

XUI

(eXtensible User Interface)

Definition Language

- Version 1.0 -



JAXFront® XML Rendering Engine
www.jaxfront.com

INHALTSVERZEICHNIS

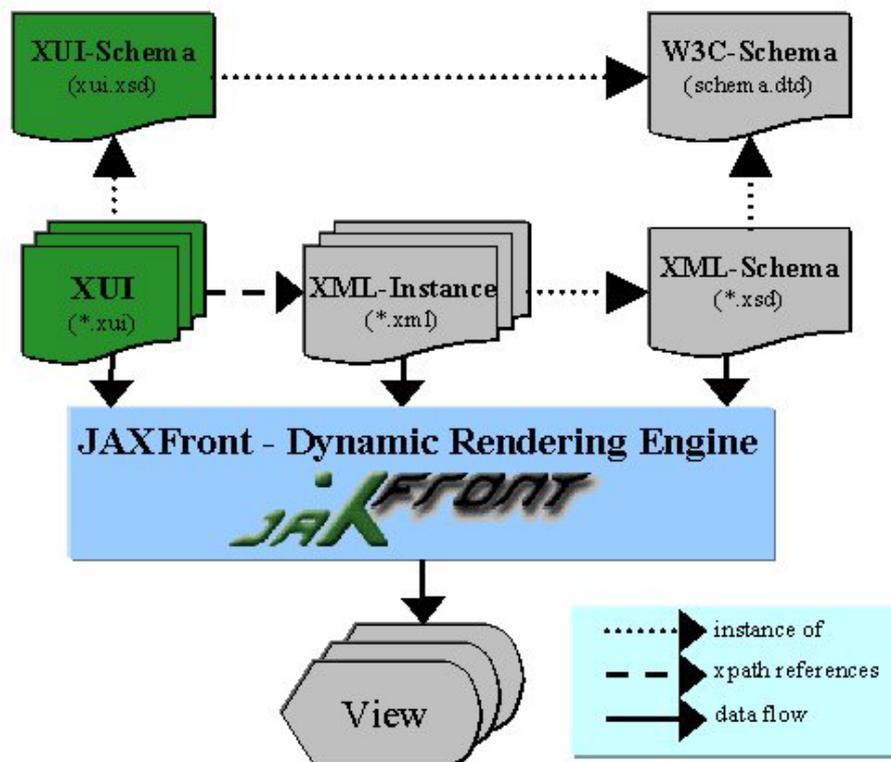
1	XUI: SPRACHDEFINITION	3
1.1	EINLEITUNG	3
1.2	LOGISCHE STRUKTUR	4
1.3	KOMPONENTENÜBERGREIFENDE DEFINITIONEN	5
1.4	STYLE DEFINITIONEN.....	6
1.5	BEHAVIOUR - DEFINITIONEN	7
1.5.1	<i>Navigationstyp</i>	7
1.5.2	<i>Formelausdruck</i>	8
1.5.3	<i>Regeln</i>	8
2	DARSTELLUNGSTYPEN	10
3	STARTEN DES GUITESTERS	12
4	EIN BEISPIEL: PURCHASE ORDER	13
4.1	DAS XML-SCHEMA (PO.XSD)	13
4.2	DIE XML-INSTANZ (PO.XML)	14
4.3	DIE XUI-DEFINITION (PO.XUI).....	14
5	XUI SCHEMA SPEZIFIKATION (XUI.XSD)	19

1 XUI: SPRACHDEFINITION

1.1 Einleitung

Um eine generisch erzeugte Benutzeroberfläche seinen eigenen Bedürfnissen anzupassen, muss die Möglichkeit bestehen, die erzeugten Darstellungstypen (siehe Kapitel ‚Darstellungstypen‘) deklarativ zu parametrisieren. Zu diesem Zweck muss der gesamte Visualisierungsbaum eines XML Schemas in ein benutzerfreundliches Format serialisiert werden. Dieses Format soll einfach zu verstehen und möglichst erweiterbar sein, da die Parameter für die genauere Spezifizierung von graphischen Elementen sehr umfassend sind. Die erstellte Spezifikation soll möglichst abstrakt, das heisst unabhängig von einem bestimmten Ausgabeformat oder Programmiersprache gehalten werden. Es wäre denkbar, für jedes zu erstellende Ausgabeformat eine eigene GUI Spezifikationssprache zu entwerfen. Das folgende Konzept konzentriert sich jedoch auf das Erzeugen von Java Swing Benutzeroberflächen.

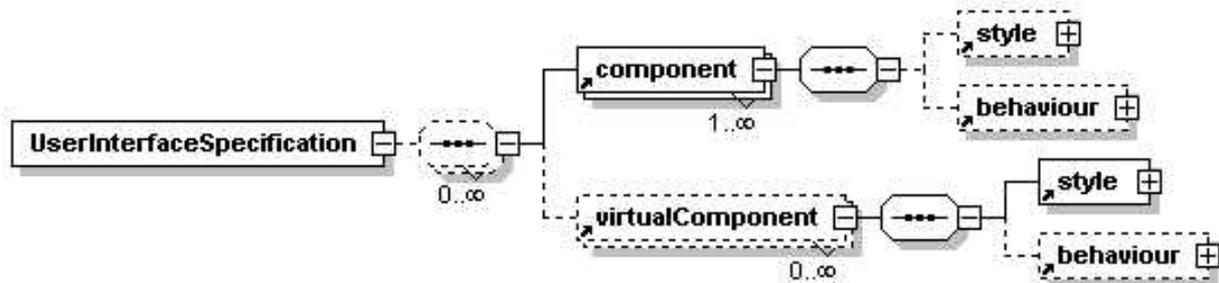
Eine XUI-Beschreibung wird immer für eine bestimmte XML-Instanz (Instance) geschrieben. Diese XML-Instanz kann nur dann von der Rendering-Engine verarbeitet werden, wenn diese eine gültige Referenz auf ein XML-Schema besitzt. Sowohl das von einer XML-Instanz referenzierte XML-Schema als auch das Schema für die GUI-Beschreibungen (XUI's) sind in der vom W3C (World-Wide-Web Consortium) standardisierten XML Schemasprache geschrieben und sind somit Instanzen der offiziellen W3C Schemaspezifikation.



Mit XPath¹ werden Teile oder Teilbereiche aus der XML-Instanz adressiert und mit layout- und regelspezifischen Definitionen bereichert. Die Möglichkeiten für diese GUI-Veredelung sind in einem XML-Schema (XUI-Schema) beschrieben. Das XUI-Schema stellt somit die Gesamtheit aller möglichen Anpassungsmöglichkeiten an die graphische Oberfläche.

¹ Mit XPath können Pfadausdrücke formuliert werden, die Teile eines XML - Dokuments lokalisieren.

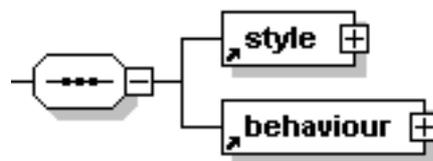
1.2 Logische Struktur



Grundsätzlich gibt es innerhalb der XUI-Beschreibung zwei verschiedene Komponenten. Alle Komponenten, welche im XML Template durch ein XPATH-Statement adressierbar sind, werden als `<component>` eingetragen. Unter virtuellen Komponenten (`<virtualComponent>`) verstehen wir graphische Elemente, die nicht in einer XML-Instanz vorzufinden sind und somit keine Daten aus der Instanz direkt modifizieren.

Um eine möglichst kleine Koppelung (engl. loose coupling) zwischen der im XML Schema beschriebenen Struktur- und der visuellen Repräsentationsverschachtelung zu erhalten, wird die beschriebene Verschachtelung der Darstellungstypen aus dem XML Schema nicht in der XUI Spezifikation wiederzufinden sein. Stattdessen wird jeder Darstellungstyp mittels absoluter Adressierung (XPath) genau identifiziert.

Die XUI Spezifikation besteht also aus einer Menge von absolut adressierten Visualisierungskomponenten, welche ein Element aus einer XML-Instanz mit einem Darstellungstypen und den dazugehörigen Parametrisierungsmöglichkeiten für diesen Typen verbindet. Grundsätzlich unterscheiden wir zwischen zwei verschiedenen Informationseinheiten für jeden Darstellungstypen:



Präsentations- relevante Informationen (style)

Definieren das Aussehen (Position, Farben, Grösse etc.) eines Darstellungstypen.

Verhaltens- relevante Informationen (behaviour)

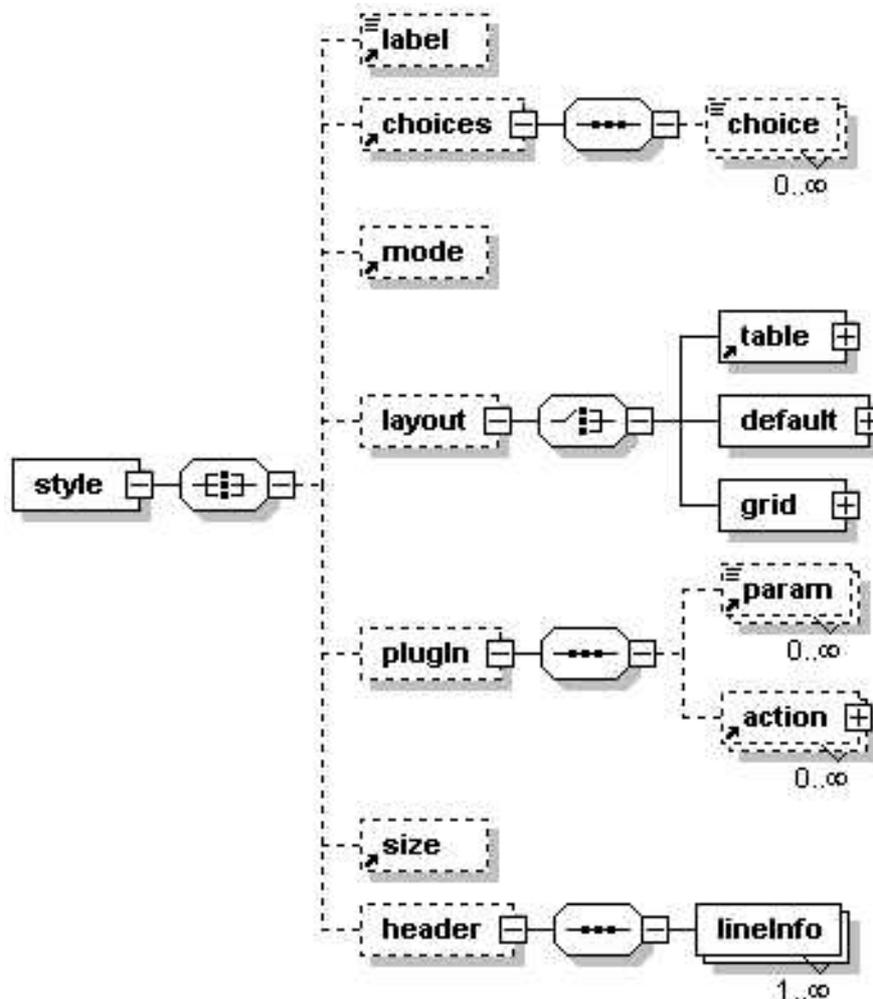
Definieren visuelle Ereignissteuerungen, Regeln (Ein-/Ausblenden, Navigationsmöglichkeiten etc.) aufgrund von eintretenden Datenänderungen.

1.3 Komponentenübergreifende Definitionen

Die unter dem WurzelElement *<UserInterfaceSpecification>* definierten Attribute-Einträge betreffen die gesamte XML-Instanz und sind bis auf die Versionsnummer (versionNo) allesamt optional:

Name	Beschreibung
versionNo	Definiert die Versionsnummer der verwendeten XUI Definitionssprache (zBsp.: 1.0)
useStatusBar	Verwendet eine Statusbar um Fehlermeldungen oder Tooltips zu visualisieren.
useButtonBar	Verwendet eine Buttonbar mit den Standardbuttons „save“ und „cancel“.
usePlugins	Besagt ob JavaBeans als Pluginklassen benutzt werden sollen.
systemExitOnClose	Besagt ob die Applikation beim schliessen des Objekteditors geschlossen werden soll.
useBackwardButton	Ermöglicht das Zurückspringen zum nächsthöheren Knoten innerhalb des Navigationsbaumes.
useForwardButton	Ermöglicht das Vorwärtsspringen zum nächsten Knoten innerhalb des Navigationsbaumes.
verticalScrollBarPolicy	Definiert das vertikale Scroll-Verhalten.
horizontalScrollBarPolicy	Definiert das horizontale Scroll-Verhalten.
useCharacterIndent	Definiert die Anzahl leerer Charakter, welche für die Erstellung der Labels der Darstellungstypen eingeschoben werden sollen.
allowSavingWithErrors	Besagt, ob eine XML Instanz trotz Validierungsfehlern gespeichert werden darf.

1.4 Style Definitionen



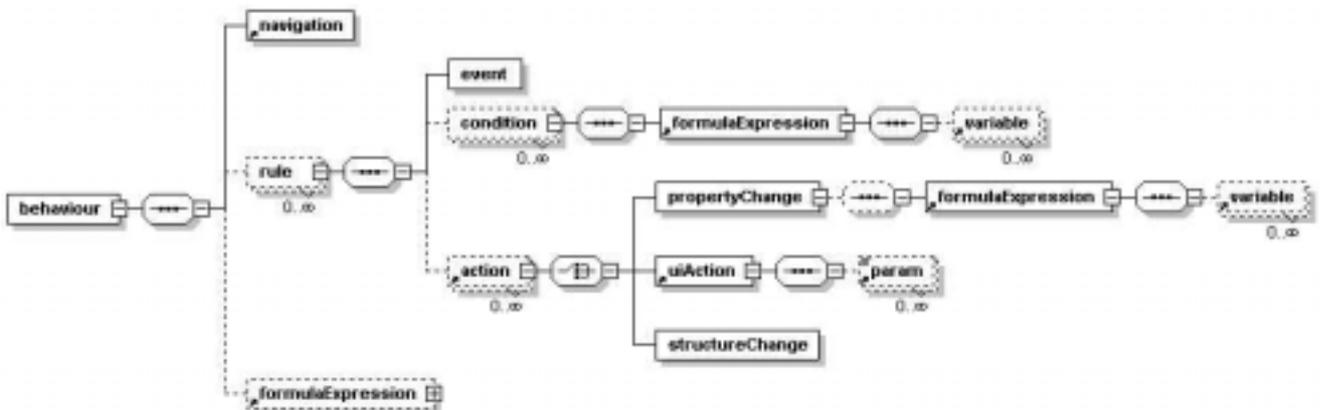
Die unter `<style>` aufgeführten Einträge erheben keineswegs den Anspruch vollständig zu sein, es soll lediglich hier illustriert werden, dass die Anzahl an Parametrisierungsmöglichkeiten jederzeit erweitert oder angepasst werden kann. So kann zum Beispiel eine Mehrsprachigkeit der Benutzeroberfläche erreicht werden, indem die gewünschte Sprache unter der Eintrag `<label>` definiert wird. Für das Anzeigen von „Roll-over“ (sogenannten Bubble-Help Texten) kann der Parameter `<tooltip>` verwendet werden, etc.

Eine spezielle Möglichkeit um das Aussehen des Darstellungstypen zu verändern, ist die Angabe einer Plugin-Klasse (`<plugin>`). Hiermit wird zur Laufzeit nicht die Standardkomponente für die Visualisierung des Darstellungstypen gewählt, sondern ein eigen erstelltes JavaBean. Das data – binding sowie die Validierung und Darstellung der Instanzdaten bleibt somit Sache dieser Pluginkomponente. Um die Kommunikation zwischen der Rendering Engine und der Pluginkomponente zu ermöglichen, wird ihr ein Java Interface (Visualizer) als Schnittstelle vorgegeben.

Beschreibung der wichtigsten Elemente:

Name	Beschreibung
label	Visueller Name einer Komponente.
choices	Hält eine Menge von möglichen Auswahlmöglichkeiten (<choice>) im Stringformat.
Mode	Ermöglicht es, eine visuelle Komponente auszublenden oder auf readOnly zu schalten.
Layout	Definiert das Layout (default, table oder grid) für eine visuelle Komponente.
PlugIn	Definiert ein Plugin JavaBean, welches anstatt der Standard-Visualisierungskomponente angezeigt werden soll.
Size	Definiert die Grösse eines Eingabefeldes (long, middle oder short) und bestimmt die prozentuale Länge des Labels.
header	Definiert die Spalten-grössen und -überschriften bei ListTypen (SimpleGroupList und ComplexGroupList).

1.5 Behaviour - Definitionen



1.5.1 Navigationstyp

Unter <navigation> wird die Art des Navigierens der betreffenden Komponente und deren Subkomponenten (definiert im XML-Schema) verstanden. Es wird zwischen vier verschiedenen Arten unterschieden:

Name	Beschreibung
Tree	Komponente wird als Knoten im Navigationsbaum angezeigt.
Panel	Komponente wird innerhalb des Objekteditors angezeigt.
Serial	Alle direkten Subkomponenten werden durch eine horizontale Linie hintereinander (seriell) dargestellt.
Tab	Alle direkten Subkomponenten werden als separates Tab angezeigt.

1.5.2 Formelausdruck

Der Wert einer einfachen Komponente (SimpleType) oder eines virtuellen Typen kann aus der Berechnung einer Formel ermittelt werden. Für diesen Zweck dient die Angabe einer `<formulaExpression>`.

Eine `formulaExpression` besteht aus mehreren Variablen und einem Ausdruck (`expression`). Jede Variable (`<variable>`) wird durch eine eindeutige ID und ein XPATH-Statement definiert. Die vergebenen ID's können dann im Formelausdruck (`<expression>`) verwendet werden. Der Formelausdruck lässt alles zu, was einem gültigen XPath-Ausdruck entspricht.

Zum Beispiel:

```
<formulaExpression>
  <variable id="price" xpath="/purchaseOrder/items/item[1]/USPrice"/>
  <variable id="quantity" xpath="/purchaseOrder/items/item[1]/quantity"/>
  <expression>price*quantity/>
</formulaExpression>
```

1.5.3 Regeln

Für die Realisierung von situationsbezogenen Verhalten (Geschäftsregeln – oft auch als Situations-Aktions-Regeln bezeichnet) dienen die `<rule>` Einträge. Zur formalen Beschreibung haben wir uns aus der Datenbanktheorie stammenden Event-Condition-Action (ECA)-Regeln inspirieren lassen. Dies Regeln setzen sich aus drei Komponenten zusammen:

- Event:** Spezifiziert wodurch die Regel ausgelöst wird.
- Condition:** Definiert welche Zustände erfüllt werden müssen.
- Action:** Beschreibt, wie reagiert werden soll.

Situationen werden somit durch Ereignisse und Bedingungen beschrieben. Die auszuführenden Reaktionen sind in den Aktionskomponenten der Regeln spezifiziert, die genau dann verarbeitet werden, wenn die Ereignisse eingetreten und alle Bedingungen erfüllt sind.

XUI kennt jedoch nur eine Art von Ereignismeldung, die von eintretenden Datenänderungen. Die Spezifikation eines Events ist somit hinfällig, da nur die vom Endbenutzer betätigten Datenänderungen über die graphische Benutzeroberfläche Ereignisse auslösen, welche in diesem Kontext interessant sind. Bei der Auslösung einer solchen Ereignismeldung werden alle `<condition>` Einträge für die visuelle Komponente mit dem aktuellen Fokus geprüft. Nur wenn alle Bedingungen erfüllt sind, werden sequentiell alle Aktionen ausgelöst.

Der Rückgabewert einer Bedingung ist demzufolge ein Boolean (wahr oder falsch). Treffen alle Bedingungen einer Regel zu, werden die definierten Aktionen ausgelöst. Es besteht die Möglichkeit, beim Nichteintreffen einer Bedingung, eine Fehlermeldung anzuzeigen. Dies wird über das Attribute `showError="true"` gesteuert.

Jede Bedingung setzt sich aus genau einem Formelausdruck zusammen (Siehe Kapitel 1.5.2).

Es gibt grundsätzlich zwei verschiedene Arten von Aktionen:

Modeländerungen (propertyChange)

Setzt den Wert eines Datenelements neu. Mit der Angabe eines Zieles (**target**) mittels XPath Statement wird auf ein Datenelement eines einfachen Typen verwiesen und somit der neue statische Wert (**anyNewValue**) gesetzt.

Visuelle Ereignisse (UIAction)

Lösen visuelle Aktionen aus (Ein-/Ausblenden von Komponenten, Farbänderungen etc.). Mit der Angabe eines Zieles (**target**) mittels XPath Statement wird auf ein Datenelement und somit dessen Darstellungstypen verwiesen und eine besagte Methode (**methodNameToInvoke**) ausgeführt. Die Definition des Methodennamens richtet sich nach der *Java Beans Definition*.

2 DARSTELLUNGSTYPEN

Der folgende Abschnitt soll einen Regelfindungsprozess aufzeigen, um jedes im XML Schema definiertes Element mittels einer allgemein gültigen visuellen Komponente zu repräsentieren. Ziel dieses Regelfindungsprozess ist es, aufgrund der im XML Schema beschriebenen Regeln eine übersichtliche, gut navigierbare und sinnvolle graphische Benutzeroberfläche zu erstellen, welche das Nachbearbeiten einer XML Instanz vereinfacht (vergleiche Kapitel 5.2.4). Zu diesem Zweck muss es möglich sein, jedem Element im XML Schema einen von mir definierten Darstellungstypen zuzuweisen, um Aussagen über die visuelle Repräsentation zu machen.

Diese Darstellungstypen repräsentieren verschiedene Arten der visuellen Informationspräsentation. Nach einer eingehenden Analyse aller zur Verfügung stehenden graphischen Visualisierungskomponenten aus dem Java Swing Toolkit [Swing 98], gibt es drei grundsätzlich verschiedene Möglichkeiten, Informationen visuell darzustellen.

Einfache, atomare Komponenten für die Darstellung eines einfachen Wertes; gemeint sind primitive Datentypen wie etwa boolean, string, integer etc.
Beispiele dafür sind: JTextField, JCheckBox, JLabel etc.

Gruppen für Komponenten, welche einen Platzhalter (Container) für zusammengehörende Werte visualisieren.
Beispiele sind: JBorder, JTabPane, JScrollPane etc.

Listen für Komponenten, welche eine Ansammlung der selben Werttypen auflistet und das Arbeiten mit den einzelnen Einträgen (erstellen, mutieren & löschen) erlaubt.
Beispiele: JTable, JComboBox, JList etc.
Die definierten Darstellungstypen richten sich nach diesen drei Arten der Informationspräsentation.

Grundsätzlich kennt jedoch die XML Schemaspezifikation nur zwei Arten von Elementtypen, die einfachen (engl. simple) und die komplexen (engl. complex). Ein komplexes Element zeichnet sich dadurch aus, dass es eine Menge von Unterelementen definieren kann. Bei den einfachen Elementen spricht man auch von primitiven (engl. primitive) Datentypen. Diese primitiven Datentypen sind zugleich die Datenbehälter und bilden, weil sie keine Unterelemente definieren können, die Menge aller Blätter (engl. leafs) innerhalb eines XML Baumes.

Für die Zuweisung eines Darstellungstypen gibt es jedoch mehr zu beachten als nur die Differenzierung zwischen einem einfachen und einem komplexen Typen. Zudem muss ein laut XML Schema definiertes, komplexes Element keineswegs komplex sein, um es sinnvoll darstellen zu können. Viel wichtiger ist es, den Darstellungstypen anhand der Topologie eines im XML Schema definierten Elements zu identifizieren. So ist zum Beispiel ein im XML Schema definierter komplexer Type keineswegs für den Darstellungsmechanismus komplex, wenn dieser nur aus primitiven Unterelementen besteht.

Mit der Topologie eines XML Schema Elements verstehe ich die Konstellation der Umgebung, gemeint sind alle direkten Unterelemente (engl. children) und deren Darstellungstypen. Entscheiden dabei ist die Art und die Menge der Darstellungstypen aller Unterelemente.

Durch die Unterscheidung der drei verschiedenen Arten der Informationspräsentation und die Differenzierung ob eine Gruppe nur aus einfachen oder auch aus komplexen Elementen besteht, ergeben sich sechs verschiedene Typen, welche für die Visualisierung von XML Schemaelemente entscheidend sind. Jedem dieser Darstellungstypen wurde ein eigenes Symbol zugeordnet.

-  **SimpleType**
sind die Blätter eines Baumes (engl. leaf nodes) und definieren primitive Datenelemente oder Attribute (boolean, integer, string etc.).
-  **SimpleGroup**
beinhalten eine fest definierte Menge von einfachen Typen (SimpleTypes) oder einfachen Kollektionen (SimpleTypeList) oder anderen einfachen Gruppen (SimpleGroup).
-  **ComplexGroup**
ist dann komplex, wenn sie eine oder mehrere andere komplexe Gruppen beinhaltet oder mindestens eine einfache Gruppe oder Kollektion in Kombination mit beliebig vielen einfachen Typen (SimpleType) umfasst.
-  **SimpleTypeList**
ist eine Kollektion des gleichen einfachen Typen mit einer Kardinalität grösser als eins.
-  **SimpleGroupList**
ist eine Kollektion einer einfachen Gruppe (SimpleGroup) mit einer Kardinalität grösser als eins.
-  **ComplexGroupList**
ist eine Kollektion einer komplexen Gruppe (ComplexGroup) mit einer Kardinalität grösser als eins.

3 STARTEN DES GUITESTERS

Mit dem GUITester können W3C XML Schema's generisch visulisiert werden. Als einziges zwingendes Argument muss ein XML Schema URL übergeben werden. Optional dazu kann eine XML Instanz und/oder eine XUI-Definition mitgegeben werden.

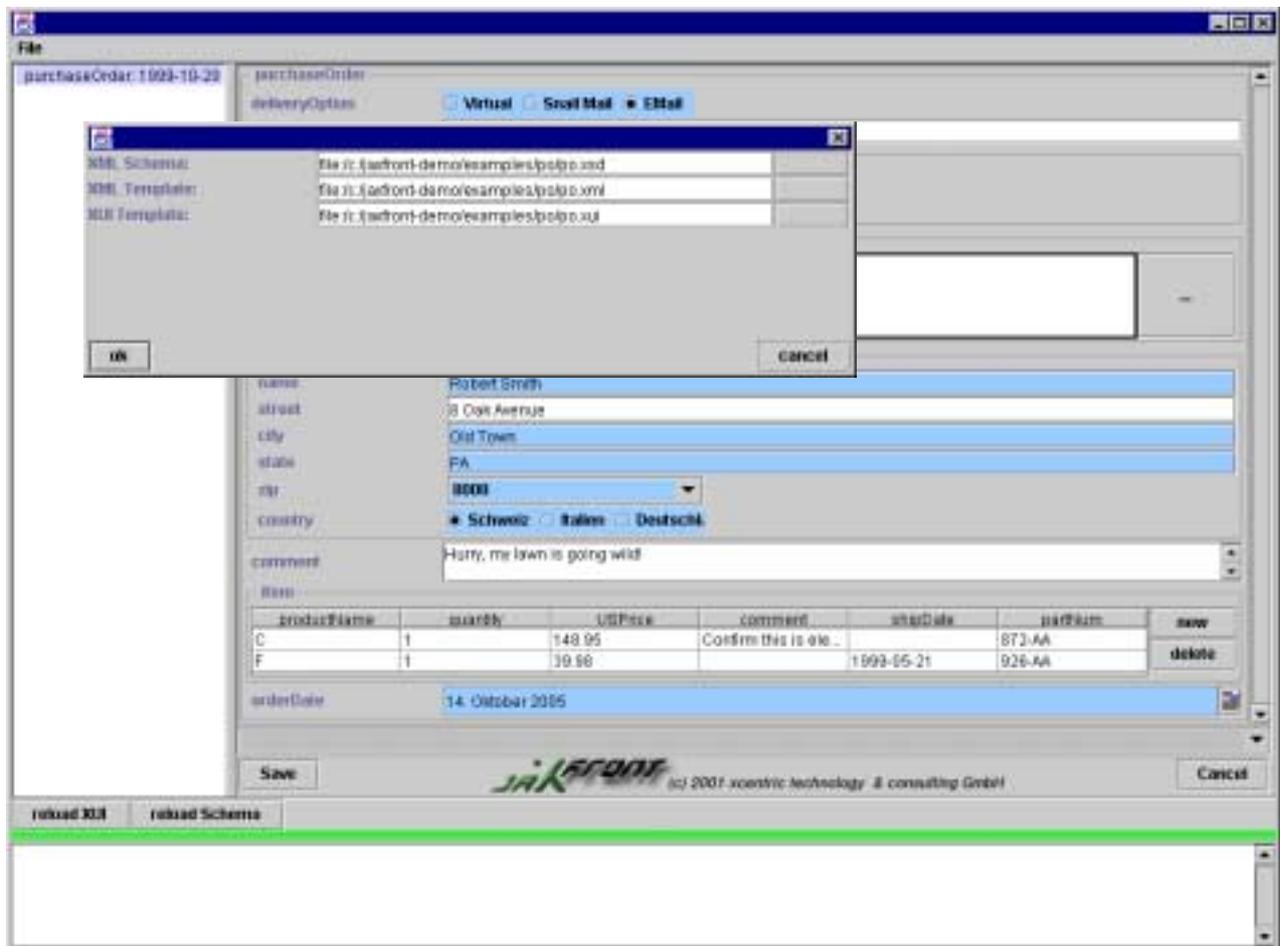
Usage: java com.jaxfront.ui.GUITester <xsd-url> {-XUI=<xui-url> -XML=<xml-url>}

xsd-url = a valid XML Schema (W3C Recommendation, 2 May 2001) URL

xui-url = a valid XUI document URL

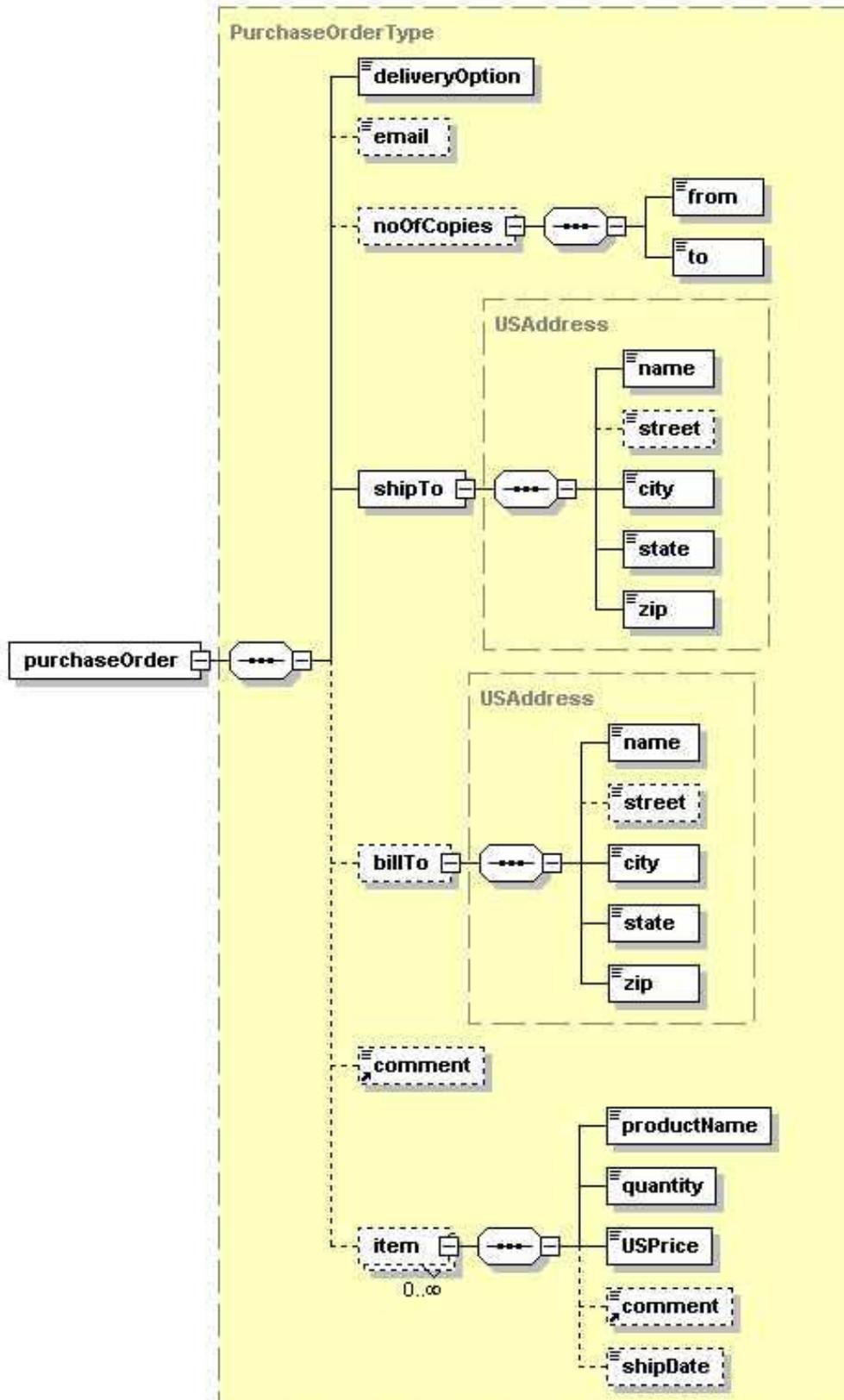
xml-url = a valid XML document URL

Das XML-Schema, die XML-Instanz und die XUI-Definition können nach dem Start über den Menüpunkt (File/open...) geändert werden.



4 EIN BEISPIEL: PURCHASE ORDER

4.1 Das XML-Schema (po.xsd)



4.2 Die XML-Instanz (po.xml)

```
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-10-20">
  <email>stephan.portmann@xcentric.ch</email>
  <noOfCopies>
    <from>1</from>
    <to>5</to>
  </noOfCopies>
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>
  <billTo country="US">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <city>Old Town</city>
    <state>PA</state>
    <zip>95819</zip>
  </billTo>
  <comment>Hurry, my lawn is going wild!</comment>
  <item partNum="872-AA">
    <productName>C</productName>
    <quantity>1</quantity>
    <USPrice>148.95</USPrice>
    <comment>Confirm this is electric</comment>
  </item>
  <item partNum="926-AA">
    <productName>F</productName>
    <quantity>1</quantity>
    <USPrice>39.98</USPrice>
    <shipDate>1999-05-21</shipDate>
  </item>
</purchaseOrder>
```

4.3 Die XUI-Definition (po.xui)

```
<?xml version="1.0" encoding="utf-8"?>
<UserInterfaceSpecification xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="xui.xsd"
  verticalScrollBarPolicy="VERTICAL_SCROLLBAR_ALWAYS"
  horizontalScrollBarPolicy="HORIZONTAL_SCROLLBAR_AS_NEEDED"
  useCharacterIndent="2"
  allowSavingWithErrors="false"
  versionNo="1.0">

  <component xpath="/purchaseOrder/">
    <style>
      <mode useNavigationTree="true"/>
    </style>
    <behaviour>
      <navigation type="Panel"/>
    </behaviour>
  </component>
  <component xpath="/purchaseOrder/@orderDate">
    <style/>
    <behaviour>
      <navigation type="Panel"/>
      <rule>
        <event type="propertyChange"/>
      </rule>
    </behaviour>
  </component>
```

```
        <condition showError="true">
            <formulaExpression expression="$noFutureDate"/>
        </condition>
    </rule>
</behaviour>
</component>
<component xpath="/purchaseOrder/comment">
    <style>
        <size labelSpace="20" type="multi"/>
    </style>
    <behaviour>
        <navigation type="Panel"/>
    </behaviour>
</component>
<component xpath="/purchaseOrder/noOfCopies/from">
    <style>
        <size labelSpace="20" type="short"/>
    </style>
    <behaviour>
        <navigation type="Panel"/>
        <rule>
            <event type="propertyChange"/>
            <condition showError="true">
                <formulaExpression expression=". &gt; $copiesTo">
                    <variable id="$copiesTo" xpath="/purchaseOrder/noOfCopies/to"/>
                </formulaExpression>
            </condition>
        </rule>
    </behaviour>
</component>
<component xpath="/purchaseOrder/noOfCopies/to">
    <style>
        <size labelSpace="20" type="short"/>
    </style>
    <behaviour>
        <navigation type="Panel"/>
    </behaviour>
</component>
<component xpath="/purchaseOrder/shipTo/street">
    <style>
        <size labelSpace="20" type="middle"/>
    </style>
    <behaviour>
        <navigation type="Panel"/>
    </behaviour>
</component>
<component xpath="/purchaseOrder/shipTo/city">
    <style>
        <size labelSpace="20" type="middle"/>
    </style>
    <behaviour>
        <navigation type="Panel"/>
    </behaviour>
</component>
<component xpath="/purchaseOrder/shipTo/state">
    <style>
        <size labelSpace="20" type="short"/>
    </style>
    <behaviour>
        <navigation type="Panel"/>
    </behaviour>
</component>
<component xpath="/purchaseOrder/shipTo/zip">
    <style>
        <size labelSpace="20" type="short"/>
        <choices orientation="horizontal" selection="8000" type="comboBox">
            <choice>8000</choice>
            <choice>9900</choice>
        </choices>
    </style>
</component>
```

```
        <choice>5600</choice>
    </choices>
    <choice>1200</choice>

</style>

<behaviour>
    <navigation type="Panel"/>
    <rule>
        <event type="propertyChange"/>
        <condition showError="true">
            <formulaExpression expression=". > 1200"/>
        </condition>
    </rule>
</behaviour>
</component>
<component xpath="/purchaseOrder/shipTo/name">
    <style>
        <size labelSpace="20" type="short"/>
    </style>
    <behaviour>
        <navigation type="Tab"/>
        <rule>
            <event type="propertyChange"/>
            <condition showError="false" useToggle="true">
                <formulaExpression expression=". = &quot;Microsoft&quot;"/>
            </condition>
            <action target="/purchaseOrder/billTo/name">
                <propertyChange>
                    <formulaExpression expression="The Devil's Home"/>
                </propertyChange>
            </action>
            <action target="/purchaseOrder/billTo/name">
                <uiAction methodName="setBackground">
                    <param name="bcolor" type="Color" value="50,60,70"/>
                </uiAction>
            </action>
            <action target="/purchaseOrder/billTo/name">
                <uiAction methodName="setForeground">
                    <param name="fcolor" type="Color" value="240,60,230"/>
                </uiAction>
            </action>
            <action target="/purchaseOrder/billTo/street">
                <uiAction methodName="setVisible">
                    <param name="streetVisibility" type="boolean" value="false"/>
                </uiAction>
            </action>
        </rule>
    </behaviour>
</component>
<component xpath="/purchaseOrder/deliveryOption">
    <style>
        <choices codeset="deliveryOption" selection="E" type="radioBox"/>
        <!--mode visible="false"/-->
    </style>
    <behaviour>
        <navigation type="Panel"/>
        <rule>
            <event type="propertyChange"/>
            <condition showError="false">
                <formulaExpression expression=". = &quot;E&quot;"/>
            </condition>
            <action target="/purchaseOrder/email">
                <uiAction methodName="setVisible">
                    <param name="visible" type="boolean" value="true"/>
                </uiAction>
            </action>
        </rule>
    </behaviour>
</component>
```

```
</behaviour>
</component>
<component xpath="/purchaseOrder/shipTo/@country">
  <style>
    <size labelSpace="20" type="short"/>
    <choices codeset="countries" selection="S" type="radioBox"/>
    <!--mode visible="false"/-->
  </style>

  <behaviour>

    <navigation type="Panel"/>
    <rule>
      <event type="propertyChange"/>
      <action target="/purchaseOrder/billTo/@country">
        <propertyChange>
          <formulaExpression expression="."/>
        </propertyChange>
      </action>
    </rule>
  </behaviour>
</component>
<component xpath="/purchaseOrder/billTo/@country">
  <style>
    <size labelSpace="20" type="short"/>
    <choices codeset="countries" selection="S" type="radioBox"/>
    <!--mode visible="false"/-->
  </style>
  <behaviour>
    <navigation type="Panel"/>
    <rule>
      <event type="propertyChange"/>
      <action target="/purchaseOrder/shipTo/country">
        <propertyChange>
          <formulaExpression expression="."/>
        </propertyChange>
      </action>
    </rule>
  </behaviour>
</component>
<component xpath="/purchaseOrder/noOfCopies/total">
  <style>
    <mode readonly="true"/>
    <size labelSpace="40" type="short"/>
  </style>
  <behaviour>
    <navigation type="Panel"/>
  </behaviour>
</component>
<component xpath="/purchaseOrder/billTo/zip">
  <style>
    <size labelSpace="20" type="short"/>
    <choices orientation="horizontal" selection="8000" type="comboBox">
      <choice>8000</choice>
      <choice>9900</choice>
      <choice>1200</choice>
      <choice>5600</choice>
    </choices>
  </style>
  <behaviour>
    <navigation type="Panel"/>
  </behaviour>
</component>
<component xpath="/purchaseOrder/item/total">
  <style/>
  <behaviour>
```

```
        <formulaExpression expression="/purchaseOrder/items/item/USPrice * /purchaseOrder/items/item/quantity"/>
    </behaviour>
</component>
<component xpath="/purchaseOrder/item">
    <style>
        <header direction="AS_COLUMN" showSequenceNumbering="true" defaultWidth="100">
            <lineInfo name="productName" xpath="/productName" width="150"/>
            <lineInfo name="quantity" xpath="/quantity" width="70"/>
            <lineInfo name="price" xpath="/USPrice" width="80"/>
            <lineInfo name="total" xpath="/quantity * ./USPrice" isExpression="true" width="90"/>
            <lineInfo name="shipDate" xpath="/shipDate"/>
        </header>
    </style>
</behaviour>
```

```
<navigation type="Panel"/>
```

```
</behaviour>
</component>
<component xpath="/purchaseOrder/item/productName">
    <style>
        <choices codeset="products" type="comboBox"/>
    </style>
    <behaviour>
        <navigation type="Panel"/>
    </behaviour>
</component>
<component xpath="/purchaseOrder/shipTo">
    <style>
        <plugIn class="com.jaxfront.demo.ui.beans.USAddressBean"/>
    </style>
</component>
<virtualComponent id="$Sum">
    <style>
        <label>Total USPrice in $</label>
        <mode readonly="true"/>
        <size labelSpace="80" type="long"/>
    </style>
    <behaviour>
        <navigation type="Panel"/>
        <formulaExpression expression="sum(/purchaseOrder/items/item/USPrice)"/>
    </behaviour>
</virtualComponent>
<virtualComponent id="$Sum_Quantity">
    <style>
        <label>Total Items</label>
        <mode readonly="true"/>
        <size labelSpace="80" type="long"/>
    </style>
    <behaviour>
        <navigation type="Panel"/>
        <rule>
            <event type="propertyChange"/>
        </rule>
        <formulaExpression expression="sum(/purchaseOrder/items/item/quantity)"/>
    </behaviour>
</virtualComponent>
</UserInterfaceSpecification>
```

5 XUI SCHEMA SPEZIFIKATION (XUI.XSD)

Elements

- [action](#)
- [behaviour](#)
- [choices](#)
- [component](#)
- [formulaExpression](#)
- [label](#)
- [mode](#)
- [navigation](#)
- [param](#)
- [refVirtualComponent](#)
- [size](#)
- [style](#)
- [table](#)
- [uiAction](#)
- [UserInterfaceSpecification](#)
- [variable](#)
- [virtualComponent](#)

element action

diagram					
children	propertyChange uiAction structureChange				
used by	elements	style/plugin behaviour/rule			
attributes	Name	Type	Use	Default	Fixed
	target	xsd:string	required		
annotation	documentation	Definiert eine Aktion, welche zur Laufzeit ausgeführt wird.			

element action/propertyChange

diagram					
children	formulaExpression				
annotation	documentation	Definiert eine Aktion welche eine Modelländerung ausführt.			

element **action/structureChange**

diagram	<div style="border: 1px solid black; padding: 5px; display: inline-block;">structureChange</div> Definiert eine Aktion welche eine Strukturänderung des darunterliegenden Instanzbaumes durchführt.				
attributes	Name type	Type xsd:string	Use required	Default	Fixed
annotation	documentation	Definiert eine Aktion welche eine Strukturänderung des darunterliegenden Instanzbaumes durchführt.			

element **behaviour**

diagram	<p>The diagram shows a class behaviour with a composition relationship (indicated by a solid line with a filled square at the 'behaviour' end) to a container class (indicated by a dashed line with a filled square at the container end). Inside this container, there are three subclasses: navigation, rule, and formulaExpression. Each subclass has a descriptive text box next to it.</p>				
children	navigation rule formulaExpression				
used by	elements	component virtualComponent			
annotation	documentation	Definieren verhaltensspezifische Eigenschaften einer Komponente.			

element behaviour/rule

diagram	<p>rule Für die Realisierung von situationsbezogenen Verhalten (Geschäftsregeln – oft auch als Situations-Aktions-Regeln bezeichnet) dienen diese Regeleinträge. Jede Regel besteht aus Event-Condition-Action (ECA).</p> <p>event Spezifiziert wodurch die Regel ausgelöst wird.</p> <p>condition 0..∞ Definiert welche Zustände erfüllt werden müssen.</p> <p>action 0..∞ Definiert eine Aktion, welche zur Laufzeit ausgeführt wird.</p>				
children	event condition action				
annotation	documentation	Für die Realisierung von situationsbezogenen Verhalten (Geschäftsregeln – oft auch als Situations-Aktions-Regeln bezeichnet) dienen diese Regeleinträge. Jede Regel besteht aus Event-Condition-Action (ECA).			

element behaviour/rule/event

diagram	<p>event Spezifiziert wodurch die Regel ausgelöst wird.</p>				
attributes	Name	Type	Use	Default	Fixed
	type	xsd:string	required	propertyChange	
annotation	documentation	Spezifiziert wodurch die Regel ausgelöst wird.			

element behaviour/rule/condition

diagram	<p>condition Definiert welche Zustände erfüllt werden müssen.</p> <p>formulaExpression Besteht aus mehreren Variablen und einem Ausdruck (expression). Jede Variable wird durch eine eindeutige ID und ein XPATH-Statement definiert. Die vergebenen ID's können dann im Formel Ausdruck (expression) verwendet werden. Der Formel Ausdruck lässt alles zu, was einem gültigen XPath-Ausdruck entspricht.</p>				
children	formulaExpression				
attributes	Name	Type	Use	Default	Fixed
	showError	xsd:boolean	optional		
	useToggle	xsd:boolean	optional		
annotation	documentation	Definiert welche Zustände erfüllt werden müssen.			

element choices

diagram	<p>Somit kann der Wert eines Blattelements (SimpleType) über eine Auswahl getroffen werden.</p>				
children	choice				
used by	element style				
attributes	Name	Type	Use	Default	Fixed
	codeset	xsd:string	optional		
	type	xsd:string	optional	comboBox	
	orientation	xsd:string	optional	horizontal	
	selection	xsd:string	optional		
annotation	documentation	Somit kann der Wert eines Blattelements (SimpleType) über eine Auswahl getroffen werden.			

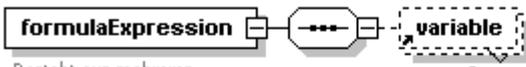
element choices/choice

diagram					
type	xsd:string				

element component

diagram	<p>Definiert (der) in der XML-Instanz vorzufindende(n) Knoten.</p> <p>style + Definiert layout-spezifische Einstellungen einer Komponente.</p> <p>behaviour + Definieren verhaltensspezifische Eigenschaften einer Komponente.</p>				
children	style behaviour				
used by	element UserInterfaceSpecification				
attributes	Name	Type	Use	Default	Fixed
	xpath	xsd:string	required		
annotation	documentation	Definiert (der) in der XML-Instanz vorzufindende(n) Knoten.			

element formulaExpression

diagram	 <p>Besteht aus mehreren Variablen und einem Ausdruck (expression). Jede Variable wird durch eine eindeutige ID und ein XPATH-Statement definiert. Die vergebenen ID's können dann im Formelausdruck (expression) verwendet werden. Der Formelausdruck lässt alles zu, was einem gültigen XPath-Ausdruck entspricht.</p> <p>0..∞ Eine Variable weist einem XPath einen vordefinierten Alias (id) zu.</p>				
children	variable				
used by	elements behaviour behaviour/rule/condition action/propertyChange				
attributes	Name	Type	Use	Default	Fixed
	expression	xsd:string	required		
annotation	documentation	Besteht aus mehreren Variablen und einem Ausdruck (expression). Jede Variable wird durch eine eindeutige ID und ein XPATH-Statement definiert. Die vergebenen ID's können dann im Formelausdruck (expression) verwendet werden. Der Formelausdruck lässt alles zu, was einem gültigen XPath-Ausdruck entspricht.			

element label

diagram	 <p>Definiert die visuell repräsentierbare Bezeichnung einer Komponente.</p>				
type	xsd:string				
used by	element style				
annotation	documentation	Definiert die visuell repräsentierbare Bezeichnung einer Komponente.			

element mode

diagram	 <p>Definiert Einstellungen bezüglich Sichtbarkeit der Komponente.</p>				
used by	element style				
attributes	Name	Type	Use	Default	Fixed
	readonly	xsd:boolean	optional		
	visible	xsd:boolean	optional		
	dateMode	xsd:string	optional		
	useNavigationTree	xsd:boolean	optional	true	
	mandatory	xsd:boolean	optional		
annotation	documentation	Definiert Einstellungen bezüglich Sichtbarkeit der Komponente.			

element navigation

diagram	 <p>Unter navigation wird die Art des Navigierens der betreffenden Komponente und deren Subkomponenten (definiert im XML-Schema) verstanden. Es wird zwischen vier verschiedenen Arten unterschieden.</p>				
used by	element behaviour				
attributes	Name	Type	Use	Default	Fixed
	type	xsd:string	required	Panel	
annotation	documentation	Unter navigation wird die Art des Navigierens der betreffenden Komponente und deren Subkomponenten (definiert im XML-Schema) verstanden. Es wird zwischen vier verschiedenen Arten unterschieden.			

element param

diagram	 <p>Definiert einen Übergabeparamter bestehend aus einem alias (name), type un einem Wert (value), welcher zur Laufzeit verwendet wird.</p>				
type	extension of xsd:string				
used by	elements style/plugin uiAction				
attributes	Name	Type	Use	Default	Fixed
	name	xsd:string	required		
	type	xsd:string	required		
	value	xsd:string	required		
annotation	documentation	Definiert einen Übergabeparamter bestehend aus einem alias (name), type un einem Wert (value), welcher zur Laufzeit verwendet wird.			

element refVirtualComponent

diagram	 <p>Definiert eine Referenz auf eine virtuelle Komponente.</p>				
used by	element table				
attributes	Name	Type	Use	Default	Fixed
	idRef	xsd:string	required		
	x_location	xsd:int	required	1	
	y_location	xsd:int	required	1	
	coll_span	xsd:int		1	
	row_span	xsd:int		1	
	options	xsd:string			
annotation	documentation	Definiert eine Referenz auf eine virtuelle Komponente.			

element size

diagram	 <p>Definiert die Größe der Komponente.</p>				
---------	--	--	--	--	--

used by	element style				
attributes	Name	Type	Use	Default	Fixed
	labelSpace	xsd:integer	required	20	
annotation	documentation	Definiert die Grösse der Komponente.			
	type	xsd:string	required	long	

element **style**

diagram	<p>The diagram shows a central 'style' element box connected to a larger container box. This container box is linked to seven child elements: 'label', 'choices', 'mode', 'layout', 'plugin', 'size', and 'header'. Each child element has a brief description of its function.</p>
children	label choices mode layout plugin size header
used by	elements component virtualComponent
annotation	documentation Definiert layout-spezifische Einstellungen einer Komponente.

element **style/layout**

diagram	<p>layout Definiert die bevorzugte Darstellungsart.</p> <p>table Definiert, dass alle Komponenten unterhalb dieser mit einem TableLayout definiert werden.</p> <p>default Definiert das Standardlayout jeder Komponente.</p> <p>grid Definiert ein Layout namens grid. Ein grid wird in verschiedene Bereiche (area) aufgeteilt, welche von sogenannten snapLines definiert werden.</p>
children	table default grid
annotation	documentation Definiert die bevorzugte Darstellungsart.

element **style/layout/default**

diagram	<p>default Definiert das Standardlayout jeder Komponente.</p> <p>virtualComponentPosition 1..∞</p>
children	virtualComponentPosition
annotation	documentation Definiert das Standardlayout jeder Komponente.

element **style/layout/default/virtualComponentPosition**

diagram	virtualComponentPosition				
attributes	Name	Type	Use	Default	Fixed
	idRef	xsd:string	required		
	position	xsd:string	required	after	
	xpath	xsd:string	optional		

element **style/layout/grid**

diagram	<p>grid Definiert ein Layout namens grid. Ein grid wird in verschiedene Bereiche (area) aufgeteilt, welche von sogenannten snapLines definiert werden.</p> <p>snapLine 1..∞ Jede snapLine bildet eine Unterteilung des Bildschirms, auf die eine area Bezug nehmen kann.</p> <p>area 1..∞ Eine Area definiert einen Bereich, in welchen eine Komponente platziert werden kann.</p>
children	snapLine area

annotation	documentation	Definiert ein Layout namens grid. Ein grid wird in verschiedene Bereiche (area) aufgeteilt, welche von sogenannten snapLines definiert werden.
------------	---------------	--

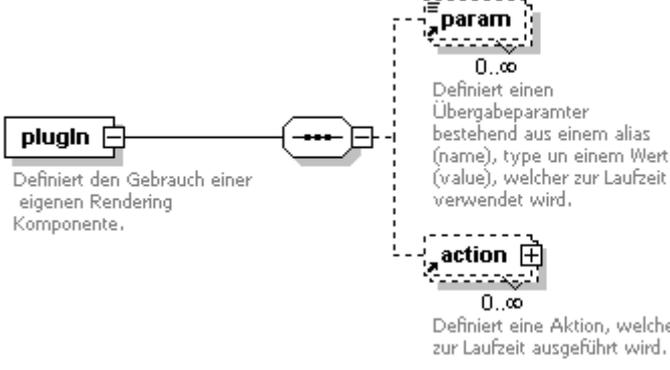
element **style/layout/grid/snapLine**

diagram					
attributes	Name	Type	Use	Default	Fixed
	name	xsd:string	required		
	margin	xsd:int	optional	0	
	position	xsd:float	required	20	
	orientation	xsd:string	required	h	
	space	xsd:int	optional	0	
annotation	documentation	Jede snapLine bildet eine Unterteilung des Bildschirms, auf die eine area Bezug nehmen kann.			

element **style/layout/grid/area**

diagram					
attributes	Name	Type	Use	Default	Fixed
	xpath	xsd:string	optional		
	label	xsd:string	optional		
	useSeparator	xsd:boolean	optional	false	
	name	xsd:string	required		
	top	xsd:string	required	TOP	
	bottom	xsd:string	required	BOTTOM	
	left	xsd:string	required	LEFT	
	right	xsd:string	required	RIGHT	
annotation	documentation	Eine Area definiert einen Bereich, in welchen eine Komponente platziert werden kann.			

element **style/plugin**

diagram					
children	param action				
attributes	Name	Type	Use	Default	Fixed
	class	xsd:string	required		
annotation	documentation	Definiert den Gebrauch einer eigenen Rendering Komponente.			

element **style/header**

diagram					
children	lineInfo				
attributes	Name	Type	Use	Default	Fixed
	direction	xsd:string	required	AS_ROW	
	showSequenceNumbering	xsd:boolean	optional	true	
	columnWidth	xsd:int	optional	100	
	tableHeight	xsd:int	optional	30	
	columnHeaderXPath	xsd:string	optional		
	isExpression	xsd:boolean	optional	false	
	allowTableFlippingOnTheFly	xsd:boolean	optional	true	
	allowExportToExcel	xsd:boolean	optional	true	
annotation	documentation	Definiert die Spaltengrößen und -überschriften bei ListTypen (SimpleGroupList und ComplexGroupList).			

element **style/header/lineInfo**

diagram					
attributes	Name	Type	Use	Default	Fixed
	name	xsd:string	required		
	xpath	xsd:string	required		
	isExpression	xsd:boolean	optional	false	
	width	xsd:int	optional	100	
annotation	documentation	Jede LineInfo beschreibt eine Kolonne in der Tabelle.			

element **table**

diagram					
children	refComponent stringComponent refVirtualComponent				
used by	element	style/layout			
annotation	documentation	Definiert, dass alle Komponenten unterhalb dieser mit einem TableLayout definiert werden.			

element **table/refComponent**

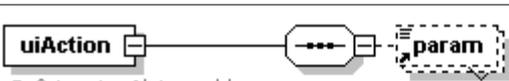
diagram					
attributes	Name	Type	Use	Default	Fixed

	xpath	xsd:string	required	
	x_location	xsd:int	required	
	y_location	xsd:int	required	
	coll_span	xsd:int		1
	row_span	xsd:int		1
	orientation	xsd:string		
	options	xsd:string		
	show_label	xsd:boolean		

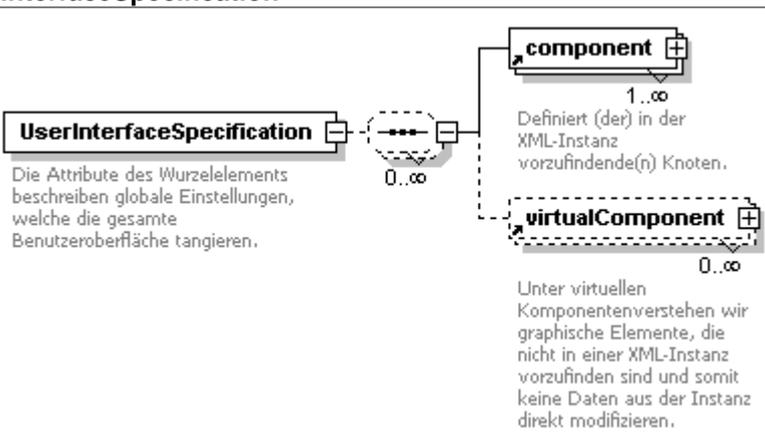
element **table/stringComponent**

diagram					
attributes	Name	Type	Use	Default	Fixed
	label	xsd:string	required		
	x_location	xsd:int	required	1	
	y_location	xsd:int	required	1	
	coll_span	xsd:int		1	
	row_span	xsd:int		1	
	options	xsd:string			

element **uiAction**

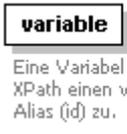
diagram					
children	param				
used by	element action				
attributes	Name	Type	Use	Default	Fixed
	methodName	xsd:string	required		
annotation	documentation	Definiert eine Aktion welche über eine Methode aufgerufen wird.			

element **UserInterfaceSpecification**

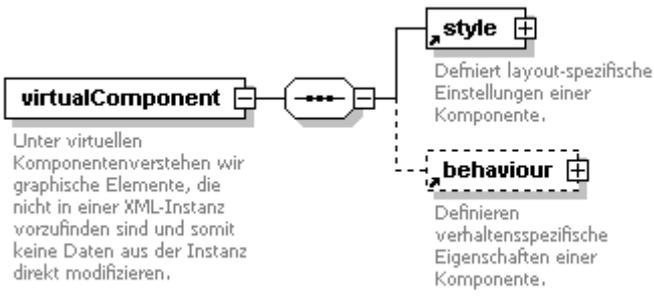
diagram					
children	component virtualComponent				
attributes	Name	Type	Use	Default	Fixed
	useStatusBar	xsd:boolean	optional	true	
	useButtonBar	xsd:boolean	optional	true	
	usePlugins	xsd:boolean	optional	true	
	systemExitOnClose	xsd:boolean	optional	true	
	useBackwardButton	xsd:boolean	optional	false	
	useForwardButton	xsd:boolean	optional	false	
	verticalScrollBarPolic	xsd:string	optional	VERTICAL_SCROLL	

	y horizontalScrollBarPo licy useCharacterIndent allowSavingWithError s versionNo	xsd:string xsd:integer xsd:boolean xsd:string	optional optional optional required	BAR_AS_NEEDED HORIZONTAL_SCR OLLBAR_AS_NEED ED 1 true
annotation	documentation	Die Attribute des Wurzelements beschreiben globale Einstellungen, welche die gesamte Benutzeroberfläche tangieren.		

element variable

diagram					
used by	element	formulaExpression			
attributes	Name	Type	Use	Default	Fixed
	id	xsd:string	required		
	xpath	xsd:string	required		
annotation	documentation	Eine Variabel weist einem XPath einen vordefinierten Alias (id) zu.			

element virtualComponent

diagram					
children	style behaviour				
used by	element	UserInterfaceSpecification			
attributes	Name	Type	Use	Default	Fixed
	id	xsd:string	required		
annotation	documentation	Unter virtuellen Komponentenverstehen wir graphische Elemente, die nicht in einer XML-Instanz vorzufinden sind und somit keine Daten aus der Instanz direkt modifizieren.			