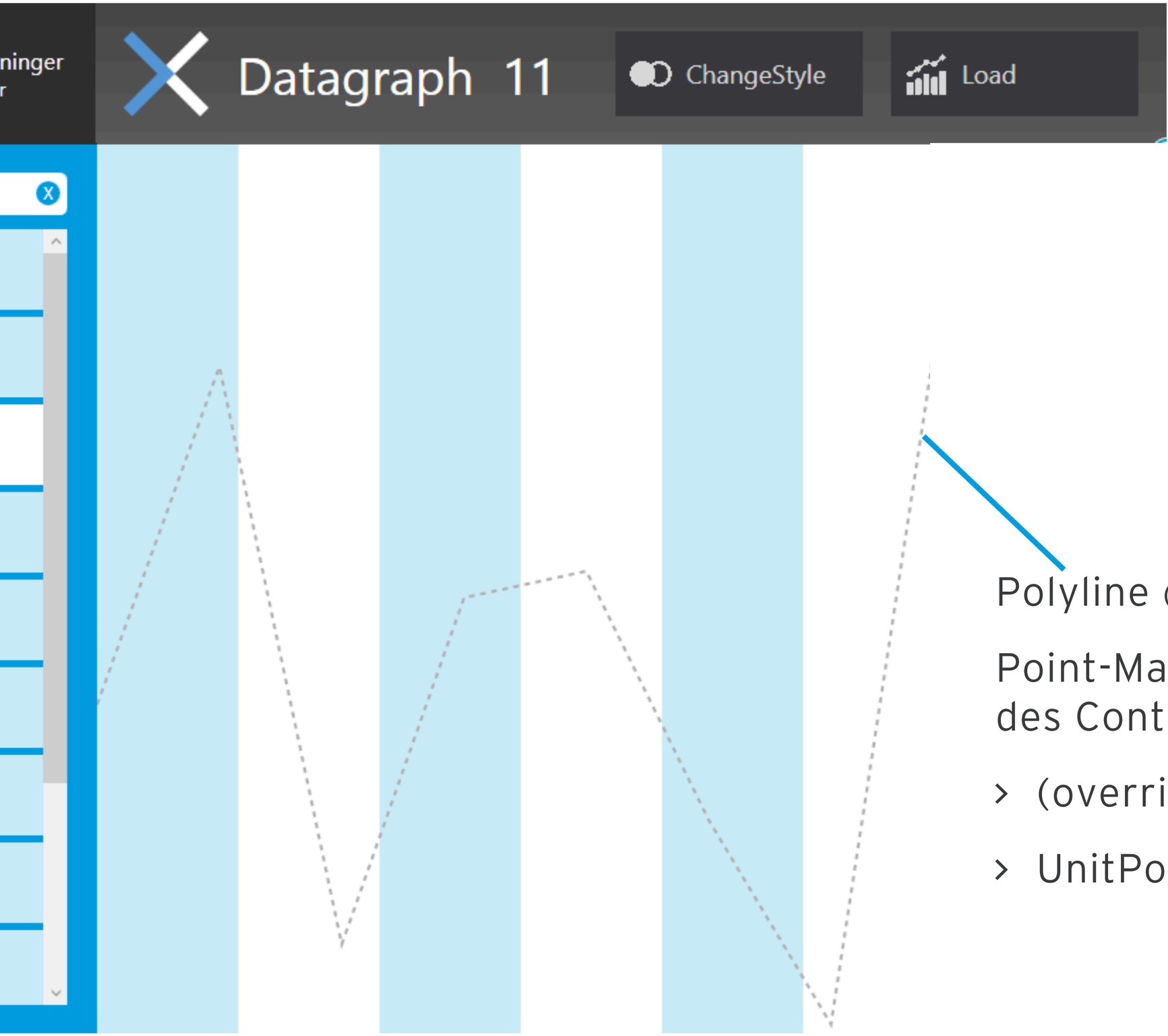


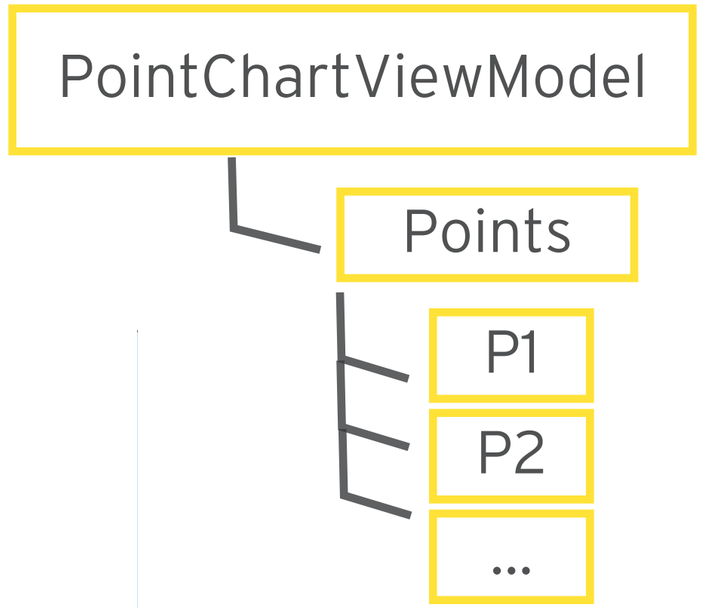


Polyline (PART_Line)

Point-Management innerhalb des Controls

› (override) ItemsChanged



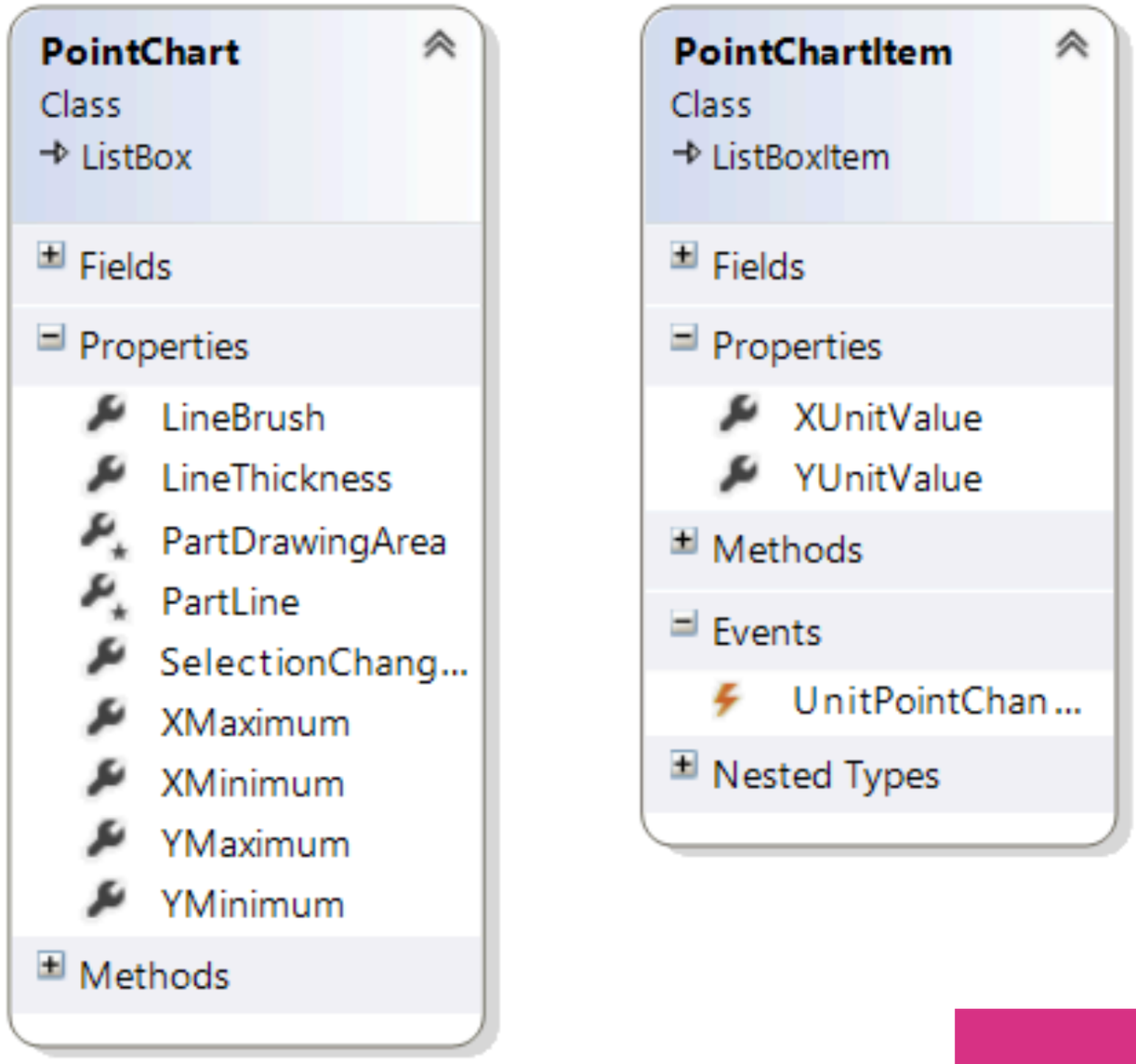


Polyline (PART_Line)

Point-Management innerhalb des Controls

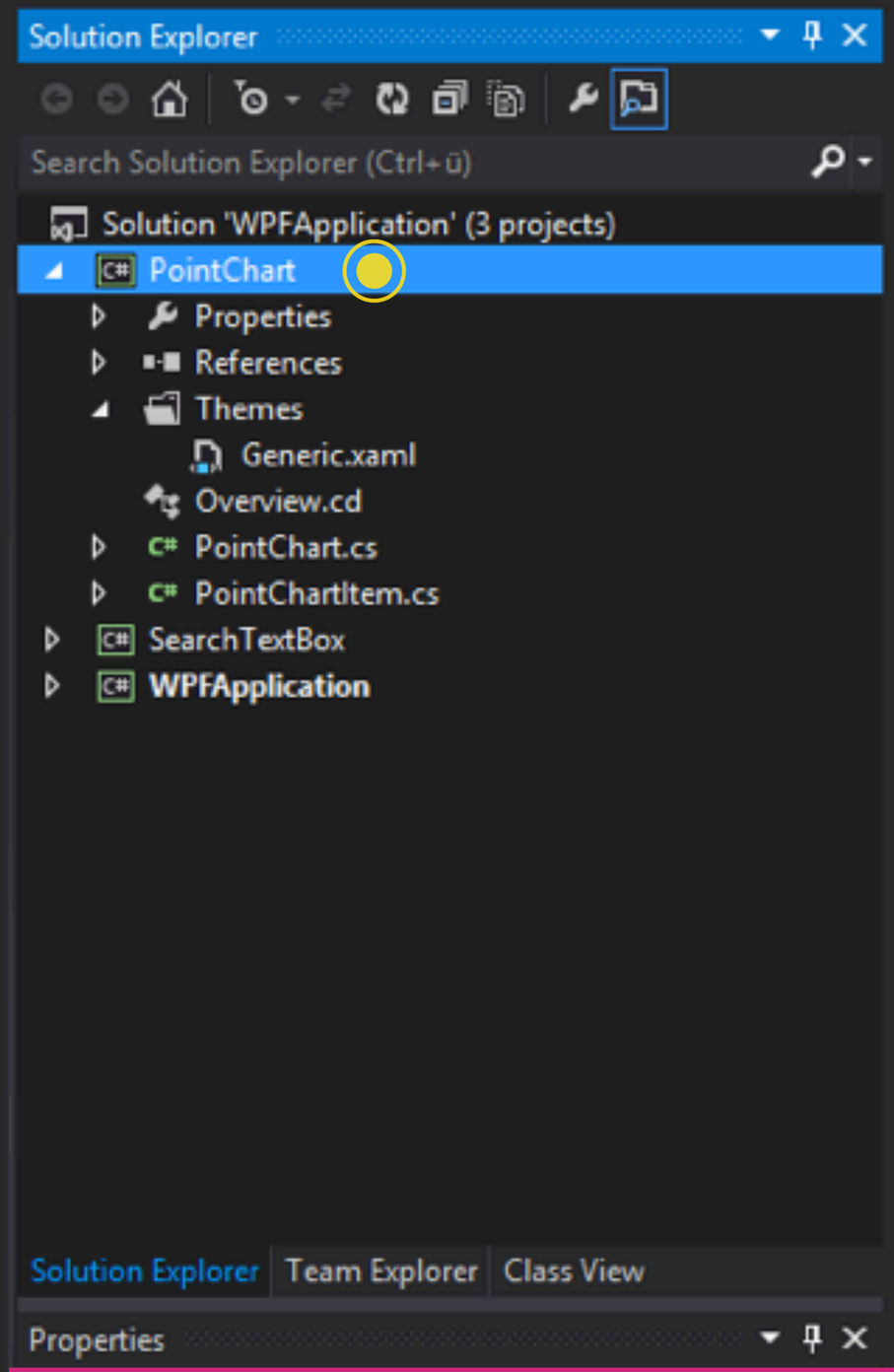
› (override) ItemsChanged

› UnitPointChanged


PointChart
Class
ListBox
Fields
Properties
LineBrush
LineThickness
PartDrawingArea
PartLine
SelectionChang...
XMaximum
XMinimum
YMaximum
YMinimum
Methods
PointChartItem
Class
ListBoxItem
Fields
Properties
XUnitValue
YUnitValue
Methods
Events
UnitPointChan...
Nested Types

That's it ;)

CUSTOM CONTROL LIBRARY



Custom Control Library erstellen

```
public class PointChart : ListBox
{
    Members

    DependencyProperties

    PropertyChangedCallbacks

    Events

    Handlers

    ClassHandlers

    Ctor

    Init/Cleanup

    OverrideMethods

    Methods

    Properties
}
```

Basisklasse bestimmen

(PointChart.cs)

```
public class PointChartItem : ListBoxItem
{
    Members

    DependencyProperties

    PropertyChangedCallbacks

    Events

    Handlers

    Ctor

    Init/Cleanup

    OverrideMethods

    Methods

    Properties
}
```

Basisklasse bestimmen

(PointChartItem.cs)

```
public class PointChartItem : ListBoxItem...
```

```
public class PointChart : ListBox...
```

Basisklasse bestimmen

(PointChart.cs)
(PointChartItem.cs)

Besonderheit ItemsControl

Besonderheit ItemsControl

- (Override) IsItemItsOwnContainerOverride

Besonderheit ItemsControl

- (Override) IsItemItsOwnContainerOverride
  - Ist das Item sein eigener Container?

Besonderheit ItemsControl

- (Override) IsItemItsOwnContainerOverride
  - Ist das Item sein eigener Container?
- (Override) GetContainerForItemOverride

Besonderheit ItemsControl

- (Override) IsItemItsOwnContainerOverride
  - Ist das Item sein eigener Container?
- (Override) GetContainerForItemOverride
  - Liefert das Element zum Anzeigen des Items

```
/// <summary> ...
protected override bool IsItemItsOwnContainerOverride(object item)
{
    return item is PointChartItem;
}

/// <summary> ...
protected override DependencyObject GetContainerForItemOverride()
{
    return new PointChartItem();
}
```

Besonderheit ItemsControl

(PointChart.cs)

STYLE & TEMPLATE

```
<Style TargetType="local:PointChart">
    <Setter Property="BorderThickness" Value="1 0 0 1" />
    <Setter Property="BorderBrush" Value="Gray" />
    <Setter Property="LineBrush" Value="Black" />
    <Setter Property="LineThickness" Value="1" />
    <Setter Property="Margin" Value="-10" />
    <Setter Property="Padding" Value="10" />
    <Setter Property="Focusable" Value="False" />
    <Setter Property="LineBrush" Value="Black" />
    <Setter Property="LineThickness" Value="1" />
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="local:PointChart">
                <Grid ClipToBounds="True">
                    <Border Margin="{TemplateBinding Padding}"
                            Background="{TemplateBinding Background}"
                            BorderBrush="{TemplateBinding BorderBrush}"
                            BorderThickness="{TemplateBinding BorderThickness}">
                        <Grid>
                            <Grid.LayoutTransform>
                                <ScaleTransform ScaleY="-1" />
                            </Grid.LayoutTransform>
                            <Polyline x:Name="PART_Line"
                                      Stroke="{TemplateBinding LineBrush}"
                                      StrokeThickness="{TemplateBinding LineThickness}" />
                            <Canvas x:Name="PART_DrawingArea"
                                    IsItemsHost="True" />
                        </Grid>
                    </Border>
                </Grid>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
```

Style & Template: PointChart

(Generic.xaml)

Style & Template: PointChartItem

(Generic.xaml)

```xaml
<Style TargetType="local:PointChartItem">
    <Setter Property="Width" Value="20" />
    <Setter Property="Height" Value="20" />
    <Setter Property="Background" Value="Transparent" />
    <Setter Property="BorderBrush" Value="Red" />
    <Setter Property="FocusVisualStyle" Value="{x:Null}" />
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="local:PointChartItem">
                <Ellipse Fill="{TemplateBinding Background}" Stroke="{TemplateBinding BorderBrush}" />
            </ControlTemplate>
        </Setter.Value>
    </Setter>
    <Style.Triggers>
        <Trigger Property="IsMouseOver" Value="True">
            <Setter Property="Background" Value="Black" />
        </Trigger>
        <Trigger Property="IsSelected" Value="True">
            <Setter Property="Background" Value="Blue" />
        </Trigger>
    </Style.Triggers>
</Style>
```

ONAPPLYTEMPLATE()

OnApplyTemplate()

OnApplyTemplate()

- Logic (C#, Klasse) und Visualisierung (Xaml, Template) getrennt

OnApplyTemplate()

- Logic (C#, Klasse) und Visualisierung (Xaml, Template) getrennt
- Elemente innerhalb Control Template werden nach Konvention mit *PART_...* benannt

OnApplyTemplate()

- Logic (C#, Klasse) und Visualisierung (Xaml, Template) getrennt
- Elemente innerhalb Control Template werden nach Konvention mit *PART_...* benannt
- Element-Referenzierung über *OnApplyTemplate()* Methode *GetTemplateChild()* Methode mit *PART_...* aufrufen

OnApplyTemplate()

- Logic (C#, Klasse) und Visualisierung (Xaml, Template) getrennt
- Elemente innerhalb Control Template werden nach Konvention mit *PART_…* benannt
- Element-Referenzierung über *OnApplyTemplate()* Methode *GetTemplateChild()* Methode mit *PART_…* aufrufen
- **Achtung:** Element müssen nicht in der Template vorhanden sein

Element-Referenzierung 1/2

(Generic.xaml)

```
<Style TargetType="local:PointChart">
    <Setter Property="BorderThickness" Value="1 0 0 1" />
    <Setter Property="BorderBrush" Value="Gray" />
    <Setter Property="LineBrush" Value="Black" />
    <Setter Property="LineThickness" Value="1" />
    <Setter Property="Margin" Value="-10" />
    <Setter Property="Padding" Value="10" />
    <Setter Property="Focusable" Value="False" />
    <Setter Property="LineBrush" Value="Black" />
    <Setter Property="LineThickness" Value="1" />
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="local:PointChart">
                <Grid ClipToBounds="True">
                    <Border Margin="{TemplateBinding Padding}"
                            Background="{TemplateBinding Background}"
                            BorderBrush="{TemplateBinding BorderBrush}"
                            BorderThickness="{TemplateBinding BorderThickness}">
                        <Grid>
                            <Grid.LayoutTransform>
                                <ScaleTransform ScaleY="-1" />
                            </Grid.LayoutTransform>
                            <Polyline x:Name="PART_Line"
                                      Stroke="{TemplateBinding LineBrush}"
                                      StrokeThickness="{TemplateBinding LineThickness}" />
                            <Canvas x:Name="PART_DrawingArea"
                                    Width="{Binding ActualWidth,
                                                    ElementName=PART_DrawingArea}"
                                    Height="{Binding ActualHeight,
                                                     ElementName=PART_DrawingArea}"
                                    IsItemsHost="True" />
                        </Grid>
                    </Border>
                </Grid>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
```

- Diese beiden Elemente sollen referenziert werden

Element-Referenzierung 2/2

(PointChart.cs)

```
/// <summary>
/// Get Template PART_-Elements
/// </summary>
public override void OnApplyTemplate()
{
    base.OnApplyTemplate();

    PartDrawingArea = GetTemplateChild("PART_DrawingArea") as FrameworkElement;
    PartLine = GetTemplateChild("PART_Line") as Polyline;

    Init();
}
```

> Referenz kann null sein!

OnApplyTemplate() - Tipps 1/2

OnApplyTemplate() - Tipps 1/2

- Nur verwenden wenn Template Binding, RoutedCommands oder RoutedEvents nicht zu dem gewünschten Ergebnis führen

OnApplyTemplate() - Tipps 1/2

- Nur verwenden wenn Template Binding, RoutedCommands oder RoutedEvents nicht zu dem gewünschten Ergebnis führen
- NullReferenceException vermeiden und defensiv programmieren

OnApplyTemplate() - Tipps 1/2

- Nur verwenden wenn Template Binding, RoutedCommands oder RoutedEvents nicht zu dem gewünschten Ergebnis führen
- NullReferenceException vermeiden und defensiv programmieren
- *OnApplyTemplate*() Methode kann mehrfach aufgerufen werden, mit internen Properties arbeiten (Mehr Kontrolle)

OnApplyTemplate() - Tipps 1/2

- Nur verwenden wenn Template Binding, RoutedCommands oder RoutedEvents nicht zu dem gewünschten Ergebnis führen
- NullReferenceException vermeiden und defensiv programmieren
- *OnApplyTemplate*() Methode kann mehrfach aufgerufen werden, mit internen Properties arbeiten (Mehr Kontrolle)
- *OnApplyTemplate()* Methode kann manuell aufgerufen werden. Dadurch wird Control Template geladen

Element-Referenzierung Tips

(PointChart.cs)

```
/// <summary> ...
protected FrameworkElement PartDrawingArea
{
    get
    {
        return partDrawingArea;
    }

    private set
    {
        if (value == partDrawingArea)
            return;

        if(partDrawingArea != null)
            partDrawingArea.SizeChanged -= partDrawingArea_SizeChanged;

        partDrawingArea = value;

        if (partDrawingArea != null)
            partDrawingArea.SizeChanged += partDrawingArea_SizeChanged;
    }
}
```

OnApplyTemplate() - Tipps 2/2

OnApplyTemplate() - Tipps 2/2

- PropertyChangedCallbacks werden noch vor OnApplyTemplate aufgerufen

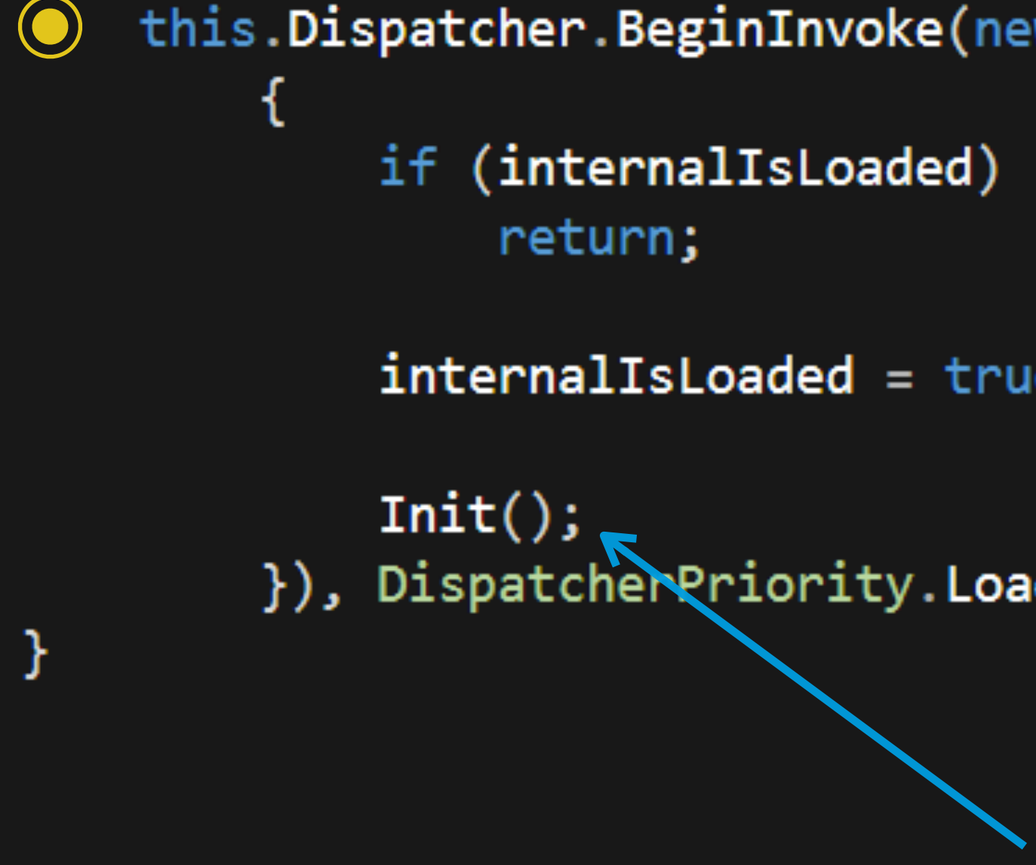
OnApplyTemplate() - Tipps 2/2

- PropertyChangedCallbacks werden noch vor OnApplyTemplate aufgerufen
- Interne Variable *internalIsLoaded* um Update-Logik erst durchzuführen wenn Template angewandt wurde (siehe Dependency Properties - Extended)

Element-Referenzierung Tips

(PointChart.cs)

```
/// <summary> ...
public PointChart()
{
    //this.Loaded += PointChartLoaded;

    this.Dispatcher.BeginInvoke(new Action(() =>
        {
            if (internalIsLoaded)
                return;

            internalIsLoaded = true;

            Init();
        }), DispatcherPriority.Loaded);
}
```



› Initialisierung ;)

DEPENDENCY PROPERTIES EXTENDED

```
/// <summary> ...
public static readonly DependencyProperty LineBrushProperty = DependencyProperty.Register(
    "LineBrush",
    typeof(Brush),
    typeof(PointChart),
    new FrameworkPropertyMetadata(Brushes.Black));

/// <summary> ...
public static readonly DependencyProperty LineThicknessProperty = DependencyProperty.Register(
    "LineThickness",
    typeof(double),
    typeof(PointChart),
    new FrameworkPropertyMetadata(1.0));

/// <summary> ...
public static readonly DependencyProperty XMinimumProperty = DependencyProperty.Register(
    "XMinimum",
    typeof(double),
    typeof(PointChart),
    new FrameworkPropertyMetadata(0.0, RangeChanged));

/// <summary> ...
public static readonly DependencyProperty XMaximumProperty = DependencyProperty.Register(
    "XMaximum",
    typeof(double),
    typeof(PointChart),
    new FrameworkPropertyMetadata(100.0, RangeChanged));

/// <summary> ...
public static readonly DependencyProperty YMinimumProperty = DependencyProperty.Register(
    "YMinimum",
    typeof(double),
    typeof(PointChart),
    new FrameworkPropertyMetadata(0.0, RangeChanged));

/// <summary> ...
public static readonly DependencyProperty YMaximumProperty = DependencyProperty.Register(
    "YMaximum",
    typeof(double),
    typeof(PointChart),
    new FrameworkPropertyMetadata(100.0, RangeChanged));

/// <summary> ...
public static readonly DependencyProperty SelectionChangedCommandProperty = DependencyProperty.Register(
    "SelectionChangedCommand",
    typeof(ICommand),
    typeof(PointChart),
    new FrameworkPropertyMetadata(null));
```

› Was ist das?

Properties anlegen 1/2

(PointChart.cs)

```
/// <summary> ...
public static readonly DependencyProperty XUnitValueProperty = DependencyProperty.Register(
    "XUnitValue",
    typeof(double),
    typeof(PointChartItem),
    new FrameworkPropertyMetadata(double.NaN, PointPropertyChanged));

/// <summary> ...
public static readonly DependencyProperty YUnitValueProperty = DependencyProperty.Register(
    "YUnitValue",
    typeof(double),
    typeof(PointChartItem),
    new FrameworkPropertyMetadata(double.NaN, PointPropertyChanged));
```

› Schon wieder?

Properties anlegen 2/2

(PointChartItem.cs)

Dependency Properties - Extended 1/2

Dependency Properties - Extended 1/2

› Reagieren auf Änderungen über Changed Callback möglich

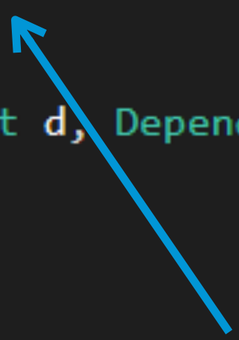
Dependency Properties - Extended 1/2

- Reagieren auf Änderungen über Changed Callback möglich
- Angabe einer statischen Funktion. Über das Metadatenobjekt innerhalb der Funktion erhält man Zugriff auf die Instanz des Controls sowie Informationen über Änderungen (*ChangedEventArgs*)

Dependency Properties - Extended 1/2

- Reagieren auf Änderungen über Changed Callback möglich
- Angabe einer statischen Funktion. Über das Metadatenobjekt innerhalb der Funktion erhält man Zugriff auf die Instanz des Controls sowie Informationen über Änderungen (*ChangedEventArgs*)
- Statische Methoden möglichst kurz halten und auf der Instanz eine Methode aufrufen

```
/// <summary> ...
public static readonly DependencyProperty XMinimumProperty = DependencyProperty.Register(
    "XMinimum",
    typeof(double),
    typeof(PointChart),
    new FrameworkPropertyMetadata(0.0, RangeChanged));

/// <summary> ...
private static void RangeChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)
{
    PointChart self = d as PointChart;

    if (!self.internalIsLoaded)
        return;

    self.UpdateAllPoints();
}
```



- Angabe Callback

- Callback mit ChangedEventArgs

Properties Changed Callback

(PointChart.cs)

Dependency Properties - Extended 2/2

Dependency Properties - Extended 2/2

- Doch was wenn bereits existierende Properties Elemente beeinflussen?

Dependency Properties - Extended 2/2

› Doch was wenn bereits existierende Properties Elemente beeinflussen?

› Hier:

Dependency Properties - Extended 2/2

- Doch was wenn bereits existierende Properties Elemente beeinflussen?

- Hier:
  - Width, Height

Dependency Properties - Extended 2/2

- Doch was wenn bereits existierende Properties Elemente beeinflussen?

- Hier:
  - Width, Height
  - Items

Dependency Properties - Extended 2/2

- Doch was wenn bereits existierende Properties Elemente beeinflussen?

- Hier:
  - Width, Height
  - Items

- **Lösung:** Überschreiben bereits vorhandener Methoden oder Properties

```
/// <summary> ...
protected override void OnItemsChanged(System.Collections.Specialized.NotifyCollectionChangedEventArgs e)
{
    base.OnItemsChanged(e);

    if(PartLine == null)
        return;

    switch (e.Action)
    {
        case NotifyCollectionChangedAction.Remove:
            PartLine.Points.RemoveAt(e.OldStartingIndex);
            break;
        case NotifyCollectionChangedAction.Add:
            PartLine.Points.Insert(e.NewStartingIndex, new Point());
            break;
        case NotifyCollectionChangedAction.Reset:
            PartLine.Points.Clear();
            for (int i = 0; i < Items.Count; i++)
                PartLine.Points.Add(new Point());
            break;
    }
}

/// <summary> ...
protected override void OnRenderSizeChanged(SizeChangedInfo sizeInfo)
{
    base.OnRenderSizeChanged(sizeInfo);

    if (!internalIsLoaded)
        return;

    UpdateAllPoints();
}
```

Override Methods

(PointChart.cs)

```
/// <summary> ...
static PointChart()
{
    DefaultStyleKeyProperty.OverrideMetadata(
        typeof(PointChart),
        new FrameworkPropertyMetadata(typeof(PointChart)));

    EventManager.RegisterClassHandler(
        typeof(PointChart),
        PointChartItem.UnitPointChangedEvent,
        new RoutedEventHandler(OnItemUnitPointChangedEvent));

    WidthProperty.OverrideMetadata(
        typeof(PointChart),
        new FrameworkPropertyMetadata(double.NaN, RangeChanged));
}

/// <summary> ...
private static void RangeChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)
{
    PointChart self = d as PointChart;

    if (!self.internalIsLoaded)
        return;

    self.UpdateAllPoints();
}
```

Override Properties

(PointChart.cs)

ROUTED EVENTS

RoutedEvents

RoutedEvents

- RoutedEvents können je nach Strategy im visuellen Baum hoch oder hinunter navigieren.

RoutedEvents

- RoutedEvents können je nach Strategy im visuellen Baum hoch oder hinunter navigieren.
- Somit können Sie mehrere Handler auf verschiedenen Elementen auslösen

RoutedEvents

- RoutedEvents können je nach Strategy im visuellen Baum hoch oder hinunter navigieren.
- Somit können Sie mehrere Handler auf verschiedenen Elementen auslösen
- Strategien:

RoutedEvents

- RoutedEvents können je nach Strategy im visuellen Baum hoch oder hinunter navigieren.
- Somit können Sie mehrere Handler auf verschiedenen Elementen auslösen
- Strategien:
  - Tunneling = Down

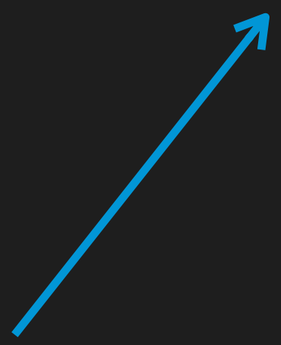
RoutedEvents

- RoutedEvents können je nach Strategy im visuellen Baum hoch oder hinunter navigieren.
- Somit können Sie mehrere Handler auf verschiedenen Elementen auslösen
- Strategien:
  - Tunneling = Down
  - Bubbling = Up

RoutedEvents

- RoutedEvents können je nach Strategy im visuellen Baum hoch oder hinunter navigieren.
- Somit können Sie mehrere Handler auf verschiedenen Elementen auslösen
- Strategien:
  - Tunneling = Down
  - Bubbling = Up
  - Direct = Direct (Vergleiche standard .NET Events)

```
/// <summary> ...
public static readonly RoutedEvent UnitPointChangedEvent = EventManager.RegisterRoutedEvent(
    "UnitPointChanged",
    RoutingStrategy.Bubble,
    typeof(UnitPointChangedEventHandler),
    typeof(PointChartItem));

/// <summary> ...
public delegate void UnitPointChangedEventHandler(object sender, RoutedEventArgs args);

/// <summary> ...
[Description("This event occurs when the x or y unit value has changed.")]
public event UnitPointChangedEventHandler UnitPointChanged...
```

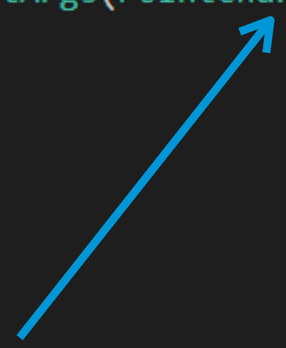


- Anlegen des Events in der auslösenden Klasse

Routed Events 1/3

(PointChartItem.cs)

```
/// <summary> ...
public static readonly DependencyProperty YUnitValueProperty = DependencyProperty.Register(
    "YUnitValue",
    typeof(double),
    typeof(PointChartItem),
    new FrameworkPropertyMetadata(double.NaN, PointPropertyChanged));

/// <summary> ...
private static void PointPropertyChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)
{
    PointChartItem self = d as PointChartItem;

    if (!self.internalIsLoaded)
        return;

    self.RaiseEvent(new RoutedEventArgs(PointChartItem.UnitPointChangedEvent, self));
}
```



- Aufrufen des angelegten Events

Routed Events 2/3

(PointChartItem.cs)

```
/// <summary> ...
static PointChart()
{
    DefaultStyleKeyProperty.OverrideMetadata(
        typeof(PointChart),
        new FrameworkPropertyMetadata(typeof(PointChart)));

    EventManager.RegisterClassHandler(
        typeof(PointChart),
        PointChartItem.UnitPointChangedEvent,
        new RoutedEventHandler(OnItemUnitPointChangedEvent));
}

/// <summary> ...
private static void OnItemUnitPointChangedEvent(object sender, RoutedEventArgs e)
{
    PointChart pointChart = sender as PointChart;
    PointChartItem pointChartItem = e.OriginalSource as PointChartItem;

    pointChart.UpdatePosition(pointChartItem, pointChart.ItemContainerGenerator.IndexFromContainer(pointChartItem));
}
```

- Event abonnieren und Aktion definieren

Routed Events 3/3

(PointChart.cs)

VIEWMODEL KOMMUNIKATION

ViewModel Kommunikation - Extended

ViewModel Kommunikation - Extended

- Siehe SearchTextBox :)

ViewModel Kommunikation - Extended

- Siehe SearchTextBox :)

- SampleViewModel.cs, PointViewModel.cs implementieren INotifyPropertyChanged

ViewModel Kommunikation - Extended

- Siehe SearchTextBox :)

- SampleViewModel.cs, PointViewModel.cs implementieren INotifyPropertyChanged

- Binding der Items an XUnitValue, YUnitValue

ViewModel Kommunikation - Extended

- Siehe SearchTextBox :)

- SampleViewModel.cs, PointViewModel.cs implementieren INotifyPropertyChanged
- Binding der Items an XUnitValue, YUnitValue
- Binding der ItemsSource an ObservableCollection Points

- Binding PointChart
  - Points
  - SelectedItem

```
<PointChart x:Name="MyPointChart"
            Panel.ZIndex="100"
            ItemsSource="{Binding Points}"
            SelectedItem="{Binding SelectedItem}"
            SelectionMode="Extended">
    <PointChart.ItemContainerStyle>
        <Style TargetType="PointChartItem" BasedOn="{StaticResource PointChartItemStyle}">
            <Setter Property="XUnitValue" Value="{Binding X}" />
            <Setter Property="YUnitValue" Value="{Binding Y}" />
            <Setter Property="IsSelected" Value="{Binding IsSelected}" />
            <Setter Property="attachedProperties:AttachedProperties.IsSearchResult" Value="{Binding IsSearchResult}" />
        </Style>
    </PointChart.ItemContainerStyle>
</PointChart>
```

- Binding PointChartItem
  - XUnitValue
  - YUnitValue
  - IsSelected
  - ...

ViewModel Kommunikation 1/3

(MainView.xaml)

```
/// <summary>
/// PointsProperty gets or sets the Points.
/// </summary>
public ObservableCollection<PointViewModel> Points
{
    get
    {
        return pointsSource;
    }

    set
    {
        if (value != this.pointsSource)
        {
            this.pointsSource = value;
            OnPropertyChanged("Points");
        }
    }
}
```

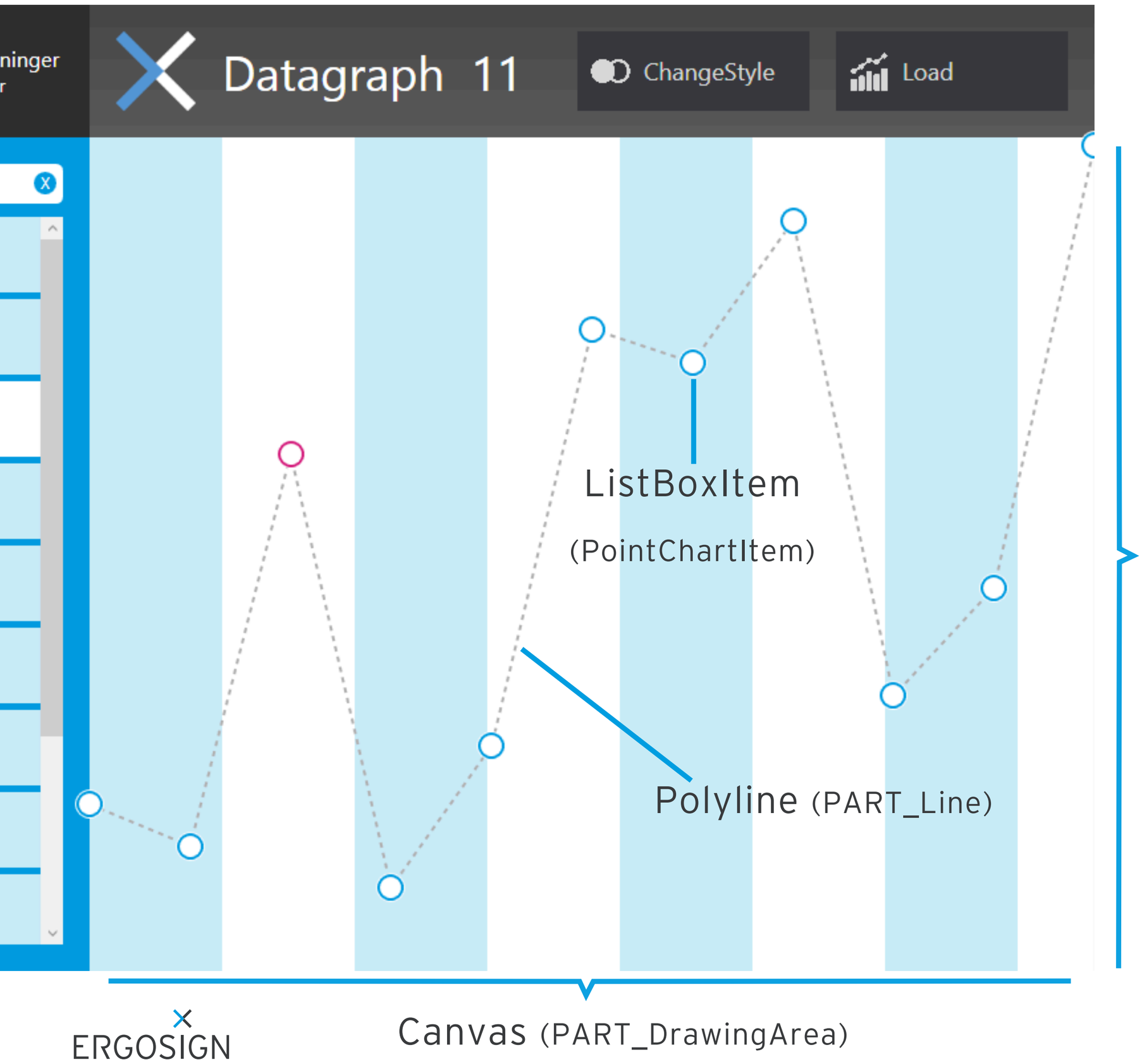
- Aufrufen von OnPropertyChanged (Update des Bindings)

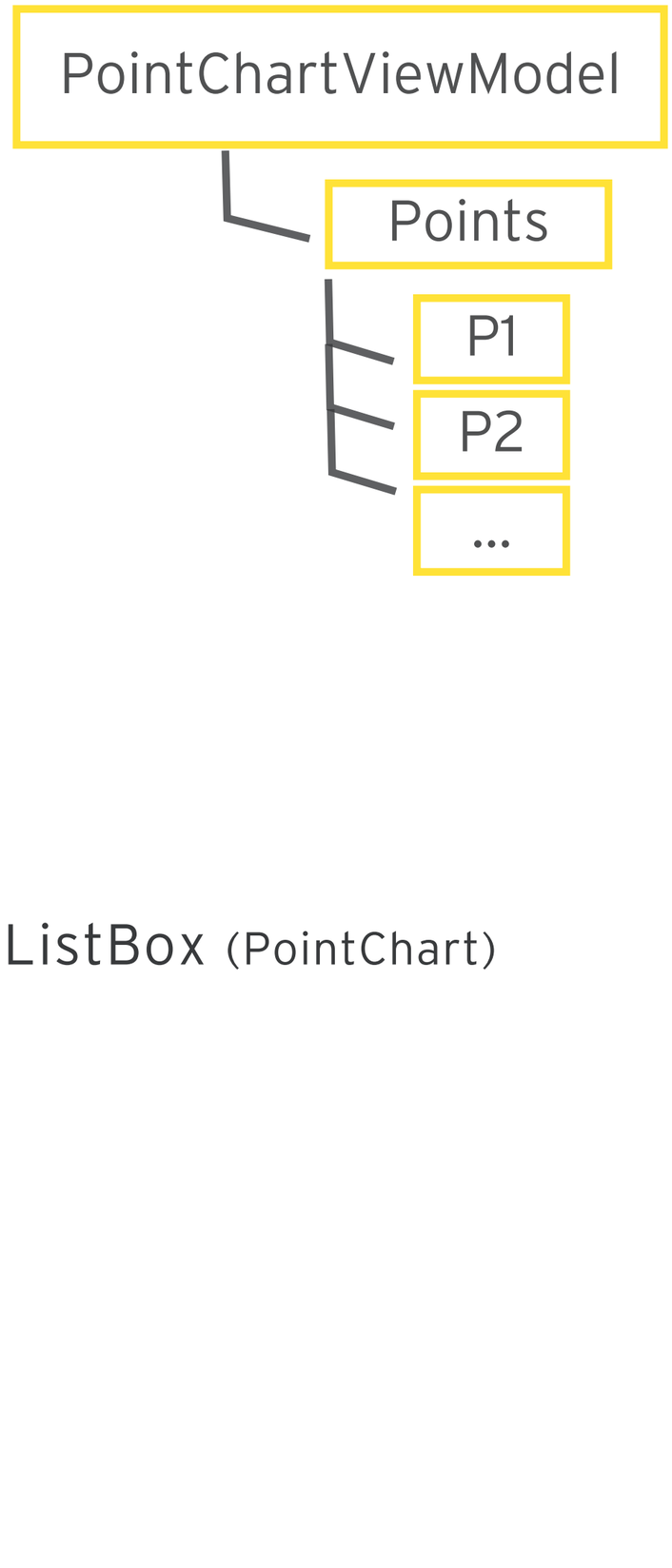
```
/// <summary>
/// SelectedItemProperty gets or sets the SelectedItem.
/// </summary>
public object SelectedItem
{
    get
    {
        return selectedItem;
    }

    set
    {
        if (value != this.selectedItem)
        {
            this.selectedItem = value;
            Debug.WriteLine("SelectedItem " + SelectedItem);
            OnPropertyChanged("SelectedItem");
        }
    }
}
```

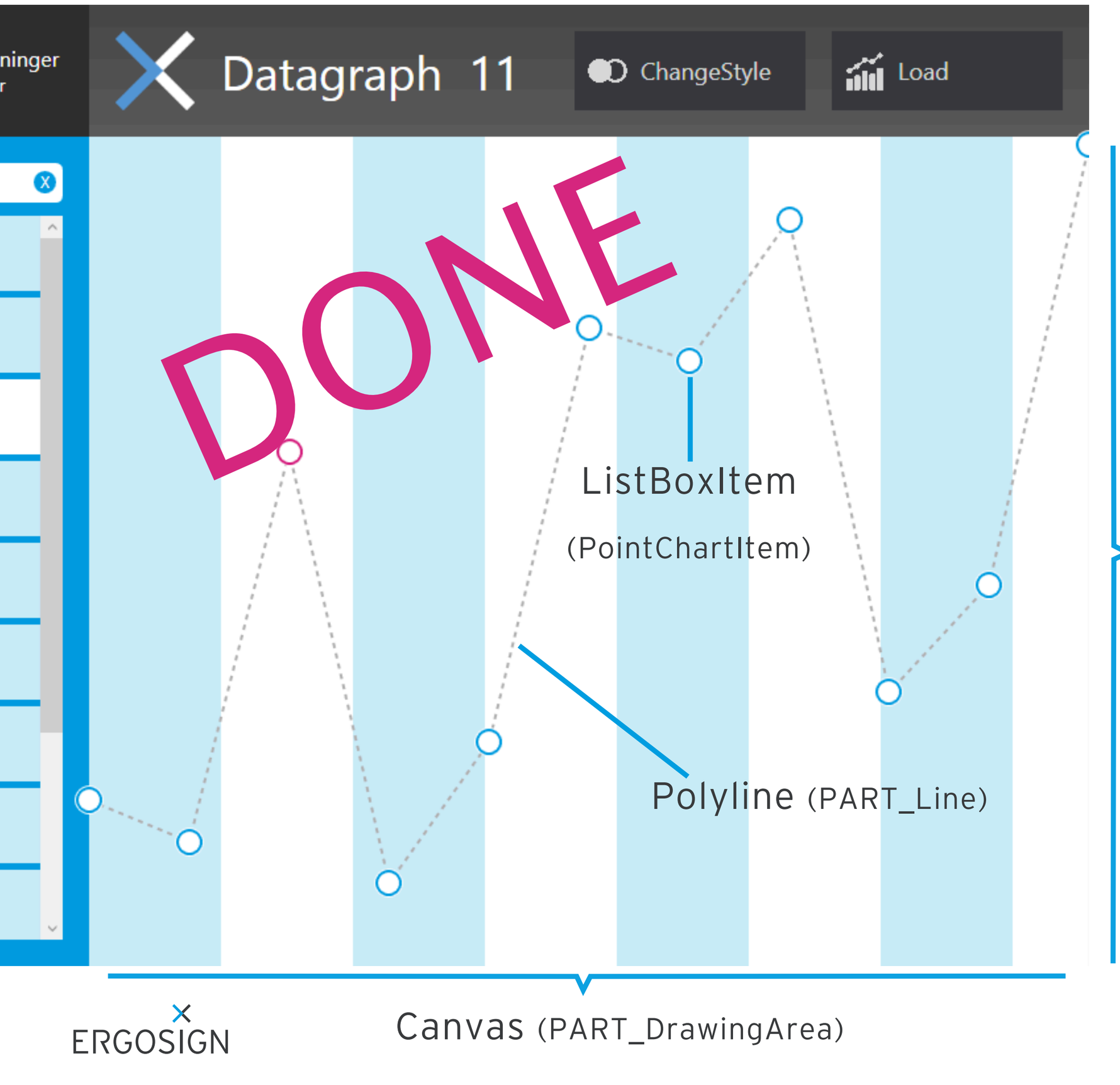
- Aufruf der Update Logik

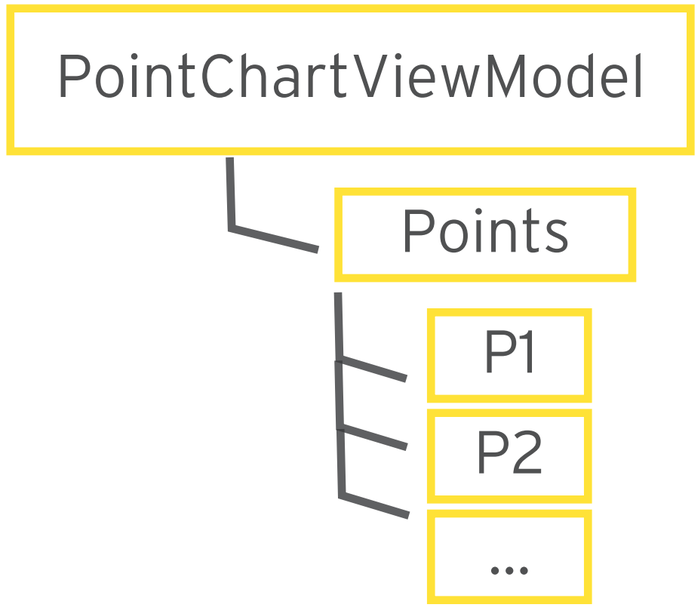
ViewModel Kommunikation 2/3

(SampleViewModel.cs)


ninger
r
Datagraph 11
ChangeStyle
Load
ListBoxItem
(PointChartItem)
Polyline (PART_Line)
Canvas (PART_DrawingArea)


PointChartViewModel
Points
P1
P2
...
ListBox (PointChart)


ninger
r
Datagraph 11
ChangeStyle
Load
DONE
ListBoxItem
(PointChartItem)
Polyline (PART_Line)
ListBox (PointChart)
Canvas (PART_DrawingArea)


PointChartViewModel
Points
P1
P2
...

BLENDABILITY

Blendability

Blendability

- Xaml friendly und Blend kompatibel

Blendability

- Xaml friendly und Blend kompatibel
- Default Property Values, Styles und Templates

Blendability

- Xaml friendly und Blend kompatibel
- Default Property Values, Styles und Templates
- Zusätzliche Ressourcen als Properties

Blendability

- Xaml friendly und Blend kompatibel
- Default Property Values, Styles und Templates
- Zusätzliche Ressourcen als Properties
- Keine tiefe Verschachtelung der Templates

Blendability

- Xaml friendly und Blend kompatibel
- Default Property Values, Styles und Templates
- Zusätzliche Ressourcen als Properties
- Keine tiefe Verschachtelung der Templates
- Mode Properties für Trigger

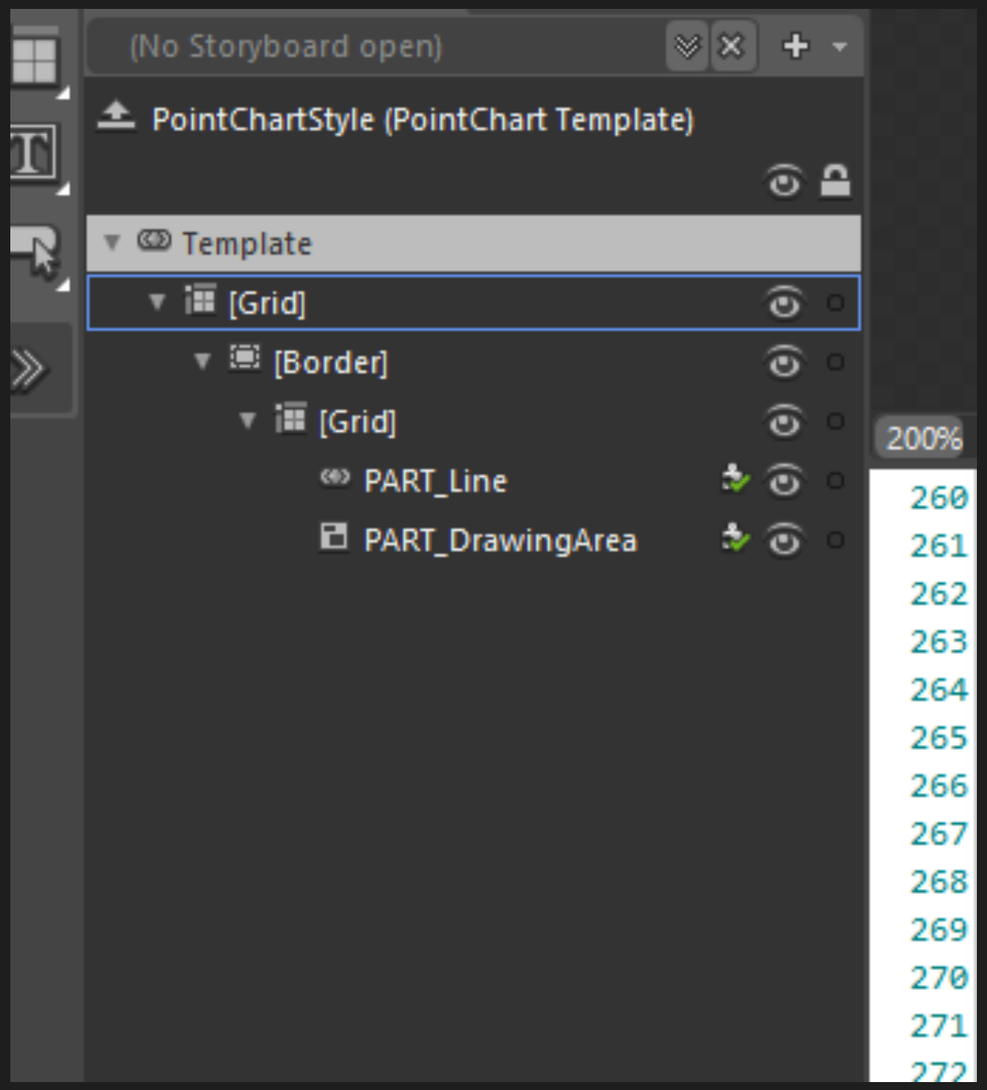
Blendability

- Xaml friendly und Blend kompatibel
- Default Property Values, Styles und Templates
- Zusätzliche Ressourcen als Properties
- Keine tiefe Verschachtelung der Templates
- Mode Properties für Trigger
- Design Time Features verwenden

Blendability

- Xaml friendly und Blend kompatibel
- Default Property Values, Styles und Templates
- Zusätzliche Ressourcen als Properties
- Keine tiefe Verschachtelung der Templates
- Mode Properties für Trigger
- Design Time Features verwenden
- …

```
/// <summary> ...
[TemplatePart(Name = "PART_DrawingArea", Type = typeof(FrameworkElement))]
[TemplatePart(Name = "PART_Line", Type = typeof(Polyline))]
public class PointChart : ListBox...
```



TemplatePart Attribute

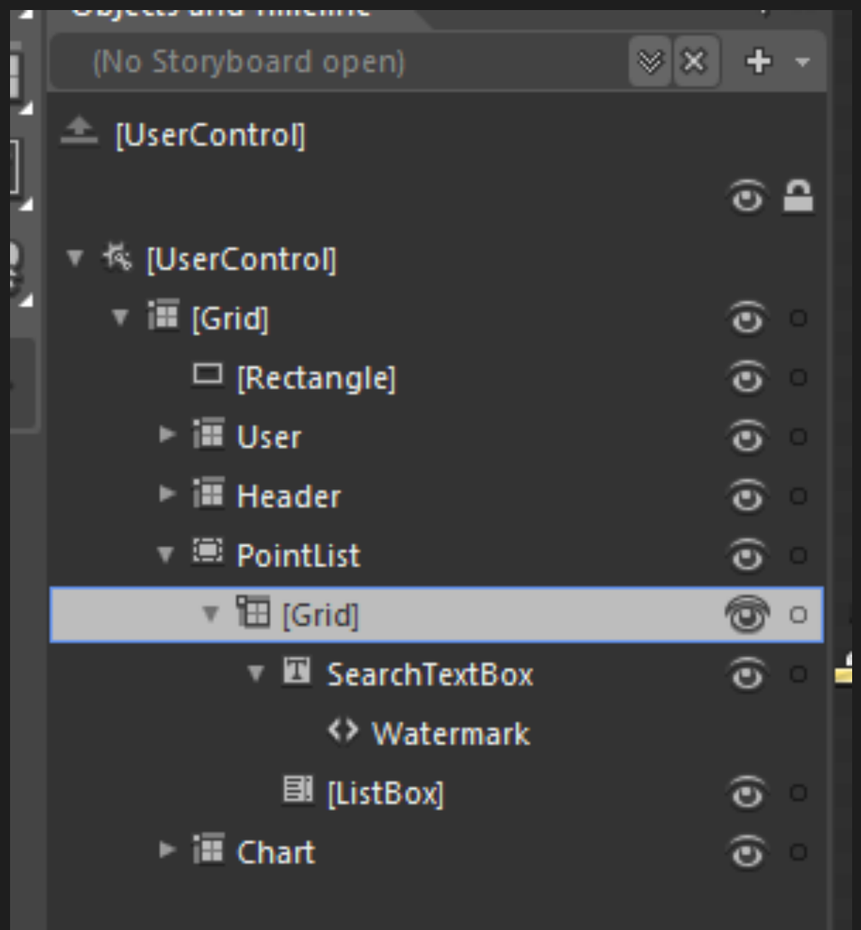
(PointChart.cs)

```
/// <summary> ...
[ContentPropertyAttribute("MyDefaultContentProperty")]
[Description("The class SearchTextBox represents a textbox with watermark and clear text command")]
public class SearchTextBox : TextBox
{
    Members

    DependencyProperties

    PropertyChangedCallbacks

    Events

    Commands

    Handlers

    Ctor

    Init/Cleanup

    OverrideMethods

    Methods

    Properties
```

ContentProperty Attribute

(SearchTextBox.cs)

```
/// <summary> ...
[Description("Gets or sets the watermark."), Category(PROPERTY_CATEGORY)]
[AlternateContentProperty]
public string Watermark
{
    get { return (string)GetValue(WatermarkProperty); }
    set { SetValue(WatermarkProperty, value); }
}
```



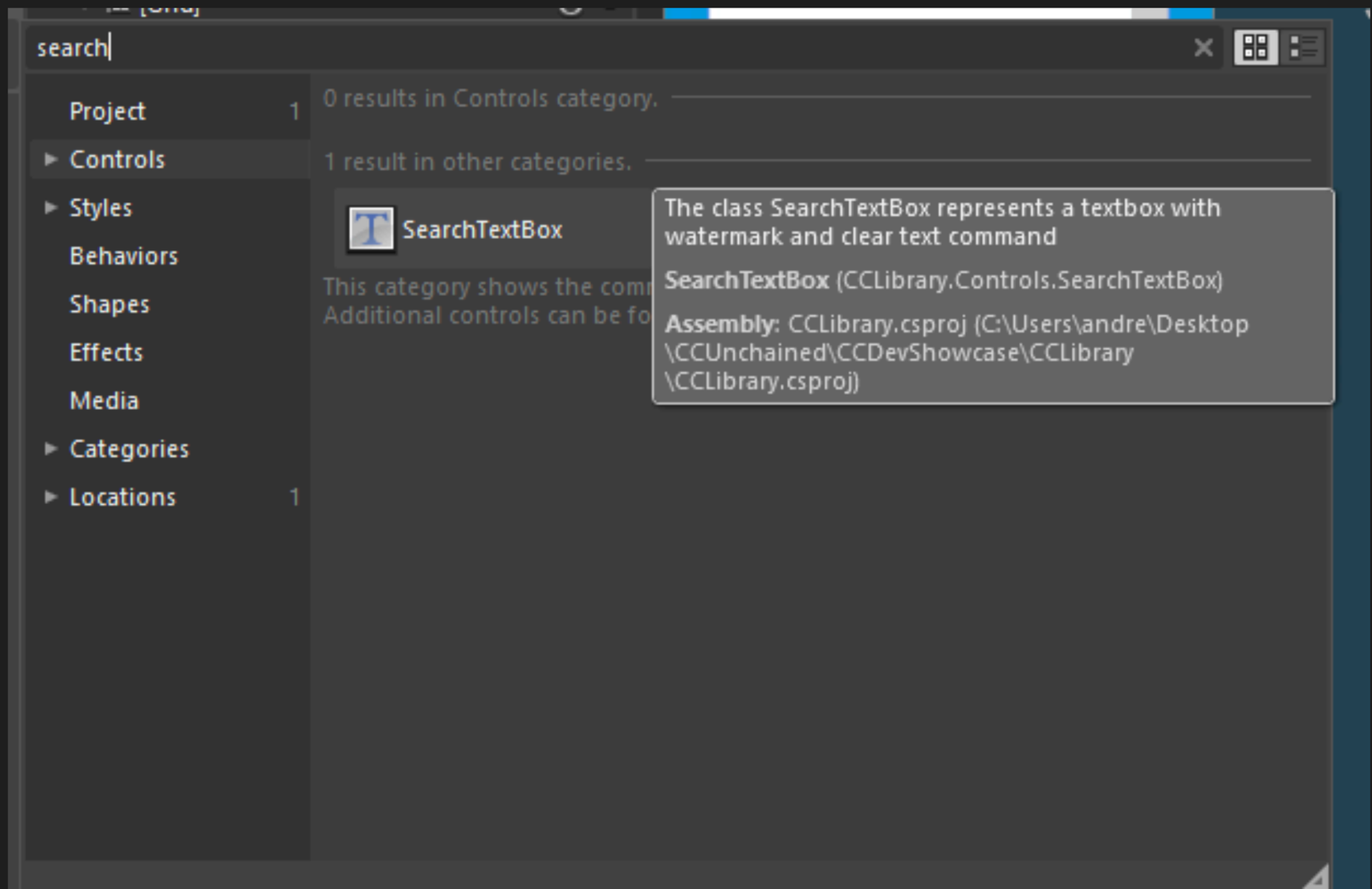
AlternateContent-Property Attribute

(SearchTextBox.cs)

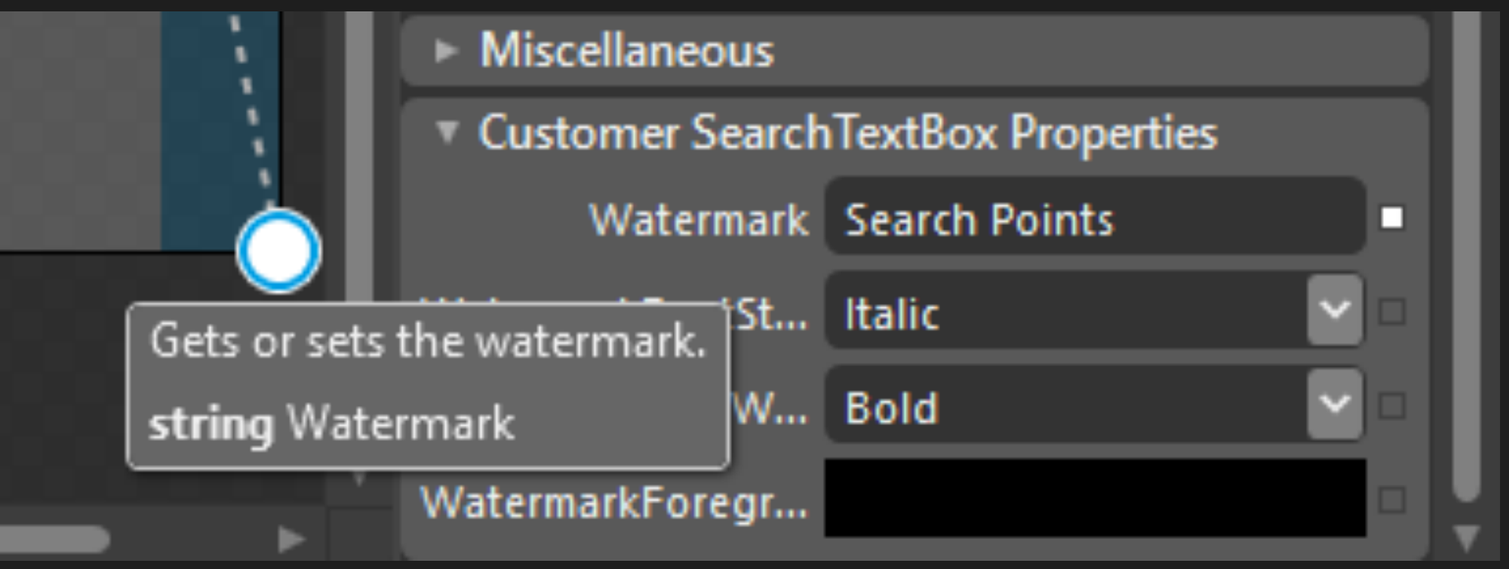
Description Attribute 1/2

(SearchTextBox.cs)

```
/// <summary> ...
[ContentPropertyAttribute("MyDefaultContentProperty")]
[Description("The class SearchTextBox represents a textbox with watermark and clear text command")]
public class SearchTextBox : TextBox
{
    Members

    DependencyProperties

    PropertyChangedCallbacks

    Events

    Commands

    Handlers

    Ctor

    Init/Cleanup

    OverrideMethods

    Methods

    Properties
```



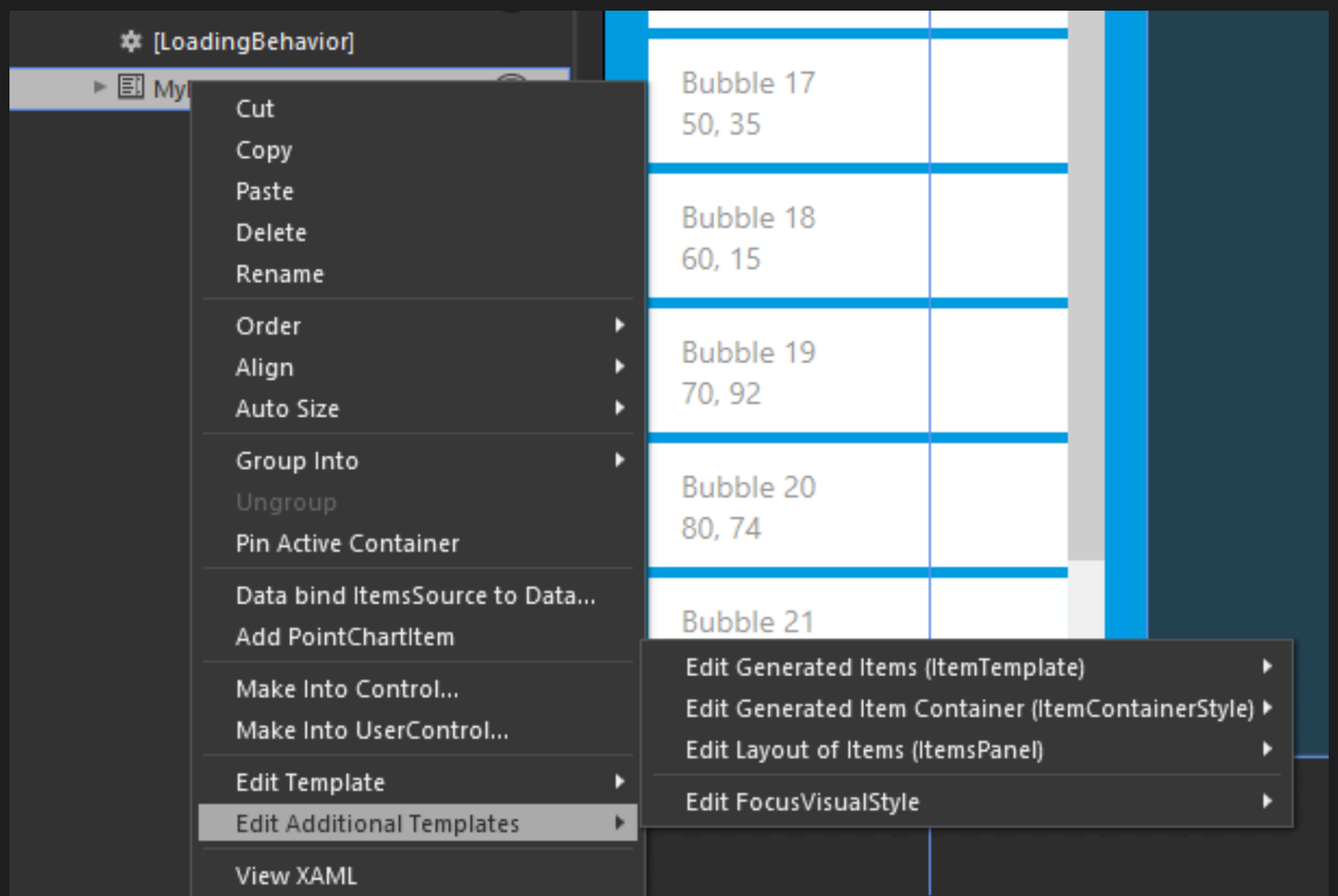
```
/// <summary> ...
[Description("Gets or sets the watermark."), Category(PROPERTY_CATEGORY)]
[AlternateContentProperty]
public string Watermark
{
    get { return (string)GetValue(WatermarkProperty); }
    set { SetValue(WatermarkProperty, value); }
}
```



Description Attribute 2/2

(SearchTextBox.cs)

```
[TemplatePart(Name = "PART_DrawingArea", Type = typeof(FrameworkElement))]
[TemplatePart(Name = "PART_Line", Type = typeof(Polyline))]
[StyleTypedProperty(Property = "ItemContainerStyle", StyleTargetType = typeof(PointChartItem))]
[Description("The class PointChart represents a selectable point chart")]
public class PointChart : ListBox
{
    Members

    DependencyProperties

    PropertyChangedCallbacks
```



StyledTypedProperty Attribute

(PointChart.cs)

```
[assembly: XmlnsDefinition("http://schemas.microsoft.com/winfx/2006/xaml/presentation", "CCLibrary.Controls.PointChart")]
[assembly: XmlnsDefinition("http://schemas.microsoft.com/winfx/2006/xaml/presentation", "CCLibrary.Controls.HighlightableTextBlock")]
[assembly: XmlnsDefinition("http://schemas.microsoft.com/winfx/2006/xaml/presentation", "CCLibrary.Controls.SearchTextBox")]
[assembly: XmlnsDefinition("http://schemas.microsoft.com/winfx/2006/xaml/presentation", "CCLibrary.Controls.CircularProgressBar")]
[assembly: XmlnsDefinition("http://schemas.microsoft.com/winfx/2006/xaml/presentation", "CCLibrary.Controls.NumericUpDown")]
[assembly: XmlnsDefinition("http://schemas.microsoft.com/winfx/2006/xaml/presentation", "CCLibrary.Behaviors")]
```

```
<serachTextBox:SearchTextBox x:Name="SearchTextBox"
                             Margin="0 0 0 10"
                             Text="{Binding SearchString,
                                            UpdateSourceTrigger=PropertyChanged}"
                             Watermark="Search Points" />

<SearchTextBox x:Name="SearchTextBoxWithDefaultMerge"
               Margin="0 0 0 10"
               Text="{Binding SearchString,
                              UpdateSourceTrigger=PropertyChanged}"
               Watermark="Search Points" />
```

Namespace Merge

(AssemblyInfo.cs)

```
if (!DesignerProperties.GetIsInDesignMode(this))
    return;
```

DesignTime

(AssemblyInfo.cs)

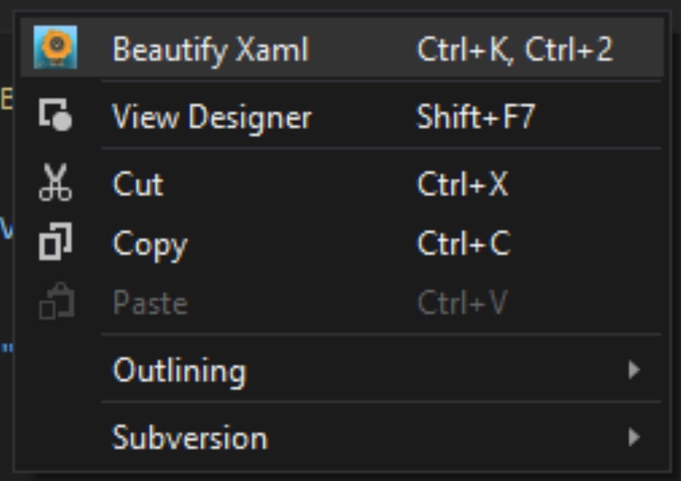
TOOLS :)

1

```
<Grid Width="{TemplateBinding MinWidth}" Height="{TemplateBinding MinHeight}">
    <Ellipse Margin="-1" Fill="{StaticResource WhiteBrush}"
                    StrokeThickness="2" />
    <Ellipse Fill="{TemplateBinding Background}"
                           Stroke="{TemplateBinding BorderBrush}"
       StrokeThickness="2" /></Grid>
<Grid x:Name="HoverBorder"
```

2

```
<Grid Width="{TemplateBinding MinWidth}" Height="{TemplateBinding MinHeight}">
    <Ellipse Margin="-1" Fill="{StaticResource WhiteBrush}"
                    StrokeThickness="2" />
    <Ellipse Fill="{TemplateBinding Background}"
                           Stroke="{TemplateBinding B
       StrokeThickness="2" /></Grid>
<Grid x:Name="HoverBorder"
      Margin="0 0 0 30" HorizontalAlignment="Center" V
      Opacity="0">
    <Grid.Effect>
        <DropShadowEffect PresentationOptions:Freeze="
    </Grid.Effect>
    <Grid.RowDefinitions>
        <RowDefinition Height="auto" />
```



3

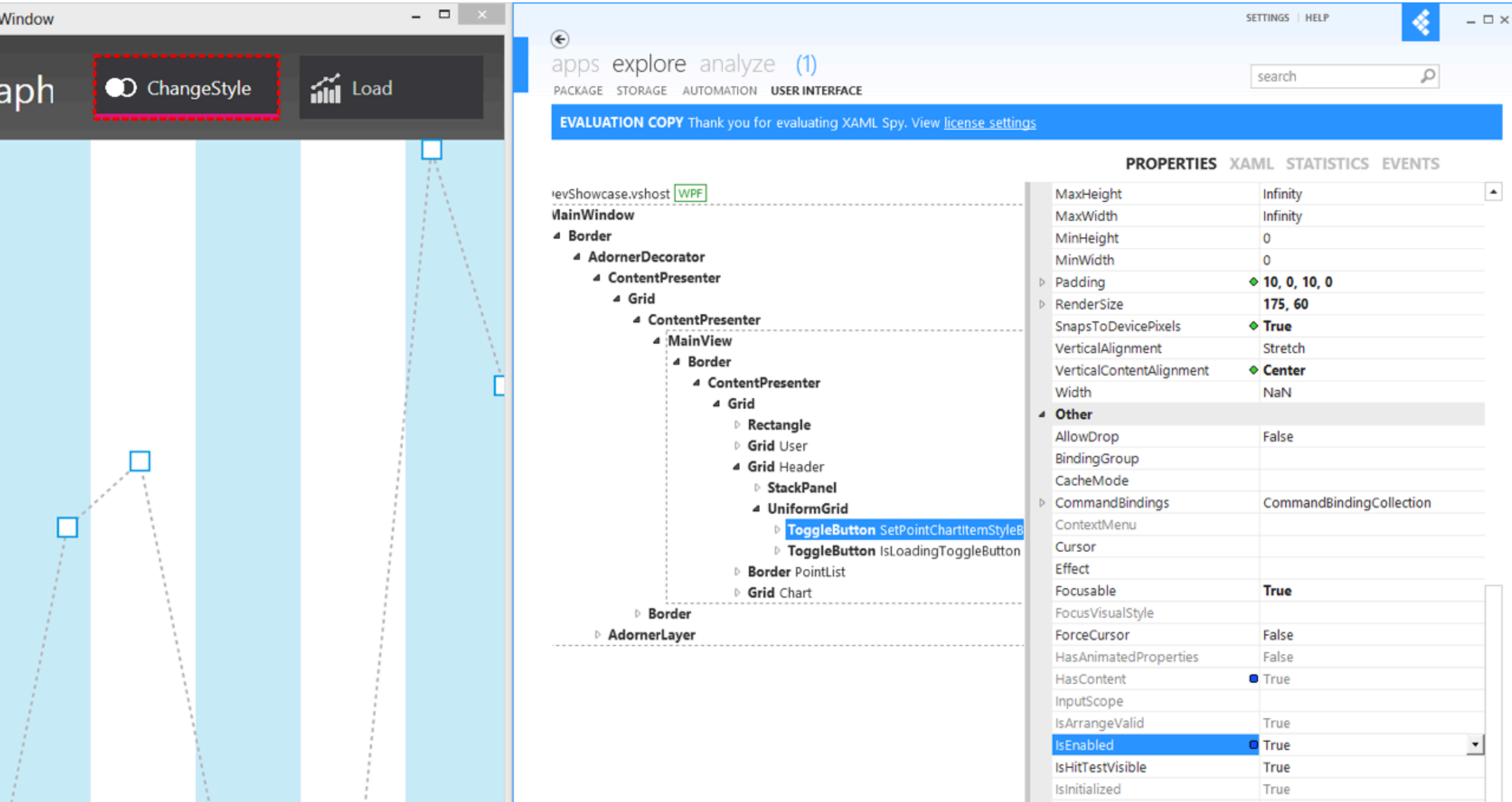
```
<Grid>
    <Grid Width="{TemplateBinding MinWidth}" Height="{TemplateBinding MinHeight}">
        <Ellipse Margin="-1"
                 Fill="{StaticResource WhiteBursh}"
                 StrokeThickness="2" />
        <Ellipse Fill="{TemplateBinding Background}"
                 Stroke="{TemplateBinding BorderBrush}"
                 StrokeThickness="2" />
    </Grid>
    <Grid x:Name="HoverBorder"
```

Xaml Styler

- Element & Speicher Analyse

Xaml Spy

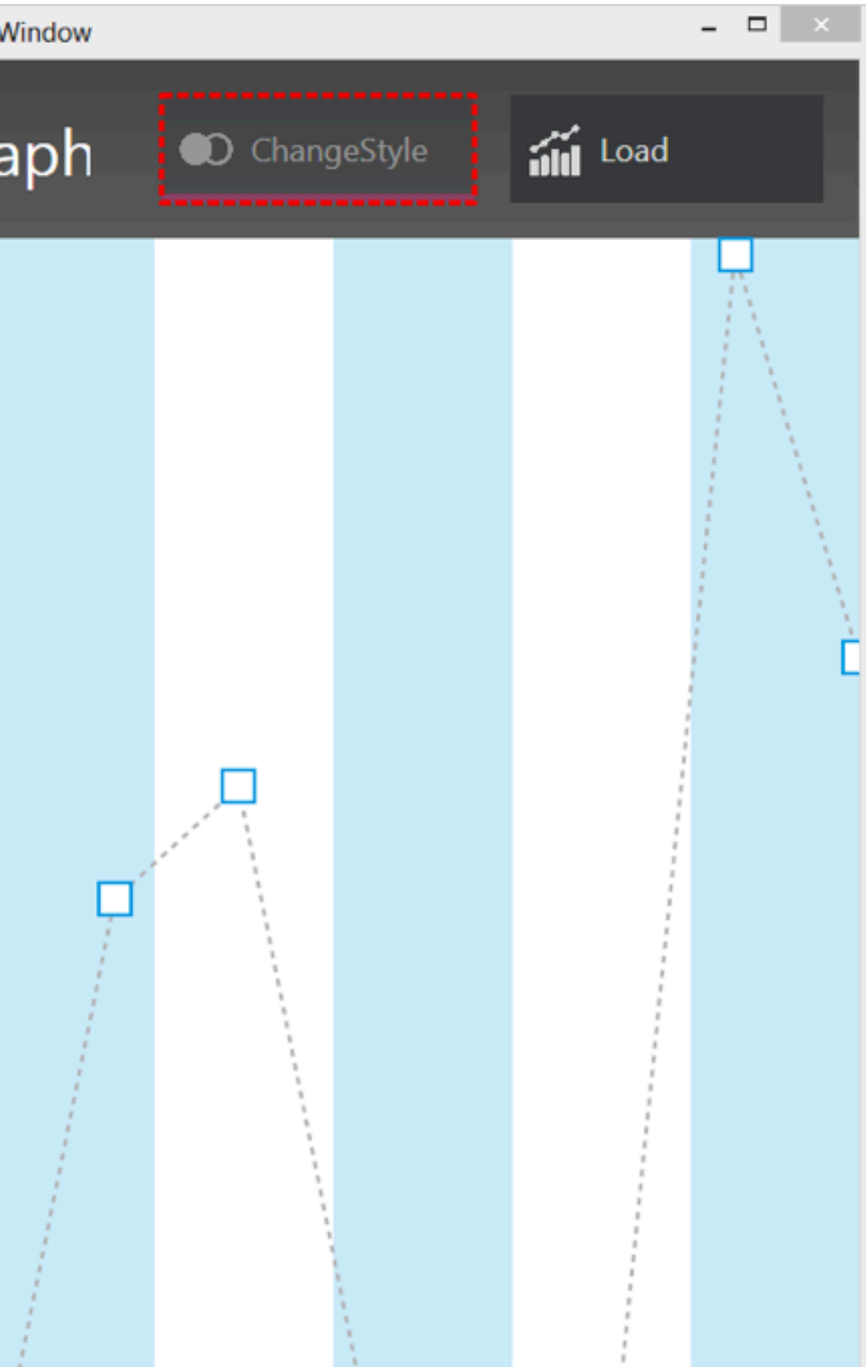
(Alternative WPF Inspector)

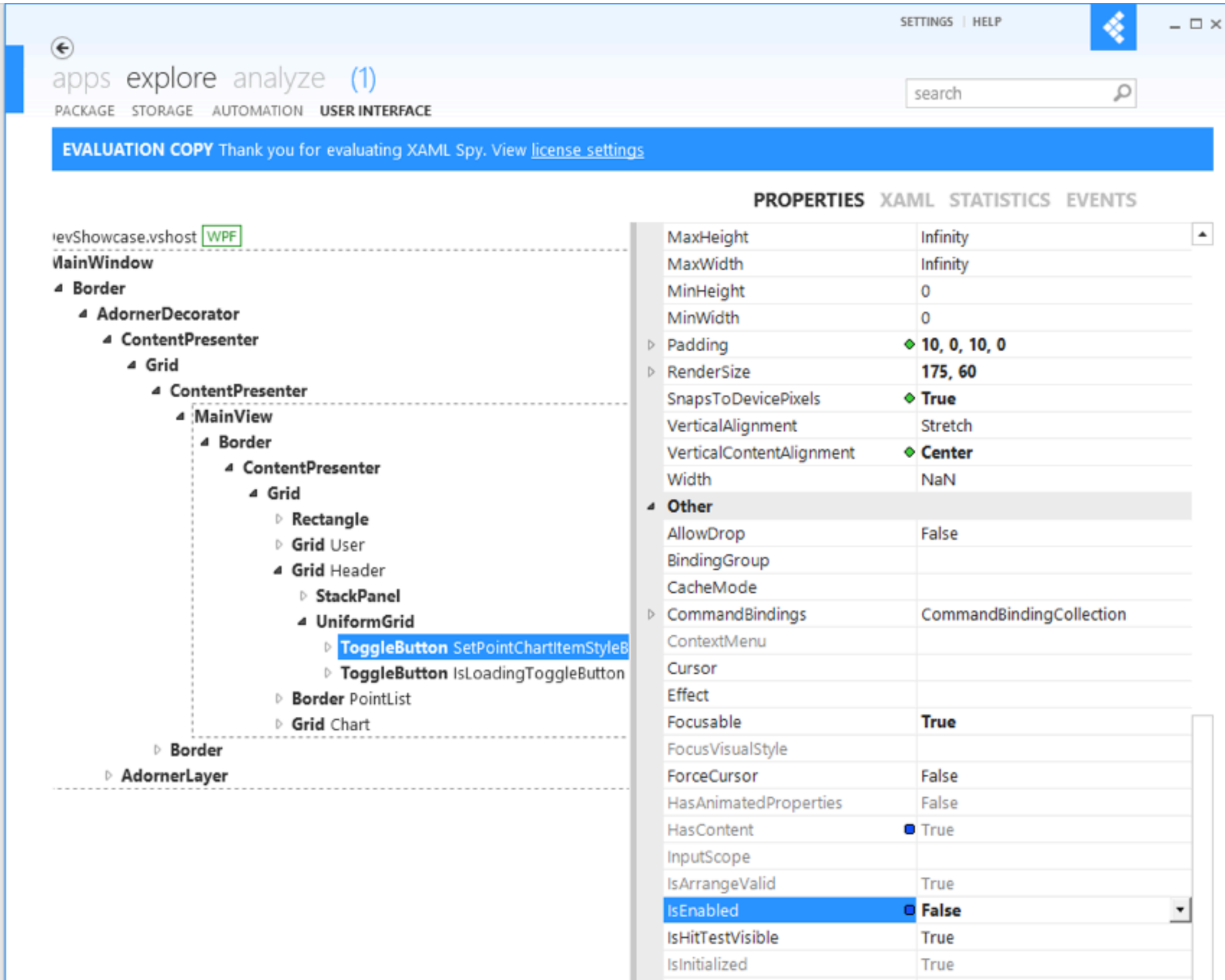


- Element & Speicher Analyse

Xaml Spy

(Alternative WPF Inspector)

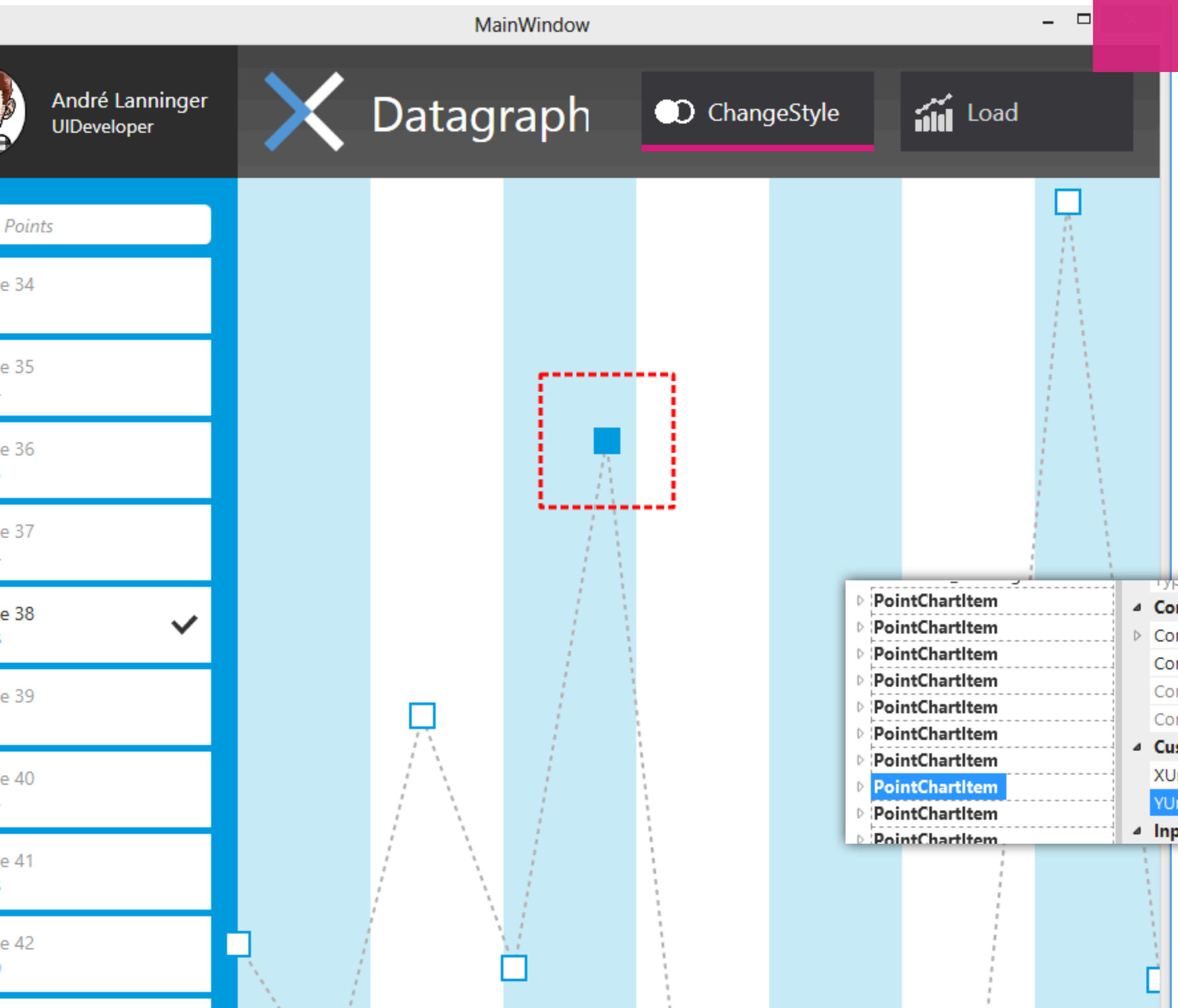




› **Live** Property Editing

Xaml Spy

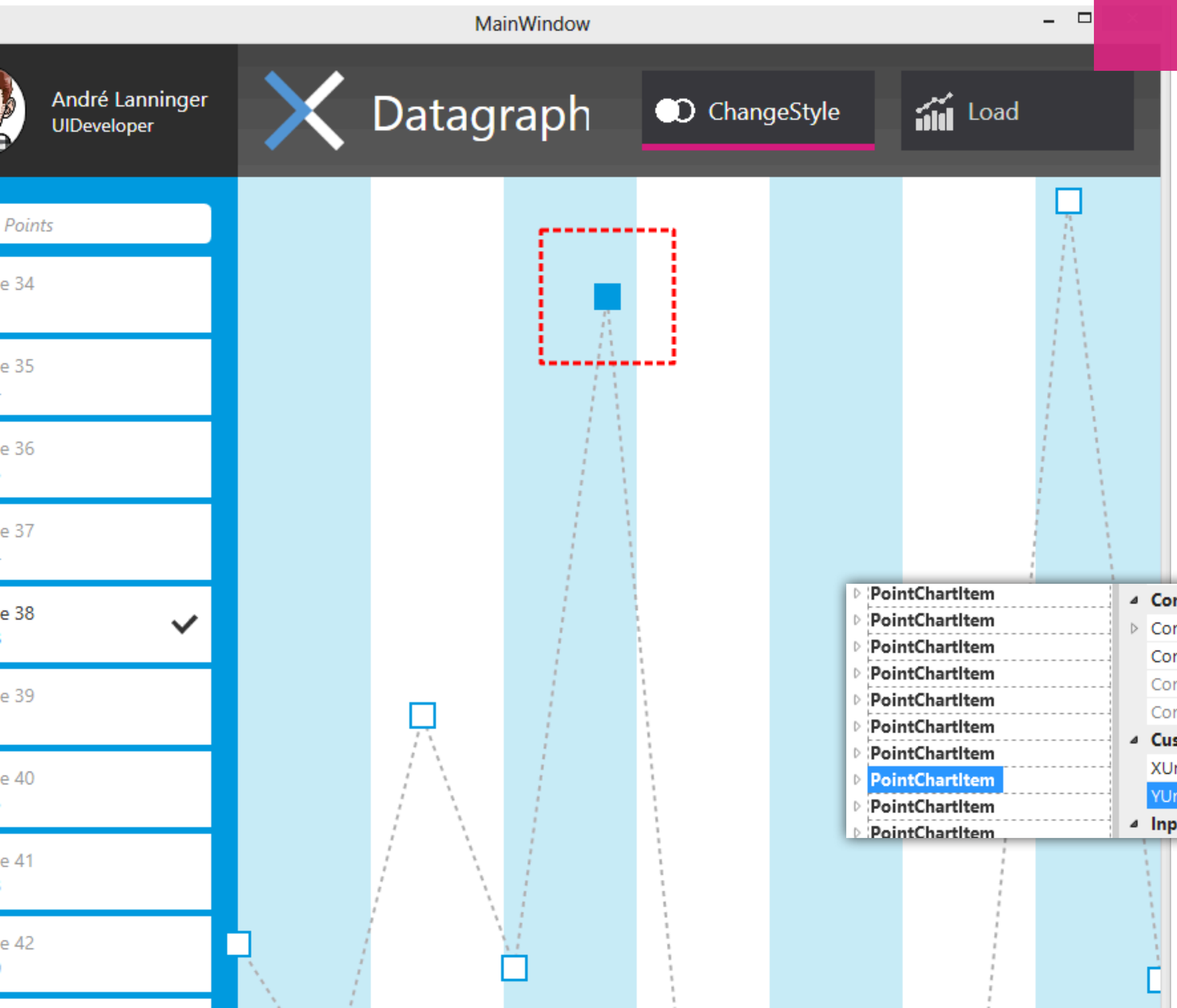
(Alternative WPF Inspector)



- **Live** Property Editing

Xaml Spy

(Alternative WPF Inspector)



FAZIT

FAZIT

- WPF bietet die optimale Basis für Custom Control Development
- Custom Controls sinnvoll einsetzen
- Sorgen für mehr Konsistenz (Bausteine)
- Erhöhen die Produktivität (Implementierung, Blendability)

VIELEN DANK.

ERGOSIGN

VIELEN DANK.

www.ergosign.de
contact@ergosign.de

ERGOSIGN

Ergosign GmbH
Europaallee 12
66113 Saarbrücken
Germany

T +49 681 988412-0
F +49 681 988412-10

Ergosign GmbH
Bernhard-Nocht-Straße 109
20359 Hamburg
Germany

T +49 40 3179868-0
F +49 40 3179868-10

Ergosign GmbH
Adams-Lehmann-Straße 44
80797 München
Germany

T +49 89 6890607-0
F +49 89 6890607-10

Ergosign Switzerland AG
Badenerstrasse 808
8048 Zürich
Switzerland

T +41 44 54293-04
F +41 44 54293-07

ERGOSIGN

Ihr Ansprechpartner
André Lanninger
UI Developer

T +49 681 988 412-0
lanninger@ergosign.de

ERGOSIGN

ERGOSIGN

Ihr Ansprechpartner
André Lanninger
UI Developer

T +49 681 988 412-0
lanninger@ergosign.de

www.ergosign.de
contact@ergosign.de

ERGOSIGN

Ergosign GmbH
Europaallee 12
66113 Saarbrücken
Germany

T +49 681 988412-0
F +49 681 988412-10

Ergosign GmbH
Bernhard-Nocht-Straße 109
20359 Hamburg
Germany

T +49 40 3179868-0
F +49 40 3179868-10

Ergosign GmbH
Adams-Lehmann-Straße 44
80797 München
Germany

T +49 89 6890607-0
F +49 89 6890607-10

Ergosign Switzerland AG
Badenerstrasse 808
8048 Zürich
Switzerland

T +41 44 54293-04
F +41 44 54293-07