Bachelor Thesis

Oryx
BPMN Stencil Set Implementation

Daniel Polak

Supervisors
Prof. Dr. Mathias Weske, Hasso-Plattner-Institute, Potsdam, Germany
Hagen Overdick, Hasso-Plattner-Institute, Potsdam, Germany
Gero Decker, Hasso-Plattner-Institute, Potsdam, Germany

30 June 2007

Abstract

Implementation of the complete Business Process Modeling Notation as a stencil set for our editor with name „Oryx“ is the topic of this Bachelor's paper, which is part of a cyclic documentation. Within these four papers, this would be the one, that documentate the creation process of a BPMN Stencil Set. This Bachelor's Paper is part of the Bachelor Project „Browser-Based Business Process Editor“, that has been performed at the Hasso-Plattner-Institut in the years 2006/2007.

Zusammenfassung

Die Implementierung eines vollständigen Business Process Modeling Notation Stencil Sets für unseren Editor mit dem Codenamen „Oryx“ wird in dieser Bachelorarbeit thematisiert. Sie ist Teil einer zyklischen Dokumentation, die aus vier Papern besteht, und dokumentiert den Prozess der Erzeugung eines BPMN Stencil Sets. Diese Bachelorarbeit ist Teil des Bachelorprojektes „Browser-Based Business Process Editor“, welches im Jahr 2006/2007 am Hasso-Plattner-Institut durchgeführt wurde.

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Potsdam, den 30. Juni 2007

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

1.1 Motivation

Prozessmodellierung ist ein wichtiger Bestandteil zur Analyse von Geschäftsprozessen in Unternehmen, um Transparenz über die bestehenden Werkstätigkeiten zu schaffen, um neue Aufgaben zu planen und laufende Abläufe zu optimieren. Gerade in größeren Unternehmen sind die Abläufe im Unternehmen so komplex, dass sie schwer oder gar nicht mehr ohne eine verständliche Modellierung vorstellbar sind und diskutiert werden können. Durch geeignete Modellierungsnotationen können auch komplexe Abläufe relativ übersichtlich visualisiert werden, wodurch die Möglichkeit besteht, Prozesse und deren Details konkret zu benennen und zu diskutieren. Damit ist die Grundlage geschaffen, gemeinsam an Erstellung und Optimierung von Prozessen zu arbeiten. Eine wichtige und moderne Notation zur Modellierung von Geschäftsprozessen ist die Business Process Modeling Notation, kurz BPMN. Spezifiziert wird sie durch die Object Management Group (OMG), einem Konsortium, das firmenübergreifend an internationalen Standarts arbeitet, und ist seit 2006 ein offizieller OMG-Standart [1]. BPMN beschreibt eine Notation von graphischen Elementen, die in ihrem Aussehen und der Bedeutung sowie der Verknüpfung solcher Elemente den festen Regeln der Spezifikation unterliegen. Eine Komposition dieser graphischen Elemente von BPMN wird als Business Process Diagram (BPD) bezeichnet.

1.2 Projekt

Unter dem Codenamen „Oryx“ haben wir im Rahmen des Bachelorprojektes ein Werkzeug geschaffen, dass komfortables Arbeiten mit der Business Process Modeling Notation ermöglicht. Durch intuitive Konzepte wird der Prozessmodellierer in seinem Arbeiten effektiv unterstützt. Genaueres dazu findet sich in der Bachelorarbeit von Willi Tscheschner [2]. BPMN wird in Oryx vollständig unterstützt, wodurch dem Modellierer große Freiheiten in der Modellierung eingeräumt werden. Ein umfangreiches Regelwerk soll verhindern, dass fehlerhafte Prozessdiagramme erstellt werden können. Vorwiegend für BPMN konzipiert soll dieser Editor aber Stencil

Sets jedweder Art zur graphischen Modellierung unterstützen und somit dem Benutzer umfangreiche Möglichkeiten zur Modellierung in verschiedenen Sprache in die Hand geben. Der Aufbau der Stencil Set Specifikation ist so gehalten, dass ein neues Stencil Set für eine graphische Modellierungssprache innerhalb kurzer Zeit erzeugt werden kann - abhängig natürlich von dem Umfang und der Komplexität der jeweiligen Notation.

1.3 Bachelorarbeit

Im Rahmen dieses Projektes entwickelte ich ein vollständiges BPMN Stencil Set, das alle BPMN Elemente sowie deren Attribute zur Verfügung stellt. Zum einen war dies Teil der Anforderung an den Editor, zum anderen verstand ich diese Aufgabe auch als Herausforderung, die Leistungsfähigkeit der formalen Stencil Set Spezifikation zu testen als Stencil Set Designer. Sowohl die Implementation als auch die Bewertung der Stencil Set Specification sind Teil dieser Bachelorarbeit. Abgerundet wird die Arbeit durch einen kritischen Blick auf den gesamten Editor aus der Sicht des Kunden, der den Editor nutzen möchte. Hierbei sollen auch Lösungen diskutiert werden im Hinblick auf spätere Erweiterungen unseres Editors.

Diese Bachelorarbeit ist Teil der Dokumentation rund um Oryx, zu dem alle vier Bachelorarbeiten des Projektteams gehören. Dabei stellt Willi Tscheschner [2] den Kern des Editors vor, die Konzepte, die dem Benutzer ein einfaches Arbeiten ermöglichen sollen, aber auch die Schnittstellen, die eine Erweiterung des Editors um weitere Funktionalität auf relativ einfache Art und Weise umsetzbar werden lassen. Vom Kern des Editors ausgehend ergeben sich zwei Themen, die ebenfalls in Arbeiten behandelt werden: Das Data Management wird von Martin Czuchra in seiner Bachelorarbeit [3] dokumentiert, wobei er im Besonderen die Integration des Editors in eine Seite sowie den Umgang mit Daten im DOM abhandelt. Nico Peters stellt in seiner Arbeit [4] die formale Spezifikation eines Stencil Sets vor, indem er eine Anleitung für den Stencil Set Designer zum Bau eines Stencil Sets gibt. Dieses ist natürlich Grundlage zur Implementation des BPMN Stencil Sets und damit essentiell für meine Arbeit.

2. Vorbetrachtungen

2.1 Herangehensweise

Zunächst stellt sich dem Stencil Set Designer die Frage, wie ein Stencil Set implementiert werden muss, und was dafür nötig ist. Da jedes Stencil Set auf der Grundlage der formalen Spezifikation aufgebaut ist, ist die Kenntnis derselben unabdingbar. Zur Implementierung eines Stencil Set ist müssen die Elemente der graphischen Notation durch die formalen Aspekte der Spezifikation umgesetzt und deren Regelwerk, welches darauf anzuwenden ist, beschrieben werden.

2.2 Besonderheiten der Business Process Modeling Notation

Die Implementierung von BPMN als Stencil Set in Oryx ist relativ komplex, verglichen mit der Umsetzung eines Stencil Sets für Petrinetze. Grund dafür ist, dass BPMN eine Fülle von graphischen hierarchisch geordneten Elementen besitzt, die allesamt eine Menge von Properties besitzen und ein umfangreiches Regelwerk. Aber der Umfang stellte nur eine der Herausforderungen an die Stencil Set Spezifikation dar. Problematisch wird es erst durch zahlreiche Besonderheiten im Detail, die sich vorab oftmals gar nicht abzeichnen. Anders als bei einer Menge von graphischen Elementen eines anderen Stencil Sets (z.B. eines Petrinetzes) zeigt das Metamodell von BPMN eine Vererbungshierarchie in den Elementen. Dies wird deutlich in den Properties, aber auch in dem Regelwerk. So geordnet diese Hierarchie auch wirkt, so durchwachsen ist sie leider auch mit Unstimmigkeiten und Abweichungen von dem Vererbungsmodell, die sich insbesondere in dem Regelwerk zeigen. Mit diesen Schwierigkeiten mit der BPMN Spezifikation sind leider z.T. Kompromisslösungen erforderlich.

3. Grundlagen

In diesem Kapitel geht es um die Grundlagen des Aufbaus eines Stencil Sets als Voraussetzung und zum Verständnis für die Implementierung des BPMN Stencil Sets.

3.1 Allgemeine Grundlagen

Ein Stencil Set wird gebildet durch eine Stencil Set Definitionsdatei, welche die Stencils beschreibt mit ihren Eigenschaften und dem zugehörigen Regelwerk. Diese Datei enthält ebenfalls Informationen dazu, welche graphischen Repräsentationen an das Stencil gebunden ist, wobei diese graphische Repräsentation nicht Bestandteil der Stencil Set Beschreibung ist, sondern nur in Form eines Verweises referenziert wird. Die graphische Darstellung eines Stencils erfolgt in SVG Dateien [5], welche ein Stencil beschreiben. Details für die Technologieentscheidung finden sich in der Bachelorarbeit von Willi Tscheschner [2]. Jede SVG Datei beschreibt genau ein Stencil, wobei mehrere Elemente einer Modellierungsnotation durchaus in einer SVG-Datei zusammengefasst werden können, wenn das Regelwerk und die Eigenschaften sowie eine ähnliche graphische Repräsentation es zulassen. Für ein in der Stencil Set Definition beschriebenes Stencil ist natürlich auch ein Icon erforderlich, welches das Stencil in Menüs im Editor repräsentiert. So wird das Icon sowohl im Stencil Repository als auch im Shape Menü angezeigt, und sollte daher eindeutig das Shape beschreiben.

3.2 Laden des BPMN Stencil Sets

Um den Editor mit dem BPMN Stencil Set zu laden, ist eine XHTML-Seite erforderlich, die den Editor einbindet und darüberhinaus die Canvas durch ein Element des Stencils Sets typisiert. Existiert diese Seite, so kann der Editor mit dem Stencil Set wie jede normale Internetseite abgerufen werden.

```
<!-- Canvas - for editor only -->
<div id="oryxcanvas1234" class="-oryx-canvas" style="width:1200px; height:600px;">
    <a rel="oryx-stencilset" href="./data/stencilsets/bpmn/bpmn.json" />
</div>
```

Durch den relativen Verweis auf die StencilSet-Datei wird das BPMN Stencil Set geladen. Der Editor wird auf die Größe des Browserfensterinhaltes vergrößert.

Durch Einfügen eines Meta-Tags im Header wird der Typ der Canvas auf ein Stencil des Stencil Sets festgelegt. Im Normalfall wird dies das Diagram Object sein.

```
<meta name="oryx.type" content="http://b3mn.org/stencilset/bpmn#BPMNDiagram"/>
```

Details zu diesem Konzept findet sich im Kapitel 4.9.

Darüberhinaus besteht die Möglichkeit, ein StencilSet nachzuladen über den Button „Stencil Set hinzufügen“, was eine Eingabebox zur Eingabe eines relativen Links auf die Stencil Set Definitionsdatei nach sich zieht.

3.3 Aufbau des BPMN Stencil Sets

Ein Stencil Set ist die Beschreibung einer Menge von graphischen Objekten mit ihren graphischen Repräsentationen, Eigenschaften und Regeln. Es wird oftmals aus der Menge aller Elemente einer Modellierungsnotation gebildet. So lassen sich Stencil Sets für BPMN, aber auch Workflow-Netze, UML, FMC, und andere Modellierungssprachen bauen. Im Rahmen des Projektes sind Stencil Sets für BPMN, Workflow-Netze, Petri-Netze und eine an BPMN angelehnte Notation unseres Schwesternprojektes „Thanis“ zur Ausführung von Prozessen entstanden.

Das BPMN Stencilset setzt sich aus der Stencil Set Definitionsdatei „bpmn.json“ und den graphischen Repräsentationen der Shapes, die in SVG geschrieben sind, sowie einem kleinen Icon der Größe 16 x 16 Pixel.

Eine SVG Datei kann exemplarisch so aussehen:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg
   xmlns="http://www.w3.org/2000/svg"
   xmlns:oryx="http://www.b3mn.org/oryx"
   width="40"
   height="40"
   version="1.0">
   <defs></defs>
   <oryx:magnets>
      <oryx:magnet oryx:cx="16" oryx:cy="16" oryx:default="yes" />
   </oryx:magnets>
   <g pointer-events="fill">
      <path id="frame" fill="white" stroke-width="1" stroke="black"
         d="M 1,16 L 16,1 L 31,16 L 16,31 L 1,16" />
      <circle cx="16" cy="16" r="7.5"
         stroke="black" fill="none" stroke-width="2.5"/>
  </g>
</svg>
```

Abbildung 3.1: SVG-Code eines Inclusive Gateways

Die Stencil Set Definitionsdatei ist wiefolgt aufgebaut:

- Definition der Stencils

- Regelwerk

Das Regelwerk wird in 4.2 ausführlich erläutert, ein BPMN Stencil wird folgendermaßen in der bpmn.json definiert - hier am Beispiel des Inclusive Gateways:

```
{
  "type": "node",
  "id":"OR_Gateway",
  "title":"OR Gateway",
  "groups":["Gateways"],
  "description":"A decision point.",
  "view":"gateway/node.gateway.or.svg",
  "icon":"new_gateway_or.png",
  "roles": [
    // Verschiedene Rollen, die das Stencil einnehmen kann
  ],
  "properties": [
    {
      "id":"id",
      "type":"String",
      "title":"Id",
      "value":"",
      "description":"",
      "readonly":false,
      "optional":false,
      "refToView":"",
      "length":"30"
    },
    // weitere Properties
  ]
}
```

Der Typ bestimmt, ob es sich um einen Knoten oder eine Kante handelt. Mit „groups“ wird die Gruppe des Stencils festgelegt, unter der es im Shape Repository angezeigt wird. Darunter befinden sich neben der textuellen Beschreibung Verweise auf die graphische Repräsentation des Stencils sowie der Ikonifizierung. Die Rollen dienen ausschließlich dem Regelwerk, das - wie schon erwähnt, später genauer beschrieben wird. Danach folgen die Attribute eines Properties; ein Konzept, das unter 4.1 genauer erläutert wird.

Hinweis: Die komplette Stencil Set Definitionsdatei (bpmn.json) kann nicht in diesem Dokument mitgeliefert werden, da sie mit über 6000 Zeilen über 100 Seiten einnehmen würde.

3.4 Ordnerstruktur und Namenskonventionen

Ein Stencil Set soll klar von anderen Stencil Sets abgetrennt sein, wodurch ein einfacher Import von Stencil Sets („Nachladen“) ermöglicht werden soll. Das Stencil Set selbst wurde hierarchisch strukturiert, um eine klare Trennung von graphischer Repräsentation eines Stencils und deren Ikonisierung im Editor zu halten. Darüberhinaus ist eine einheitliche Benennung von Ordnern und Dateien gerade dann unablässig, wenn eine Vielzahl an Stencils in einem Stencil Set implementiert werden sollen. In diesem Abschnitt wird das für die Implementation von BPMN erarbeitete und in anderen Stencil Sets übernommene Schema erläutert werden.

3.4.1 Lage des Stencil Set im Editor

Stencil Sets befinden sich im Data-Ordner des Editors:

```
./oryx/data/stencilsets/[Stencil Set Name]/
```

Hierbei verwendet jedes Stencil Set seinen eigenen Unterordner im Ordner „stencilsets“, wobei der Name auf die umgesetzte Modellierungsnotation hinweisen sollte. Im Falle von BPMN lautet der Pfad zum Stencil Set:

```
./oryx/data/stencilsets/bpmn/
```

Dieses ist das Wurzelverzeichnnis für alle nachfolgend beschriebenen Strukturen.

3.4.2 Ordnerstruktur im Stencil Set

Die folgende Ordnerstruktur ist essentiell für jedes Stencil Set:

```
[bpmn]
  |- icons   // Symbole der Stencils
  |- view    // Graphische Repräsentation der Stencils (SVG-Dateien)
```

Um bei der großen Anzahl von Stencils in BPMN Übersicht zu wahren, wurde eine tiefere Struktur eingeführt. Diese ist angelehnt an die im Editor verwendeten Untergruppen für Stencils, die im Shape Repository für eine logische Strukturierung der Stencils sorgen. Durch dieses Konzept soll trotz des Umfangs an BPMN Elementen Übersicht gewährt bleiben. Aus dieser Überlegung heraus entstand eine Unterstrukturierung, die Elemente von BPMN, die zueinander ähnlich sind, gemeinsamen Gruppen zuweist. Der vollständige Baum sieht wiefolgt aus:

```
[bpmn]
  |- icons   // Symbole der Stencils
  |- view    // Graphische Repräsentation der Stencils (SVG-Dateien)
    |- activity           // Enthält Subprozesse und Tasks
    |- artifact           // Enthält Data Object, Group und Text Annotation
    |- connection         // Enthält alle Connecting Shapes (Sequence Flow, etc.)
    |- diagram            // Enthält ein Diagram Shape
    |- endevent           // Enthält alle Endevents
    |- gateway            // Enthält alle Gateways
    |- intermediateevent  // Enthält alle Intermediateevents
    |- startevent         // Enthält alle Startevents
    |- swimlane           // Enthält Pools und Lanes)
```

3.4.3 Namensschemata

Ein einheitliches Schema zur Benennung schafft für jeden Stencil Set Designer Transparenz und Verständlichkeit, von daher wurde ein Namensschema im Rahmen des BPMN Stencil Sets definiert.

3.4.3.1 Stencil Set Definitions Datei

Die Stencil Set Definitions Datei ist gleichnamig zu dem Stencil Set benannt. Die BPMN Stencil Set Definitions Datei heißt „bpmn.json“.

3.4.3.2 SVG-Dateien

Jede graphische Repräsentation eine BPMN Stencils liegt in Form einer SVG-Datei vor und folgt einem festen Namensschema.

Konnektoren

Die SVG-Repräsentation aller Konnektoren sind in dem Unterordner „view/connection/“ abgelegt. Sie folgen dem Namensschema „connection.[Name des Konnektors].svg“.

Beispiel:

```
connection.sequenceflow.svg
```

Einige Konnektoren gleichen Typs besitzen mehrere graphische Repräsentationen, die sich lediglich in den Enden (in SVG: Marker) und partiell auch in den auf die Stencils anzuwendenden Verbindungsregeln unterscheiden. Dies betrifft in BPMN die SequenceFlow-Subklassen ConditionalFlow und Defaultflow, sowie die Assoziationen mit keiner, einer, oder zwei Richtungspfeilen.

Sequenceflow:

- Normaler Sequenceflow

  ```
  connection.sequenceflow.normal.svg
  ```

- Conditionaler Sequenceflow

  ```
  connection.sequenceflow.default.svg
  ```

- Default Sequenceflow

  ```
  connection.sequenceflow.default.svg
  ```

Assoziation:

- Ungerichtete Assoziation

  ```
  connection.association.undirected.svg
  ```

- Einfach gerichtete Assoziation

  ```
  connection.association.unidirectional.svg
  ```

- Doppelt gerichtete Assoziation

  ```
  connection.association.bidirectional.svg
  ```

Sonstige Objekte

Objekte, die keine Konnektoren darstellen, werden allesamt als Knoten im Editor angesehen. Daher lautet das Namensschema der SVG-Dateien „node.[Name des Objekts].svg“.

Beispiel:

```
node.subprocess.svg
```

Wie bei den Konnektoren gilt auch für die Knotenobjekte, dass Subtypen durch Anhängen des Typs im Namen namentlich spezifiziert werden. Bei BPMN zeigt sich das insbesondere bei den Gateway- und Eventtypen.

Beispiel 1 (Gateways):

```
node.gateway.xor.databased.svg
```

Beispiel 2 (Events):

```
node.event.message.start.svg
node.event.message.intermediate.svg
node.event.terminate.end.svg
...
```

3.4.3.3 Stencil Icons

Die Icons für Stencils liegen im „icons“-Ordner im Stammverzeichnis eines Stencils Sets. Die implizite Namenskonvention der Stencils gilt auch hier, nur werden statt Punkten Unterstriche geschrieben, und einige Namen auch vereinfacht.

Beispiel:

```
new_task.png
new_multipleevent.png
new_association_undirected.png
...
```

Anzumerken ist hierbei, dass die Schreibweise historisch gewachsen ist und wohl bei der nächsten Überarbeitung vollständig der Namenskonvention der SVG-Stencil-Repräsentationen angepasst werden.

3.5 Hilfsprogramme

Als Hilfsmittel zur Gestaltung eines Stencil Sets bieten sich folgende zwei Werkzeuge an:

3.5.1 Paint.NET

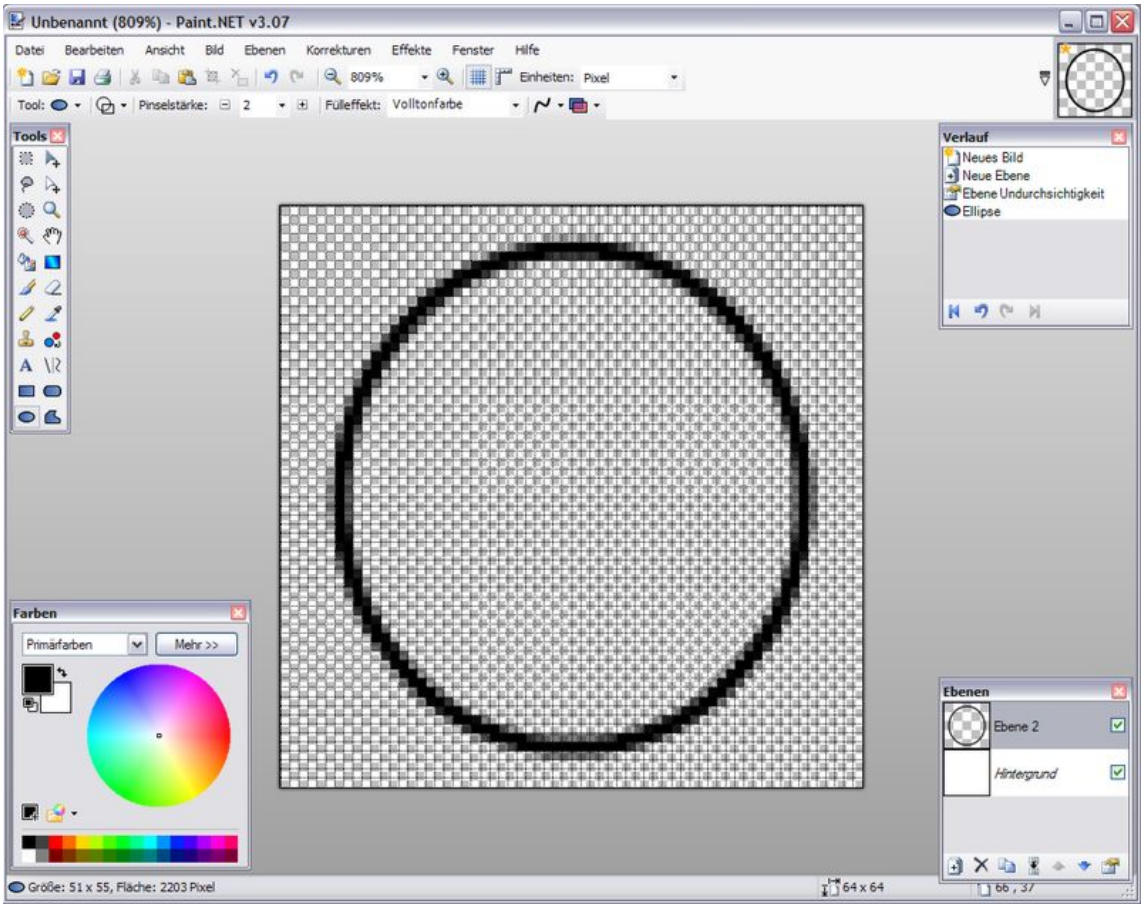

Abbildung 3.2: Paint.NET

Paint.Net ist ein freies Bildbearbeitungsprogramm unter MIT-Lizenz, das von der Washington State University entwickelt und nun von dem Paint.NET-Team weiterentwickelt wird. Es besitzt zwar nicht dem Umfang großer Grafikbearbeitungssuites, überzeugt jedoch mit seinem an PhotoShop angelehnten Ebenensystem und den zahlreichen Spezialeffekten. Die Funktionalität lässt sich durch PlugIns erweitern. Die wichtigsten Grafikformate werden nativ unterstützt, weitere lassen sich mit Hilfe von PlugIns anzeigen.

Dieses Programm eignet sich hervorragend für die Erstellung von Icons für das Stencil Repository und das Shape Menü. Die Bilder aus dem Ordner Icons wurden teilweise mit diesem Programm erstellt. Durch das Ebenensystem und dem Alphablending erlaubt es, die erzeugten Icons als PNG-Dateien (Portable Network Graphics) zu exportieren, wobei die Alphatransparenz erhalten bleibt.

Verfügbar ist das Programm für Windows XP und Vista derzeit in der Version v3.08 vom 1. Juni 2007 unter: http://www.getpaint.net/index2.html

3.5.2 Inkscape

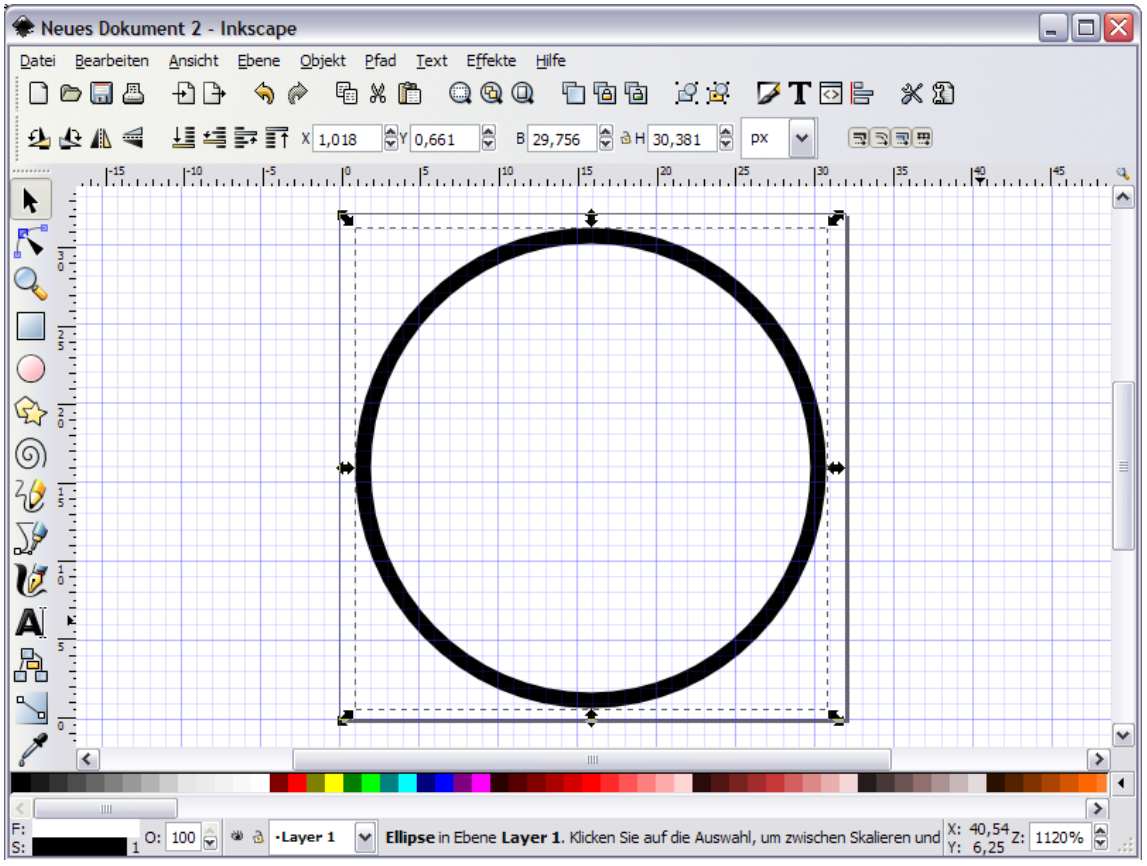

Abbildung 3.3: Inkscape

Inkscape ist ein freies Vektorgrafik-Zeichenprogramm, dass unter der GPL Lizenz steht. Entwickelt wird es von der Inkscape Community, die besonderen Wert auf die Konformität von SVG mit dem Standart legt. Sogar Collaboration soll nun in den neuen Versionen integriert sein, um mit mehreren Personen gleichzeitig an einer Grafik zu zeichnen wie an einem Whiteboard. Es besitzt viele Features, wie Fließtext auf Pfaden, graphische Effekte und Layer. Besonders lobenswert ist die Möglichkeit, Vektorgrafiken in Rastergrafiken zu konvertieren.

Mit diesem Programm lassen sich einfach und ohne viel Vorwissen zu SVG auch komplexe graphische Repräsentationen von Stencils für ein StencilSet zeichnen. Allerdings ist dann immer noch Handarbeit gefragt, um Oryx-spezifische Attribute und Elemente im Oryx-Namesspace (z.B. Magnets) zu ergänzen. Gezeichnete Stencils können auch skaliert und dann als Icon für das Stencil im PNG-Format exportiert werden. Alphatransparenz wird dabei selbstverständlich unterstützt.

Das Programm ist für Windows, Linux und Mac OSX verfügbar in der Version 0.45.1 vom 23. März 2007 unter: http://www.inkscape.org/?lang=de

4. Implementierung

Dieses Kapitel beschreibt die Implementierung von BPMN als Stencil Set für den Oryx-Editor. Aufbauend auf den vorangehend beschriebenen Grundlagen werden nun die Grundkonzepte „Properties“ und „Regeln“ bezüglich BPMN erläutert, anschließend die Überführung aller Elemente von BPMN in das Stencil Set.

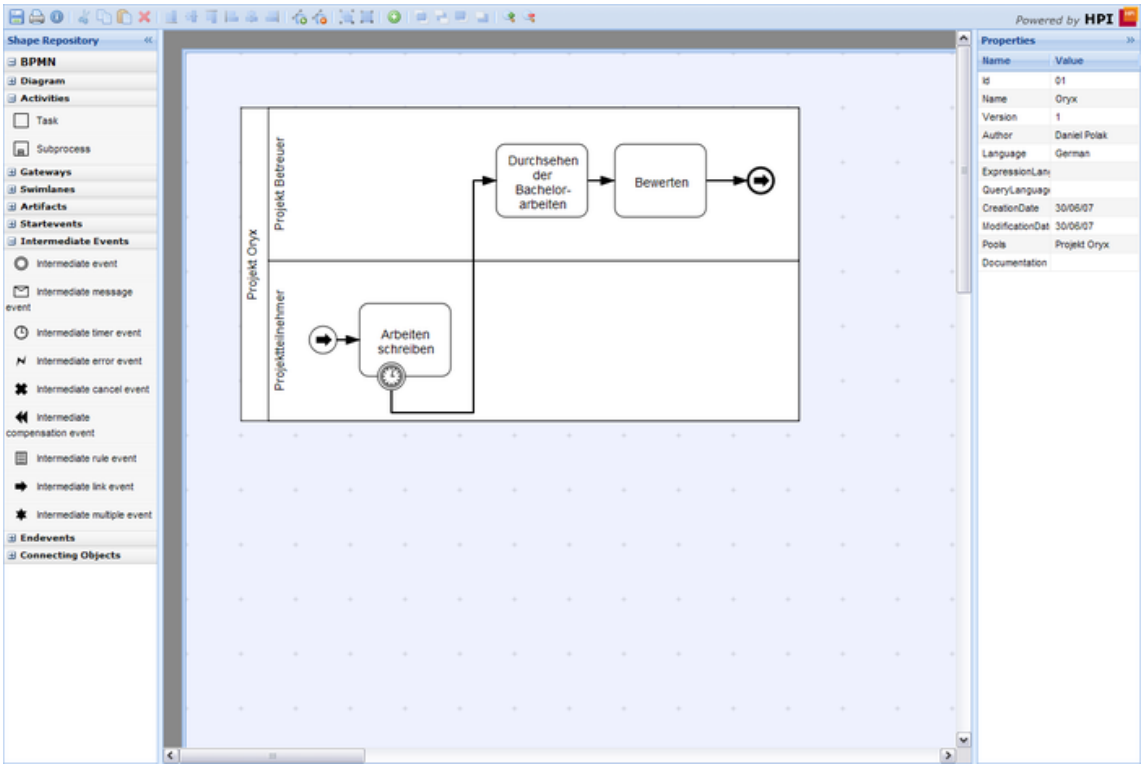

Abbildung 4.1: Oryx mit geladenem BPMN Stencil Set und modelliertem Prozess

4.1 Properties

Properties sind ein einfaches Konzept zur Bindung von Eigenschaften an ein Shape. Dies können Eigenschaften aus der im Stencil Set beschriebenen Modelierungssprache sein als auch werkzeugbezogene Eigenschaften, welche das Aussehen des Prozessgraphen im Tool bestimmen. In BPMN werden jedem Element eine Fülle von Attributen mitgegeben, welche zum Teil das Aussehen, den Zustand mit Umgebungswissen und das Verhalten bei Ausführung eines Elements beschreiben. So kennt beispielsweise jedes Objekt in BPMN seine eindeutige Id im Business Process Diagram. In Petrinetzen kennt eine Transition seine Beschriftung dank eines Properties, so sie denn eine hat. Und in wieder anderen Modelierungssprachen gibt es andere Eigenschaften, die dem Stencil zugeordnet werden sollen. In diesem Abschnitt wird das Konzept „Property“ anhand des BPMN Stencil Sets erläutert. Auf alle Attribute einzeln einzugehen würde bei BPMN allerdings den Rahmen sprengen, daher ist die Stencil Set Definitionsdatei auch nicht als Anhang beigefügt.

Properties sind in der Stencil Set Definitionsdatei beim Stencil selbst angelegt und sehen beispielsweise wiefolgt aus:

```
"properties": [
  {
    "id":"Id",
    "type":"String",
    "title":"Id",
    "value":"unique10245",
    "description":"This is a unique Id that distinguishes the Diagram from other Diagrams.",
    "readonly":false,
    "optional":false,
    "length":"50"
  }
]
```

Der Aufbau des Property-Konstruktes wird in der Bachelorarbeit von Nicolas Peters [4] genau beschrieben, hier sei kurz erwähnt, dass dieses Property vom Typ String eine mandatory Id beschreibt.

Manche Properties nehmen Einfluss auf die graphische Gestaltungsform eines BPMN Shapes. So bestimmt in einem Data-Based XOR Gateway das Attribut „MarkerVisible“ darüber, ob ein Kreuz in der Mitte des Shapes angezeigt wird oder nicht.

Property im Stencil Set:

```
{
  "id":"markerVisible",
  "type":"Choice",
  "title":"MarkerVisible",
  "value":"false",
  "description":"",
  "readonly":false,
  "optional":false,
  "refToView":"marker",
  "items": [
    {
      "id":"c1",
```

```
      "title":"c1",
      "value":"false",
      "refToView":"none"
    },
    {
      "id":"c2",
      "title":"c2",
      "value":"true",
      "refToView":"cross"
    }
  ]
}
```

In dem obig definierten Choice-Property hat jedes Item eine Referenz, die auf eine Id in der SVG Datei verweist. Das aktuell gewählte Item lässt das gleichnamige Element in der SVG-Datei einblenden, und verbirgt dafür alle anderen.

SVG-Datei des Data-Based XOR Gateways:

```
<g id="none"></g>

<g id="cross">
  <path d="M 11.5 9.5 L 14.5 9.5 L 20.5 22.5 L 17.5 22.5 L 11.5 9.5"
        fill="black" stroke="black" stroke-width="1" />
  <path d="M 11.5 22.5 L 17.5 9.5 L 20.5 9.5 L 14.5 22.5 L 11.5 22.5"
        fill="black" stroke="black" stroke-width="1" />
</g>
```

Es wird immer nur das g-Element mit der ausgewählten Id angezeigt, alle anderen werden ausgeblendet. Somit können Inhalte variabel gestalten werden. Ist nun das Item „none“ ausgewählt, so wird das Kreuz, wie links ersichtlich, nicht angezeigt.

Abbildung 4.2: None

Ist das Item „cross“ ausgewählt, wird das Kreuz angezeigt. Im Übrigen hat es keinerlei funktionelle Bedeutung, es sind nur zwei verschiedene Darstellungen für ein und dasselbe Stencil.

Abbildung 4.3: Cross

Ein editorspezifisches Property ist in allen Objekten enthalten, die einen gefüllten Hintergrund haben: Die Möglichkeit, die Hintergrundfarbe zu ändern mit Hilfe eines ColorPickers.

4.1.1 Mapping von Properties

BPMN ist eine relativ abstrakte Modellierungssprache, die ihre eigenen Typen für Properties definiert und keinerlei Rücksicht darauf nimmt, ob Computer damit umgehen können. So besitzt sie elf spezielle Typen, die im nachfolgenden Abschnitt besprochen werden. Im Allgemeinen lassen sich die meisten Attribute in Strings anzeigen, angefangen von Id's, Objektreferenzen, Namen und Beschreibungen. Einige Attribute in BPMN erlauben mehrere Objekte es Typs. Klassisches Beispiel dafür ist das „Lanes"-Attribut im Pool. In BPMN ist nicht spezifiziert, wie auf die Lanes gezeigt werden soll, noch wie mehrere Lanes aufgeführt werden sollen. Denkbar wäre natürlich gewesen, dass man eine dynamische Unterproperty-Struktur aufbaut, die es ermöglicht, vergleichbar zum Stencil Repository einfach alle Unterelemente auszuklappen und Neue hinzuzufügen und Bestehende zu manipulieren und zu ändern. Natürlich ist „Lanes" ein Attribut, welches automatisch vom Stencil Set her mit Werten gefüllt werden müsste. Nach reiflicher Überlegung boten sich zwei vorstellbare Lösungen an: Das Property ist vom Typ String und enthält alle Id's der zugehörigen Lanes, aufgeführt als mit Kommata getrennte Liste, oder aber in einer Multiline-Box, wo dann Zeilenumbrüche als Trennzeichen verwendet werden. Auf diese Weise wurden viele Attribute, die mehrere Werte enthalten können, als Strings im Editor behandelt, wobei je nach Kontext eine Multilinebox verwendet werden konnte. Es lässt sich anmerken, dass alle komplexen Attribute in BPMN sich durch Strings halten lassen, wobei eine feste Syntax zur Interpretation derselben definiert werden muss. Bisher arbeitet der Editor allerdings nur insofern mit den Attributen, als dass er sie als Properties visualisiert.

4.1.2 Mapping von Supporting Types

Da die formale Stencil Set Spezifikation derzeit keine komplexen Properties unterstützt, müssen BPMN Property Typen in editorverständliche Typen umgeformt werden, damit mit ihnen gearbeitet werden kann. BPMN besitzt elf Supporting Types, welche wie folgt interpretiert werden:

Supporting Type	Abgebildet auf Typ	Bedeutung
Assignment	String	Anweisung
Entity	String	Name des Entity
Expression	String	Ausdruck der Expression
Message	String	Bildet die Nachricht ab
Object	String	Zeigt die Id eines Objekts
Participant	String	Zeigt die Rolle eines Teilnehmers (auf Supporting Type „Role" gemappt)
Property	String	Name/Typ als Wertepaar
Role	String	Name der Rolle
Rule	String	Name und Ausdruck der Regel
Transaction	String	Transaktions Id, Protokoll und Methode als Feld (Multiline String)
Web Service	String	Teilnehmer, Interface und Operation als Feld (Multiline String)

4.2 Regeln

Benutzerfreudlich wird der Editor gerade deshalb, weil er nicht nur erlaubt, Stencil Sets mit vordefinierten Stencils zu erzeugen, sondern gerade wegen des semantischen Verständnisses darüber, was auf welche Art und Weise mit wem verbunden sein darf und was nicht. Um das zu erreichen, gibt die Stencil Set Spezifikation dem Stencil Set Designer die Möglichkeit, eine komplexe Menge an deklarativen Regeln festzulegen, die dafür sorgen, dass nur in sich konsistente Diagramme erstellt werden können. Natürlich ist ein Regelwerk sehr komplex, besonders dann, wenn das zu Grunde liegende Stencil Set sehr umfangreich ist. Regeln zum Bau valider Petri-Netz-Diagramme sind sehr einfach, da im einfachen Petri-Netz-Modell mit Kanten, Transitionen und Stellen lediglich als Regel definiert werden muss, dass mit Kanten immer Stellen nur mit Transitionen verbunden werden dürfen. In BPMN ist so ein Regelwerk allerdings deutlich komplexer, da die Anzahl der gesamten Elemente und insbesondere auch der Kantenelemente deutlich höher ist als bei andere Notationen.

Die Stencil Set Spezifikation arbeitet grundlegend so, dass sie alles verbietet, was nicht explizit erlaubt ist. Folglich müssen also Regeln definiert werden, die sagen, was erlaubt ist. Es gibt keine Anti-Regeln, die etwas explizit verbieten. Die Details zu diesem Konzept finden sich in der Bachelorarbeit von Nicolas Peters [4].

Die Visualisierung des Ergebnisses von der Gültigkeitsprüfung beim Modellieren ist in Oryx sehr benutzerfreundlich gestaltet, indem das Zielshape durch einen roten Rahmen eine Regelverletzung und durch einen grünen Rahmen einen zulässigen Schritt verdeutlichen.

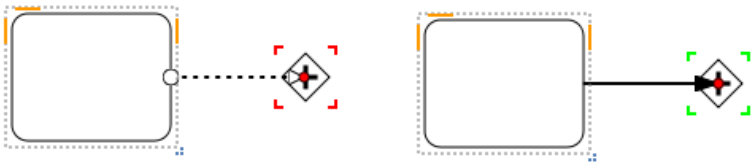

Abbildung 4.4: Visualisierung von Verbindungssemantik

Es stellt sich schnell die Frage, welche Arten von Regeln benötigt ein Stencil Set, welches durch das Erarbeiten der Anforderungen von BPMN deutlich wurde und wiederum in die Stencil Set Spezifikation einfloss. In BPMN finden derzeit drei Regel-Konzepte Anwendung:

- Regeln, die gültige Verbindungen von Flussobjekten bestimmen
- Regeln, die bestimmen, wie oft ein Objekt vorkommen darf oder wieviele Verbindungen ein- bzw ausgehen dürfen
- Regeln, die festlegen, ob ein Element ein anderes enthalten darf

Eine vollständige Liste aller in das BPMN Stencil Set implementierten Regeln findet sich im Anhang A.

4.2.1 Verbindungsregeln (Connection Rules)

Die wichtigste Komponente an Regeln stellen sicherlich die Verbindungsregeln dar, welche bestimmen, welches Knotenobjekt mit welchem anderen Knotenobjekt verbunden werden darf. In BPMN sind bei 3 verschiedenen Kantenarten mit partieller Spezialisierung (Sequence Flow und Assoziation) einige Regeln notwendig, damit BPMN konform modelliert werden kann. Interessanterweise dürfen in Spezialfällen nicht nur Knotenobjekte mit Knotenobjekten, sondern auch mit Kantenobjekten durch Kanten verbunden sein.

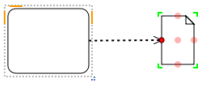

Abbildung 4.5: Beispiel Verbindungsregel

In der BPMN Stencil Set Spezifikation werden Verbindungsregeln unter der Sektion „connectionRules“ aufgeführt.

Beispiel einer Verbindungsregel:

```
{
  "role":"SequenceFlow",
  "connects": [
    {
      "from":"sequencestart",
      "to":["sequenceend"]
    }
  ]
}
```

In dem gewählten Beispiel wird erlaubt, eine Verbindung von einem Element, dass die Rolle „sequencestart“ einnehmen kann, zu einem Element, das die Rolle „sequenceend“ einnehmen kann, eine Verbindung durch einen Sequenzfluss zu etablieren. Auf diese Art und Weise lassen sich sämtliche Verbindungsregeln mühelos etablieren.

Eine besondere Form der Verbindung stellt die Möglichkeit dar, IntermediateEvents an den Rahmen einer Aktivität anzuknüpfen - von Projektbetreuer Hagen Overdick auch als „Scope Events bezeichnet“. Diese lassen ebenfalls mit den Verbindungsregeln abbilden, wobei die Zwitterrolle des IntermediateEvents als Knoten und Kante zugleich sehr deutlich wird, da sie in der Regel auch die Rolle der verbindenden Kante einnimmt.

Beispiel eines Intermediate „Scope“ Event:

```
{
  "role":"IntermediateErrorEvent",
  "connects": [
    {
      "from":"Task",
      "to":["IntermediateErrorEvent"]
    }
  ]
}
```

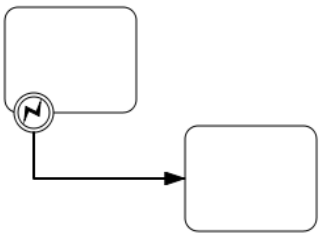

Abbildung 4.6: Intermediate „Scope“ Event

4.2.2 Kardinalitätsregeln (Cardinality Rules)

Die Stencil Set Spezifikation erlaubt auch Regeln zur Bestimmung von maximalen ein- und ausgehenden Kanten sowie dem maximalen Vorkommen eines Shapes.

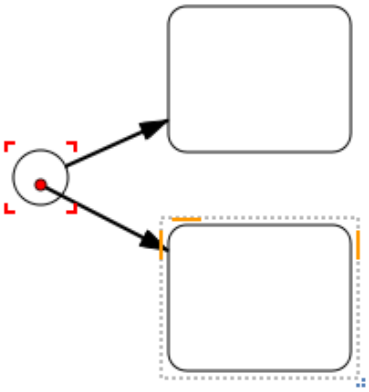

Abbildung 4.7: Beispiel Kardinalitätsregel

In der BPMN Stencil Set Spezifikation werden Kardinalitätsregeln unter der Sektion „cardinalityRules“ aufgeführt.

Beispiel einer Verbindungsregel:

```
{
  "role":"Startevents_all",
  maximumOccurrence:undefined,
  outgoingEdges: [
    {
      role:"SequenceFlow",
      maximum:1,

    }
  ]
}
```

In dem gewählten Beispiel wird zunächst das maximale Auftreten von Shapes mit dieser Rolle auf unbegrenzt festgesetzt, danach die Anzahl der ausgehenden Kanten festgesetzt. Ein Start Event darf keine eingehenden Kanten besitzen. Um zu verhindern, dass eine Kante ein- oder ausgehen darf, ist es jedoch sinnvoller, das über ConnectionRules zu realisieren. Hier werden die ausgehenden Kanten eines Start Events auf maximal einen Sequenzfluss beschränkt. Ebenso kann man die Anzahl

der Kanten, die aus einem Intermediate „Scope“ Event ausgehen, so auf maximal eine limitieren. maximumOccurrence wird in der Form in BPMN nicht benötigt, im Zusammenhang mit dem Diagramm-Stencil wäre denkbar, festzulegen, dass nur ein einziges Diagramm pro Seite dargestellt werden darf.

4.2.3 Enthaltenseinsregeln (Containment Rules)

Zuguterletzt sind vor allem auch die Containment Rules wichtig, welche darüber bestimmen, welches Shape auf einem anderen Shape liegen darf und damit Kind des anderen ist. So erlaubt ein Pool nur Lanes als Kinder, und ein Diagram nur Pools.

Abbildung 4.8: Beispiel Enthaltenseinsregel

In der BPMN Stencil Set Spezifikation werden Enthaltenseinsregeln unter der Sektion „containmentRules“ aufgeführt.

Beispiel einer Enthaltenseinsregel:

```
{
  "role":"Pool",
  "contains": [
    "Lane"
  ]
}
```

Diese Regel erlaubt einem Pool, Lanes als Kindelemente zu besitzen.

4.3 Activities

Aktivitäten sind die grundlegenden Bausteine eines Prozessen, denn sie enthalten die Arbeit, die im Prozessfluss ausgeführt wird. Während alle anderen Flussobjekte den Fluss beeinflussen, beschreiben Aktivitäten, was bei Prozessausführung getan wird. Bei Aktivitäten lassen sich in drei verschiedene Arten unterscheiden:

- Process
- Sub-Process
- Task

Ein Prozess ist allerdings kein graphisch dargestelltes Objekt in BPMN. Daher wurde zur Darstellung von Prozessen eine andere Lösung gefunden, die im Abschnitt „Pools“ diskutiert wird. Es bleiben also Task und Sub-Process als Aktivitäten, die im Folgenden nun genau spezifiziert werden.

4.3.1 Task

Eine Task ist eine atomare Aktivität, bei der keine feinere Spezialisierung der Arbeit der Task vom Modellierer gewünscht oder angedacht ist.

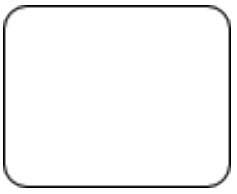

Abbildung 4.9: Task

SVG-Datei (node.task.svg)

Eine Task ist ein Rechteck mit abgerundeten Ecken.

```
<rect
     id="taskrect"
     oryx:resize="vertical horizontal"
     x="0" y="0" width="100" height="80"
     rx="10" ry="10"
     stroke="black" stroke-width="1" fill="white"
/>
```

Im BPMN Stencil Sets des Oryx-Editors sind die Textbeschreibungen von Taskobjekten zentriert in die Task gelegt. Derzeit wird von Editorseite noch kein automatisches Wordwrap unterstützt, da zum Bestimmen der Textlänge der Text zuerst einmal gerendert werden muss, bevor seine Ausdehnung bekannt ist, was Voraussetzung zum Einhalten der Bounds der Task ist.

```
<text
     font-size="14"
     id="acttext"
     x="50" y="40"
     oryx:align="middle center"
     stroke="black">
          Some Task
</text>
```

Die verschiedenen Typen an Tasks wurden zu einer einheitlichen Task vereinigt, welche Informationen zu allen Typen enthält. Ursache für diese Entscheidung war, das dem Modellierer nur eine einzige Task zur Verfügung gestellt werden soll, die allen Anforderungen genügt, wodurch eventuelles Auswechseln zur Designzeit nicht nötig ist. Nachteilig wirkt sich aus, das diese Task die Attribute alle Task-Typen vereinigt und anbietet. Es ist nun an dem Modellierer, zu wissen, welche Properties dann wichtig sind.

Loop, Multiple Instance und Compensation werden über Properties gesteuert und durch die entsprechenden Symbole am unteren Rahmen der Task angezeigt. Um Compensation über ein Property möglich zu machen, wurde die Task um ein Attribut (Boolean) erweitert, welches angibt, ob eine Task eine Compensation Task ist.

Attribute

Eine Task vereinigt die Attribute des Elements selbst mit denen, die für alle Aktivitäten und für alle Flussobjekte gelten und auch den für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id des Elements im Diagramm
Categories	String	nein	ja	Kategorien für das Objekt
Documentation	String	nein	ja	Text Dokumentation zum Objekt
Name	String	nein	ja	Text, bestimmt Beschriftung des Shapes
Pool	String	nein	nein	Angabe des Pools
Lanes	String	nein	ja	Angabe der Lanes
ActivityType : Task	String ja	nein	Typ der Aktivität	
Status	Choice	nein	nein	None | Ready | Active | Cancelled | Aborting | Aborted | Completing | Completed
InputSets	String	nein	ja	Anforderung an Datenobjekten als Input
Inputs	String	nein	ja	Datenobjekte als Input
OutputSets	String	nein	ja	Anforderung an Datenobjekten als Output
Outputs	String	nein	ja	Datenobjekte als Output
IORules	String	nein	ja	Beziehungen zwischen InputSets und OutputSets
StartQuantity	Integer : 1	nein	ja	Anzahl der Tokens zum Start
LoopType	Choice	nein	nein	None | Standart | Multi-Instance
LoopCondition	String	nein	ja	Loop Ausdruck (Boolean)
LoopCounter	Integer : 1	nein	ja	Zählt Durchläufe
LoopMaximum	Integer : 1	nein	ja	0-1
TestTime	Choice	nein	nein	Before | After
MI_Condition	String	nein	ja	Loop Ausdruck (Integer)
MI_Ordering	Choice	nein	nein	Sequential | Parallel
MI_FLowCondition	Choice	nein	nein	None | One | All | Complex
CMI_FLowCondition	String	nein	ja	Loop Ausdruck (Boolean)
isCompensation	Boolean	nein	ja	Compensation oder nicht

TaskType	Choice	nein	nein	Service | Recieve | Send | USer | Script | Manual | Reference | None
InMessage	String	nein	ja	Message
OutMessage	String	nein	ja	Message
Implementation	Choice	nein	nein	Web Service | Other | Unspecified
Message	String	nein	ja	Message
Instantiate : false	Boolean nein	ja	Instantierungsmechanismus	
Performers	String	nein	ja	Durchführende(r) (Mensch)
Script	String	nein	ja	Script, das ausgeführt werden soll
TaskRef	String	nein	ja	Referenzierte Task

Abbildung 4.10: Attribute der Task

Regeln

Task sind Flußobjekte und erlauben daher Sequenzflüsse. Desweiteren können Tasks mit anderen Flußobjekten (außer Gateways) in anderen Pools oder dem anderen Pool selbst über MessageFlows verbunden sein. Über Assoziationen können mit Datenobjekten Verbindungen geschaffen werden.

Magneten

- Oben
- Unten
- Links
- Rechts
- Zentral : Default

4.3.2 Subprocess

Ein Sub-Prozess ist eine Aktivität, die entweder im expandierten oder kollabierten Zustand vorkommt. Sie unterscheidet sich von einer Task, indem Sie einen eigenen Prozess enthält - den sogenannten Sub-Prozess. Man kann sie auch als komplexe Aktivität ansehen, die durch den Modellierer detailierter verfeinert wurde.

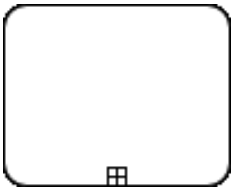

Abbildung 4.11: Sub-Process

SVG-Datei (node.subprocess.svg)

Eine Sub-Prozess ist ein Rechteck mit abgerundeten Ecken. Im kollabierten Zustand hat er ein kleines Rechteck mit einem Plus am unteren Rand zentriert. Im Oryx-Editor sind Subprozesse nur kollabiert vorhanden. Über ein Property, welches auf die Adresse des Sub-Prozesses verweist, ist der Inhalt des Sub-Prozess erreichbar. Sub-Prozesse werden also wie Prozesse behandelt. Wenn ein Sub-Prozess eine Transaction ist, besitzt er eine doppelte Umrahmung. Dies wird durch das doppelte Rechteck in nachfolgendem Code erreicht.

```
<rect
     id="border"
     oryx:resize="vertical horizontal"
     x="0" y="0" width="100" height="80"
     rx="10" ry="10"
     stroke="black" stroke-width="1" fill="none"
/>
<rect
     id="taskrect"
     oryx:resize="vertical horizontal"
     x="2" y="2" width="96" height="76"
     rx="8" ry="8"
     stroke="black" stroke-width="1" fill="white"
/>
```

Im BPMN Stencil Sets des Oryx-Editors sind die Textbeschreibungen von Sub-Prozessen zentriert. Auch hier gibt es derzeit kein automatisches Wordwrap.

```
<text
     font-size="14"
     id="acttext"
     x="50" y="40"
     oryx:align="middle center"
     stroke="black">
          Some Sub-Process
</text>
```

Loop, Multiple Instance AdHoc und Compensation werden über Properties gesteuert und durch die entsprechenden Symbole am unteren Rahmen des Sub-Prozesses angezeigt. Um Compensation und AdHoc über ein Property zu steuern, wurde der Sub-Prozess um je ein Attribut (Boolean) erweitert, welches angibt, ob er ein Compensation/AdHoc Sub-Prozess ist.

Attribute

Ein Sub-Prozess vereinigt die Attribute des Elements selbst mit denen, die für alle Aktivitäten und für alle Flussobjekte gelten und auch den für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung

Id	String	nein	nein	Einheitliche referenzierbare Id des Elements im Diagramm
Categories	String	nein	ja	Kategorien für das Objekt
Documentation	String	nein	ja	Text Dokumentation zum Objekt
Name	String	nein	ja	Text, bestimmt Beschriftung des Shapes
Pool	String	nein	nein	Angabe des Pools
Lanes	String	nein	ja	Angabe der Lanes
ActivityType : Task	String ja	nein	Typ der Aktivität	
Status	Choice	nein	nein	None | Ready | Active | Cancelled | Aborting | Aborted | Completing | Completed
InputSets	String	nein	ja	Anforderung an Datenobjekten als Input
Inputs	String	nein	ja	Datenobjekte als Input
OutputSets	String	nein	ja	Anforderung an Datenobjekten als Output
Outputs	String	nein	ja	Datenobjekte als Output
IORules	String	nein	ja	Beziehungen zwischen InputSets und OutputSets
StartQuantity	Integer : 1	nein	ja	Anzahl der Tokens zum Start
LoopType	Choice	nein	nein	None | Standart | MultiInstance
LoopCondition	String	nein	ja	Loop Ausdruck (Boolean)
LoopCounter	Integer : 1	nein	ja	Zählt Durchläufe
LoopMaximum	Integer : 1	nein	ja	0-1
TestTime	Choice	nein	nein	Before | After
MI_Condition	String	nein	ja	Loop Ausdruck (Integer)
MI_Ordering	Choice	nein	nein	Sequential | Parallel
MI_FLowCondition	Choice	nein	nein	None | One | All | Complex
CMI_FLowCondition	String	nein	ja	Loop Ausdruck (Boolean)
MI_Condition	String	nein	ja	Loop Ausdruck (Integer)
MI_Ordering	Choice	nein	nein	Sequential | Parallel

subProcessType : Reference	String ja	nein	Typ des Sub-Prozesses	
isATransaction : false	Boolean nein	ja	ist eine Transaktion?	
Transaction	String	nein	ja	Transaktion (falls oben true)
SubProcessRef	Uri	nein	ja	Referenz auf den enthaltenen Prozess
isAdHoc	Boolean	nein	ja	AdHoc oder nicht
isCompensation	Boolean	nein	ja	Compensation oder nicht

Abbildung 4.12: Attribute des Sub-Prozesses

Die letzten beiden Attribute sind keine BPMN Attribute, sie bestimmen jedoch auf einfache Art und Weise, ob der Sub-Prozess adHoc ausgeführt wird und ob er ein Kompensations-Sub-Prozess ist. Es fehlen hier die spezifischen Attribute für eingebettete Sub-Prozesse, da es diese in Oryx nicht gibt.

Regeln

Auch ein Sub-Prozess ist ein Flussobjekt und damit durch Sequenzfluss verbindbar. Desweiteren können Sub-Prozesse mit anderen Flussobjekten (außer Gateways) in anderen Pools oder dem anderen Pool selbst über MessageFlows verbunden sein. Über Assoziationen können - wie bereits bei Tasks erwähnt - mit Datenobjekten Verbindungen geschaffen werden.

Magneten

- Oben
- Unten
- Links
- Rechts
- Zentral : Default

4.4 Gateways

Gateways kontrollieren den Sequenzfluss, indem sie den Fluss aufsplitten oder zusammenführen. BPMN erlaubt es, ein Gateway sowohl zum Zusammenführen als auch zum Aufteilen gleichzeitig zu verwenden, folglich kann ein Gateway mehrere ein- und ausgehende Kanten zugleich haben. Dieses Feature wird im BPMN Stencil Set ohne Einschränkung unterstützt. In BPMN gibt es fünf verschiedene Typen:

- Data-Based Exclusive (XOR)
- Event-Based Exclusive (XOR)
- Inclusive (OR)
- Complex
- Parallel (AND)

Die Implementation der Gateways hat zwei grundlegende Lösungsmöglichkeiten angeboten, zum einen, dass man nur ein Gateway-Typ anbietet (standartmäßig Data-Based XOR), der durch das Attribut „GatewayType“ dann nachträglich gesetzt bzw geändert werden kann und dabei die graphische Repräsentation des Stencils verändert. Dagegen sprach jedoch die große Anzahl von Attributen, die nicht als gemeinsame Attribute für alle Gateways gültig waren, sondern nur exklusiv für einen einzigen Typen. Die große Diversität bzw. Unvereinbarkeit der Properties verbunden mit der speziellen Regel, das nur an Data-Based Exclusive Gateways ein Default Flow beginnen darf, führten dazu, dass die Gateways als einzelne Stencils realisiert wurden.

4.4.1 Exclusive (XOR)

Ein XOR Gateway ist ein Gateway, das als Decision Point einen Sequenzfluss nur an einen Weg weitergibt. Dabei werden Ausdrücke („Gates“) an die Ausgänge gebunden. Im Falle des Ausführung bestimmt also die zutreffende Condition Expression, welcher Gate geschaltet wird, womit der weitere Sequenzfluss vorbestimmt wird. Es gibt zwei Subtypen, die nachfolgend behandelt werden.

4.4.1.1 Data-Based

Abbildung 4.13: Data-Based XOR

Das Data-Based Exlusive Gateway verwendet boolsche Ausdrücke in den Gates, wodurch ein exklusiver Weg für den Sequenzfluss gefunden wird. Wie diese Ausdrücke allerdings auszusehen haben, wird in BPMN nicht spezifiziert. Ebenso gibt es auch keine Aussage, was passieren soll, wenn kein gültiges Gate bei einer Ausführung gefunden werden kann. Da allerdings kein impliziter Fluss erfolgen darf, so darf man Exlusive Gateways ohne gültiges Gate als ungültiges Modell betrachten. Data-Based XOR Gateways nutzen als Ausdrücke Informationen aus dem Prozess, sogenannte Prozessdaten (process data), darin unterscheiden sie sich auch von den Event-Basierten XOR Gateways. Es wird immer der erste zutreffende Ausdruck gewählt, der das Gate freigibt zum Sequenzfluss. Default Alternativen sind nicht erforderlich, wohl aber möglich. Ebenso kann ein Gateway diesen Types den Sequenzfluss nicht nur teilen (Decision), sondern auch wieder zusammenführen (Merge).

SVG-Datei (node.gateway.xor.databased.svg)

Ein Data-Based Exclusive Gateway besteht aus einem auf der Ecke stehenden Quadrat. Es gibt zwei Repräsentationen: Eine ist auf dem Bild oben ersichtlich, aber auch ohne das Kreuz in der Mitte ist es ein Data-Based XOR Gateway.

Abbildung 4.14: Data-Based XOR Gateway ohne Kreuz

Das Umschalten zwischen beiden graphischen Repräsentationen geschieht über das Attribut „MarkerVisible“. Das Kreuz selbst wird aus zwei gefüllten Pfaden beschrieben.

```
<path
     id="frame"
     fill="white" stroke-width="1" stroke="black"
     d="M 1,16 L 16,1 L 31,16 L 16,31 L 1,16"
/>
```

Die Anzeige von Text am Gateway ist im BPMN Stencil Set nicht vorgesehen.

Attribute

Ein Data-Based Exclusive Gateway vereinigt die Attribute des Elements selbst mit denen, die für alle Gateways und für alle Flussobjekte gelten und auch den für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id des Elements im Diagramm
Categories	String	nein	ja	Kategorien für das Objekt
Documentation	String	nein	ja	Text Dokumentation zum Objekt
Pool	String	nein	nein	Angabe des Pools
Lanes	String	nein	ja	Angabe der Lanes
GatewayType	String : XOR	ja	nein	Typ des Gateway
XORType	String : Data	ja	nein	Typ des XOR Gateway
MarkerVisible : false	Boolean nein	ja	Kreuz angezeigt/ausgeblendet	

Gates	String	nein	ja	Gates
[Gates] OutgoingSequenceFlow	String	nein	ja	Assoziierter Sequenzfluss
[Gates] Assignments	String	nein	ja	Assignment expression
DefaultGate	String	nein	ja	DefaultGate
[Gate] OutgoingSequenceFlow	String	nein	ja	Assoziierter Sequenzfluss
[Gate] Assignments	String	nein	ja	Assignment expression

Abbildung 4.15: Attribute des Data-Based Exclusive Gateway

Die Umsetzung von dem Gate-Sequenceflow-Konzept ist nicht trivial, doch sind, wie beim Mapping in Abschnitt 4.1 exemplarisch erklärt, die Aufzählungen der Gates als Liste vorgesehen, so dass ein Multiline-String das Attribut am besten wiedergibt.

Regeln

Ein Data-Based Exclusive Gateway wird über Sequenzfluss mit anderen Flußobjekten verbunden. Als spezielle Form von Sequenzfluss sei der DefaultFlow erwähnt, der einem Gate zugehörig sein darf. Message Flows von oder zu Gateways sind nicht erlaubt.

Magneten

- Zentral : Default

4.4.1.2 Event-Based

Abbildung 4.16: Event-Based XOR

Das Event-Based Exlusive Gateway unterscheidet sich vom Data-Based XOR Gateway in der Ausgangsseite: Das Finden eines passenden Gates, der geschalten wird, bestimmt sich durch das Auftreten von Events (häufig: MessageEvent). Modelliert wird das, indem an die ausgehenden Sequenzflüsse eines Event-Based Exlusive Gateway IntermediateEvents des entsprechenden Typs gebunden werden. Zusätzlich können individuelle Events durch eine Task vom Typ „Recieve“ herangezogen werden. Es ist - wie beim Data-Based Exclusive Gateway - nur ein Sequenzfluss aus dem Gateway möglich.

SVG-Datei (node.gateway.xor.eventbased.svg)

Ein Event-Based Exclusive Gateway besteht aus einem auf der Ecke stehenden Quadrat mit einem Intermediate Multiple Event innerhalb des Rahmens (6-zackiger Stern in zwei einfachen Kreisen).

```
<path
     id="frame"
     fill="white" stroke-width="1" stroke="black"
     d="M 1,16 L 16,1 L 31,16 L 16,31 L 1,16" />
<circle
     cx="16" cy="16" r="8"
     stroke="black" fill="none" stroke-width="0.5"/>
<circle
     cx="16" cy="16" r="9"
     stroke="black" fill="none" stroke-width="0.5"/>
<polygon
     stroke="black" fill="black" stroke-width="1"
     points="16,10 18,13 21.5,13 19.5,16 21.5,19 18,19
     16,22 14,19 10.5,19 12.5,16 10.5,13 14,13"
     stroke-linecap="butt" stroke-linejoin="miter" stroke-miterlimit="10" />
```

Die Anzeige von Text am Gateway ist im BPMN Stencil Set nicht vorgesehen.

Attribute

Ein Event-Based Exclusive Gateway vereinigt die Attribute des Elements selbst mit denen, die für alle Gateways und für alle Flussobjekte gelten und auch den für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id des Elements im Diagramm
Categories	String	nein	ja	Kategorien für das Objekt
Documentation	String	nein	ja	Text Dokumentation zum Objekt
Pool	String	nein	nein	Angabe des Pools
Lanes	String	nein	ja	Angabe der Lanes
GatewayType	String : OR	ja	nein	Typ des Gateway
Gates	String	nein	ja	Gates
[Gates] OutgoingSequenceFlow	String	nein	ja	Assoziierter Sequenzfluss
[Gates] Assignments	String	nein	ja	Assignment expression
DefaultGate	String	nein	ja	DefaultGate
[Gate] OutgoingSequenceFlow	String	nein	ja	Assoziierter Sequenzfluss
[Gate] Assignments	String	nein	ja	Assignment expression

Abbildung 4.17: Attribute des Event-Based Exclusive Gateway

Regeln

Ein Event-Based Exclusive Gateway - wird wie alle Gateways - über Sequenzfluss mit anderen Flußobjekten verbunden. Message Flows von oder zu Gateways sind verboten.

Magneten

- Zentral : Default

4.4.2 Inclusive (OR)

Dieses Gateway funktioniert genauso wie ein XOR-Gateway, mit dem Unterschied, dass die Überprüfung der Condition Expressions der vorhandenen Gates nicht bei einer erfolgreichen Auswertung abgebrochen werden, sondern weitergeführt werden, was zur Folge hat, dass mehrere ausgehende Gates an die anliegenden Sequenzflüsse Token weitergeben können.

Abbildung 4.18: Inclusive (OR)

SVG-Datei (node.gateway.or.svg)

Ein Inclusive Gateway besteht aus einem auf der Ecke stehenden Quadrat mit einem Kreis innerhalb des Rahmens, welcher fettgedruckt ist.

```
<path
     id="frame"
     fill="white" stroke-width="1" stroke="black"
     d="M 1,16 L 16,1 L 31,16 L 16,31 L 1,16" />
<circle
     cx="16" cy="16" r="7.5"
     stroke="black" fill="none" stroke-width="2.5" />
```

Attribute

Ein Inclusive Gateway vereinigt die Attribute des Elements selbst mit denen, die für alle Gateways und für alle Flussobjekte gelten und auch den für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id des Elements im Diagramm
Categories	String	nein	ja	Kategorien für das Objekt

Documentation	String	nein	ja	Text Dokumentation zum Objekt
Pool	String	nein	nein	Angabe des Pools
Lanes	String	nein	ja	Angabe der Lanes
GatewayType	String : OR	ja	nein	Typ des Gateway
Gates	String	nein	ja	Gates
[Gates] OutgoingSequenceFlow	String	nein	ja	Assoziierter Sequenzfluss
[Gates] Assignments	String	nein	ja	Assignment expression
DefaultGate	String	nein	ja	DefaultGate
[Gate] OutgoingSequenceFlow	String	nein	ja	Assoziierter Sequenzfluss
[Gate] Assignments	String	nein	ja	Assignment expression

Abbildung 4.19: Attribute des Inclusive Gateway (OR)

Regeln

Ein Inclusive Gateway kann zu anderen Flussobjekten über Sequenzflüsse eine Verbindung haben. Was verwunderlich ist in der Spezifikation, dass auf einem Diagramm an einem Inclusive Gateway ein DefaultFlow angezeigt wird, obgleich dies nach der Verbindungsspezifikation des DefaultFlows selbst nur an Data-Based XOR erlaubt sein soll. Message Flows von oder zu Gateways sind nicht erlaubt.

Magneten

- Zentral : Default

4.4.3 Complex

Complex Gateways werden eingesetzt, wenn die Entscheidung, welche Gates zum Schalten geöffnet werden, dem Modellierer nach nicht trivial ist. Es sollte nach Möglichkeit nur an eine ausgehende Sequence Flow ein Token weitergegeben werden, welches durch Auswerten der Prozessdaten und des Status des eingehenden Sequenzflusses bestimmt wird.

Abbildung 4.20: Complex

SVG-Datei (node.gateway.complex.svg)

Ein Complex Gateway besteht aus einem auf der Ecke stehenden Quadrat mit einem achtstrahligen Asterisk innerhalb des Rahmens.

```
<path
     id="frame"
     fill="white" stroke-width="1" stroke="black"
     d="M 1,16 L 16,1 L 31,16 L 16,31 L 1,16" />
<path
     fill="white" stroke-width="3" stroke="black"
     d="M 8.5,16 L 23.5,16
        M 16,8.5 L 16,23.5
        M 10.5,10.5 L 21.5,21.5
        M 10.5,21.5 L 21.5,10.5" />
```

Attribute

Ein Complex Gateway vereinigt die Attribute des Elements selbst mit denen, die für alle Gateways und für alle Flussobjekte gelten und auch den für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id des Elements im Diagramm
Categories	String	nein	ja	Kategorien für das Objekt
Documentation	String	nein	ja	Text Dokumentation zum Objekt
Pool	String	nein	nein	Angabe des Pools
Lanes	String	nein	ja	Angabe der Lanes
GatewayType	String : Complex	ja	nein	Typ des Gateway
Gates	String	nein	ja	Gates
[Gates] OutgoingSequenceFlow	String	nein	ja	Assoziierter Sequenzfluss
[Gates] Assignments	String	nein	ja	Assignment expression
IncomingCondition	String	nein	nein	Bedingung des eingehenden Flusses
OutgoingCondition	String	nein	nein	Bedingung des ausgehenden Flusses

Abbildung 4.21: Attribute des Complex Gateways

Regeln

Ein Complex Gateway wird über Sequence Flow mit anderen Flussobjekten verbunden. Verbindungen mit Message Flows sind nicht erlaubt.

Magneten

- Zentral : Default

4.4.4 Parallel (AND)

Abbildung 4.22: Parallel

Mit Hilfe eines Parallel Gateways lassen sich zwei Flüsse synchronisieren. Dieses Gateway überprüft keine Ausdrücke, nach denen es die ausgehenden Gates überprüft, um festzulegen, über welche Ausgänge ein Token weitergeleitet wird, sondern es gibt beim Schalten an jeden ausgehenden Sequenzfluss ein Token weiter. Im Falle eines Joining Gateways erwartet es aus jeder eingehenden Kante, dass diese ein Token liefert, bevor es ein Token weitergibt. Bekommt es nicht aus allen eingehenden Sequence Flows ein Token, ist es blockiert.

SVG-Datei (node.gateway.and.svg)

Ein Parallel (AND) Gateway besteht aus einem auf der Ecke stehenden Quadrat mit einem Pluszeichen im Rahmen.

```
<path
     id="frame"
     fill="white" stroke-width="1" stroke="black"
     d="M 1,16 L 16,1 L 31,16 L 16,31 L 1,16" />
<path
     fill="white" stroke-width="3" stroke="black"
     d="M 8.5,16 L 23.5,16 M 16,8.5 L 16,23.5" />
```

Attribute

Ein Parallel Gateway vereinigt die Attribute des Elements selbst mit denen, die für alle Gateways und für alle Flussobjekte gelten und auch den für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id des Elements im Diagramm
Categories	String	nein	ja	Kategorien für das Objekt
Documentation	String	nein	ja	Text Dokumentation zum Objekt
Pool	String	nein	nein	Angabe des Pools
Lanes	String	nein	ja	Angabe der Lanes
GatewayType	String : AND	ja	nein	Typ des Gateway
Gates	String	nein	ja	Gates
[Gates] OutgoingSequenceFlow	String	nein	ja	Assoziierter Sequenzfluss
[Gates] Assignments	String	nein	ja	Assignment expression

Abbildung 4.23: Attribute des AND Gateways

Regeln

AND Gateways werden über Sequence Flow mit anderen Flussobjekten verbunden. Verbindungen mit Message Flows sind nicht erlaubt.

Magneten

- Zentral : Default

4.5 Events

Events sind Geschehnisse im Laufe einer Prozessausführung, die den Fluss im Process beeinflussen. So kann auf ein Event hin die Prozessausführung beginnen, abbrechen oder auch terminieren. Events können sehr vielfältiger Natur sein, es kann das Eintreffen einer Nachricht sein, das Ablaufen einer Zeitfrist oder das Auftreten eines Fehlers.

Prinzipiell unterscheidet man drei Arten von Events (Flow Dimension), abhängig von ihrem Auftreten im Prozess.

- Start Event
- Intermediate Event
- End Event

Ein Start Event lässt die Ausführung des Prozesses beginnen, ein Intermediate Event tritt irgendwo zwischen Start- und End Event auf und beeinflusst zwar den Prozess, ohne aber ihn zu starten oder zu beenden, und ein End Event beendet jeden Prozessfluss. Bei Start und Intermediate Events existiert ein sogenannter „Trigger“, ein Auslöser, welcher auf das Ereignis reagiert, bei einem End Event ist das Ereignis eine Folge des Events. So wird ein Start Message Event durch den Empfang einer Nachricht ausgelöst, während ein End Message Event eine Nachricht beim Beenden sendet.

Es gibt folgende Typen von Events (Type Dimension), die hier unabhängig von der Art des Events aufgelistet sind:

None

Der Modellierer muss den Typ des Events nicht zwingend angeben. Außerdem findet dieses Event Verwendung bei Sub-Prozessen, wo die Ausführung des Sub-Prozesses durch seinen Elternprozess ausgelöst, der Zustand sich geändert oder der Sub-Prozess wieder verlassen (terminiert) wird. In diesem Stencil Set ist letzteres Argument allerdings nicht relevant, da es keine expandierten Subprozesse gibt.

Abbildung 4.24: None

Message

Der Empfang einer Nachricht kann auch ein auftretendes Event sein, dass während oder zu Beginn einer Prozessausführung stattfindet. Im Falle eines End-Events wird als Ergebnis eine Nachricht an einen Teilnehmer gesendet.

Abbildung 4.25: Message

Timer

Dieses Event wird durch einen Zeitpunkt oder eine Zeitdauer ausgelöst und kann den Prozess starten. Als Intermediate Event wird es als Verzögerungsmechanismus eingesetzt.

Abbildung 4.26: Timer

Error

Ein Error Event wird verwendet, wenn ein Fehler entdeckt oder geworfen werden soll. Es kann auf benannte Fehler als auch auf unspezifizierte Fehler reagieren.

Abbildung 4.27: Error

Cancel

Dieses Event kommt nur in Sub-Prozessen vor, die Transaktionen sind. Dabei wird es an den Rahmen des Sub-Prozesses als sogenanntes Scope-Event angehangen. Zweck ist es, auf Cancel Events in einer Transaktion zu reagieren, die durch ein Transaktionsprotokoll geworfen werden kann.

Abbildung 4.28: Cancel

Compensation

Zum Compensation Handling wird dieses Event verwendet. Auf den Rahmen einer Aktivität gesetzt, wird es mit einer Compensation Assoziation mit einer Compensations Task verbunden, die das Event kompensiert.

Abbildung 4.29: Compensation

Rule

Dieses Event wird geworfen, wenn ein Ausdruck aus den Prozessdaten, der für das Event als Expression definiert wurde, wahr wird. Verwendung findet es bei Exception Handling

Abbildung 4.30: Rule

Link

Ein Link Event dient als Anzeiger dafür, dass der Prozess woanders fortgeführt wird. Als Startevent bedeutet dies, dass der Prozess in einem anderen Diagramm beginnt und hier nun fortgesetzt wird, als Endevent bedeutet es, dass der aktuelle Prozess an einem anderen Ort fortgeführt wird. Üblicherweise wird er eingesetzt, um Übersichtlichkeit zu wahren, oder wenn der Rahmen des Diagrams begrenzt ist und der Prozess nicht komplett ins Diagramm passt.

Abbildung 4.31: Link

Multiple

Ein Multiple Event vereinigt mehrere Möglichkeiten, ausgelöst zu werden, wobei ein Ereignis genügt. Im Falle eines End Events bedeutet es, dass viele Folgen aus dem End Event folgen (z.b. das Werfen einer Exception und das Senden einer Nachricht).

Abbildung 4.32: Multiple

Terminate

Das Terminate Event beendet einen Prozess in seiner Ausführung sofort. Es wird kein Compensation oder Error Handling durchgeführt.

Abbildung 4.33: Terminate

Im BPMN Stencil Set des Oryx-Editors ist für jedes Event ein eigenes Stencil verfügbar, das das Event beschreibt. Sortiert sind die Event Stencils im Repository nach ihrem Typ als Start, Intermediate oder End Event. Ursache für diese strikte Trennung ist unter anderem die Menge an unterschiedlichen Attributen, die Attribute von Stencils mit verschiedenen „Type Dimensions“ haben. Vor allem aber sollte der Benutzer die Möglichkeit bekommen, ein Event a priori mit einem bestimmten Typ zu erzeugen, anstatt es im Nachhinein durch ein Property ändern zu müssen. Die Trennung nach der „Flow Dimension“ ist dabei allerdings natürlich, denn die Attribute weichen zu stark ab, zudem möchte der Benutzer im vornherein wissen, welche Art von Event er nun bekommt. Eine intuitive, aber derzeit nicht unterstützte Lösungsmöglichkeit wird im nächsten Kapitel diskutiert.

Magneten

- Center : Default

4.5.1 Start Event

Ein Start Event zeigt an, dass hier der Prozess beginnt. Je nach Eventtyp sind dazu bestimmte Auslöser notwendig. Wird es ausgelöst, wirft es ein Token in den ausgehenden Sequenzfluss. Ein Prozess muss allerdings kein Start Event haben, um gestartet werden zu können.

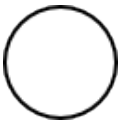

Abbildung 4.34: Start

Mögliche Eventtypen bei Start Events:

- None
- Message
- Timer
- Rule
- Link
- Multiple

SVG-Datei (node.event.[typ].start.svg)

Ein Start Event besteht aus einem einfachen Kreis. Das Symbol im Kreis gibt Aufschluss über den Typ des Events.

```
<circle
     id="frame" cx="16" cy="16" r="15"
     stroke="black" fill="white" stroke-width="1"
/>
```

Attribute

Start Events vereinigen die Attribute des Elements selbst mit denen, die für alle (Start) Events und für alle Flussobjekte gelten und auch den für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id des Elements im Diagramm
Categories	String	nein	ja	Kategorien für das Objekt
Documentation	String	nein	ja	Text Dokumentation zum Objekt
Pool	String	nein	nein	Angabe des Pools
Lanes	String	nein	ja	Angabe der Lanes
EventType	String : Start	ja	nein	Definition als Start Event
Trigger	Choice : None	nein	nein	None | Message | Timer | Rule | Link | Multiple
Message	String	nein	ja	Message Event: Nachricht
Implementation	Choice	nein	nein	Message Event: Technologie zum Erhalt der Message
TimeDate	Date	nein	ja	Timer Event: Zeitpunkt, wann Event getriggert wird
TimeCycle	String	nein	ja	Timer Event: Zeitspanne, wann Event getriggert wird
RuleName	String	nein	ja	Rule Event: Expression
LinkId	String	nein	ja	Link Event: Id für die Referenz
ProcessRef	String	nein	ja	Link Event: Prozess-Referenz
Triggers	String	nein	ja	Multiple Event: Angabe mehrerer Trigger

Abbildung 4.35: Attribute der Start Events

Regeln

Start Events haben keine eingehenden Sequenzflüsse, nur ein ausgehender Sequenzfluss ist gestattet. Ausgehende Message Flows sind ebenso nicht gestattet, erreichen dürfen Message Flows ein Start Event nur ausgehend von Pools, Aktivitäten und End Events.

4.5.2 Intermediate Event

Ein Intermediate Event tritt zwischen Start und End Event im Laufe des Prozesses auf. Es beeinflusst zwar den Fluß, aber startet und beendet den Prozess an sich selbst nicht. Intermediate Events können Exception Handling betreiben, Nachrichten verschicken und empfangen oder Verzögerungen anbieten.

Abbildung 4.36: Intermediate

Eine besondere Form des Intermediate Events tritt bei Compensation Handling auf. Hier wird ein Intermediate Event auf den Rahmen einer Aktivität gelegt. In diesem Fall nimmt das Intermediate Event eine Zwitterrolle ein, nämlich die einer Kante und eines Knotens. Im BPMN Stencil Set wurde dies umgesetzt, indem ein Intermediate Event nicht nur einen Magnet besitzt, sondern darüberhinaus auch einen zentralen Docker, mit dessen Hilfe ein Intermediate Event an den Rahmen einer Aktivität angedockt werden kann. Dementsprechend müssen dafür auch Regeln geschaffen werden, die unter 4.2 erläutert werden.

Mögliche Eventtypen bei Intermediate Events:

- None
- Message
- Timer
- Error
- Cancel
- Compensation
- Rule
- Link
- Multiple

SVG-Datei (node.event.[typ].intermediate.svg)

Ein Intermediate Event besteht aus einem doppelten Kreis. Das Symbol im Kreis gibt, wie bereits bekannt, Aufschluss über den Typ des Events.

```
<circle
    id="frame" cx="16" cy="16" r="15"
    stroke="black" fill="white" stroke-width="1"
/>
<circle
    id="inner" cx="16" cy="16" r="12"
    stroke="black" fill="none" stroke-width="1"
/>
```

Attribute

Intermediate Events vereinigen die Attribute des Elements selbst mit denen, die für alle (Intermediate) Events und für alle Flussobjekte gelten und auch den für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id des Elements im Diagramm
Categories	String	nein	ja	Kategorien für das Objekt
Documentation	String	nein	ja	Text Dokumentation zum Objekt
Pool	String	nein	nein	Angabe des Pools
Lanes	String	nein	ja	Angabe der Lanes
EventType	String : Intermediate	ja	nein	Definition als Intermediate Event
Trigger	Choice : Message	nein	nein	None | Message | Timer | Error | Cancel | Compensation | Rule | Link | Multiple
Target	String	nein	ja	Aktivität, an die das Event „attached“ wird
Message	String	nein	ja	Message Event: Nachricht
Implementation	Choice	nein	nein	Message Event: Technologie zum Erhalt der Message
TimeDate	Date	nein	ja	Timer Event: Zeitpunkt, wann Event getriggert wird

TimeCycle	String	nein	ja	Timer Event: Zeitspanne, wann Event getriggert wird
ErrorCode	String	nein	ja	Error Event: Fehlernachricht
Activity	String	nein	ja	Compensation Event: Kompensations-aktvität
RuleName	String	nein	ja	Rule Event: Expression
LinkId	String	nein	ja	Link Event: Id für die Referenz
ProcessRef	String	nein	ja	Link Event: Prozess-Referenz
Triggers	String	nein	ja	Multiple Event: Angabe mehrerer Trigger

Abbildung 4.37: Attribute der Intermediate Events

Regeln

Intermediate Events sind im Sequenzfluss frei, sowohl ein- als auch ausgehende Kanten sind dabei erlaubt. Ausgehende Message Flows sind bei Intermediate Events nicht gestattet, erreichen dürfen Message Flows nur ausgehend von Pools, Aktivitäten und End Events.

4.5.3 End Event

Ein End Event zeigt an, dass der Prozess hier in seiner Ausführung endet. Ein ankommendes Token wird hier entfernt, jeder Sequenzfluss ist hier zuende.

Abbildung 4.38: End

Mögliche Eventtypen bei End Events:

- None
- Message
- Error
- Cancel
- Compensation
- Link
- Multiple
- Terminate

SVG-Datei (node.event.[typ].end.svg)

Ein End Event besteht aus einem Kreis mit dickem Rand. Wie gehabt wird der Typ des Events durch ein Symbol im Kreis angezeigt.

```
<circle
    id="frame" cx="16" cy="16" r="14"
    stroke="black" fill="white" stroke-width="3"
/>
```

Attribute

End Events vereinigen die Attribute des Elements selbst mit denen, die für alle (End) Events und für alle Flussobjekte gelten und auch den für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id des Elements im Diagramm
Categories	String	nein	ja	Kategorien für das Objekt
Documentation	String	nein	ja	Text Dokumentation zum Objekt
Pool	String	nein	nein	Angabe des Pools
Lanes	String	nein	ja	Angabe der Lanes
EventType	String : Start	ja	nein	Definition als End Event
Result	Choice	nein	nein	None | Message | Error | Cancel | Compensation | Link | Multiple | Terminate
Message	String	nein	ja	Message Event: Nachricht
Implementation	Choice	nein	nein	Message Event: Technologie zum Erhalt der Message
ErrorCode	String	nein	ja	Error Event: Fehlernachricht
Activity	String	nein	ja	Compensation Event: Kompensations-aktvität
LinkId	String	nein	ja	Link Event: Id für die Referenz
ProcessRef	String	nein	ja	Link Event: Prozess-Referenz
Triggers	String	nein	ja	Multiple Event: Angabe mehrerer Trigger

Abbildung 4.39: Attribute der End Events

Regeln

End Events haben nur eingehende Sequence Flows, Ausgehende sind nicht erlaubt. Ein End Event darf keine eingehenden Message Flows besitzen, wohl aber selbst Message Flow Verbindungen zu Pools, Aktivitäten, Start und Intermediate Events haben.

4.6 Artifacts

Artifacts sind jene Elemente in BPMN, welche zusätzliche Informationen anzeigen, die keinen direkten Bezug zum Informationsfluss haben. Aus diesem Grund darf ein Artifact weder durch ein MessageFlow noch durch einen SequenceFlow mit irgendeinem anderen Element verbunden sein. In BPMN gibt es drei verschiedene Typen:

- Group
- Text Annotation
- Data Object

Nachfolgend wird die Umsetzung der drei Artifact-Typen beschrieben.

4.6.1 Group

Das Group Element kann eingesetzt werden, um Prozesselemente graphisch zu gruppieren. Das Besondere dabei ist, dass Groups auch Pool-übergreifend verwendet werden können. Somit gelten auch keine Restriktionen durch Swimlane Elemente.

Abbildung 4.40: Group

SVG-Datei (node.group.svg)

Eine Groupbox besteht aus einem Rechteck mit einer einfachen Strichpunkt-Linie als Begrenzung, welche an den Ecken abgerundet ist. Der Hintergrund eins Group Artefakts ist transparent. Sie sind frei skalierbar.

```
<rect id="c"
      oryx:resize="vertical horizontal"
      x="0" y="0"
      width="120" height="90"
      rx="5" ry="5"
      stroke="black" stroke-dasharray="2, 4, 10, 4" stroke-width="1"
      fill="none"
/>
```

Die BPMN Spezifikation [1] beschreibt explizit nicht die Verwendung von Beschriftungen bei Group Artefakten, jedoch finden sich in derselben Abbildungen mit Text am oberen Rand. Im BPMN Stencil Set wird der optional zu vergebende Name oben linksbündnig in der Box angezeigt.

```
<text font-size="14" id="text" x="2" y="2"
      oryx:align="top left" stroke="black">
           Some Group
</text>
```


Abbildung 4.41: Group Beschriftung

Attribute

Ein Group Artefakt vereinigt die Attribute des Elements selbst mit denen, die für alle Artefakte gelten und auch den für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id des Elements im Diagramm
Categories	String	nein	ja	Kategorien für das Objekt
Documentation	String	nein	ja	Text Dokumentation zum Objekt
ArtifactType : Group	String	ja	nein	Spezifiziert den Typ des Artefaktes
Pool	String	nein	ja	Angabe des Pools, um die Lage des Artefakts zu lokalisieren
Lanes	String	nein	ja	Angabe der Lanes
Name	String	nein	ja	Angabe des Namens, bestimmt Beschriftung des Shapes (siehe oben)

Abbildung 4.42: Attribute der Group

Das Ändern des „Name“-Attributes wirkt sich auf die graphische Repräsentation aus, indem der Wert als Beschriftung des Objektes innerhalb des Rahmens oben linksbündig angezeigt wird.

Regeln

Ein Group Element ist ein Artifact, womit jede Verbindung mit anderen Objekten durch Message Flow oder Sequence Flow nicht erlaubt ist. Somit sind der Spezifikation nach nur Assoziationen zugelassen. Einzig üblich ist die Verbindung einer Group mit einer Textannotation durch eine ungerichtete Assoziation, wobei die Textannotation als Kommentar der Gruppe zu sehen ist.

Magneten

Keine

4.6.2 Text Annotation

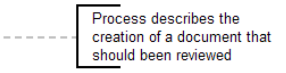

Abbildung 4.43: Text Annotation

Textannotationen sind Kommentarobjekte in BPMN. Ihre Funktion besteht darin, Anmerkungen und Notizen in Diagramme einzubringen. Der Text kann dabei auch fern von dem Punkt im Diagramm, auf den sich der Kommentar bezieht, angelegt werden, denn stets zeigt eine Assoziation auf das betreffende Objekt oder in dessen Richtung. Textannotationen sind absolut frei in ihrem Vorkommen und können daher an jeder beliebigen Stelle im Diagramm eingesetzt werden.

SVG-Datei (node.text.annotation.svg)

Eine Textannotation wird in BPMN oft als Verbindung von der Box und ihrer Beschriftung sowie der assoziierten Verbindung betrachtet - quasi als zusammengesetztes Element. Im BPMN Stencil Set wird dies strikt getrennt, weil die Assoziation einer Textannotation den gleichen Regeln folgt und dieselben Attribute hat wie jede normal auftretende Assoziation und somit nicht in die Annotation integriert werden sollte.

Somit besteht eine Textannotation aus einem rechts offenen Rechtecks, quasi in Form einer großen, eckigen Klammer.

```
<rect
     id="taskrect"
     oryx:resize="vertical horizontal"
     x="1" y="1"
     width="100" height="50"
     stroke="none"
     fill="none"
/>
<path
     d="M15,1 L1,1 L1,50 L15,50"
     oryx:resize="vertical horizontal"
     stroke="black" stroke-width="1"
     fill="none"
/>
```

Der Text wird in der Öffnung des rechts geöffneten Rechtecks linksbündig angezeigt, wobei mehrzeilige Texte möglich sind.

```
<text
     font-size="14" id="text" x="5" y="20"
     oryx:align="middle left" stroke="black">
          Some comment for this process
</text>
```

Attribute

Eine Textannotation vereinigt die Attribute des Elements selbst mit denen, die für alle Artefakte gelten und auch den für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id des Elements im Diagramm
Categories	String	nein	ja	Kategorien für das Objekt
Documentation	String	nein	ja	Text Dokumentation zum Objekt
ArtifactType : TextAnnotation	String	ja	nein	Spezifiziert den Typ des Artefaktes
Pool	String	nein	ja	Angabe des Pools, um die Lage des Artefakts zu lokalisieren
Lanes	String	nein	ja	Angabe der Lanes
Text	String	nein	ja	Text, bestimmt Beschriftung (Kommentar) des Shapes (siehe oben)

Abbildung 4.44: Attribute der Text Annotation

Das Ändern des „Text“-Attributes wirkt sich auf die graphische Repräsentation aus, indem der Wert als Kommentarbeschriftung des Objektes innerhalb des Rahmens linksbündig angezeigt wird.

Regeln

Auch die Textannotation ist ein Artifact, womit jede Verbindung mit anderen Objekten durch Message Flow oder Sequence Flow nicht erlaubt ist; nur Assoziationen sind zugelassen. Mit Hilfe von einer einzelnen, ungerichteten Assoziation kann eine Textannotation mit allen anderen Shapes verbunden werden oder in ihre Richtung zeigen.

Magneten

- Links : Default

4.6.3 Data Object

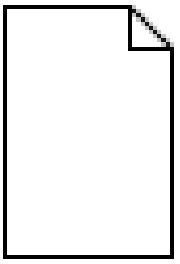

Abbildung 4.45: Data Object

Datenobjekte (DataObject) sind Informationen, die an einen Prozess assoziiert sind. Sie können als Eingangs/Ausgangs-informationen für Aktivitäten dienen oder als angehangene („attached“) Informationen an Sequenzflüssen. Im ersten Fall kann ein Datenobjekt als eingehende oder ausgehende Information angesehen werden, die in der Ausführung einer Task verwendet wird, oder aber auch als Informationsspeicher oder Wissenspool, auf das eine Aktivität zurückgreifen kann. In jedem Fall ist ein Datenobjekt nur eine zusätzliche Information, dass heißt, ein Prozess lässt sich auch ohne Datenobjekte modellieren und somit Unordnung bei komplexen Prozessen vermeiden. Da sie - wie bereits erwähnt - den Prozessfluss nicht beeinflussen, werden sie mit Assoziationen an Flussobjekte und Konnektoren angehangen.

SVG-Datei (node.data.object.svg)

Ein Datenobjekt wird in einem Business Process Diagram als Rechteck dargestellt, das eine nach vorn geknickte, rechte obere Ecke besitzt. Es soll symbolhaft ein Blatt Papier imitiert werden.

```
<polygon
      id="frame"
      stroke="black" stroke-width="1"
      points="0,0 0,60 40,60 40,10 30,0 30,10 40,10 30,0"
      fill="white"
/>
```

Die Lage des Textes ist, wie so oft in der BPMN Spezifikation [1], nicht explizit festgesetzt, doch sind Datenobjekte in Beispielen oft mit Text unter dem graphischen Objekt dargestellt. Im BPMN Stencil Sets des Oryx-Editors sind die Textbeschreibungen von Datenobjekten zentriert in das Datenobjekt gelegt.

```
<text
     font-size="14"
     id="text"
     x="20" y="30"
     oryx:align="middle center"
     stroke="black">
          Some Data
</text>
```

Attribute

Das Datenojekt vereinigt die Attribute des Elements selbst mit denen, die für alle Artefakte gelten und auch den für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung

Id	String	nein	nein	Einheitliche referenzierbare Id des Elements im Diagramm
Categories	String	nein	ja	Kategorien für das Objekt
Documentation	String	nein	ja	Text Dokumentation zum Objekt
ArtifactType : DataObject	String	ja	nein	Spezifiziert den Typ des Artefaktes
Pool	String	nein	ja	Angabe des Pools, um die Lage des Artefakts zu lokalisieren
Lanes	String	nein	ja	Angabe der Lanes
Name	String	nein	ja	Text, bestimmt Beschriftung (Kommentar) des Shapes (siehe oben)
State	String	nein	ja	Zustand, der den Einfluss des Prozesses auf das Objekt anzeigt
Properties	String	nein	ja	Ermöglicht einem Modellierer, eigene Eigenschaften hinzuzufügen
RequiredForStart	Boolean	nein	nein	Gibt an, ob der Input vor Start der Aktivität benötigt wird
ProducedAtCompletion	Boolean	nein	nein	Gibt an, ob der Output erst bei Beendigung der Aktivität erzeugt wird

Abbildung 4.46: Attribute des Data Objects

Das Ändern des „Name“-Attributes ändert, wie gehabt, die Beschriftung des Objektes innerhalb des Rahmens zentriert und mittig.

Regeln

Datenobjekte erlauben, wie alle Artefakte, keine Verbindung mit anderen Objekten durch Message Flow oder Sequence Flow. Nur Assoziationen sind als Konnektoren erlaubt.

Data Objekts als Input oder Output einer Aktivität haben entsprechend eine unidirektionale Assoziation, wobei die Pfeilrichtung bestimmt, ob es ein Input oder Output ist. Dient ein Objekt sowohl als In- als auch als Output, so ist eine bidirektionale Assoziation möglich. Datenobjekte, die mit Sequenzflüssen oder MessageFlows verbunden sind, werden mit ungerichteten Assoziationen an die Mitte des Konnektors angehangen.

Magneten

- Oben
- Unten
- Links
- Rechts
- Zentral : Default

4.7 Swimlanes

Swimlanes in BPMN sollen ein Rollenkonzept anbieten. Dabei gibt es zwei Elemente: Pools und Lanes. Während Pools die Rolle eines Teilnehmers im Kontext eines Prozesses anbietet, sind Lanes eine Möglichkeit zur Unterteilung innerhalb dieser Rolle, wobei immer mindestens eine Lane vorhanden sein muss in einem Pool. Ebenso gehört zu jedem nichtleeren Prozess ein Pool, welcher Flussobjekte und Konnektoren einbettet.

Die Umsetzung von Swimlane Elementen der BPMN Spezifikation [1] erwies sich als nicht trivial, weswegen lange ein brauchbares Konzept zur Umsetzung diskutiert wurde. Dabei beherrschten mehrere Probleme das Suchen nach einer guten Lösung.

- Geeignetes Konzept zum Umgang mit Lanes in Pools (Hinzufügen, Löschen, etc.)
- Layouting für dynamisch an den Pool angepasste Lanes
- Graphische Umsetzung als SVG-Datei

Ideen, Lanes automatisch in einen Pool zu integrieren als ein Shape, dessen graphische Repräsentation sich durch Ändern der Anzahl der Lanes in einem Property verändert erschien dabei eine denkbar ungeeignete Lösung im Zuge des ressourcenorientierten Denkens, bei dem jedes - und das meint auch jede einzelne Lane - Shape eine Ressource sein sollte. Folglich entschieden wir uns für die Umsetzung strikt nach der BPMN als zwei einzelne Shapes, und nahmen die Herausforderung der Probleme an. Dieses war auch ein Punkt, an dem wohl die meisten Änderungen in die Stencil Set Specification eingeflossen sind, denn nun musste z.B. das Problem des Layoutings behandelt werden. Auch eine mangelnde Umsetzung des SVG-Text-Objektes im Firefox erschwerte es, überzeugende graphische Repräsentationen anzubieten, da die Beschriftung der Spezifikation nach um 90° gegen den Uhrzeigersinn gedreht ist. SVG bietet dafür ein Attribut an, das „Glyph orientation“ erlaubt, nur wird dies im Firefox nicht unterstützt. Also erzeugen wir den Text ganz normal und drehen ihn um 270° (im Uhrzeigersinn) mithilfe einer gewöhnlichen Objekt-Rotation. Im Zuge des Layoutings wurden auch die Probleme des Hinzufügens und Löschens von Pools gelöst, wodurch ein intuitives Konzept gewachsen war.

4.7.1 Pools

Pools repräsentieren Teilnehmer in einem Prozess in BPMN. Dieses können Einzelpersonen als auch abstrakte Gruppen von Menschen sein, Unternehmen wie Geschäftsrollen (wie Verkäufer, Berater, etc.). Ein Pool enthält genau einen oder keinen Prozess. Letzteres ist der Fall, wenn der Pool eine Blackbox darstellt, mit der ein anderer Prozess interagiert (über Message Flow). Da ein Prozess keine graphische Repräsentation besitzt, haben wir die Eigenschaften eines Pools mit denen eines Prozesses vereint. In der Attributsliste finden sich daher Attribute beider Elemente der BPMN.

Abbildung 4.47: Pool

SVG-Datei (node.pool.svg)

Ein Pool ist ein Rechteck mit einer schmalen Umrandung. Am linken Rand entlang steht vertikal als Beschriftung die Rolle des Teilnehmers, abgetrennt vom Rest der Box durch eine vertikale Linie.

```
    <rect id="c" oryx:resize="vertical horizontal"
          x="0" y="0" width="600" height="250"
          stroke="none" fill="white" />
    <rect id="border" oryx:resize="vertical horizontal"
          x="0" y="0" width="600" height="250"
          stroke="black" stroke-width="1" fill="none" />
    <text x="10" y="125"
          font-size="14"
          id="text_v" oryx:align="middle center" oryx:anchors="left" oryx:rotate="270"
          fill="black" stroke="black">
               Some Pool
    </text>
  </g>
```

Mit Hilfe des Oryx-Attributes oryx:rotate wird der Text gedreht (Objekt-Rotation) und mit oryx.anchors an der linken Seite ausgerichtet. Ein Pool besitzt ein Attribut „BoundaryVisible“, welches bestimmt, ob ein Pool explizit sichtbar sein soll oder aber die Rahmen ausgeblendet werden sollen. Dementsprechend ist dies für Pools umgesetzt, indem durch das Setzen des Wertes bestimmt wird, welches der beiden Rechtecke angezeigt werden soll, das ohne Rahmen oder das mit Rahmen.

Attribute

Ein Pool vereinigt die Attribute des Elements selbst mit den Process-Attributen und jenen, die für alle Swimlanes gelten und auch den für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
PoolId	String	nein	nein	Einheitliche referenzierbare Id des Pools im Diagramm
PoolCategories	String	nein	ja	Kategorien für den Pool
PoolDocumentation	String	nein	ja	Text Dokumentation zum Pool
Name	String	nein	ja	Text, bestimmt Beschriftung des Shapes (Teilnehmer, Rolle)
Participant	String	nein	ja	Name des Teilnehmers
Lanes	Boolean	nein	nein	Zeigt alle Lanes eines Pools an
BoundaryVisible : true	Boolean	nein	nein	Gibt an, ob der Rahmen sichtbar ist oder nicht
ProcessId	String	nein	ja	Einheitliche referenzierbare Id des Prozesses im Diagramm
ProcessName	String	nein	ja	Name des Prozesses
ProcessType	Choice	nein	nein	None | Private | Abstract | Collaboration
Status	Choice	nein	nein	None | Ready | Active | Cancelled | Aborting | Aborted | Completing | Completed
AdHoc : false	Boolean nein	ja	Specifiziert, ob der Prozess adhoc ist	
AdHocOrdering	Choice	nein	nein	Aktivitäten parallel oder sequentiell ausführen
AdHocCompletionCondition	String	nein	ja	Bedingsausdruck beim Completing
AdHoc : false	Boolean nein	ja	Ein BPEL4WS Attribut	
AdHoc : false	Boolean nein	ja	Ein BPEL4WS Attribut	
ProcessCategories	String	nein	ja	Kategorien eines Prozesses

ProcessDocumentation	String	nein	ja	Dokumentation zu einem Prozess

Abbildung 4.48: Attribute des Pools

Das Ändern des „Name“-Attributes ändert den Rollennamen im Shape, „BoundaryVisible“ die Anzeige des Rahmens. Derzeit ist es nicht möglich, zu überprüfen, ob in einem Diagram maximal ein Pool ohne Rahmen angezeigt wird, wie es die Spezifikation verlangt.

Die Attribute „Process“ des Pools und „GraphicalElements“ vom Prozess wurden nicht umgesetzt, weil in ersterem Fall der Prozess ja direkt an den Pool angehangen ist, im zweiten Fall sich die Menge der graphischen Elemente durch das Enthaltensein ergibt und die Anzeige derselben unnötige und nutzlose Redundanz bedeutet. Ebenso wurden „Assignments“ und „Properties“ nicht umgesetzt, da die Umsetzung von Assignments selbst recht schwierig sich gestalten und der Nutzen desselben in Frage gestellt werden kann.

Regeln

Ein Pool ist kein Flussobjekt und kann daher nicht durch Sequence Flows verbunden sein. Aber da ein Pool einen Teilnehmer in BPMN einnimmt, kann er über Message-Flows mit anderen Pools bzw Elementen in dem Pool kommunizieren. Auch Verbindungen mit Textannotationen und Datenobjekten über Assoziationen sind möglich. Pools enthalten ausschließlich Lanes, mit Ausnahme der Elemente, die unabhängig von Pools sind, wie Artefakte.

Magneten

- Oben
- Unten
- Links
- Rechts
- Zentral : Default

4.7.2 Lanes

Jeder Pool enthält mindestens eine Lane, welche den Inhalt des Prozesses hält. Eine Lane ist eine Unterteilung eines Pools und damit eines komplexen Teilnehmers.

Abbildung 4.49: Lane

SVG-Datei (node.lane.svg)

Eine Lane ist ein Rechteck mit einer schmalen Umrandung. Am linken Rand entlang steht vertikal als Beschriftung die Rolle der Lane.

```
<rect
     id="frame" oryx:resize="vertical horizontal"
     x="0" y="0" width="600" height="250"
     stroke="black" stroke-width="1" fill="white" />
<text
     x="10" y="125" oryx:rotate="270"
     font-size="14" id="text_v"
     oryx:align="middle center" oryx:anchors="left"
     fill="black" stroke="black">
         Some Lane
</text>
</g>
```

Das Textverhalten bei Lanes ist identisch zu dem bei Pools.

Attribute

Eine Lane besitzt die Attribute des Elements selbst sowie auch alle Swimlane Attrinute und auch die für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id der Lane im Diagramm
Categories	String	nein	ja	Kategorien für die Lane
Documentation	String	nein	ja	Text Dokumentation zur Lane
Name	String	nein	ja	Text, bestimmt Beschriftung des Shapes (Teilnehmer, Rolle)
ParentPool	String	nein	nein	Gibt den Pool an, dem eine Lane angehört
ParentLane	Boolean	nein	ja	Zeigt ParentLane an, wenn Lane in einer anderen Lane enthalten ist

Abbildung 4.50: Attribute der Lane

Das Ändern des „Name“-Attributes ändert - wie bereits bekannt - die Beschriftung im Shape. NestedLanes, also Lanes, die kaskadiert weitere Lanes enthalten können,

ist derzeit nicht umgesetzt in unserem BPMN Stencil Set. Jeder Pool kann beliebig viele Lanes enthalten, aber alle befinden sich auf der ersten Ebene und haben ausschließlich den Pool als ParentElement.

Regeln

Eine Lane ist, wie der Pool, kein Flussobjekt und wird daher nicht durch Sequence Flows verbunden. Einzig und allein Verbindungen mit Textannotationen und Datenobjekten über Assoziationen sind möglich. Lanes sind im Container Pool enthalten und besitzen als Inhalt sämtliche Flussobjekte und Konnektoren. Korrekterweise muss man sagen, dass die Konnektoren im Editor von der Enthaltenseinsbeziehung ausgeschlossen sind und nicht eine Lane als ParentObject haben.

Magneten

Keine

4.8 Connections

Konnektoren sind auch graphische Objekte, sowohl in BPMN als auch in unserem Editor, die jedoch besonderen Eigenschaften unterliegen. Primär werden sie aus einem Pfad bestehend aus einem Start- und einem Endpunkt gebildet, können sie jedoch durch Benutzerinteraktion um weitere Punkte im Verlauf des Pfades erweitert werden. An jedem definierten Punkt eines Pfades sitzt ein Docker, der das anfassen des Konnektors erlaubt und vor allem das Andocken ermöglicht. Infolgedessen besitzt ein Konnektorshape keine Magnets, dafür aber mindestens zwei Docker. Auch in der Namensgebung spiegelt sich wieder, das sie nicht zu den Knotenobjekten gehören, denn alle beginnen mit „connection.[...]“ Eine weitere Besonderheit stellt die Tatsache dar, dass Konnektoren keinem parentialen Pool oder einer Lane angehören.

In BPMN gibt es zwei verschiedene Typen von Verbindungsobjekten:

- Flussobjekte (Sequence und Message Flow)
- Assoziationen

4.8.1 Flussobjekte

Flussobjekte (Flow objects) sind Konnektoren, die die Ausführung von Aktivitäten direkt beeinflussen. Flussobjekte sind gerichtet und lassen den Fluss in genau eine Richtung zu.

4.8.1.1 Sequence Flow

Ein Sequenzfluss verbindet Flussobjekte untereinander innerhalb eines Pools. Ausgehend von einem Startevent kann der Sequenzfluss über Aktivitäten, Gateways und Events verlaufen und mündet im Endevent. Ein Sequenzfluss kann lane-übergreifend verlaufen, jedoch niemals die Poolgrenzen überschreiten. Im Zuge der Ausführung eines Prozesses bestimmt der Sequenzfluss die Reihenfolge der Abarbeitung der Aktivitäten.

Abbildung 4.51: Sequence Flow

Sequenzflüsse können Bedingungen haben, die bestimmen, ob der Fluss über diesen Konnektor verlaufen kann oder nicht. Dieser Ausdruck muss zuerst ausgewertet werden, bevor ein Token weitergehen kann.

Abbildung 4.52: Conditional Flow

An Aktivitäten und Data-Based XOR-Gateways kann zum Conditional Flow auch ein Default Flow angegeben werden, der ausgeführt wird, wenn keine der Bedingungen zutrifft.

Abbildung 4.53: Default Flow

SVG-Datei (connection.sequenceflow.normal.svg)

Ein Sequence Flow besteht aus einer einfachen, durchgezogenen, schwarzen Linie zwischen zwei Shapes. Am Ende der Linie sitzt eine einfache, dreiecksförmige ausgefüllte Spitze.

```
  <g id="edge">
<path
     d="M10 50 L210 50"
     stroke="black" fill="none" stroke-width="2"
     stroke-linecap="round" stroke-linejoin="round"
     marker-start="url(#start)" marker-end="url(#end)" />
</g>
```

Der Startmarker ist bei dem normalen Sequenzfluss leer, bei Conditional und Defaultflow wird der entsprechende Marker angezeigt.

Attribute

Ein Sequenzfluss besitzt die Attribute des Elements selbst sowie auch alle gemeinsamen Connection Object Attribute und auch die für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id im Diagramm
Categories	String	nein	ja	Kategorien
Documentation	String	nein	ja	Text Dokumentation
Name	String	nein	ja	Namensbeschriftung
Source	String	nein	nein	Quellobjekt

Target	String	nein	ja	Zielobjekt
ConditionType	String	nein	nein	None | Expression | Default
ConditionExpression ConditionType= Expression	String	nein	nein	Ausdruck der Bedingung
Quantity : 1	Integer nein	nein	Zielobjekt	

Abbildung 4.54: Attribute des Sequence Flow

Man wird sich fragen, warum alle drei Arten nicht in einem Shape realisiert wurden. Zunächst sollte dies auch so realisiert werden, das durch das Attribut „ConditionType“ bestimmt wird, ob es sich vielleicht um einen ConditionalFlow handelt oder nicht. Zum Zeitpunkt der Entscheidung fehlten dafür aber die Vorraussetzungen, Marker auch über ein Property ein und auszuschalten. Folglich ging die Entwicklung in Richtung einzelne Shapes. Bestätigt wurde die Entscheidung schlussendlich dadurch, dass das Regelwerk für die drei Sequenzflusstypen unterschiedlich aussieht. Obgleich die Möglichkeit zum Ein- und Ausblenden von Markern nun auch implementiert ist, so werden ConditionalFlow und DefaultFlow immer eigene Stencils bleiben aufgrund des letztgenannten Arguments.

Regeln

Generell haben Konnektoren keine Zugehörigkeit im Editor zu einem Pool oder einer Lane.

Sequence Flow Verbindungen dürfen nur zwischen Aktivitäten, Gateways und Events auftreten. Ein Startevent darf keinen eingehenden Sequenzfluss besitzen. Ebenso darf ein Endevent keine ausgehenden Sequenzflüsse haben. Conditional Flows dürfen nur an Gateways oder Aktivitäten beginnen, Default Flows nur an Data-Based XOR-Gateways und Aktivitäten. Regeln, die das Verbinden von Flussobjekten in unterschiedlichen Pools verhindern, sind zum derzeitigen Stand geplant, jedoch noch nicht implementiert.

4.8.1.2 Message Flow

Ein Message Flow zeigt den Fluss von Nachrichten zwischen zwei Teilnehmern an. Verbindungen sind dabei nur zwischen zwei unterschiedlichen Pools bzw den Flussobjekten in den Pools möglich.

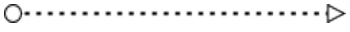

Abbildung 4.55: Message Flow

SVG-Datei (connection.messageflow.svg)

Eine Message Flow besteht aus einer einfachen, gestrichelten, schwarzen Linie zwischen zwei Shapes. Am Ende der Linie sitzt eine einfache, dreiecksförmige Spitze, die nach hinten zu offen ist. Der Anfang wird durch einen schwarz umrahmten Kreis gebildet.

```
<path
     d="M10 50 L210 50"
     stroke="black" fill="none" stroke-width="2"
     stroke-dasharray="3, 4"
     marker-start="url(#start)" marker-end="url(#end)" />
```

Attribute

Ein Message Flow besitzt die Attribute des Elements selbst sowie auch alle gemeinsamen Connection Object Attribute und auch die für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id im Diagramm
Categories	String	nein	ja	Kategorien
Documentation	String	nein	ja	Text Dokumentation
Name	String	nein	ja	Namensbeschriftung
Source	String	nein	nein	Quellobjekt
Target	String	nein	ja	Zielobjekt
Message	String	nein	ja	Message to be send

Abbildung 4.56: Attribute des Message Flow

Regeln

MessageFlows gehören verständlicherweise zu keinem Pool oder einer Lane in dem Editor. Message Flow Verbindungen dürfen nur zwischen Aktivitäten, Pools und Events auftreten. Ein Start- und Intermediateevent darf keinen ausgehenden Message Fluss besitzen. Ebenso darf ein Endevent keinen eingehenden Message Fluss haben. Eine Regel, die das Verbinden von gültigen Objekten innerhalb eines Pools verhindert, ist derzeit noch nicht implementiert.

4.8.2 Assoziation

Assoziationen sind notwendig, um Informationen und Artefakte an Flussobjekte zu binden. Dies kann ein Kommentar durch eine TextAnnotation sein wie auch ein DataObjekt. Im Allgemeinen lässt sich sagen, dass Assoziationen die einzige Möglichkeiten sind, um Flussobjekte und Konnektoren mit Nicht-Flussobjekten zu verbinden. Ein Spezialfall ist die Compensation Association, die von einer Aktivität in

eine Compensationsaktivität führt. Assoziationen gibt es ungerichtet, gerichtet und bidirektional:

Abbildung 4.57: Ungerichete Assoziation

Abbildung 4.58: Gerichete Assoziation

Abbildung 4.59: Bidirektionale Assoziation

SVG-Datei (connection.association.unidirectional.svg)

Eine Assoziation besteht aus einer einfachen, gestrichelten, schwarzen Linie. Am Anfang und Ende der Linie kann - je nach Typ - eine einfache, dreiecksförmige, hinten offene Spitze sitzen.

```
  <g id="edge">
   <path
        d="M10 50 L210 50"
        stroke="black" fill="none"
        stroke-width="2" stroke-dasharray="3, 4"
        marker-start="url(#start)" marker-end="url(#end)" />
</g>
```

Attribute

Eine Assoziation besitzt die Attribute des Elements selbst sowie auch alle gemeinsamen Connection Object Attribute und auch die für alle graphischen Objekte zugeordneten Eigenschaften. Nachfolgend sind alle Attribute gelistet.

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id im Diagramm
Categories	String	nein	ja	Kategorien
Documentation	String	nein	ja	Text Dokumentation
Name	String	nein	ja	Namensbeschriftung
Source	String	nein	nein	Quellobjekt
Target	String	nein	ja	Zielobjekt
Direction	String	nein	nein	None | To | Both

Abbildung 4.60: Attribute des Sequence Flow

Auch hier kommt die Frage auf, warum alle drei Arten nicht in einem Shape realisiert wurden. Der Grund ist derselbe wie bei den Sequence Flow Typen, da sich die Assoziationen im Regelwerk unterscheiden. Zum jetztigen Stand des StencilSets sind Abhängigkeiten von der Anwendung der Regeln von Properties nicht möglich, sodass nur generelle Regeln bei einem einzigen Shape möglich wären. Aus diesem Grund ist das Attribut „Direction“ auch nur ein String und kein Choice.

Regeln

Konnektoren gehören im Oryx-Editor zu keinem Pool oder einer Lane, folglich unterliegen sie keiner Containment-Regel. Einfach und doppelt gerichtete Assoziationen erlauben im Editor nur das Verbinden eines Datenobjektes mit einer Aktivität oder einem Event. Ungerichtete Assoziationen hingegen erlauben auch das Anhängen eines Datenobjektes an eine Sequence- oder Message Flow sowie das Setzen von Textannotationen.

4.9 Diagramm

Die Canvas wurde getypt, denn sie ist eine Ressource. Da das ParentObject in einem BPMN Modell ein Diagramm (Diagram) ist, wird die Canvas mit dem Diagramm getypt. Folglich lassen sich Pools auf die Canvas ziehen, denn die Canvas übernimmt die Rolle des Diagrams hinsichtlich der Regeln und der Properties. Durch anklicken der Canvas werden die Diagramm-Eigenschaften im Property-Window angezeigt. Ebenso lässt sich über dieses Konzept die Canvas auch mit einem anderen Stencil typen, zum Beispiel mit einer Lane. Entscheidend dafür, welches Stencil den Typ der Lane bestimmt, ist ein Meta-Tag in der XHTML-Datei, die den Editor einbindet.

```
<meta name="oryx.type" content="http://b3mn.org/stencilset/bpmn#BPMNDiagram"/>
```

Der Teil der URL bis zur Raute stellt den Namespace des Stencil Sets dar, wie er im Kopf der Stencil Set Definitionsdatei (bpmn.json) angegeben wurde, nach der Route wird die Id des Stencils angegeben. Um eine Lane zum Typ der Canvas zu machen, schreibt man folgendes:

```
<meta name="oryx.type" content="http://b3mn.org/stencilset/bpmn#Lane"/>
```

In jedem Fall hat ein Diagram-Object auch eine graphische Repräsentation und Regeln. Hinweis: Dieses Shape ist nicht Teil der Spezifikation und hat eine von uns entwickelte graphische Repräsentation. Lediglich die Attribute erbt es von dem BPMN Diagram Element.

Abbildung 4.61: Diagram

SVG-Datei (node.diagram.svg)

Ein Diagram besteht aus einem Rechteck mit verstärktem Rand und einem Schlagschatteneffekt.

```
<polygon
    stroke="black" fill="black" stroke-width="1"
    points="0,0 0,590 9,599 799,599 799,9 790,0"
    stroke-linecap="butt" stroke-linejoin="miter" stroke-miterlimit="10"
/>
<rect
    id="diagramcanvas" oryx:resize="vertical horizontal"
    x="0" y="0" width="790" height="590"
    stroke="black" stroke-width="2" fill="white"
/>
<text
    font-size="22" id="diagramtext" x="400" y="25"
    oryx:align="top center" stroke="black">
        Some Diagram
</text>
```

Attribute

Ein Diagram besitzt folgende Attribute:

Attribut	Typ	ReadOnly	Optional	Bemerkung
Id	String	nein	nein	Einheitliche referenzierbare Id des Diagramms
Name	String	nein	ja	Namensbeschriftung

Version	String	nein	ja	Version des Diagramms
Author	String	nein	ja	Autor des Diagramms
Language	String : English	nein	ja	Natürliche Sprache im Diagramm
Expression Language	String	nein	ja	Sprache von Ausdrücken
Query Language	String	nein	ja	Sprache von Queries
CreationDate	Date	nein	nein	Datum der Erstellung
Modification Date	Date	nein	nein	Datum der letzten Änderung
Pools	String	nein	nein	Enthaltene Pools
Documentation	String	nein	ja	Dokumentation

Abbildung 4.62: Attribute des Diagram Objects

Regeln

Die einzige Regel, die ein Diagram definiert, bestimmt, dass es Pools enthalten darf. Kein Sequenz- oder Messagefluss ist möglich.

5. Evaluierung

In diesem Kapitel wird die Stencil Set Spezifikation aus der Sicht des Stencil Set Designers betrachtet und die Erfahrungen im Erzeugen neuer Stencil Sets resümierend geschildert werden. Probleme werden ebenso offen angesprochen wie bestehende Lösungsmöglichkeiten. Ein Ausblick am Ende rundet die Bewertung mit der Vorstellung von neuen Konzepten und Ideen ab.

5.1 Bewertung

Bei der Implementation des BPMN Stencil Sets wurde die implementierte Stencil Set Spezifikation auf den Prüfstand gestellt und die Anforderungen teilweise auch neu ausgerichtet. Da BPMN aufgrund der Vielzahl an Elementen, komplexen Regeln und einer Fülle an Attributen zum Teil mit eigenen Typen besitzt, erwies sich der Vorgang, diese Notation mit der gegebenen Stencil Set Spezifikation zu einem vollständigen Stencil Set zu überführen, als nicht trivial und zeitaufwendig. 43 SVG-Dateien enthalten die Informationen zum Aussehen von Stencils, 33 Icons wurden gezeichnet, und alle Elemente in die BPMN Stencil Set Definitionsdatei (bpmn.json) eingefügt, Rollen und - darauf aufbauend - Regeln zum Verbinden, zu Enthaltenseinsbeziehungen und Kardinalitätsanforderungen definiert, mehrere hundert Properties zu den Stencils definiert, was die BPMN Stencil Set Spezifikationsdatei mit einer Gesamtlänge von derzeit 6525 Zeilen nicht gerade einfach und übersichtlich macht. Zum Vergleich wurden auch Stencil Sets für andere Notationen erzeugt, darunter ein Workflownet- und ein Petrinetz-Stencil Set. Auch dieses dienten dazu, die Stencil Set Spezifikation zu testen und nachzubessern. Desweiteren sollte der Aufwand für den Designer zum Erstellen eines neuen Stencil Sets getestet werden. Die Implementierung eines einfachen Petrinetz Stencil Sets hat 1 Arbeitsstunde benötigt, um die drei grundlegenden Elemente Kante, Stelle und Transition in ihrer SVG Repräsentation, den Icons und der Stencil Set Definitionsdatei inklusive Regelwerk und einfachen Eigenschaften zu erzeugen.

Das Regelwerk von BPMN erforderte ein umfangreiches Rollenkonzept, da in BPMN zum einen 7 verschiedene Verbindungstypen existieren, für die Verbindungsregeln geschaffen werden müssen, zahlreiche Flussobjekte, die bei den Verbindungsregeln

bedacht und selber auch Enthaltenseinsbeziehungen beachten müssen. Desweiteren sind auch Kardinalitätsregeln als mögliche Regelform für BPMN auf Notwendigkeit zu überprüfen, und - zuguterletzt - muss überprüft werden, ob sich alles mit den bestehenden Regeln abbilden lässt. Die Regel Spezifikation für Verbindungen erwies sich als sehr effektiv, insbesonders wegen dem zugrunde liegenden Rollenkonzept, welches effiziente Regelbildung erlaubt. Trotzdem sind Regeldefinitionen ein komplexer Prozess, der gut bedacht werden muss, da keine Ausnahme für eine Regel definiert werden kann, also zum Beispiel eine Verbindungsregel, die allgemein eine bestimmte Verbindung zulässt, außer bei einem bestimmten Shape, obwohl es die Rolle auch besitzt. In weniger komplexen Notationen sollte dies jedoch überschaubar bleiben, BPMN hingegen erfordert schon besonderes Augenmerk auf die Sonderfälle. Enthaltenseinsbeziehungen sind deutlich leichter zu implementieren bei BPMN, da diese Notation klar definiert, welche Elemente welche Anderen enthalten dürfen. Kardinalitätsregeln werden bei BPMN in der Form kaum benötigt, wohl aber im Thanis Stencil Set. Es gibt allerdings noch keine Beachtung der Regeln, dass ein Sequenzfluss nicht über die Grenzen eines Pools hinaus stattfinden kann und ebenso ein Messageflow nur zwischen verschiedenen Pools auftreten darf. Dies lässt sich mit der sonst generischen Stencil Set Spezifikation nicht lösen. Dafür gibt es allerdings auch eine Lösung, die im nächsten Abschnitt vorgestellt wird.

Die Möglichkeit, Properties für Stencils zu definieren, war ein wesentliches Kriterium an die Stencil Set Spezifikation. Es sind im Entwicklungszeitraum sehr innovative Konzepte entstanden, mit Properties umzugehen. So lässt sich beispielsweise die graphische Repräsentation eines Stencil durch Properties beeinflussen, sei es nun die Anzeige von Rahmen und Symbolen, dem Text oder der Hintergrundfarbe. Nur durch Angabe einer ReferenzId im Property, welche mit der Id in der zugehörigen SVG-Stencil Datei identisch sein musste, lies sich die Darstellung abhängig vom Typ des Properties ändern. Natürlich stellt das auch eine Einschränkung dar, dass die Typen die Art der Änderung bestimmen. Das Stencil Set sollte nicht nur angeben, worauf sich ein Property auswirken soll, sondern auch in welcher Form. Die Möglichkeiten, durch Properties andere Properties zu beeinflussen oder Regeln zuzulassen oder zu verweigern, sind derzeit nicht in Stencil Set Spezifikation implementiert. Hilfreich wäre dies zum Beispiel im Falle der Assoziation gewesen, bei der sich alle drei Subtypen (Undirected | Unidirectional | Bidirectional) sich in den Properties gleichen, nur besitzen sie ein unterschiedliches Regelwerk. Eine Lösungsmöglichkeit dafür wird ebenfalls im nachfolgenden Abschnitt diskutiert. Das Properties andere Properties beeinflussen, ist leider derzeit auch nicht möglich. Beeinflussen kann bedeuten, dass bestimmte Properties nur angezeigt werden, wenn ein bestimmtes Property einen bestimmten Wert hat oder True ist (Choice oder Boolean), oder aber auch den Wert eines anderen Properties direkt beeinflussen. Sinnvoll wäre dies zum Beispiel bei Tasks, bnei denen eine Unterscheidung in verschiedene Task-Types vorhanden ist, von der abhängig TaskType-spezifische Properties angeboten werden. Derzeit werden alle Properties angezeigt und es wird dem Entwickler überlassen, die notwendigen Properties zu füllen. Derzeit werden auch keine Properties automatisch dynamisch mit Werten belegt. So hält ein Pool in einem Property alle seine Lanes, was idealerweise beim Modellieren automatisch eingetragen wird. Dafür sind allerdings Callbacks für die Properties notwendig. Schwierig war teilweise auch das Mapping von BPMN-spezifischen Propertytypen auf Typen, mit denen der Editor umgehen kann. Dies wurde auch bereits im Kapitel 4.2.1 ausführlich diskutiert.

So verständlich das Konzept der Typisierung einer Canvas ist, so verwirrend ist es auch, damit umzugehen. Versuche, einfach auf der Canvas Flussobjekte zu zeichnen, wurden wiederholt beobachtet, obgleich bekannt ist, dass jedes Flußobjekt in einem Diagramm Kind eines Lanes, und der wiederum Kind eines Pools ist. Dabei wird das Diagramm-Shape im Stencil Repository angezeigt, was ebenfalls zu Verwirrung führt. Ein brauchbares Konzept zur Lösung dieses Problems liegt derzeit nicht vor, denn ein Ausblenden des Stencils aus dem Stencil Repository, welches auch Typ der Canvas darstellt, ist zwar bei Diagrammen ganz Praktisch, aber nicht mehr sinnvoll, wenn z.B. ein Sub-Prozess den Typ der Canvas bildet.

Schlussendlich lässt sich sagen, dass das generisch orientierte Konzept zur Definition von Regeln und Eigenschaften für Stencils sehr gut umgesetzt wurde und auch sehr gut funktioniert. Selbst komplexe Stencil Sets lassen sich erzeugen, und es darf mit Recht behauptet werden, dass die Business Process Modeling Notation erfolgreich und vollständig als Stencil Set dem Oryx-Editor zur Verfügung gestellt wird.

5.2 Lösungsideen und Ausblick

Für alle benannten Probleme brauchbare Lösungsansätze zu finden ist schwer, für einige jedoch sind konzeptionelle Entwürfe zur Behandlung der Schwierigkeiten vorhanden, auf die im Folgenden eingegangen werden soll.

Restriktionen von Flüssen bezüglich Pools

Wie bereits erwähnt, ist derzeit im BPMN Stencil Set keine Regel vorhanden, welche Sequenzflüsse über die Grenzen eines Pools hinaus oder Message Flows innerhalb eines Pools verbietet. MIt den herkömmlichen Verbindungsregeln lässt sich keine Einschränkung bezüglich dieser Regel definieren, da das Erkennen dieses Falles komplexer ist, aber auch gleichzeitig die Lösung mitliefert: Für den Sequence Flow muss überprüft werden, ob der Pool (Parent vom Parent vom Flussobjekt) des Startshapes nicht derselbe ist wie der vom Zielshape. In diesem Falle muss eine Verbindung untersagt werden. Die Stencil Set Spezifikation bietet seit kurzem einen „canConnect“-Callback an, welcher dementsprechenden Code bekommen kann, der dies erkennt. Eine einfache, unvollständige Form davon könnte wiefolgt aussehen:

```
canConnect: function(args) {
    if(args.result) {
        if(args.sourceShape.parent.parent.id == args.targetShape.parent.parent.id) {
                args.result = false;
        }
    }
    return args.result;
}
```

Auf diese Art und Weise müsste auch für Message Flows eine Überprüfung stattfinden, ob die Verbindung zwischen zwei Elementen innerhalb eines Pools etabliert werden soll.

Properties mit Werten füllen

Derzeit werden Properties leider nicht automatisch mit Prozesswissen versorgt. So ist zwar in der internen Darstellung klar, dass ein Pool zum Beispiel Lanes enthält, dies wird zur Designzeit aber von dem entsprechenden Property nicht beachtet. Wünschenswert wäre es, wenn ein Property sich selbst diesen Wert sucht. Dies kann durch ein Serialize-Callback geschehen, wie es partiell auch schon im Thanis Stencil Set etabliert wurde. Dabei wird im Anschluss an die Rollendefinition ein Serialize Callback eingefügt, welcher Code zum Füllen eines Properties enthalten kann.

Hier ein Beispiel aus dem Thanis Stencil Set:

```
"serialize":function(shape, data) {
    var incomingNum = 0;
    shape.getIncomingShapes(function(s) {
        if(s.getStencil().id() === "http://b3mn.org/stencilset/b3mn#SequenceFlow") {
            incomingNum++;
        }
    });

    data.push({name:"ports",
               prefix:"raziel",
               value:incomingNum,
               type: 'literal'});
    return data;
}
```

Die Umsetzung dieses Konzeptes würde das BPMN Stencil Set stark verbessern, jedoch ist der Prozess nicht einfach und vor allem langwierig.

Wordwrap

Ein Must-have ist eigentlich das automatische Wordwrap anhand der Rahmenbegrenzung für ein Shape. Das Problem in SVG ist dabei allerdings, dass zum Berechnen der Schriftbreite der Text bereits einmal gerendert werden muss. Dies sieht nicht sehr schön aus und ist zudem aufwendig und schlägt sich auf die Performance nieder. Fakt ist, dass für dieses Feature bisher kein brauchbarer Ansatz gefunden wurde, um die damit verbundenen Probleme zu umgehen.

Erweiterte Property-Abhängigkeiten

Derzeit ist es, wie bereits ausführlich erwähnt, nur möglich, über Properties die graphische Darstellung eines Stencils zu beeinflussen. Dabei wird vom Typ des Properties bestimmt, um was für eine Art der Änderung es sich handelt. Die Möglichkeiten würden dadurch wachsen, indem man nicht nur die Id des Objektes, auf das sich die Änderungen beziehen, mit angibt, sondern auch die Art der Änderung.

Beispiel:

```
{
    "id":"name",
    "type":"String",
    "title":"Name",
    "value":"",
```

```
    "description":"",
    "readonly":false,
    "optional":true,
    "refToView":"acttext",
    "targetBehavoir":"innerContent",
    "length":"50",
    "wrapLines":true
}
```

Durch „targetBehavoir“:“innerContent“ wird bestimmt, um welche Art von Änderung es sich bei der in refToView referenzierten Id handelt. In diesem Fall wäre es den Inhalt des referenzierten SVG-Objekts. Properties sollten aber auch abhängig von anderen Properties sein können. Eine einfache Form der Abhängigkeit würde man erreichen, indem jedes Property um eine Eigenschaft erweitert wird: dependency und dependencyValue.

```
{
    "id":"name",
    "dependency":"[IdOfOtherProperty]",
    "dependencyValue":"[Value]",
    "type":"String",
    "title":"Name",
    "value":"",
    "description":"",
    "readonly":false,
    "optional":true,
    "refToView":"acttext",
    "length":"50",
    "wrapLines":true
}
```

In diesem Falle wird dieses Property nur eingeblendet, wenn der Wert von „dependencyValue“ dem Wert des „values“ des in „dependency“ angegebenen Properties entspricht. Dabei sollte nur auf Properties vom Typ Boolean oder Choice referenziert werden.

In dieser Form lassen sich Regeln auch durch ein Property bestimmen, indem eine Regeldefinition um zwei Werte erweitert wird: dependency und dependencyValue.

```
{
    "role":"SequenceFlow",
    "dependency":"[IdOfOtherProperty]",
    "dependencyValue":"[Value]",
    "connects": [
        {
            "from":"sequence_start",
            "to":["sequence_end"]
        }
    ]
}
```

Es lassen sich auch noch andere Möglichkeiten benennen, aber die genannten sind die vielleicht Interessantesten, über die diskutiert werden kann.

A. Übersicht Regelwerk

Das ist der Quelltext für alle Regeln, die in der BPMN Stencil Set Definitionsdatei aufgeführt werden:

```
"rules": {
   "connectionRules": [
      {
         "role":"SequenceFlow",
         "connects": [
            {
               "from":"sequence_start",
               "to":["sequence_end"]
            }
         ]
      },
      {
         "role":"DefaultFlow",
         "connects": [
            {
               "from":"default_start",
               "to":["sequence_end"]
            }
         ]
      },
      {
         "role":"ConditionalFlow",
         "connects": [
            {
               "from":"conditional_start",
               "to":["sequence_end"]
            }
         ]
      },
      {
         "role":"MessageFlow",
         "connects": [
            {
               "from":"messageflow_start",
```

```
            "to":["messageflow_end"]
         }
      ]
   },
   {
      "role":"Association_Unidirectional",
      "connects": [
         {
            "from":"from_task_event",
            "to":["DataObject"]
         },
         {
            "from":"DataObject",
            "to":["to_task_event"]
         },
      ]
   },
   {
      "role":"Association_Bidirectional",
      "connects": [
         {
            "from":"from_task_event",
            "to":["DataObject"]
         },
         {
            "from":"DataObject",
            "to":["to_task_event"]
         },
      ]
   },
   {
      "role":"Association_Undirected",
      "connects": [
         {
            "from":"SequenceFlow",
            "to":["DataObject"]
         },
         {
            "from":"DataObject",
            "to":["SequenceFlow"]
         },
         {
            "from":"MessageFlow",
            "to":["DataObject"]
         },
         {
            "from":"DataObject",
            "to":["MessageFlow"]
         },
         {
            "from":"fromtoall",
            "to":["TextAnnotation"]
         },
         {
            "from":"TextAnnotation",
            "to":["fromtoall"]
         }
      ]
   },
```

```
{
   "role":"IntermediateEvent",
   "connects": [
      {
         "from":"Task",
         "to":["IntermediateEvent"]
      }
   ]
},
{
   "role":"IntermediateMessageEvent",
   "connects": [
      {
         "from":"Task",
         "to":["IntermediateMessageEvent"]
      }
   ]
},
{
   "role":"IntermediateTimerEvent",
   "connects": [
      {
         "from":"Task",
         "to":["IntermediateTimerEvent"]
      }
   ]
},
{
   "role":"IntermediateErrorEvent",
   "connects": [
      {
         "from":"Task",
         "to":["IntermediateErrorEvent"]
      }
   ]
},
{
   "role":"IntermediateCancelEvent",
   "connects": [
      {
         "from":"Task",
         "to":["IntermediateCancelEvent"]
      }
   ]
},
{
   "role":"IntermediateCompensationEvent",
   "connects": [
      {
         "from":"Task",
         "to":["IntermediateCompensationEvent"]
      }
   ]
},
{
   "role":"IntermediateRuleEvent",
   "connects": [
      {
         "from":"Task",
```

```
            "to":["IntermediateRuleEvent"]
         }
      ]
   },
   {
      "role":"IntermediateLinkEvent",
      "connects": [
         {
            "from":"Task",
            "to":["IntermediateLinkEvent"]
         }
      ]
   },
   {
      "role":"IntermediateMultipleEvent",
      "connects": [
         {
            "from":"Task",
            "to":["IntermediateMultipleEvent"]
         }
      ]
   }
],
"cardinalityRules": [
   {
      "role":"Startevents_all",
      maximumOccurrence:undefined,
      outgoingEdges: [
         {
            role:"SequenceFlow",
            maximum:1,

         }
      ],
      incomingEdges: [
         {
            role:"SequenceFlow",
            maximum:0,

         }
      ]
   },
   {
      "role":"Endevents_all",
      maximumOccurrence:undefined,
      outgoingEdges: [
         {
            role:"SequenceFlow",
            maximum:0,

         }
      ],
      incomingEdges: [
         {
            role:"SequenceFlow",
            maximum:1,

         }
      ]
```

```
      }
   ],
   "containmentRules": [
      {
         "role":"BPMNDiagram",
         "contains": [
            "Pool"
         ]
      },
      {
         "role":"Pool",
         "contains": [
            "Lane"
         ]
      },
      {
         "role":"Lane",
         "contains": [
            "tc"
         ]
      },
      {
         "role":"fromtoall",
         "contains": [
            "Group"
         ]
      },
      {
         "role":"fromtoall",
         "contains": [
            "DataObject"
         ]
      },
      {
         "role":"fromtoall",
         "contains": [
            "TextAnnotation"
         ]
      }
   ],
}
```

Literaturverzeichnis

[1] OMG. Business process modeling notationspecification. Technical report, OMG, 2006.

[2] Willi Tscheschner. Oryx - documentation.

[3] Martin Czuchra. Oryx - data management.

[4] Nicolas Peters. Oryx - bpmn stencil set specification.

[5] Dean Jackson Jon Ferraiolo, FUJISAWA Jun. Scalable vector graphics (svg) 1.1 specification. Technical report, January 2003. http://www.w3.org/TR/2003/REC-SVG11-20030114/.