Technische Universität Hamburg-Harburg
Arbeitsbereich Softwaresysteme

Diplomarbeit

Semantische Filterung von Webseiteninhalten für Mobiltelefone

Betreuer

Prof. Dr. Ralf Möller

Betrieblicher Betreuer

Kamil Sokolski

Vorgelegt von

Ke Huang

2009

Erklärung:

Ich erkläre hiermit dass ich die vorliegende Arbeit selbständig angefertigt habe und die aus benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Ke Huang
Hamburg, am 09. September 2009

Vorwort:

Diese vorliegende Diplomarbeit bildet den Abschluss meines Studiums im Diplomstudiengang Informatik-Ingenieurwesen an der Technische Universität Hamburg-Harburg (TUHH).

Besonderen Dank gilt seitens der Hochschule Herrn Prof. Dr. Ralf Möller, der mir auch vor der Diplomarbeit am Institut immer eine nette und freundliche Atmosphäre geboten hat.

Ebenfalls bedanken möchte ich mich bei Herrn Kamil Sokolski für seine Geduld, Korrekturen und Formulierungsvorschläge.

Ke Huang

Inhaltverzeichnis

1 Einleitung

1.1 Motivation

Welches Produkt ist derzeit das populärste Elektrogerät in der Welt? Das Ergebnis ist Mobiltelefon. Bis zum Dezember 2008 gab es weltweit insgesamt zirka 4 Milliarden Mobiltelefonbenutzer. Diese Zahl ist größer als die Summe von PC, TV, und Kreditkarten. Das Mobiltelefon wird wegen seiner exzellenten Beweglichkeit und Tragbarkeit derzeit nicht nur als Kommunikationsmittel in unserem gewöhnlichen Leben verwendet, sondern ist bereits ein populäres Multimediagerät geworden. Eine wichtige Funktion davon ist das Besuchen von Webseiten.

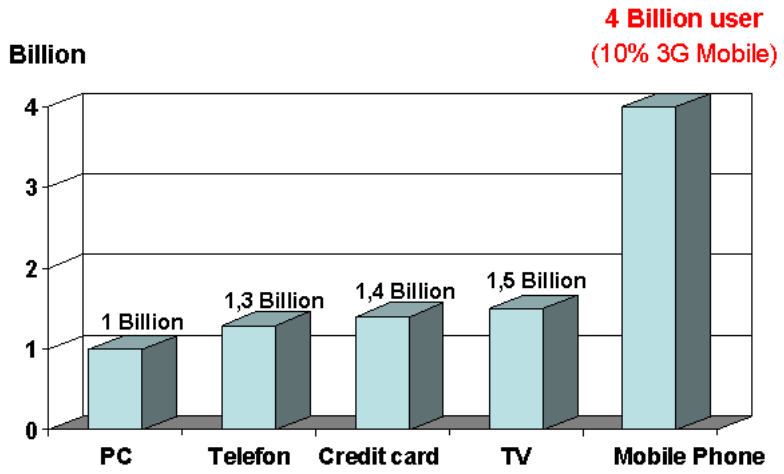

Abbildung-1.1
Die fünf populärsten Elektrogeräte in 2008[1]

Obwohl 3G-Mobiltelefon, welche das 3G-Mobilfunknetz mit schneller Datenübertragungsgeschwindigkeit von bis 2Mbps nutzen, durch das HTTP-kompatible WAP2.0-Protokoll die Webseiten direkt anzeigen können, sind die Darstellungsmöglichkeiten wegen kleiner Displays und schwacher Softwareunterstützung für Inhaltsreichen Webseiten und komplexere Programmierung noch nicht ausreichend. Das Layout und die Multimediadaten von den meisten Webseiten müssen zur Anpassung des Displays des Mobiltelefons erneut angeordnet und komprimiert werden. Diese Anpassung wird häufig nicht durch eine Reduzierung der Webseiteninhalte, sonder nur durch eine Änderung der Webseitenstruktur vorgenommen.

[1] **Quelle:** Informationen von Messeberichten der International Consumer Electronics Show (CES) 2008

Außerdem kann man auch zwei Versionen von Webseiten separat für Internetbenutzer und Mobiltelefonbenutzer vorhalten, wie die mobile Version von der SPIEGEL-Webseiten (m.spiegel.de), oder die graphisch reduzierte Version der IAAF-Seiten (www.iaaf.org) und BBC-Sport-Seiten(www.bbc.co.uk). Diese Webseiten sind aber nur ein geringer Teil im Internet. Es ist auch sehr aufwendig für alle Webseiten eine mobile Version aufzubauen.

Eine interessante Fragestellung ist, ob die großen Webseiteninhalte mit Hilfe einer Filterung, die auf der Bedeutung der Inhalte basiert, gemäß der gegebenen semantischen Domäne reduziert werden können. damit nur die für den Nutzer interessanten Inhalte in dem Mobiltelefon gezeigt werden. Anschließend ist zu prüfen, ob die Nutzung eines semantischen Filters für die Generierung einer kompakten Darstellung der Inhalte von Vorteil ist.

1.2 Aufgabenstellung

Ziel dieser Diplomarbeit ist es, ein Filter aufzubauen, welcher gemäß den gegebenen Bedingungen der Filterregeln die Webseiteninhalte auf der semantischen Ebene filtert und dann die gefilterten Inhalte in das Mobilseiten-Format übersetzt. Die Filterregeln sollen dazu extern definiert und änderbar sein. Anschließend ist zu prüfen, ob die Nutzung eines semantischen Filters für die Generierung einer kompakten Darstellung der Inhalte von Vorteil ist.

1.3 Aufbau der Arbeit

Das Kapitel 2 führt die notwendigen Grundlagen ein, die für diese Arbeit relevant sind. Hierzu werden der aktuelle Stand der Technik für mobiles Internetsurfen und semantische Beschreibung von Information erläutert. Außerdem werden auch die notwendigen Grundkenntnissen über BOEMIE und RacerPro in diesem Kapitel vorgestellt. Im Kapitel 3 wird die Entwurfsidee und Funktionsweise des semantischen Filters eingehend erläutert. Das Kapitel 4 beschäftigt sich mit dem praktischen Teil, um einen Prototypen des semantischen Filters mit Java zu realisieren. Im Kapitel 5 wird der aufgebaute Prototyp auf einige Webseiten angewendet, um die Funktionalität und Durchführbarkeit zu bewerten. Im letzten Kapital 6 wird die gesamte Arbeit zusammengefasst und die Ergebnisse der Arbeit werden bewertet.

2 Grundlagen

Vor Beginn der Entwicklung des in 1.2 beschriebenen semantischen Filters ist es notwendig die grundlegenden Begriffe und Kenntnisse, die in der Entwicklung des semantischen Filters genutzt werden, zu erläutern. In diesem Kapitel werden die Grundkenntnisse über Internetsurfen via Mobiltelefon und die zentrale Begriffe der Beschreibungslogik sowie auch das BOEMIE-Projekt und RacerPro vorgestellt.

2.1 Internetsurfen via Mobiltelefon

Wir können heute durch PC und World-Wide-Web Webseiten besuchen, um Informationen zu bekommen. Aber im Gegensatz zum PC kann das Mobiltelefon nicht direkt die Webseiten im Internet besuchen, da PC und Mobiltelefon unterschiedliche Netzwerkprotokolle zum Datenaustausch und unterschiedliche Auszeichnungssprachen zur Anzeige der Webseiteninhalte benutzen.

2.1.1 WAP1.0

Vor 2002 benutzten Mobiltelefone das WAP1.0-Protokoll (Wireless Application Protocol 1.0) um sich mit dem Internet zu verbinden. Das WAP1.0-Protokoll unterstützt nicht die TCP/IP- und HTTP-Protokolle, sondern benutzt neue Protokolle zum Datenaustausch und zum Parsen der Webseiten. Mit WAP1.0 kann ein Mobiltelefon nur mobile Webseiten besuchen, welche nicht mit HTML, sondern mit einer anderer Strukturierungssprache, WML (Wireless Markup Language) genannt, beschrieben wurden.

Die Übersetzung zwischen WAP1.0 und TCP/IP bzw. zwischen HTML und WML ist durch so genannte „WAP-Gateways“ implementiert. Das WAP-Gateway von WAP1.0 ist häufig von der technischen Seite als ein Proxy-Webserver implementiert, welcher nicht nur das Protokoll auf der Kommunikationsebene, sondern auch die Protokolle auf den anderen OSI-Ebenen übersetzt (Abbildung-2.1). Das WAP-Gateway komprimiert die WML-Seiten in einem binären Format, da der WML-Browser des Mobiltelefons nur binäres Format lesen kann.

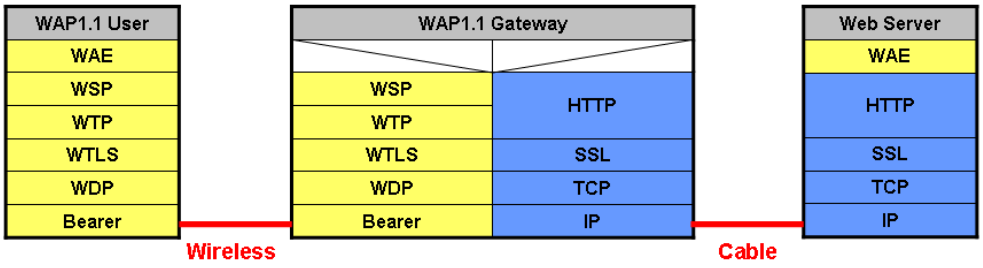

Abbildung-2.1
OSI-Ebenen für WAP1.1

Die Funktionsweise des mobilen Internetsurfens wird in Abbildung-2.2 illustriert. Das Mobiltelefon verbindet sich zuerst mit dem lokalen Internetanbieter (ISP) durch das GSM/GPRS-Funknetz und sendet eine WAP-Anfrage, um die Information aus dem Internet bekommen.

Der ISP verbindet dann mit einem WAP-Gateway und leitet die WAP-Anfrage weiter. Das WAP-Gateway übersetzt die WAP-Anfrage in eine konventionelle HTTP-Anfrage und schickt diese zu dem betreffenden Webserver. Die Antwort von dem Webserver wird wieder zurück zu dem WAP-Gateway geschickt und vom WAP-Gateway übersetzt, in ein binäres Format komprimiert und dann durch das GSM/GPRS-Netzwerk zurückgeschickt.

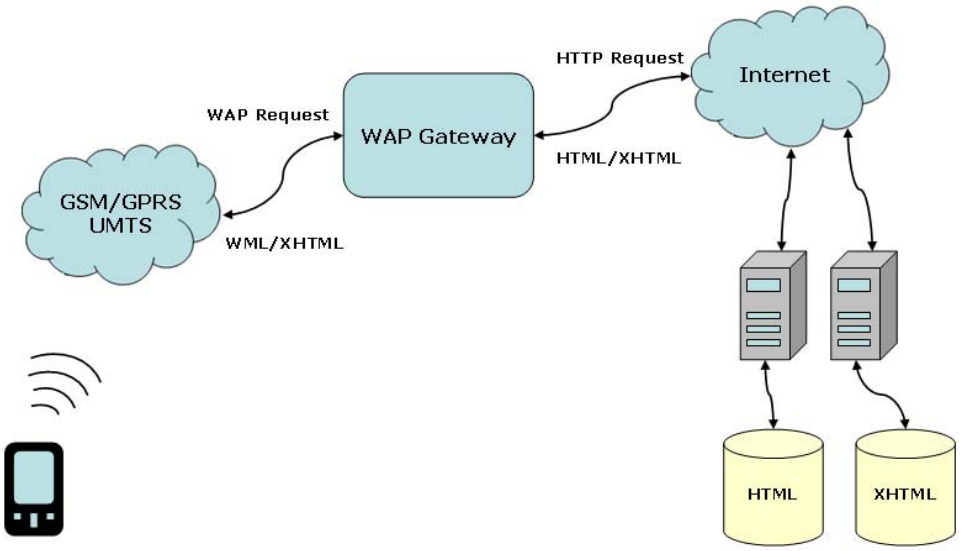

Abbildung-2.2
Funktionsweise des Internetsurfens via Mobiltelefon

2.1.2 WAP2.0

Das neue Protokoll WAP2.0 ist im Jahr 2002 veröffentlicht worden. Es verzichtet auf alle Mobilfunk-Spezifikationen und ersetzt die ursprünglichen WAP-Protokolle WSP, WTP und WTLS durch HTTP und SSL. Das WAP2.0 unterstützt sowohl WML als auch XHTML MP

(XHTML Mobile Profile), welches als eine vereinfachte XHTML Version für mobile Geräte verstanden werden kann. Die XHTML-Seite kann sowohl von dem normalen PC-Browser als auch direkt von dem Browser des Mobiltelefons gelesen werden. Außerdem wurde auch eine Stylesheet-Sprache WAPCSS, eine Untermenge von CSS, in WAP2.0 bereitgestellt. Durch diese Plattformunabhängigkeit von XHTML und die Kompatibilität von WAP2.0 werden drahtlose Funknetzwerk und das Internet lückenlos verbunden. WAP2.0 ist rückwärtskompatibel zu WAP1.x, d.h., die WML-Seite kann auch von dem Mikrobrowser des WAP2.0-Mobiltelefons gelesen werden.

Mit WAP 2.0 wurde auch die Funktion von WAP-Gateways aufgeweicht. Wie Abbildung-2.3 zeigt, wird das WAP-Gateway nicht wie in WAP1.0 zur Übersetzung der Protokolle benutzt und die Webseiten vom Internet müssen auch nicht bei dem WAP-Gateway in die banale WML-Seite umgeformt werden. Das Mobiltelefon kann sich direkt mit dem Internet verbinden und durch HTTP-Protokoll das XHTML-Seite besuchen.

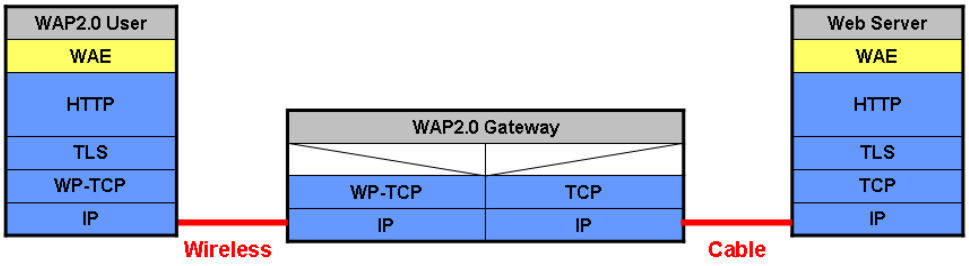

Abbildung-2.3
OSI-Ebenen für WAP2.0

Obwohl Mobiltelefone durch WAP2.0 die Webseiten von Internet direkt besuchen können, können Layouts der Webseiten häufig nicht wie die originalen Webseiten in Mikrobrowser des Mobiltelefons eingehalten werden müssen vor der Darstellung durch ein sogenannt „Layout-Engine“ erneut angeordnet, um die Webseiten zu den Displays des Mobiltelefons anzupassen.

2.1.3 WML

WML ist eine XML-basierte Auszeichnungssprache und wurde als ein Teil von WAE (Wireless Application Environment) auf der Applikationsebene von WAP1.0 vorgeschrieben, um mobile Webseiten aufzubauen, welche durch den WAP-Mikrobrowser dargestellt werden können.

Der Inhalt einer WML-Seite wird häufig nicht in einer Seite in Mobiltelefon dargestellt, sondern in vielen kleinen Seiten, so genannte „Karte“ verteilt, welche durch Links

aufeinander verbunden werden. Das Display des Mobiltelefons zeigt jedes Mal nur eine Karte, die übrigen Karten können durch Tastatur des Mobiltelefons überblättert werden.

Im folgenden Code-2.1 wird ein Beispiel für eine WML-Seite dargestellt. Durch die ersten Zeile des Codes ist es offensichtlich, dass die WML-Seite ein gültiges XML-Dokument ist und daher eine XML-Deklaration und die Angabe über den Dokumententyp beinhalten muss. Außer der XML-Deklaration wird der Hauptteil der WML-Datei mit dem WML-Tags umgeschlossen. Innerhalb des WML-Tags werden noch zwei Karten mit dem card-Tag strukturiert. Jede Karte hat eine ID, einen Titel und verweist durch anchor-Tags als Link auf andere Karten.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
 "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="no1" title="card 1">
<p>card1 content</p>
<p>
<anchor>Next page<go href="#no2"/></anchor>
</p>
</card>
<card id="no2" title="card 2">
<p>card2 content</p>
<p>
<anchor>Back<go href="#no1"/></anchor>
<anchor>Next Page<go href="#no3"/></anchor>
</p>
</card>
</wml>
```

Code-2.1: Beispiel für eine einfache WML-Seite mit zwei Karten.

Wie in HTML gilt es auch in WML die Tags für Textformatierung, die in folgender Tabelle zusammengefasst werden.

Tag	**Styling**
<b>	Dickschrift
<big>	Große Schriftart
	Ausgeprägte Schriftart
<i>	Kursivschrift
<small>	Kleine Schriftart

	Gewichtige Schriftart
<u>	Unterschtreichen

Tabelle-2.1: Angabe für WML-Tags

Außerdem unterstützen WML wie HTML auch das img-Tag zum Anzeigen von Bildern und auch neue Tags, wie zum Beispiel das select-Tag zur Erstellung der Auswahllisten und das templet-Tag zur Ereignisbehandlung. Diese Teile und Details über WML-Script werden wegen Nichtbenutzung in der Arbeit in diesem Abschnitt nicht weiter beschrieben.

2.1.4 XHTML MP

XHTML ist einen Standarddokumenttyp von World-Wide-Web-Consortium(W3C), welche aufgrund HTML4 mit XML erneut formuliert wird. Die Tags von XHTML sind ganz identisch von HTML aber grammatisch mehr strikter als HTML. Diese grammatisch saubere Eigenschaft führt zu einer leichteren und korrekteren Analyse und Extraktion der Inhalte von Webdokument. Dieser Vorteil ist für die mobilen Funkgeräte, deren Funktionsfähigkeit zum Parsen der Webseiten sehr eingeschränkt ist, sehr sinnvoll. Zurzeit unterstützen alle Webbrowser XHTML. XHTML wird allmählich HTML als die wichtigste Auszeichnungssprache für Webseiten ersetzen.

XHTML MP (XHTML Mobile Profile) ist eine vereinfachte XHTML Version speziell für mobile Geräte und wird auch in WAP2.0 als eine neue Auszeichnungssprache vorgeschrieben. Die Webseiten, die mit XHTML MP geschrieben werden, können nicht nur von üblichen Webbrowsern, sondern auch von Mikrobrowsern des Mobiltelefons gelesen werden. Der Webseiteentwickler braucht damit nicht mehr mit zwei unterschiedlichen Strukturierungssprachen zwei unterschiedliche Versionen von einer Webseite zu entwickeln.

Im folgenden Code-2.2 wird ein Beispiel für eine XHTML-MP-Seite dargestellt. Durch die erste Zeile des Codes ist es offensichtlich, dass die XHTML-MP-Seite ein gültiges XML-Dokument ist und daher eine XML-Deklaration und die Angabe über den Dokumententyp beinhalten muss. Unter dem Kopfteil des Codes ist der Hauptteil grammatisch identisch mit HTML und damit auch durch das html-Tag umgeschlossen.

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
   "http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
<title>XHTML MP page title</title>
</head>
<body>
<p>XHTML MP page title content.</p>
</body>
</html>
```

Code-2.2: Beispiel für eine XHTML-MP-Seite

Außer der grammatischen Regeln von HTML muss ein XHTML-MP-Dokument auch die folgenden Regeln einhalten.

- Jedes Tag muss mit entsprechendem End-Tag richtig geschlossen werden.
- Tags und Attributen müssen klein geschrieben werden.
- Attributswerte müssen in Anführungszeichen gesetzt werden.
- Attribute, welche booleschen Wert bewertet sind, müssen den Attributname und Attributwert beide vollständig geschrieben werden.

2.2 Beschreibungslogik

Wie schon im Unterkapitel 1.2 erwähnt, wurde der Inhalt jeder in der Arbeit zu filternden Webseite mit einem formalen und maschinen-lesbaren geschriebenen Dokument beschrieben. Dieses formal geschriebene Dokument nennen wir „ABox“ und die formale Beschreibungsmethode nennen wir Beschreibungslogik.

Beschreibungslogiken sind eine Familien von logistischen Sprachen zur Wissensrepräsentation, welche die Welt in abstrakter Weise formal beschreiben können. Beschreibungslogiken sind die Untermenge der Logiken der ersten Stufe, deren Eigenschaft die Entscheidbarkeit ist. D.h., dass jede Aussage der Beschreibungslogik in begrenzter Zeit entschieden werden kann. Es gibt folgende drei Grundelemente:

- **Konzepte** repräsentieren die Begriffe von Objekten und sind vergleichbar mit Klassen in der objektorientierten Programmierung, wie z.B. Person, Mann, Frau, Auto, usw.
- **Rollen** beschreiben die binären Relationen zwischen Objekten, wie z.B. hat_Kind, hat_Eltern, verheiratet, usw.
- **Individuen** sind die konkreten Entitäten und Instanzen von Konzepten, z.B. sind Sabine und Betina zwei Instanzen von dem Konzept Frau.

Man unterteilt die Wissensbasis einer Beschreibungslogik formal in eine Terminological Box (Tbox) und eine Assertional Box (Abox).

2.2.1 TBox

Fast alle in der realen Welt existierten Objekte können durch einem Name und Begriff definiert werden, wie z.B. Frau, Auto. Diese Begriffe werden in der Beschreibungslogik „Konzept“ genannt und in dem so genannt „Terminologischen Box“ (TBox) formal definiert. Eine TBox enthält hierbei das Wissen um die Konzepte einer Domäne, das terminologische Wissen.

Die grundlegenden Konzepte werden in der TBox mit atomaren Konzepte (unäre Prädikate) und atomaren Rollen (binäre Prädikate) definiert, z.B.

$Frau \equiv Person \sqcap Weiblich$

Durch die atomaren Konzepte Person und Weiblich mit dem Vereinigungssymbol wird die Aussage für die Definition von Frau dargestellt und es bedeutet, dass eine Frau eine weibliche Person ist. Mit diesem neuen Konzept können viele andere Konzepte weiter definiert werden, z.B. wird mit dem Konzept von Frau und der Rolle *hat_Kinder* wird das neue Konzept von Mutter wie unten definiert.

$Mutter \equiv Frau \sqcap \exists_{>}, hat\ \ Kind$

Diese Aussage stellt nicht nur die Definition von Mutter dar, sondern bildet auch eine implizite Beziehung zwischen die Konzepte von Frau und Mutter ab. Falls eine Frau die Beziehung *has_Kind* mit einem oder mehreren Kinder hat, gehört diese Frau zu dem Konzept Mutter. Das gleiche Konzept kann durch unterschiedliche Darstellung definiert werden. z.B,

$Ehefrau \equiv Frau \sqcap \exists_{=}, hat\_Ehepartner$

$Ehefrau \equiv Frau \sqcap verheiratet$

Die TBox für die Familie von den oben gegebenen Beispielen kann so dargestellt werden,

$$
\begin{aligned}
TBox: \{ Frau &\equiv Person \sqcap Weiblich \\
Mann &\equiv Person \sqcap Männlich \\
Kind &\equiv Person \sqcap \exists_{\geq 1}, hat\_Eltern \\
Vater &\equiv Mann \sqcap \exists_{\geq 1}, hat\_Kind \\
Mutter &\equiv Frau \sqcap \exists_{\geq 1}, hat\_Kind \\
Eltern &\equiv Person \sqcap \exists_{\geq 1}, hat\_Kind \\
Ehepartner &\equiv Person \sqcap Heiratet \}
\end{aligned}
$$

Für die Konzeptdefinitionen in einer TBox müssen folgende Punkte beachtet werden,

- Konzeptnamen in der TBox müssen eindeutig sein, ein Name darf nur für eine Definition benutzt werden.
- Konzepte dürfen sich nicht selbst referenzieren, müssen also azyklisch sein.

2.2.2 ABox

Mit den in einer TBox definierten Konzepten und Beziehungen können nun alle Individuen der realen Welt beschrieben werden. Diese Darstellung erfolgt in einer "Assertional Box" (ABox), welche über den Wissensbereich sogenanntes erweitertes Wissen enthalten. Es bezieht sich auf erweiterte Aussagen über konkrete Individuen des Konzeptes und deren Beziehungen untereinander. In folgendes Bespiel werden zwei Individuen *Betina* und *Sabina* vom Konzept Frau und die Rolle *hat_Kind* zwischen *Sabina* und *Betina* dargestellt.

Frau(Sabina), Frau(Betina)
hat_Kind(Sabina, Betina)

Die ABox für die Familie kann beispielweise so dargestellt werden.

ABox: {Frau(Sabina), Frau(Betina), Mann(Wolff)
hat_Ehepartner(Wolff,Sabina)
hat_Kind(Sabina, Betina)}

2.2.3 Ontologie

Die Ontologie ist ursprünglich eine Teildisziplin der Philosophie und versteht man in der Informatik eine explizite formale Spezifikation einer Konzeptualisierung. Ontologie bietet ein fachliches Vokabular, welches alle Konzepte und untereinander stehende Relationen von einer bestimmten Domäne des Wissens sammelt, um die verschiedenen konkreten Objekte von der Domäne zu modellieren. Eine Ontologie ist ähnlich einer Sammelung von den fachlichen Wörtern einer Domäne.

Eine Ontologie kann aus einer anderen Ontologie oder mehreren anderen Ontologien zusammen aufgebaut werden. Beispielweise kann die Oberontologie Sport aus den Ontologien von Athletik, Fußball, Handball, und andere Unterontologie aufgebaut werden, jede Unterontologie sammelt alle fachlichen Konzepte von einer Sport-Unterdomäne.

2.3 Semantische Beschreibung von Information

Mit Beschreibungslogik können die Bedeutung der Information in einem maschinen-lesbaren Format beschrieben werden. Damit wird die Zusammenarbeit zwischen Menschen und Computern viel erleichtert. Das ist auch die Grundidee des Semantic Web von Tim Berners-Lee. Um die Bedeutung (auch Semantik genannt) der Informationen zu beschreiben, werden im Semantic Web Metadaten im RDF oder OWL Format verwendet. Die Meta-Daten erlauben es Computer auf die Bedeutung eines Dokuments zuzugreifen. In dem Kapitel werden die Grundkenntnisse über semantische Metadaten sowie auch die Beschreibungssprache RDF und OWL vorgestellt.

2.3.1 Semantische Metadaten

Die Metadaten sind die Daten, welche als Basisinformationen zur Beschreibung von andren Daten (data about data) insbesondere für die Attribute anderer Daten verwendet werden.
Es gibt keine allgemeingültige Unterscheidung zwischen Metadaten und gewöhnlichen Daten. Je nach Zweck der Metadaten können verschiedene Aspekte der Daten beschrieben werden. Zum Beispiel, die Metadaten einer Nachricht auf der Webseite für den Internetbesucher sind vielleicht Überschrift, Autors und Ausgabezeit. Für den Webseitenentwickler sind die Metadaten vielleicht die Schriftart, die Schriftgroße und die Schriftfarbe. Die Metadaten werden beispielsweise für die folgenden Zwecke verwendet.

- Inhaltbeschreibung der Daten mit menschlicher Sprache, wie Zusammenfassung, Stichwörter usw.
- Attributsbeschreibung der Daten für die Datenverwaltung, wie Änderungsdatum, Datentyp usw.
- Technische Beschreibung, wie Auflösung, Kompressionsverfahren usw.
- Rechtliche Beschreibung, wie Urheberrecht, Nutzungsrichtlinien usw.

Die klassischen Metadaten werden häufig in Form von Schlüssel-Wert-Paare dargestellt. Diese Darstellung ist trotzdem leicht zu verstehen und macht keine großen Probleme bei der Indizierung und Suche der beschriebenen Informationen, ist aber für Beschreibung der Zusammenhänge zwischen Werten nicht geeignet. Es ist nicht möglich mit klassischen Metadaten die impliziten Zusammenhänge der Informationen darzustellen. Mit Hilfe von semantischen Metadaten können aber solche Zusammenhänge ausgedrückt werden.

Semantische Metadaten kombinieren die klassischen Metadaten und Ontologien mit Anwendung von konzeptuellen Modellen die semantischen Bedeutungen der Informationen zu beschreiben. Im semantischen Web werden die semantischen Metadaten mit einer formalen Beschreibungssprache in den maschinen-lesbare Formaten beschrieben, damit die beschriebenen Webinformationen vom Computer intelligent verwaltet und bearbeitet werden können.

2.3.2 Resource Description Framework-RDF

Um das semantische Ontologiemodell und Wissensrepräsentation bequemer und leichter darstellen zu können wurde von W3C zuerst eine Standardspezifikation „Resources Description Frame“ (RDF) erstellt, welche als eine XML-basierte Beschreibungssprache zur formalen Beschreibung von Webinformationen, die durch eindeutigen Bezeichner(URI) identifiziert werden, verwendet wird. Es ist ursprünglich als Datenmodelle zur Beschreibung der Metadaten im Web entwickelt worden. [WIKI-RDF]

Jeder RDF-Ausdruck ist eine Menge von RDF-Tripel, welche aus drei Grundelementen besteht.

- **Subjekt**, die mit einen URI identifiziert wird.
- **Prädikat**, welche Beziehungen zwischen Subjekte und Objekte definieren, sehr ähnlich wie Rolle von Beschreibungslogik Z.B. *hasName*, *hasDatum*, usw.
- **Objekt**, die Werte von Prädikate und wird auch häufig mit Attributwert des Subjekts genannt.

Ein RDF-Tripel ist eine Aussage, dass eine Beziehung zwischen dem Subjekt und dem Objekt besteht. Diese Beziehung ist gerichtet, nämlich von Subjekt zum Objekt, und mit dem Prädikat benannt. Folgendes Beispiel (Code-2.3) zeigt ein einfachstes RDF-Dokument.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.sample.com/cd/elements/">
<rdf:Description
 rdf:about="http://www.sample.com/cd/object-1/">
  <cd:hasName>Matthias Wolff</cd:hasName>
 <cd:hasGender>Male</cd:hasGender>
  <cd:hasBirthdate>10.10.1967</cd:hasBirthdate>
  <cd:hasNationality>German</cd:hasNationality >
```

```
</rdf:Description>
</rdf:RDF>
```

Code-2.3: Beispiel für eines RDF-Dokument

Die Deklaration von der ersten Reihe zeigt, dass das Dokument ein XML-Dokument ist. Nachfolgend wurden zwei Domänenamen „rdf“ und „cd“ definiert. Der Hauptteil dieses RDF-Dokument wird in dem Description-Element von RDF-Syntax geschlossen, welche eine Ressourcebeschreibung definiert. Die Ressource wird mit einem URI von „*http://www. sample.com/cd/object-1/"* eindeutig identifiziert und hat vier Prädikaten, welche die Eigenschaften von einen Person indizieren.

Durch RDF kann man einfach mit den Prädikaten und Objekte für die Webressourcen beschreiben, aber in der praktischen Anwendung bracht man häufig die neuen Vokabeln zur Beschreibung der bestimmten Domäne. Diese selbstdefinierten Vokabeln werden durch sogenanntes RDF-Schema (RDFS) definiert werden, welche eine semantische Erweiterung von RDF ist und auch als eine Standard von W3C erstellt. Folgendes Beispiel zeigt eine einfachste Anwendung von RDFS.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://www.animals.fake/animals#" >
<rdfs:Class rdf:ID="Person" />
<rdfs:Class rdf:ID="Male">
  <rdfs:subClassOf rdf:resource="#Person"/>
</rdfs:Class>
</rdf:RDF>
```

Code-2.4: Beispiel für eine RDFS-Dokument

Davon wird zwei Konzepte „Person“ und „Male“ deklariert und „Male“ ist ein Unterkonzept vom „Person“. Durch diese Konzeptdeklaration kann man direkt die Individuen vom Konzept Male definieren.

2.3.3 Beschreibungssprache-OWL

Obwohl die Webressourcen, Eigenschaften und dazwischen untereinander stehende Relationen durch RDFS repräsentiert werden können, kann RDFS nicht zur Beschreibung

der Konzepten, die aus unterschiedlichen semantischen Bedingungen beschränkt sind, verwenden. Folgenden Bedingungen können nicht mit RDFS dargestellt werden [GRO09].

- **Für Bereichsbeschränkungen:** z.B.: Fleischfresser und Pflanzenfresser.
- **Für Kardinalitätsbeschränkungen:** z.B.: Eine Person hat exakt zwei Eltern.
- **Für disjunktive Eigenschaft:** z.B.: Frau und Mann, Tier und Mensch
- **Für inverse Eigenschaft:** z.B. hat Kind und hat Eltern.
- **Für Transitive Eigenschaft:** z.B. mehr als, kleiner als.

Dazu erstellte W3C die neue Beschreibungssprache „Web Ontology Language“ (OWL) als neue Standard, welche technisch auf der RDF-Syntax basiert, um die Ontologiemodelle anhand einer formalen Beschreibungssprache ausführlich und vollständig darstellen zu können.

OWL besteht aus drei Untersprachen OWL Lite, OWL DL und OWL Full, welche mit unterschiedlichen Ausdrucksmächtigkeiten und Verarbeitbarkeit für unterschiedliche Anwendungen [GRO09].

- **OWL Lite** bietet nur eine Klassifikationshierarchie und mit einfachste Beschreikungen. Z.B. erlaubt es zwar Kardinalitätsbeschränkung, aber nur die Werte von 0 und 1.
- **OWL DL** bietet das maximale Ausfrücksmächtigkeit und auch die vollständige Verarbeitbarkeit. Es schließt alle OWL Sprachkonstrukte ein und kann aber nur unter bestimmten Bedingungen verwendet werden.
- **OWL Full** bietet auch die maximale Ausdrücksmächtigkeit aber nicht garantierte Verarbeitbarkeit. Es benutzt alle syntaktischen Strukturen von RDF und erlaubt das Vokabular von einer Ontologie selbstdefiniert zu erweitern.

Unteres Beispiel (Code-2.5) zeigte eine Anwendung von OWL, welche Drei Konzepte deklariert. Die disjunktive Eigenschaft zwischen Male und Female wird durch das OWL-Element *„disjoinWith“* definiert.

```
<owl:Class rdf:ID=„Person">
<rdfs:label>Person</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Male">
<rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
```

```
<owl:Class rdf:ID="Female">
<rdfs:subClassOf rdf:resource="#Person"/>
<owl:disjointWith rdf:resource="#Male"/>
</owl:Class>

<owl:Class rdf:ID=„Person">
<rdfs:label>Person</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Male">
<rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
<owl:Class rdf:ID="Female">
<rdfs:subClassOf rdf:resource="#Person"/>
<owl:disjointWith rdf:resource="#Male"/>
</owl:Class>
```

Code-2.5: Beispiel für disjunktive Eigenschaft in einem OWL-Dokument

Unteres Beispiel (Code-2.6) basiert auf oberes Beispiel weiter für drei Rollen (oder Prädikat genannte*)* *„hasParent“*, *„hasFather“ und „hasChild“* zu definieren. Die inverse Eigenschaft zwischen *hasFather* und *hasChild* wird durch das OWL-Element *„inverseOf“* definiert.

```
<owl:ObjectProperty rdf:ID="hasParent">
<rdfs:domain rdf:resource="#Person"/>
<rdfs:range rdf:resource="#Person"/>
</ owl:ObjectProperty>
< owl:ObjectProperty rdf:ID="hasFather">
<rdfs:subPropertyOf rdf:resource="#hasParent"/>
<rdfs:range rdf:resource="#Male"/>
</ owl:ObjectProperty>
< owl:ObjectProperty rdf:ID="hasChild">
< owl :inverseOf rdf:resource="#hasParent"/>
</ owl:ObjectProperty>
```

Code-2.6: Beispiel für inverse Eigenschaft in einem OWL-Dokument

Weiterhin kann die Personenzahl von den Eltern durch das OWL-Element „cardinality“ für die Kardinalitätsbeschränkungen definiert (Code-2.7).

```
<owl:Class rdf:about="#Person">
<rdfs:subClassOf>
<owl:Restriction owl:cardinality="2">
```

```
<owl:onProperty rdf:resource="#hasParent"/>
</ owl :Restriction>
</rdfs:subClassOf>
</ owl:Class>
```

Code-2.7: Beispiel für Kardinalitätsbeschränkung in einem OWL-Dokument

Unter Verwendung der OWL-Syntaxen und RDF-Syntaxen kann man alle semantischen Ontologiemodelle aufbauen, damit die Webressourcen durch semantische Relationen konkret und eindeutig Beschrieben werden können. Das Ziel von semantischer Beschreibung ist um die Ressourcen im zukünftigen semantischen Web von den Maschinen intelligent verwaltet und exakt gesucht werden zu können.

2.4 BOEMIE Projekt

BOEMIE ist eine Abkürzung von „Bootstrapping Ontology Evolution with Multimedia Information Extraction“. BOEMIE-Projekt wurde in März 2006 gestartet. Das Ziel von BOEMIE-Projekt ist um ein Ontologie-basiertes System durch allmähliche Evolution der Ontologie, so genannt Bootstrapping-Prozess, zur Unterstützung der semantischen Extraktion von Multimedia zu entwickeln.

Durch Einsatz des BOEMIE-Projekts können die stetig anwachsende Anzahl Multimediadaten im Web mit einer Bedeutung, die eine maschinellen Verarbeitung zugängig ist, versehen werden, damit wird der Zugriff auf diese Daten vereinfacht [BoePR06].

2.4.1 Bootstrapping-Prozess

BOEMIE bietet eine Methode, sogenannten „Bootstrapping-Prozess“, welche durch zwei Seiten, die sich untereinander unterstützt und hilft, implementiert. Bei einer Seite wird die Ontologie durch Ontologie-Entwickler angereichert und nennen wir diesen Prozess Ontologie-Evolution. Bei anderer Seite wird die angereicherte Ontologie wieder zur Extraktion der Information verwendet und nennen wir diesen Prozess semantische Extraktion [BoeAR08].

Aufgrund der bereits in Ontologie definierten Konzepte wird die Information von Multimediaressourcen auf unterschiedlichen Interpretationsebenen extrahiert. Es gibt drei Phasen in dem Prozess von semantischer Extraktion [SOKO08].

- **Analysis:** Zur Erwerb der Individuen/Relationen von den Mid-Level-Conzepts (MLCs), welche die Information von den Multimediaressourcen auf niedriger Ebene interpretiert.
- **Interpretation:** Zur Erwerb der Individuen/Relationen von den Hight-Level-Conzepts (HLCs), die durch Kombination der MLCs und Relationen von Analysis-Phase die weitere Information von den Multimediaressourcen auf hoher Ebene interpretiert werden.
- **Fusion:** Zur Interpretation der gemeinsamen Merkmale von den Individuen durch Vereinigung MLCs-Individuen und HLCs-Individuen.

Die Individuen, welche nicht in Interpretation-Phase durch MLCs und HLCs interpretiert werden, können nach Ontologie-Evolution wieder zur semantischen Extraktion bearbeitet werden. Ontologie-Evolution wird durch folgende zwei Phasen implementiert [MIDOE08].

- **Population:** Zur Einfügen der neuen Individuen in Ontologie.
- **Enrichment:** Zur Einfügen der neuen Konzepte und Relations in Ontologie

Der Zustandübergang für Bootstrapping-Prozess wird durch folgendes Zustanddiagramm in der Abbildung-2.4 dargestellt.

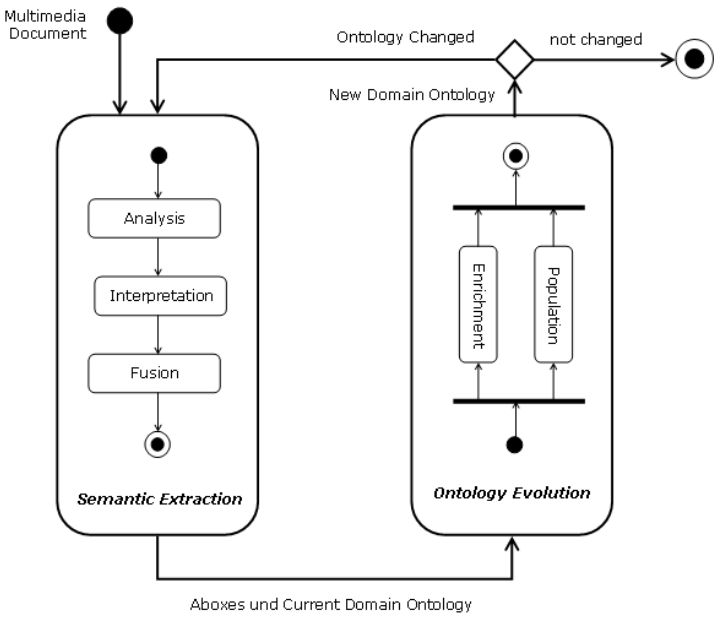

Abbildung-2.4
Zustandsdiagramm für Bootstrapping-Prozess [SOKO08]

2.4.2 BOEMIE Ontologie

Im BOEMIE-Projekt wird vier Ontologien zur Aufbau des semantischen Modells für Multimediaressourcen verwendet. Zwei sind Domain-spezifizierte Ontologien, sogenannt AEO und GIO, und zwei sind Media-spezifiziert Ontologien, sogenannt MCO und MDO. Die verschiedene Ontologien werden im semantischen Modell untereinander verbindet.

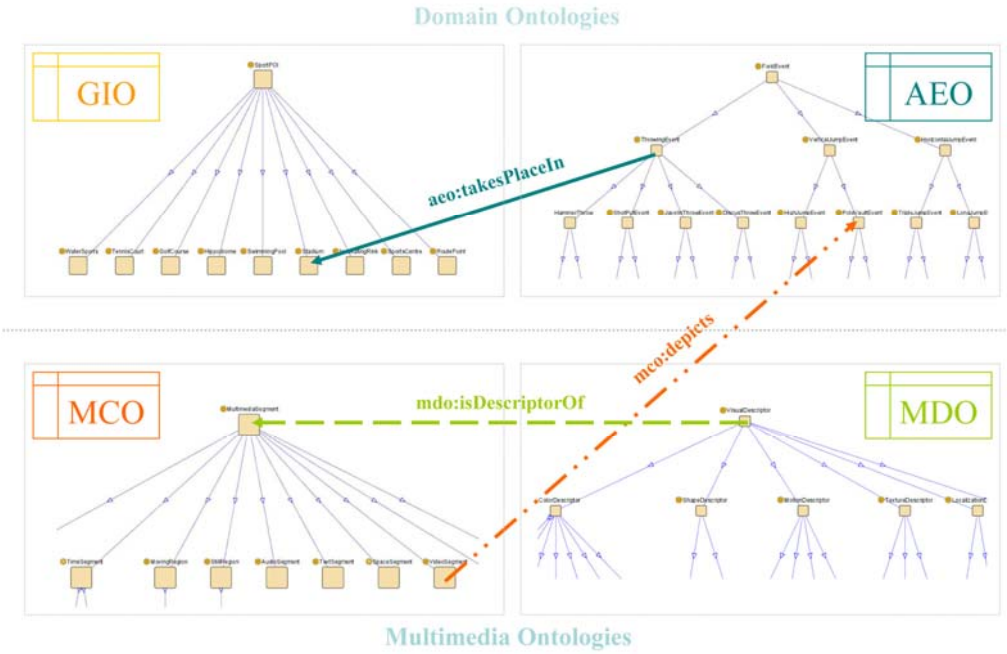

Abbildung-2.5
Das semantische Modell für Multimedia in BOEMIE-Projekt [ITI712]

Die Abbildung-2.5 zeigt das Semantische Modell für Multimedia in BOEMIE-Projekt, welche mit Ontologien AEO, GIO, MCO und MDO aufgebaut ist. Die Anwendungen der vier Ontologien werden im folgend zusammengefasst:

- **Athletics Event Ontology(AEO)** definiert alle Konzepte und Relationen über athletische Begriffe aufgrund der Konkurrenzregelung und technische Vorordnung von IAAF.

- **Geographic Information Ontology(GIO)** definiert alle Konzepte und Relationen über geographische Begriffe mit Erweiterung des Modellschemas von TeleAtlas, welches durch digitales Mappen die Dienste für Navigation und Lokalisierung bietet.

- **Multimedia Content Ontology(MCO)** definiert die Kenntnisse zur Beschreibung der Inhaltstrukturen und Links von den Multimediaressourcen wie Text, Bilder, Audio usw. [MCDO08].

- **Multimedia Descriptor Ontology(MDO)** bietet eine komplette Reihe von visuellen Deskriptoren und Audio-Deskriptoren an, zur Beschreibung der Inhalte der Multimediaressourcen auf der Feature-Ebene [MCDO08].

2.4.3 Semantische Extraktion von Webpages

In BOEMIE wurden zwei Schwerpunkte für semantische Extraktion untersucht. Der erste Schwerpunkt betrifft die Webseiten von IAAF, davon werden nur die Informationen auf den Webseiten, die Text und Bilder beinhaltet, zur Analysis, Interpretation und Fusion bearbeitet. Der zweite Schwerpunkt betrifft die Video-Sequenz über athletische Ereignisse, die Informationen aus den Videoshots werden zur Analysis, Interpretation und Fusion bearbeitet. In der Arbeit werden nur die Webseiten von IAAF vom BOEMIE-Objekt als die Eingangdaten für semantischen Filter verwendet. Dieser Teil wird in der Entwicklungsphase des semantischen Filters weiter vorgestellt.

Die semantische Extraktion wird in BOEMIE durch eine sogenannte Dekomposition-Fusion-Methode implementiert, die ähnlich wie Bootstrapping-Prozess aber aus zwei Schritten, Dekomposition und Fusion, zur semantischen Extraktion besteht [SEFMC09].

- **Dekomposition:** In dem ersten Schritt werden die Inhaltobjekte des Multimedia-Dokuments gemäß Typen in unterschiedlichen einzelnen Medienelemente zerlegt und dann durch angemessene Methode separat analysiert. Diese Dekomposition von Multimedia-Dokument bezieht sich nicht nur auf die einzelne Medienelemente sondern auch die entsprechende Relationen.
- **Fusion:** in dem zweiten Schritt werden alle einzelne Medienelemente des Multimedia-Dokuments nach einzelne Prozess zusammen mit die Information über die Struktur des Multimedia-Dokuments zur Interpretation der vollständigen Webseite vereinigt und dann mit OWL formal im Dokument geschrieben und im Ausgang ausgegeben.

Das Folgende Diagramm von der Abbildung-2.6 zeigt ein grobes Arbeitsprinzip von Dekomposition-Fusion-Methode. Die weiteren ausführlichen Informationen können in dem technischen Dokument[BoeSE09] von der BOEMIE-Homepage erhalten werden.

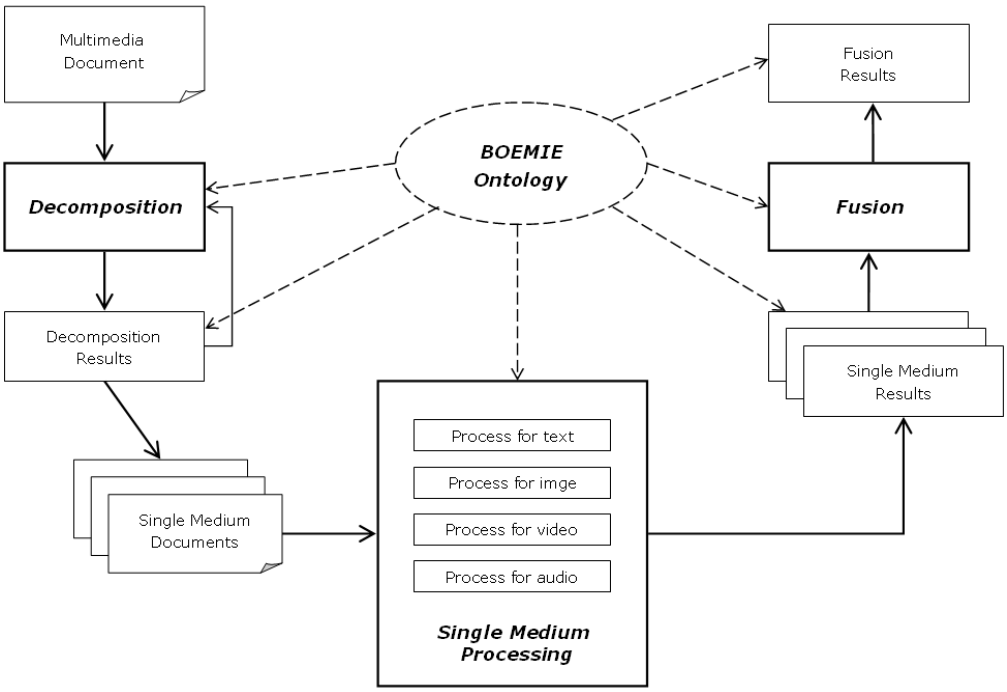

Abbildung-2.6
Das Arbeitsprinzip für Dekomposition-Fusion-Methode [BoeSE09]

2.5 Semantisches Information Retrieval

Am Ende des Abschnitts 2.3.3 wurde bereits eine Zusammenfassung über das Ziel für semantische Beschreibung der Webressourcen erwähnt, dass die Webressourcen im zukünftigen semantischen Web exakt gesucht werden können. Diese Informationssuche, die durch die Analysis der semantischen Information von den Webressourcen implementiert wird, wird „semantisches Information Retrieval“ genannt.

2.5.1 RacerPro

RacerPro(Renamed Abox and Concept Expression Reasoner) ist eine Software von der Firma Racer Sytems GmbH&Co. KG entwickelt und wird als eine Reasoning System mit graphisches Benutzerinterface und vollständigen Werkzeuge zur Analysis und Wiedergewinnung der semantischen Informationen von ABoxen und TBoxen, die mit OWL oder RDF die Ressourcen von dem Semantischen Web beschreiben, verwendet. Wie andere Beschreibungslogik-Systemen benutzt RacerPro die Open-World-Assumption(OWA) für Reasoning. Das bedeutet, die Aussagen, die nicht mit den bereits existierenden Wissens als wahr bewiesen werden können, sind nicht als falsch betrachtet. Mit der Anreicherung der existierenden Wissens ist es möglich ist, die Wahrheit der Aussagen bewiesen werden zu können. RacerPro bietet zwei Default-Port 8088 und 8080 für TCP-

und HTTP-Kommunikation. Außerdem bietet Racerpro noch zwei APIs JRacer und LRacer, die auf TCP-Sockets aufgebaut werden, für Java-Programm und Common-Lisp zum Dienstaufruf von RacerPro.

Weitere ausführliche Benutzeranleitung und technische Dokumente für RacerPro können auf der Webseite von Racer System GmbH erhalten werden. In der Arbeit wird RacerPro Version 1.9 benutzt, um die semantischen Individuen von ABoxen der Webseiten wieder zu gewinnen.

2.5.2 RacerPro Query Sprache (nRQL)

Die Abfragen, die nach RacerPro geschickt und zur Wiedergewinnung der semantischen Individuen von der ABox verwendet sind, werden mit der Abfragesprache nRQL (new RacerPro Query Language) geschrieben. Die nRQL-Sprache ist häufig für folgende Funktionen zu benutzen [RPUG08-C6],

- Abfrage der RacerPro-ABoxen;
- Abfrage der RacerPro-TBoxen;
- Abfrage der RDF- oder RDFS-Dokumente;
- Abfrage der OWL-Dokumente;

Eine nRQL-Abfrage besteht im Wesentlichen aus einem „Query-Head“ und einem „Query-Body“. Die einfachen Retrieval-Abfragen werden beispielsweise mit der formalen Syntax *(retrieve <query-head> <query-body>)* dargestellt werden, wobei

- **query-head:** spezifiziert das Format der Abfrageantworten und kann auch die Individuen enthalten.
- **query-body:** spezifiziert die Bedingungen der Abfragen.

Das Query-Head und Query-Body werden weiter von den „Query-Atoms“ aufgebaut. Jedes Query-Atom ist entweder unär oder binär [RPUG08-C6]. Ein unäres Query-Atom bezieht sich nur auf ein Objekt und wird als „Concept-Query-Atom“ für Abfrage der Konzepte verwendet. Ein binäres Query-Atom bezieht sich auf zwei Objekte und wird als „Role-Query-Atom“ für Abfrage der Beziehungen zwischen die Konzepte oder auch als „Constrain-Query-Atom“ für Beschränkung der Beziehung zwischen die Konzepte verwendet. Jedes Objekt ist entweder eine Variabel oder ein Individuum. Die Variabeln müssen mit einem Präfix „?“ oder „$?“ (für injektive Variabeln) in nRQL-Sprache dargestellt werden. Einige Beispiele werden folgend aufgezeigt [WES06].

- Beispiel für Individuen: *Bettina, individual-1, family*.
- Beispiel für Variabeln: *?x, ?Y, $?yX*.
- Beispiel für das Concept-Query-Atom zur Retrieval alle Individuen, welche mit dem Konzept *woman* definiert wurden: *?(retrieve (?x)(?x woman))*; Antwort: *>(((?X BETINA))(?x SABINE)))*.
- Beispiel für Role-Query-Atom zur Retrieval alle Individuenpaare, welche die Beziehung von *has-child* haben: *?(retrieve (?x ?y)(?x ?y has-child))*; Antwort: *>(((?X MATTHIAS)(?Y JOHANNES))((? X MATTHIAS)(?y BETINA)))* .
- Beispiel für Constaint-Query-Atom zur Retrieval alle Individuen, deren Vater und Muter im gleichen Alter sind: *?(retrieve (?x)(?x ?x (constraint (has-father age)(has-mother age) =))*; Antwort: *(((?X CHRISTINA)))* .

Durch Verwendung der „Head-Projection-Operators“ in Query-Head können auch die Attribute oder die Datatype-Properties der Objekte abgefragt werden. Ein Head-Projection-Operator ist ähnlich wie eine Funktion, welche zur Verbindung und Operation der aktuellen Variabeln oder Individuen verwendet wird [RPUG08-C6]. Damit können die zusätzlichen Attribute von den aktuellen Variabeln oder Individuen abgefragt werden. Einige Beispiele werden folgend aufgezeigt [WES06].

- Beispiel für den Projection-Operator zur Retrieval alle Individuen, welche mit dem Konzept *grandmother* definiert und ein Attribut *AGE* besitzt, und auch die entsprechenden Attributnamen von dem Attribut AGE : *?(retrieve (?x (AGE ?x)) (?x (and grandmother (an age))))*; Antwort: *>(((?X SABINE) ((AGE ?X)(SABINE-AGE))))* .
- Beispiel für den Projection-Operator *told-values* zur Retrieval alle Individuen, welche mit dem Konzept *grandmother* definiert und ein Attribut *AGE* besitzt, und auch die entsprechenden Attributwert von dem Attribut AGE : *?(retrieve (?x (TOLD-VALUE (AGE ?x))) (?x (and grandmother (an age))))*; Antwort: *>(((?X SABINE) ((AGE ?X)((:TOLD-VALUE (AGE ?X)) (80))))* .

In der nRQL-Sprache wurden bereits einigen Head-Projection-Operators für häufige benutze Abfragen definiert. Die Funktionsangaben aller vordefinierten Head-Projection-Operators kann man in dem Bedingungshandbuch [RPRM07] für RacerPro finden. Außerdem bietet die nRQL-Sprach auch eine einfache Ausdruckssprache „MiniLisp“, welche eine Submenge von Common Lisp ist und mit einem so genannt „Lambda-Head-Operator“ die eigene Head-Projection-Operators definieren kann. Die Funktion, die mit Lambda-Head-Operator aufgebaut wird, wird eine anonyme Funktion oder Lambda-

Funktion genannt und zur Operation der aktuellen Variabeln oder Individuen verwendet. das Format der Lambda-Funktion und die zugehörige Beispiele werden wie folgend gezeigt.

- Format der Lambda-Funktion:
  ((lambda (variabeln) (Operationsausdruck)) (Wertzuweisung))
- Beispiel für eine Lambda-Funktion zum Berechnen der Summe von zwei Zahlen: *((lambda (x) (+ x x)) (* 3 4))*; Ergebnis: *24*
- Beispiel für eine ABox vom Bücherladen [RE07-C5]:

```
(full-reset)
(instance b1 book)
(instance b2 book)
(related b1 a1 has-author)
(related b1 a2 has-author)
(related b2 a3 has-author)
(define-concrete-domain-attribute price :type real)
(instance b1 (= price 10.0))
(instance b2 (= price 20.0))
```

- Beispiel für eine Abfrage mit Lambda-Funktion zur Retrieval aller Individuen, welche mit dem Konzept *book* definiert wurden, und zugleich auch die Anzahl des entsprechenden Autors jedes Individuums [RE07-C5]:

```
? (retrieve (?x
              ((lambda (book)
                 (let ((authors
                         (retrieve `(?a) `(,book ?a has-author))))
                   (length authors)))
               ?x))
      (?x book)
      :dont-show-lambdas-p t)
> (((?x b1) 2) ((?x b2) 1))
```

Der Vorteil durch die Anwendung von MiniLisp liegt daran, dass die mehreren untereinander abhängigen einzelnen Abfragen in einer Abfrage konstruiert und bei der Server-Seite (RacerPro) zusammen ausgeführt werden können. Weitere Angaben über die Anwendung der Lambda-Funktion und MiniLisp-Ausdrücke können in den Lehrmaterialen [RE07-C5], [RPUG08-C6] vom Internet gefunden werden.

3 Konzeption

Dieses Kapitel stellt das Entwurfskonzept des semantischen Filters vor. Im Unterkapitel 3.1 wird zunächst das Arbeitsprinzip des semantischen Filters kurz vorgestellt. im Unterkapitel 3.2 werden der grundlegende Aufbau des semantischen Filters durch einem Komponentendiagramm graphisch vorgestellt. Die Konstruktiven Einzelheiten über die wichtigen Bauteile werden separat in späteren Unterkapiteln weiter erläutert.

3.1 Funktionsweise

Diese Arbeit wird als eine Erweitere Erforschung vom BOEMIE-Projekt für Mobiltelefon vorgenommen. In der Arbeit wird versucht, ein semantisches Filtersystem aufzubauen, welches mit Hilfe vom Reasoning-System RacerPro die semantischen Metadaten (ABox) der Webseiten analysiert und gemäß den extern definierten Filterregeln die uninteressierten Webseiteninhalte filtert. Die gefilterten Webseiten sollen auch dadurch in einem mobiltelefon-lesbaren Format übersetzt. Damit können die Webseiten in kleinen Mobiltelefonen besser dargestellt werden. Eine Übersicht über die Funktionsweise des semantischen Filters wird in der Abbildung-3.1 grob dargestellt.

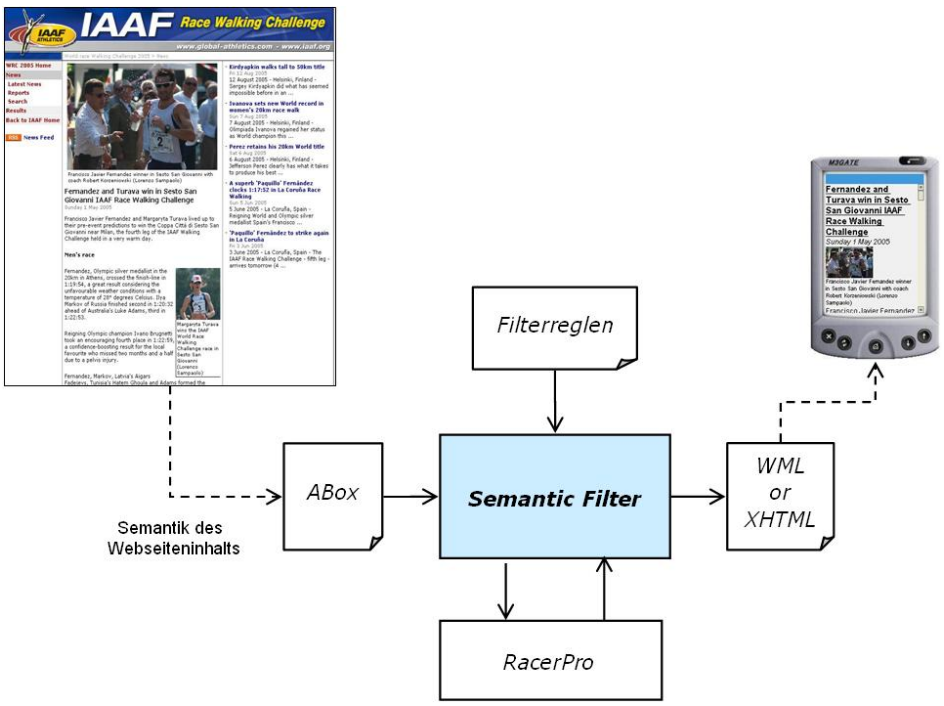

Abbildung-3.1
Übersicht über die Funktionsweise des semantischen Filters zum Filtern des Webseiteinhalts für Mobiltelefon

Der semantische Filter besteht aus vier Prozessschritten: Retrieval, Extraktion, Filterung und Übersetzung. Der Prozessvorgang vom semantischen Filter wird in Abbildung-3.2 schematisch dargestellt.

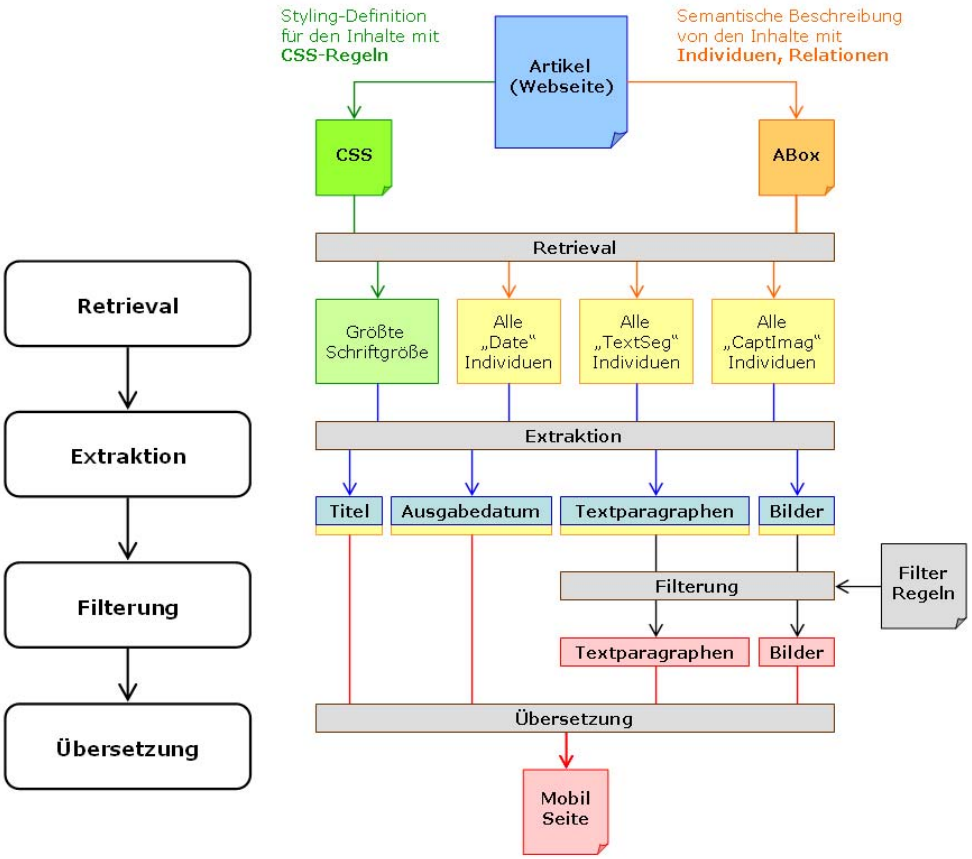

Abbildung-3.2
Schematische Darstellung für den Prozessvorgang des semantischen Filters

Als Webseiteninhalte werden nur die Artikel, welche aus Artikelelemente Titel, Ausgabedatum, Textinhalte, Bilderinhalte bestehen, in der Arbeit untergesucht. Da die Ontologien von BOEMIE nur über die athletischen Domänen für die Webseiten aufgebaut sind und die meisten untergesuchten Webseiten von BOEMIE durch einen Artikel über die athletischen Wettkämpfe berichten. Die semantische Metadaten von den Webseiteninhalten werden in einer ABox von BOEMIE geschrieben. Die Styling-Definitionen, wie Schriftart, Schriftgröße für die Webseiteninhalten werden in einer CSS-Datei geschrieben.

Beim Retrievalsprozess werden zuerst die benötigen semantischen Informationen und die Style-Attribute über die Artikelelemente von ABox und CSS-Datei der Webseiten wieder gewonnen. Durch die Analysis der semantischen Information und Style-Attribute

werden die Artikelelemente: Titel, Ausgabedatum, Textparagraph und Bilder beim Extraktionsprozess auf dem Quellcode der Webseiten heraus extrahiert und dann zusammen mit ihren beinhalteten semantischen Individuen in verschiedenen Datenstrukturen gespeichert. Die Textparagraphen und Bilder werden bei Filterungsprozess gemäß den extern definierten Filterregeln gefiltert. Nach dem Filterungsprozess werden alle Artikelelemente beim Übersetzungsprozess in einem mobiltelefon-lesbaren Format übersetzt. Die Einzelheiten jedes Schritts werden in späteren Unterkapiteln weiter erläutert.

3.1.1 ABoxen der Webseiten

Die Grundkenntnisse für die ABox wurden bereits im Abschnitt 2.4.2 vorgestellt. In der Arbeit wurden einige Webseite-ABoxen bereits als Beispiele von BOEMIE-Projekt generiert und sind online erhältlich. Die URL-Adressen für Webseiten und ABoxen können im Anhang-A gefunden werden.

Die Bilder und Textsegmente von einer Webseite, deren Begriffe bereits in BOEMIE-Ontologien definiert worden sind, sowie auch untereinander liegende Zusammenhänge werden als die Individuen und Relationen in einer ABox mit OWL-Sprache formal beschrieben. Eine Webseite von IAAF wird als Beispiel in der unteren Abbildung-3.5 links dargestellt. Wir nehmen einen Teil (blauer Rahmen) vom zentralen Inhalt heraus und vergrößern in der Abbildung-3.3 rechts, um die semantischen Informationen von der entsprechenden ABox zu erklären.

Der in der Abbildung-3.3 rechts gezeigte Inhalt besteht in der ABox aus zwei Individuen von den Konzepten kommentiertes Bild und ein Textinhalt, die in der Abbildung-3.3 rechts markiert werden. Das Bild wird weiter mit den Subindividuen vom Konzept Still-Region wie Gesicht und Körper von der dargestellten Person beschrieben. Jede Still-Region hat eigene Flächenkoordinate zu lokalisieren. Der Bildkommentar und Textinhalt wird weiter in mehreren Subindividuen vom Konzept Textsegmente zersetzt, welche auf dem Textinhalt mit roten Rahmen markiert wurden, um die semantischen Bedeutungen von den Textinhalte zu beschrieben. Jedes Text-Segment kann durch eine Offset-Koordinate lokalisiert werden. Alle Konzepte für oben erwähnten Individuen und Subindividuen wurden bereits mit Anwendung von BOEMIE-Ontologien im TBox formal definiert. Im BOEMIE-Projekt werden nicht alle Inhalte in der ABox beschrieben. Da die notwendige Konzepte noch nicht vollständig in den BOEMIE-Ontologie definiert und evolviert werden.

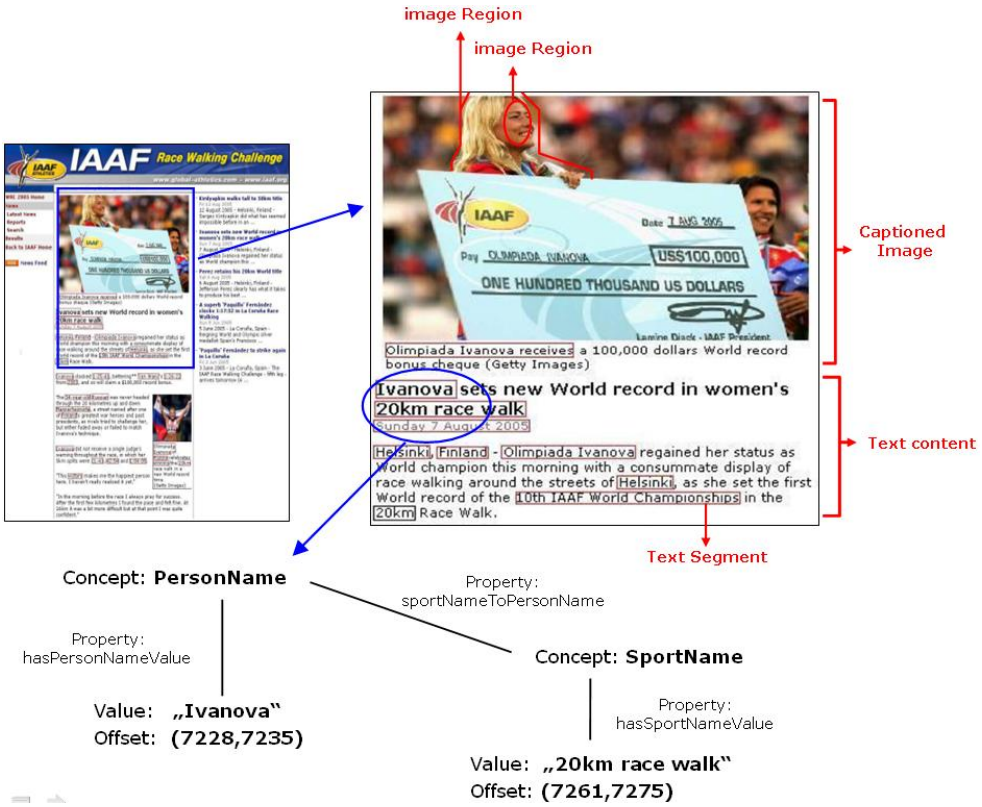

Abbildung-3.3
Beispiel für eine Webseite(IAAF) und die semantischen Individuen vom Webseiteninhalt

Nicht nur Individuen sondern auch die Relationen, die zwischen Individuen existieren, werden auch in der ABox beschrieben. Wie in der Abbildung-3.3 unten gezeigt wurde, dass es zwei Individuen aus dem Titel gibt, Eines wird vom Konzept *PersonName* definiert und mit dem Wert *„Ivanova"* und der Offset-Koordinate (7228, 7235) zugewiesen und Anderes wurde vom Konzept *SportName* definiert und mit dem Wert *„20km race walk"* und der Offset-Koordinate (7261, 7275) zugewiesen. Der Zusammenhang zwischen den zwei Individuen wird durch die Relation *sportNameToPersonName* dargestellt.

Eine Webseite-ABox ist tatsächlich eine OWL-Datei. Das Individuum „Ivanova" kann beispielsweise mit folgendem OWL-Code beschrieben werden.

```
<mco:TextSegment rdf:ID="IDF1AF878E-6865-11DD-89ED-00137238A351-boemie_text_ncsr-skel_phrase_2">
<mco:depicts rdf:resource="#IDF1AF878E-6865-11DD-89ED-00137238A351-boemie_text_ncsr-skel_name_1225"/>
<mco:hasSegmentLocator rdf:resource="#IDF1AF878E-6865-11DD-89ED-00137238A351-boemie_text_ncsr-skel_phrase_2-mco_TextualLocator"/>
</mco:TextSegment>

<aeo:PersonName rdf:ID="IDF1AF878E-6865-11DD-89ED-00137238A351-boemie_text_ncsr-skel_name_1225">
<aeo:hasPersonNameValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string">olimpiada ivanova
</aeo:hasPersonNameValue>
```

```
</aeo:PersonName>

<mco:TextualLocator rdf:ID="IDF1AF878E-6865-11DD-89ED-00137238A351-boemie_text_ncsr-skel_phrase_2-
mco_TextualLocator">
<mco:hasStartOffset rdf:datatype="http://www.w3.org/2001/XMLSchema#int">7228</mco:hasStartOffset>
<mco:hasEndOffset rdf:datatype="http://www.w3.org/2001/XMLSchema#int">7235</mco:hasEndOffset>
</mco:TextualLocator>
```

Code-3.1: formale Beschreibung für Individuum "Ivanova" in OWL-Dokument

Der im Code-3.1 gezeigte Präfix „mco" und „aeo" ist den Ontologiename, der bereits im Abschnitt 2.4.2 vorgestellt wurden. Nach den Präfixen folgten die Konzeptnamen oder Relationsnamen.

3.1.2 Filterregeln

Die Filterregeln wird in einer externen Datei erstellt und können jeder Zeit nach unterschiedliche Anforderungen geändert werden. Wie im letzten Abschnitt erwähnt wird, werden die semantischen Bedeutungen der Webseiteninhalte in der ABox durch Individuen semantisch repräsentiert. Die Konzepte der Individuen sind in Ontologien definiert. Die Filterregeln sind tatsächlich eine Gruppe von semantischen Nebenbedingungen, die durch Beschränkung des Bereichs von Konzepte und Individuen das Bereich der wichtigen Webseiteninhalte bestimmt. Die in den Filterregeln zur Beschränkung der wichtigen Inhalte benutzten Konzepte und Individuen werden **„Fokuskonzept"** und **„Fokusindividuen"** genannt. Die Webseiteinhalte, deren Semantik nicht die Fokuskonzepte und Fokusindividuen beinhaltet, können als unwichtige Inhalte weg gefiltert werden. Die weiteren Einzelheiten für Filterregeln werden bei Filterungsprozess im Unterkapitel 3.5. weiter erläutert.

3.2 Aufbau des semantischen Filters

Das semantische Filter besteht aus folgenden Hauptkomponenten, deren Zusammenhänge durch das Komponentendiagramm von der Abbildung-3.4 dargestellt werden.

- **Semantic-Filter-Model** ist der zentrale Bauteil des semantischen Filters zur Implementierung der Extraktionsprozess und Filterungsprozess vom Arbeitsprinzip des semantischen Filters.
- **Web-Page-Property-Picker** ist ein Bauteil zum Gewinnung der nicht-semantischen Metadaten wie CSS-Regeln und Quellcode von den Webseiten beim Retrievalsgsprozess vom Arbeitsprinzip des semantischen Filters verwendet.

- **Web-Page-Semantic-Picker** ist ein Bauteil zum Gewinnung der semantischen Metadaten von den Webseiten beim Retrievalsprozess vom Arbeitsprinzip des semantischen Filters verwendet.
- **WML-Page-Generator** ist ein Bauteil beim Übersetzungsprozess zur Generierung der mobilen Seiten.

Außerdem wird das RacerPro als eine Resoning-Maschine zur Analysis der Webseite-ABoxen. Die Filterregeln werden in einer externen Datei geschrieben und kann wieder definiert werden.

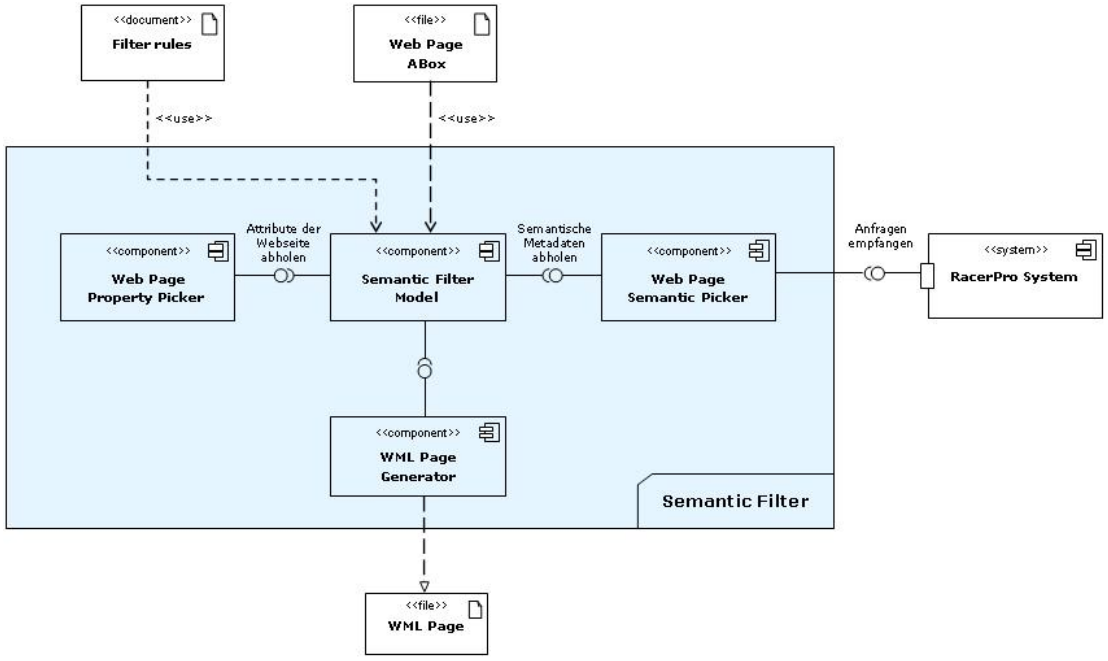

Abbildung-3.4
Komponentendiagramm des semantischen Filters

3.3 Retrievalsprozess

In diesem Unterkapitel geht es um die Einzelheiten von den Prozessen zur Sammlung der semantischen und nicht-semantischen Information von der Webseite, welche für semantische Analysis und Identifizierung der Inhaltselemente benötigt sind. Im Abschnitt 3.3.1 wird die Inhaltselemente, die in der Arbeit als zentrale Inhalte der Webseite berücksichtigt sind, vorgestellt. Im Abschnitt 3.3.2 wird der Prozess zur Gewinnung der Attributwerte von der CSS-Datei vorgestellt. Im Abschnitt 3.3.3 wird der Prozess zur Gewinnung der Semantik von der ABox vorgestellt.

3.3.1 Inhaltselemente der Webseite

Eine übliche Webseite besteht häufig aus vier Zonen von Kopffeld, Navigationsfeld, Körperfeld und Fußzeilenfeld auf der Layoutebene, die wie Abbildung-3.5 gezeigt wurden. Jede Zone wird auch gemäß der Funktionalität und Entwurfskonzepte der Website in unterschiedlichen Formen dargestellt. Wir konzentrieren uns in der Arbeit nur auf die Multimediainhalte vom Körperfeld von den Webseiten, die von IAAF erzeugt werden und nur einen Artikel über athletische Nachrichten beinhalten. Die anderen Zonen der Webseite werden in der Arbeit nicht berücksichtigt. Der Inhalt des Körperfelds ist häufig durch unterschiedliche Multimedieninhalte wie Text, Tabelle, List, Bilder sogar Video repräsentiert. Die Artikel von den IAAF-Webseiten werden als zentrale Objekte und die Inhalte der mobilen Webseiten zur semantischen Filterung verwendet.

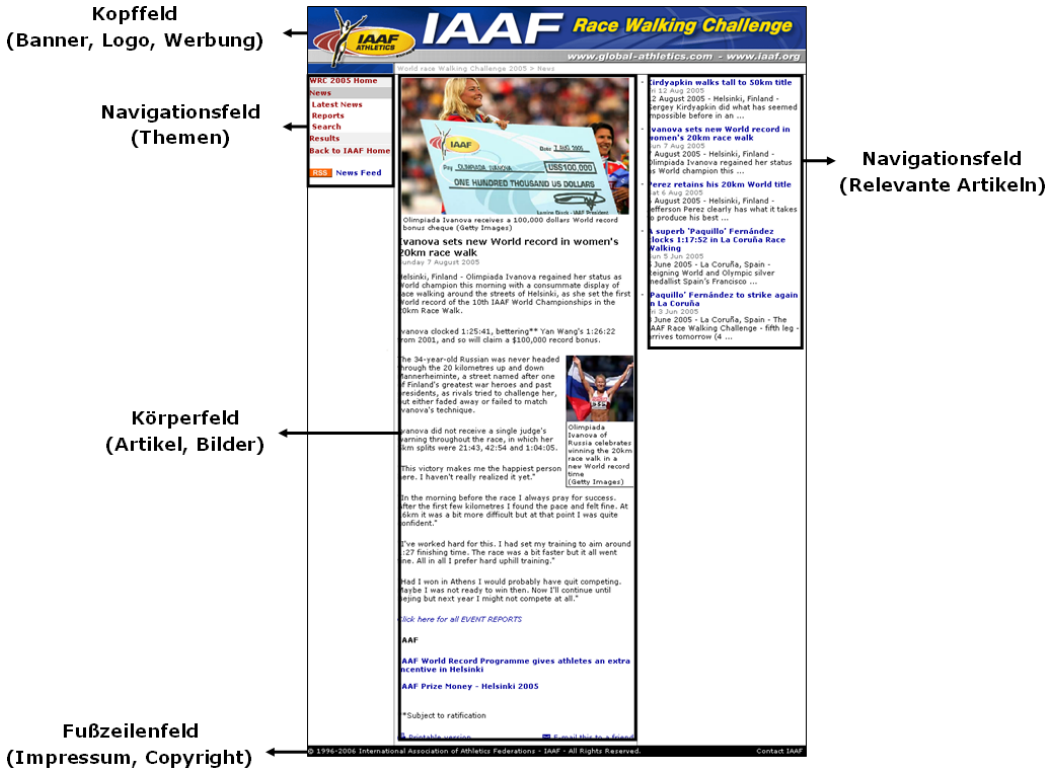

Abbildung-3.5
Beispiel für die Layoutzonen der IAAF-Webseiten

Ein Artikel besteht weiterhin aus Titel, Datum, Textsegment und Bildsegment. Die weiteren Medienobjekte wie Flash, Video werden in der Arbeit nicht berücksichtigt. Das Beispiel für einen Artikel vom Körperfeld der IAAF-Webseite wird in der Abbildung-3.6 dargestellt. Diese vier Bestandteile vom Artikel werden als die wesentlichen Inhaltselemente für die endlichen mobilen Webseiten in der Arbeit betrachtet und mit den Name „Artikelelemente“ genannt, sie sollen mit unterschiedlichen Filterregeln separat gefiltert werden.

Durch folgende Charakteristik können die Artikelelemente vom HTML-Quellecode heraus identifiziert.

- **Überschrift (heading):** Ein Textsegment, das mit größter Schriftgröße definiert wird.
- **Ausgabedatum (update date):** das erste auftretende Individuum vom Konzept „Date“ im Artikel und es wird auch mit einer Vollständigen Form von „Tag-Monat-Jahr“ dargestellt.
- **Textabschnitt (paragraph):** Ein Textsegment, welches mit HTML-Tag … geklammert und die Individuen, die in der Webseite-ABox geschrieben wurden, beinhaltet.
- **Bildabschnitt(captioned image):** Ein Individuum, welches vom Konzept „CaptionedImage“ definiert und in der Webseite-ABox beschrieben.

Diese oben gegebenen Definitionen sind nicht für alle Webseiten geeignet, es handelt sich nur um die Webseiten von IAAF, die in der Arbeit als Beispiele benutzt werden. Die Individuen für Textabschnitte und Bildabschnitte wurden bereits in der Webseite-ABox mit OWL beschrieben und können direkt durch RacerPro von der ABox wieder gewonnen werden.

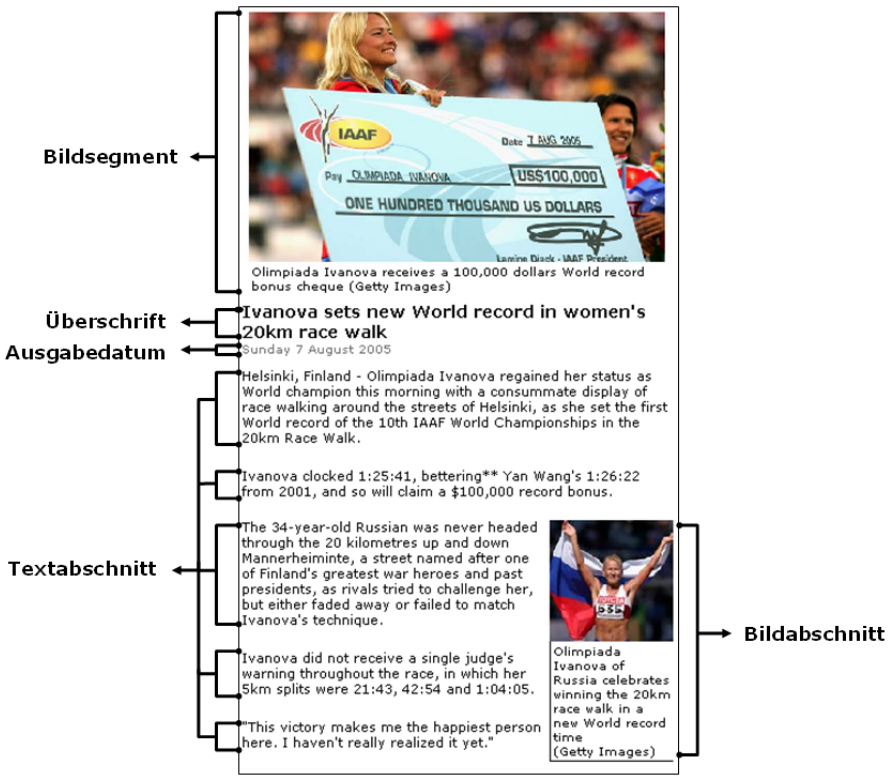

Abbildung-3.6
Artikelelemente von der IAAF-Webseite

3.3.2 Gewinnung der Attribute von CSS-Datei

Nicht alle semantischen Bedeutungen von den Webseiten werden in der ABox beschrieben. Das Inhaltselement, deren Semantik nicht in der ABox geschrieben werden, müssen durch Gewinnung und Analysis ihrer Attributwerte von CSS-Datei oder Quellcode der Webseiten identifiziert werden. Wir nehmen eines Beispiel zu erklären, wie wird eine Artikelüberschrift mit Hilfe der Attributwerte von CSS-Datei heraus identifiziert. Da es kein entsprechendes Konzept in den BOEMIE-Ontologien gibt, um die Artikelüberschrift zu definieren. Die Artikelüberschrift wird normalerweise im HTML-Code entweder durch HTML-Tags wie H1, H2…H6 oder durch DIV-Tag mit CSS-Regeln konstruiert. Durch Prüfung von HTML-Tag und Vergleichen der Attributwerte von CSS-Regeln wird die Artikelüberschrift vom Webseite-Quellcode heraus gefunden werden.

Außerdem können durch Gewinnung von weiteren Attributwerten wie Schriftart, Schriftfarbe, Layout von CSS-Regeln die anderen Inhaltselemente der Webseite identifiziert werden. Diese Attribute der Inhaltselemente können auch bei Filterungsprozess als Nebenbedingungen der Filterregeln verwendet werden. Der Prozess zur Gewinnung der Attribute von CSS-Regeln wird in der Komponente Web-Page-Property-Picker implementiert.

3.3.3 Gewinnung der Semantik von der ABox

Die Gewinnung der semantischen Informationen von der ABox ist durch Versand der Abfragen nach RacerPro realisiert. Die einigen Medienobjekte wie Textsegmente und Bilder auf der Webseiten sind bereits in der ABoxen des BOEMIE-Projekts als Individuen der Webseiteinhalte semantisch beschrieben worden und können später beim Filterungsprozess zur Prüfung des Fokusbereichs gemäß der Beschränkung von Fokuskonzepte in der Filterregeln verwendet werden. Durch Gewinnung der semantischen Informationen von der ABox sind auch die Attribute und Inhalte der Inhaltselemente wie URL der Bilder und Bildtexte einfach erhältlich, da diese Informationen in der ABox geschrieben worden sind.

Die Antworten von RacerPro werden im LISP-Ausdruck mit verschachtelten Klammern dargestellt und können nicht benutzt werden. Die unnützlichen Symbole müssen von den Antworten beseitigt werden.

Der oben erwährte Prozessvorgang wird in der Abbildung-3.7 in der Komponente *Web-Page-Semantic-Picker* implementiert. Es gibt noch drei zusätzlichen Komponenten *RacerPro-Query-Maker, RacerPro-Connector* und *RacerPro-Parser* zur Implementierung

der verschiedenen Teilaufgaben des Prozesses. Die kurzen Angabe für jede Komponente werden wie unten geschrieben, die weitere Einzelheiten zu jeder Komponente werden im Kapitel 4 weiter vermittelt.

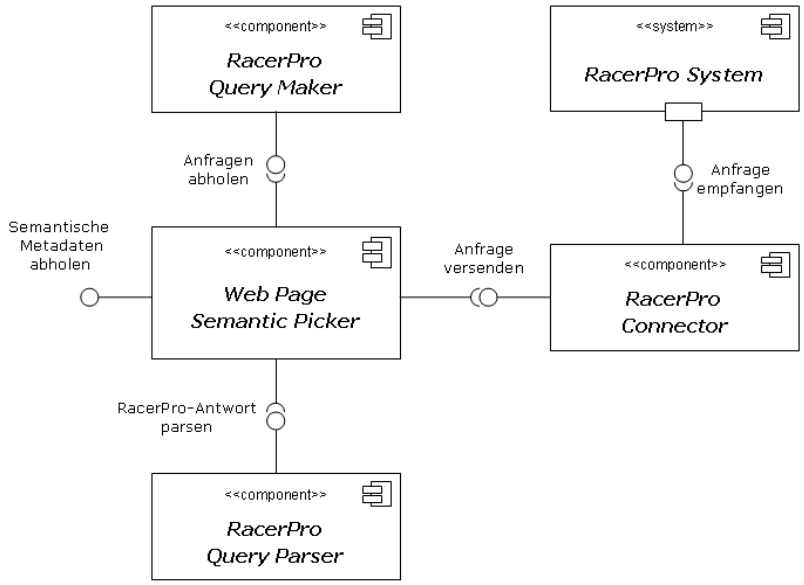

Abbildung-3.7
Komponentendiagramm für Retrieval der semantischen Informationen von der ABox

- **RacerPro-Query-Maker:** In dem werden alle zur Retrieval der semantischen Informationen verwendete RacerPro-Abfragen konstruiert.
- **RacerPro-Connector:** Es versendet die Abfrage nach RacerPro und empfängt die Antwort von RacerPro.
- **RacerPro-Query-Parser:** Es analysiert das von RacerPro beantwortete Ergebnis und extrahiert die semantischen Metadaten raus.
- **Web-Page-Semantic-Picker:** Es verwaltet die drei oben genannten Komponenten, um den Prozess zur Retrieval des semantischen Informationen auszuführen und steht das Ergebnis an der öffentlichen Methode zur Verfügung.

3.4 Extraktionsprozess

Nachdem die benötigen Attributwerte von CSS-Regeln und semantischen Informationen von der ABox gewonnen werden, können die Artikelelemente vom HTML-Quellecode heraus extrahiert werden. Wie im Abschnitt 3.3.1 erwährt wurde, dass die Artikelelemente aus Überschrift, Ausgabedatum, Textabschnitt und Bildabschnitt besteht

und als die Inhalte der mobilen Webseiten zur semantischen Filterung verwendet werden. In diesem Abschnitt werden die Methoden zur Extraktion jedes Artikelelements erläutert.

3.4.1 Extraktion der Überschrift

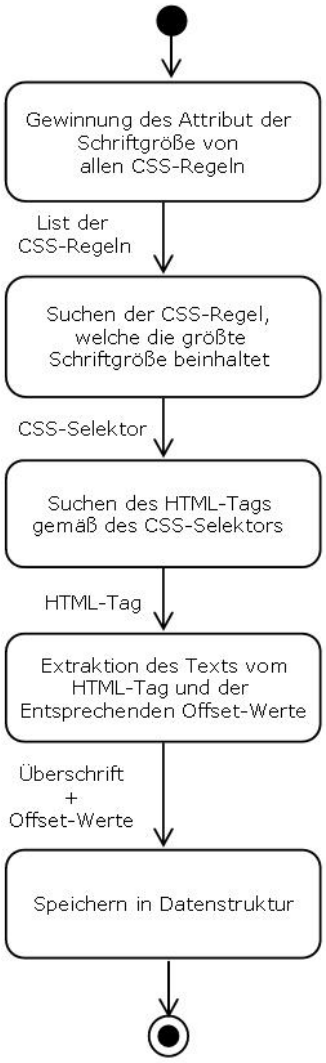

Abbildung-3.8
Aktivitätsdiagramm für den Prozess zur Extraktion der Überschrift des Artikels

Der Algorithmus zur Extraktion der Überschrift des Artikels wird durch das Aktivitätsdiagramm von der Abbildung-3.8 illustriert. Das Grundprinzip des Algorithmus ist zum Suchen eines Textsegments vom Artikel der Webseite, das mit der größten Schriftgroße geschrieben wird.

Es gibt normalerweise zwei Methoden in HTML zur Definition einer Überschrift. Die erste Methode ist durch Anwendung von H1- bis H6-Tag direkt die Texte der Überschriften zu kennzeichnen. Der H1-Tag definiert die größte Überschrift und der H6-Tag definiert die kleinste Überschrift. Die Schriftgroße von solchen HTML-Tags kann auch mit CSS-Regeln erneut umgeschrieben werden. Die zweite Methode ist durch Anwendung vom DIV-Tag und den CSS-Regeln realisiert. DIV-Tag kennzeichnet den Text der Überschrift und Die CSS-Regel definiert die Attribute der Überschrift. In den IAAF-Webseiten sind die Überschriften der Artikel durch DIV-Tag und CSS-Regeln definiert.

Dazu müssen das Attribut der Schriftgröße von alle CSS-Regeln übergeprüft werden, um die CSS-Regel, die größte Schriftgröße definiert, zu finden. Wenn die angepasste CSS-Regel gefunden würde, kann man den entsprechenden DIV-Tag, der diese CSS-Regel benutzt und die Überschrifttext beinhaltet, vom HTML-Quellcode bestimmen. Damit die Überschrift und die entsprechenden Offset-Werte, die Position der Überschrift, einfach heraus extrahiert werden können.

3.4.2 Extraktion des Ausgabedatums

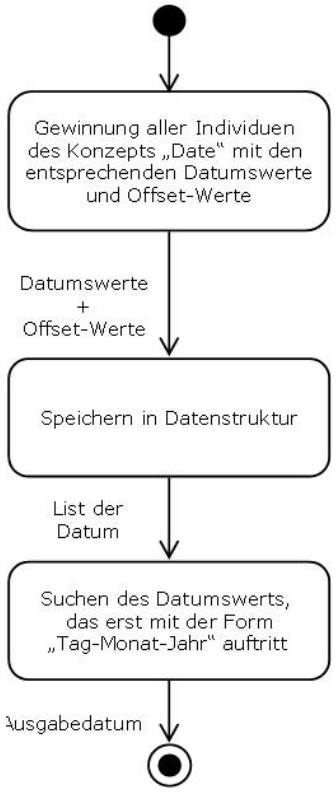

Abbildung-3.9
Aktivitätsdiagramm für den Prozess zur Extraktion des Ausgabedatums des Artikels

Der Algorithmus zur Extraktion des Ausgabedatums des Artikels wird durch das Aktivitätsdiagramm von der Abbildung-3.9 illustriert. Das Grundprinzip des Algorithmus ist zum Suchen eines Individuums, welches in der ABox mit dem Konzept „Date" in einer vollständiger Darstellung von Tag-Monat-Jahr definiert wurde und unten der Überschrift des Artikels zuerst vorkommt.

Die Position eines Individuums wird durch die Offset-Werte festgelegt. Alle in der ABox beschriebenen Individuen haben zwei Attribute „hasStartOffset" und „hasEndOffset" um die Position in der Webseite zu markieren. Die Werte von diesen zwei Attribute sind eine zweidimensionale Koordinate mit zwei ganzen Zahlen zur Notierung der Abweichung zwischen dem Anfangspunkt der Webseite und dem Beginn bzw. Ende des Individuums. Die Reihenfolge von alle „Date"-Individuen kann durch Vergleichen der Offset-Werte angeordnet werden. Tatsächlich in der ABox alle Individuen sind bereits gemäß aufsteigender Reihenfolge angeordnet.

Das Ausgabedatum muss vollständig im Form Tag-Monat-Jahr dargestellt werden. Es ist eine wichtige Nebenbedingung zur Erkennung des Ausgabedatums von andere „Date" Individuen.

3.4.3 Extraktion der Textabschnitte

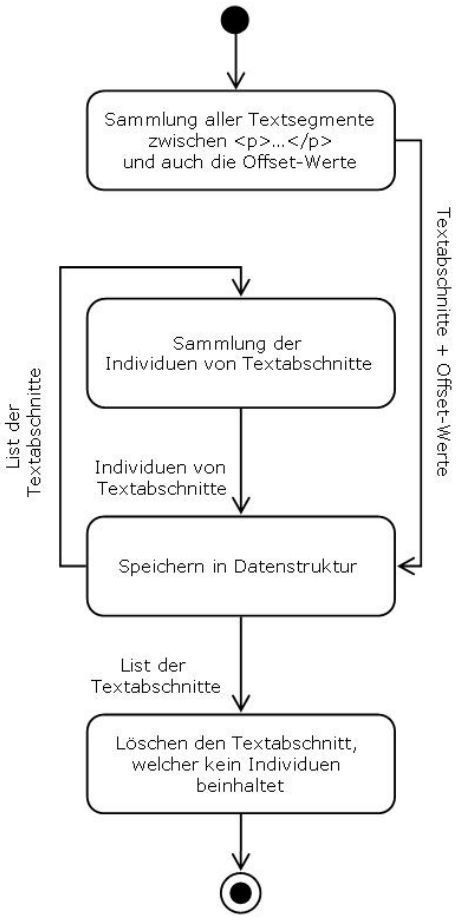

Abbildung-3.10
Aktivitätsdiagramm für den Prozess zur Extraktion der Textabschnitte des Artikels

Der Algorithmus zur Extraktion der Textabschnitte des Artikels wird durch das Aktivitätsdiagramm von der Abbildung-3.10 illustriert. Das Grundprinzip des Algorithmus ist zur Sammelung aller Textsegmente vom HTML-Quellcode, die mit dem P-Tag eingeschlossen und außerdem deren Semantik auch in der ABox beschrieben werden.

Der P-Tag ist in HTML speziell zur Kennzeichnung der Paragraphen verwendet. Die Extrahierung der Textinhalte vom P-Tag kann einfach durch Programm implementiert werden. Dazu müssen die unrelevante Symbole wie HTML-Tage von den extrahierten Texten beseitigt werden. Außerdem sollen auch die Offset-Werte vom Beginnpunkt und Endpunkt des Textabschnitts heraus gefunden werden, um dazwischen liegende Individuen, die in der ABox vom Konzept „Textsegment“ definiert, zu sammeln. Das Ziel zur Sammelung aller „Textsegment“-Individuen ist, um den Textabschnitt zu prüfen, ob er die Fokuskonzepte und Fokusindividuen, welche im Abschnitt 3.1.2 erwähnt und in den Filterregeln definiert werden, beinhaltet.
Alle bekommende Textabschnitt sowie auch die zugehörigen Offset-Werte und semantischen Individuen werden in einer speziellen Datenstruktur gespeichert. Die Textabschnitte, die keine semantischen Individuen beinhaltet, werden nicht berücksichtigt.

3.4.4 Extraktion der Bildabschnitte

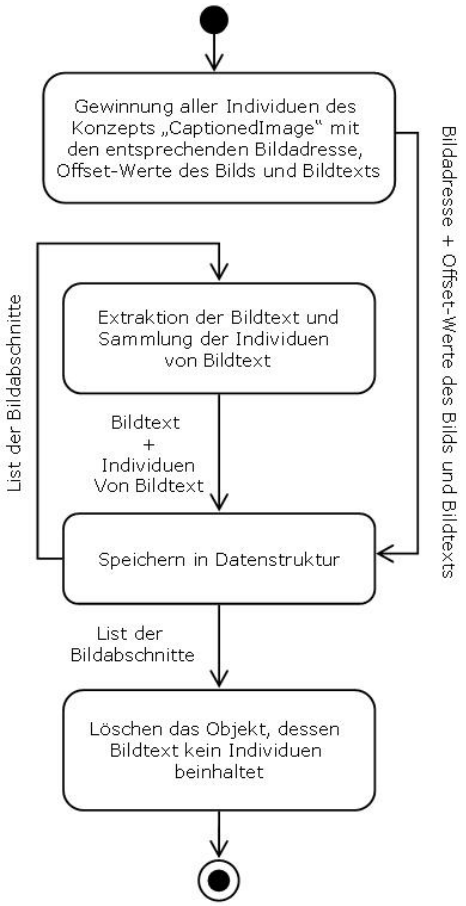

Abbildung-3.11
Aktivitätsdiagramm für den Prozess zur Extraktion der Bildabschnitte des Artikels

Der Algorithmus zur Extraktion der Bildabschnitte des Artikels wird durch das Aktivitätsdiagramm von der Abbildung-3.11 illustriert. Das Grundprinzip des Algorithmus ist zur Sammelung aller Individuen des Konzepts „CaptionedImage“ von der ABox. Da die Bildadresse und die Offset-Werte des Bild und Bildtexts bereits in der ABox beschrieben worden sind. Diese drei Attributwerte von allen „CaptionedImage“ wurden beim Retrievalsprozess gewonnen und können bei Extraktionsprozess einfach bekommen werden. Die Bildtexte können gemäß seiner Offsetwerte vom HTML-Quellcode gefunden und heraus extrahiert werden.

Außerdem sollen auch die semantischen Individuen von den Bildern gesammelt werden. Das Ziel zur Sammelung aller Individuen von den Bildern ist wie bei Extrahierungsprozess für Textabschnitte, um den Bildabschnitt zu prüfen, ob er die Fokuskonzepte und Fokusindividuen, die in den Filterregeln definiert werden, beinhaltet. Alle bekommende Bildadresse, Bildtexte und die zugehörigen Offset-Werte werden in einer speziellen Datenstruktur gespeichert. Die Bildabschnitte, die keine semantischen Individuen beinhaltet, werden nicht berücksichtigt.

3.5 Filterungsprozess

Nachdem alle Artikelelemente vom HTML-Quellcode extrahiert worden sind, kann der Inhalt jedes Artikelelements gemäß unterschiedlichen Filterregeln gefiltert werden. In der Arbeit filtern wir nur den Inhalt von Textabschnitte und Bildabschnitte von den Artikeln. Da die Prozesse zur Extraktion und Filterung jedes Artikelelements untereinander unabhängig sind, kann für jedes Artikelelement ein Threads zur Extraktion und Filterung verwendet werden, um die Laufzeit des Programms zu kürzen. Wie die Aktivitätsdiagramm von Abbildung-3.14 gezeigt wird, dass nach dem Retrieval der semantischen Informationen vom ABox und der Attribute von den CSS-Regeln werden vier Threads gleichzeitig zur Extraktion oder auch zur Filterung von den Artikelelemente gestartet.

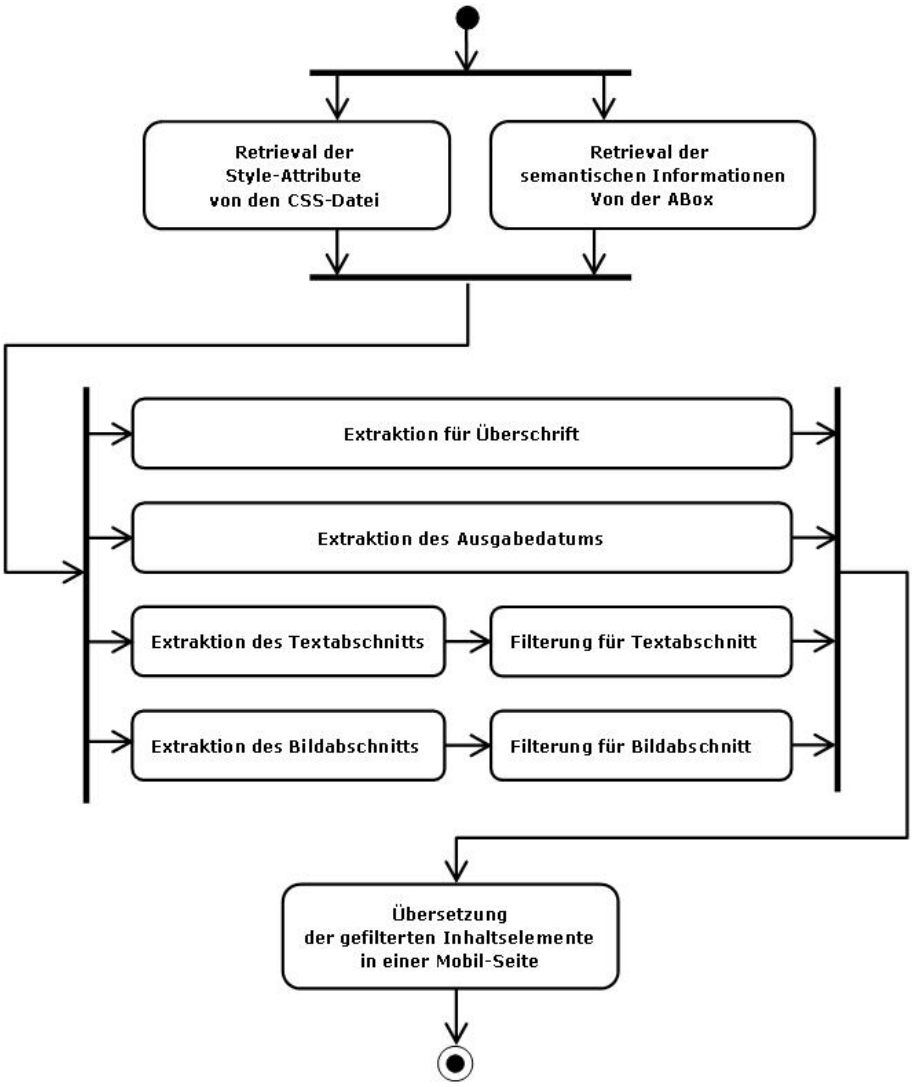

Abbildung-3.14
Aktivitätsdiagramm für die Prozessen des semantischen Filters

Die Prozesse für Extraktion der Artikelelemente wurden bereits im Abschnitt 3.4.4 mit Aktivitätsdiagramm graphisch dargestellt. Die folgenden zwei Abschnitte werden die Filterregeln für Filterung von Textabschnitt und Bildabschnitt erläutern.

3.5.1 Filterregeln für Textabschnitte

Das Grundprinzip für Filterung von Textabschnitte ist einfach, um die unwichtigen Textabschnitte zu beseitigen. Dazu müssen wir aber zunächst überlegen, wie entscheidet man die Wichtigkeit von einem Textabschnitt? Die „Wichtigkeit“ ist ein relativer Begriff, ein gleiches Textsegment hat unterschiedliche Wichtigkeiten für unterschiedliche Textleser. Deshalb sollen die Filterregeln nicht im Programm festgelegt werden, sondern außer dem Programm extern änderbar sein. Die Wichtigkeit eines Textabschnitts wird von seiner Inhaltbedeutung bestimmt, die Inhaltbedeutung wird weiter von den beinhalteten semantischen Individuen dargestellt. Das heißt, die semantische Filterung von Textabschnitt ist zur Prüfung der semantischen Individuen vom Textsegment, ob seine Bedeutungen im wichtigen semantischen Bereich befinden. Der so genannte „wichtige semantische Bereich“ wird durch Beschränkung vom Konzeptbereich der Individuen realisiert und in den externen Filterregeln definiert.

Die grundlegenden und fachlichen Begriffe, die im athletischen Bereich häufig benutzt sind, und so wie auch die dazwischen benötigten Attribute werden bereits in BOEMIE-Ontologie als Konzepte und Relationen definiert. Damit können die Webseiteninhalte semantisch in der ABox beschreiben werden. Aber nicht alle Webseiteninhalte sind von den BOEMIE-Ontologien repräsentierbar. Zum Beispiel in der Abbildung-3.5, nur die Textsegmente, die mit roten Rahmen markiert werden, können im ABox beschrieben werden. Da die Konzepte von den nicht markierten Textsegmenten noch nicht in den Ontologien definiert. Durch allmähliche Evolution und Anreicherung von Ontologien können mehr semantische Bedeutung aus den Textsegmenten identifiziert werden.

Die meisten Webseite von IAAF beinhaltet nur eine Nachricht bzw. einen Artikel, die über die sportlichen Ereignisse berichtet. Wir konzentrieren uns in der Arbeit nur auf die Artikel von den Webseiten, Da sie viele sinnvolle Bedeutungen über Athletik wie Spielername, Spielerergebnisse und Ergebnisranglist beinhalten und auch meistens von den Webseitenleser interessiert sind. Die Konzepte und Individuen, die vom Leser interessiert sind, nennen wir ***„Fokuskonzept“*** bzw. **„Fokusindividuen“** und werden in den externen Filterregeln definiert.

Eine wichtige Voraussetzung für Filterregeln ist, dass die Bedeutung der Textabschnitt nach der Filterung nicht geändert wird und auch menschlich verständlich sind. Das Wort ist eine minimale selbstständige Bedeutungseinheit vom Text, mit den Wörtern können weitere komplexe Bedeutungen wie z.B. Sätze, Paragraph aufgebaut werden. Eine Textnachricht besteht aus mindestens einem Paragraf, die mindestens mit einem Satz für eine Bedeutung berichtet. In den meisten Fälle besteht ein Artikel nicht nur aus einer Bedeutung, sondern aus mehren Bedeutungen, die aus mehren Sätze in verschiednen Paragrafen interpretiert werden. In der Arbeit wird jeder Textabschnitt als eine Bedeutungseinheit bei Filterungsprozess benutzt. Da der Zusammenhang zwischen Textabschnitt mehr schwächer als der Zusammenhang zwischen den Sätze und Wörter ist. Die Filterung von einem Textabschnitt hat relative kleinere Einwirkung auf gesamtes Inhaltverständnis. Damit können die Filterregeln für Textabschnitt so definiert werden:

wenn die Semantik eines Textabschnittes die Fokusindividuen oder die Individuen der Fokuskonzepte beinhaltet, wird der Textabschnitt geblieben, sonst weg gefiltert.

Beispielsweise für die Fokuskonzepte „PersonName“ und Fokusindividuen „IAAF World Championships“ wird der folgende erste Textabschnitt des Artikels, die in der Abbildung-3.6 gezeigt ist, nicht durch semantischen Filter weg gefiltert. Da der erste Textabschnitt das Individuum „Olimpiada Ivanova“ von den Fokuskonzepte „PersonName“ und das Fokusindividuen „„IAAF World Championships“ beinhaltet, welche im Folgenden mit der Unterstreichung kennzeichnet sind. Die zweite Textabschnitt beinhaltet nur das Individuum „Ivanova“ von den Fokuskonzepte „PersonName“ und soll weg gefiltert werden.

“ Helsinki, Finland - Olimpiada Ivanova regained her status as World champion this morning with a consummate display of race walking around the streets of Helsinki, as she set the first World record of the 10th IAAF World Championships in the 20km Race Walk. ”

“ Ivanova clocked 1:25:41, bettering Yan Wang's 1:26:22 from 2001, and so will claim a $100,000 record bonus. ”

3.5.2 Filterregeln für Bildabschnitte

Die Bedeutung der Bilder unterscheidet sich graphische Bedeutung und wörtliche Bedeutung. Es gibt die speziellen Konzepte in den BOEMIE-Ontologien zur Definition der

menschlichen Körperteile, wie zum Beispiel Kopf und Körper. Außerdem gibt es so genannt „Descriptors“ durch die Offset-Werte zur Beschreibung der graphischen Figur der Bildinhalte. Die wörtliche Bedeutung von der Bildinhalte ist durch den zugehörigen Bildtext repräsentiert, der als ein Textsegment in der ABox beschrieben wird. In der Arbeit wird nur die wörtliche Bedeutung der Bilder bei Filterungsprozess analysiert.

Ähnlich wie die Filterregeln für Textabschnitte können wir durch Definition der Fokuskonzepte und Fokusindividuen das semantische Inhaltbereich vom Bildtext beschränken. Diese Beschränkungen können nicht nur auf den Textinhalt sondern auch den Bildinhalt verwendet werden. Da im ABox außer dem Bildtext noch die Teilregion des Bilds wie z.B. Gesicht und Körper des Spielers im ABox semantisch beschrieben wurden. Die Bildabschnitte können durch Definition der Fokuskonzepte der Individuen vom Bildabschnitte gefiltert werden. Die Filterregeln für Bildabschnitte können so definiert werden:

Wenn die Semantik eines Bildabschnittes die Fokusindividuen oder die Individuen der Fokuskonzepte nicht beinhaltet, wird der Bildabschnitt weg gefiltert.

Beispielsweise für die Fokuskonzepte „PersonName“ und „PersonFace“ wird das folgende Bildabschnitt des Artikels, die in der Abbildung-3.6 gezeigt ist, nicht vom semantischen Filter weg gefiltert. Da der Bildinhalt ein Individuum des Fokuskonzept „PersonFace“ beinhaltet, welches im Bild mit roten Rahmen kennzeichnet ist, und der Bildtext ein Individuum des Fokuskonzept „PersonName“ beinhaltet, welches im Bildtext mit Unterstreichung kennzeichnet ist.

“Olimpiada Ivanova receives a 100,000 dollars World record bonus cheque (Getty Images)“

Die Bilder, die nach den Filterregeln ausgefüllt sind, müssen noch komprimiert werden, um den Bildschirm des Mobiltelefons anpassen zu können. Der Prozess zur Komprimierung der Bilder wird bei Generierung der mobilen Webseiten implementiert.

3.6 Übersetzungsprozess

Nachdem alle Artikelelemente gemäß den Filterregeln gefiltert worden sind, können die gefilterten Artikelelemente in der mobilen Webseite übersetzt werden. In der Arbeit benutzen wir die WML-Seiten als die endlichen mobilen Webseiten. Der Prozess zur Generierung der WML-Seiten wird in der Komponente WML-Page-Generator aufgebaut. In diesem Unterkapitel stellen wir die Methode für Generierung der MWL-Seiten vor.

3.6.1 Vorverarbeitung des Seiteninhalts

Bevor Generierung der WML-Seite sollen alle Inhaltkomponenten der WML-Seite gemäß ihrer Auftrittsordnung von der originalen Webseiten angeordnet und auch zur Anpassung von Bildschirm des Mobiltelefons erneut formatiert werden. Bei Extraktion der Artikelelemente wurden bereits die Offset-Koordinaten von jedem Artikelelement berechnet, um den Startpunkt und Endpunkt jedes Artikelelements zu notieren. Gemäß den Offset-Koordinaten können alle Artikelelemente erneut in der WML-Seite anordnen, damit die richtige Reihenfolge beim Lesen einzuhalten. Außerdem müssen auch alle großen Bilder gemäß der Abmessung des Mobiltelefon-Bildschirms komprimiert werden. in der Arbeit wird eine festgelegte Bildschirmabmessung mit der Auflösung 320x240 Pixel zur Generierung der WML-Seiten verwendet.

3.6.2 Schreiben des gefilterten Inhalte in WML-Karte

Die oben erwähnten gefilterten Artikelelemente werden weiter in einer WML-Seite übersetzt. Die Grundkenntnisse über die WML-Seite wurden bereits im Abschnitt 2.3.1 vorgestellt. Eine WML-Seite ist ein XML-basiertes Dokument. Ein Beispiel für WML-Seite wird im Code-3.2 gezeigt, die nur eine WML-Karte beinhaltet und aus zwei Teile besteht, ein Teil ist eine Deklaration des XML-Dokuments (Reihe1-2) und anderer Teil ist den Inhaltkörper (Reihe3-11). Der Inhaltkörper der WML-Seite wird durch eine Reihe von der so genannte „Karte“ aufgebaut. Alle Inhalte der WML-Seite können in mehreren Karten verteilt werden. Im Bildschirm des Mobiltelefons darf gleichzeitig nur der Inhalt von einer Karte dargestellt werden. Das Durchblättern zwischen den Karten wird durch die Tastatur

des Mobiltelefons realisiert. In der Arbeit wird nur eine Karte für eine WML-Seite verwendet werden.

In die WML-Karte werden verschiedene WML-Tags für Darstellung jeder Artikelelemente verwendet. Ähnlich wie HTML ist der p-Tag zur Darstellung eines Textabschnitts verwendet und der img-Tag ist zur Darstellung eines Bildes verwendet. Außerdem werden auch die anderen WML-Tags wie big-Tag, b-Tag, u-Tag, i-Tag für die Überschrift (Reihe5 vom Code-3.2) und Ausgabezeit (Reihe6 vom Code-9) zum Unterschied von den Textsegmenten benutzt. Die Angaben für die WML-Tags können Sie im Abschnitt 2.3.1 finden. Die generierte WML-Seiten und auch sowie die zugehörigen komprimierte Bilder werden lokal gespeichert für weiteren Prozess zur Verfügung gestanden.

```
1.  <?xml version="1.0"?>
2.  <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN
    "http://www.wapforum.org/DTD/wml_1.1.xml">
3.  <wml>
4.  <card id="page-1" title="sample page">
5.  <p><big><b><u>Überschrift</u></b></big></p>
6.  <p><i>Ausgabezeit</i></p>
7.  <p>Textabschnitt</p>
8.  <p><img src="sample.png" alt="image1"/><br/>
9.  <small>Bildtext</small></p>
10. </card>
11. </wml>
```

Code-3.2: Beispiel für eine einfache WML-seite mit einer Card zur Aufbau eines Artikels mit vier Artikelelemente.

4 Implementierung

Dieses Kapitel behandelt den praktischen Teil der Diplomarbeit, um einen Prototyp des semantischen Filters gemäß der Konzeption vom Kapitel 3 aufzubauen, damit die Funktionalität und Durchführbarkeit des semantischen Filters untersucht werden können. Der Prototyp des semantischen Filters wird in der Arbeit als eines Java-Programm in NetBeans-IDE entwickelt und durch eine graphische Benutzeroberfläche(GUI) gesteuert. Es gebt drei Teile in diesem Kapitel, im Unterkapitel 4.1 wird zunächst die Benutzeroberfläche des Prototyps vorgestellt. Im Unterkapitel 4.2 wird die zentralen Prozessbauteile und Prozessprinzip des Prototyps vorgestellt. Im Unterkapiteln 4.3-4.5 werden separat drei wichtigen Prozessphasen des Prototyps vorgestellt.

4.1 Benutzeroberfläche

Die Benutzeroberfläche des Prototyps bietet folgende Funktionen an:

- Auswahl der zu filternden Webseite-Datei, die im BOEMIE-Projekt als Beispiele verwendet wurde.
- Auswahl der entsprechenden OWL-Datei, die als ABox der Webseiten bereits vom BOEMIE-Projekt erzeugt wurde.
- Konfiguration von RacerPro, wie Adresse und Port.
- Starten der Filterprozesse.
- Anzeige der Quellecode der endlich generierten mobilen Webseite.

Damit die Benutzeroberfläche und die Hintergrundprozesse in der Zukunft unabhängig geändert oder ausgetauscht werden zu können wird das MVC(Model View Controller)-Entwurfsmuster in der Arbeit verwendet. Die grundlegende Architektur vom MVC-Entwurfsmuster ist wie das Klassendiagramm von der Abbildung-4.1 gezeigt. Gemäß der MVC-Architektur werden alle Ereignisse, die von dem Benutzer auf die Benutzerfläche ausgelöst werden, nicht direkt in der Benutzeroberfläche sondern in anderer Steuerungskomponente bearbeitet, welche in der Arbeit „GUI-Controller“ genannt wird. Der GUI-Controller übernimmt die Benutzerereignisse und verteilt die Aufgaben in den Hintergrundprozessen zur bearbeiten und danach schickt die Ergebnisse wieder in der Benutzeroberfläche zu darstellen. Die konkreten Algorithmen zur Bearbeitung der Benutzdaten werden nicht in dem GUI-Controller aufgebaut, er bietet nur die Schnittstelle zur Aufgabenverteilung. Die Benutzeroberfläche und die Hintergrundprozesse werden durch den GUI-Controller entkoppelt, d.h. die Benutzeroberfläche und Hintergrundprozesse wissen nichts voneinander. Die Verwendung

von der andren Seite müssen durch den GUI-Controller implementiert werden. Im Unterkapitel 4.1.1 und 4.1.2 wird vorstellt, wie die GUIView-Klasse und GUIController-Klasse nach der MVC-Architektur aufgebaut wird.

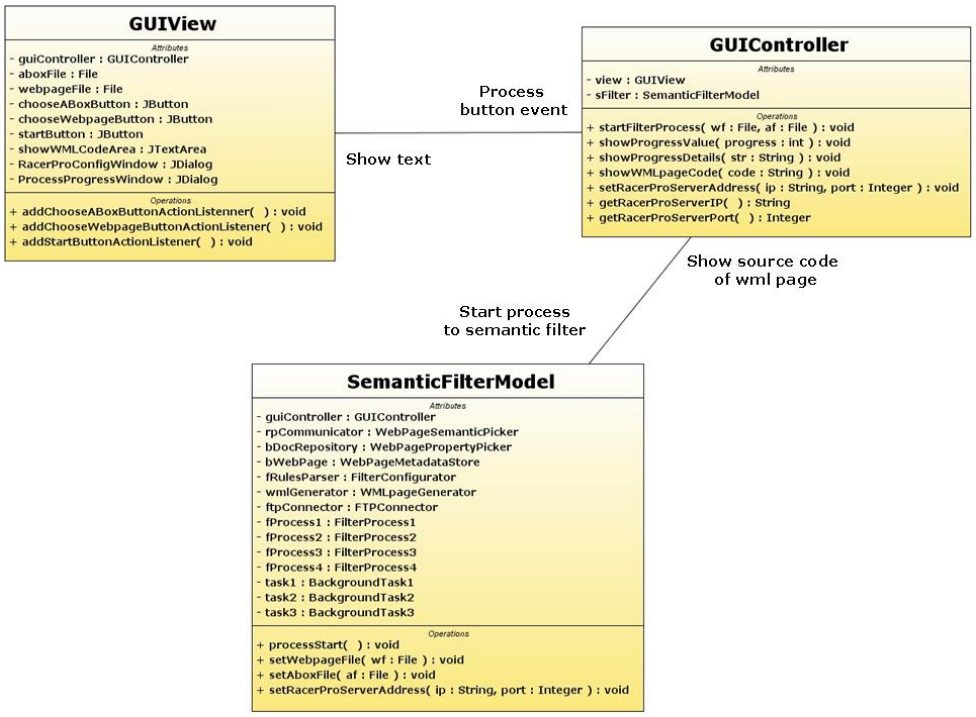

Abbildung-4.1
Klassendiagramm für Benutzeroberfläche mit Anwendung von MVC-Architektur
(-: privat, +: public, #: protected)

4.1.1 GUIview-Klasse

Die GUIView-Klasse wird zur Aufbau und Darstellung der visuellen Komponenten verwendet. Die NetBeans-IDE bietet eine visuelle Java-Entwicklungsplattform zur Aufbau und Anordnung der GUI-Komponenten. Die Inhalte über die visuelle Programmierung mit NetBeans-IDE werden hierbei nicht als Schwertpunkt weiter vorgestellt, die Einzelheit konsultieren Sie bitte das entsprechende Handbuch. In der GUIView-Klasse vom Klassendiagramm im Abbildung-4.1 werden die führenden Bauteile und Methoden gezeigt. Durch Klick der Knopfe „chooseWebpageButton“ und „chooseABoxButton“ können die zur semantischen Filterung benutzten Webseite-Datei und die entsprechende ABox-Datei von lokalem Computer ausgewählt werden und dann Klick des Knopfs „startBotton“ wird die semantische Filterprozessen gestartet. Die Aktion/Ereignis jedes Knopfs werden durch die entsprechenden Klassen vom Klassenpacket java.awt.event abgehört und bearbeitet. Die Endergebnisse bzw. die Quellecode der WML-seiten werden durch das Anzeigenfeld

„showWMLCodeArea“ aufgezeigt. Die Abbildung-4.2 zeigt die von GUIView-Klasse dargestellte Benutzeroberfläche.

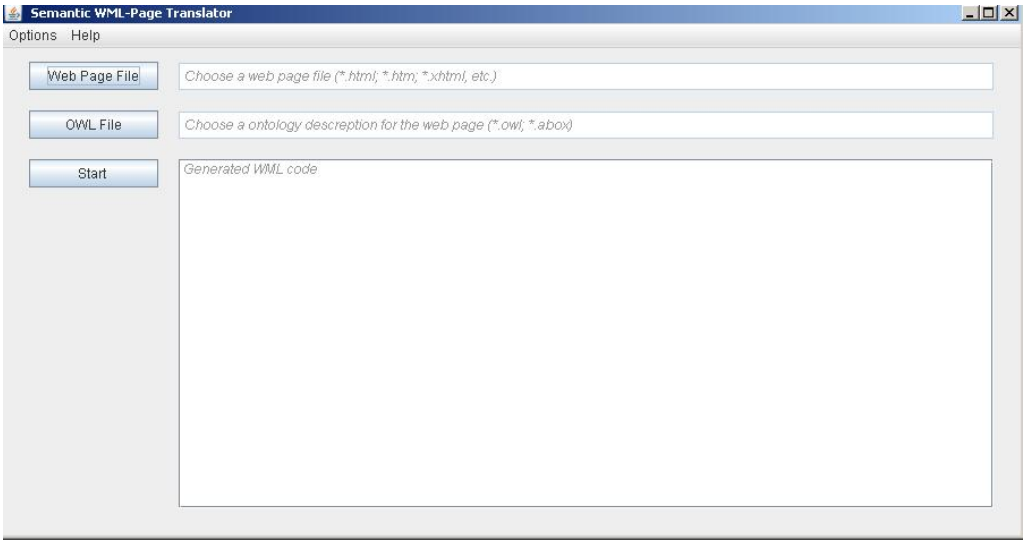

Abbildung-4.2
Benutzeroberfläche des Prototyps

Weitere Funktionen können im Options-Menü von der Benutzeroberfläche hinzugefügt werden. Im Prototyp wird Die Konfiguration von RacerPro durch ein separates Konfigurationsfenster implementiert, das als eine Funktion im options-Menü integriert worden ist. Das RacerPro wurde bereits in lokalem Betriebssystem installiert und ist mit der IP-Adresse 127.0.0.1 und Default-Port 8088 zugängig. Die Konfigurationsfester ist wie Abbildung-4.3 dargestellt, mit dem der Benutzer die Adresse und den Port von RacerPro überprüfen oder erneut einstellen kann.

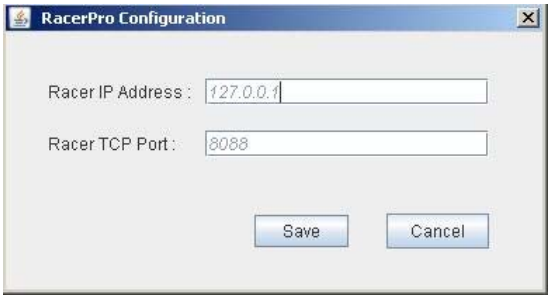

Abbildung-4.3
Konfigurationsfenster für RacerPro

Nachdem *startButton* geklickt wurde, wird der semantische Filterprozess gestartet, zugleich wird noch ein separates Fenster ausgelöst, um den zeitliche Programmverlauf zu zeigen. Das Fenster für Programmverlauf wie Abbildung-4.4 dargestellt.

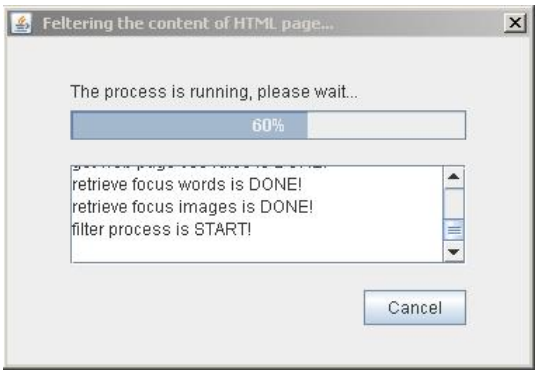

Abbildung-4.4
Fenster zur Überwachung des Prozesslaufs

4.1.2 GUIController-Klasse

Bei Klick auf *startButton* wird nicht direkt die im Hintergrund stehenden Filterprozessen aufgerufen, sondern die öffentliche Methode von der GUIController-Klasse aufgerufen, um das gefangene Ereignis zu bearbeiten. Diese MVC-Architektur dient dazu, dass die Benutzeroberfläche ohne Kenntnisse der Hintergrundprozesse ausgeführt werden zu können. Die Verwaltungen für die Hintergrundprozesse und die Benutzeroberfläche werden durch GUIController-Klasse implementiert. Die Endergebnisse von den Filterprozessen werden auch durch GUIController-Klasse in der Benutzeroberfläche angezeigt. Dazu wissen die Hintergrundprozesse auch nicht, wo und wie die Endergebnisse bearbeitet werden. Damit sind die Benutzeroberfläche und Hintergrundprozesse komplett entkoppelt und jede Seite kann unabhängig geändert und umgetauscht werden.

Im Klassendiagramm von Abbildung-4.1 werden zwei private Objekte von GUIView-Klasse und SemanticFilterModel-Klasse definiert, um die öffentlichen Methode von den beiden Klassen aufzurufen. Jede öffentliche Methode von GUIController bildet eine Funktion von der Benutzeroberfläche ab.

Wenn der Knopf *startButton* vom Benutzer gedrückt würde, würde die GUIView-Klasse die Methode *startFilterProcess()* von GUIController-Klasse aufrufen. Die Methodenaufrufe innerhalb *startFilterProcess()* werden durch das Sequenzdiagramm von der Abbildung-4.5 illustriert. Die öffentlichen Methoden *setWebpageFile(wf:File):void* und *setABoxFile(af:file):void* von SemanticFilterModel-Klasse werden zuerst aufgerufen, um die Webseite-Datei und ABox-Datei auszuwählen. Danach starten die Filterprozessen. Der Prozessverlauf, Zwischeninformationen und Endergebnisse von der *SemanticFilterModell-Klasse* werden durch Aufrufe die Methoden *showProgressDetails(str:String):void*,

showPrograssValue (progress:int):void und *showWMLCode(code:String):void* in der Benutzeroberfläche heraus gezeigt.

Abbildung-4.5
Sequenzdiagramm für Methodenaufrufe in MVC-Architektur

4.2 Semantischer-Filter-Modell

Die SemanticFilterModel-Klasse ist der zentrale Bestandteil vom Prototyp des semantischen Filters, um alle in den Filterprozessen verwendete Bestandteile zu verwalten und auch den Prozessverlauf zu kontrollieren. Es gibt vier führenden öffentlichen Methoden in der SemanticFilterModel-Klasse:

- **setRacerProServerAddress (ip:String, port:Integer):void** Diese Mothode wird zur Konfiguration von IP-Adresse und Port von RacerPro verwendet.
- **setWebpageFile (wf :File) :void**
  Diese Methode wird zum Einsatz der Webseite-Datei verwendet.

- **setAboxFile (af :File) :void**
  Diese Methode wird zum Aufnahme der ABox-Datei von der GUI verwendet.
- **processStart () :void**
  Diese Methode wird zum Starten der Filterprozessen verwendet.

Die Funktionalitäten von diesen vier Methoden werden durch die Benutzeroberfläche repräsentiert. Wie in Abbildung-4.2 und Abbildung-4.3 gezeigt wurde. Die Methodenaufrufe werden durch GUIController-Klasse implementiert. Der komplette Programmcode für Filterprozesse ist tatsächlich nicht direkt in der Methode *processStart()* aufgebaut, sondern in drei unterschiedlichen Phasen verteilt.

4.2.1 Hintergrundprozesse für semantische Filterung

Die Filterprozesse werden in folgenden drei Phasen verteilt,

- **Information-Retrieval-Phase：** Diese Phase ist die erste Phase von den Filterprozessen und unterscheidet sich weiter in Retrieval der Individuen von ABox-Datei und Retrieval der Attributen von CSS-Datei. Diese Phase bereitet die notwendigen semantischen und nicht-semantischen Metadaten der Webseite für nächste Extraktion-Filterung-Phase vor.

- **Extraktion-Filterung-Phase：** Diese Phase ist die zweite Phase von den Filterprozessen und hat zwei Aufgaben. Erstens, Extraktion aller Artikelelemente von der Webseite gemäß der von der ersten Phase bekommende Metadaten. Zweitens, Filterung der Artikelelemente gemäß der extern definierten Filterregeln.

- **Generierung-Freigabe-Phase：** Diese Phase ist die dritte Phase von den Filterprozessen. In der Phase werden die gefilterten Artikelelemente in eine WML-Seite übersetzt. Die generierte WML-Seite wird dann im lokalen System gespeichert und auch zugleich nach entferntem Server hochgeladen.

Die drei Prozessphasen sind voneinander abhängig. Zur Ausführung einer Phase wird das Ergebnis von der vorherigen Phase benötigt. Aber das Problem ist, in welchem Zeitpunkt ist das Ergebnis von der vorherigen Phase erhältlich? Außerdem gibt es noch anderes Problem: der Arbeitsthread von der GUI wird gesperrt, während die im Hintergrund legende Filterprozessen ausgeführt sind. Wie die Abbidlung-4.6(a) gezeigt. Da ein Swing-Programm normalerweise nur ein Event-Dispatching-Thread(EDT) hat, das für Darstellung und Aktualisierung der GUI verantwortet. Um die oben erwähnten Probleme

zu lösen wird die SwingWorker-Klasse in der Arbeit benutzt. Die SwingWorker-Klasse was erst in Java-SE6 veröffentlicht und kann das EDT sofort freigegeben und erzeugt ein asynchrones Thread, um den Prozess im Hintergrund auszuführen, wie Abbildung-4.6(b) gezeigt wird.

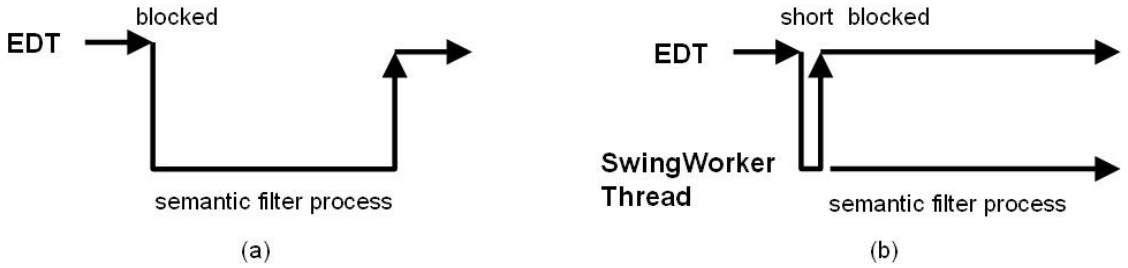

Abbildung-4.6
(a): Swing Programm ohne Anwendung von SwingWorker[JOC07]
(b): Swing Programm mit Anwendung von Swingworker[JOC07]

Mit Verwendung der SwingWorker-Klasse wird nicht nur die Reaktion von der Benutzeroberfläche mehr schneller, können auch die Benutzdaten in und nach dem Thread-Verlauf einfach bearbeitet werden. Im Folgenden werden die wichtigen Methoden von SwingWorker-Klasse vorgestellt.

- **Execute():** Diese Methode erzeugt eine Hintergrundthread zur Ausführung der Methode *doInBackground()*, ähnlich wie die Methode *start()* von Thread-Klasse.
- **doInBackground():** In dieser Methode werden die Algorithmen vom Hintergrundprogramm aufgebaut, ähnlich wie die Methode *run()* von der Thread-Klasse.
- **publish(V… chunks):** Die Methode sendet die Zwischenergebnisse nach die Methode *progress(List<V> chunks)* zu bearbeiten.
- **process(List<V> chunks):** Die Methode bearbeitet die Zwischenergebnisse vom Hintergrundprogramm.
- **done():** Die Methode wird nach der Ausführung von *doInBackground()* automatisch aufgerufen. in dieser Methode werden häufig die Algorithmen zu Bearbeitung des Endergebnisses aufgebaut.
- **get():** Die Methode wird in der Methode *done()* verwendet, um das Engergebnis von *doInBackground()* zu erhalten.

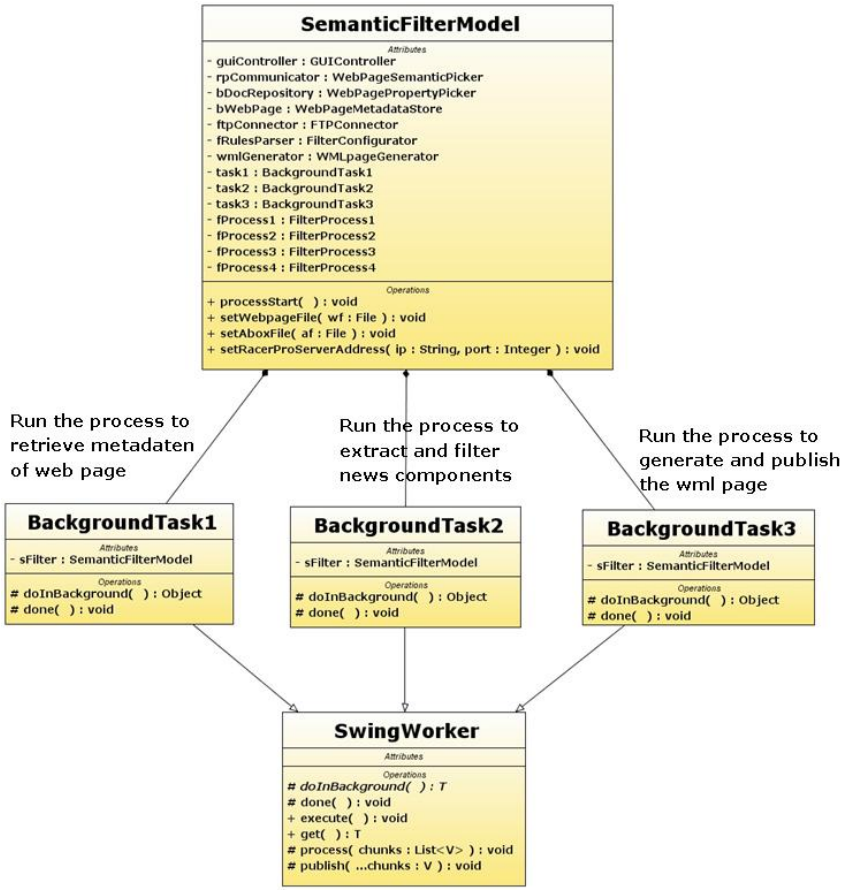

Abbildung-4.7
Klassendiagramm für Hintergrundprozessen mit Anwendung von SwingWorker-Klasse
(-: privat, +: public, #: protected)

In den Filterprozessen werden drei Unterklassen von *SwingWorker* (*BackgroundTask1, BackgroundTask2, BackgroundTask3*) definiert, um die drei Prozessphasen zu implementieren. Wie in der Abbildung-4.7 gezeigt wird, erben die drei Unterklassen zwei Methoden *doInBackground()* und *done()* von ihrer Oberklasse *SwingWorker* zur Ausführung der Filterprozess und Bearbeitung der Ergebnisse. Alle drei Unterklassen der SwingWorker-Klasse werden als die verschachtelten Klassen oder sogenannte inneren Klassen in SemanticFilterModel-Klasse konstruiert. Der Algorithmus jeder Prozessphase wird in späteren Unterkapiteln weiter eingehend vermittelt.

4.3 Prozessphase I – Retrieval der Information

Wie im letzten Unterkapitel bereits erwähnt wurde, gibt es drei Prozessphasen in semantischen Filterprozessen. Dieses Kapitel behandelt die erste Prozessphase, um die semantischen Individuen von der ABox und Attributwerte von der CSS-Datei zu gewinnen. Im Abschnitt 4.3.1 werden führend in der ersten Prozessphase benutzten Klassen

vorgestellt. Im Abschnitt 4.3.2 werden die Einzelheiten des Klassenaufbaus für Gewinnung der semantischen Individuen von ABox-Datei erläutert. Im Abschnitt 4.3.3 werden die Einzelheiten der Klassenaufbau für Gewinnung der Attribute der CSS-Regeln von CSS-Datei erläutert.

4.3.1 Klassen von der Prozessphase I

Der Algorithmus von der ersten Prozessphase wird in der BackgroundTask1-Klasse aufgebaut. Es geht um drei Klassen (in Abbildung-4.8 gelb hervorgehoben) und drei Datenstrukturen (blaue hervorgehoben). Die Funktionen von den Klassen und Datenstrukturen werden wie unten angegeben.

- **WebPageSemanticPicker-Klasse:** Diese Klasse ist zur Wiedergewinnung der semantischen Individuen von der ABox-Datei durch Sendung der Abfragen nach RacerPro verwendet. Sie bietet BackgroundTast1-Klasse in der ersten Prozessphase vielen verschiedenen öffentlichen Methoden zur Gewinnung der verschiedenen semantischen Individuen von der ABox-Datei.
- **IndvdTextSeg-Datenstruktur:** Diese Datenstruktur ist zur Speicherung der von ABox-Datei wieder gewonnene Text-Individuen sowie auch seine Attributwerte wie ID, Konzept und Offset-Koordinaten verwendet.
- **IndvdCaptImg-Datenstruktur:** Diese Datenstruktur ist zur Speicherung der von ABox-Datei wieder gewonnene Bild-Individuen sowie auch seine Attributwerte wie ID, Konzept und Offset-Koordinaten verwendet.
- **WebPageProtertyPicker-Klasse:** Diese Klasse ist zur Wiedergewinnung der Attribute von der Webseite und CSS-Regeln verwendet. Sie bietet BackgroundTast1-Klasse in der ersten Prozessphase verschiedenen öffentlichen Methoden zur Gewinnung der verschieden Attribute von Webseite und CSS-Regeln.
- **CssRuleItem-Datenstruktur:** Diese Datenstruktur ist zur Speicherung der CSS-Regeln verwendet. In der Arbeit werden nur 4 Attribute für Selector-Name, Selector-Type, Font-Size und Font-Color in dieser Datenstruktur definiert.
- **WebPageMetadataStore-Klass:** Diese Klasse verwendet die obere drei Datenstrukturen zur Sammelung und Verwaltung aller semantischen und nicht-semantischen Metadaten von der Webseite. Sie bietet außerdem alle anderen in der Filterprozessen benötigen Metadaten durch Analysis des Quellecode der Webseite. Diese Klasse wird als einen Träger zum Datenaustausch während dem Prozessverlauf verwendet. Die gefilterte mobile Webseite wird vor Freigabe auch in dieser Klasse gespeichert.

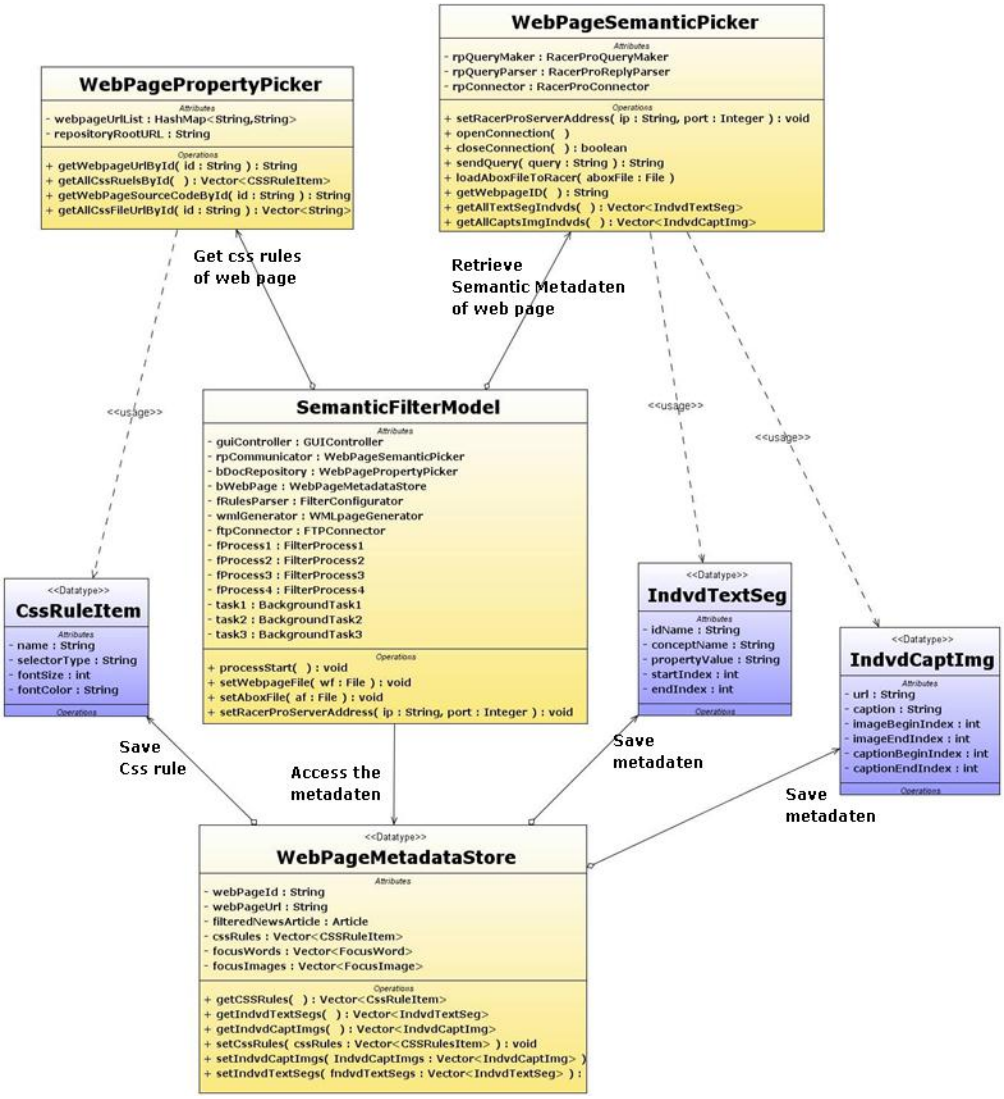

Abbildung-4.8
Klassendiagramm für ersten Prozessphase der semantischen Filterung
(gelb: Klasse, Blau: Datenstruktur, -: privat, +: public, #: protected)

4.3.2 Wiedergewinnung der Individuen vom ABox

Aufgrund des Komponentendiagramms von Abbildung-3.7 im Abschnitt 3.3.3 sind die Algorithmen zur Wiedergewinnung der semantischen Individuen vom ABox tatsächlich nicht komplett in der *WebPageSemanticRetrieval-Klasse*, sondern noch mit anderen drei Hilfeklassen zusammen implementiert. Die drei Hilfeklassen werden *RacerProQueryMaker-Klasse*, *RacerProConnector-Klasse* und *RacerProReplyParser-Klasse* genannt und speziell für Erzeugung der Abfragen, Sendung der Abfragen und Analysis der Antwort verwendet. Wie Abbildung-4.9 gezeigt wird, werden drei Objekte von den drei Hilfeklassen als private Attribute der WebPageSemanticRetrieval-Klasse definiert, die

Funktionalitäten von den drei Hilfsklassen sind nur in der WebPageSemanticPicker-Klasse gültig.

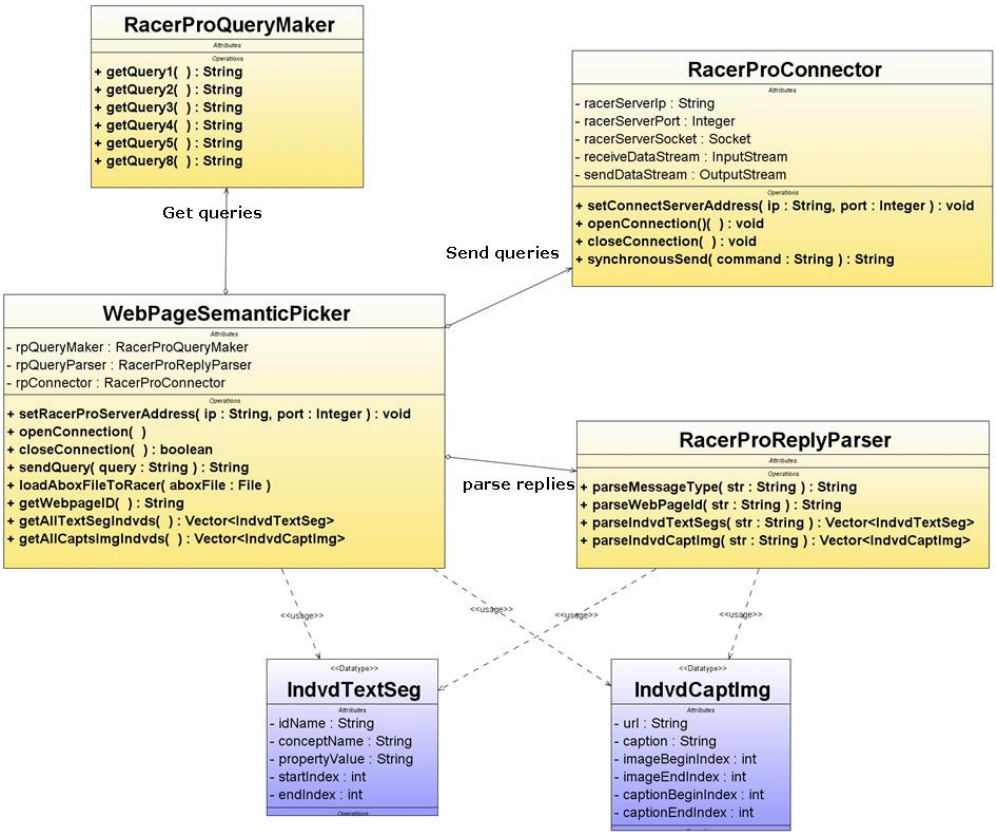

Abbildung-4.9
Klassendiagramm für Retrieval von semantischen Metadaten
(gelb: Klasse, Blau: Datenstruktur, -: privat, +: public, #: protected)

Folgendes Sequenzdiagramm von Abbildung-4.10 zeigt, wie die öffentliche Methode *getAllTextSegIndvds()* der WebPageSemanticPicker-Klasse von der BackgroundTask1-Klasse abgerufen wird, um die Text-Individuen von ABox-Datei wieder zu gewinnen.

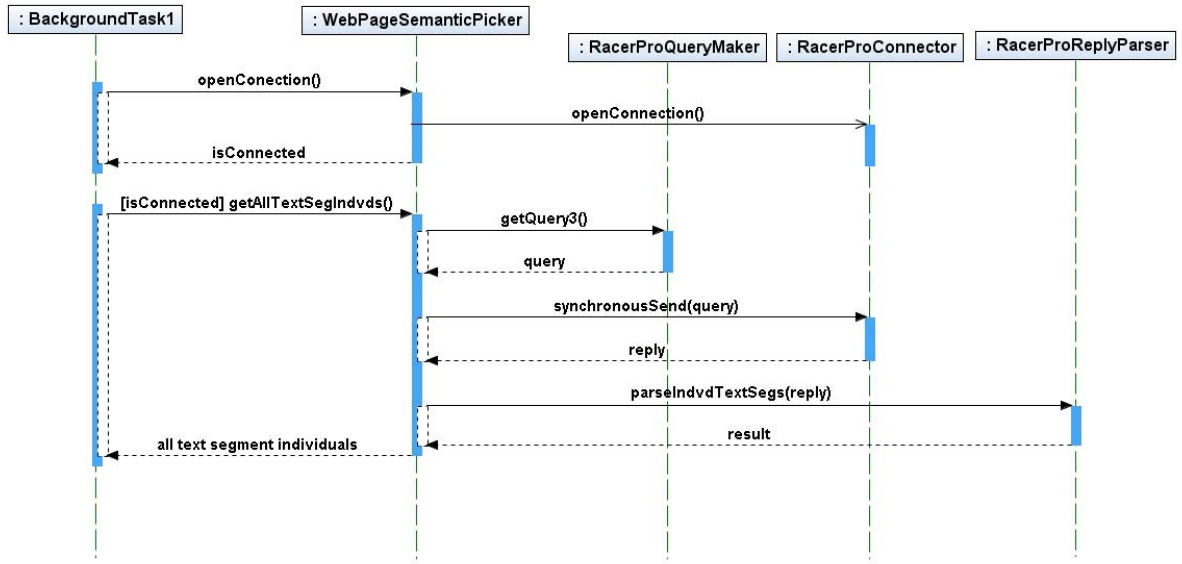

Abbildung-4.10
Sequenzdiagramm für Retrieval der Text-Individuen

4.3.3 Widergewinnung der CSS-Regeln von CSS-Datei

Die Wiedergewinnung der CSS-Attribute ist durch WebPagePropertyRetrieval-Klasse realisiert. Die CSS-Regeln sind in CSS-Datei definiert und durch LINK-Tag in HTML-Quellecode wirksam. Z.B. in dem Code-4.1 wird die CSS-Datei „sample.css“ eingesetzt.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<link rel="stylesheet" type="text/css" href="../css/sample.css" />
</head>
<body>...</body>
</html>
```

Code-4.1: Beispiel für Verwendung der CSS-Datei in HTML-Quellcode

Jede CSS-Regel wird von einem Selektor kennzeichnet. Der Selektor repräsentiert eine Struktur, die als die Bedingungen zur Anwendung der CSS-Regeln benutzt wird Wenn alle Bedingungen eines Selektor für ein bestimmtes Element zutreffen, stimmt der Selektor mit dem Element überein, und die Regeln können auf das Element angewandt werden [SELEK09]. Der Selektor entscheidet sich in Typ-Selektor, Universal-Selektor, ID-Selektor, Klasse-Selektor oder Pseudo-Selektor. Die Anwendung für jeden Typ wenden Sie bitte an das technische Dokument von W3C im Angang-A. Der Code-4.2 zeigte ein Beispiel, in dem vier Artikelbestandteile Überschrift, Ausgabezeit, Textsegment und

Bildsegment mit vier DIV-Tags ausgezeichnet und die ID-Selektor und Klasse-Selektor angewendet werden.

```
<BODY>
<DIV class="news">
<DIV id="titel">IAAF News</DIV>
<DIV id="updateTime">04.07.2009</DIV>
<DIV class="contenText"><p>...</p> </DIV>
<DIV class="contenImage"><img.../></DIV>
</DIV>
</BODY>
```

Code-4.2: Markierung der HTML-Tags mit Anwendung der Selektor ID oder CLASS

im Code-4.3 werden die zugehörigen CSS-Regeln für DIV-Tags, die im Code--4.2 gezeigt wurden, definiert. In der geschweiften Klammer sind die Attribute der CSS-Regel , der Name vor der geschweiften Klammer ist der Selector. Ein ID-Selektor wird mit dem Präfix „#“ kennzeichnet und seine CSS-Regeln darf nur in einem HTM-Tag angewendet werden. Ein Klasse-Selektor wird mit dem Präfix „.“ kennzeichnet und seine CSS-Regeln darf mehrfach in verschiedene HTM-Tags angewendet werden.

```
BODY .news { width: 980px; … }
.news #titel{ font-size: 12px; color: #333333; }
.news #updateTime{ font-size: 9px; color: #000000; }
.news .contentText{ position: relative; … }
.news .contentImage{ widht: 980px; heigh: 180px; }
```

Code-4.3: Beispiel der CSS-Regeln für unterschiedlichen Selektors

In der WebPagePropertyPicker-Klasse gibt es entsprechende Methoden gemäß die Webseite-ID zur Erhalt die absolute Adresse von Webseite und CSS-Datei sowie auch die Methoden zur Parsen die Attribute von CSS-Datei. Wie im Klassendiagramm von der Abbildung-4.11 gezeigt werden, können alle CSS-Regeln durch die öffentliche Methode *getAllCssRulesById(id:String):Vector<CssRuleItem>* erhalten werden. Die CSS-Attribute sowie auch die Bedingungen und Typ des Selektors werden in der CSSRuleItem-Datenstruktur gespeichert und als Rückgabewerte von der Methode zurückgegeben. Der Quellcode von der Webseite und CSS-Datei wird durch die Hilfeklasse FileWRTools gewonnen.

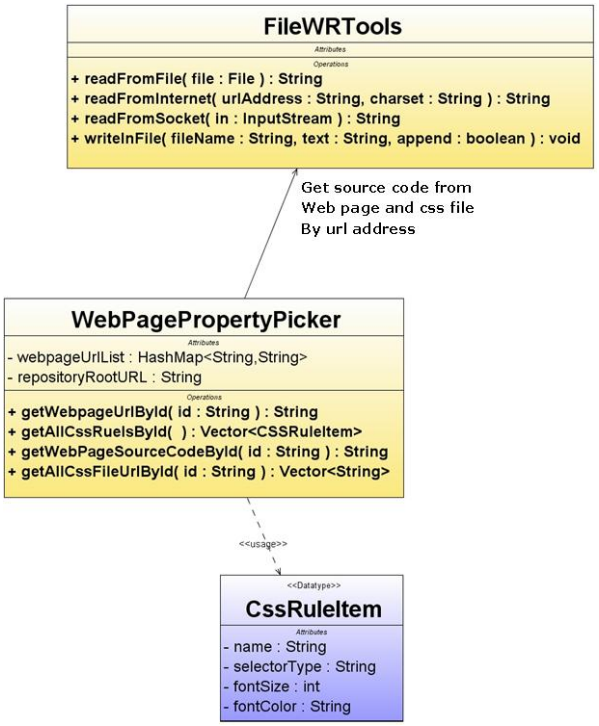

Abbildung-4.11
Klassendiagramm für Retrieval der von CSS-Regeln
(gelb: Klasse, Blau: Datenstruktur, -: privat, +: public, #: protected)

4.4 Prozessphase II – Extraktion und Filterung

Dieses Unterkapitel behandelt die zweite Prozessphase, die Artikelelemente von HTML-Quellcode heraus zu extrahieren und dann gemäß den Filterregeln die Inhalte der Artikelelemente zu filtern. Im Abschnitt 4.4.1 werden die Klassen und Prozessmethoden von der zweiten Prozessphase vorgestellt. Im Abschnitt 4.4.2 werden die extern definierten Filterregeln vermittelt.

4.4.1 Klassen und Prozessmethoden in der Prozessphase II

Nachdem alle Algorithmen von der Methode *doInBackground()* in der innere BackgroundTask1-Klasse fertig ausgeführt wird, werden alle von ABox-Datei und CSS Datei gewonnene semantische und nicht-semantische Metadaten in der WebPageMetadataStore-Klasse gespeichert und gleichzeitig wird die Methode *done()* von der BackgroundTask1-Klasse als nächsten Schritt ausgeführt. In der Methode *done()* wird die zweite Prozessphase durch Aufruf der Methode *start()* in der inneren BackgroundTask2-Klasse gestartet. Die zweite Prozessphase bezieht sich auf fünf Klassen

(in Abbildung-4.12 gelb hervorgehoben Klassendarstellung in UML-Syntax) und zwei Datenstrukturen (in Abbildung-4.12 blaue hervorgehoben).

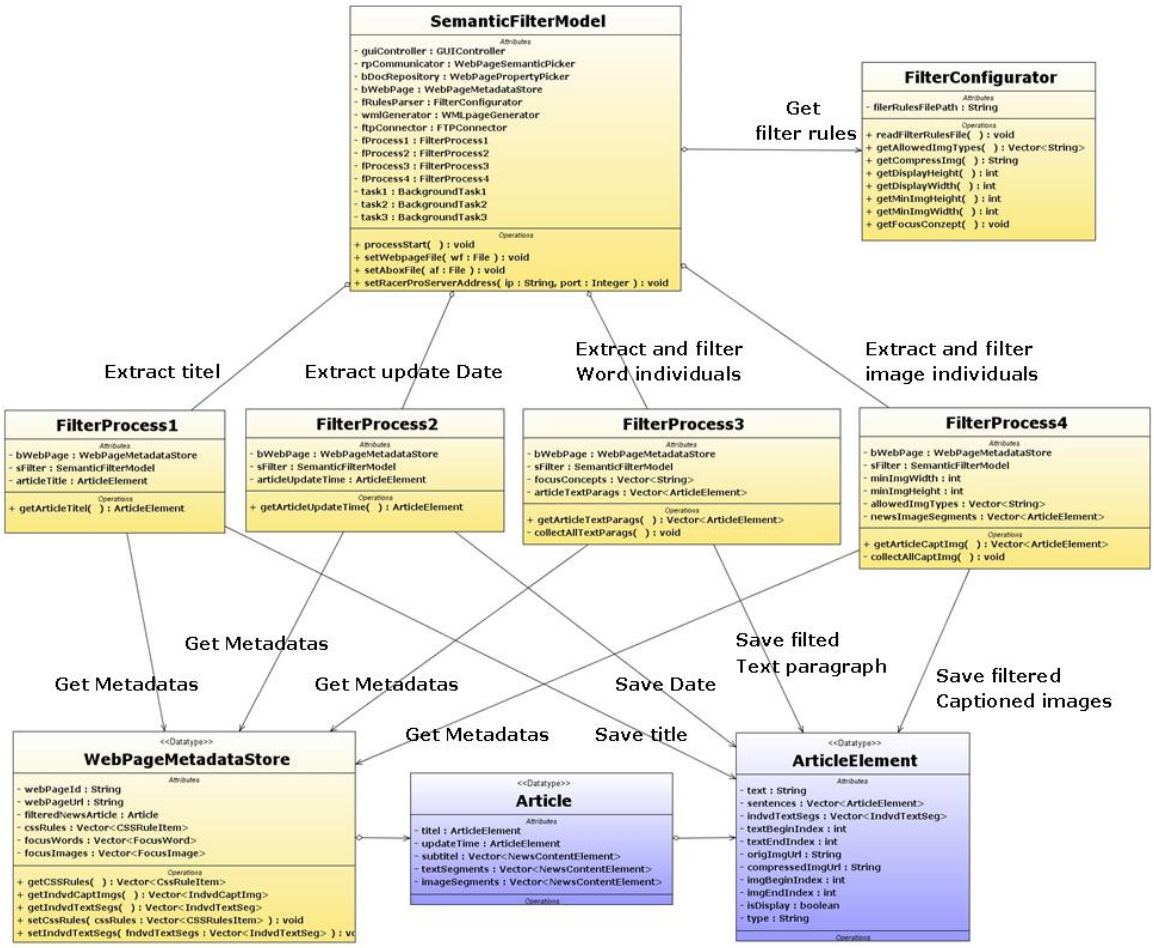

Abbildung-4.12
Klassendiagramm für Filterung der Artikelelemente
(gelb: Klasse, Blau: Datenstruktur, -: privat, +: public, #: protected)

Die Funktionen von oben dargestellte Klassen und Datenstrukturen werden wie unten geschrieben.

- **FilterConfigurator-Klasse:** Diese Klasse ist zur Erhalt der extern definierten Programmparametern und Filterregeln verwendet.
- **FilterProcess1-Klasse:** Eine Unterklasse von Thread-Klasse, in der die Algorithmen zur Extraktion von Überschrift des Artikels aufgebaut werden.
- **FilterProcess2-Klasse:** Eine Unterklasse von Thread-Klasse, in der die Algorithmen zur Extraktion von Ausgabezeit des Artikels aufgebaut werden
- **FilterProcess3-Klasse:** Eine Unterklasse von Thread-Klasse, in der die Algorithmen zur Extraktion und Filterung von Textsegmenten des Artikels aufgebaut werden.

- **FilterProcess4-Klasse:** Eine Unterklasse von Thread-Klasse, in der die Algorithmen zur Extraktion und Filterung von Bildsegmenten des Artikels aufgebaut werden.
- **ArticleElement-Datenstruktur:** In dieser Datenstruktur werden nicht nur die Inhalte jedes Artikelelemente wie Text und Bild-Adresse sondern auch die semantische Individuen sowie die entsprechende Offset-Koordinaten des Artikelelemente, die im ABox beschrieben wurden, gespeichert
- **Article-Datensstruktur:** In dieser Datenstruktur werden die vollständige Artikelinhalte inklusive alle Artikelelemente, welche die Objekte von ArticletElement-Klasse sind, gespeichert.

Die mehr Einzelheiten über die SemanticFilterModoel-Klasse und WebPageMetadataStore-Klasse wurden bereits im Unterkapitel 4.2 und Unterkapitel 4.3 geschrieben.

Wie im Abschnitt 3.4.5 erwähnt wurden, werden die vier Artikelelemente Überschrift, Ausgabezeit, Textsegment und Bildsegment durch Ausführung der vier verschiedenen Threads extrahiert und gefiltert. Die Algorithmen von den vier Threads werden in vier Unterklassen FilterProcess1, FilterProcess2, FilterProcess3, FilterProcess4, welche die Runnable-Interface implementiert, konstruiert und dann durch vier Thread-Objekte gleichzeitig aufgerufen, wie unter Code-4.4 gezeigt wird.

```
Thread process1 = new Thread(fProcess1);
Thread process2 = new Thread(fProcess2);
Thread process3 = new Thread(fProcess3);
Thread process4 = new Thread(fProcess4);
process1.start();
process2.start();
process3.start();
process4.start();
```

Code-4.4: Codesegment von der Klasse BackgroundTask2 zum gleichzeitigen Starten der Threadprozessen

4.4.2 Externe definierte Parametern für Filterprozessen

Einige Parameter von den Filterprozessen, wie die Filterregeln, das bearbeitete Bildformat, die Abmessung der mobilen Webseite usw. werden in einer externer XML-Datei „configuration.xml“ definiert. Der Inhalt dieser XML-Datei ist wie folgenden Code dargestellt.

```
<?xml version="1.0" encoding="UTF-8"?>

<stmProject>
<semanticFilter>
<filterRules type="textSegment">
<parameter name="focusConcept" value= "PersonName"/>
<parameter name="focusConcept" value="Performance"/>
<parameter name="focusConcept" value="Ranking"/>
</filterRules>
<filterRules type="captionedImage">
<parameter name="minImgHeight" value="30"/>
<parameter name="minImgWidth" value="30"/>
<parameter name="allowedImgType" value="jpg"/>
</ filterRules>
<mobilePage type="owl">
<parameter name="displayWidth" value="240" />
<parameter name="displayHeight" value="320" />
<parameter name="imgHeightToLineRate" value="16" />
<parameter name="compressImgType" value="png"/>
</mobilePage>
<mobilePage type="xhtml">
</mobilePage>
</semanticFilter>
</stmProject>
```

Code-4.5: extern definiertes Dokument configuration.xml

Im Elemente "filterRules" wurden alle Filterreglen definiert, wie Definition der Fokuskonzepte für Textfilterung und Definition der minimalen Bildabmessung für Bildfilterung. Im Elemente "MobilPage" wurden alle Parameters bezüglich mobilen Webseite definiert, wie die Bildschirmabmessung des Mobiltelefons usw.

Alle in der configuration.xml definierten Parameters werden während der Initialphase der SemanticFilterModel-Klasse durch die öffentliche Methode parseConfigFile() von FilterFonfigurator-Klasse herausgeparst. Die FilterConfigurator-Klasse steht noch die öffentlichen Methoden zur Verfügung, um die gewonnene Parameter anzubieten.

4.5 Prozessphase III - Generierung und Freigabe

Dieses Unterkapitel behandelt die dritte Prozessphase, um die gefilterte Artikelinhalte in einer WML-Seite zu übersetzen und dann in einem entfernen Server zu veröffentlichen. Im Abschnitt 4.5.1 werden die führenden Klassen von der dritten Prozessphase vorgestellt. Im Abschnitt 4.4.2 und 4.4.3 werden die Methoden zur Generierung und Freigabe der WML-Seite vermittelt.

4.5.1 Führende Klassen in der Prozessphase III

In der dritten Prozessphase werden drei führende Klassen (in Abb.-4.13 gelb hervorgehoben Klassendarstellung in UML-Syntax) und zwei Datenstrukturen (in Abb.-4.13 blaue hervorgehoben) benutzt.

Abbildung-4.13
Klassendiagramm für Generierung und Freigabe der mobilen Webseite
(gelb: Klasse, Blau: Datenstruktur, -: privat, +: public, #: protected)

Die Funktionen von oben dargestellte Klassen und Datenstruktur werden wie unten geschrieben.

- **WMLPageGenerator-Klasse:** Alle Algorithmen zur Generierung der WML-Seite werden in dieser Klasse aufgebaut.
- **WMLCard-Datenstruktu:** Diese Datenstruktur ist eine innere Klasse von WMLPageGenerator-Klasse und wird zur Konstruierung der WML-Karte verwendet. Sie bietet öffentliche Methoden zur Übersetzung der von zweiter Prozessphase gefilterten Artikelelemente im WML-Code.

- **FTPConnector-Klasse:** Diese Klasse wird zur Hochladen der generierte WML-Seite in einem FTP-Server verwendet.

Die mehr Einzelheiten über die SemanticFilterModoel-Klasse und WebPageMetadataStore-Klasse wurden bereits im Unterkapitel 4.2 und Unterkapitel 4.3 geschrieben.

4.5.2 Generierung der mobilen Webseiten

Die Algorithmen zur Generierung der mobilen Webseiten sind in der Methode *doInBackground()* von der inneren BackgroundTask3-Klasse realisiert. Wie im Unterkapitel 3.5 vorgestellt wurde, werden der gefilterte Artikelinhalt in eine WML-Karte oder mehrere WML-Karten verteilt. Es gibt spezielle Methoden in der WMLCard-Klasse zur Generierung des Codes für einzelne Karte und multiple Karten. Der fertig generierte Quellcode von der Karte wird in einem WMLCard-Objekt gespeichert.

Die Generierung einer WML-Seite ist durch Aufruf der öffentlichen Methode *codeGenerate(wbms:WebPageMetadataStore, methode:String):String* von der WMPPageGenerator-Klasse realiseirt. Der Parameter "methode" entscheidet einzelne Karte oder Multiple-Karten. Die Generierung der einzelnen Karten wird durch die private Methode *singleCardGenerate(article:Article): WMLCard* implementiert. Die Generierung der Multiple-Karten wird durch die private Methode *multiCardsGenerate(article: Article):Vector<WMLCard>* implementiert.

Die WMLCard-Klasse bietet unterschiedliche Methoden zur Übersetzung jedes Artikelelementes im WML-Code. wie die Methode *setArticleTitel(titel:String)*, sie übersetzt die Überschrift des Artikels im entsprechenden WML-Code und fügt dann im Quellcode der mobilen WML-Seite hinzu. Außerdem wurden in der WMLCard-Klasse noch die Attribute wie maximale zulässige Zeilen und maximale zulässige Charakteranzahl sowie auch die maximal zulässige Bildabmessung für eine WML-Karte definiert. Diese Attributwerte werden von der Bildschirmsauflösung des Mobiltelefons festgelegt. Die Bildschirmsauflösung wurde bereits in externer Datei „configuration.xml“ definiert und ist durch Aufruf der öffentlichen Methode von Filterconfigurator-Klasse erhältlich.

4.5.3 Freigabe der generierten mobilen Webseiten

Nachdem die Methode *doInBackground()* von BackgroundTask3-Klasse fertig ausgeführt wurde, wird die Methode *done()* aufgerufen, um die generierte mobilen WML-seite in folgende drei Gebiete zu veröffentlichen.

- **Darstellung in Benutzeroberfläche**: Durch Aufruf der öffentliche Methode *showWMLCode(code:String)* von GUIController-Objekt in SemanticFilterModel-Klasse wird der Quellcode der WML-Seite direkt in Benutzeroberfläche dargestellt.
- **Speicherung im lokalen System**: Durch Aufruf der öffentliche Methode *writeInFile(filename:String, text:String, append:boolean)* von der Hilfsklasse FileWRTools wird der Quellcode der WML-Seite in einer WML-Datei des lokalem Systems gespeichert.
- **Hochladen nach entfernen FTP-Server**: Es wird durch Aufruf der öffentlichen Methode *uploadFile(localFilePath:String,destinationPath:String)* von FTPConnector -Klasse realisiert. Die URL-Adresse und die Zugangsdaten von dem FTP-Server wurden in der externen Datei ftpConfig.properties definiert.

4.6 Benutzung des Prototyps

In diesem Kapitel wird erläutert wie der Prototyp des semantischen Filters zur Filterung der Webseiteninhalte benutzt wird. Das Programm des Prototyps ist mit Java geschrieben und braucht nicht im Computer installiert werden, sondern kann direkt durch JAR-Datei im Unterverzeichnis „dist“ gestartet werden. Der Prototyp muss mit dem Internetanschluss arbeiten. Alle in der Arbeit als Beispiele benutzten Webseiten und ABoxen sind vom BOEMIE-Projekt erhalten und online zugänglich. Die URL-Adressen von Webseiten und ABoxen können im Anhang-A gefunden werden.

4.6.1 Konfiguration des Prototyps

Vor dem Starten der Filterprozesse soll der Prototyp bei folgenden drei Teilen konfiguriert werden.

I. Konfiguration der Zugangsadresse von RacerPro

Die IP-Adresse und der zum Empfang der Abfrage benutzte Port von RacerPro kann einfach durch „RacerPro-Configuration“ im Hauptmenü „Option“ auf der

Benutzeroberfläche eingestellt werden. Das Konfigurationsfenster ist wie Abbildung-4.14 gezeigt. Die Defaultwerte für IP-Adresse ist 127.0.0.1 für und für Port ist 8088.

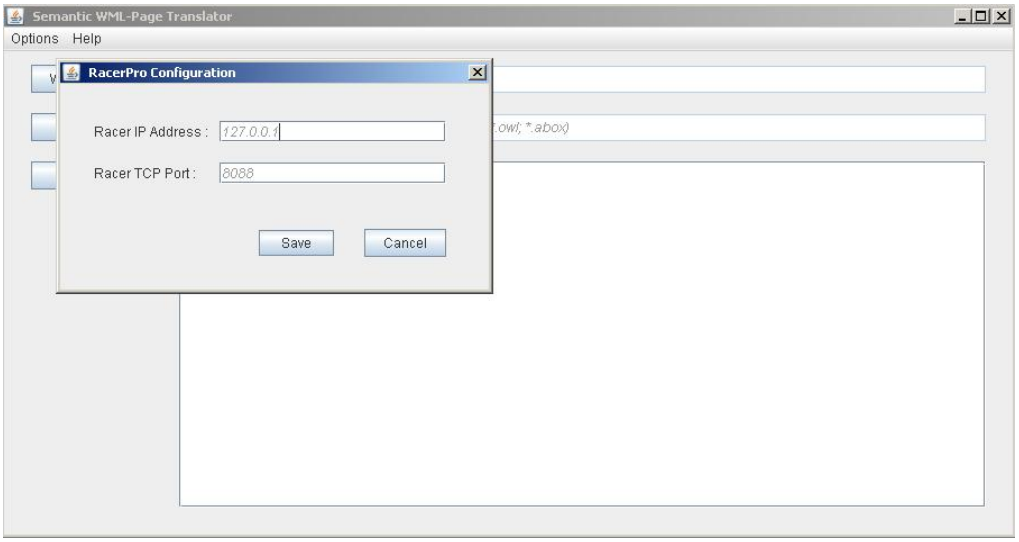

Abbildung-4.14
Konfigurationsfenster zur Änderung der Zugangsadresse von RacerPro

II. Konfiguration der wichtigen Parameter des semantischen Filters

Alle wichtigen Parameter des semantischen Filters werden in der externen Datei „configuration.xml“ definiert und können durch Text-Editor einfach umschrieben werden. Der Inhalt der Konfigurationsdatei wird in der Abbildung-4.15 gezeigt.

Alle benutzten Parameter werden aufgrund verschieden Aufgaben in folgenden zwei xml-Elemente definiert.

- **filterRules-Element:** Es sammelt alle Filterregeln und unterscheidet zwei Typen. Im Typ „text“ werden die Filterregeln für Textinhalte definiert. Der Parameter „FocusConcept“ definiert die Fokuskonzepte, die zur Beschränkung des Inhaltbereichs der Webseiten verwendet bestimmt werden. die weiteren Informationen über Fokuskonzept können in Abschnitt 3.4.6 gefunden werden. Im Typ „image“ werden die Filterregeln für Bildinhalte definiert. Der Parameter „miniImgHeight“ und „miniImgWidth“ definiert die minimale Größe der Bilder. Der Parameter „allowedImgType“ definiert das erlaubte Bildformat.

- **mobilPage-Element:** Es definiert die Formate der mobilen Webseiten und unterscheidet zwei Typen. In der Arbeit werden nur die Parameter für das Format „wml“ definiert. Der Parameter „displayHeight“ und „displayWidth“ definiert die Größe der mobilen Webseiten. Der Parameter „imgHeightToLineRate“ definiert das

Verhältnis zwischen Bildhöhe und Reihenzahl. Der Parameter „compressImg-Type“ definiert das Format der Bilder, die wegen Überschreiten der maximalen Größe der mobilen Seiten komprimiert werden müssen.

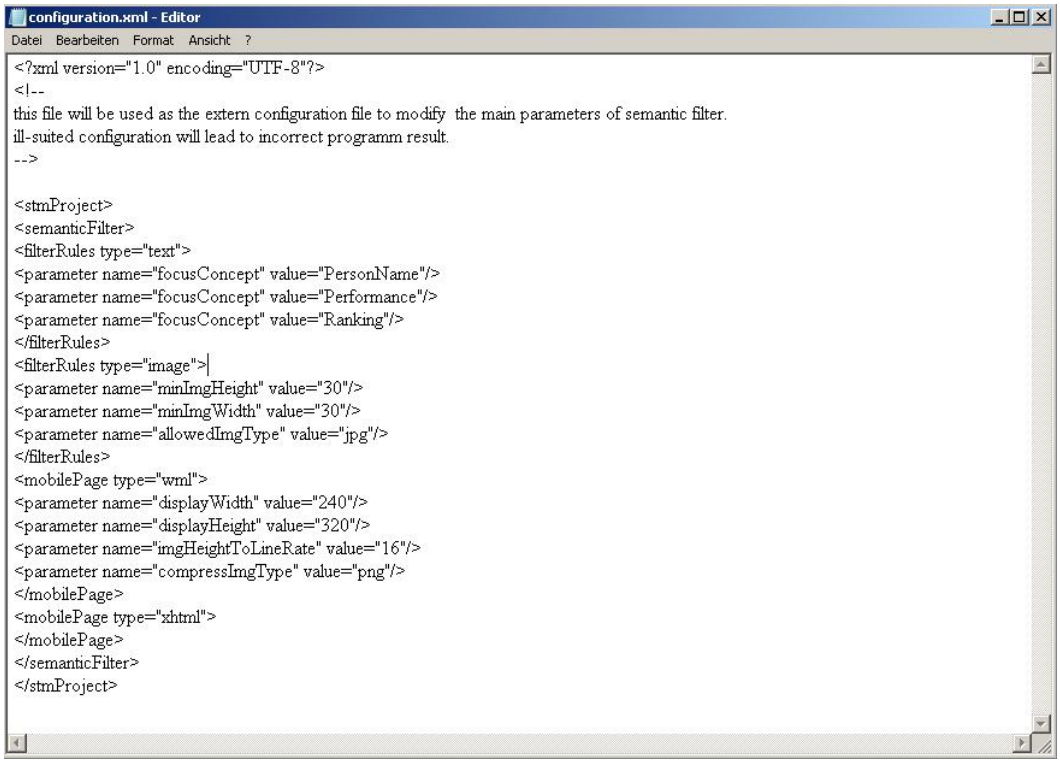

Abbildung-4.15
Konfigurationsdatei des semantischen Filters

III. Konfiguration der Zugangdaten des FTP-Servers

Die vom Prototyp generierten mobilen Seiten sollen in einem FTP-Server veröffentlicht. Die Zugangdaten des FTP-Servers werden in dem Datei „ftpConfig.propterties“ definiert und können durch Text-Editor wie Abbildung-4.16 einfach umschrieben werden.

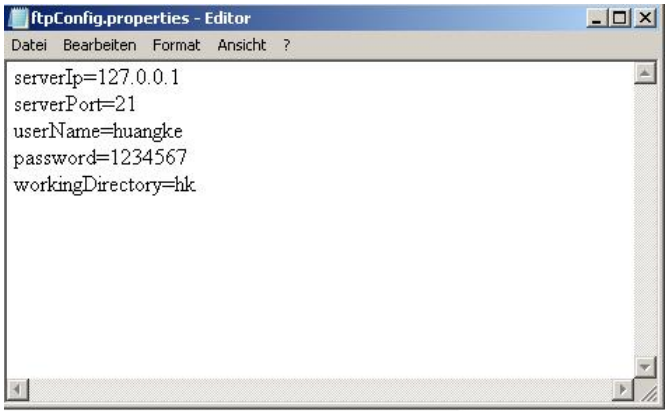

Abbildung-4.16
Zugangdaten des FTP-Servers

4.6.2 Starten des Prototyps

Wir nehmen eine Webseite von IAAF, die in der Abbildung-4.17 gezeigt wird, als Beispiel die Funktion des Prototyps zu zeigen. Diese Webseite und die zugehörige ABox sind vom BIEMIE-Projekt online erhältlich, die Ressourcenadressen können in Anhang-A gefunden werden.

Abbildung-4.17
Ein Beispiel für eine zu filternde Weibseite

Das Computer, welches den Prototyp betreibt, muss mit Internet verbinden, um die BOEMIE-Ontologien zu besuchen und die generierten mobilen Seiten nach FTP-Server hochzuladen. Außerdem muss das RacerPro vor dem Starten des Filterprozesses gestartet werden. Solange der Internetanschluss und RacerPro vorbereitet sind, kann der Benutzer die interessierte Webseite und die zugehörige ABox auswählen und dann durch den Start-Knopf den Filterprozess starten. Hierbei muss beachtet werden, dass jedes Mal nur eine Webseite mit HTML-Format und eine ABox mit OWL-Format bearbeitet werden darf. Der Prozessverlauf wird durch einem Progressfenster gezeigt. Die Benutzeroberfläche des Prototyps beim Starten des Prototyps ist in der Abbildung-4.18 dargestellt.

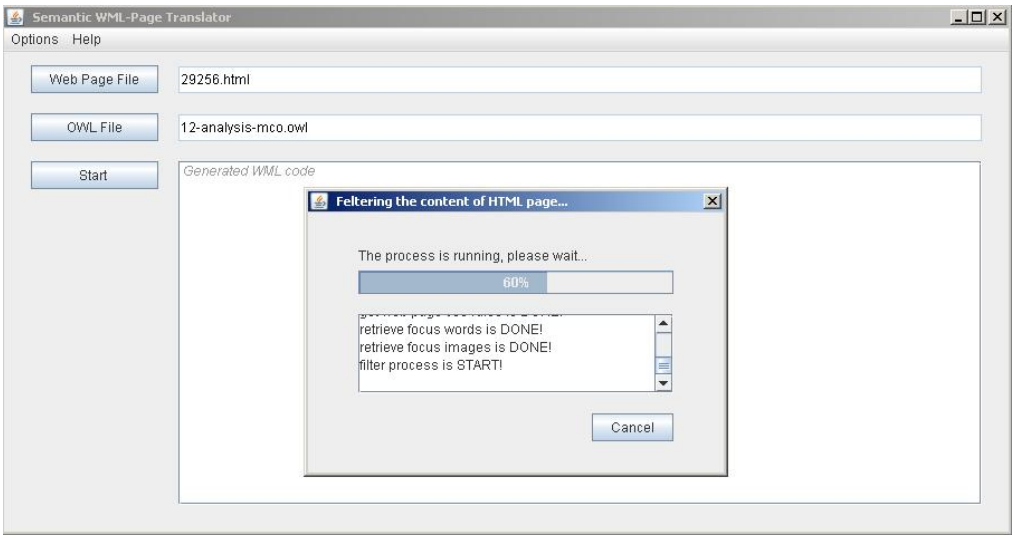

Abbildung-4.18
Benutzeroberfläche beim Starten des Filterprozesses

Nachdem der Filterprozess fertig gelaufen ist, wird der Quellcode der generierten mobilen Seite in der Benutzeroberfläche gezeigt (Abbildung-4.19 unten). Diese mobile Seite wird auch zugleich in dem lokalen Computer gespeichert und gemäß vorher definierten Zugangdaten nach dem FTP-Server hochgeladen.

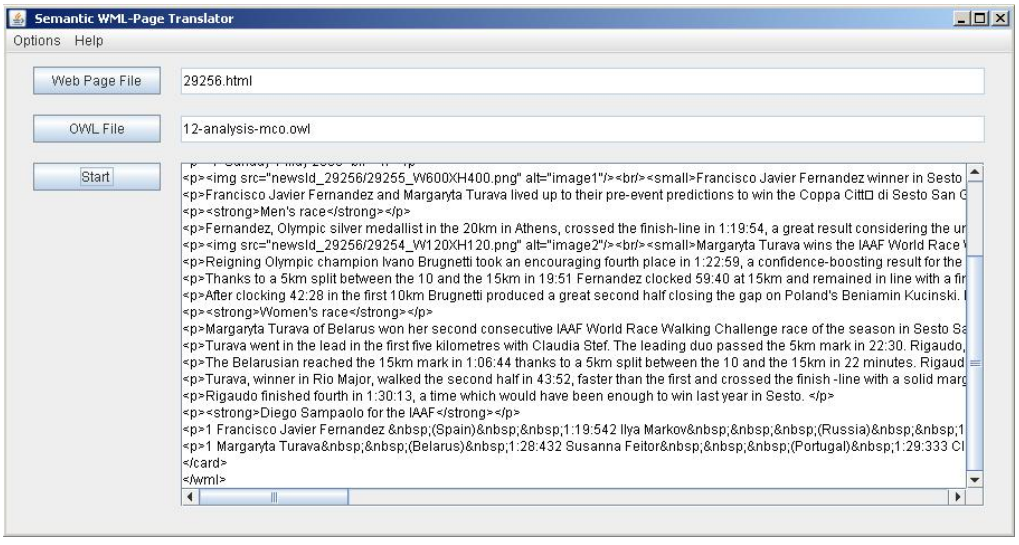

Abbildung-4.19
Benutzeroberfläche zum Zeigen des Quellcodes von der generierten mobilen Seite

Die Abbildung-4.20 zeigt eine Darstellung von der generierten mobilen Webseite für das in Abbildung-4.17 gezeigte Beispiel im M3GATE-Simulator, welcher den Mikrobrowser vom Mobiltelefon zur Darstellung der mobilen Seiten simuliert.

Abbildung-4.20
Darstellung der mobilen Seite im M3GATE Mobiltelefon-Simulator

5 Ergebnisse und Schlussfolgerung

Um die Anwendbarkeit des Prototyps zu bewerten, werden zwanzig Beispiele aus siebenhundert Webseiten, die im BOEMIE-Projekt durch ABoxen beschrieben worden sind, zufällig ausgewählt. Die entsprechenden Quelladressen von den zwanzig Webseiten werden in Anhang A aufgelistet. In diesem Kapitel werden die Filterungsergebnisse von den zwanzig Webseiten sowie auch die speziellen Situationen von den Filterregeln diskutiert.

Die Ontologien von BOEMIE sind in der Domäne der Leichtathletik aufgebaut. Die in BOEMIE untersuchten Webseiten sind überwiegend aus IAAF-Website und berichten über über Nachrichten von athletische Wettkämpfen. Die zentralen Inhalte von den athletischen Wettkämpfen bestehen häufig aus Spielereignis, Spieldatum, Spielort, Spielername, Spielort und Spielergebnisse, diese Daten interessieren die Webseitenleser am meisten. Wir nennen diese Elemente „wesentliche athletische Nachrichtenelemente“ (WAN-Elemente). Die Begriffe von den WAN-Elementen werden als Konzepte in TBoxen definiert und dann in ABoxen zur Beschreibung der Webseiteninhalte angewendet. Die WAN-Elemente können als Fokuskonzepte in den Filterregeln benutzt, um das Bereich der interessierten Inhalte zu bestimmen. Die Inhalte von den Webseiten, die nicht die Individuen der Fokuskonzepte beinhaltet sind, werden von dem semantischen Filter gefiltert. Die weiteren Einzelheiten Filterprozess wurden in Kapitel 3 erläutert und hierbei nicht weiter vorzustellen.

Webseiten	20 Webseiten	20 Webseiten	20 Webseiten
Zeichenanzahl vor Filterung (ZvF)	3278	3278	3278
Bildanzahl vor Filterung (BvF)	2,6	2,6	2,6
Fokuskonzepte für Textabschnitte	„PersonName“	„PersonName“ „Performance“	„PersonName“ „Performance“ „Ranking“
Fokuskonzepte für Bildtext	„PersonName“	„PersonName“	„PersonName“ „Ranking“
Prozessdauer (Durchschnittwerte)	14s52ms	16s31ms	20s26ms
Zeichenanzahl nach Filterung (ZnF)	2432	1693	1334
Bildanzahl nach Filterung (BnF)	2,5	2,5	1,8
Durchschnittliche Filterungsrate	14,8%	26,1%	45%

Tabelle-5.1: Prozessergebnis von Filterung der 20 Webseiten

Die Filterung der zwanzig Webseiten wird durch drei verschiedenen Fokuskonzeptgruppen in den Filterregeln für Textabschnitte und Bildtext ausgeführt, die in der Tabelle-5.1

gezeigt werden. Durch Vergleich der Zeichenanzahl und Bilderanzahl auf der Webseite vor und nach der Filterung kann der Filterungsrate berechnet werden. Der Filterungsrate zeigt, wie viele Prozente der Webseiteninhalte vom Prototyp weg gefiltert werden. Das Ergebnis von der Tabelle-5.1 zeigt, dass zirka 14,8% Webseiteninhalte weg gefiltert werden, wenn man das Fokuskonzept „PersonName“ für Text und Bild wählt. Der Filterungsrate wird auf 26.1% anwachsen, wenn das weitere Fokuskonzept „Performance“ für Text hinzugefügt wird. Die Weitere Ergänzung des Fokuskonzepts „Ranking“ für Texte und Bild führt zu einer Steigerung von der Filterungsrate auf 45%. Das Ergebnis zeigt, dass mehr Fokuskonzepte mehr Webseiteninhalte weg gefiltert werden lassen.

Außerdem können auch die Fokusindividuen in den Filterregeln definiert werden, um die Semantik der Webseiteninhalten mehr ausführlich zu beschränken. Wie das Beispiel in der Tabelle-5.2 gezeigt, werden zwei Fokuskonzepte „PersonName“ und „Country“ zur Filterung der Bilder zu benutzt. Das heißt, dass nur die Bilder, deren Semantik sowohl die Spielernamen als auch die Ländernamen beinhaltet, nicht weg gefiltert werden. Weiterhin kann der semantische Bereich von „Country“ mit dem Fokusindividuum „china“ verkleinern, wenn der Webseitenleser nur für Bilder mit chinesischen Spielern interessiert. Die Fokusindividuen können auch für die Filterung der Textinhalte gleichfalls verwendet werden, wie das Beispiel in der Tabelle-5.3 gezeigt.

Webseite	Webseite-1	Webseite-1	Webseite-1
Zeichenanzahl vor Filterung (ZvF)	5167	5167	5167
Bildanzahl vor Filterung (BvF)	5	5	5
Fokuskonzept für Textabschnitte	„PersonName“	„PersonName“ „Performance“	„PersonName“ „SportsName“ „Performance“ „Ranking“
Fokuskonzepte für Bildtext	„PersonName“	„PersonName“ „Ranking“	„PersonName“ „Country“
Fokusindividuen für Bildtext	-	-	„China“
Prozessdauer (Durchschnittwerte)	19s41ms	17s38ms	17s75ms
Zeichenanzahl nach Filterung (ZnF)	4994	4210	777
Bildanzahl nach Filterung (BnF)	5	1	3
Filterungsrate (1-(Znf/ZvF+Bnf/BvF)/2)	1,7%	49,3%	62,5%

Tabelle-5.2: Prozessergebnis von Filterung der Webseite-1 mit Fokusindividuen

Webseite	Webseite-3	Webseite-3	Webseite-3
Zeichenanzahl vor Filterung (ZvF)	3453	3453	3453
Bildanzahl vor Filterung (BvF)	2	2	2
Fokuskonzept für Textabschnitte	„PersonName“ „Performance“ „Ranking“	„PersonName“ „Performance“ „Ranking“	„PersonName“ „Performance“ „Ranking““
FokusIndividuen für Textabschnitte	-	„Spain“	„Spain“
Fokuskonzept für Bildtext	„PersonName“	„PersonName“	„PersonName“ „SportsName“
Prozessdauer (Durchschnittwerte)	14s31ms	13s43ms	15s43ms
Zeichenanzahl nach Filterung (ZnF)	1810	1257	1257
Bildanzahl nach Filterung (BnF)	2	2	0
Filterungsrate (1-(Znf/ZvF+Bnf/BvF)/2)	23.8%	31,9%	81,8%

Tabelle-5.3: Prozessergebnis von Filterung der Webseite-2 mit Fokusindividuen

Alle obigen Resultate werden dahin zusammengefasst, dass die in Kapitel 3 erläuterte Methode, welche durch Beschränkung des semantischen Bereichs der Webseiteninhalte zur Filterung der Webseiten entworfen wurde, in der praktischen Benutzung funktioniert und durchführbar ist. Die durchschnittliche Prozessdauer der Prüfergebnisse ist durchschnittlich innerhalb 20 Sekunden gemessen, sie ist für Entwicklung akzeptabel, aber für die praktische Situation noch relativ lang. Auf Grund der Einschränkung auf eine Webseitenquelle und der unvollständigen semantischen Beschreibung für Webseiten in der ABox haben die Prüfergebnisse keine Allgemeingültigkeit. Für die verschiedenartige Webseiten-Layouts und Webseiteninhalte hat diese Methode noch viele Begrenzungen.

Durch allmähliche Selbstevolution und Erweiterung der Definitionsbereiche von der Ontologien in BOEMIE können die Bedeutungen der Webseiteninhalte mehr ausführlich und verständlich in ABox geschrieben werden. Die Filterregeln können auch nicht nur für die Semantik der Textabschnitte und Bildtext, sondern auch für die graphischen Bedeutungen von Bildern und Videos sogar für die Webseiten-Layouts definiert. Damit kann die semantische Filterung für Webseiten mehr sinnvoll ausgeführt werden.

Die Untersuchung von der Arbeit wird nicht nur zur Reduzierung der Informationsmenge von den Webseiten verwendet, sondern kann auch zu anderen Bereichen beitragen. wie z.B. durch Analysis der Semantik auf Webseiteninhalte zur Identifikation und Lokalisierung von Web-Ressourcen oder zur Blockierung und Filterung der illegale Webseiteninhalte.

6 Zusammenfassung

Die Vorliegende Arbeit untersucht einen semantischen Filter, welcher durch Beschränkung des semantischen Bereichs von Webseiteninhalten die uninteressierten Webseiteninhalten von der Webseite filtert, um die Webseiten in kleinen Mobiltelefonen leichter anzeigen zu können. Dazu wurden zuerst die notwendigen Grundkenntnisse über mobiles Internetsurfen, Beschreibungslogik und semantische Informationsbeschreibung, sowie auch BOEMIE-Projekt und Reaoning-System RacerPro im Kapitel 2 vorgestellt.

Aufgrund der Grundkenntnisse wurden In Kapitel 3 das Arbeitsprinzip und der grundlegende Aufbau des semantischen Filters entworfen. Das Arbeitsprinzip des semantischen Filters beruhte auf vier Prozessschritte: Retrieval, Extraktion, Filterung und Übersetzung, welche durch vier Bauteile: Web-Page-Property-Picker (WPPP), Web-Page-Semantic-Picker (WPSP), Semantic-Filter-Model (SFM) und WML-Page-Generator (WPG) implementiert. Beim Retrievalsprozess wurden für die Filterung notwendige semantischen und nicht-semantischen Informationen aus der ABox und CSS-Datei der Webseiten wieder gewonnen. Mit Hilfe dieser semantischen und nicht-semantischen Informationen wurden die Artikelelemente, die in der Arbeit als zentrale Webseiteninhalte betrachtet wurden, zunächst beim Extraktionsprozess vom HTML-Quellcode der Webseiten heraus extrahiert und dann beim Filterungsprozess gemäß den Filterregeln gefiltert. Die Filterregeln wurden extern definiert und könnten nach eigenen Interessen erneut umgeschrieben werden. Die gefilterten Webseiteninhalte wurden am Ende beim Übersetzungsprozess ins Format der mobilen Seiten übersetzt.

Um die praktischen Funktionalitäten vom vorgelegten Entwurfskonzept zu überprüfen, wurde im Kapitel 4 ein Prototyp aufgebaut. Der Prototyp wurde in der Programmiersprache Java mit Hilfe der Entwicklungsumgebung NetBeans6.0 entwickelt und durch eine Benutzeroberfläche (GUI) gemäß der MVC-Architektur kontrolliert. Durch das GUI könnten die zu filternde Webseite und das entsprechende ABox ausgewählt und der Prozess des semantischen Filter gestartet werden. Außerdem könnte auch die RacerPro durch GUI konfiguriert werden. Die Filterregeln und das Format der mobilen Seiten wurden in einer Konfigurationsdatei außer dem Programm extern definiert. Nach dem Filterungsprozess des Prototyps wurde eine mobile WML-Seite am Ende generiert und auf einem entfernten FTP-Server veröffentlicht.

Die Funktionalität und Durchführbarkeit des Prototyps wurde im Kapitel 5 durch 20 Webseiten, deren ABox bereits von BOEMIE erzeugte wurden, geprüft und beurteilt.

Literaturverzeichnis

[RPUG08-C1] Racer Systems GmbH&Co.KG: **RacerPro User's Guide Version 1.9.2**, 18.Oktober 2008, Chapter 1
http://www.racer-systems.com/products/racerpro/users-guide-1-9-2-beta.pdf

[RPUG08-C6] Racer Systems GmbH&Co.KG: **RacerPro User's Guide Version 1.9.2**, 18.Oktober 2008, Chapter 6
http://www.racer-systems.com/products/racerpro/users-guide-1-9-2-beta.pdf

[RPRM07] Racer Systems GmbH & Co.KG: **RacerPro Reference Manual Version 1.9.2**, 18.October 2007
http://www.racer-systems.com/dl.php?file=PDF%252Freference-manual-1-9-2-beta.pdf&typ=file&name=RacerPro-Reference-Manual-1-9-2-beta.pdf

[WES06] Michael Wessel: **nRQL Tutorial**, Presentation for Racer System, 2006
http://www.sts.tu-harburg.de/people/mi.wessel/papers/nRQL-tut3.pdf

[GRO09] Florian Großmann : **RDF v. OWL-Inhaltssicht**, Lehrmaterial von HKI Uni-Köln, 22.Januar 2009
http://old.hki.uni-koeln.de/teach/ws0809/hs/tag12/RDFvOWL-Inhalt.ppt

[WIKI-RDF] Wikipedia : **Resource Description Framework**, Online-Document
http://de.wikipedia.org/wiki/Resource_Description_Framework

[WIKI-RDFS] Wikipedia: **RDF-Schema**, Online-Dokument
http://de.wikipedia.org/wiki/RDFS

[WIKI-OWL] Wikipedia : **Web Ontology Language**, Online-Document
http://de.wikipedia.org/wiki/Web_Ontology_Language

[RE07-C5] TUHH STS: **Reasoning Engine-Version 2**, Report of BOEMIE project, 07. August, Chapter 5
http://www.sts.tu-harburg.de/~r.f.moeller/papers/2007/EKMM+07.pdf

[BoePR06] BOEMIE offizielle Website: **BOEMIE Press release**, Online-Document, 06.October 2006
http://www.boemie.org/press_german

[MIDOE08] Silvana Castano, Sofia Espinosa, Alfio Ferrara, Vangelis Karkaletsis, Atila Kaya: **Multimedia Interpretation for Dynamic Ontology Evolution**, In Journal of Logic and Computation. Oxford University Press, 2008.
http://www.sts.tu-harburg.de/papers/2008/CEF+08.pdf

[BoeAR08] BOEMIE official Website: **BOEMIE Annual Reports 2008**, Online-Document
http://www.boemie.org/sites/default/files/BOEMIE%20Annual%20Report%202008_final.pdf

[SOKO08] Kamil Sokolski: **An Agent-oriented Architecture for Semantic Extraction and Ontology Evolution from Multimedia**, paper from TUHH, January 2008
http://www.sts.tu-harburg.de/pw-and-m-theses/2008/soko08.pdf

[SEFMC09] BOEMIE official Website: **D2.9 Semantics Extraction from Fused Multimedia Content-Final Version**, Public deliverables, March 2009

[MCDO08] BOEMIE official Website: **D3.8 Multimedia content and descriptor ontologies–final version**, Public deliverables, March 2008
http://www.boemie.org/sites/default/files/D3%208_25072008.pdf

[ITI712] Informatics and Telematics Institute: **BOEMIE Ontology**, Online Document
http://mklab.iti.gr/node/712

[SELEK09] Michael Jendryschik: **Selektoren**, Einführung in XHTML, CSS und Webdesign, 2009
http://jendryschik.de/wsdev/einfuehrung/css/selektoren

[JOC07] John O'Conner: **Improve Application Performance With SwingWorker in Java SE 6**, online technical Article of java.sun.com, January 2007
http://java.sun.com/developer/technicalArticles/javase/swingworker

Anhang-A Online Referenzen

BOEMIE Repository:

http://repository.boemie.org:8000/BoemieRepository/

Quelle der ABoxen vom BOEMIE:

http://repository.boemie.org/ontology_repository_abox/development2/abox/

W3C technisches Dokument für CSS Selektors

http://www.w3.org/TR/css3-selectors/#context

JAVA API für SwingWorker

http://java.sun.com/javase/6/docs/api/javax/swing/SwingWorker.html

In der Arbeit benutzten Webseiten und ABoxen:

Webseite-1:

http://repository.boemie.org:8000/BoemieRepository/url605/www.iaaf.org/WRC05/news/Kind=2/newsId=29202.html

ABox-1:

http://repository.boemie.org/ontology_repository_abox/development2/abox/1-3/10-analysis-mco.owl

Webseite-2:

http://repository.boemie.org:8000/BoemieRepository/url607/www.iaaf.org/WRC05/news/Kind=2/newsId=29256.html

ABox-2:

http://repository.boemie.org/ontology_repository_abox/development2/abox/1-3/12-analysis-mco.owl

Webseite-3

http://repository.boemie.org:8000/BoemieRepository/url608/www.iaaf.org/WRC05/news/Kind=2/newsId=29617.html

ABox-3:

http://repository.boemie.org/ontology_repository_abox/development2/abox/1-3/13-analysis-mco.owl

Webseite-4:

http://repository.boemie.org:8000/BoemieRepository/url609/www.iaaf.org/WRC05/news/Kind=2/newsId=31800.html

ABox-4:

http://repository.boemie.org/ontology_repository_abox/development2/abox/1-3/14-analysis-mco.owl

Webseite-5:

http://repository.boemie.org:8000/BoemieRepository/url263/www.iaaf.org/GP05/news/Kind=2/newsId=30459.html

ABox-5:

http://repository.boemie.org/ontology_repository_abox/development2/abox/1-1/101-analysis-mco.owl

Webseite-6:

http://repository.boemie.org:8000/BoemieRepository/url264/www.iaaf.org/GP05/news/Kind=2/newsId=30592.html

ABox-6:

http://repository.boemie.org/ontology_repository_abox/development2/abox/1-1/102-analysis-mco.owl

Webseite-7:

http://repository.boemie.org:8000/BoemieRepository/url265/www.iaaf.org/GP06/news/Kind=2/newsId=33724.html

ABox-7:

http://repository.boemie.org/ontology_repository_abox/development2/abox/1-1/103-analysis-mco.owl

Webseite-8:

http://repository.boemie.org:8000/BoemieRepository/url267/www.iaaf.org/GP06/news/Kind=2/newsId=34694.html

ABox-8:

http://repository.boemie.org/ontology_repository_abox/development2/abox/1-1/103-analysis-mco.owl

Webseite-9:

http://repository.boemie.org:8000/BoemieRepository/url87/www.iaaf.org/news/newsId=36938,printer.html

ABox-9:

http://repository.boemie.org/ontology_repository_abox/development2/abox/1-3/99-analysis-mco.owl

Webseite-10:

http://repository.boemie.org:8000/BoemieRepository/url86/www.iaaf.org/news/newsId=36350,printer.html

ABox-10:

http://repository.boemie.org/ontology_repository_abox/development2/abox/1-3/98-analysis-mco.owl

Webseite-11:
http://repository.boemie.org:8000/BoemieRepository/url319/www.iaaf.org/news/Kind=2/newsId=24849.html
ABox-11:
http://repository.boemie.org/ontology_repository_abox/development2/abox/1-1/123-analysis-mco.owl
Webseite-12:
http://repository.boemie.org:8000/BoemieRepository/url336/www.iaaf.org/news/Kind=2/newsId=27688.html
ABox-12:
http://repository.boemie.org/ontology_repository_abox/development2/abox/1-1/142-analysis-mco.owl
Webseite-13:
http://repository.boemie.org:8000/BoemieRepository/url353/www.iaaf.org/news/Kind=2/newsId=29721.html
ABox-13:
http://repository.boemie.org/ontology_repository_abox/development2/abox/1-1/159-analysis-mco.owl
Webseite-14:
http://repository.boemie.org:8000/BoemieRepository/url371/www.iaaf.org/news/Kind=2/newsId=33157.html
ABox-14:
http://repository.boemie.org/ontology_repository_abox/development2/abox/1-1/177-analysis-mco.owl
Webseite-15:
http://repository.boemie.org:8000/BoemieRepository/url380/www.iaaf.org/news/Kind=2/newsId=34357.html
ABox-15:
http://repository.boemie.org/ontology_repository_abox/development2/abox/1-1/188-analysis-mco.owl
Webseite-16:
http://repository.boemie.org:8000/BoemieRepository/url190/www.iaaf.org/GLE04/news/Kind=2/newsId=26415.html
ABox-16:
http://repository.boemie.org/ontology_repository_abox/development2/abox/1-1/42-analysis-mco.owl
Webseite-17:

http://repository.boemie.org:8000/BoemieRepository/url531/www.iaaf.org/WAF07/news/Kind=2/newsId=40056.html

ABox-17:

http://repository.boemie.org/ontology_repository_abox/development2/abox/1-2/132-analysis-mco.owl

Webseite-18:

http://repository.boemie.org:8000/BoemieRepository/url530/www.iaaf.org/WAF07/news/Kind=2/newsId=39848.htm

ABox-18:

http://repository.boemie.org/ontology_repository_abox/development2/abox/1-2/131-analysis-mco.owl

Webseite-19:

http://repository.boemie.org:8000/BoemieRepository/url528/www.iaaf.org/WAF07/news/Kind=2/newsId=38997.html

ABox-19:

http://repository.boemie.org/ontology_repository_abox/development2/abox/1-2/129-analysis-mco.owl

Webseite-20:

http://repository.boemie.org:8000/BoemieRepository/url64/www.iaaf.org/GLE05/news/Kind=2/newsId=31946.html

ABox-20:

http://repository.boemie.org/ontology_repository_abox/development2/abox/1-3/47-analysis-mco.owl