Diplomverteidigung

Realisierung von Interaktionen und Verhalten in dokumentbestimmten, komponentenbasierten 3D-Applikationen

Betreuer: Dipl.-Inform. R. Dachselt
Betreuender Hochschullehrer: Prof. Dr.-Ing. K. Meißner

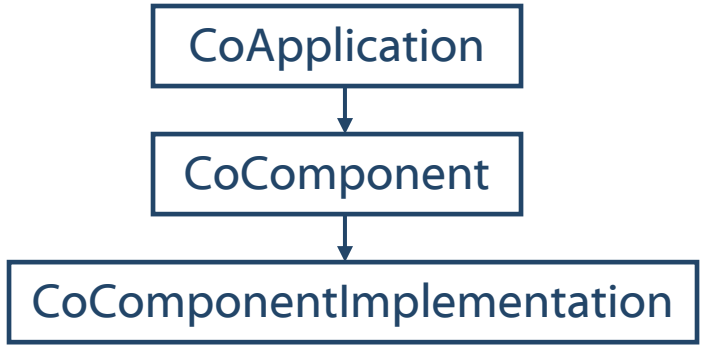
Enrico Rukzio
Heinz - Nixdorf - Stiftungslehrstuhl für Multimediatechnik
Institut für Software- und Multimediatechnik
Fakultät Informatik
Technische Universität Dresden

Gliederung

- Einführung
  - Contigra-Projekt
  - Aufgabenstellung der Diplomarbeit
  - Begriffsklärung
- Behavior3D
  - Analyseergebnisse
  - Konzept und Realisierung
  - Einbindung in Contigra
- Übersetzung in konkrete 3D-Formate
- Verhaltensmodellierung im ContigraBuilder
- Zusammenfassung und Ausblick

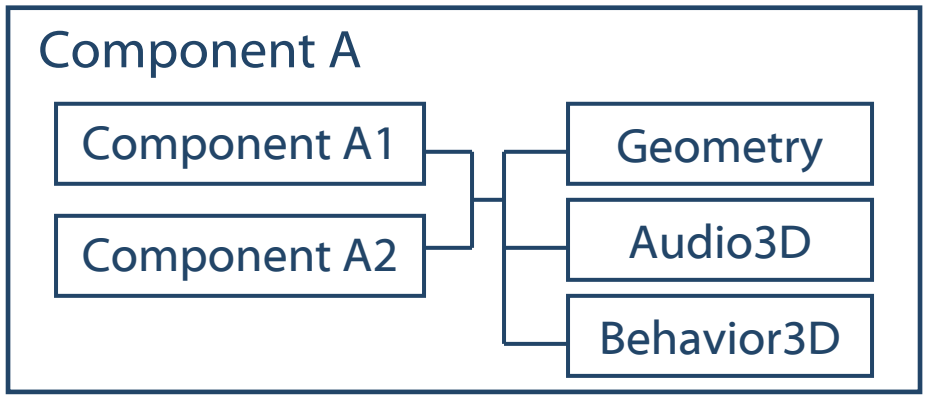
Einführung
Contigra-Projekt

- Einfache Komponenten-basierte Erstellung interaktiver Web3D-Applikationen
- Bibliothek von Contigra-Komponenten (Slider, Button)
- XML-basierte Beschreibung der Applikation, der Komponentenschnittstelle und der Komponenten-implementierung



Contigra-Dokumentenarchitektur

Einführung
Aufgabenstellung der Diplomarbeit



Contigra-Komponente

- Stand der Technik und Forschungsergebnisse bei der deklarativen Verhaltensmodellierung
- Behavior3D: Format zur deklarativen Beschreibung des Verhaltens von Komponenten mittels wiederverwendbarer Verhaltensbausteine
- Übersetzung in konkrete 3D-Formate
- Verhaltensmodellierung im ContigraBuilder

Einführung
Begriffsklärung

- Szenengraph
  - Knoten beschreiben Geometrie-, Audio- und Verhaltensobjekte und deren Eigenschaften (Felder)
- Ereignis
  - Knoten können Ereignisse auslösen und empfangen
  - Ereignis = Erzeuger + Entstehungszeitpunkt + Datenwert
  - Durch Benutzerinteraktionen oder Zeitabhängigkeiten ausgelöst
- Ereignispfad
  - Ereignis-basierte Verknüpfung von Knoten

Verknüpfung von 2 Knoten

Behavior3D
Analyse Verhaltensmodellierung

- Relevante multimediale Formate und Forschungsergebnisse auf dem Gebiet der deklarativen Verhaltensmodellierung
- VRML97/X3D
  - Deklaratives, standardisiertes Beschreibungsformat für interaktive Web3D-Applikationen
- VRML++ [Diehl 1997, Diehl 1998]
  - Erweiterung von VRML97 um objektorientierte Konzepte
- SMIL 2.0
  - Intuitives Animations- und Zeitkonzept
  - Synchronisations- und Gruppierungskonzept
- Viewpoint
  - Deklarative Verhaltensmodellierung auf Basis von Zustandsautomaten

Behavior3D
Notwendigkeit und Ziele

- Deklaratives Format für alle Verhaltensarten
- Erweiterbarkeit, Vollständigkeit
  - Erstellung und Verwendung von Behavior3D-Knoten
- Objektorientierung
  - Wartung, Stabilität, Polymorphie, Typisierung
- Wiederverwendung
  - Vererbung, Komposition
- Trennung von Geometrie, Audio und Verhalten
- Strukturierung, Lesbarkeit
  - Gruppierung von Behavior3D-Knoten

Behavior3D
Konzept

- Alternativen
  - VRML97/X3D (Erweiterbarkeit, Objektorientierung, Trennung)
  - Völlig neues Konzept (Implementierung, Konvertierung, Evaluation)
- Lösung
  - Behavior3D baut auf X3D auf
  - Verbesserung der erkannten Problembereiche (Einheitliches Knotenkonzept, Verbesserte Schnittstellendefinition)
  - Einbeziehung aktueller Forschungsergebnisse
  - Berücksichtigung der Analyse der anderen Formate
  - Weiterentwicklung von X3D-Verhaltensaspekten
- Behavior3D
  - Dynamische XML-Schema-Grammatik (Behavior3DNodes)
  - Interpretation/Implementierung der Behavior3D-Knoten

Behavior3D
Behavior3D-Knoten

- Existierende Behavior3D-Knoten
  - Integration existierender X3D-Verhaltenskomponenten (Environmental Sensor, Event Utilities, Interpolation, Pointing Device Sensor, Scripting)
  - Neue Behavior3D-Knoten
- Erstellung von neuen Verhaltensknoten
  - VRML97/X3D: Built-In-Knoten, Prototypen, Script-Knoten, Klassen (VRML++)
  - Behavior3D: Ein einheitliches Konzept zur Erstellung und Verwendung von Verhaltensknoten
- Schritte zur Erstellung eines Behavior3D-Knotens

  (1) Abstrakte Beschreibung der Knotenschnittstelle

  (2) Generierung von Dokumenten

  (3) Implementierung des Verhaltens

  (4) Einbinden des neuen Knotens in Behavior3DNodes

Behavior3D
(1) Abstrakte Beschreibung der Knotenschnittstelle

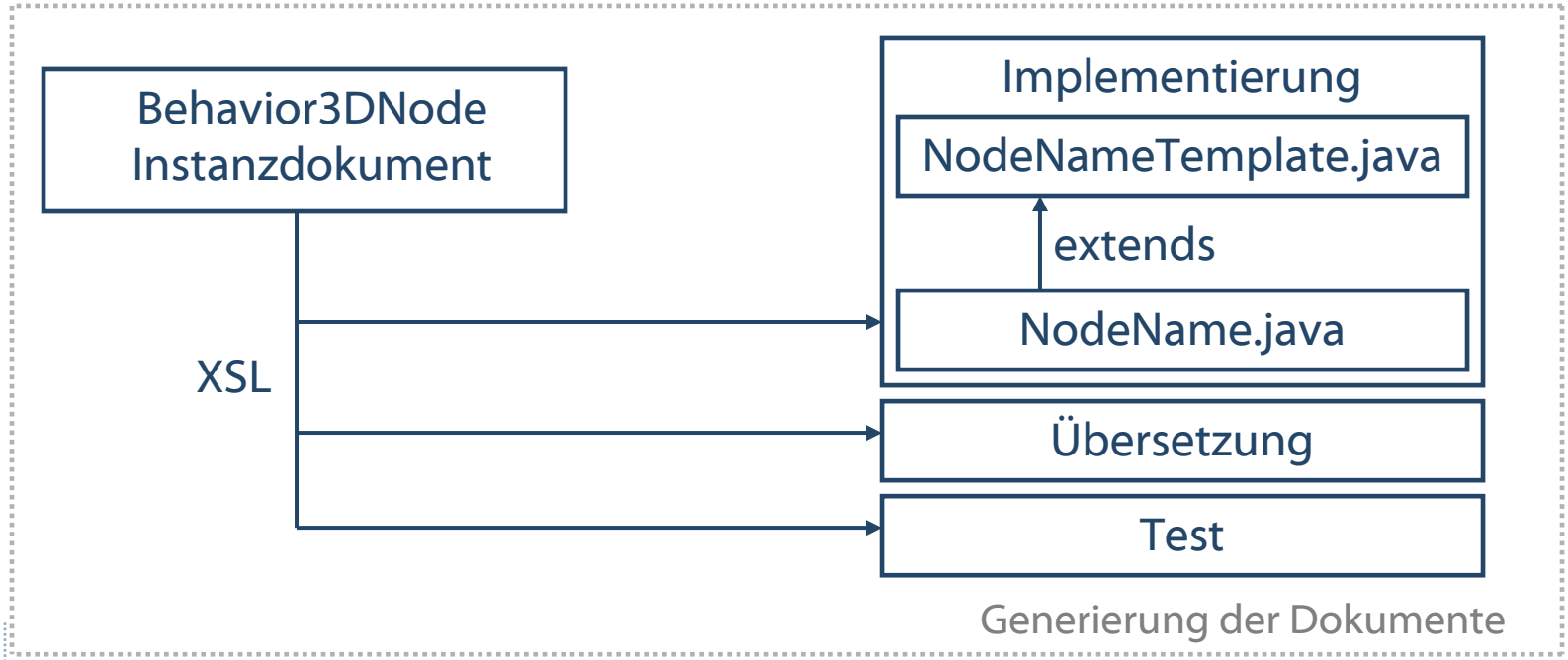
- XML-Schema-Grammatik Behavior3DNode
  - Header
  - Interface

```
<Behavior3DNode >
 <Header name="TimeContainer"  component="Animation" profile="Interactive"/>
 <Interface nodeType="abstract" extends="AnimationTimeSensor">
  <Fields>
   <Field name="children" dataType="AnimationTimeSensors" default="NULL">
    <ChangeMode configurable="true" receivesEvents="false" generatesEvents="false"/>
   </Field>
  </Fields>
 </Interface>
</Behavior3DNode>
```

Deklaration der Knotenschnittstelle

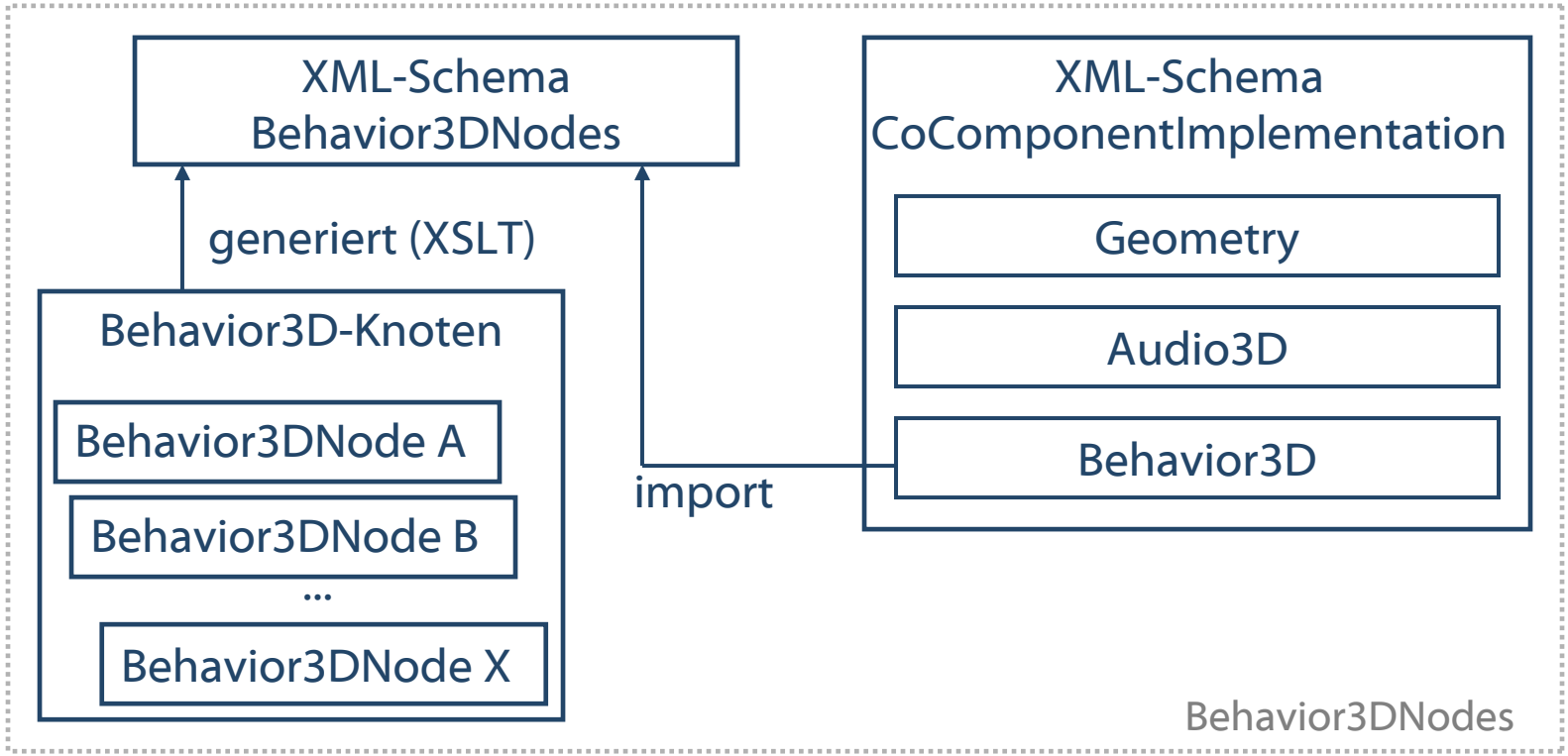
Behavior3D
(2) Dokumentengenerierung und (3) Implementierung

- Generierung der Dokumente (XSLT, XPath, Xalan)
  - Implementierung, Übersetzung, Test
- Implementierung
  - NodeNameTemplate.java: Zugriff auf die Felder, Hilfsmethoden
  - NodeName.java: Zu implementierender Klassenrumpf



Behavior3D
(4) Einbindung in den Behavior3D-Sprachumfang

- XML-Schema-Grammatik Behavior3DNodes
  - Repräsentiert den Sprachumfang aller Behavior3D-Knoten
  - Einbindung in CoComponentImplementation



Behavior3D
Konzipierte und implementierte Verhaltenskomponenten

- 2 neue Verhaltenskomponenten
- Animation-Component
  - Intuitive Beschreibung von komplexen Animationen
  - Adaption von SMIL 2.0 – Konzepten
  - 6 Animationsknoten (AnimateTranslation, AnimateRotation, AnimateColor, ...)
  - 2 Gruppierungs- und Synchronisationsknoten (Parallel, Sequential)
- State Machine – Component
  - Intuitive Beschreibung von komplexen Verhalten und Interaktionen mit Hilfe von verknüpfbaren Zustandsautomaten
  - Viewpoint-Zustandsautomatenkonzept

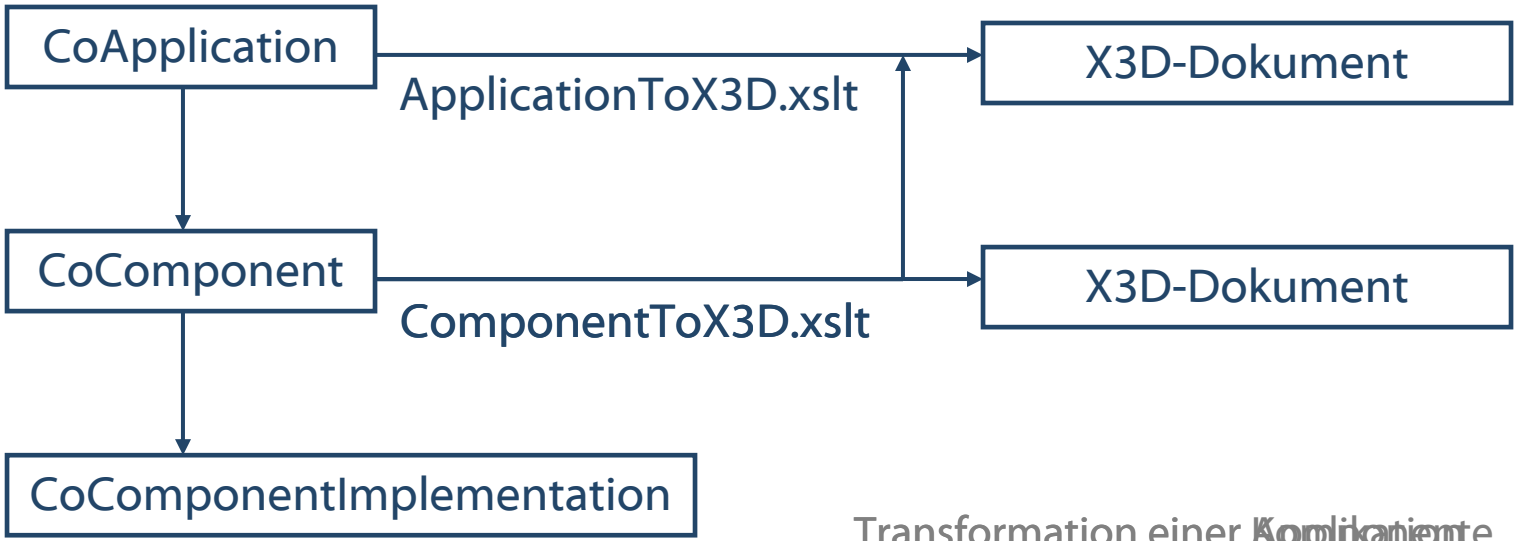
Behavior3D
Anwendungsbeispiel

```
<TimeSensor DEF="StateMachinePulse" cycleInterval="1" loop="true"/>
<Script DEF="StateMachine" url="javascript: function initialize (value, eventTime) {    state=1;  } function inPulse (value, eventTime)
      {switch (state) {case 1: state=2; start1=eventTime; break;case 2: state=3; start2=eventTime; break;case 3: state=4; start3=eventTime;
      break;case 4: state=1; start4=eventTime; break; }}">
      <field accessType="eventOut" name="start1" type="Time"/><field accessType="eventOut" name="start2" type="Time"/>
      <field accessType="eventOut" name="start3" type="Time"/><field accessType="eventOut" name="start4" type="Time"/>
      <field accessType="eventIn" name="inPulse" type="Time"/>
</Script>
<ROUTE fromField="cycleTime" fromNode="StateMachinePulse" toField="inPulse" toNode="StateMachine"/>
<TimeSensor DEF="MoveDownTime" cycleInterval="1"/>
<ROUTE fromField="start1" fromNode="StateMachine" toField="startTime" toNode="MoveDownTime"/>
<PositionInterpolator DEF="InterpolatorMoveDown" key="0, 0.5, 1" keyValue="0 1 0,0 0 0, 0 -1 0 "/>
<ROUTE fromField="fraction_changed" fromNode="MoveDownTime" toField="set_fraction" toNode="InterpolatorMoveDown"/>
<TimeSensor DEF="DownAnimateTime" cycleInterval="1"/>
<ROUTE fromField="start2" fromNode="StateMachine" toField="startTime" toNode="DownAnimateTime"/>
<OrientationInterpolator DEF="DownOrientationInterpolator" key="0, 1" keyValue="0 1 0 0,0 1 0 3.14"/>
<ColorInterpolator DEF="DownColorInterpolator" key="0,0.33,0.66,1" keyValue="0.678 0.063 0.094, 0.808 0.3563 0.063,0.937 0.71 0,0.678
      0.063 0.094"/>
<ROUTE fromField="fraction_changed" fromNode="DownAnimateTime" toField="set_fraction" toNode="DownOrientationInterpolator"/>
<ROUTE fromField="fraction_changed" fromNode="DownAnimateTime" toField="set_fraction" toNode="DownColorInterpolator"/>
<TimeSensor DEF="MoveUpTime" cycleInterval="1"/>
<ROUTE fromField="start3" fromNode="StateMachine" toField="startTime" toNode="MoveUpTime"/>
<PositionInterpolator DEF="InterpolatorMoveUp" key="0, 0.5, 1" keyValue="0 -1 0,0 0 0, 0 1 0"/>
<ROUTE fromField="fraction_changed" fromNode="MoveUpTime" toField="set_fraction" toNode="InterpolatorMoveUp"/>
<TimeSensor DEF="UpAnimateTime" cycleInterval="1"/>
<ROUTE fromField="start4" fromNode="StateMachine" toField="startTime" toNode="UpAnimateTime"/>
<OrientationInterpolator DEF="UpOrientationInterpolator" key="0, 1" keyValue="0 1 0 0,0 1 0 -3.14"/>
<ColorInterpolator DEF="UpColorInterpolator" key="0,0.33,0.66,1" keyValue="0.678 0.063 0.094, 0.224 0.518 0.451,0.353 0.612 0.71,0.678
      0.063 0.094"/>
<ROUTE fromField="fraction_changed" fromNode="UpAnimateTime" toField="set_fraction" toNode="UpOrientationInterpolator"/>
<ROUTE fromField="fraction_changed" fromNode="UpAnimateTime" toField="set_fraction" toNode="UpColorInterpolator"/>
```

```
<bno:Sequential loop="true" cycleInterval="5.0">
<bno:AnimateTranslation DEF="MoveDown" key="0 0.5 1" to="0 1 0,0 0 0, 0 -1 0"/>
</bno:Parallel>
      <bno:AnimateRotation DEF="BottomRotate" key="0 1" to="0 1 0 0,0 1 0 3.14"/>
      <bno:AnimateColor DEF="BottomColorAnimation" key="0 0.33 0.66 1"
      to="0.678 0.063 0.094, 0.808 0.3563 0.063,0.937 0.71 0,0.678 0.063 0.094"/>
</bno:Parallel>
<bno:AnimateTranslation DEF="MoveUp" key="0 0.5 1" to="0 -1 0,0 0 0, 0 1 0"/>
<bno:Parallel>
      <bno:AnimateRotation DEF="TopRotate" key="0 1" to="0 1 0 0, 0 1 0 -3.14"/>
      <bno:AnimateColor DEF="TopColorAnimation" key="0 0.33 0.66 1"
      to="0.678 0.063 0.094, 0.224 0.518 0.451,0.353 0.612 0.71,0.678 0.063 0.094"/>
</bno:Parallel>
</bno:Sequential>
```

Übersetzung in konkrete 3D-Formate

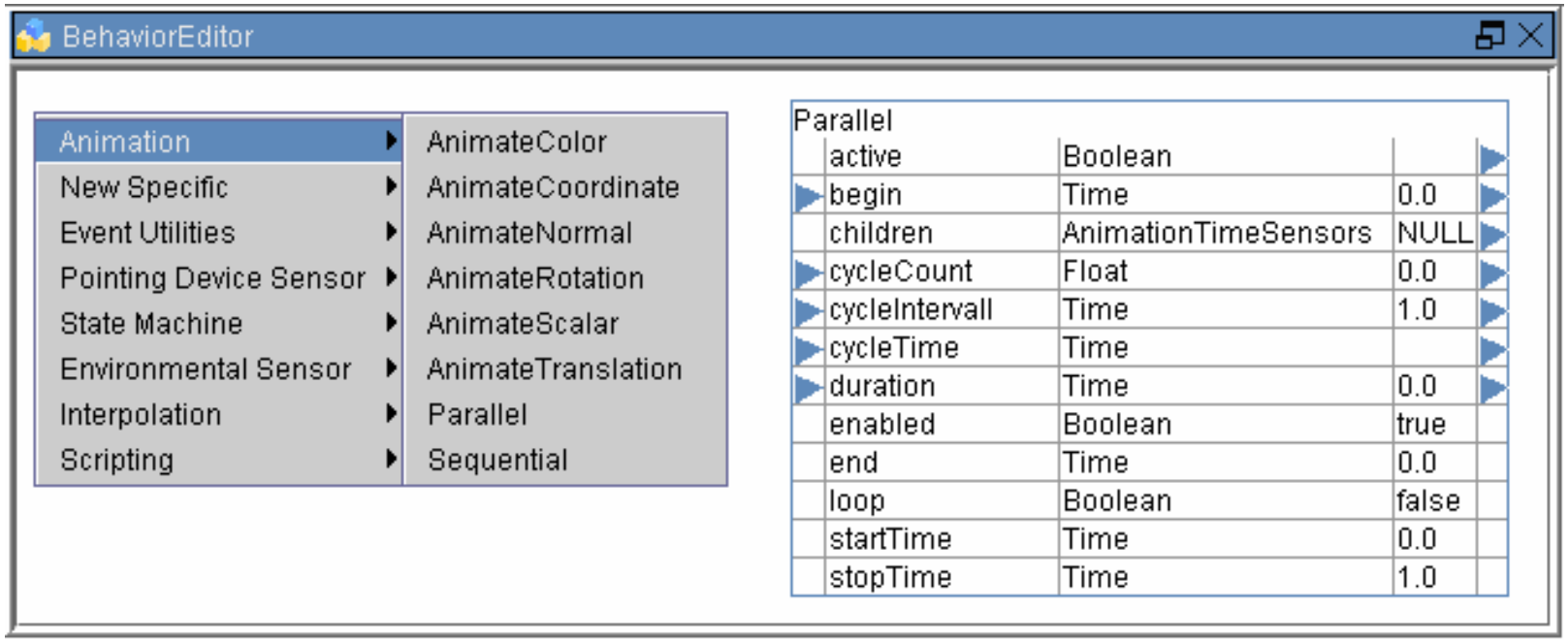
- Konzeptionelle Nähe von Contigra und X3D
  - Beschreibung von Geometrie, Verhalten (Behavior3D), Datentypen, Instanziierungskonzepte
- Technologien
  - XSL (XSLT, XPath), Xalan
- Transformation nach X3D



Transformation einer Komponente

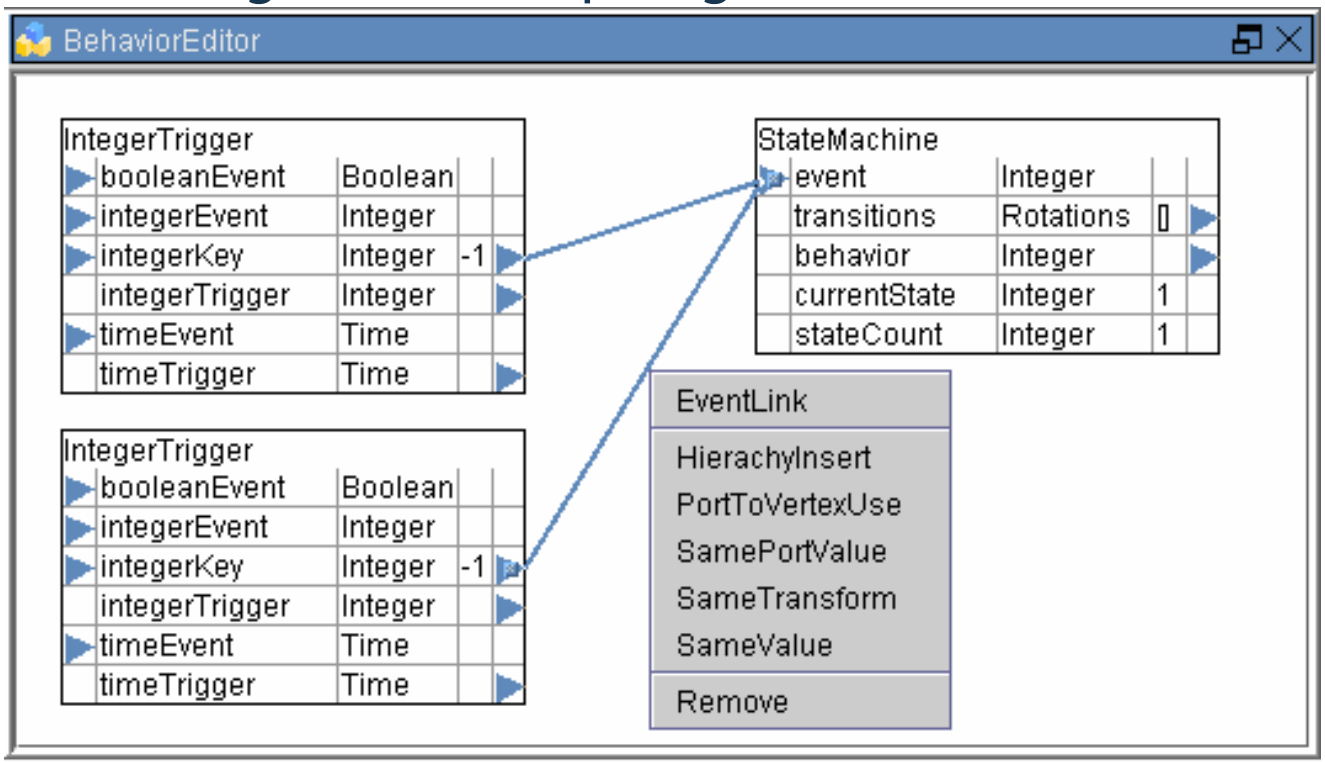
Verhalten im ContigraBuilder (1/2)

- Analyse von 3D-Autorenwerkzeugen (Verhalten)
  - Alice99, Cult3D Designer, Axel, Spazz3D, ISA
- Konzeption von Verhaltenseditoren
  - Konzentration auf den BehaviorEditor
- BehaviorEditor
  - Instanziierung / Parametrisierung von Behavior3D-Knoten



Verhalten im ContigraBuilder (2/2)

- BehaviorEditor
  - Erstellung von Verknüpfungen



  - JGraph (Java-API zur Darstellung von Graphen)
    Probleme: Editieren von Knoten und Komponenten, Verknüpfungen, Performance

Zusammenfassung und Ausblick

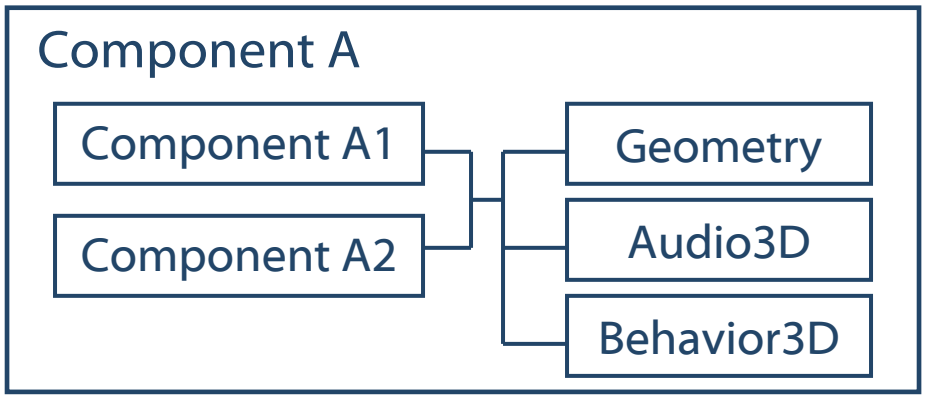
- Zusammenfassung
  - Analyse (Formate, Forschung, Werkzeuge)
  - Konzeption und Implementierung von Behavior3D
  - Verknüpfungskonzept
  - Integration in Contigra-Architektur
  - Konzeption und Implementierung der Übersetzung von Applikationen und Komponenten nach VRML97/X3D
  - Konzeption und prototypische Implementierung von Verhaltensaspekten des ContigraBuilder (BehaviorEditor)
- Nächste Schritte:
  - Entwicklung BehaviorEditor
  - Entwicklung neuer Verhaltenskomponenten (Constraints, Event Utilities, Sensoren)

Compact vs. Compromise

```
<!-- compact -->
<Shape>
    <Appearance>
        <Material/>
    </Appearance>
    <Box/>
</Shape>

<!-- compromise -->
<Shape>
    <appearance>
        <Appearance>
            <material>
                <Material/>
            </material>
        </Appearance>
    </appearance>
    <geometry>
        <Box/>
    </geometry>
</Shape>
```

Contigra-Verknüpfungskonzept



Contigra-Komponente

- Bestandteil der Contigra-Grammatiken
- 6 verschiedene Linktypen
  - z.B. EventLinkType, HierachyInsertType
- Kardinalitäten (1:1, 1:n, m:1, m:n)

```
<Link xsi:type="EventLinkType">
    <From vertex="VertexA" port="somePort"/>
    <To vertex="VertexB" port="somePort"/>
    <To vertex="VertexC" port="somePort"/>
</Link>
```

1:n - Verknüpfung

Literaturverzeichnis

[Diehl 1997] Stephan Diehl. *VRML++: A Language for Object-Oriented Virtual-Reality Models*. In: Proceedings of the 24th International Conference on Technology of Object-Oriented Languages and Systems TOOLS. Bejing, Asia, 1997.

[Diehl 1998] Stephan Diehl. *Object-Oriented Animations with VRML++*. In: Proceedings of Virtual Environments Conference and 4th Eurographics Workshop. Stuttgart, Germany, 1998.