



Institut für Multimediale und Interaktive Systeme
Direktor: Prof. Dr. rer. nat. Michael Herczeg

NRML

-

Entwurf einer Beschreibungssprache für
das narrative System Jeherazade

Studienarbeit
Praktische Semesterarbeit

Vorgelegt von:

Markus Pöstinger

Prüfer:

Prof. Dr. rer. nat. Michael Herczeg
Institut für Multimediale und Interaktive Systeme

Betreuer:

Dipl.-Ing. Peter Hoffmann

Abstract

Diese Arbeit behandelt den Entwurfsverlauf und die Entstehung einer Beschreibungssprache basierend auf XML, genannt *NRML* (NaRration Markup Language), für das in der Entwicklung befindliche *Jeherazade*, ein narratives System, im Rahmen einer Studienarbeit. Diese neue Sprache soll dabei die Schnittstelle zwischen dem narrativen System sowie Ein- und Ausgabegeräten verschiedener Art bilden.

Teil dieser Arbeit ist außerdem die Dokumentation dieser Sprache.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Jeherazade	1
1.2 Aufgabenstellung	1
2 Ausgangspunkt / Stand der Technik	2
2.1 Vorhandene Programmiersprachen	2
3 Aufgaben- und Benutzeranalyse	4
3.1 Benutzeranalyse	4
3.2 Aufgabenanalyse	5
4 Konzeption	6
4.1 Notwendige Eigenschaften	6
4.2 Sprachkonzept	6
5 Realisierung	8
5.1 Sprachstruktur	8
5.2 Sprachelemente	8
5.3 Sprachrealisierung	24
6 Zusammenfassung und Ausblick	26
Glossar	27
Literaturhinweise	28
Anhang	30

1 Einleitung

Gegenstand dieses Kapitels ist die Einführung in das narrative System Jeherazade sowie die der Arbeit zugrunde liegende Aufgabenstellung.

1.1 Jeherazade

Bei *Jeherazade* handelt es sich um ein in der Entwicklung befindliches System zur ausführlichen Spezifikation von Narrationen mit dem Schwerpunkt der Informationsvermittlung in Präsentationen.

Gängige Präsentationssysteme sind in hohem Maße linear und geben dem Nutzer nur bedingt Möglichkeiten zur Interaktion. Jeherazade will sich im Gegensatz dazu den Interessen und Besonderheiten des jeweiligen Nutzers anpassen, Spannung und Motivation beim Nutzer erzeugen, um damit die Präsentation für diesen interessanter und lehrreicher zu gestalten. Es integriert hierzu die Möglichkeiten von Nichtlinearität, Interaktivität, Adaptivität und Multi- & Hypermedialität in Präsentationssysteme.

Jeherazade wird am Institut für Multimediale und Interaktive Systeme der Universität zu Lübeck von Prof.Dr.rer.nat. Michael Herczeg und Dipl.Ing. Peter Hoffmann entwickelt.

1.2 Aufgabenstellung

Gefordert war die Entwicklung einer Beschreibungssprache für das Jeherazade-System, mit welcher die einzelnen Narrationen oder Präsentationen erzeugt werden können. Im Detail sollten

- das Vorhandensein und das Aussehen jeder Bühne
- für jede Bühne Charaktere und Equipment
- Interaktionsmöglichkeiten
- Inhalte und deren Quellen
- Abhängigkeiten der Inhalte untereinander
- als Kernpunkt dieser Arbeit die Struktur der Präsentation

definiert werden können.

2 Ausgangspunkt und Stand der Technik

In diesem Kapitel wird untersucht, ob überhaupt eine neue Sprache entwickelt werden muß, und warum dies notwendig oder unnötig ist.

2.1 Vorhandene Programmiersprachen

Sowohl in den Bereichen der Präsentationssysteme als auch der Narrationssysteme gibt es vorhandene Programmiersprachen, die zumindest einen genauen Blick wert sind. Jedoch erkennt man, daß diese Sprachen schon auf den ersten Blick nicht genau die Funktionen abdecken, die für Jeherazade benötigt werden. Denn eine Programmiersprache hierfür muß sowohl die Funktionen aufweisen Narrationen zu spezifizieren als auch Präsentationen zu erstellen. Leider decken die dem Autor bekannten Sprachen in diesem Bereich aber nur einen Teil davon ab. Trotzdem werden die Möglichkeiten jener betrachtet, da unter Umständen eine Sprache als Ausgangspunkt dienen könnte.

2.1.1 Flash

Flash (Macromedia, 2005) ist ein Produkt von der Firma Macromedia. Hierbei handelt es sich um keine Programmiersprache, sondern um ein Entwicklungstool, welches an eine Mischung aus einer Bildverarbeitung und einem Videoschnittprogramm erinnert und mit dem man sogenannte Flash-Filme erstellen kann. Diese Filme können wie SMIL alle Arten von multimedialen Inhalten enthalten, ebenso Elemente zur Benutzer-Interaktion. Flash ist wegen seines gut funktionierenden Players, der ebenfalls von Macromedia entwickelt wurde, wegen des professionellen Entwicklungstools und des Supports der Firma bisher unangefochten.

Flash ist für diese Arbeit eher wenig interessant, da man Filme ohne das Entwicklungstool nur sehr kompliziert automatisiert erstellen kann. Außerdem fehlen die Möglichkeiten zur Narration, die der Sprache ebenfalls mit hohem Aufwand beigefügt werden müßten. Zudem ist Flash natürlich als Produkt der Wirtschaft lizenzierungspflichtig, wodurch jede Ausstellung für ihre Geräte eine Lizenz erwerben müßte.

2.1.2 SMIL

SMIL (W3C, 2005) ist eine vom World Wide Web Consortium als Standard entwickelte, auf XML basierende Beschreibungssprache zur Darstellung von multimedialen Inhalten wie Video, Audio, Text und Grafik in html-Seiten. Eigenständige Player stehen allerdings ebenfalls zur Verfügung, um die erstellten Dateien abzuspielen.

Die Sprache hat sich bis heute auf dem Markt nicht durchsetzen können. Zum einen fehlt es an korrekt funktionierenden Playern und zum anderen an Entwicklungstools, um SMIL auch Personen zu öffnen, die nur geringe Programmierkenntnisse besitzen.

SMIL ist für diese Arbeit allerdings sehr interessant, da sie sich als Beschreibungssprache sehr gut als Basis für eine Weiterentwicklung eignet. Die Sprache besitzt praktisch alle zur Präsentation auf dem Zielsystem notwendigen Fähigkeiten und müßte nur noch um die Narrationselemente erweitert werden. Außerdem ist sie nicht zum Einsatz auf verteilten Systemen gedacht, weswegen hierzu ein Kernmodul programmiert werden müßte. Dieses Kernmodul ist mit Jeherazade aber vorhanden.

2.1.3 ADL

ADL, nach Brengle und Cunniff (1987), ist eine ältere Programmiersprache, die in den Achziger Jahren entwickelt wurde, um Textadventures zu programmieren. Sie ist ein Derivat von DDL (Dungeon Definition Language), welches ebenfalls zur Programmierung von Rollenspielen verwendet wurde und von Adler, Kostanick, Stein, Urban und Usui (1981) entwickelt wurde.

Die Sprache besitzt wie DDL eine C- oder Perl-ähnliche Syntax und ist im Hinblick auf ihren Zweck recht kompliziert und wenig intuitiv. Es wurde nur ein kurzer Blick auf ADL geworfen, jedoch offenbarte sich schon bald, daß die Sprache für diese Arbeit nicht zu verwenden ist, da bei ADL die für diese Zwecke notwendigen Präsentationselemente fehlen, da die Sprache rein auf Text als Ausgabe abzielt.

2.1.4 StoryML

Bei StoryML, von Hu und Fejjs (2003) entwickelt, handelt es sich um eine XML-basierte Beschreibungssprache zur Spezifikation von interaktiven Präsentationen auf verteilten Systemen. Die Sprache findet Erwähnung in einigen Papern und wurde an der Universität Eindhoven entwickelt.

Die Zielsetzung klingt für die Zwecke dieser Arbeit sehr passend, allerdings bleibt ein genauer Blick auf die Möglichkeiten von StoryML verwehrt, da sie seit einiger Zeit scheinbar nicht mehr weiterentwickelt wird und nicht über die Analysephase hinausgekommen zu sein scheint. Sie hätte sich für eine Basis gut eignen können, allerdings wäre dann vermutlich eine Einbindung von Möglichkeiten zur Spezifikation von Präsentationen notwendig gewesen, da sich StoryML in ihrem Ansatz sehr auf Narrationsspezifikationen konzentriert hat.

3 Aufgaben- und Benutzeranalyse

Gegenstand dieses Kapitels ist eine Analyse der Benutzer, die mit der Programmiersprache später arbeiten, und eine Aufgabenanalyse, in der untersucht wird, welche Anforderungen die Benutzer an die Sprache stellen.

3.1 Benutzeranalyse

Den eigentlichen Nutzern von Jeherazade, also den Besuchern der Ausstellungen, kann keine Erfahrung mit Computersystemen unterstellt werden. Jedoch richtet sich die Programmiersprache NRML nicht an die Nutzer des Systems, sondern vielmehr an die Autoren, welche die Präsentationen erstellen. Während sich also die eigentliche Ausgabe der Präsentationen, nämlich die Bühne, auf der diese dargestellt werden, an unerfahrene Nutzer beziehungsweise bestenfalls Gelegenheitsbenutzer richten sollte, werden die Autoren, die die Präsentationen erstellen, des Öfteren mit Computersystemen gearbeitet haben. Sie besitzen also mindestens die Fähigkeiten eines Routinebenutzers.

Diese Programmiersprache richtet sich an zwei Gruppen von Nutzern, deren Aufgabenbereiche, Sichten, Kenntnisse, Erfahrungen, Fertigkeiten und Erwartungen sich stark voneinander unterscheiden.

3.1.1 Die Programmierer

Der Aufgabenbereich der Programmierer ist es, aus vorhandenen Grafiken, Filmen, Musikdateien und Texten eine Präsentation zusammenzustellen. Um dieses zu realisieren, benötigt der Programmierer bestimmte Werkzeuge. Optimal wäre hierzu ein Autorensystem, jedoch kann der Programmierer die Präsentation auch mit Hilfe eines von ihm bevorzugten Texteditors erstellen. Editiert werden kann NRML als XML-Darivat mit jedem gewünschten Texteditor oder einer XML-Umgebung, die zum Beispiel die Open-Source Plattform Eclipse (OpenSource, 2005) zur Verfügung stellt.

3.1.2 Die Redakteure

Später kann es sinnvoll sein, ein Autorensystem zu erstellen, um nicht nur Programmierern, sondern auch Redakteuren die Erstellung von Präsentationsmodulen möglichst einfach zu gestalten. Dieses müßte die Struktur der Knoten sowie den Aufbau eines Knotens selbst inklusive des Aussehens seines Templates im WYSIWYG-Modus auf leicht überschaubare Weise zugänglich machen. Die

Konzeption und Gestaltung solch eines Autorentools sprengt jedoch den Rahmen dieser Arbeit.

3.2 Aufgabenanalyse

Programmierer wünschen sich eine Programmiersprache, mit der sie ohne großen Aufwand und ohne lange Einarbeitungszeit eine Präsentation erstellen können. Optimalerweise sollten sich die Funktionen und Befehle der Sprache an schon bekannten Programmiersprachen orientieren, um die Einarbeitungszeit möglichst gering zu halten. Da sich die Erarbeitung einer Narration und die spätere Darstellung der Inhalte stark voneinander unterscheiden, sollte diese Trennung auch innerhalb der Programmiersprache unterstützt werden. Ebenso sollte es eine Möglichkeit geben, das ähnliche Layout von Knoten innerhalb einer Ausstellung nur einmal definieren zu müssen. Eine gute Dokumentation der Programmiersprache runden die Anforderungen ab.

Redakteure haben ähnliche Anforderungen an die Sprache, jedoch verlangt es ihnen zusätzlich noch nach einem Autorentool, welches sie bei ihrer Arbeit zuverlässig unterstützt. Sie benötigen während der Programmierung hilfreiche Tips, welche Funktionen an welcher Stelle anzuwenden sind und welche Möglichkeiten sie mit einem bestimmten Befehl haben; gerade auch deswegen, weil sie üblicherweise keine Vorkenntnisse mit einer anderen Programmiersprache besitzen.

4 Konzeption

Dieses Kapitel beinhaltet Überlegungen, welche Features die zu entwickelnde Sprache bieten muß, und ob eine vorhandene Programmiersprache verwendet werden kann oder ob eine neue Sprache entwickelt werden muß. Es wird darüber hinaus eine Begründung für den jeweiligen Weg gegeben. Außerdem kann im letzten Abschnitt eine Auflistung der verwendeten Ideen aus anderen Papern gefunden werden.

4.1 Notwendige Eigenschaften

Die Forderungen der Arbeit machen die notwendigen Features für die Beschreibungssprache beinahe offenkundig: Sie muß Möglichkeiten zur Spezifikation der Präsentation bieten; also unter anderem Möglichkeiten, um deren Struktur, das Vorhandensein sowie das Aussehen jeder Bühne und die verwendeten Charaktere, Equipments sowie Interaktionsmöglichkeiten zu definieren. Außerdem nötig sind die Möglichkeiten zur Spezifikation der Narration; im Detail die Definition von Inhalten und deren Quellen, die Gliederung dieser Inhalte in Abschnitte und die Kennzeichnung der Abhängigkeiten der Inhalte untereinander.

4.2 Sprachkonzept

Jede der vorhandenen Sprachen liefert für unsere Zwecke recht gute Ansätze, jedoch kann keine diese mehr als nur zur Hälfte erfüllen. So hat Flash zwar eine ausgereifte Umgebung zu bieten, um Präsentationen zu erstellen; jedoch fehlt bei dieser Umgebung die Möglichkeit Präsentationen automatisiert zu erstellen. Darüber hinaus fehlen Narrationselemente. Mit SMIL wiederum können automatisiert Präsentationen erstellt werden, da es sich auf ein XML-Datenmodell stützt. Jedoch sind die zur Verfügung stehenden Player wenig ausgereift und interpretieren den SMIL-Standard unterschiedlich. Zudem fehlen auch hier Narrationselemente. ADL bietet wiederum einige Möglichkeiten zur Narration, jedoch fehlen hier gute Möglichkeiten zur Erstellung von Präsentationen. Zuletzt bietet StoryML theoretisch zwar einige Möglichkeiten, um Narrationen zu erstellen, bleibt aber unkonkret und beachtet Möglichkeiten zur Präsentation kaum.

So gibt die Analyse der vorhandenen Sprachen Anlaß zur Überlegung, eine eigene Sprache zu entwickeln; jedoch macht die Sprachstruktur von SMIL einen recht guten Eindruck, so daß sie als Basis der neuen Sprache herhalten kann. Für unsere Zwecke genügt es darüber hinaus, nur einige der Elemente von SMIL zu übernehmen und anzupassen, da wir nicht alle Features dieser Sprache benötigen.

Da die neue Sprache aus zwei sehr unterschiedlichen Funktionsteilen besteht, nämlich einerseits aus den Elementen, um die Präsentation zu erstellen, und andererseits aus den Elementen, um die Narration zu erstellen, ist es sinnvoll eine Möglichkeit vorzusehen, um die beiden Teile auch in der Sprachstruktur voneinander zu trennen. Damit könnten sich jeweils für sich Layouter um die Präsentation und Redakteure um die Narration kümmern; sie müßten sich dann nur über eine gemeinsame Schnittstelle verständigen. Die Sprache wird also eine Möglichkeit vorsehen, um Präsentation und Narration voneinander getrennt zu spezifizieren.

Weiter scheint es sinnvoll, die einzelnen Knoten der Narration Obergruppen zuzuteilen, die Kapitel genannt werden sollen. Damit könnte eine Form der sogenannten Spannungskurve des Aristoteles nach Hoffmann (2004) unterstützt werden. Nach diesem ist eine spannende, zum Lesen motivierende Geschichte in mehrere Abschnitte zu unterteilen: Der erste Abschnitt kümmert sich um die Vorstellung der Charaktere in der Geschichte und die Umgebung, in der die Charaktere leben. Der zweite Abschnitt sorgt für die Vorstellung und die Intensivierung einiger Konflikte, die der Geschichte Spannung verleihen und die zum Weiterlesen motivieren. Der dritte Abschnitt enthält den Höhepunkt der Geschichte, den sogenannten Climax, während dem der Hauptcharakter einen entscheidenden Schritt zur Lösung eines Problems macht. Der vierte und letzte Abschnitt kümmert sich um eine Auflösung der letzten offenen Konflikte und um einen logischen Schluß. Die Spannungskurve des Aristoteles ist zusammen mit dem Begriff des Suspense, also der Spannung, auch von Hoffmann und Herczeg (2004) noch einmal ausführlich beschrieben. Die Sprache wird also auch eine Möglichkeit vorsehen, um die Knoten in Kapitel zu unterteilen.

Immanent bei einer Narration, die aus voneinander abhängigen Knoten besteht, die aber je nach den Interessen des jeweiligen Lesers neu angeordnet werden können, ist es, die Abhängigkeit der Knoten untereinander in der Sprache auch vernünftig abbilden zu können. Es muß also eine Möglichkeit vorgesehen werden, für einen Knoten der Narration die unbedingt zu besuchenden Vorgängerknoten zu definieren, die nötig sind, um den Knoten zu verwenden. Eine genaueren Überblick über dieses Problem liefern Hoffmann und Herczeg (2004).

5 Realisierung

Dieses Kapitel beinhaltet die Erläuterung jedes einzelnen Elements und jedes Attributs in seinem Kontext. Damit erfüllt es den Zweck einer Dokumentation.

5.1 Sprachstruktur

NRML (NaRration Markup Language) besteht wie jedes XML-Derivat aus einem Baum von Elementen. Es gibt ein Wurzelement, welches jedes weitere Element umschließt; dies ist auch das einzige Basiselement. Darunter folgt eines von zwei Bereichen.

Entweder folgt das Templateelement, mit dem die NRML-Datei als Präsentationstemplate definiert wird, welches von einem Knoten der Narration zur Präsentation verwendet werden kann. Oder es folgt das Narrationselement, mit dem die NRML-Datei als Narrationsdatei definiert wird. In dieser Narrationsdatei ist die Struktur der Narration notiert, die Gliederung der Inhalte in Kapitel und Knoten sowie auch die Inhalte selbst.

5.2 Sprachelemente

In diesem Kapitel werden alle Elemente und Attribute erläutert. Zur visuellen Darstellung der Elemente werden Tabellen verwendet, wie sie auch von Bulterman und Rutledge (2004) verwendet worden sind. Diese tragen in ihrem Kopfbereich den Titel des entsprechenden Elements und im Fußbereich sowohl die zugehörigen Attribute als auch die möglichen Vater- und Kind-Elemente.

Es wäre auch eine formellere Notation, zum Beispiel in Bachus-Naur-Form, möglich gewesen; allerdings ist diese für XML-Derivate zu umfangreich und eher für Programmiersprachen mit einer eigenen Struktur zu empfehlen. Den XML-Sprachen liegt schon eine Baumstruktur mit Unterteilung in Elemente und Attribute zugrunde, die man in BNF mit jeder Sprache neu definieren müßte. Die in dieser Arbeit entlehnten Tabellen geben jedoch diese Struktur von sich aus sehr gut wieder.

5.2.1 Basiselemente

Element: `<nrm1>`

<code><nrm1></code>	
attributes	xmlns
children	<code><template></code> <code><narration></code>

Tabelle 1: Das `<nrm1>`-Element

<nrml> ist das Wurzel-Element jedes NRML-Elements und identifiziert die XML-Datei als ein NRML-Dokument. Der Inhalt des <nrml>-Elements ist immer ein <template>-Element, um ein Präsentationstemplate zu erzeugen, oder ein <narration>-Element, um die Narration zu spezifizieren. Als einziges Attribut ist das xmlns-Attribut erlaubt: Die Namespace-Deklaration bestimmt, welche Elemente und Attribute das XML-Dokument unterstützt und an welchem Platz das XML-Schema zu finden ist.

5.2.2 Präsentationselemente

Element: <template>

<template>	
attributes	color volume sound
parents	<nrml>
children	<region>

Tabelle 2: Das <template>-Element

Mittels des <template>-Elements wird der zentrale Container definiert, in dem die gesamte Präsentation für einen bestimmten Knoten abläuft. Eine genauere Platzierung von Inhalten wird mit dem <region>-Element vorgenommen, welches dazu unterhalb dieses Elements verwendet werden muß. Das Template unterstützt eine Hintergrundfarbe und die Angabe des Lautstärkepegels dieser Region, was mit den Attributen color, sound und volume definiert werden kann.

Attribut: color

Mit diesem Attribut kann in zahlreichen Elementen die Hintergrundfarbe festgelegt werden. Es erwartet als Eingabe einen RGB-Wert in hexadezimalen Format wie er auch bei html üblich ist.

Attribut: sound

Es wird ein Pfad zu einer abzuspielenden Sounddatei erwartet.

Attribut: volume

Hiermit kann die Lautstärke festgelegt werden. Es wird eine Prozentangabe erwartet, wobei 100% die Maximallautstärke bilden.

Element: <region>

<region>	
attributes	id coords, fit x, y, z color volume, sound
parents	<region> <template>
children	<bottom>, <right> <width>, <height> <video>, <audio> , <text> <a>, <area> <character> <equipment> <region> <event> <link> <behaviour>

Tabelle 3: Das <region>-Element

Das <region>-Element erfüllt die Funktion einer im Template frei verschiebbaren Hülle, innerhalb der der Inhalt liegt. Die Position der Region wird mittels der x-, y- und z-Attribute angegeben. Zur Definition der Größe der Region ist es darüber hinaus nötig, daß zusätzlich noch mit den <width>- und <height>-Elementen die Breite und Höhe oder mit den <bottom>- und <right>-Elementen die Position der rechten, unteren Ecke angegeben wird. Mittels des coords-Attributs kann angegeben werden, ob die angegebenen Koordinaten absolut zum Template oder relativ zum übergeordneten Container sein sollen. Zusätzlich besitzt die Region neben einer eindeutigen Bezeichnung ebenfalls eine Hintergrundfarbe und einen Hintergrundsound.

Einer Region können neben gewöhnlichen Inhalten noch Charaktere und Equipment zugewiesen werden, genauso wie sie rekursiv in weitere Regionen aufgeteilt werden kann. Mittels des fit-Attributes kann angegeben werden, wie sich der Inhalt der Region verhalten soll, wenn er über die Grenzen dieser hinaus geht.

Attribut: id

Die meisten XML-Dokument-Klassen sehen eine ID (Identifier) für die Elemente vor, die später innerhalb der Dokumente angesprochen werden sollen. Speziell für die Regionen und Medien sind

IDs sehr wichtig; es wird später noch im Abschnitt über die Narrationselemente näher darauf eingegangen.

Attribut: x, y, z

Mit diesen Attributen wird zum einen zur Positionsangabe die (X,Y)-Koordinate der linken, oberen Ecke der Region notiert. Zum anderen wird hiermit der Z-Index notiert, der die Position der Ebene im dreidimensionalen Raum angibt; je höher der Index, desto weiter hinten beziehungsweise unten liegt die Region. Tiefere Regionen werden von höheren überlappt.

Attribut: coords

Dieses Attribut kann nur zwei Werte annehmen: relative oder absolute. Absolute Koordinatenangaben beziehen sich auf die Position im Template insgesamt, relative beziehen sich auf die Position gegenüber der Vater-Region.

Attribut: fit

Mit dem Attribut fit wird mittels folgender Schlüsselwörter bestimmt, wie mit über die Grenzen einer Region hängenden Inhalten umgegangen werden soll:

hidden – Überhängendes wird nicht angezeigt;

scroll – Überhängendes wird nicht angezeigt, aber es erscheinen Scrollbalken am Rand der Region, um die nicht sichtbaren Stellen zu erreichen;

fill – Die Inhalte werden paßgenau in die Region gesetzt, auch wenn sich das Seitenverhältnis verändert;

meet – Die Inhalte werden so angepaßt, daß sie in ihrer Höhe oder Breite genau in die Region passen, dabei aber noch alle Inhalt sichtbar sind; dabei erscheint zwingend ein leerer Bereich an der nicht passenden Seite;

slice – Die Inhalte werden so angepasst, daß sie in ihrer Höhe oder Breite genau in die Region passen; dabei gibt es zwingend Überhang an der nicht passenden Seite, welcher abgeschnitten wird.

Elemente: <bottom> / <right>

<bottom>/<right>/<width>/<height>	
attributes	none
parents	<region>
children	none

Tabelle 4: Die Positionierungselemente

Treten nur gemeinsam als Kinder von <region> auf. Zusammen geben sie die Position der rechten, unteren Ecke einer Region an. Notwendig, wenn man die Region nicht mit <width> und <height> spezifizieren möchte.

Element: <width> / <height>

Treten nur gemeinsam als Kinder von <region> auf. Zusammen geben sie die Breite und Höhe einer Region an. Notwendig, wenn man die Region nicht mit <bottom> und <right> spezifizieren möchte. Erwartet wird entweder eine Angabe in Pixeln oder eine prozentuale Angabe, die sich an dem übergeordneten Element orientiert.

Element: <video>

<video>	
attributes	id src color type repeat
parents	<region> background-color: #00ff00; color: black;"><a>, <area>
children	<begin>, <end> <clipbegin>, <clipend> <dur> background-color: #800080; color: white;"><event> background-color: #696969; color: white;"><link> background-color: #800080; color: white;"><behaviour>

Tabelle 5: Das <video>-Element

Mit diesem Element kann eine Video-Datei, die durch das Attribut src referenziert wird, in eine Region gelegt werden. Mit dem Attribut type kann darüber hinaus der Typus der Video-Datei noch näher bestimmt werden. Ein Videoelement hat außerdem eine Hintergrundfarbe und eine ID, durch die das Element eindeutig bestimmt wird. Außerdem kann durch die Angabe des Attributs repeat

angegeben werden wie oft das Video wiederholt werden soll.

Mittels der Kind-Elemente `<begin>` und `<end>` kann weiterhin angegeben werden, zu welchem Zeitpunkt der Präsentation das Video angezeigt werden soll. `<dur>` gibt dabei die Abspieldauer an. `<clipbegin>` und `<clipend>` können ebenfalls verwendet werden, um die Abspieldauer des Clips anzugeben: Sie haben die Funktion von Abspielmarken, wenn beispielsweise nur ein bestimmter Bereich der Video-Datei wiedergegeben werden soll.

Attribut: src

Mit diesem Attribut wird beim Element `<video>` der Pfad zu der zu verwendenden Video-Datei angegeben. Der Pfad wird entweder im Format einer URL oder eines Pfades auf dem lokalen Dateisystem erwartet. Es findet auch bei den anderen Medientypen und darüber hinaus an anderen Stellen von NRML Verwendung, ist aber zum Beispiel beim Element `<text>` optional, da auch direkt Text als Kind des Elementes eingegeben werden kann.

Attribut: type (bei Medienelementen)

Optional kann bei allen Medientypen außer Text der MIME-Typ der verwendeten Medienkodierung angegeben werden, um den Player bei der Suche nach dem entsprechenden CoDec zu unterstützen. MIME-Typen sind beispielsweise `“video/mpeg”` für ein Video oder `“audio/x-mp3”` für eine Audio-Datei. Näheres zu MIME-Typen findet man unter [7] nachgelesen werden.

Attribut: repeat (bei Medienelementen)

Ein ganzzahliger Wert zur Angabe, wie oft das Medium wiederholt abgespielt werden soll. Soll das Medium unendlich oft wiederholt werden, kann `inf` angegeben werden.

<code><audio></code>	
attributes	id src type repeat
parents	<code><region></code>
children	<code><begin></code> , <code><end></code> <code><clipbegin></code> , <code><clipend></code> <code><dur></code>

Tabelle 6: Das `<audio>`-Element

Element: <audio>

Das Element <audio> verhält sich sehr ähnlich gegenüber dem <video>-Element. Es besitzt allerdings keine Hintergrundfarbe; außerdem ist es nicht möglich, Ausschnitte aus der Audio-Datei oder insgesamt zu verlinken.

Elemente: / <text>

	
attributes	id src type color
parents	<region>
	<a>, <area>
children	<begin>, <end>
	<dur>
	<event>
	<link>
	<behaviour>

Tabelle 7: Das -Element

<text>	
attributes	id src type color
parents	<region>
	<a>
children	<begin>, <end>
	<dur>
	<a>
	<event>
	<link>
<behaviour>	

Tabelle 8: Das <text>-Element

Diese Elemente verhalten sich ähnlich gegenüber dem <video>-Element. Sie besitzen allerdings keine Möglichkeit zur Wiederholung sowie Ausschnittmarken zu definieren, weil sie keine zeitlichen Medien sind. Teile des Textes können verlinkt werden, näheres dazu im Abschnitt über das <a>-Element.

Element: <object>

<object>	
attributes	id src color type repeat
parents	<region> background-color: #00FF00; color: white;"><a>, <area>
children	<begin>, <end> background-color: #808080; color: white;"><dur> background-color: #800080; color: white;"><event> background-color: #4B4B4B; color: white;"><link> background-color: #800080; color: white;"><behaviour>

Tabelle 9: Das <object>-Element

Mit dem <object>-Element können andere Medien zeitabhängig eingebunden werden. Über das src-Attribut wird die Quelle referenziert, während über das type-Attribut wie gewohnt der Typ des Objektes näher bestimmt werden kann. Ebenso hat das Element eine Hintergrundfarbe und es kann eine Wiederholungsrate bestimmt werden. Nicht unterstützt werden allerdings die Elemente <clipbegin> und <clipend>, auch wenn manche Objekte sie logischerweise unterstützen müßten.

Elemente: <begin> / <end> / <dur>

Mit diesen Elementen kann definiert werden, in welchen Zeiträumen innerhalb einer Präsentation ein Medium angezeigt wird. Dafür nötig sind Kombinationen aus <begin> und <end>, also die Angabe eines exakten Anfangspunktes sowie eines Endpunktes, oder <begin> und <dur>, also die Angabe eines Anfangspunktes und der Anzeigedauer des Mediums. Die Zeitangaben werden im Format HH:MM:ss.mm erwartet. Wichtig ist die korrekte Reihenfolge der Elemente: Nach einem <begin> folgt immer entweder ein <end> oder ein <dur>. Danach kann wieder mit <begin> ein neuer Zeitraum gestartet werden.

<begin>/<end>/<dur>	
attributes	none
parents	<video>, <audio> background-color: #00FF00; color: white;">, <text> background-color: #00FF00; color: white;"><a>, <area>
children	none

Tabelle 10: Elemente zur zeitl. Beeinflussung der Medienanzeige

Elemente: <clipbegin> / <clipend>

<clipbegin>/<clipend>	
attributes	none
parents	<video>, <audio>
children	none

Tabelle 11: Elemente zur Definition von Ausschnitten

Hiermit kann ein Ausschnitt aus einem Video- oder Audio-Clip definiert werden, welcher anstelle des gesamten Clips abgespielt wird. Erneut wird eine Angabe in HH:MM:ss.mm erwartet. Die Definition ist für jeden angegebenen Zeitraum, in dem das Medium angezeigt wird, erneut zu tätigen; der korrekte Sequenz ist hierbei: <begin>, <clipbegin>, <clipend>, <end>/<dur>. Danach kann ein neuer Zeitraum folgen.

Element: <a>

<a>	
attributes	href
parents	<region> background-color: #00b050;"> <text>
children	<begin>, <end> <dur>
	<video>, <object> , <text>

Tabelle 12: Das <a>-Element

Hiermit können ganze Medien oder nur Teile innerhalb von Texten verlinkt werden. Zur Verlinkung eines gesamten Mediums wird das Element oberhalb des Medienelements platziert. Mittels des Attributs href kann dann das Ziel des Links eingegeben werden. Zur Verlinkung von Teilen eines Textes wird das Element entsprechend innerhalb des Textes um die zu verlinkenden Teile platziert. Dieses Element kann ebenfalls mittels der <begin>/<end>/<dur>-Elemente zeitabhängig eingeblendet werden.

Element: <area>

Mit dem <area>-Element können Ausschnittbereiche innerhalb von Bildern und Videos verlinkt werden. Die Form des Ausschnittes ist dabei rechteckig und wird mit zwei X/Y-Koordinaten angegeben.

<area>	
attributes	href shape coords
parents	<region>
children	<begin>, <end> <dur> <div style="background-color: #00FF00; color: white; padding: 2px;"><video></div> <div style="background-color: #00FF00; color: white; padding: 2px;"></div>

Tabelle 13: Das <area>-Element

Attribut: href

Mit diesem Attribut wird das Ziel eines Links angegeben. Erwartet wird eine eindeutige URI.

Attribut: coords

Es werden zwei X/Y-Koordinaten erwartet, wobei die einzelnen Zahlen durch Kommata getrennt werden müssen. Zwischen den Koordinaten wird ein Viereck aufgespannt, in dessen Innenraum der Link gültig ist. Weitere Formen sind momentan nicht vorgesehen.

Attribut: shape

Hiermit wird die Form der Ebene angegeben, innerhalb der der Link gültig ist. Momentan sind keine anderen vorgesehen als ein Viereck, für das der Wert shape angegeben werden muß.

Element: <character>

<character>	
attributes	id src type
parents	<region>
children	<event> <div style="background-color: #696969; color: white; padding: 2px;"><link></div> <div style="background-color: #800080; color: white; padding: 2px;"><behaviour></div>

Tabelle 14: Das <character>-Element

Mit diesem Element können externe Charakter-Dateien in NRML eingebunden werden. Mittels des type-Attributs wird der Typ angegeben und mit src die Quelle der Datei. Zugeordnet werden können dem Charakter entweder getrennte oder über einen Link zusammenhängende Events und Verhaltensweisen.

Attribut: type (bei <character>)

Es wird folgendes erwartet: Entweder XML für eine Spezifikation, die noch vom Kernmodul interpretiert wird, oder Binary für eine Behandlung des Charakters wie ein Video.

Element: <equipment>

<equipment>	
attributes	id type
parents	<region>
children	<event>
	<link>
	<behaviour>
	<input>

Tabelle 15: Das <equipment>-Element

Dieses Element ist für Eingabemöglichkeiten seitens des Nutzers vorgesehen. Hier können mittels der <input>-Elemente Formulare eingegeben werden. Außerdem können wie bei einem Charakter Verhaltensweisen und Events oder eine Kombination der beiden über Links angegeben werden.

Element: <input>

<input>	
attributes	id
	color
	textcolor
	src
	type
parents	<equipment>
children	none

Tabelle 16: Das <input>-Element

<input>-Elemente stellen Eingabemöglichkeiten für den Nutzer dar. Mittels des type-Attributes können über die Angabe der Schlüsselwörter radio, button oder text Radio-Buttons, Knöpfe und Texteingabefenster eingefügt werden. Mit den Attributen color und textcolor können Hintergrundfarbe und Textfarbe festgelegt werden. Mit src können fremde Grafiken als Knöpfe verwendet werden.

Element: <event>

<event>	
attributes	type repeat
parents	<region>
	<video>, <audio>
	, <text>
	<character>
	<equipment>
children	none

Tabelle 17: Das <event>-Element

Ein Event kann auf ein Medienelement, einen Charakter, ein Equipment oder eine ganze Region gelegt werden. Hiermit wird ein Trigger definiert, mit dem eine bestimmte Verhaltensweise (siehe <behaviour>) ausgelöst wird, die ebenfalls auf dem Element liegt.

Attribut: type (bei <event>)

Über das type-Attribut wird festgelegt, auf welche Nutzeraktion das entsprechende Element reagieren soll. Möglichkeiten sind:

- onnothing (keine Aktion),
- on- oder onclick (Trigger bei Mausklick oder Maus loslassen),
- on- oder offfocus (Reaktion bei Element aktiv oder inaktiv),
- on- oder offlook (Reaktion bei Zeiger über Element oder geht von Element weg).

Attribut: repeat (bei <event>)

Über das Attribut repeat kann man einstellen, ab wann das Element reagieren soll. Also Beispielsweise in Kombination mit onnothing erst nach 2, 3 oder 4 Sekunden oder in Kombination mit onclick nach 1 oder 2 Klicks.

Element: <behaviour>

Eine Verhaltensweise eines Elements wird durch ein <event> ausgelöst, daß wie das <behaviour> innerhalb des entsprechenden Elements zu finden ist. Es gibt mehrere Möglichkeiten von Aktionen und Subaktionen, die ein Event auslösen kann; diese werden mit den Attributen action und subaction angegeben.

<behaviour>	
attributes	id action subaction
parents	<region>
	<video>, <audio> , <text>
	<character> <equipment>
children	none

Tabelle 18: Das <behaviour>-Element

Attribut: action

Bei einem Audio oder einem Video können folgende Aktionen ausgelöst werden: play und stop. Die Geschwindigkeit des Abspielvorgangs wird in subaction angegeben. Bei Medienelementen allgemein kann die Aktion cycle ausgelöst werden, mit dem eine Art Slideshow gestartet werden kann – beispielsweise können Bilder einer Liste nacheinander angezeigt werden. Ein Charakter kann mit den Verhaltensweisen talk und stop belegt werden, um etwas vorzutragen; die Quelle dessen wird in subaction angegeben. Außerdem kann allgemein die Lautstärke der Umgebung durch die Schlüsselwörter volumeup und volumedown angepasst werden.

Attribut: subaction

Wird die Aktion play ausgeführt, kann die Geschwindigkeit des Abspielvorgangs mit einem Wert aus {-2,-1,0,1,2} angegeben werden. Ist die Aktion cycle aktiv, kann in subaction die Richtung angegeben werden, also forward oder backward. Bekommt ein Charakter in action den Auftrag zu sprechen, kann hier die Quelle angegeben werden. Also entweder ein Text oder der Pfad zu einer Audiodatei.

Element: <link>

Das Element <link> dient dazu, eine Verknüpfung zu einem Event oder einer Verhaltensweise herzustellen. Diese Verknüpfung kann anstelle einer Neudefinition innerhalb der Elemente verwendet werden und dient der Möglichkeit für den Autor, vielbenutzte Events oder Verhaltensweisen mehrmals zu verwenden.

<link>	
attributes	id type
parents	<region>
	<video>, <audio>
	, <text>
	<character>
	<equipment>
children	none

Tabelle 19: Das <link>-Element

5.2.3 Narrationselemente

Element: <narration>

<narration>	
attributes	none
parents	<nxml>
children	<chapter>

Tabelle 20: Das <narration>-Element

Analog zum <template>-Element bei den Präsentationselementen ist dies das umschließende Element um die gesamte Narration. Eine Narration besteht aus mehreren Kapiteln, die unterhalb dieses Elements angegeben werden können.

Element: <chapter>

<chapter>	
attributes	id title
parents	<narration>
children	<node>

Tabelle 21: Das <chapter>-Element

Kapitel, die mittels dieses Elementes definiert werden können, dienen der Gliederung von (Erzähl-) Knoten der Narration in einzelne Themengebiete. Nützlich ist dies, um innerhalb der Narration zu überprüfen, welche Themen der Nutzer schon besucht oder nicht besucht hat, um seine Interessen festzustellen und den Plotverlauf für ihn daraufhin anzupassen. Kapitel bestehen aus Knoten, die unterhalb dieses Elements angegeben werden können, und besitzen eine ID sowie einen beschreibenden Titel.

Attribut: title

Mittels dieses Attributs kann einem Element ein beschreibender Titel hinzugefügt werden. Notwendig ist dies bei Kapiteln und Knoten.

Element: <node>

Dieses Element repräsentiert einen einzelnen Knoten innerhalb des Systems. Ein Knoten enthält ein zugewiesenes Template, in dem schon Inhalt zu finden sein kann. Mittels des <on>-Elements kann dann etwaiger Inhalt zu bestimmten Situationen angepaßt und verändert werden. Außerdem können entsprechend der Nutzer-Historie Nachfolgerknoten vorgeschlagen werden.

Ein Knoten besitzt neben einem Template noch eine eindeutige ID und einen Titel.

<node>	
attributes	id name template
parents	<chapter>
children	<suggest> <add> <replace> <on>

Tabelle 22: Das <node>-Element

Attribut: template

Mit diesem Attribut muß ein Template angegeben werden, um die Inhalte innerhalb des Knotens anzuzeigen. Es wird ein relativer Pfad zu einem NRML-Template erwartet.

Element: <suggest>

<suggest>	
attributes	id
parents	<node>
children	none

Tabelle 23: Das <suggest>-Element

Das <suggest>-Element dient dazu, Nachfolgeknoten vorzuschlagen, die der Nutzer nach dem aktuellen Knoten besuchen sollte. Die angegebenen Knoten eignen sich am Besten, um den Plot voran zu bringen. Anzugeben ist dazu nur der entsprechende Knoten per ID.

Element: <on>

<on>	
attributes	visited unvisited predecessor
parents	<on> <node>
children	<add> <replace> <on>

Tabelle 24: Das <on>-Element

Dieses Element stellt einige Events zur Verfügung, mit denen Inhalte kontextabhängig an den Nutzer angepasst werden können: Mit den Attributen visited, unvisited und predecessor können AND- und OR-Abhängigkeiten abgebildet werden. Beispielsweise:

<on visited="n1,n2,n3" unvisited="n4,n5">.../on> - Wenn Knoten n1,n2 und n3 besucht ODER Knoten n4 und n5 unbesucht, dann bearbeite Inhalte.

<on visited="n1"><on visited="n2">...</on></on> - Wenn Knoten n1 UND Knoten n2 besucht, dann bearbeite Inhalte.

Attribute: visited, unvisited, predecessor

Angabe von Knoten-IDs. Wenn mehrere eingefügt werden, durch Kommata getrennt.

Elemente: <add> / <replace>

<add>/<replace>	
attributes	id src
parents	<on> <node>
children	<video>, <audio> , <text>

Tabelle 25: Elemente zur Manipulation von Inhalten

Mit diesen Elementen kann man etwaige in den Templates vordefinierte Inhalte austauschen oder neue Inhalte hinzufügen. Anzugeben ist dazu mittels des Attributs id die eindeutige ID einer Region, der etwas hinzugefügt wird, oder die ID eines Medienelements, an welches Inhalte angehängt

werden sollen oder das ersetzt werden soll. Mit dem src-Attribut müssen die einzufügenden Medieninhalte referenziert werden.

**Element: **

	
attributes	id
parents	<on> <node>
children	none

Tabelle 26: Das -Element

Mit dem -Element können entsprechend Medieninhalte aus den Templates entfernt werden ohne etwas anderes hinzuzufügen. Hierfür notwendig ist wie bei den <add>- und <replace>-Elementen die ID der Region oder des zu löschenden Medienelements.

5.3 Sprachrealisierung

In diesem Kapitel werden die Techniken vorgestellt, mit denen die in Kapitel 5 definierte Gramatik implementiert und der Parser erstellt wurde.

5.3.1 XML Schema Definition (XSD)

XSD (W3C, 2005) ist eine Empfehlung des W3C zur Definition von XML-Dokumentstrukturen. Es ist verwandt mit den DTDs (Document Type Definition) (W3C, 2005), mit denen ebenfalls XML-Dokumentstrukturen definiert werden können. Allerdings sind XML-Schemata durch ihre erweiterten Möglichkeiten wesentlich komplexer. Die Strukturen werden außerdem im Gegensatz zu den DTDs in Form eines XML-Dokuments beschrieben und es werden eine große Anzahl an verschiedenen Datentypen unterstützt. Die oben definierte Sprache wurde als Teil dieser Arbeit in einer XSD implementiert und findet sich im Anhang.

5.3.2 Parser

Ein Parser ist ein Programm, das überprüft, ob ein gegebener String zur Sprache einer bestimmten Grammatik gehört. Diese Grammatik wurde in Kapitel 5 definiert und mittels XSL in eine maschinenlesbare Form übertragen. Der Parser bekommt als Eingabe eine XML-Datei und vergleicht deren Inhalt mit der im XSL-Format gegebenen Grammatik. Während des Parsens erfolgt eine syntaktische Überprüfung der Eingangsdaten. Bei der Überprüfung wird in der Regel aus den Daten ein sogenannter Ableitungsbaum aufgebaut, um die Daten anschließend weiterverarbeiten zu

können. Diese Weiterverarbeitung erfolgt später durch den Interpreter, der als Modul von Jeherazade noch implementiert werden muß.

Verwendet wurde als Parser der SAX (SourceForge, 2005) und DOM (W3C, 2005) Parser des Apache Xerces2 Projekts (Apache, 2005). Der Code dieses Parsers ist Open-Source, kann also komplett eingesehen und selbst weiterentwickelt werden, was für die Zwecke dieser Arbeit optimal ist. Xerces2 besitzt einen großen Verbreitungsgrad, weswegen in diesem Bereich auf vorhandene Problemlösungsstrategien zurückgegriffen werden kann.

5.3.3 Document Object Model (DOM)

Das DOM ist ein programmiersprachenunabhängiges Application Programming Interface (API). Die API stellt Klassen, Methoden und Funktionen bereit, mit denen auf XML- oder HTML-Dokumente zugegriffen werden kann und mit denen diese auch verändert oder erstellt werden können. Damit erleichtert das DOM die Arbeit mit Dateien im XML-Format, da nicht mehr selbst ein kompletter Parser programmiert werden muß. Das DOM besteht aus in einer Baumstruktur organisierten Knoten, die die Elemente und Attribute der gegebenen XML-Datei enthalten. Damit kann relativ komfortabel auf die Daten zugegriffen werden; jedoch führt diese Methode bei großen Dokumenten zu einer schlechten Performance.

5.3.4 Simple API for XML (SAX)

SAX besitzt im Wesentlichen die gleichen Fähigkeiten wie DOM, hat jedoch eine andere Herangehensweise: Es erstellt aus den gegebenen XML-Daten keine Baumstruktur, so daß die Performance besser ist. Die Daten werden sequentiell geparsed und dafür von der Anwendung sogenannte Callback-Funktionen erstellt, mittels denen diese im späteren Verlauf auf die entsprechenden Daten zugreifen kann. Damit werden nur die Eingabedaten betrachtet, die für die Anwendung von Belang sind.

6 Zusammenfassung und Ausblick

Das Ziel der Arbeit, eine Narrations-Beschreibungs-Sprache für das System Jeherazade zu entwickeln, wurde erreicht. Ebenso wurden alle notwendigen Features, seien sie explizit durch die Arbeit vorgegeben gewesen oder im Laufe der Konzeptionsphase entstanden, implementiert. Inwiefern Elemente fehlen, kann sich nur im Einsatz zeigen; jedoch sind die notwendigen Funktionen und Möglichkeiten vorhanden.

Es gibt unter dem Verbund Jeherazade noch weitere Studienarbeiten, mit denen diese Arbeit zahlreiche Schnittstellen besitzt. Darum wird es sowieso unumgänglich sein, daß NRML noch öfter angefasst und verändert wird. Beispielsweise wird der Autor der Studienarbeit, die die Bühne des Systems behandelt, auf zahlreiche Ideen zur Präsentation kommen wie Möglichkeiten der Textformatierung. Auch wird der Autor der Diplomarbeit, die das Kernmodul des Systems behandelt, vermutlich auf Ideen kommen wie neue Möglichkeiten zur Narration. Veränderungen an NRML gestalten sich allerdings sehr einfach; so müssen doch dafür nur die Dateien zur Grammatik angepaßt und die neuen Elemente hinzugefügt werden.

Außerdem entscheidend in diesem Zusammenhang ist die Entwicklung der Charaktere. Optimal wäre aus der Sichtweise dieser Arbeit eine Schnittstelle, der man einen Audiostream übergibt, der einen gesprochenen Text enthält. Der Charakter bewegt dann seinen Mund gemäß der Charakteristik des Streams. Angenehm wären auch zusätzliche Signale, so daß man das Mienenspiel des Charakters manipulieren könnte.

Kommt das Gesamtsystem zur Vollendung, sollte auch über ein Autorentool für NRML nachgedacht werden. Die Autoren der Narrationen werden in den seltensten Fällen Personen mit Programmierkenntnissen sein; wie in Kapitel 3 dargestellt, werden eher Redakteure diese Aufgabe besitzen. Solch ein Editor stellt sich optimalerweise im Bereich der Templateeditierung als WYSIWYG-System dar: Man sollte mit einem Zeigegerät Regionen erstellen, frei positionieren und mit statischem oder dynamischem Content versehen können. Schwieriger gestaltet sich das schon bei einem Editor für den Narrationsbereich: Auf abstrakte Weise müßten die Knoten mit ihrem Content und die Beziehungen untereinander dargestellt werden.

Natürlich sollte aber als nächster Schritt nun der Interpreter folgen, damit die Sprache überhaupt zum Einsatz kommen kann. Dieser muß sich zwar als Subsystem in das Kernmodul eingliedern, jedoch kann er auch getrennt davon entwickelt und schließlich in Jeherazade integriert werden.

Glossar

Bühne – Auf der Bühne findet die Handlung der Narration statt; sie stellt die Präsentation dar. Sie ist damit das Ausgabemedium eines Knotens für den Nutzer.

Charakter – Charaktere sind Personen innerhalb der Präsentation. Zum Beispiel ein Gesicht, welches dem Nutzer etwas erzählt. Charaktere haben ihren Platz auf der Bühne.

Equipment – Ein Equipment ist ein Eingabewerkzeug für die Nutzer des Systems, zum Beispiel ein Textfenster, wo etwas eingegeben werden kann, oder ein Button. Equipment hat seinen Platz auf der Bühne.

Handlung – Die Handlung (*Plot*) ist das Kernstück der Narration. Sie motiviert den Nutzer, sich weiter durch die Narration und damit die Ausstellung zu bewegen.

Knoten – Eine Ausstellung besteht aus mehreren Exponaten oder abstrakt mehreren Punkten von Interesse. Diese Punkte werden Knoten genannt und bilden eine feste Position innerhalb der Narration. Ein Knoten besteht aus der Bühne und einigen Eingabewerkzeugen für den Nutzer.

Narration – Die Narration (Erzählung) besteht aus mehreren Knoten, um die herum eine Handlung aufgebaut wird. Jeder Nutzer bekommt seine eigene Narration gemäß seiner Interessen präsentiert.

Nutzer – Der Nutzer bewegt sich durch die Narration beziehungsweise die Ausstellung und besucht die für ihn interessanten Knoten.

Präsentation – Die Präsentation der Narration findet an den Knoten auf der Bühne statt. Sie besteht nicht nur aus den Inhalten, sondern auch aus Charakteren und Equipment.

Literatur und weiterführende Links

Bücher

- Bulterman, D.C.A. & Rutledge, L. (2004). SMIL 2.0 – Interactive Multimedia for Web and Mobile Devices. Springer, Berlin
- Dumbill, E. & van der Vlist, E. & Jelliffe, R. & Dodds, L. (2001). xml.com – The Guide to XML Schemata. O'Reilly, Köln
- Stein, M. & Dellwig, I. (2001). XML. Addison-Wesley, München
- van der Vlist, E. (2002). XML Schema. O'Reilly, Köln

Tagungsbeiträge

- Adams, E. (1999). Three Problems for Interactive Storytellers. GamaSutra (www.gamasutra.com)
- Adler, B. & Kostanick, C. & Stein, M. & Urban, M. & Usui, W. (1981). A Brief Description of UCLA Dungeon Definition Language (DDL). University of California, California
- Boll, S. & Eidenberger, H. (2002). Medienmix – SMIL 2.0: Markup-Sprache für Multimedia-Präsentationen. iX 11/2002
- Brengle, T. & Cunniff, R. (1987). The ADL Programmer's Reference Manual. Hewlett-Packard Company, California
- Brooks, K.M. (1997). Programming Narrative. MIT Media Lab
- Gerdt, P. & Kommers, P. & Suhonen, J. & Sutinen, E. (2002). StoryML: An XML Extension for Woven Stories. University of Joensuu, Finland & University Twente, The Netherlands
- Gerdt, P. & Suhonen, J. & Sutinen, E. (2002). Using StoryML for Content Representation. University of Joensuu, Finland
- Hoffmann, P. (2004). Expanding the Story Line. University of Luebeck, Germany
- Hoffmann, P. & Herczeg, M. (2004). Distributed Storytelling for Narrative in Spacious Areas. University of Luebeck, Germany
- Hu, J. & Fejis, L. (2003). An Adaptive Architecture for Presenting Interactive Media onto Distributed Interfaces. Philips Research Laboratories, The Netherlands & Eindhoven. University of Technology, The Netherlands
- Hu, J. (2003). StoryML: Enabling Distributed Interfaces for Interactive Media. Eindhoven, University of Technology, Netherlands
- Littlejohn, R. (2001). The Need to Adapt the Tools of Drama to Interactive Storytelling. GamaSutra (www.gamasutra.com)

Webseiten

Apache (12/2004). Webseite zum Apache Xerces2 Projekt.

<http://xml.apache.org>

Macromedia (12/2004). Produktseite zu Flash.

<http://www.macromedia.com>

OpenSource (12/2004). Webseite zur OpenSource-Programmier-Plattform Eclipse.

<http://www.eclipse.org>

r.a.int-fiction (01/2005). FAQ zur Newsgroup Interactive Fiction.

<http://www.plover.net/~textfire/raiffaq/FAQ>

SourceForge (01/2005). Webseite zum Projekt SAX (Simple API for XML).

<http://sax.sourceforge.net>

TeamOne (01/2005). Webseite zu MIME-Types.

<http://de.selfhtml.org/diverses/mimetypen.htm>

W3C (01/2005). Spezifikationen von SMIL.

<http://www.w3.org/AudioVideo>

W3C (01/2005). Webseite zu DOM (Document Object Model).

<http://www.w3.org/DOM>

W3C (01/2005). Webseite zu XML Schemata.

<http://www.w3.org/TR/xmlschema-0>

Wikipedia (01/2005). Webseite zu DTD (Document Type Definition).

<http://de.wikipedia.org/wiki/DTD>

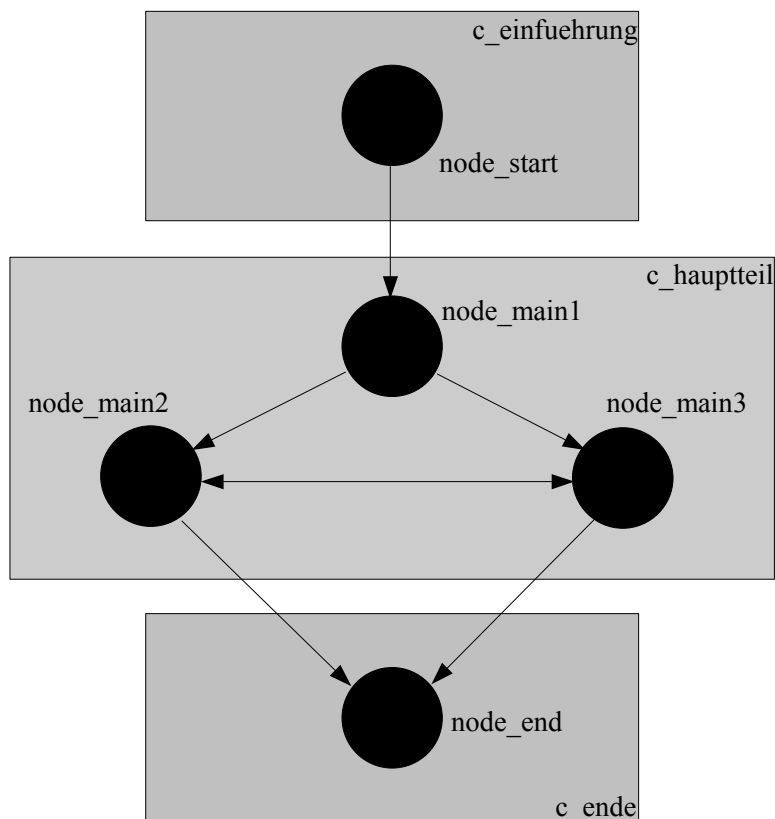
Anhang

In diesem Kapitel ist der komplette Source-Code zur Arbeit aufgelistet.

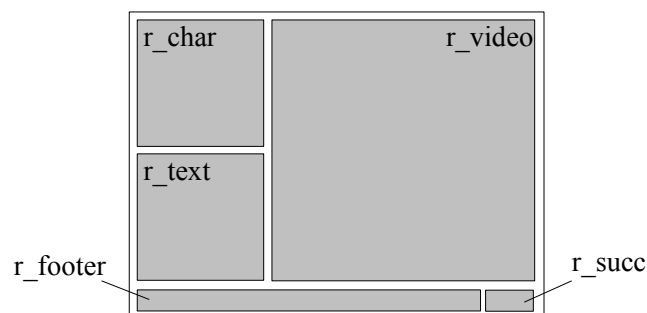
A. Beispiel

Hier findet sich ein Beispiel für die entwickelte Sprache.

Im Folgenden ist die Narrationsstruktur für dieses einfache Beispiel abgebildet. Es teilt sich in drei Kapitel, wobei das erste und letzte Kapitel nur einen Knoten besitzt. Das mittlere Kapitel besitzt drei Knoten, von denen durch die Struktur zwei angesehen werden müssen und ein Knoten optional ist.



Die folgende Skizze umfasst die Struktur des Templates. Das Beispiel besitzt insgesamt fünf



Regionen, wobei jeweils eine für textuellen und audiovisuellen Inhalt verwendet wird. Eine weitere Region beinhaltet einen Charakter oder Avatar, der den textuellen Inhalt vorlesen könnte oder etwas getrennt vom Video zum Exponat erzählen könnte. In diesem Beispiel wird allerdings nur eine Referenz auf eine Charakter-Datei angegeben. Zuletzt gibt es noch eine statische Fußzeile, in der beispielsweise der Name der Ausstellung stehen könnte. Außerdem gibt es eine kleine Region, deren Inhalt Vorschläge für die nächsten zu besuchenden Knoten umfasst.

nrml-template-example.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<nrml xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:noNamespaceSchemaLocation='nrml.xsd'>

  <template color="#000000" sound="pfad/zu/sound.mp3" volume="20%">

    <!-- Erstellung einer Region zur Video-Darstellung -->
    <region id="r_video" x="120" y="10" z="5" color="#cccccc" fit="meet" volume="50%">
      <width>200</width><height>200</height>
      <!-- Position für dynamischen Content (Video z.B.) -->
    </region>

    <!-- Erstellung einer Region für einen Charakter -->
    <region id="r_char" x="10" y="10" z="0" color="#aaaaaa" volume="50%">
      <width>100</width><height>100</height>
      <!-- Position für dynamischen Content (Character z.B.) -->
    </region>

    <!-- Erstellung einer Region zur Text-Darstellung -->
    <region id="r_text" x="10" y="120" z="0" color="#aaaaaa">
      <width>100</width><height>100</height>
      <!-- Position für dynamischen Content (Text z.B.) -->
    </region>

    <!-- Erstellung einer Region zur Präsentation Nachfolgeknoten -->
    <region id="r_succ" x="220" y="230" z="0" color="#ffffff">
      <width>100</width><height>40</height>
      <!-- Position für dynamischen Content (Nachfolgeknoten) -->
    </region>

    <!-- Erstellung einer Region für Footer -->
    <region id="r_footer" x="10" y="230" z="0" color="#ffffff">
      <width>200</width><height>40</height>
      <text id="footer_text">
        Demonstration der einfachen Möglichkeiten der Sprache NRML - entwickelt am
        <a href="http://www.imis.uni-luebeck.de">IMIS</a>
      </text>
    </region>

  </template>
</nrml>
```

nrml-narration-example.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<nrml xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:noNamespaceSchemaLocation='nrml.xsd'>

  <narration>

    <chapter title="Einführung" id="c_einfuehrung">
      <node id="node_start" title="Anfang" template="nrml_template.xml">
        <add src="pfad/videos/einfuehrung.mpg" id="vid_einf"/>
        <add src="pfad/chars/einfuehrung.chr" id="char_einf"/>
        <add src="pfad/texts/einfuehrung.txt" id="text_einf"/>
        <suggest id="node_main1" region="r_succ"/>
      </node>
    </chapter>

    <chapter title="Hauptteil" id="c_hauptteil">
```

```

<node title="Knoten 1" template="nrml_template.xml" id="node_main1">
  <on visited="c_einfuehrung">
    <add src="pfad/videos/node1.mpg" id="vid_1"/>
    <add src="pfad/chars/node1.chr" id="char_1"/>
    <add src="pfad/texts/node1.txt" id="text_1"/>
    <suggest id="node_main2" region="r_succ"/>
    <suggest id="node_main3" region="r_succ"/>
  </on>
  <on unvisited="c_einfuehrung">
    <suggest id="node_start" region="r_succ"/>
  </on>
</node>
<node title="Knoten 2" template="nrml_template.xml" id="node_main2">
  <on visited="c_einfuehrung,node_main1">
    <add src="pfad/videos/node2.mpg" id="vid_2"/>
    <add src="pfad/chars/node2.chr" id="char_2"/>
    <add src="pfad/texts/node2.txt" id="text_2"/>
    <on unvisited="node_main3">
      <suggest id="node_main3" region="r_succ"/>
    </on>
    <suggest id="node_end" region="r_succ"/>
  </on>
  <on unvisited="c_einfuehrung">
    <suggest id="node_start" region="r_succ"/>
  </on>
  <on unvisited="node_main1">
    <suggest id="node_main1" region="r_succ"/>
  </on>
</node>
<node title="Knoten 3" template="nrml_template.xml" id="node_main1">
  <on visited="c_einfuehrung,node_main1">
    <add src="pfad/videos/node3.mpg" id="vid_2"/>
    <add src="pfad/chars/node3.chr" id="char_2"/>
    <add src="pfad/texts/node3.txt" id="text_2"/>
    <on unvisited="node_main2">
      <suggest id="node_main2" region="r_succ"/>
    </on>
    <suggest id="node_end" region="r_succ"/>
  </on>
  <on unvisited="c_einfuehrung">
    <suggest id="node_start" region="r_succ"/>
  </on>
  <on unvisited="node_main1">
    <suggest id="node_main1" region="r_succ"/>
  </on>
</node>
</chapter>

<chapter title="Schluss" id="c_ende">
  <node id="node_start" title="Anfang" template="nrml_template.xml">
    <on visited="c_einfuehrung,c_hauptteil">
      <add src="pfad/videos/schluss.mpg" id="vid_einf"/>
      <add src="pfad/chars/schluss.chr" id="char_einf"/>
      <add src="pfad/texts/schluss.txt" id="text_einf"/>
    </on>
    <on unvisited="c_hauptteil">
      <suggest id="node_main1" region="r_succ"/>
    </on>
    <on unvisited="c_einfuehrung">
      <suggest id="node_einf" region="r_succ"/>
    </on>
  </node>
</chapter>

</narration>

</nrml>

```

B. Grammatik

Die mittels XML Schema definierte Grammatik von NRML.

nrml.xsd

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <xsd:include schemaLocation="nrml_simpleTypes.xsd"/>
  <xsd:include schemaLocation="nrml_complexTypes_narration.xsd"/>
  <xsd:include schemaLocation="nrml_complexTypes_template.xsd"/>

  <!-- Referenz auf das Root-Element als Ausgangspunkt -->
  <xsd:element name="nrml">
    <xsd:complexType>
      <xsd:choice>
        <xsd:sequence>
          <xsd:element name="template" type="templateType" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:sequence>
          <xsd:element name="narration" type="narrationType" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
      </xsd:choice>
    </xsd:complexType>
    <xsd:key name="k_region">
      <xsd:selector xpath="./template/region"/>
      <xsd:field xpath="@id"/>
    </xsd:key>
  </xsd:element>
</xsd:schema>
```

nrml_simpleTypes.xsd

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <!-- Definition einfacher Typen -->

  <!-- Typ fuer ID-Bezeichnung -->
  <xsd:simpleType name="idType">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
  <!-- Typ fuer Element-Namen -->
  <xsd:simpleType name="nameType">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
  <!-- Typ fuer eine Koordinaten-Position wie x oder y -->
  <xsd:simpleType name="positionType">
    <xsd:restriction base="xsd:integer"/>
  </xsd:simpleType>
  <!-- Typ fuer eine Laengen-Angabe wie width oder height -->
  <xsd:simpleType name="sizeType">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d+"/>
      <xsd:pattern value="\d{1,2}%"/>
      <xsd:pattern value="100%"/>
    </xsd:restriction>
  </xsd:simpleType>
  <!-- Typ fuer eine Volume-Angabe -->
  <xsd:simpleType name="soundType">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d+"/>
      <xsd:pattern value="\d{1,2}%"/>
    </xsd:restriction>
  </xsd:simpleType>
```

```

        <xsd:pattern value="100%"/>
    </xsd:restriction>
</xsd:simpleType>
<!-- Typ fuer Sources -->
<xsd:simpleType name="srcType">
    <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
<!-- Typ fuer Audio-, Video-, Bild-Typen -->
<xsd:simpleType name="typeType">
    <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
<!-- Typ fuer Wiederholungszahl bei Audios, Videos, Bildern, Events -->
<xsd:simpleType name="repeatType">
    <xsd:restriction base="xsd:integer"/>
</xsd:simpleType>
<!-- Typ fuer Typus der Koordinatenangaben -->
<xsd:simpleType name="coordType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="relative"/>
        <xsd:pattern value="absolute"/>
    </xsd:restriction>
</xsd:simpleType>
<!-- Typ fuer Linktypus -->
<xsd:simpleType name="linkArt">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="behaviour"/>
        <xsd:pattern value="event"/>
    </xsd:restriction>
</xsd:simpleType>
<!-- Typ fuer Koordinatenangaben bei area-Tag -->
<xsd:simpleType name="coordsType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="\ ([0-9]+, [0-9]+, ([0-9]{1,2}%) | (100%)) "/>
        <xsd:pattern value="\ ([0-9]+, [0-9]+, [0-9]+, [0-9]+) "/>
    </xsd:restriction>
</xsd:simpleType>
<!-- Typ fuer Form bei area-Tag -->
<xsd:simpleType name="shapeType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="rect"/>
    </xsd:restriction>
</xsd:simpleType>
<!-- Typ fuer Zeitangaben -->
<xsd:simpleType name="timeType">
    <xsd:restriction base="xsd:time"/>
</xsd:simpleType>
<!-- Typ fuer Links -->
<xsd:simpleType name="hrefType">
    <xsd:restriction base="xsd:anyURI"/>
</xsd:simpleType>
<!-- Typ fuer Ausrichtung bei Regions -->
<xsd:simpleType name="fitType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="hidden"/>
        <xsd:pattern value="scroll"/>
    </xsd:restriction>

```

```

        <xsd:pattern value="fill"/>
        <xsd:pattern value="meet"/>
        <xsd:pattern value="slice"/>
    </xsd:restriction>
</xsd:simpleType>
<!-- Typ fuer Farbangaben -->
<xsd:simpleType name="colorType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="#[0-9a-fA-F]{6}"/>
    </xsd:restriction>
</xsd:simpleType>
<!-- Typ fuer ModulNamen bei Events -->
<xsd:simpleType name="moduleType">
    <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
<!-- Typ fuer AktionsNamen bei Events -->
<xsd:simpleType name="eventtypeType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="on_click"/>
        <xsd:pattern value="off_click"/>
        <xsd:pattern value="on_focus"/>
        <xsd:pattern value="off_focus"/>
        <xsd:pattern value="on_look"/>
        <xsd:pattern value="off_look"/>
        <xsd:pattern value="on_nothing"/>
    </xsd:restriction>
</xsd:simpleType>
<!-- Typ fuer Actiontypen bei Verhaltensweisen -->
<xsd:simpleType name="behaviouractionType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="play|stop"/>
        <xsd:pattern value="cycle"/>
        <xsd:pattern value="talk"/>
        <xsd:pattern value="up|down"/>
    </xsd:restriction>
</xsd:simpleType>
<!-- Typ fuer SubActiontypen bei Verhaltensweisen -->
<xsd:simpleType name="behavioursubactionType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[\-]{0,1}[0,1,2]{1}"/>
        <xsd:pattern value="forward|backward"/>
    </xsd:restriction>
</xsd:simpleType>
<!-- Typ fuer Inputtypen bei Equipment -->
<xsd:simpleType name="inputType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="text"/>
        <xsd:pattern value="button"/>
        <xsd:pattern value="radio"/>
    </xsd:restriction>
</xsd:simpleType>
<!-- Typ fuer Sources -->
<xsd:simpleType name="yesnoType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="yes"/>

```

```

        <xsd:pattern value="no"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- /Position -->
<!-- /Definition einfacher Typen -->

</xsd:schema>

```

nrml_complexTypes_narration.xsd

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <!-- Zentrale Definition einer Gruppe für die Medienobjekte -->
    <xsd:group name="g_content">
        <xsd:sequence>
            <xsd:element name="img" type="imgType" minOccurs="0"/>
            <xsd:element name="video" type="videoType" minOccurs="0"/>
            <xsd:element name="audio" type="audioType" minOccurs="0"/>
            <xsd:element name="text" type="textType" minOccurs="0"/>
            <xsd:element name="object" type="objectType" minOccurs="0"/>
        </xsd:sequence>
    </xsd:group>

    <!-- Zentrale Definition einer Gruppe für die Medienobjekte -->
    <xsd:group name="g_linkable_content">
        <xsd:sequence>
            <xsd:element name="img" type="imgType" minOccurs="0"/>
            <xsd:element name="video" type="videoType" minOccurs="0"/>
            <xsd:element name="text" type="textType" minOccurs="0"/>
            <xsd:element name="object" type="objectType" minOccurs="0"/>
        </xsd:sequence>
    </xsd:group>

    <!-- Zentrale Definition einer Gruppe für die Medienobjekte -->
    <xsd:group name="g_linkable_area_content">
        <xsd:sequence>
            <xsd:element name="img" type="imgType" minOccurs="0"/>
            <xsd:element name="video" type="videoType" minOccurs="0"/>
            <xsd:element name="object" type="objectType" minOccurs="0"/>
        </xsd:sequence>
    </xsd:group>

    <!-- Definition einer Gruppe für die Editierungsmöglichkeiten -->
    <xsd:group name="g_edit_content">
        <xsd:sequence>
            <xsd:element name="add" minOccurs="0">
                <!--
                <add>
                    Hinzufuegen von Inhalt zu einer Region oder hinter ein Element
                -->
            <xsd:complexType>
                <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                    <xsd:group ref="g_content"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:sequence>
    </xsd:group>

```

```

        <xsd:attribute name="id" type="idType" use="required"/>
        <xsd:attribute name="src" type="srcType" use="required"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="del" minOccurs="0">
    <!--
        <del>
            Loeschen von Inhalt in einer Region oder eines genauen Elements
        -->
    <xsd:complexType>
        <xsd:attribute name="id" type="idType" use="required"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="replace" minOccurs="0">
    <!--
        <replace>
            Ersetzen von Inhalt in einer Region oder eines genauen Elements
        -->
    <xsd:complexType>
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
            <xsd:group ref="g_content"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="idType" use="required"/>
        <xsd:attribute name="src" type="srcType" use="required"/>
    </xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:group>

<!--
    <on>
        Modifizierung des Inhalts bei Knoten X schon besucht oder nicht besucht
    -->
<xsd:complexType name="onType">
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="on" type="onType"/>
        <xsd:group ref="g_edit_content"/>
    </xsd:sequence>
    <xsd:attribute name="visited" type="idType"/>
    <xsd:attribute name="unvisited" type="idType"/>
    <xsd:attribute name="predecessor" type="idType"/>
</xsd:complexType>

<!--
    <narration>
        Spezifizierung der Narrationsstruktur
    -->
<xsd:complexType name="narrationType">
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <!--
            <chapter>
                Zur detaillierten Spezifizierung der Narrationsstruktur
                Dient der Gruppierung der einzelnen Knoten
            -->
            <xsd:element name="chapter" minOccurs="0">

```

```

<xsd:complexType>
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="node" minOccurs="0" maxOccurs="unbounded">
      <!--
        <node>
          Spezifizierung des Inhalts eines Narrationsknotens
        -->
      <xsd:complexType>
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
          <xsd:group ref="g_edit_content"/>
          <xsd:element name="on" type="onType"/>
          <xsd:element name="suggest" minOccurs="0">
            <!--
              <suggest>
                Biete Knoten zum Besuch an
              -->
            <xsd:complexType>
              <xsd:attribute name="id" type="idType" use="required"/>
              <xsd:attribute name="region" type="idType" use="required"/>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="id" type="idType" use="required"/>
        <xsd:attribute name="title" type="nameType" use="required"/>
        <xsd:attribute name="template" type="idType" use="required"/>
        <xsd:attribute name="keynode" type="yesnoType"/>
      </xsd:complexType>
      <xsd:key name="k_node">
        <xsd:selector xpath="node"/>
        <xsd:field xpath="@id"/>
      </xsd:key>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="id" type="idType" use="required"/>
  <xsd:attribute name="title" type="nameType" use="required"/>
</xsd:complexType>
<xsd:key name="k_chapter">
  <xsd:selector xpath="chapter"/>
  <xsd:field xpath="@id"/>
</xsd:key>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>

```

nrml_complexTypes_template.xsd

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- Definition einer Gruppe zur Koordinatenauswahl -->
  <!-- Definition der Koordinatenauswahl mittels Breite und Höhe -->
  <xsd:group name="coordDefinitionBH">

```

```

    <xsd:sequence>
      <xsd:element name="width" type="sizeType" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="height" type="sizeType" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:group>
<!-- Definition der Koordinatenauswahl mittels rechtem, unterem Eckpunkt -->
  <xsd:group name="coordDefinitionRUE">
    <xsd:sequence>
      <xsd:element name="bottom" type="positionType" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="right" type="positionType" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:group>
<xsd:group name="g_coord_choice">
  <xsd:choice>
    <xsd:group ref="coordDefinitionBH"/>
    <xsd:group ref="coordDefinitionRUE"/>
  </xsd:choice>
</xsd:group>

<!-- Definition einer Gruppe für die Links, Events und Behaviours -->
<xsd:group name="g_event_behaviour_link">
  <xsd:sequence>
    <xsd:element name="event" minOccurs="0">
      <!--
      <event>
        Spezifikation eines Events
      -->
    <xsd:complexType>
      <xsd:attribute type="eventtypeType" name="type" use="required"/>
      <xsd:attribute type="repeatType" name="repeat"/>
    </xsd:complexType>
  </xsd:element>
    <xsd:element name="behaviour" minOccurs="0">
      <!--
      <behaviour>
        Spezifikation von Verhaltensweisen
      -->
    <xsd:complexType>
      <xsd:attribute name="id" type="idType" use="required"/>
      <xsd:attribute name="action" type="behaviouractionType" use="required"/>
      <xsd:attribute name="subaction" type="behavioursubactionType"/>
    </xsd:complexType>
  </xsd:element>
    <xsd:element name="link" minOccurs="0">
      <!--
      <link>
        Ein Link auf ein Event oder ein Behaviour (damit mehrere Charaktere und
        Equipments dieselben Items nutzen können)
      -->
    <xsd:complexType>
      <xsd:attribute name="type" type="linkArt" use="required"/>
      <xsd:attribute name="id" type="idType" use="required"/>
    </xsd:complexType>
  </xsd:element>
  </xsd:sequence>

```

```

</xsd:group>

<!-- Definition einer Gruppe für Zeitsteuerung bei Texten und Bildern -->
<xsd:group name="g_media_begin_end">
  <xsd:sequence>
    <xsd:element name="begin" type="timeType" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="end" type="timeType" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="dur" type="timeType" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:group>

<!-- Definition einer Gruppe für Zeitsteuerung bei Videos und Audios -->
<xsd:group name="g_media_begin_clip_end">
  <xsd:sequence>
    <xsd:element name="begin" type="timeType" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="clipbegin" type="timeType" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="clipend" type="timeType" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="end" type="timeType" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="dur" type="timeType" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:group>

<!--
  <region>
    Zur detaillierten Spezifizierung des Layouts
    Dient als Huelle fuer die Medienobjekte
-->
<xsd:group name="g_region_content">
  <xsd:sequence>
    <xsd:group ref="g_content"/>
    <xsd:element name="character" type="characterType" minOccurs="0"/>
    <xsd:element name="equipment" type="equipmentType" minOccurs="0"/>
    <xsd:element name="a" type="aType" minOccurs="0"/>
    <xsd:element name="area" type="areaType" minOccurs="0"/>
    <xsd:element name="region" type="regionType" minOccurs="0"/>
  </xsd:sequence>
</xsd:group>
<xsd:complexType name="regionType">
  <xsd:sequence>
    <!-- Zuerst die simplen Kind-Elemente (zu region gehoerig)-->
    <xsd:group ref="g_coord_choice" minOccurs="1" maxOccurs="1"/>
    <xsd:group ref="g_region_content" minOccurs="1" maxOccurs="unbounded"/>
    <xsd:group ref="g_event_behaviour_link" minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="idType" use="required"/>
  <xsd:attribute name="x" type="positionType" use="required"/>
  <xsd:attribute name="y" type="positionType" use="required"/>
  <xsd:attribute name="z" type="positionType"/>
  <xsd:attribute name="coords" type="coordType"/>
  <xsd:attribute name="fit" type="fitType"/>
  <xsd:attribute name="color" type="colorType"/>
  <xsd:attribute name="sound" type="hrefType"/>
  <xsd:attribute name="volume" type="soundType"/>
</xsd:complexType>

```

```

<!--
  <template>
    Spezifizierung des Aussehens der Praesentation
-->
<xsd:complexType name="templateType">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="region" type="regionType" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="color" type="colorType"/>
  <xsd:attribute name="sound" type="hrefType"/>
  <xsd:attribute name="volume" type="soundType"/>
</xsd:complexType>

<!--
  <equipment>
    Spezifikation eines Equipments
-->
<xsd:complexType name="equipmentType">
  <xsd:sequence>
    <!--
      <input>
        Die Equipment-Möglichkeiten
      -->
      <xsd:element name="input">
        <xsd:complexType>
          <xsd:attribute name="type" type="inputType" use="required"/>
          <xsd:attribute name="id" type="idType" use="required"/>
          <xsd:attribute name="color" type="colorType"/>
          <xsd:attribute name="textcolor" type="colorType"/>
          <xsd:attribute name="src" type="srcType"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:group ref="g_event_behaviour_link" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="idType" use="required"/>
    <xsd:attribute name="type" type="typeType" use="required"/>
  </xsd:complexType>

<!--
  <character>
    Spezifikation eines Charakters
-->
<xsd:complexType name="characterType">
  <xsd:sequence>
    <!-- Zuerst die simplen Kind-Elemente (zu character gehoerig)-->

    <xsd:group ref="g_event_behaviour_link" minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="idType" use="required"/>
  <xsd:attribute name="type" type="typeType" use="required"/>
  <xsd:attribute name="src" type="srcType" use="required"/>

</xsd:complexType>

<xsd:complexType name="imgType">

```

```

<xsd:sequence>
  <!-- Die simplen Kind-Elemente (zu img gehoerig) -->
  <xsd:group ref="g_media_begin_end" minOccurs="0" maxOccurs="1"/>
  <xsd:group ref="g_event_behaviour_link" minOccurs="1" maxOccurs="unbounded"/

</xsd:sequence>
<xsd:attribute name="id" type="idType" use="required"/>
<xsd:attribute name="src" type="srcType"/>
<xsd:attribute name="type" type="typeType"/>
<xsd:attribute name="color" type="colorType"/>
</xsd:complexType>

<xsd:complexType name="videoType">
  <xsd:sequence>
    <!-- Die simplen Kind-Elemente (zu video gehoerig) -->
    <xsd:group ref="g_media_begin_clip_end" minOccurs="0" maxOccurs="1"/>
    <xsd:group ref="g_event_behaviour_link" minOccurs="1" maxOccurs="unbounded"/

  </xsd:sequence>
  <xsd:attribute name="id" type="idType" use="required"/>
  <xsd:attribute name="src" type="srcType" use="required"/>
  <xsd:attribute name="type" type="typeType"/>
  <xsd:attribute name="color" type="colorType"/>
  <xsd:attribute name="repeat" type="repeatType"/>
  <xsd:attribute name="volume" type="soundType"/>
</xsd:complexType>

<xsd:complexType name="audioType">
  <xsd:sequence>
    <!-- Die simplen Kind-Elemente (zu audio gehoerig) -->
    <xsd:group ref="g_media_begin_clip_end" minOccurs="0" maxOccurs="1"/>
    <xsd:group ref="g_event_behaviour_link" minOccurs="1" maxOccurs="unbounded"/

  </xsd:sequence>
  <xsd:attribute name="id" type="idType" use="required"/>
  <xsd:attribute name="src" type="srcType" use="required"/>
  <xsd:attribute name="type" type="typeType"/>
  <xsd:attribute name="repeat" type="repeatType"/>
  <xsd:attribute name="volume" type="soundType"/>
</xsd:complexType>

<xsd:complexType name="textType" mixed="true">
  <xsd:sequence>
    <!-- Die simplen Kind-Elemente (zu text gehoerig) -->
    <xsd:group ref="g_media_begin_end" minOccurs="0" maxOccurs="1"/>
    <xsd:group ref="g_event_behaviour_link" minOccurs="1" maxOccurs="unbounded"/

  </xsd:sequence>
  <xsd:attribute name="id" type="idType" use="required"/>
  <xsd:attribute name="src" type="srcType"/>
  <xsd:attribute name="type" type="typeType"/>
  <xsd:attribute name="color" type="colorType"/>
</xsd:complexType>

<xsd:complexType name="objectType">

```

```

<xsd:sequence>
  <!-- Die simplen Kind-Elemente (zu object gehoerig) -->
  <xsd:group ref="g_media_begin_end" minOccurs="0" maxOccurs="1"/>
  <xsd:group ref="g_event_behaviour_link" minOccurs="1" maxOccurs="unbounded"/

</xsd:sequence>
<xsd:attribute name="id" type="idType" use="required"/>
<xsd:attribute name="src" type="srcType"/>
<xsd:attribute name="type" type="typeType"/>
<xsd:attribute name="repeat" type="repeatType"/>
<xsd:attribute name="color" type="colorType"/>
</xsd:complexType>

<xsd:complexType name="aType">
  <xsd:sequence>
    <!-- Zuerst die simplen Kind-Elemente (zu a gehoerig) -->
    <xsd:group ref="g_media_begin_end" minOccurs="0" maxOccurs="1"/>
    <xsd:group ref="g_linkable_content" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="href" type="hrefType" use="required"/>
</xsd:complexType>

<xsd:complexType name="areaType">
  <xsd:sequence>
    <!-- Zuerst die simplen Kind-Elemente (zu area gehoerig) -->
    <xsd:group ref="g_media_begin_end" minOccurs="0" maxOccurs="1"/>
    <xsd:group ref="g_linkable_area_content" minOccurs="0" maxOccurs="unbounded"/

  </xsd:sequence>
  <xsd:attribute name="href" type="hrefType" use="required"/>
  <xsd:attribute name="shape" type="shapeType" use="required"/>
  <xsd:attribute name="coords" type="coordsType" use="required"/>
</xsd:complexType>

</xsd:schema>

```

C. Java-Code-Snippets für den Parser

In diesem Abschnitt sind die Code-Snippets zu finden, um einen rudimentären Parser zum Laufen zu bringen. Dieser validiert eine gegebene XML-Datei gegen die obige Grammatik im XSD-Format und verwendet die Xerces2-Bibliotheken von Apache.

XsdParserErrorHandler.java

```
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

/**
 * @author Markus Poestinger
 */
public class XsdParserErrorHandler implements ErrorHandler {

    public void warning(SAXParseException anException) throws SAXException {
        System.out.println("[warning] " + anException);
    }

    public void error(SAXParseException anException) throws SAXException {
        System.out.println("[error] " + anException);
    }

    public void fatalError(SAXParseException anException) throws SAXException {
        System.out.println("[fatal error] " + anException);
    }

}
```

XsdParser.java

```
import java.io.IOException;
import org.apache.xerces.parsers.DOMParser;
import org.w3c.dom.Document;
import org.xml.sax.SAXException;
import org.xml.sax.SAXNotRecognizedException;
import org.xml.sax.SAXNotSupportedException;

/**
 * @author Markus Poestinger
 */

class XsdParser {

    public static void main(String args[]) {

        XsdParserErrorHandler myErrorHandler = new XsdParserErrorHandler();
        DOMParser myParser = new DOMParser();
        Document myDocument = null;

        // Parser in validierenden Modus schalten und Angabe der XSD
        try {
```

```

// Dokument validieren und Fehler melden
myParser.setFeature("http://xml.org/sax/features/validation", true);
// Schema validieren
myParser.setFeature("http://apache.org/xml/features/validation/schema", true);

// Start-Element und Pfad zum Schema angeben (hier im selben Verzeichnis)
myParser.setProperty("http://apache.org/xml/properties/schema/external-schemaLocation",
"nrml nrml.xsd");
myParser.setErrorHandler(myErrorHandler);
} catch (SAXNotRecognizedException e) {
    System.out.print("Property unbekannt: ");
} catch (SAXNotSupportedException e) {
    System.out.print("Property wird nicht unterstuetzt: ");
}

// XML-Datei parsen
try {
    myParser.parse("nrml-template-example.xml");
    myDocument = myParser.getDocument();
} catch (IOException ie) {
    System.out.println("Kann die Datei nicht lesen.");
} catch (SAXException e) {
    System.out.print("Kann Dokument nicht erstellen: ");
    System.out.println(e.getMessage());
}

}
}

```

Erklärung

Ich versichere hiermit, daß ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Lübeck, den _____

Unterschrift: _____