

Konzeption und Implementierung eines
Multi-Agentensystems zur Informationsgewinnung
im Rahmen einer mobilen Applikation

Master's Thesis in Informatik
von
Daniel Brügge

Konzeption und Implementierung eines Multi-Agentensystems zur Informationsgewinnung im Rahmen einer mobilen Applikation

Master's Thesis in Informatik
von
Daniel Brügge

Technische Universität München
Fakultät für Informatik
Lehrstuhl für Angewandte Informatik und Kooperative Systeme

In Kooperation mit
track-u mobility services GmbH, München

Aufgabensteller:	Prof. Dr. Johann Schlichter
Betreuer:	Dipl. Inf. Karlheinz Toni
Externer Betreuer:	Dipl. Kfm. Hans Maier-Dech
Datum der Abgabe:	15. Mai 2007

Ich versichere hiermit, dass ich die vorliegende Master's Thesis selbstständig verfasst und dabei keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

.....
(Daniel Brügge)

München, den 14. Mai 2007

Zusammenfassung

Seit den Anfängen des World Wide Web gewinnt die Informationsextraktion (IE) auf semi-strukturierten und natürlich-sprachlichen Texte immer mehr an Bedeutung. Auch die vorliegende Masterarbeit widmet sich diesem umfangreichen Gebiet und konzipiert und entwickelt ein System zur Extraktion von Informationen aus Internet-Quellen.

Dabei stehen Informationen im Vordergrund, die im Projekt „track-u“ der track-u mobility services GmbH, einer mobilen Anwendung zugute kommen sollen.

Das System basiert dabei auf der Nutzung eines Multiagentensystems (MAS), welches die Aufgabe der Quellen-Überwachung übernimmt und mit Hilfe der FIPA-kompatiblen JADE-Agentenplattform umgesetzt wird.

Zusätzlich widmet sich die Arbeit neben der Analyse bestehender IE-Techniken und der Entwicklung eines eigenen Wrappers auf Basis des MAS auch dem Entwurf und der Umsetzung einer Webapplikation, die zur Verwaltung des Systems dient. Dabei kommen Techniken wie das MVC Webapplikations-Framework Ruby on Rails und XML-RPC zum Einsatz.

INHALTSVERZEICHNIS

1	Einleitung	1
1.1	Das Projekt „track-u“	1
1.1.1	Grundidee des Systems	1
1.1.2	Erkennung von Bewegungsmustern	1
1.1.3	Sicherheitszonen	2
1.1.4	Eskalationslogik	2
1.2	Ziele der Arbeit	2
1.3	Gliederung der Arbeit	2
2	Motivation und Projektintegration	4
2.1	Motivation	4
2.1.1	Szenario 1: Beeinflussung der Sicherheitsstufen	4
2.1.2	Szenario 2: Beantwortung von Anfragen der Eskalationslogik	5
2.2	Die Integration der Arbeit in das Projekt	5
3	Informationsgewinnung	6
3.1	Aufbau und Struktur von Texten	6
3.1.1	Unstrukturierte Texte	6
3.1.2	Strukturierter Text	7
3.1.3	Semi-strukturierte Texte	7
3.2	Die Informationsextraktion	9
3.2.1	Grundlegende IE-Ansätze (manuell vs. automatisch)	10
3.2.2	IE bei natürlich-sprachlichen Texten	10
3.2.3	IE-Ansätze bei semi-strukturierten Texten	15
3.2.4	Unterschiede in diesem System	18
3.3	Projektrelevante Informationen: Ereignisgruppe	18

3.4	Typen von Informationsquellen	19
3.4.1	Analyse anhand bestehender Beispielquellen	20
3.4.2	Präsentation, Verteilung und Erreichbarkeit von Ereignisgruppen	22
3.5	Praxisnahe IE-Techniken	28
3.5.1	Gruppen-Extraktion anhand von Semistruktur und Mustern	29
3.5.2	Gruppen-Extraktion aus natürlich-sprachlichen Texten	32
4	Systemanforderungen und -aufbau	33
4.1	Anforderungen	33
4.1.1	Multiagentensystem	33
4.1.2	Informationsgewinnung	33
4.1.3	Definition von Extraktionsregeln	34
4.2	Grundstruktur	34
5	IE-Systementwurf	36
5.1	Anforderungen an die IE-Komponente	36
5.2	Laden der Informationsquellen	37
5.3	Auffinden und Extraktion von Informationen innerhalb eines Textes	37
5.3.1	Extraction-Pattern	39
5.3.2	Zusammenführung von Ereignisgruppen	41
5.3.3	Natürlich-sprachliche Methoden	43
5.4	Navigation in der Quelle	44
5.4.1	Verkettung von Extraction Pattern	44
5.4.2	Vorgehen bei den unterschiedlichen Verteilungstypen	46
5.5	Formularanfragen	50
5.5.1	POST- und GET-Requests	50
5.6	Aktualisierte Systemstruktur	51
6	Entwurf des Multiagentensystems	53
6.1	Agentensystem	53
6.1.1	Agent	53
6.1.2	Agentenplattform	54
6.1.3	Vorteile	55
6.2	Anforderungen an das MAS	56

6.3	Informationsagent	56
6.4	IE-Handling im Agenten	58
6.4.1	Initiierung der IE	58
6.4.2	Geokodierung von Ortsangaben	58
6.4.3	Verhaltensweisen nach Extraktion	61
6.4.4	Berücksichtigung von Einflüssen auf Sicherheitszonen	63
6.4.5	Aktualisierung Ausführungszeitpunkt	63
6.5	Zusatzaufgaben der Plattform	64
6.5.1	Zustands-Wiederherstellung nach Ausfällen	64
6.5.2	Erstellung neuer Agenten	64
6.6	Aktualisierte Systemstruktur	65
7	Entwurf der Managementapplikation	67
7.1	Anforderungen	67
7.2	Architektur	67
7.3	Erstellen der Extraktionsregeln	68
7.3.1	Intuitive Benutzeroberfläche bei Verkettung	69
7.3.2	Semi-automatische Erstellung regulärer Ausdrücke	69
7.3.3	Vorschau der Extraktion	70
7.4	Hinzufügen neuer Informationsquellen	70
7.5	Kommunikation mit der Agentenplattform	71
7.6	Informative und administrative Aufgaben	72
7.7	Aktualisierte Systemstruktur	73
8	Repository-Design	75
8.1	Struktur	75
8.2	Extraktionsregeln	75
8.3	Informationsquellen	77
8.4	Ereignisse	78
8.5	Aktualisierte Systemstruktur	78

9	Systemumsetzung	80
9.1	Verwendete Technologien	80
9.1.1	JADE Agentenframework	80
9.1.2	XML-RPC	81
9.1.3	Ruby on Rails Web-Framework	83
9.2	IE-Kernkomponente	84
9.2.1	Datenobjekte	84
9.2.2	Ausführende Komponenten	86
9.3	Geokodierung und Ereignisverwaltung	89
9.4	Agenten, Quellen-Verwaltung und XML-RPC	91
9.4.1	Laden und Starten der Agenten	91
9.4.2	Starten der Informationsextraktion	91
9.5	Verwaltungsapplikation	92
9.5.1	Erstellen der Extraktionsregeln	93
9.5.2	Hinzufügen neuer Informationsquellen	94
9.5.3	Überwachung	95
10	Ausblick und Erweiterungen	97
10.1	Informationsagenten	97
10.2	Informationsextraktion	98
10.3	Managementapplikation	99
10.4	Anbindung an track-u	99
11	Zusammenfassung und Probleme	101
11.1	Zusammenfassung	101
11.2	Probleme	102
	Abbildungsverzeichnis	104
	Tabellenverzeichnis	105
	Beispiele	106
	Literaturverzeichnis	110

1

Einleitung

Das vorliegende Kapitel soll eine kurze Einführung in diese Thesis geben. Da die Arbeit im Kontext des Projekts „track-u“ entstanden ist, wird mit einer kurzen Beschreibung des Projekts begonnen. Danach werden die Ziele erwähnt, die das System erfüllen muss und hier ebenfalls der Bezug zum track-u Projekt hergestellt. Zum Schluss wird eine Gliederung der Arbeit vorgenommen, bei der jedes Kapitel inhaltlich beschrieben wird, um einen besseren Überblick zu gewährleisten.

1.1 Das Projekt „track-u“

Die vorliegende Arbeit ist in Kooperation mit der Firma track-u mobility services GmbH¹ im gleichnamigen Projekt „track-u“ entstanden. Im folgenden soll das Projekt vorgestellt werden.

1.1.1 Grundidee des Systems

Die Grundidee von track-u ist es, Bewegungen von Kindern intelligent und selbstständig zu verfolgen, um in Gefahren- oder anderen Sondersituationen automatisch Warnmeldungen geben zu können. Als solche Situationen gelten dabei „ungewöhnliche“ Abweichungen von Standard-Bewegungsmustern und das Betreten unsicherer Gebiete.

Den Eltern des Kindes wird dadurch die Möglichkeit gegeben, dass das Kind nicht eine ständige Anwesenheit eines Erziehungsberechtigten benötigt und sich „frei“ bewegen kann, ohne aber ganz auf eine Aufsicht verzichten zu müssen.

1.1.2 Erkennung von Bewegungsmustern

Damit das System funktioniert, werden die Bewegungen des Kindes benötigt. Hierfür werden mittels GPS (*Global Positioning System*) und mit Hilfe eines Mobilfunkgerätes die Koordinaten des Kindes in bestimmten Zeitabständen an einen Server übermittelt und in der so genannten *Geo-Datenbank* gespeichert. Durch die Protokollierung dieser einzelnen Punkte, wird nach einer gewissen Zeitspanne ein detailliertes Bewegungsmuster des Kindes erzeugt. So zählt zum Beispiel der regelmässige Weg zur Schule, zum Kindergarten oder zu den Freunden zu diesem Muster.

Falls das Kind nun von diesem Muster abkommt, indem es unter Umständen einen anderen Weg zu einem bestimmte Ziel wählt, kann dieses vom track-u System automatisch erkannt werden.

¹<http://www.track-u.de>

1.1.3 Sicherheitszonen

Neben den Bewegungsmustern mit denen ein Kind seine Standard-Routen definiert, verwaltet das track-u System zusätzlich noch so genannte „Zonen“, welche ebenfalls innerhalb der Geo-Datenbank gespeichert werden. Eine Zone definiert dabei einen bestimmten geographischen Bereich. Diese Zonen können mit unterschiedlichen Sicherheitsstufen belegt werden und beispielsweise stufenlos vom Zustand „unsicher“ bis zum Zustand „sicher“ gehen.

Diese Sicherheitsstufen definieren, ob beim Betreten der Zone durch das Kind, eine Warnmeldung rausgehen sollte (bei unsicheren Zonen) bzw. eine Meldung nicht notwendig ist (bei sicheren Zonen).

Die Sicherheitsstufen dieser Zonen können manuell eingetragen werden (z.B. von den Eltern) oder aber im begrenzten Umfang automatisch festgelegt werden. Mit der automatischen Festlegung dieser Sicherheitsstufen beschäftigt sich u.a. die vorliegende Arbeit, wie im nächsten Kapitel über die Projektintegration noch detaillierter beschrieben wird.

1.1.4 Eskalationslogik

Im vorherigen Abschnitt war die Rede von Warnmeldungen, die versendet werden können, falls das Kind eine unsichere Zone betritt oder von den Bewegungsmustern abweicht. Die Komponente im track-u System, die diese Aufgabe übernimmt, ist die *Eskalationslogik*.

Diese ist beispielsweise dafür zuständig, eine entsprechende Meldung zu versenden, falls das Kind nicht mehr über GPS auffindbar ist. Kurz gesagt, immer dann, wenn mögliche Probleme auftreten, handelt die Eskalationslogik.

Nachdem nun das track-u System kurz vorgestellt wurde, soll auf die Ziele der Arbeit eingegangen werden.

1.2 Ziele der Arbeit

Im großen und ganzen ist das Ziel der vorliegenden Arbeit, ein „System zur Extraktion von relevanten Informationseinheiten aus Internetquellen auf Basis eines Multiagentensystems“ zu entwickeln. Die extrahierten Informationen müssen vom System zusätzlich in der Form aufbereitet werden, dass sie im track-u Projekt genutzt werden können.

Um dieses Gesamtziel zu erreichen, müssen in der Arbeit einige Teilziele verfolgt werden.

Dazu zählt u.a. die Untersuchung bestehender Techniken der Informationsextraktion (IE), um dadurch einen Ein- und Überblick über diesen Bereich zu bekommen. Zusätzlich muss das Umfeld analysiert werden, in dem die spätere Extraktion stattfinden wird – das Internet mit seinen unterschiedlichen Informationsquellen. Aufgrund der Heterogenität dieser Quellen muss ein Wrapper entwickelt werden, der es erlaubt, auf die Besonderheiten jeder Quelle eingehen und mit diesen umgehen zu können.

Ebenso muss dazu eine Schnittstelle entworfen und umgesetzt werden, die es einem Benutzer erlaubt, das System zu überwachen und zu steuern.

1.3 Gliederung der Arbeit

In den vorherigen Abschnitten wurde das track-u Projekt schon kurz mit einigen seiner wichtigen Komponenten und Konzepte vorgestellt. Mit **Kapitel 2** wird detaillierter auf das Projekt eingegangen und die Motivation der vorliegenden Arbeit anhand einiger Szenarien nochmals unterstrichen. Zudem werden die Möglichkeiten der späteren Projektintegrationen vorgestellt, bevor sich in Kapitel **Kapitel 3** dem

Thema der „Informationsgewinnung“ gewidmet wird.

Wie sich bei der Definition der Ziele schon herausgestellt hat, ist es dafür notwendig, eine Untersuchung bestehender Techniken der Informationsextraktion vorzunehmen. Dabei werden ebenfalls Unterschiede herausgearbeitet, die zwischen herkömmlichen Ansätzen und den Ansätzen in dem zu entwickelnden System bestehen. Zum Schluss werden einige Techniken vorgestellt, die sich für die spätere Umsetzung anbieten.

Nachdem nun die Arbeit bis Kapitel 3 eher einen allgemeineren Charakter hatte, beginnt mit **Kapitel 4** die Konzentration auf das entstehende System. Hierbei wird anhand der Systemanforderungen eine grobe Grundstruktur entwickelt und es werden die Hauptkomponenten und Schnittstellen des System herausgearbeitet. Der Entwurf dieser Komponenten wird dann in den Folgekapiteln behandelt, wobei mit dem Entwurf der Extraktions-Komponente in **Kapitel 5** begonnen wird. Danach folgt der Entwurf des Multiagentensystems in **Kapitel 6**, wobei in diesem Kapitel auch die wichtigen Grundbegriffe wie „Agentensystem“ und „Agenten“ erläutert werden. Zudem wird die Anbindung an die IE-Komponente diskutiert und in den Entwurf integriert.

Die Konzeption der Benutzerschnittstelle des Systems wird in **Kapitel 7** in Form einer Webapplikation vorgenommen, aber es werden ebenfalls mögliche Alternativen diskutiert. Auch hier wird die Verbindung zu den übrigen Komponenten erläutert und entworfen, bevor sich anschließend mit **Kapitel 8** dem Thema der Datenspeicherung zugewendet wird. Dabei wird eine Datenstruktur entwickelt, in der alle im System genutzten Daten persistent gespeichert werden können. Dieses Kapitel bildet dabei den Abschluss der Entwurfsphase und präsentiert die komplette Systemstruktur, die mit jedem der Entwurfskapitel gewachsen ist.

Die Umsetzung bzw. Implementierung des System wird in **Kapitel 9** beschrieben, wobei in einem Anfangsabschnitt einige wichtige Technologien vorgestellt werden, die im späteren System genutzt werden. Zum einen wird dabei das JADE-Agentenframework präsentiert, aber auch Technologien wie XML-RPC und das Webapplikations-Framework Ruby on Rails sind Themen dieses Abschnitts.

Im restlichen Teil des Kapitels wird auf die Umsetzung des Wrappers, der Webapplikation und des Multiagentensystems eingegangen und die Algorithmen bzw. die Klassenstrukturen werden vorgestellt.

Da die Systemumsetzung in einer Art prototypischen Form erfolgt, sind viele Möglichkeiten für spätere Systemergänzungen gegeben. Aus diesem Grund wird mit **Kapitel 10** ein ausgiebiger Ausblick auf mögliche Erweiterungen und Verbesserungen gewagt.

Am Ende der Arbeit werden durch **Kapitel 11** die Resultate der Thesis nochmals zusammengefasst und einige Probleme aufgezeigt, die während der Konzeption und Umsetzung aufgetreten sind und Probleme erwähnt, die u.U. bei einem späteren Einsatz des Systems auftreten können.

2

Motivation und Projektintegration

In der Einleitung wurde eine kurze Übersicht über das „track-u“-Projekt gegeben, in dessen Rahmen diese Arbeit entstanden ist. Das vorliegende Kapitel soll nun zeigen, inwiefern das zu entwickelnde System Bestandteil vom Gesamtprojekt ist. Das heißt, dass die Motivation erläutert wird und die Art und Weise der Integration dargestellt wird. Zudem sollen einige beschreibende Szenarien das System verdeutlichen.

2.1 Motivation

Wie schon erwähnt wurde, lassen sich die Sicherheitsstufen der einzelnen Zonen bislang hauptsächlich manuell festlegen. Da sich aber die Sicherheit in den geographischen Bereichen abhängig von auftretenden Geschehnissen ständig ändern kann, sollte dieses im System mit berücksichtigt werden.

Die vorliegende Arbeit geht auf diese Problematik ein und stellt ein System vor, das es ermöglicht, relevante Informationen automatisch aus Internetquellen einzubeziehen und damit Sicherheitsstufen zu beeinflussen.

Eine weitere Möglichkeit, die durch den Einsatz des Systems geboten wird, ist die Bereitstellung zusätzlicher Informationen, um Anfragen der Eskalationslogik besser beantworten zu können.

Im folgenden werden zwei Szenarien in Prosa vorgestellt, die das eben erwähnte näher beschreiben.

2.1.1 Szenario 1: Beeinflussung der Sicherheitsstufen

Ein bestimmter geographischer Bereich ist innerhalb des Systems (in der Geo-Datenbank) in Zonen unterteilt, welche jeweils über vordefinierte Sicherheitsstufen verfügen. Diese Stufen wurden beim Anlegen durch die Eltern beispielsweise definiert oder direkt vom System vorgegeben.

Zum Beispiel handelt es sich bei den Zonen um

1. den Bereich der Schule, welcher als „sicher“ gekennzeichnet ist (Zone: z1)
2. das Haus der Großeltern, das ebenfalls als „sicher“ markiert ist. (Zone: z2)
3. das elterliche Haus mit Stufe „sicher“. (Zone: z3)
4. das Hafenviertel mit einem „unsicheren“ Zustand. (Zone: z4)

Das Agentensystem überwacht nun mehrere Quellen, welche Informationen aus diesen Bereichen liefern. Beispielsweise die lokalen Polizeinachrichten, die Nachrichten der Feuerwehr und eine Quelle für

Veranstaltungen.

In der Quelle der Polizei erscheint nun folgende Meldung:

„Im Bereich der Straße x in Hamburg wurde ein Kind von einem unbekanntem Mann belästigt.“

Wie es sich herausstellt, liegt die Straße x laut Geokoordinaten im Bereich des Hauses der Großeltern in Zone z2. Da Informationen aus der Quelle der Polizei sich negativ auf die Sicherheitsstufe einer Zone auswirken, wird nun die Stufe in der Zone z2 in Richtung „unsicher“ gehen. Das hat zur Folge, dass eine Meldung o.ä. an die Kontaktpersonen gesendet wird, falls das erfasste Kind sich in dieser Zone aufhält.

2.1.2 Szenario 2: Beantwortung von Anfragen der Eskalationslogik

In der Geo-Datenbank des track-u Systems sind einige Bewegungsmuster eines erfassten Kind vorhanden. Das heißt, dass dem System diese Standard-Pfade des Kindes bekannt sind und es bei eventuellen Abweichungen Meldung geben kann.

Das Problem ist nun hier, dass auch „Fehlalarme“ auftreten können. Zum Beispiel wäre es möglich, dass das Kind jeden Tag im Zeitraum von 14:00-15:00 eine bestimmte Strecke A verwendet, um von der Schule zum elterlichen Haus zu gelangen. Auf dieser Strecke befindet sich auch ein Teilstück, welches gewöhnlich von 14:20-14:40 mit der U-Bahn zurückgelegt wird. In diesem Bereich kann kein GPS-Signal empfangen werden.

Nun ist das Kind das letzte Mal um 14:20 vom System erfasst worden, aber noch nicht um 14:40 mit einem neuen GPS-Signal in die Erfassung „zurückgekehrt“ und mittlerweile ist es schon 15:00. Das System würde nun eine Meldung an die Kontaktpersonen rausschicken. Diese Meldung sagt aber nichts über den Grund der Verspätung aus. Hier greift nun die vorliegende Arbeit ein.

Da von dem Agentensystem auch die Meldungen der lokalen Verkehrsbetriebe überwacht werden, stellt sich heraus, dass im Bereich der Ziel-U-Bahn-Station eine Verspätung von ca. 30 Minuten aufgrund von Gleisarbeiten stattgefunden hat.

Die Eskalationslogik hat die Möglichkeit anzufragen, ob für einen bestimmten geographischen Bereich Meldungen vorhanden sind, wie es in diesem Beispiel der Fall ist. Die Eltern können danach in Kenntnis gesetzt werden und auch noch erläuternde Informationen zugesendet bekommen.

2.2 Die Integration der Arbeit in das Projekt

Die Szenarien aus dem letzten Abschnitt haben beispielhaft dargestellt, in welcher Art und Weise extrahierte Informationen vom track-u System genutzt werden können. Zum einen kann das in dieser Thesis entstehende System direkt auf die Geo-Datenbank zugreifen und dort die Sicherheitsstufen der entsprechenden Zonen abändern.

Zum anderen wird dem track-u System eine Möglichkeit geboten, auf eine Datenbasis zuzugreifen, die vom System aus dieser Arbeit zur Verfügung gestellt wird und alle notwendigen Daten enthält, um z.B. Hintergrundinformationen liefern zu können (u.a. erläuternder Text für die Eltern).

Der Zugriff auf diese Daten kann dabei über eine wohldefinierte Schnittstelle erfolgen, welche keine direkten Datenbankzugriffe notwendig macht. Andererseits kann auch direkt auf die Datenbank zugegriffen werden¹.

¹Alle die eben vorgestellten Integrationsmöglichkeiten werden in der Arbeit behandelt, umgesetzt wird letztendlich die Bereitstellung einer Datenbank mit den extrahierten Informationen.

3

Informationsgewinnung

Wie der Titel der Arbeit schon vermuten lässt, stellt die Informationsgewinnung, welche im weiteren auch als „Informationsextraktion“ (eng. *information extraction*) bezeichnet wird, einen wesentlichen Teil des Systems dar¹.

Um eine Vorstellung zu bekommen, was der Begriff Informationsextraktion bedeutet, wird am Anfang dieses Kapitels auf die Entstehung der IE eingegangen und verschiedene Methoden bzw. Techniken vorgestellt. Zudem werden die Unterschiede zwischen den allgemein gültigen Definitionen und der Extraktion in dieser Thesis herausgearbeitet.

Später im Kapitel werden einige für das System relevante Informationsquellen vorgestellt und analysiert und auf Basis dessen Extraktionskonzepte entwickelt, die in dem späteren Entwurf bzw. in der Umsetzung Anwendung finden werden.

Zunächst einmal werden einige wichtige Begrifflichkeiten eingeführt, die für das weitere Kapitel von Bedeutung sind und sich auf die Struktur von Texten beziehen.

3.1 Aufbau und Struktur von Texten

Informationsgewinnung bzw. Informationsextraktion (IE) kann auf verschiedenen Medienformaten, wie beispielsweise Audiodaten, durchgeführt werden. Im Rahmen dieser Arbeit konzentriert sich die IE aber ausschließlich auf Dokumente, die im Textformat vorliegen.

Diese Dokumente können dabei unterschiedlich strukturiert sein, genauer gesagt **unstrukturierten**, **strukturierten** bzw. **semi-strukturierten** Text aufweisen.

3.1.1 Unstrukturierte Texte

Die einfachste Form bei Texten weist der *unstrukturierte Text* auf, welcher im folgenden auch häufig „Freitext“ genannt wird. Hierbei handelt es sich um einen Text, der in keiner Form eine Struktur beinhaltet bzw. nicht einem vorher definierten Datenschema folgt [Gei03]. Alle natürlich-sprachlichen Texte fallen unter diese Kategorie, wie beispielsweise ein Bericht in einer Zeitung [Sod99].

Um aus unstrukturierten Texten Informationen bzw. Attribute zu extrahieren, sind linguistische Methoden erforderlich, die u.a. Beziehungen zwischen Wörtern oder deren Bedeutung erkennen [Eik99].

Freitexte sind häufig Bestandteil der später untersuchten und verwendeten Quelle in dem System. Folgender Ausschnitt² ist ein Beispiel für einen unstrukturierten Text:

¹Auch „Information Retrieval“ kann mit „Informationsgewinnung“ übersetzt werden. Aus diesem Grund wird im folgenden häufiger der Begriff Informationsextraktion synonym mit Informationsgewinnung verwendet.

²Der Ausschnitt wurde dem Online-Angebot der Süddeutschen Zeitung entnommen.

Die IG Metall macht ihre Ankündigung wahr – und stellt heute im Fall Siemens gleich an zwei Orten Strafanzeige gegen Unbekannt. Es geht um die Organisation AUB. Die Gewerkschaft wirft dabei dem Konzern die Begünstigung der dort aktiven arbeitgeberfreundlichen Betriebsräte vor.

Beispiel 3.1: Unstrukturierter Text

Jegliche Information (z.B. von welchen Organisationen bzw. Personen die Rede ist) befindet sich innerhalb dieses Freitextes. Ohne sprachwissenschaftliche Kenntnisse können diese Informationen nicht genau identifiziert und extrahiert werden.

Den kompletten Gegensatz zu diesen unstrukturierten Texten stellen die strukturierten dar.

3.1.2 Strukturierter Text

Ein strukturierter Text folgt einem festen definierten Format – einem Schema. Wenn anstelle von reinen Textdokumenten allgemein „Daten“ betrachtet werden, zählen beispielsweise Daten in relationalen Datenbanken zu den strukturierten Daten [Eik99]. Aber auch Textdateien können strukturiert sein, wenn sie ein durchgehend striktes Format einhalten, wie das folgende Beispiel zeigt:

```
<person>
  <name>
    <vorname>Daniel</vorname>
    <nachname>Brügge</nachname>
  </name>
  <beruf>Student</beruf>
</person>
```

Beispiel 3.2: Strukturierter Text

Es ist zu erkennen, dass jede Information innerhalb des Textes durch eine Struktur klar abgegrenzt und dadurch erkennbar ist.

Eine Art „Mittelweg“ zwischen den strukturierten bzw. unstrukturierten Texten stellen die so genannten semi-strukturierten Texte dar.

3.1.3 Semi-strukturierte Texte

Ein semi-strukturiertes Textdokument verfügt zwar teilweise über eine Struktur, diese sagt aber häufig nichts über die Semantik der Daten aus, sondern dient u.a. nur gestalterischen Zwecken (z.B. Tabellen in Webseiten).

Zudem hält sich semi-strukturierter Text nicht an grammatikalische Regeln und ist häufig in einem telegrammartigen Stil geschrieben [Eik99]. Ausschnitt 3.3 zeigt einen Teil eines semi-strukturierten Dokumentes:

```
<table>
  <tr>
    <td>Beschreibung</td>
    <td>Entwurf und Implementierung eines verteilten MAS zur
      <b>Informationsgewinnung</b> im Rahmen einer mobilen Applikation
      von<i>Daniel Brügge</i></td>
  </tr>
</table>
```

Beispiel 3.3: Semi-strukturierter Text

Zu den bekanntesten semi-strukturierten Textdokumenten zählen Webseiten, welche im restlichen Kapitel Grundlage weiterer Untersuchungen sein werden, da sie die Hauptinformationsquellen für das spätere System darstellen.

Bemerkung: In einigen Publikation werden auch Webseiten zu den strukturierten Textdokumenten gezählt, wenn die gesuchten Informationen durch ihre Anordnung oder durch Muster klar erkennbar und dadurch extrahierbar sind [SS05, Yin06, HD98, Eik99]. In dieser Thesis zählen diese nicht zu strukturierten Textdokumenten.

Der Grund hierfür ist der, dass eine Webseite zwar beispielsweise in validem XML (z.B. in Form von XHTML) vorliegen kann und zudem noch alle Informationen in einer Liste klar getrennt enthält. Dennoch befinden sich in diesem Textdokument zahlreiche Konstrukte, die nichts mit den eigentlichen Daten zu tun haben und nur Formatierungszwecken dienen oder zum Grundgerüst der Webseite gehören.

```
<html>
<head>...</head><body>
<p><b>Ort:</b>München, Lothstraße<br><b>Zeit:</b>12.03.2007, 13:00</p>
<p><b>Ort:</b>...<br><b>Zeit:</b>...</p>
...
</body></html>
```

Beispiel 3.4: Semi-strukturierter Text der intern die Informationen in einer Art Struktur auflistet

In Beispiel 3.4 sind zwar alle Informationen der Reihe nach in einer strukturierten Form präsentiert, aber die Elemente, die vorhanden sind, dienen letztendlich Darstellungszwecken und nicht datenstrukturellen Zwecken.

Aus diesem Grund werden im folgenden alle Webseiten zu den semi-strukturierten Texten gezählt. Genauer gesagt, werden Webseiten in dieser Arbeit als eine Kombination zwischen unstrukturierten und semi-strukturierten Texten betrachtet. Beispielsweise kommt es in vielen Fällen vor, dass eine gesuchte Information innerhalb eines natürlich-sprachlichen Textes steht, welcher wiederum in die Semistruktur der Webseite eingebettet ist.

Der Vollständigkeit halber muss noch erwähnt werden, dass es auch Webseiten gibt, in denen die Informationen in strukturiertem XML vorliegen und die Konstrukte, die die Präsentation an den Endbenutzer vornehmen, ausgelagert sind und beispielsweise in einem XSL (*Extensible Stylesheet Language*) Stylesheet vorliegen. Ein *User Agent*³ wie der Webbrowser erstellt nun aus den reinen Daten mit Hilfe der Stylesheet-Anweisungen die entsprechend für den Menschen formatierte Darstellung [Bre03].

Da aber nur wenige Browser diese Funktionalität anbieten und derartige Webseiten dementsprechend selten vorkommen und keine Quelle in dieser Form Untersuchungsobjekt der Thesis ist – wie später noch gezeigt wird –, werden sie in der Arbeit nicht näher untersucht, können aber vom späteren System ebenfalls verarbeitet werden⁴.

Nachdem die Strukturbegriffe vorgestellt worden sind, wird sich im nächsten Abschnitt der eigentlichen Informationsextraktion zugewendet und einige Konzepte vorgestellt.

³In diesem Fall wird jede Client-Applikation die über HTTP auf die Internetquelle zugreifen kann als *User Agent* bezeichnet.

⁴Dabei wird aber nur das XML-Dokument genutzt und das XSL-Stylesheet ignoriert.

3.2 Die Informationsextraktion

Informationsextraktion (IE) ist ein Forschungsgebiet welches Ende der 1980er Jahre auf Initiative der DARPA (*Defense Advanced Research Projects Agency*⁵) vor allen Dingen im Rahmen der von ihr organisierten *Message Understanding Conference*⁶ (MUC) maßgeblich entstanden und vorangebracht worden ist [AI99].

Im Gegensatz zum Bereich des *Information Retrieval* (IR) konzentriert sich die IE nicht auf das Auffinden relevanter Dokumente innerhalb größerer Sammlungen von Dokumenten, sondern auf das Extrahieren relevanter Informationen aus Dokumenten. Beide Gebiete, IR sowie IE, lassen sich aber miteinander verbinden und ergänzen sich dadurch bei der Verarbeitung von Texten [GW98].

Obwohl es zahlreiche Definition von IE gibt, kann grundsätzlich gesagt werden, dass jeder Prozess zur Informationsextraktion gezählt werden kann (im Kontext der Verarbeitung von Textdokumenten), der in einem oder mehreren Texten relevante Informationen auffinden, extrahieren und miteinander kombinieren kann, so dass diese später in einem strukturierten Format vorliegen und beispielsweise zur weiteren Verarbeitung in einer Datenbank genutzt werden können [CL96].

Das ursprüngliche Arbeitsumfeld bei der Informationsextraktion stellen, wie beim Information Retrieval und auch beim artverwandten Gebiet der *Text Understanding Systeme*⁷, natürlich-sprachliche Texte dar.

Metriken Um IE-Systeme bewerten und vergleichen zu können, wurden im Rahmen der Message Understanding Conferences einige Metriken definiert, die als Grundlage die schon des öfteren erwähnten „relevanten Informationen“ nutzen.

Bei diesen Metriken wurde sich an den Standardmaßen des Information Retrieval, den so genannten *recall*- und *precision*-Werten, orientiert. Im IE gibt dabei der **recall**-Wert (R) an, wie viele korrekte bzw. relevante Informationen aus dem Dokument extrahiert worden sind – im Verhältnis zu allen im Dokument enthaltenen relevanten Informationen. Wenn beispielsweise in einem Text 10 relevante Informationen vorkommen und diese 10 auch extrahiert werden, ist der recall-Wert am höchsten. Ein Wert von 1.0 ist hierbei also erstrebenswert.

$$R = \frac{\text{Anzahl der korrekt extrahierten Informationen}}{\text{Anzahl der im Text vorhandenen relevanten Informationen}}$$

Der **precision**-Wert (P) gibt hingegen an, wie viele der von der Extraktion gelieferten Informationen wirklich relevant sind. Das heißt, ein Extraktionsalgorithmus der beispielsweise 1000 Informationen liefert von denen aber letztendlich nur 2 relevant bzw. korrekt sind, ist als nicht präzise anzusehen.

$$P = \frac{\text{Anzahl der korrekt extrahierten Informationen}}{\text{Anzahl der insgesamt extrahierten Informationen}}$$

Die beiden Metriken können auch kombiniert werden, um einen besseren Vergleich zu erlangen. Eine Möglichkeit ist dabei der so genannte *F-Measure* Wert, der in [VR79] vorgestellt wurde:

⁵Die DARPA ist die zentrale Forschungs- und Entwicklungsorganisation innerhalb des Verteidigungsministeriums der Vereinigten Staaten von Amerika. Siehe auch <http://www.darpa.mil/>

⁶Insgesamt wurden sieben Konferenzen abgehalten, wobei die letzte im Jahr 1997 stattfand. Der Mittelpunkt dieser Konferenzen bildeten dabei Evaluierungen von IE-Systemen. Jede Konferenz hatte dabei ein anderes Thema, das von den IE-Systemen verarbeitet werden musste [GS96].

⁷Text Understanding oder auch *Story Understanding* Systeme haben im Gegensatz zum IE das Ziel, ein komplettes Verständnis des zugrunde liegenden Textes zu erhalten. Bei IE liegt der Fokus auf bestimmten Informationen [Ril99].

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

β gibt hierbei an, wie stark der recall-Wert R gegenüber dem precision-Wert P bevorzugt werden soll. Eine Gleichgewichtung von beiden würde einen β von 1 nach sich ziehen [Eik99].

Weiter soll auf den F-Measure auch nicht mehr eingegangen werden, aber [Eik99, VR79] bieten einige weiterführende Informationen.

Nach dieser kurzen Einführung in die Informationsextraktion soll nun auf Ansätze eingegangen werden, die sich auf die Erstellung von Extraktionsregeln beziehen.

3.2.1 Grundlegende IE-Ansätze (manuell vs. automatisch)

Grundsätzlich basiert jedes IE-System bzw. fast jede Teilkomponente eines solchen, auf der Anwendung von Extraktionsregeln (eng. *extraction rules*) auf einem Text [Sod99]. Die Erstellung dieser Regeln kann dabei manuell durch eine oder mehrere Personen erfolgen, die über das spezifische Fachwissen verfügen, um diese Regeln zu erstellen [AI99].

Dieser Prozess ist ein inkrementeller Prozess, bei dem die manuell erstellten Regeln auf einem Testtext angewendet werden und ggf. von der Person nachbearbeitet werden, bis ein gewünschtes Resultat erreicht wird.

Dieser Ansatz wird auch als **Knowledge Engineering** bezeichnet und die Person dementsprechend als *Knowledge Engineer*. Knowledge Engineering zieht meistens einen großen Aufwand nach sich, da es einige Zeit dauern kann, bis die kompletten Regeln erstellt worden sind.

Neben dem eben genannten manuellen Vorgehen zur Definition der Regeln, stellt der **Automatic Training** Ansatz die automatische Variante dar. Hierbei ist zwar kein Knowledge Engineer erforderlich, um die Regeln zu erstellen, aber trotzdem muss – wenigstens am Anfang des Prozesses – eine Person mit Wissen über die Domäne und über die gesuchten Informationen verfügbar sein. Zwei unterschiedliche Gründe gibt es hierfür, die kurz erläutert werden.

1. Die gesuchten Informationen (z.B. Ortsangaben) müssen in einem Korpus von Testtexten – den Trainingsdaten – am Anfang speziell markiert bzw. annotiert werden. Auf Grundlage dieser Annotationen wird dann ein Trainingsalgorithmus ausgeführt, welcher versucht, daraus Extraktionsregeln abzuleiten.
2. Während der Trainingsphase wird der Benutzer regelmässig gefragt, ob die gefundenen Informationen korrekt sind. Die Annotation erfolgt hierbei also aufgrund der Hypothesen des Algorithmus.

Die eben vorgestellten IE-Ansätze bzw. Paradigmen gelten sowohl bei un-, semi- und strukturierten Texten. Bei den beiden Letzteren nutzen die erstellten Regeln dabei hauptsächlich die vorkommenden Strukturelementen. Dieses ist bei unstrukturierten natürlich-sprachlichen Texten nicht möglich, weswegen bei diesen noch einige andere Verarbeitungsschritte benötigt werden [Sod99].

3.2.2 IE bei natürlich-sprachlichen Texten

Natürlich-sprachliche Texte waren, bevor semi-strukturierte Textdokumente immer mehr an Bedeutung gewonnen haben, zentrales Untersuchungsobjekt der Informationsextraktion. Heutzutage spielen die beiden Strukturarten eine gleichbedeutende Rolle und auch in dem zu entwickelnden System werden

beide Bereiche behandelt, da teilweise auch unstrukturierte Texte die gesuchten Informationen enthalten (siehe 3.1.3). Aus diesem Grund sollen im folgenden einige grundlegende Ansätze der IE im Bereich der natürlich-sprachlichen Texte eingeführt werden.

Im vorherigen Abschnitt wurde schon erwähnt, dass bei natürlich-sprachlichen Texten einige zusätzliche Schritte erforderlich sind, damit Extraktionsregeln auf den Texten effizient arbeiten können. Diese „Vorbereitungsschritte“ haben sich im Laufe der Zeit im Bereich der Informationsextraktion herauskristallisiert und die traditionellen NLP (*Natural Language Processing*) Vorgehensweisen⁸ weitestgehend ersetzt bzw. verfeinert und dadurch eine Art Standardarchitektur für IE-Systeme geschaffen [Car97]. Diese Architektur ist zwar nicht verpflichtend für ein IE-System, kommt aber bei vielen Systemen vor, die natürlich-sprachliche Texte verarbeiten sollen.

Standardarchitektur

Die Standardarchitektur für die Informationsextraktion von natürlich-sprachlichen Texten kann in einige Verarbeitungsschritte aufgeteilt werden, die in fast allen IE-Systemen anzutreffen sind. Dabei kann es von System zu System aber auch kleinere Unterschiede geben. Beispielsweise teilt [AI99] ein IE-System in die vier großen Hauptkomponenten

1. Tokenisierung (*Tokenization*),
2. Morphologische und lexikalische Verarbeitung (*Morphological and Lexical Processing*),
3. Syntaktische Analyse (*Syntactic Analysis*) und
4. Domänenanalyse (*Domain Analysis*)

auf. Diese vier primären Schritte können je nach Bedarf noch um zusätzliche Schritte erweitert werden. So kann Schritt (1) noch um eine *Word Segmentation* Komponente, (2) um einen *Part of Speech* bzw. *Word Sense* Tagger erweitert werden.

Bei der syntaktischen Analyse ist zusätzlich noch ein vollständiges Parsen des Textes möglich (*Full Parsing*) und die Domänenanalyse kann noch um eine Merging-Komponenten zum Zusammenfassen von Teilergebnissen oder um eine Komponente zur Auflösung von Koreferenzen ergänzt werden (siehe Abbildung 3.1).

Diese Schritte sind, wie oben schon gesagt wurde, nicht für jedes System gleich. Zum Beispiel wird bei [Car97] die Tokenisierung und das Taggen in einem Schritt zusammengefasst, wobei hier noch nicht das Word Sense Tagging enthalten ist. Dieses folgt im Rahmen des syntaktischen Analyse und findet vor, während oder nach dieser statt. Danach folgt der so genannte „Extraction“-Schritt, der eine domänen-spezifische Verarbeitung vornimmt und Relationen zwischen Wörtern oder Wortgruppen erkennt. Ein Merging-Schritt und die Erzeugung der relevanten Informationsgruppen beenden die Verarbeitung.

Im folgenden sollen nun die wichtigen Schritte erläutert werden, wobei Bezug auf die Architektur von [AI99] genommen wird.

Tokenisierung Bei der Tokenisierung wird der vorliegende Text in Sätze und danach die Sätze in die einzelnen Wörter aufgeteilt. Dieses kann anhand vorhandener Leerzeichen bzw. Satzzeichen erfolgen⁹. Ein Beispiel einer Tokenisierung für den Satz „Hello World!“ unter Verwendung des *Natural Language*

⁸Traditionelle Ansätze haben jeden Satz eines Textes komplett syntaktisch und semantisch analysiert und auf diesen Ergebnissen Diskursanalysen durchgeführt.

⁹Sprachen wie Japanisch oder Chinesisch erfordern die spezielle Aufteilung der einzelnen Wörter (*word segmentation*)

Toolkits (NLTK) ist in Beispiel 3.5 zu sehen (hierbei wurde anhand von Leerzeichen die Tokenisierung vorgenommen) [BL04].

[<Hello>, <World!>]

Beispiel 3.5: Tokenisierung unter Verwendung des Natural Language Toolkits [BL04]

Morphologische und lexikalische Verarbeitung In diesem Hauptschritt stellen eigentlich die Erweiterungsschritte den interessanten Teil dar, aber zuvor wird eine morphologische bzw. lexikalische Analyse auf dem Text durchgeführt.

Der morphologische Arbeitsschritt ist hierbei abhängig von der Sprache in der der Text vorliegt. Beispielsweise ist bei englischen Texten keine morphologische Analyse notwendig, während sie bei deutschen Texten unabdingbar ist.

Die lexikalische Analyse überprüft Wörter anhand von Wortlisten und kann somit schon einige Zuordnungen vornehmen, wobei hier extrem große Listen eher einen nachteiligen Effekt haben, da dort viele mehrdeutige Wörter auftauchen können.

Neben diesen beiden Initiierungsschritten stellt das Tagging (in der Bedeutung von „markieren“) einen der interessantesten Teile dar. Dieses kann abhängig von dem verwendeten Tagger mehrere Zwecke erfüllen. Zum einen kann beispielsweise ein POS-Tagger (*Part of Speech*) eingesetzt werden, um die Wortarten eines Textes zu markieren, um dadurch nachfolgende Analysen zu erleichtern [AI99]. Beispiel 3.6 zeigt den durch einen POS-Tagger (hier: Trigrams'n'Tags oder TnT) markierten Satz „Der

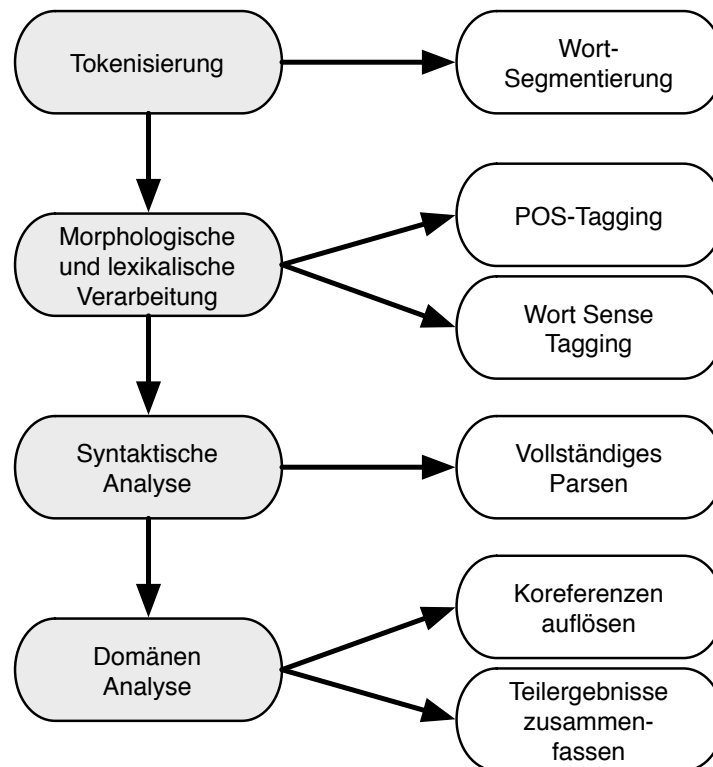


Abbildung 3.1: Standard-Verarbeitungsschritte eines IE-Systems zur Verarbeitung von natürlich-sprachlichen Texten nach [AI99]

Mandolinen-Club Falkenstein und der Frauenchor aus dem sächsischen Königstein gestalten die Feier gemeinsam.“

Der	ART
Mandolinen-Club	NN
Falkenstein	NE
und	KON
der	ART
Frauenchor	NN
aus	APPR
dem	ART
sächsischen	ADJA
Königstein	NN
gestalten	VVINF
die	ART
Feier	NN
gemeinsam	ADJD
.	\$.

Beispiel 3.6: POS-Tagger Ausgabe [Bra98]

Eine vor allem für diese Arbeit interessantere Methode des Taggings ist das Zuweisen von semantischen Klassen zu einzelnen Bestandteilen eines Textes, wobei dieser Prozess häufig als *word sense tagging* [AI99] oder als *semantic tagging* [BAJ⁺01] bezeichnet wird.

Mit „semantischen Klassen“ sind in diesem Zusammenhang beispielsweise Uhrzeiten, Datumsangaben, ausgeschriebene Nummern und besonders Eigennamen von Personen, Firmen, Organisationen, Produkten oder Orten gemeint, wobei speziell Ortsangaben für diese Thematik interessant sind. Zudem können systemabhängig auch noch zusätzliche Klassen hinzukommen (z.B. Typ „Adresse“ wie Telefonnummer, E-Mail, URL etc.) [MTU⁺01].

In der Literatur werden diese Klassen auch als „Entitätstypen“ bezeichnet und eine konkrete Ausprägung solch eines Typs dementsprechend als eine Entität bzw. benannte Entität (eng. *named entity*).

Dieser Prozess der Erkennung bzw. Klassifizierung wird dabei auch **Named Entity Recognition** genannt (NER).

Von all den oben genannten Klassen stellen vor allen Dingen die Eigennamen die Tagging-Komponente vor Probleme.

Bei Uhrzeiten, Datumsangaben gibt es zwar auch zahlreiche Möglichkeiten, dass ein Wort- bzw. eine Wortkette in diese Klasse fällt, aber hier können Strukturen zu einer relativ genauen Erkennung und dadurch einer Zuordnung führen (z.B. deuten drei Zahlen, die durch Dezimalpunkte „.“ getrennt sind und wobei die erste Zahl im Intervall von 1-30, die zweite im Intervall 1-12 liegt und die dritte Zahl eine vierstellige Zahl ist, auf ein Datum hin).

Bei der Klasse der Ortsnamen gibt es die Möglichkeit, alle bekannten Ortsnamen in einem Lexikon zu speichern, da diese nicht großen Veränderungen unterworfen sind und relativ konstant bleiben. Bei den Eigennamen handelt es sich hingegen um eine sehr flexible Klasse, da so gut wie jeder Teil eines Textes einen Eigennamen darstellen kann. Zudem weisen Eigennamen auch noch von Sprache zu Sprache Unterschiede auf, wie beispielsweise die Anordnung von Vor- und Nachnamen [AI99].

Der Entwurf von IE-Komponenten zur Erkennung von Eigennamen stellt also eine größere Schwierigkeit dar und es gibt unterschiedliche Ansätze dieses Problem zu lösen. Dabei gibt auch hier wieder die Möglichkeit (siehe 3.2.1), dass die Umsetzung manuell über die Definition von Regeln oder durch einen automatischen Trainingsansatz erfolgt.

Bei dem manuellen Ansatz wird vor allem mit Lexika gearbeitet, in denen beispielsweise tausende von Firmen-, Vor- und Nachnamen oder Orte aufgelistet werden. Danach werden Texte untersucht und das Auftreten der gesuchten Eigennamen analysiert. Anhand dieser Analyse werden Regeln erstellt, die

z.B. eindeutige Muster erkennen und nutzen können (z.B. die Kombination eines Wortes mit der darauf folgenden Zeichenkette „GmbH“ lässt darauf schließen, dass es sich um einen Firmennamen handelt). Auch Groß- und Kleinschreibungen können für die Erkennung verwendet werden.

Die Entwicklung der Regeln ist wie immer ein inkrementeller Prozess, bei dem der Knowledge Engineer seine Regeln Schritt für Schritt optimiert und ggf. erweitert, bis am Ende eine hohe Erkennungsrate erreicht wird [AI99].

Der automatische Ansatz basiert nicht auf der Verwendung von Wortlisten und der manuellen Erstellung von Regeln. Vielmehr werden Texte in mehreren Phasen durchlaufen in denen ein Benutzer die Eigennamen speziell annotiert. Diese Informationen werden einem Modell zugeführt, anhand dessen später errechnet wird, ob ein Wort oder eine Wortkette ein Eigenname ist.

Eine Möglichkeit solch eines Modells, welches in diesem Bereich häufig Verwendung findet, ist das **Hidden Markov Model** (HMM) – ein Wahrscheinlichkeitsmodell.

Das Hidden Markov Model weist dabei dem Auftreten von Wortsequenzen im Text Wahrscheinlichkeitswerte zu, welche die Zugehörigkeit zu einer Klasse repräsentieren. Diese Wahrscheinlichkeiten entstehen durch den Annotierungsprozess und werden dem Modell zugefügt, welches mit jeder Annotierung genauer wird.

Das Modell basiert dabei auf einem Zustandsautomaten, wobei die Zustände hier aber im Gegensatz zu einem normalen Markov Modell nicht direkt sichtbar sind, also unsichtbar bzw. *hidden* [AI99, ?].

In dem mit den Wahrscheinlichkeitsübergängen versehenen Modell werden nun maximale Wahrscheinlichkeitspfade bestimmt, welche Wortketten mit einer hohen Wahrscheinlichkeit den Klassen zuweisen können. Ein Algorithmus der solch einen Pfad bestimmen kann, ist beispielsweise der *Viterbi Algorithmus*, auf den hier aber nicht näher eingegangen werden soll.

Während beim eben beschriebenen Hidden Markov Modell Ansatz mit Wörtern als Eingabe gearbeitet wurde, gibt es auch die Möglichkeit, auf eine kleinere Einheiten runter zu gehen – z.B. auf einen zeichenbasierten Ansatz. [KSNM03] zeigt hierzu zwei unterschiedliche Markov Modelle, die diese Vorgehensweise nutzen und dadurch vor allem bei unbekanntem, noch nicht annotierten Wörtern, ein besseres Ergebnis liefern.

Allgemein gesehen, ist die Erkennung von Entitäten innerhalb von natürlich-sprachlichen Texten in fast jedem IE-System an irgendeiner Stelle enthalten, so dass der Entwurf entsprechender Komponenten Thema zahlreicher Arbeiten ist. In dieser Thesis werden NER-Techniken insoweit genutzt, dass sie in den späteren Entwurf an der Stelle einbezogen werden, an der andere Extraktionsmöglichkeiten nicht mehr gegeben sind.

Syntaktische Analyse Einen zweiten großen Verarbeitungsschritt in einem IE-System stellt die syntaktische Analyse dar. Hier werden Wortgruppen identifiziert und im Gegensatz zu traditionellen NLP-Ansätzen nur die Teile des Satzes geparkt, die auch für die Informationsextraktion relevant sind (eng. *partial parsing*)[Car97].

Domänen-spezifische Analyse Beim dem letzten Schritt, der Analyse mit Bezug auf die jeweilige Domäne in der die Informationsextraktion stattfindet, geht es vor allen Dingen darum, Koreferenzen aufzulösen und die Informationen zu extrahieren, die letztendlich relevant sind. Von einer „Koreferenz“ (eng. *coreference*) wird dabei gesprochen, falls sich mehrere Wörter bzw. Ausdrücke in dem Text auf dasselbe beziehen [Wik06, Car97], wie mit dem Beispiel 3.7 verdeutlicht werden soll.

Frau Müller wurde am 21.03. in München überfallen.
Dem Opfer wurden 40 Euro gestohlen.

Beispiel 3.7: Koreferenz

In diesem Beispiel bezieht sich der Ausdruck „Dem Opfer“ auf „Frau Müller“, also sind beide koreferent zueinander.

Das Thema der IE von natürlich-sprachlichen Texten stellt ein umfassendes Gebiet dar, von dem nur ein Bruchteil kurz vorgestellt wurde. Im folgenden Abschnitt wird sich nun Extraktionsansätzen von semi-strukturierten Texten zugewendet, da diese für das spätere System ausschlaggebend sind.

3.2.3 IE-Ansätze bei semi-strukturierten Texten

Die im vorherigen Abschnitt vorgestellten Möglichkeiten der Informationsextraktion innerhalb von natürlich-sprachlichen Texten sind auf semi-strukturierten Textdokumenten, wie beispielsweise Webseiten, nicht mehr allgemein übertragbar und anwendbar. Der Grund hierfür liegt darin, dass NLP-Methoden grammatikalisch vollständige und korrekte Texte erwarten [Eik99]. Konstrukte wie beispielsweise HTML-Elemente behindern diese Arbeit oder machen sie gar unmöglich. Zudem sind Webseiten, wie schon in 3.1.3 erwähnt wurde, häufig in einem telegrammartigen Stil geschrieben und erschweren NLP-Analysen damit zusätzlich die Arbeit.

Diese Gründe haben zu der Entwicklung von speziellen Werkzeugen geführt, die eine Extraktion aus semi-strukturierten Texten, speziell Webseiten, möglich machen sollen. Ansätze wie die Erstellung von **Wrappern** zur Extraktion von relevanten Daten aus Webquellen wurden dabei vor allem aus dem Datenbankumfeld beeinflusst [Myl01].

Bevor auf die Wrapper näher eingegangen wird, soll zuvor aber noch der Begriff der „idealen Extraktion“ eingeführt werden, um ein besseres Verständnis der folgenden Untersuchungen zu gewährleisten.

Die „ideale“ Extraktion

Fast alle Informationen, die in semi-strukturierten Dokumenten auftauchen sind dynamisch erstellte Inhalte, die zum größten Teil aus Datenbanken stammen. Diese Datenbank-Inhalte werden vom Webserver gelesen und als Webseite, Newsfeed, Webservice etc. an den User-Agent ausgeliefert (z.B. einen Webbrowser, Newsreader oder Softwareagenten).

Wie in dem Abschnitt über Textstrukturen erläutert wurde, handelt es sich bei den Daten in der Datenbank um strukturierte Daten, deren Struktur jedoch verloren geht, wenn sie in un- bzw. semi-strukturierter Textform in den Informationsquellen auftauchen.

Die Aufgabe einer Informationsextraktion ist es nun, die semi- bzw. unstrukturierten Daten aus einer Quelle zu lesen und ihnen wieder eine Struktur bzw. Bedeutung zu geben.

So wird z.B. aus einer Zeichenfolge in einer Quelle durch Hinzufügen der Semantik „Uhrzeit“ auf einmal ein Zeitpunkt. Oder es wird aus einem Wort in einer Quelle eine Ortsangabe durch Hinzufügen der Bedeutung „Ort“.

Die *ideale Extraktion* ist somit, wenn aus den un- bzw. semi-strukturierten Daten aus einem Text, Daten gewonnen werden, die genau die Datenstruktur haben, die auch in der Ursprungsdatenbank vorhanden war [Myl01].

Abbildung 3.2 zeigt eine ideale Extraktion anhand eines Beispiels. Dabei sind die ursprünglichen strukturierten Daten in einer Datenbank gespeichert und werden von einem Webserver aufbereitet und als Webdokument (hier: HTML) im Internet zur Verfügung gestellt. Die Extraktionskomponente liest nun

diese semi-strukturierten Daten und extrahiert genau die Datenstruktur, die auch in den Ursprungsdatenbank verwendet wurde. Dadurch ist eine ideale Extraktion erreicht worden.

Bei genau diesem Prozess der Informationsextraktion aus einer semi-strukturellen Quelle handelt es sich um das bereits erwähnte *wrapping* bzw. die entsprechende Komponente den *Wrapper* [CL01, BFG01a], welcher im nächsten Abschnitt näher erläutert wird.

Wrapper

Einführend kann gesagt werden, dass die Aufgabe eines Wrappers darin besteht, aus einer semi-strukturierten Informationsquelle, die Inhalte zu extrahieren, die relevant sind und diese danach in ein strukturiertes Datenformat zu übertragen, das eine spätere Verarbeitung zulässt [Eik99, KWD97, HD98].

Der Prozess des Wrappings ist seit Jahren Thema vieler Arbeit im Bereich der IE und weist dementsprechend teilweise unterschiedliche Interpretationen auf. Die ersten Wrapper-Ansätze sind davon ausgegangen, dass das Wrapping auf Ergebnisseiten (*result pages* bzw. *response pages*) von Suchanfragen angewendet werden soll.

Hierbei war es zum einen die Aufgabe des Wrappers, die Benutzer-Suchanfrage selbst – den *Query* – so umzuwandeln, dass dieser zu einem Formular-Request für eine Suchmaschine wird. Die resultierenden Ergebnisseiten sollten dann ebenfalls vom Wrapper in das strukturierte Format gebracht werden; heißt, es sollten genau die Informationen aus den Ergebnislisten extrahiert werden, die durch die ursprüngliche Benutzeranfrage verlangt wurden [CBC97].

Andere Wrapper-Interpretationen setzten ebenfalls eine Anfrage voraus, die aber nicht durch ein Wrapper bearbeitet werden musste. Hierbei dienten nur die Ergebnisseite als Eingabe für den Wrapper [Kus00].

In dieser Arbeit wird ebenfalls der Begriff des Wrappers für das spätere Extraktionskonzept genutzt. Zudem hält sich dabei die Definition an die grundlegende Aufgabe, die ein Wrapper erfüllen soll, und wie es am Anfang dieses Abschnittes erwähnt wurde und in Abbildung 3.3 dargestellt wird.

Zusätzlich wird es auch nicht als Ziel des Wrappers angesehen, eine möglichst generische Anfragesprache zur Extraktion von Daten zu definieren, wie es in anderen Arbeiten (z.B. [QRS⁺95]) der Fall ist.

Nachdem auf die Begriffsdefinition des Wrappers eingegangen worden ist, wird im folgenden die Konstruktion solcher Wrapper betrachtet. Hierbei kann, in Anlehnung an die grundlegenden IE-Ansätze aus Abschnitt 3.2.1, wieder zwischen

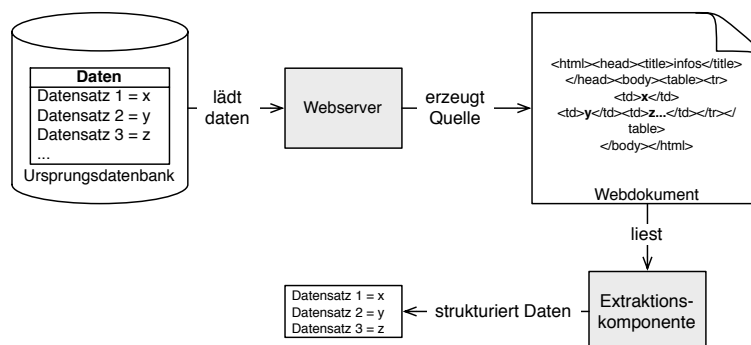


Abbildung 3.2: Ideale Extraktion von Daten

- manueller Wrapper-Konstruktion und
- automatischer Wrapper-Konstruktion

unterschieden werden [KWD97], wobei zusätzlich auch eine semi-automatische Konstruktion möglich ist [Eik99].

Manuelle Wrapper-Erstellung Im allgemeinen wird unter einer „manuellen Erstellung“ verstanden, dass bei der Wrapper-Konstruktion ein Knowledge Engineer erforderlich ist, der die semi-strukturellen Texte analysiert und auf Basis dieser, Regeln zur Extraktion erstellt.

Bei vielen manuellen Ansätzen ist sogar das explizite Schreiben von Programmcode für Extraktionsroutinen erforderlich, wobei der Code hierbei für jede Quelle neu angepasst werden muss [Eik99, Bre03].

Das Problem bei der manuellen Vorgehensweise ist nicht nur, dass die Erstellung zeitaufwendig und fehleranfällig ist, sondern auch wartungs-intensiv sein kann. Letzteres tritt auf, falls die Textdokumente häufig Änderungen unterlegen sind, welche dann das ständige manuelle Nachbessern der Regeln bzw. des Programmcodes nach sich ziehen [KWD97].

Semi-automatische Erstellung Bei der semi-automatischen Erstellung wird der Benutzer von einer Applikation bei der Erstellung der Regeln unterstützt. Beispielsweise können relevante Teile innerhalb des Textes markiert werden und die Applikation erstellt aufgrund dessen die Regeln.

Der Vorteil ist der, dass kein Knowledge Engineer mit Expertenwissen (z.B. Programmierkenntnissen) mehr benötigt wird, sondern einfaches Wissen in der jeweiligen Domäne ausreicht. Ein System welches semi-automatische Erstellung ermöglicht, ist beispielsweise Lixto [BFG01b, BFG01a].

Problematisch bei diesem Ansatz ist ebenfalls, dass für jedes neue Textdokument bzw. für jede neue Webseite dieser Vorgang wiederholt werden muss, da das System die Struktur einer neuen unbekanntenen Seite nicht automatisch verarbeiten kann, sondern immer die Hilfe von aussen benötigt, um die relevanten Daten zu finden [Eik99].

Automatische Erstellung Die Ansätze, die eine automatische Wrapper-Generierung erlauben, sollen vor allem die Probleme beseitigen, die bei manueller sowie semi-automatischer Erstellung auftreten – nämlich das ständige Nachbessern bei Änderungen in der Quelle (hoher Wartungsaufwand), sowie die Notwendigkeit der Wrapper-Neuerstellung bei neu hinzukommenden Quellen.

Eine Technik, die bei der automatischen Wrapper-Erzeugung in vielen Fällen genutzt wird, ist die so genannte **Wrapper Induktion** (*wrapper induction*), eine Methode, die auf induktivem Lernen basiert [Bre03, Eik99].

Hierbei werden, ähnlich wie bei der semi-automatischen Erstellung, relevante Informationen in Texten markiert. Diese Markierungsphase wird dabei auf einem Korpus von Beispieltexten – den Trainingsdaten – durchgeführt. Anhand der Texte und der markierten Informationen lernt nun ein Induktionsalgorithmus die passenden Extraktionsregeln und erstellt den Wrapper.

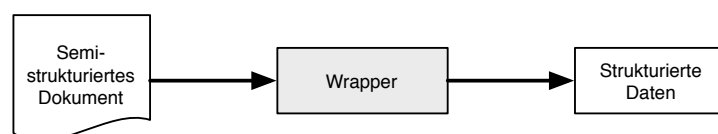


Abbildung 3.3: Aufgabe eines Wrappers im Kontext dieser Arbeit

3.2.4 Unterschiede in diesem System

Es gibt zwar einige Gemeinsamkeiten zwischen den im letzten Abschnitt erwähnten Definitionen bzw. Techniken und den letztendlich im späteren System eingesetzten Methoden, aber es können doch Unterschiede festgestellt werden.

Zunächst einmal setzen die zuletzt genannten Wrapper-Ansätze voraus, dass die Informationen in kleinen, durch Delimiter¹⁰ begrenzten Bereichen innerhalb der Semistruktur vorkommen, also nicht innerhalb eines natürlich-sprachlichen Textes liegen.

In dieser Thesis wird aber davon ausgegangen (wie die späteren Quellenanalysen noch zeigen werden), dass in manchen semi-strukturierten Quellen die Informationen innerhalb von Freitexten enthalten sind und zusätzlich noch in begrenzten Bereichen liegen können. Somit stellen die Wrapping-Ansätze neben den NLP-Methoden nur einen Teil dar.

Trotzdem wird das komplette Paket der Extraktion in dieser Arbeit als *Wrapper* bezeichnet.

Das vorliegende IE-System baut zudem auf die manuelle Erzeugung von Extraktionsregeln durch Experten auf und hat nicht zum Ziel, diese Regeln möglichst automatisch zu erzeugen. Vielmehr ist es das Ziel, dass ein unter Realbedingungen funktionierendes System entsteht, welches alle Informationsquellen verarbeiten kann, die relevante Informationen enthalten.

Hier einen automatischen Ansatz der Regelerstellung zu wählen, ist bei der heutigen Struktur der meisten Informationsquellen (speziell Webseiten) eine Aufgabe, die sehr komplex und nahezu unmöglich ist. Zudem haben beispielsweise die induktiven Ansätze den Nachteil, dass sie eine Lernphase benötigen, die Zeit kostet [Bre03]. Das System soll aber möglichst schnell neue Quellen aufnehmen können und das wäre mit einer Wrapper-Induktion nur schwer möglich.

Im folgenden soll nun speziell auf die Informationen eingegangen und die Quellen analysiert werden, die für das spätere System relevant sind. Es werden dabei Strukturen herausgearbeitet, die für den weiteren Entwurf notwendig sind und dem Verständnis dienen sollen.

3.3 Projektrelevante Informationen: Ereignisgruppe

Da diese Thesis speziell im Rahmen der mobilen Anwendung track-u entstanden ist, kümmert sich das System auch um ganz spezielle Informationen, die gewonnen werden sollen und stellt keinen komplett generischen Ansatz für die IE dar – ist also sehr domänen-spezifisch.

Projektrelevante Informationen sind hierbei Daten über Ereignisse, die stattgefunden haben oder stattfinden werden. Ein Ereignis definiert sich in diesem Zusammenhang durch

1. einen **Ort** (ein Ort definiert sich dabei durch eine Strasse, eine Hausnummer und eine Stadt),
2. einen **Zeitpunkt** (als Zeitpunkt wird die Kombination von Tag, Monat und Jahr gesehen. Uhrzeiten werden in dieser Arbeit vernachlässigt, können aber bei Bedarf ergänzt werden) ab dem es stattfindet und zusätzlich
3. durch eine **Beschreibung**, die das Ereignis näher charakterisiert.

Diese drei Informationen bilden zusammen eine Gruppe, die so genannte *Ereignisgruppe*, die eine aussagekräftige Komplet-Information bildet und Ziel der Informationsextraktion ist. Ein Beispiel für solch eine Gruppe ist in Tabelle 3.1 dargestellt.

¹⁰Ein Delimiter ist ein Begrenzungszeichen wie beispielsweise ein HTML-Element wie z.B.

Ort	Lothstraße, München
Zeit	01. April 2007
Beschreibung	Überfall auf die Bäckerei xy

Tabelle 3.1: Beispiel einer Ereignisgruppe

Wie gesehen werden kann, ist es nicht das Ziel der Thesis, eine „ideale Extraktion“ zu erreichen wie sie auf Seite 15 definiert wurde. Vielmehr ist es ausreichend und als „ideal“ zu bezeichnen, wenn aus einer Quelle alle dort vorhandenen Ereignisgruppen vollständig extrahiert werden können. Andere Informationen können vernachlässigt werden.

Zudem kann eine Art von Priorität der zu extrahierenden Informationen festgelegt werden. So sind die Ortsinformationen einer Ereignisgruppe am wichtigsten zu bewerten, da sie später vom track-u System genutzt werden (in Form von Geokoordinaten, wie im Systementwurf noch näher gezeigt wird).

Zeit-Informationen können in manchen Fällen ebenfalls vernachlässigt und vom System u.U. selbst ergänzt werden (z.B. durch Einfügen der aktuellen Zeit beim Extrahieren der Information).

Die Beschreibung muss in einigen Fällen ebenfalls nicht aufwendig gesucht werden, da hier umliegender Text hergenommen werden kann, falls keine explizite Beschreibung vorhanden ist.

3.4 Typen von Informationsquellen

Die Informationen, die später in das System einfließen sollen, entstammen unterschiedlichsten Quellen. Die Gemeinsamkeiten dieser Quellen sind

1. die Erreichbarkeit im *World Wide Web* (WWW) per URL mittels HTTP (*Hypertext Transfer Protocol*) und
2. das Vorliegen im Textformat.

Andere Medienformate wie beispielsweise gesprochener Text oder andere Erreichbarkeiten wie z.B. über E-Mail-Nachrichten sind zwar Thema diverser Forschungsarbeiten im Bereich der Informationsextraktion [MTU⁺01], werden aber in dieser Arbeit nicht weiter behandelt. Stattdessen konzentriert sich das System auf Quellen, die entweder in Form einer Webseite, eines Newsfeeds oder über einen Webservice¹¹ angeboten werden.

Um was es sich bei einer Webseite handelt, soll hier nicht näher erläutert werden. Es soll nur soviel gesagt werden, dass wenn im weiteren von „Webseiten“ die Rede ist, über WWW erreichbare Textdokumente gemeint sind, welche in Form von Markup-Sprachen wie beispielsweise HTML oder XHTML (bzw. XML) vorliegen¹².

Bei den, in den letzten Jahren in Mode gekommenen, Newsfeeds bzw. *Web-Feeds* handelt es sich um XML-Textdokumente, die ebenfalls über das WWW zugänglich sind. Sie dienen dazu, Inhalte anzubieten, ohne die ganzen Designkonstrukte zu nutzen, die in Webseiten häufig zu finden sind. Dieses „Anbieten“ wird auch als *Syndication* bezeichnet. Es gibt unterschiedliche XML-Formate für Newsfeeds

¹¹Webservices werden nur sporadisch in dieser Arbeit behandelt, da keine Beispielquellen gefunden worden sind, die hätten untersucht werden können. Nichtsdestotrotz werden sie in einigen Abschnitten der Vollständigkeit halber ebenfalls erwähnt.

¹²Für detaillierte Informationen wird auf die Seiten vom W3C unter <http://www.w3.org/> verwiesen. Dort finden sich zu allen Markup-Sprachen zahlreiche Informationen.

wobei einige eigene XML-Strukturen nutzen (Atom, RSS 2.0, RSS 0.91) und andere auf spezifizierten Standards wie RDF (*Resource Description Framework*) basieren (RSS 0.9 und 1.0) [Wik07c].

Nach dieser kurzen Übersicht über die unterschiedlichen Arten von Quellen, sollen im weiteren Beispielquellen untersucht werden, um dadurch für das System anwendbare Regeln definieren zu können.

3.4.1 Analyse anhand bestehender Beispielquellen

Dieser Abschnitt untersucht einige für die spätere Informationsgewinnung repräsentative Quellen in Bezug auf deren Aufbau und Struktur.

Die Kriterien, die die Untersuchung einbezieht, werden im folgenden kurz tabellarisch aufgelistet und erläutert:

Tabelle 3.2: Kriterien der Quellenbewertung

Kriterium	Beschreibung
Typ	Gibt den Typ der Quelle an (Webseite, Webservice oder Newsfeed)
Auszeichnungssprache	Mögliche Werte sind HTML oder XML, wobei bei XML noch die Unterklassifizierung in XHTML, RSS, RDF, Atom und andere XML-Formate vorgenommen wird.
Syntaxvalidität	Gibt an, ob die Quelle über eine valide Syntax verfügt. Als Kontrolle werden hierbei die offiziellen Validatoren des W3C (http://validator.w3.org/) und anderer Organisationen (für RSS z.B. Sam Ruby's RSS-Validator unter http://feedvalidator.org/) genutzt. Die Syntaxvalidität ist insofern interessant, da sie später auch darüber entscheiden kann, welche Methoden zur Extraktion angewendet werden können.
Datenstruktur	Gibt den Strukturierungstyp an (un/semi/-strukturiert).
Informationsverteilung	Beschreibt die Verteilung der gesuchten Information in der Informationsquelle. Dieses ist wichtig, da u.U. mehrere Schritte notwendig sind, um die Information vollständig zu finden.
Vollständigkeit	Gibt an, ob die Quelle überhaupt alle notwendigen Daten enthält, um eine vollständige Information an das System liefern zu können. Also ob eine vollständige Ereignisgruppe möglich ist.

Veranstaltungsservice „eventim.de“

Diese Informationsquelle bietet für verschiedene Städte eine Auflistung von unterschiedlichen Veranstaltungen an. Die Quelle ist vom Aufbau her charakteristisch für Veranstaltungs-Webseiten, indem sie alle Veranstaltungen in einer Liste anbietet, die über mehrere Seiten verteilt ist.

Newsfeed der Polizei Hamburg

Im Gegensatz zum vorherigen Beispiel handelt es sich bei der dieser Quelle nicht um eine Webseite, sondern um einen Newsfeed im RSS-Format (*Really Simple Syndication*). Dieses Format bietet zur späteren Verarbeitung einige Vorteile, die die Sprache XML und damit auch RSS mit sich bringt. Dadurch ist es zum Beispiel möglich, besser einzelne Elemente zu selektieren und somit genauer in der Quelle zu navigieren.

Tabelle 3.3: Quellenanalyse: eventim.de

Typ	Webseite
Auszeichnungssprache	HTML
Syntaxvalidität	invalides HTML
Struktur	semi-strukturiert
Informationsverteilung	Ereignisgruppen über mehrere Seiten. Gesamte Information für eine Gruppe ist nur mit einem zusätzlichen Navigationsschritt zu erreichen
Vollständigkeit	Ort, Zeit und ergänzender Text vorhanden – also vollständig.

Tabelle 3.4: Quellenanalyse: Newsfeed der Polizei Hamburg

Typ	Newsfeed
Auszeichnungssprache	RSS
Syntaxvalidität	Valides RSS
Struktur	semi-strukturiert, da innerhalb der Elemente unstrukturierter Text vorkommt.
Informationsverteilung	die Quelle bietet immer eine bestimmte Anzahl von Ereignisgruppen an. Wobei die komplette Information an einer Stelle vorhanden und somit keine Navigation notwendig ist.
Vollständigkeit	Ort, Zeit und ergänzender Text vorhanden, also vollständig.

Fahrplanänderungen der Münchner Verkehrsbetriebe

Informationen von Verkehrsbetrieben sind vor allen Dingen für spätere Hintergrundinformation für die Eskalationslogik des track-u Systems interessant (siehe dazu auch Kapitel 2).

Die vorliegende Quelle der Münchner Verkehrsbetriebe¹³ (MVV) bietet somit einen guten Anlaufpunkt für den Raum München, um etwaige Gründe für Verspätungen herauszufinden.

Ein Problem bei dieser Quelle ist, dass die Informationen recht ungenau sind. So gibt es zum Beispiel keine genauen Angaben zu Orten und es fehlen beispielsweise Straßennamen. Das liegt aber vor allen Dingen daran, dass der Mensch, der ja der eigentliche Nutzer ist, über das notwendige Hintergrundwissen verfügt, um beispielsweise aus einer groben S-Bahn-Ortsangabe auf den genauen Ort zu schließen. Für das System ist dieses natürlich problematisch¹⁴.

Internetauftritt der „Süddeutschen Zeitung“

Diese Quelle unterscheidet sich grundlegend von den vorher untersuchten. Es handelt sich hierbei zwar ebenfalls um eine Webseite wie bei den meisten anderen genannten auch, aber die Informationsextraktion gestaltet sich um einiges komplexer. Der Grund hierfür ist der, dass die Quelle zwar selbst semi-strukturiert ist, aber die Informationen in unstrukturiertem Freitext ohne wiedererkennbare Muster enthalten sind. Zudem verfügt nicht jeder Artikel, der auf der Webseite auftaucht über eine für track-u interessante Ereignisgruppe.

¹³<http://www.mvv-muenchen.de>

¹⁴Eine Anfrage vom 22. Februar 2007 beim MVV hat auch ergeben, dass keine Schnittstelle mit strukturierteren Daten zur Verfügung gestellt werden kann.

Tabelle 3.5: Quellenanalyse: Fahrplanänderungen der Münchner Verkehrsbetriebe

Typ	Webseite
Auszeichnungssprache	HTML
Syntaxvalidität	invalide
Struktur	semi-strukturiert
Informationsverteilung	An einer Stelle, keine Navigation notwendig
Vollständigkeit	Ort, Zeit und ergänzender Text vorhanden. Der Ort ist aber sehr ungenau gefasst und beinhaltet nur Namen von Gemeinden

Um nun einen Text zu erkennen, der für die Extraktion in Frage kommt, muss der Text „verstanden“ werden, also Techniken des *Story Understanding* angewendet werden. Beispielsweise könnte anhand von Schlüsselwörtern eine Vorauswahl getroffen werden, um entsprechende Text zu erkennen.

Tabelle 3.6: Quellenanalyse: Internetauftritt der „Süddeutschen Zeitung“

Typ	Webseite
Auszeichnungssprache	HTML
Syntaxvalidität	invalide
Struktur	semi-strukturiert, aber die Information befindet sich in einem unstrukturiertem Freitext
Informationsverteilung	Artikel werden nur „angeteasert“, der vollständige Inhalt ist erst nach einem Navigationsschritt erreichbar.
Vollständigkeit	Ungewiss, da jeder Artikel unterschiedlich vom Inhalt ist

3.4.2 Präsentation, Verteilung und Erreichbarkeit von Ereignisgruppen

Die Ereignisgruppen stellen den Mittelpunkt der Informationsquellen dar. Damit ist gemeint, dass das Ziel ist, diese Gruppen in der Quelle ausfindig zu machen und zu extrahieren.

Nun verfügt jede Quelle normalerweise über eine Vielzahl solcher Gruppen und stellt diese aber nicht an einer Stelle (z.B. auf genau einer Seite, die mit einem einzigen Aufruf zu erreichen ist) in einer übersichtlichen Anordnung zur Verfügung. Stattdessen sind die Gruppen verteilt, bzw. es existieren nur Teilgruppen und die Auflistung der Gruppen findet in einer für die automatische Extraktion komplizierten Semistruktur statt. Im folgenden soll diese Problematik erläutert und definiert werden, wobei dabei zwischen

- der Präsentation,
- der Verteilung und
- der Erreichbarkeit von Ereignisgruppen

unterschieden wird.

Präsentation der Ereignisgruppen

Gestalterisch sind heute vor allen Dingen Webseiten fast keine Grenzen mehr gesetzt. Durch die Verwendung von *Cascading Style Sheets* (CSS), Tabellen, Formularen, dynamischen Inhalten mit JavaScript/Ajax¹⁵ und unzähligen anderen Elementen von Markup-Sprachen (HTML, XHTML etc.) lässt sich heute vieles darstellen und Inhalte lassen sich ansehnlich „verpacken“ bzw. präsentieren.

Was für den Benutzer der Seite meistens einen Vorteil darstellt, ist für die maschinelle Verarbeitung häufig von Nachteil. Ein automatisches Auffinden von gesuchten Informationen, genauer gesagt der Ereignisgruppen, in diesen semi-strukturierten Texten wird zu einer teils schwierigen Aufgabe. Die schon des öfteren genannten Gründe hierfür sind die unzähligen Formatierungs-Konstrukte, die die gesuchten Informationen umgeben.

Dieser Abschnitt stellt einige charakteristische Merkmale bei der Präsentation von Ereignisgruppen vor. Allgemein werden derartige Extraktionen von mehreren Gruppen innerhalb eines Textes auch häufig als *multi-slot extraction* bezeichnet [Sod99].

Erkennbare Beziehung zwischen Elementen einer Gruppe Keine der repräsentativen Quellen stellt nur eine Ereignisgruppe an einer Stelle gleichzeitig zur Verfügung, sondern eine Menge von Gruppen (z.B. tabellarische Auflistung von Ereignissen). Jede Gruppe besteht dabei, wie in 3.3 schon erwähnt wurde, aus den drei Teilen Ort, Zeit und Beschreibung. Diese Teile sind nun in der Semistruktur enthalten und müssen zusammenhängend extrahiert werden, damit die Gruppen entstehen können. Eine gute Voraussetzung hierfür ist eine erkennbare Beziehung – Relation – zwischen Elementen einer Gruppe, die am idealsten schon anhand der Semistruktur erkennbar ist.

Beispiel 3.8 soll dieses verdeutlichen, wobei hierbei davon ausgegangen wird, dass die Menge der Ereignisgruppen auf einer Seite zur Verfügung steht. Andere komplexere Fälle, wie z.B. die Verteilung über mehrere Seiten, wird in den darauf folgenden Abschnitten erläutert.

```
...
<div class='event'>
  <p>Ort: München, Lothstraße<br>Datum: 02.04.2007<br>Beschreibung:
    Baustelle</p>
</div>
<div class='event'>
  <p>Ort: München, Sendlingertor<br>Datum: 03.04.2007<br>Beschreibung:
    Straßensperrung wg. Rohrbruch</p>
</div>
...
```

Beispiel 3.8: Eindeutig identifizierbare Beziehung zwischen Gruppen-Elementen

In diesem kurzen exemplarischen Ausschnitt einer Webseite sind die Ereignisgruppen jeweils innerhalb eines `<div>`-Elementes (eng. *div-tag*) eingefasst, welches sich auch noch innerhalb der Webseite durch einen Klassen-Namen (hier: „event“) bestimmen lässt. Dadurch kann dem System später durch Regeln gesagt werden, dass Gruppen-Elemente innerhalb dieser Delimiter in einer Relation zueinander stehen, dass heißt eine Ereignisgruppe bilden.

Diese Präsentation ist dadurch für die spätere Extraktion sehr gut geeignet, da die einzelnen Blöcke gut selektiert werden können.

Besonders Newsfeeds bieten aufgrund der schon vom Format vorgegebenen Auflistung der Informationsblöcke bzw. -einheiten (z.B. `item`-Elemente) eine gute Voraussetzung für das schnelle Erkennen

¹⁵Ajax steht für *Asynchronous JavaScript and XML* und ermöglicht das dynamische Laden bzw. Nachladen von Inhalten innerhalb einer Webseite. Dieses wird durch die Ausführung einer HTTP-Anfrage mittels `XMLHttpRequest` an einen Server ermöglicht.

von Relationen. Aber auch innerhalb von Webseiten sind oft durch z.B. Tabellenstrukturen solche Beziehungen erkennbar.

Keine erkennbare Beziehung zwischen Elementen einer Gruppe Falls die vorhandene Semistruktur keine Unterstützung bei der Erkennung von Relationen zwischen gefundenen Informationen bietet, wie Beispiel 3.12 zeigt, wird die Extraktion erschwert.

```
...
<b>Aktuelle Nachrichten in München:</b>
Ort: München, Lothstraße<br>02.04.2007, eine <i>Baustelle</i>. ... Ort:
München, Sendlingertor<br>03.04.2007, Straßensperrung wg. Rohrbruch ...
...
```

Beispiel 3.9: Nicht-eindeutig identifizierbare Beziehung zwischen Gruppen-Elementen

Hier ist nicht eindeutig, durch eine schon in der Semistruktur vorgenommenen Gruppierung, erkennbar, welche Informationen zu einer Ereignisgruppe zusammengehören.

Wobei dieses Beispiel nicht gerade ein Beispiel für eine besonders aufwendige Extraktion ist. Der Vorteil ist hier nämlich, dass ein erkennbares Muster vor jeder Gruppe steht (hier: „Ort“). Dieses umschließt zwar nicht jeweils eine gesamte Gruppe in der Struktur, markiert sie aber dennoch. Zudem ist eine gleiche Anzahl von Orten, Zeiten und Beschreibungen vorhanden.

Ein Algorithmus zur Suche der Ereignisgruppen könnte beispielsweise folgendes bei dem Durchlaufen des Textdokuments machen:

- suche das Muster „Ort:“. Extrahiere alle Wörter bis zum nächsten Zeilenumbruch (
). Dieses sind die Orte.
- suche nach einem Datum, das nach dem Wort „Ort:“ auftaucht und speichere es als Ereignis-Zeit.
- Extrahiere nach dem Muster „Ort:“ den kompletten Text bis zum nächsten „Ort:“ und nehme ihn als Beschreibung.

Nach diesem Ablauf würde das System eine gleiche Anzahl von Ortsangaben, Zeiten und Beschreibungen verfügen und diese könnten in einfacher Form zu Ereignisgruppen zusammengeführt werden (*merging*).

Problematisch wird es, falls unvollständige Angaben im Text gemacht werden und somit keine Zusammenführung mit einfachen Mitteln möglich ist. Im Beispiel 3.10 sind nun einige Informationen weggelassen worden.

```
...
<b>Aktuelle Nachrichten in München:</b>
Ort: München, Lothstraße<br>02.04.2007, eine <i>Baustelle</i>. ... Ort:
München, Sendlingertor<br>Straßensperrung wg. Rohrbruch ...
...
```

Beispiel 3.10: Nicht eindeutig identifizierbare Beziehung zwischen unvollständigen Gruppen-Elementen

Hier kann das System mit der oben genannten Technik nicht eindeutig die Ereignisgruppen identifizieren.

Ein anderes interessantes Szenario ist in Beispiel 3.11 zu sehen. Bei Betrachtung durch den Benutzer mittels eines Webbrowsers werden die Ereignisgruppen zwar zusammenhängend angezeigt (siehe Abbildung 3.4), aber im dem semi-strukturierten Quelltext, zeigt sich ein anderes Bild.

Durch die absolute Positionierung der Informationen kann der eigentliche Text an einer beliebigen Stelle im Dokument stehen. Dazwischen können noch unzählige andere Konstrukte vorhanden sein. Um hier eine Extraktion vorzunehmen, bedarf es einer ausgiebigen Analyse durch einen Knowledge Engineer. In diesem Beispiel und unter der Voraussetzung, dass in jeder Ebene die gleiche Anzahl von Gruppenelementen vorkommen, könnte wieder ein Ansatz wie in einem der vorherigen Beispiele gewählt werden, bei dem die Daten zum Schluss zusammengeführt werden.

```

...
<div style='position: absolute;left: 50px;top: 50px; width:200px;'>
  <strong>Orte</strong><br><br> München Lothstraße<br>München Sendlingertor
</div>
...
<div style='position: absolute;left: 250px;top: 50px; width:200px;'>
  <strong>Datum</strong><br><br> 01. April 2007<br>10. August 2006
</div>
...
<div style='position: absolute;left: 450px;top: 50px; width:200px;'>
  <strong>Grund</strong><br><br> Baustelle<br>Rohrbruch
</div>

```

Beispiel 3.11: Nicht eindeutig erkennbare Gruppen-Relationen durch Verteilung auf der Seite

Noch schwieriger sieht es bei einer Semistruktur aus, bei der die Ereignisgruppe innerhalb eines natürlich-sprachlichen Textes integriert ist.

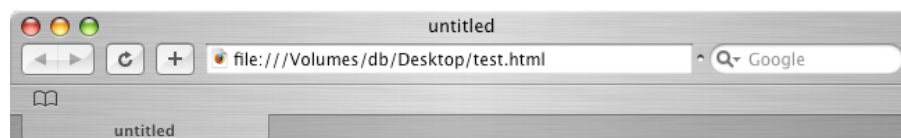
```

...
<p>
Wie die aktuellen Nachrichten aus München verlauten liessen , gibt es gerade
eine Baustelle in der Lothstraße , die seit dem 2. April dort ist und einen
Rohrbruch am Sendlingertor. Der ist am Tag danach , am 3. aufgetreten .
</p>
...

```

Beispiel 3.12: Nicht eindeutig identifizierbare Beziehung zwischen Gruppen-Elementen aufgrund von unstrukturiertem Text

Die zwei Ereignisgruppen aus diesem Freitext zu extrahieren ist der komplizierteste Fall, da hier keinerlei strukturelle Anhaltspunkte gegeben sind und nur noch natürlich-sprachliche Extraktionsmethoden weiterhelfen.



Orte	Datum	Grund
München Lothstraße	01. April 2007	Baustelle
München Sendlingertor	10. August 2006	Rohrbruch

Abbildung 3.4: Verteilung von Informationen mittels Layern

Verteilung der Ereignisgruppen

Neben der Präsentation der Ereignisgruppen in den Quellen ist auch die Verteilung ein wichtiges Kriterium. Wie schon am Anfang dieses Abschnittes kurz erwähnt wurde, ist mit der Verteilung u.a. die Anzahl der Navigationsschritte bis zum Erlangen der gesuchten Informationen gemeint.

Dieses ist ein wichtiges Kriterium, da nicht jede Quelle an einer Stelle über die kompletten Elemente für eine Gruppe verfügt. Beispielsweise stellen die oben betrachteten Präsentationen der Ereignisgruppen, z.B. innerhalb Tabellen etc., nur die einfachste Verteilung dar.

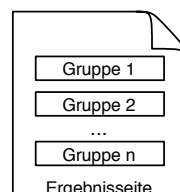
Jeder Block beinhaltet häufig unvollständige Informationen, welche erst durch das Verfolgen einiger Hyperlinks in den Dokumenten vervollständigt werden können. Im folgenden werden die unterschiedlichen Verteilungen erläutert.

Die Begrifflichkeiten in diesem Abschnitt orientieren sich hierbei hauptsächlich an den Untersuchungen in [CBC97], wobei deren Fokus auf Ergebnislisten von Suchmaschinen gelegen hat (siehe Abschnitt über Wrapper). Die verwendeten Begriffe und das Konzept dahinter lassen sich aber übertragen und in dem Kontext dieser Thesis verwenden.

Alle Ereignisgruppen an einer Stelle Am einfachsten kann das System zu einer vollständigen Liste von Ereignisgruppen gelangen, wenn die Informationen komplett an einer Stelle vorliegen. Das heißt, wenn jeder Bestandteil der Informationen an einer Stelle in der Quelle ist und keine weitere Navigation vonnöten ist, um die Gruppen zu erstellen. [CBC97] spricht in diesem Fall von so genannten *one-level one page* Ergebnissen. *One-level* bedeutet dabei, dass keine Navigation notwendig ist, um alle Bestandteile der Gruppe zu erhalten und *one page*, dass alle Ereignisgruppen auf genau einer Seite zu finden sind (siehe Abbildung 3.5).

Verteilung über mehrere Seiten Informationsquellen in denen alle in ihr verfügbaren Gruppen an einer Stelle vorhanden sind, sind im Internet so gut wie nicht anzutreffen. Es müssen immer weitere Navigationsschritte erfolgen, um die Daten zu erhalten bzw. zu vervollständigen. Beispiele hierfür ist das „Blättern“ durch Ergebnislisten oder die Navigation über Menüpunkte. Im folgenden werden die unterschiedlichen Möglichkeiten vorgestellt, wobei sich die ersten beiden genannten ebenfalls an [CBC97] orientieren und der letzte Begriff neu in dieser Thesis hinzugekommen ist.

- *one-level multi page*: Hierbei handelt es um eine Quelle, die zwar eine Liste von Ereignisgruppen anbietet und jede Gruppe vollständig an einer Stelle zur Verfügung steht. Aber die Gesamtmenge alle in der Quelle vorhandenen Gruppen ist nicht vollständig an einer Stelle vorhanden. Es muss also navigiert werden, um jede Teilliste zu erreichen. Beispiele hierfür sind Ergebnislisten in denen über mehrere Seiten geblättert werden kann (siehe Abbildung 3.6).



□ Vollständige Ereignisgruppe

Abbildung 3.5: *one-level one page* Verteilung

- *two-level pages*: Falls die Informationen für eine Ereignisgruppe nicht komplett an einer Stelle erreichbar ist und es genau einen Navigationsschritt auf eine zusätzliche *Detailseite* bedarf, um die Gruppe zu vervollständigen, spricht man von *two-level pages* (Abbildung 3.7). [CBC97] erwähnt es nicht explizit, aber es kann davon ausgegangen werden, dass sich die Gesamtheit der Gruppen hier ebenfalls über mehrere Seiten verteilen können (also eine Art *two-level multi page*)
- *multi-level pages*: Eine Möglichkeit, die nicht in [CBC97] auftaucht, aber in heutigen Zeiten im World Wide Web häufig auftritt, sind *multi-level pages*. Die Gruppen werden dabei nicht nur durch einen Schritt vervollständigt, sondern es bedarf mehrerer Navigationsschritte, um dieses zu erreichen (z.B. Veranstaltungen: „auf der ersten Seite steht der Name, auf der zweiten die Zeit und eine dritte Seite verweist auf eine Straßenkarte für den Ort“). Abbildung 3.8 zeigt eine „multi-level pages“-Verteilung auf mehreren Ergebnisseiten mit jeweils zwei Detailseiten.

Erreichbarkeit von Ereignisgruppen

Nicht jede Auflistung von Ereignisgruppen ist direkt per URL im Internet als Seite aufrufbar. Häufig sind diese Listen nur über Formularanfragen¹⁶ erreichbar (z.B. Eingabe von Suchkriterien wie Datum usw.). In solchen Fällen fallen die Informationsquellen unter das so genannte *Deep Web* oder *Hidden Web*, worunter alle Teile des World Wide Web fallen, die nicht direkt über URLs aufrufbar und dadurch z.B. nicht von Suchmaschinen auffindbar sind [Wik07a, Eik99].

Obwohl keine Quelle unter den untersuchten Beispielen dieses zusätzliche Hindernis aufgewiesen hat,

¹⁶Es wird davon ausgegangen, dass POST-Requests zum Aufruf der Formulare notwendig sind. Einfache GET-Requests könnten auch direkt mit einer URL und den entsprechenden Parametern ausgeführt werden.

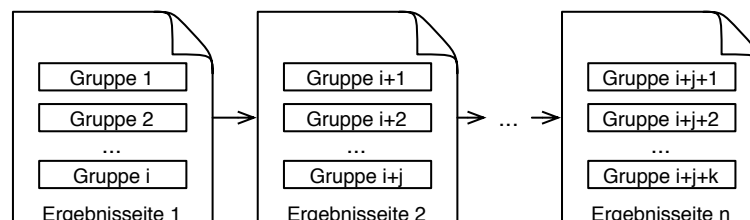


Abbildung 3.6: Beispiel einer *one-level multi page*-Quelle

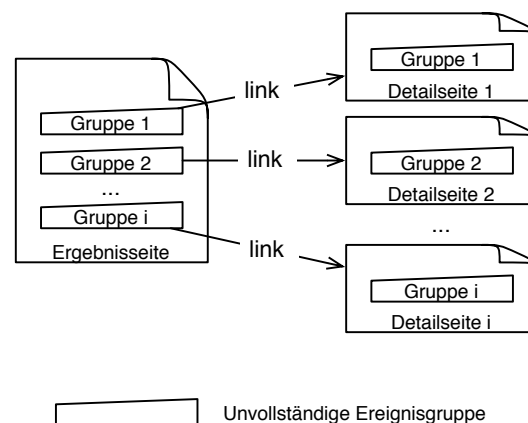


Abbildung 3.7: Beispiel einer *two-level pages*-Quelle

wird das Problem im Systementwurf trotzdem untersucht und behandelt werden.

Falls die Formularanfrage ausgeführt wurde und die Liste mit den Informationen zur Verfügung steht, können wiederum die anderen Erreichbarkeitskriterien zutreffen.

Zum Abschluss dieses Abschnitts zeigt die Abbildung 3.9 nochmal eine komplexere Möglichkeit der Verteilung und Erreichbarkeit von Ereignisgruppen. Hierbei existieren mehrere Ergebnisseiten, die aufgrund einer Formular-Suche zustande gekommen sind. Jede Seite enthält dabei eine Teilmenge aller Ereignisgruppen. Zudem ist jede Gruppe nicht vollständig auf einer Seite vorhanden, sondern ein zusätzlicher Navigationsschritt ist notwendig, um die Rest-Informationen zu erlangen.

3.5 Praxisnahe IE-Techniken

In den vorherigen Abschnitten wurden grundlegende IE-Ansätze vorgestellt und relevante Informationsquellen unter Beachtung verschiedener Aspekte analysiert und kategorisiert. Nun soll aufgrund der Ergebnisse der Quellenanalysen Wrapper-Ansätze und Technologien vorgestellt werden, die auch später im System Anwendung finden werden, also für dieses System als „praxisnah“ angesehen werden.

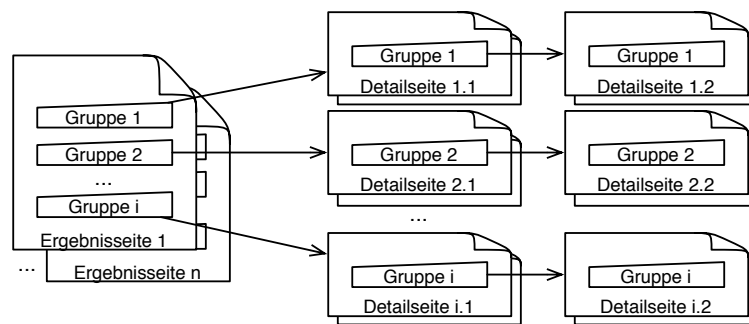


Abbildung 3.8: Beispiel einer *multi-level pages*-Quelle mit zwei Detailseiten

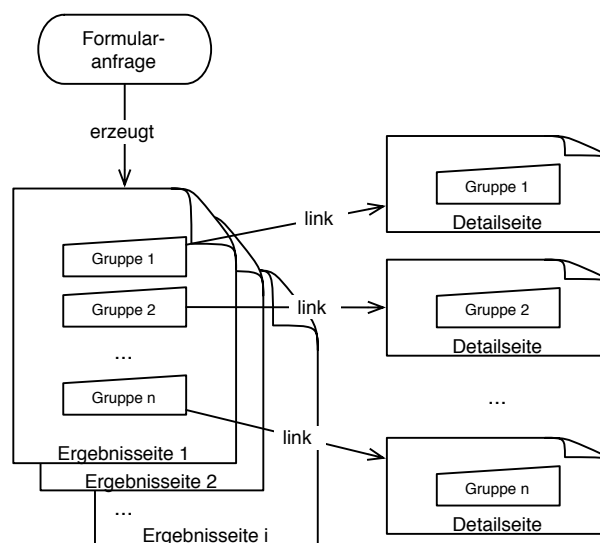


Abbildung 3.9: Beispielauflistung von Ereignisgruppen mit *two-level multi pages* und Formular-Erreichbarkeit

3.5.1 Gruppen-Extraktion anhand von Semistruktur und Mustern

Auf Seite 23 wurden unterschiedliche Arten der Präsentation von Ereignisgruppen vorgestellt. Hier wurde schon einmal erwähnt, dass wiedererkennbare Muster und die u.U. vorhandene Semistruktur die Extraktion erleichtern können. Dieser Vorteil soll im System ausgenutzt werden. Beispiele für derartig unterstützende Strukturen und Muster sind u.a.

- die Grundstruktur, das Layout, einer Webseite (Tabellen, Bereiche),
- der Aufbau von Newsfeeds wie beispielsweise RSS-, Atom oder RDF-Formaten,
- der Aufbau einer SOAP-Nachricht oder auch
- wiederkehrende Worte in Textbereichen die Inhalte näher spezifizieren und vom übrigen Text abgrenzen (z.B. „Ort:“).

Im folgenden werden die beiden Technologien kurz vorgestellt, die in der Arbeit verwendet werden, um Muster zu erkennen bzw. genutzt werden, um sich in der Seitenstruktur zu bewegen.

Reguläre Ausdrücke

Bei regulären Ausdrücken wird ein Muster definiert, welches auf einen Text angewendet werden kann und Teile des Textes „auswählen“ bzw. erkennen kann.

Da reguläre Ausdrücke auf semi- bzw. unstrukturierten Texten angewendet werden können und dadurch recht flexibel sind und zudem einen generischen Ansatz zur Mustererkennung bieten, stellen sie eine gute Methode im Bereich der Informationsextraktion dar.

Beispielsweise lassen sich hiermit innerhalb von semi-strukturierten Texten Blöcke mit den Ereignisgruppen selektieren. Abbildung 3.10 zeigt eine einfache Auflistung von Ereignissen innerhalb einer Webseite, die sich mittels des regulären Ausdrucks aus Beispiel 3.13 selektieren lassen.



Abbildung 3.10: Beispiel eines regulären Ausdrucks zur Selektion von Informationsblöcken in einem semistrukturellen Dokument

```
<div id='event'>([\w\-\.\S\, ]*)</div>
```

Beispiel 3.13: Selektion der Ereignisse in Abb. 3.10 durch einen regulären Ausdruck

Selektion von Elementen in semi-strukturierten Texten kann mittels regulären Ausdrücken aber auch zu einer aufwendigen Arbeit werden, da mit zunehmender Komplexität der Quellen häufig auch die Komplexität der Ausdrücke steigt. Eine Alternative bietet hierzu XPath (*XML Path Language*), falls es sich bei den semi-strukturierten Dokumenten um XML-valide Texte handelt.

XPath

Dank XML wurde in den letzten Jahren immer mehr Struktur in das Internet gebracht¹⁷. So stehen heutzutage ein Teil der Webseiten in der Markup-Sprache XHTML, einer Weiterentwicklung von HTML, zur Verfügung¹⁸. XHTML bietet dabei die Vorteile von XML, welche andere Quellen wie Newsfeeds und Webservices (SOAP) ebenfalls bieten.

Da für Dokumente, die in XML vorliegen, eine große Menge von Verarbeitungsmöglichkeiten geboten wird, liegt es nahe, diese ebenfalls für die Informationsextraktion zu nutzen.

Eine dieser Möglichkeiten bietet dabei die Lokalisierungs-Sprache *XPath* [HM03]. Diese Sprache, die selber kein XML-Format ist, findet im Falle dieser Thesis vor allen Dingen in der Selektion der Informationsblöcke (abgrenzbare Bereiche, die die Gruppenelemente enthalten) in Auflistungen Anwendung. Beispielsweise lassen sich mittels XPath in Newsfeed-Dokumenten schnell die einzelnen Einträge¹⁹ durchlaufen.

In XHTML-Dokumenten kann mit XPath z.B. eine Tabelle gezielt selektiert werden, in dem eine etwaiges eindeutiges Element-Attribut ausgewählt wird. Da es mit XPath möglich ist, bestimmte Bereiche durch Angabe von Ausdrücken zu selektieren bzw. zu erkennen, wurde XPath in diesen Abschnitt zur Mustererkennung aufgenommen.

Folgendes Beispiel vermittelt einen kurzen Eindruck der Möglichkeiten. Für detaillierte Informationen bietet [CD99] einen sehr guten Anlaufpunkt.

```
...
<div class='event'>
  <strong>Ort:</strong>Lothstraße , München<br />
  <strong>Datum:</strong>18.04.2007<br />
  <p><em>Baustelle aufgrund eines Rohrbruches</em></p>
</div>
<div class='event'>
  <strong>Ort:</strong>Sendlingertor , München<br />
  <strong>Datum:</strong>19.04.2007<br />
  <p><em>Großdemonstration</em></p>
</div>
...
```

Beispiel 3.14: Aufzählung von Ereignissen in XHTML

In diesem Beispiel, welches einen Teil des Quellcodes aus Abbildung 3.10 darstellt, werden einzelne Ereignisse aufgezählt, wobei die Zeit- und Ortsinformationen durch Delimiter identifizierbar sind. Zudem befindet sich jedes Ereignis innerhalb eines `div`-Blocks, der anhand eines `class`-Attributes selektierbar ist.

¹⁷Mit „Struktur“ sind hier nicht strukturierte Daten, sondern ein strukturierter Dokumenten-Aufbau gemeint

¹⁸XML in Kombination mit XSL wird hier nicht getrennt betrachtet, aber die Ergebnisse können genauso gut auf diese Art von Webseite angewendet werden.

¹⁹Im Falle von RSS wären dieses z.B. `item`-Elemente, bei Atom entsprechend die `entry`-Elemente.

Ein Ansatz wäre nun, dass zunächst einmal alle Ereignisse mit Hilfe eines XPath-Ausdrucks selektiert bzw. extrahiert werden. Der Ausdruck

```
//div[@class='event']
```

Beispiel 3.15: XPath-Ausdruck zur Selektion der Ereignisse von Beispiel 3.14

liefert alle diese Blöcke zurück. Nun kann dieser alleinige XPath-Ausdruck nicht genau die Informationen extrahieren, die relevant ist, sondern nur eine weitere Semistruktur. In dieser sind allerdings alle Elemente einer Ereignisgruppe enthalten. Ein regulärer Ausdruck ermöglicht es nun, in jedem Block innerhalb der Delimiter die Informationen auszuwählen. Den Ort würde beispielsweise folgender regulärer Ausdruck liefern:

```
</strong>([\w\-\.\ ]*),([\w\-\.\S\, ]*)</br >
```

Beispiel 3.16: Regulärer Ausdruck zur Selektion des Ortes in Beispiel 3.14

Anhand dieses Beispiels kann allgemein gesagt werden, dass XPath alleine nicht als IE-Mittel genutzt werden kann. Es kann aber unterstützend eingesetzt werden.

Der Grund hierfür ist, dass die relevanten Informationen meistens innerhalb von unstrukturierten Texten liegen, welche wiederum in einer Semistruktur eingebettet sind. Um mittels XPath aber wirklich gezielt die einzelnen Informationen extrahieren zu können, müssen diese in Elementen eingeschlossen sein und zwar ohne irgendwelche anderen Zeichenketten, die nicht zur Information gehören. Dieses ist aber in den meisten Fällen nicht der Fall.

XPath wird daher ein Hilfsmittel sein, um die Elemente einer Art „Vor-Selektion“ zu unterziehen, die den entsprechenden unstrukturierten Text enthalten. Auf den Text selbst können dann andere Methoden wie beispielsweise die regulären Ausdrücke angewendet werden, wie in dem Beispiel eben gezeigt wurde.

Es muss zudem gesagt werden, dass solch eine Vor-Selektion auch mittels regulären Ausdrücken möglich ist. Die Informationen die dadurch extrahiert werden, benötigen aber ebenfalls, wie im Falle von XPath, eine darauf folgende Extraktion, um die genauen Informationen zu erhalten.

In der weiteren Thesis wird diese Art von vor-selektierten Informationen als *Rohinformation* bzw. *Raw-Information* bezeichnet, die eigentliche Information dementsprechend als *Feininformation* bzw. als *Precision-Information*.

Dabei besteht eine vollständige Feininformation für das Ereignisgruppen-Element „Ort“ aus, wie in 3.3 definiert wurde, einer Straße, einer Hausnummer und einer Stadt. Die Feininformation für einen Zeitpunkt aus Tag, Monat und Jahr und die Beschreibung ist nicht weiter unterteilt.

Abbildung 3.11 zeigt Beispiele von Roh- und Feininformationen anhand von einer Orts- bzw. einer Zeitangabe.

Zudem muss bei einer Nutzung von XPath beachtet werden, dass immer noch der Großteil der im World Wide Web befindlichen Dokumente in keinem validen XML-Format vorliegen und somit eine Verarbeitung mit XPath bei vielen Quellen schon von Anfang an ausgeschlossen ist.

Für nicht-XML-valide Dokumente existieren zwar Werkzeuge zur Konvertierung in ein XML-Format, wie beispielsweise Dave Raggett's *HTML TIDY* [Rag, Myl01] welches HTML in XHTML umwandelt, aber diese Werkzeuge sind meist recht experimentell und nicht für die Informationsextraktion auf komplexen Quellen anwendbar²⁰.

Die eben vorgestellten Techniken der Informationsextraktion, nehmen mittels eines Abgleichs von vordefinierten Mustern bzw. Knotennamen auf semi-strukturierten Dokumenten, die Extraktion vor.

²⁰Einige Testläufe auf mittels TIDY umgeformten Quellcodes von Seiten wie z.B. <http://www.sueddeutsche.de/> haben dieses ergeben

Falls ein Textdokument nun aber keine Struktur bzw. wiedererkennbare Muster aufweist, die genutzt werden können – es sich also um *Freitext* handelt, wie in 3.1.1 schon kurz erwähnt wurde, müssen andere Ansätze gewählt werden, die einen Umgang mit natürlich-sprachlichen Text erlauben.

3.5.2 Gruppen-Extraktion aus natürlich-sprachlichen Texten

Wie im vorherigen Abschnitt schon erwähnt wurde, weisen natürlich-sprachliche Texte keine Struktur auf, die zur schnellen Erkennung der Ereignisgruppen-Elemente verwendet werden könnten. Im schlimmsten Fall sind nicht einmal einfache wiedererkennbare Muster vorhanden (z.B. „Uhrzeit:“, „Ort:“ usw.), die eine Selektion mit regulären Ausdrücken möglich machen würden.

Solche komplett unstrukturierten Texte kommen jedoch teilweise in Informationsquellen vor. Auch in semi-strukturierten Texten sind Teile der Ereignisgruppen unter Umständen in solchen Freitexten enthalten. Um diese zu extrahieren, werden die Methoden benötigt, die in Abschnitt 3.2.2 vorgestellt worden sind. Beispielsweise könnten durch ein Tagging-Schritt Ortsangaben markiert und später vom System extrahiert werden.

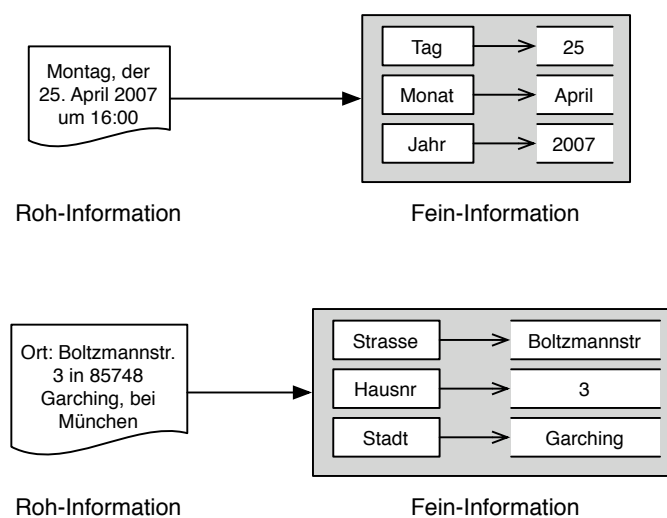


Abbildung 3.11: Beispiel von Roh- und Feininformation anhand von einer Ortsangabe und einem Zeitpunkt

4

Systemanforderungen und -aufbau

Nach den grundlegenden Kapiteln soll nun mit diesem Kapitel der eigentliche Entwurf des Systems beginnen. Dabei sollen hier nicht die einzelnen Bestandteile des Systems im Detail behandelt werden, denn dafür sind die Folgekapitel zuständig. Vielmehr soll dieses Kapitel das System analysieren, die Anforderungen herausarbeiten und eine Systemgrundstruktur dabei entwickeln, die in den späteren Teilen der Ausarbeitung verfeinert wird.

4.1 Anforderungen

Der Titel der Arbeit beinhaltet schon einige wichtige Informationen zu den Aufgaben und Anforderungen des Systems. Zum einen wäre dort der Begriff des „Multiagentensystems“ (MAS) und zum anderen die „Informationsgewinnung“. Auch die Integration in eine „mobile Applikation“ stellt einen wichtigen Aspekt dar, welcher schon in 2 ausgiebig erörtert wurde.

Aber was soll das System nun eigentlich leisten?

Ähnlich wie in Abschnitt 1.2 kann kurz zusammengefasst gesagt werden, dass die Aufgabe des Systems darin besteht, automatisiert mit Hilfe eines Multiagentensystems ausgewählte, im Internet verfügbare Informationsquellen (z.B. Webseiten usw. siehe Kapitel 3) aufzusuchen und relevante Informationen aus diesen Quellen zu extrahieren, um sie später der mobilen Anwendung in überarbeiteter Form zukommen zu lassen.

Diese recht kompakt formulierte Zusammenfassung soll jetzt im folgenden näher betrachtet und in die einzelnen Bestandteile zerlegt werden.

4.1.1 Multiagentensystem

Die Verwendung eines Multiagentensystems (MAS) als Plattform zur Überwachung der Informationsquellen ist ein Hauptbestandteil des Systems. Hierbei ist es das Ziel, dass Software-Agenten die Aufgabe der „Kontrolle“ der verschiedenen Quellen übernehmen.

Die Agentenplattform benötigt dazu Zugriff auf die unterschiedlichen Typen von Quellen und muss diese entsprechend verarbeiten und die resultierenden Informationen aufbereiten und persistent speichern können. Für die persistente Speicherung wird also ein Speicher oder *Repository* benötigt.

Das Multiagentensystem wird später einen Großteil der Applikationslogik des Systems beinhalten.

4.1.2 Informationsgewinnung

Der Informationsgewinnung oder auch *Informationsextraktion* (IE) wurde schon ein eigenes Kapitel gewidmet, in dem unterschiedliche Ansätze diskutiert und schließlich Möglichkeiten für dieses System

herausgearbeitet wurden.

Grundsätzlich kann an dieser Stelle gesagt werden, dass die IE in Form einer Wrapper-Komponente auf Ebene der Software-Agenten ins MAS integriert werden wird. Die Agenten greifen dabei auf die Quellen zu und initiieren die Ausführung des Wrappers. Dieser wendet eine Reihe von Extraktionsregeln auf die Quellen an und muss dabei über Möglichkeiten der Verarbeitung von semi-strukturellen Texten mittels regulären Ausdrücken und XPath verfügen und zusätzlich in der Lage sein, natürlich-sprachliche Methoden zu verwenden (siehe auch 3.2.4 zur Definition von „Wrapper“ in dieser Thesis).

4.1.3 Definition von Extraktionsregeln

Wie eben schon erwähnt wurde, erfolgt die Informationsgewinnung mit Hilfe von Regeln und Mustern innerhalb einer Wrapper-Komponente. Die Erstellung dieser Komponente erfolgt dabei auf Basis eines manuellen Ansatzes und erfordert einen Knowledge-Engineer mit Expertenwissen.

Um dieser Aufgabe gerecht zu werden, erfordert das System eine Benutzerschnittstelle, über die der Anwender für jede Quelle die entsprechenden Regeln definieren und zusammenstellen kann.

Die Schnittstelle ist Teil einer Anwendung, die es zudem auch noch ermöglichen soll, das System zu überwachen und einige administrative Aufgaben durchzuführen. Im folgenden wird die Applikation als „Verwaltungsapplikation“ oder „Managementapplikation“ bezeichnet.

4.2 Grundstruktur

Nach der kurzen Definition der Anforderungen und den allgemeinen Überlegungen kann an dieser Stelle schon eine grobe Grundstruktur für das System aufgezeigt werden.

Diese Struktur integriert dabei auch die externen Komponenten wie das track-u System und die Informationsquellen. Das eigentliche zu entwickelnde System verfügt dabei innerhalb seiner Systemgrenze über die vier Hauptbestandteile

1. Repository,
2. Wrapper,
3. Multiagentensystem und
4. Managementapplikation.

Bemerkung: Es muss gesagt werden, dass der Wrapper ja letztendlich in dem Multiagentensystem integriert ist. Da aber die eigentliche Arbeit (Quellen aufsuchen und Informationen extrahieren) des Wrappers zwischen den externen Quellen und dem MAS stattfindet, soll der Wrapper auch noch als eigenständige Komponente im System behandelt werden.

Neben diesen Hauptbestandteilen werden noch die Kommunikationsverbindungen bzw. die Schnittstellen zwischen den Komponenten in den jeweiligen Entwurfskapiteln betrachtet.

Abbildung 4.1 verdeutlicht die entwickelte Grundstruktur des Systems. Hierbei wird jeder wichtigen Einheit ein eigenes Kapitel zugeordnet, welches sich im weiteren ausführlich mit dem Entwurf der jeweiligen Komponente befasst. Die Systemstruktur wird dadurch von Kapitel zu Kapitel detaillierter und entsprechend erweitert.

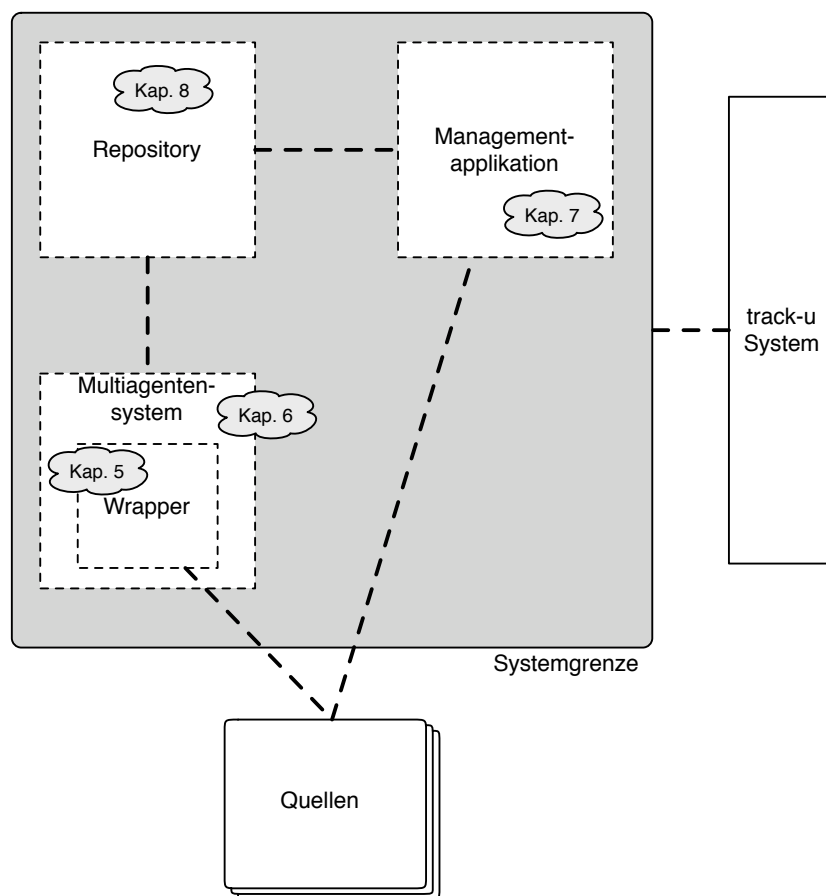


Abbildung 4.1: Grundstruktur des Systems

5

IE-Systementwurf

Eines der zentralen Themen dieser Thesis – die Informationsgewinnung – wurde in Kapitel 3 theoretisch behandelt und es wurden die verschiedenen Möglichkeiten der Informationsextraktion (IE) vorgestellt. Das vorliegende Kapitel wird nun da ansetzen, wo das Kapitel 3 im Bereich der IE aufgehört hat – nämlich bei der Konzeption der IE-Komponente, des Wrappers, in dem zu entwickelnden System.

Das vorliegende Kapitel hat nun die Aufgabe, die verschiedenen Algorithmen vorzustellen, die bei der Extraktion verwendet werden. Zudem werden Fragen, wie die Erlangung von vollständigen Ereignisgruppen sowie die Anforderungen an die Komponente, beantwortet.

5.1 Anforderungen an die IE-Komponente

Um einen Einstieg in die Thematik zu erhalten, werden im folgenden die allgemeinen funktionalen Anforderungen an die Extraktions-Komponente vorgestellt.

Aus Kapitel 3 ist ersichtlich geworden, dass das ganze IE-Konzept in dieser Arbeit zum Großteil auf der Anwendung von Mustererkennung auf den un- und semi-strukturierten Inhalten der verwendeten Quellen und auf der Navigation u.a. mittels XPath beruht.

Die Analyse der Beispielquellen hat zudem gezeigt, dass der Großteil der Informationen sich so extrahieren lässt und nur in einigen Fällen ein Einsatz von natürlich-sprachlichen Methoden notwendig ist. Daraus und durch die Einbeziehung der unterschiedlichen Erreichbarkeits-Typen und Präsentationsarten ergeben sich folgende Anforderungen an die IE-Komponente bzw. den Wrapper:

1. Laden von Informationsquellen
2. Auffinden und Extraktion von Informationen innerhalb eines Textes
3. Extraktion von Orts- und Zeitangaben aus natürlich-sprachlichen Texten mittels NER-Methoden
4. Navigationsmöglichkeit innerhalb der Quellen, abhängig von der Verteilung der Informationen. In diesem Zusammenhang auch Verkettung von Extraktions-Regeln, um komplexere Quellen verarbeiten zu können.
5. Möglichkeit Formulare anzusprechen und deren Ergebnisse zu verarbeiten, falls die Informationen nur durch Formularanfragen erreichbar sind

5.2 Laden der Informationsquellen

Das Laden der Informationsquelle stellt den ersten Schritt dar, den die IE-Komponente ausführen muss, um überhaupt zunächst einmal an das Textdokument zu gelangen, welches verarbeitet werden soll. Hierbei muss zwischen

1. XML-validen und
2. nicht-XML-validen Dokumenten

unterschieden werden, um von Anfang an gleich die richtige Auswahl an Verarbeitungsmethoden treffen zu können. So nutzt das System bei XML-validen Dokumenten einen XML-Parser, der das Textdokument in einen DOM-Baum (*Document Object Model*) lädt. Bei nicht-XML-validen Textdokumenten wird der Text mit einer standardmässigen Lade-Routine in eine Zeichenkette (eng. String) geladen, auf der später beispielsweise mit regulären Ausdrücken gearbeitet werden kann¹.

Ob es sich um ein XML-valides Dokument handelt oder nicht, kann entweder das System automatisch entscheiden oder ein menschlicher Benutzer manuell vornehmen, wenn die Quelle ins System eingefügt wird.

Das als DOM oder Zeichenkette geladene Textdokument wird nicht in dieser Form im System weitergegeben. Stattdessen wird ein eigenes Datenobjekt eingeführt, das so genannte *Processing-Subject* (PS). Dieses Objekt stellt im System die Daten dar, die einer Verarbeitung unterzogen werden können. Diese Daten können entweder das Dokument sein, was zu Beginn geladen wird, aber es kann sich dabei auch um Teilbereiche im Dokument handeln, auf denen Extraktionen stattfinden können. Falls beispielsweise eine Tabelle in einer Webseite selektiert wird, um dort Informationen zu extrahieren, handelt es sich bei der Tabelle bzw. dem HTML-Quelltext um ein Processing-Subject.

Ein Processing-Subject verfügt dabei über einen Inhaltstyp, der entweder ein DOM-Baum sein kann oder eine Zeichenkette. Das System entscheidet aufgrund des PS-Typs, welche Methoden (XPath bzw. reguläre Ausdrücke) zum Einsatz kommen, beispielsweise beim späteren „Durchlaufen“ etwaiger Ereignisblöcke.

Abbildung 5.1 veranschaulicht nochmals diesen Entscheidungsprozess und bezieht der Vollständigkeit halber auch noch HTML-Parser mit ein, die aber im weiteren nicht detaillierter behandelt werden.

5.3 Auffinden und Extraktion von Informationen innerhalb eines Textes

Die eigentliche Lokalisierung und die Extraktion von Informationen innerhalb eines Textes sind die beiden wichtigsten Schritte bei der IE-Komponente. Dabei befolgt das System Regeln, die vorher durch einen Knowledge Engineer manuell definiert worden sind (siehe auch spätere Verwaltungsapplikation in Kapitel 7).

Eine andere Methode wäre das in 3.2.1 erwähnte automatische Erstellen der Regeln durch beispielsweise induktives Lernen. Dieser Ansatz wurde aber für dieses System bereits verworfen, da eine Trainingsphase zu zeitintensiv wäre.

Der vorliegende Abschnitt soll sich noch nicht mit der kompletten Zusammenstellung vollständiger Ereignisgruppen-Listen in einer komplexen Quelle beschäftigen, da zum Verständnis dieses Prozesses

¹Es besteht auch die Möglichkeit, anstelle des normalen Ladevorgangs, einen Parser zu benutzen, der das Laden nicht-valider Dokumente unterstützt. So existieren einige HTML-Parser, die in vielen Fällen versuchen, Nicht-XML-Quellen zu bereinigen und daraus XML-valide Texte zu erstellen. In dieser Arbeit werden sie aber nicht verwendet, da sie häufig komplexere Dokumente nicht verarbeiten können und somit eine weitere Fehlerquelle im System darstellen würden.

noch einige Erklärungen notwendig sind. Stattdessen bezieht sich dieser Abschnitt ausschließlich auf den Extraktionsentwurf innerhalb eines einzelnen Textes, d.h. ohne Besonderheiten wie beispielsweise Navigation oder Formularanfragen zu betrachten.

Der Wrapper muss dabei in der Lage sein, aus dem Text die drei folgenden unterschiedlichen Typen von Informationen zu extrahieren:

1. Informationen, die Bestandteil der Ereignisgruppen sind (Zeit, Ort, Beschreibung)
2. Informationen, die Beziehungen zwischen Ereignisgruppen-Elementen erkennbar machen
3. Informationen, die zur Navigation in der Quelle dienen (Hyperlinks)

Auch wenn die drei Typen von der Semantik her unterschiedlich sind, werden zum Auffinden die gleichen Methoden verwendet. Nur bei den Ereignisgruppen-Elementen kommen u.U. noch natürlichsprachliche Methoden zum Einsatz.

Die zur Extraktion genutzten Extraktionstechniken wären zum einen die regulären Ausdrücke und zum anderen XPath-Ausdrücke. Die Letzteren natürlich nur, falls diese auch aufgrund eventuell vorhandener XML-Validität eingesetzt werden können.

Die Extraktion der drei Informationstypen erfolgt im System dabei unter Verwendung eines speziellen Datenobjekts, welches die Extraktionsregeln repräsentiert – dem *Extraction-Pattern*.

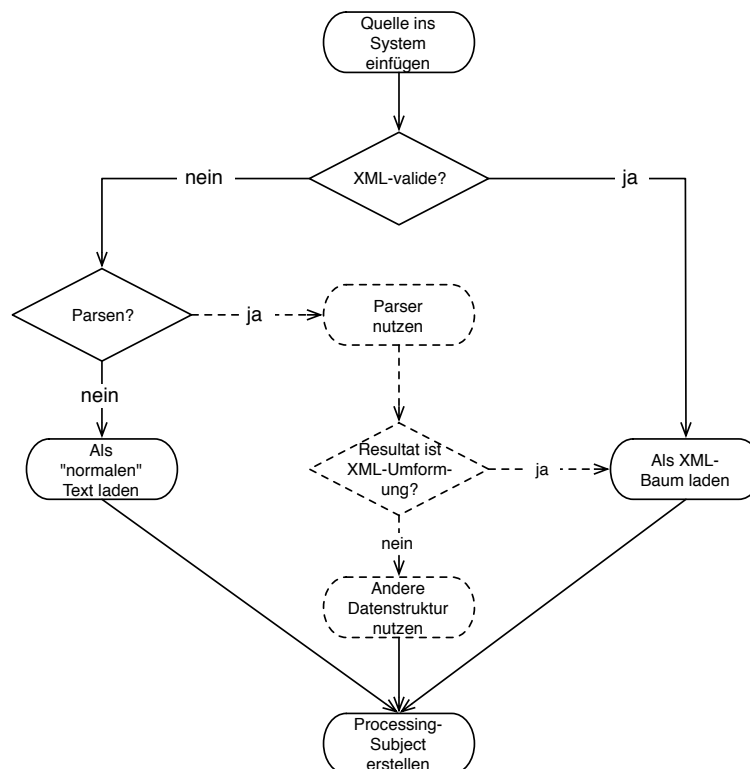


Abbildung 5.1: XML-Validität prüfen und danach entsprechende Verarbeitungsmethoden wählen

5.3.1 Extraction-Pattern

Die Informationsextraktion im System wird durch Ausführung von speziellen Extraktionsregeln erreicht, die vorher definiert wurden. Die Regeln werden durch das Extraction-Pattern dargestellt, welches sich aus folgenden Bestandteilen zusammensetzt und immer auf einem Processing-Subject ausgeführt wird:

1. IE-Methodik für die Rohinformation (*Raw-Pattern*)
2. IE-Methodik für die Feininformation (*Precision-Pattern*)
3. Typ der Regel, welcher Aussagen über weitere Verarbeitungsschritte trifft

IE-Methodik für die Rohinformation

In Abschnitt 3.5.1 wurde der Begriff der „Rohinformation“ schon eingeführt. Diese Information entsteht in einem ersten Extraktionsschritt durch Ausführung des so genannten Raw-Patterns, bei dem Teilbereiche (z.B. Elementinhalte) in der Quelle vor-selektiert werden.

Das Raw-Pattern ist deshalb auch Bestandteil des Extraction-Patterns. Beispielsweise extrahiert der XPath-Ausdruck in Beispiel 3.16 eine Liste von Rohinformationen zu den Orten und stellt somit eine Möglichkeit für ein Raw-Pattern dar.

Das Raw-Pattern ist dabei ebenfalls ein spezielles Datenobjekt, welches intern über einen Verarbeitungsausdruck (*Process-Expression*) verfügt, der entweder ein XPath-Ausdruck oder ein regulärer Ausdruck ist.

IE-Methodik für die Feininformation

Die Feininformation, ebenfalls in Abschnitt 3.5.1 eingeführt, repräsentiert die wirklich gesuchte Information in einer strukturierten Form. Zur Extraktion dieser Information wird das so genannte Precision-Pattern genutzt.

Im Gegensatz zum Raw-Pattern, welches nur über einen Verarbeitungsausdruck verfügt, muss das Precision-Pattern über eine Vielzahl solcher Ausdrücken verfügen. Der Grund hierfür ist der, dass aus den Rohinformationen im Falle von beispielsweise einer gesuchten Ortsangabe, drei Informationen extrahiert werden müssen, die drei unterschiedliche Process-Expressions benötigen (hier: Straße, Hausnummer und Stadt).

Neben diesen drei Ausdrücken für den Ort, kommen noch drei zusätzliche für Zeitangaben hinzu (Tag, Monat und Jahr), zudem noch eine Anweisung für Beschreibungen und eine für Hyperlinks, den u.U. gefolgt werden muss.

Welche dieser Verarbeitungsausdrücke letztendlich genutzt werden, entscheidet dabei der Typ, den das Extraction-Pattern hat, wie im folgenden gezeigt wird.

Typ

Der Typ vom Extraction-Pattern gibt an, welche Art von Informationen extrahiert werden und ob bzw. welche Verarbeitungsausdrücke vom Precision-Pattern dabei verwendet werden sollen. Es definiert ebenfalls, wie mit den extrahierten Informationen weiterverfahren werden soll, d.h. welche Schritte der Algorithmus auszuführen hat, nachdem Daten extrahiert wurden. Mögliche Pattern-Typen sind dabei:

- Elemente der Ereignisgruppe
 - Ortsangabe (*location*)
 - Zeitangabe (*time*)
 - Beschreibung (*description*)
- Hyperlinks zur Vervollständigung von Informationen (über Detailseiten, Ergebnisseiten usw.). Im folgenden wird dieser Typ auch als so genannter *follow-link* bezeichnet
- Auflistungs-Gruppen wie sind in Abb. 5.7 zu sehen sind, das heißt, falls erkennbaren Beziehungen vorhanden sind (siehe auch Abschnitt 3.4.2). Dieser Typ wird im weiteren als *group-relation* bezeichnet.

Jeder Typ bestimmt, wie schon erwähnt wurde, die weiteren Schritte, die bei der Extraktion ausgeführt werden und zieht somit unterschiedliche Verhaltensweisen des Algorithmus nach sich. Wie die Schritte im einzelnen aussehen, wird nun anhand einiger Beispiele gezeigt.

Location, Time, Description: Ein Extraction-Pattern vom Typ location, time oder description, extrahiert anhand des Raw-Pattern eine Liste von Rohinformationen. Diese Extraktion kann entweder mit Hilfe von XPath erfolgen oder regulären Ausdrücken. Welche Technik Verwendung findet, entscheidet dabei der Inhaltstyp des Processing-Subject auf dem das Pattern ausgeführt wird. Auf den dadurch erhaltenen Rohinformationen wird danach das Precision-Pattern angewendet, wobei am Ende dieses Prozesses abhängig vom Pattern-Typ entweder

- eine Beschreibung,
- ein Straßename, die Hausnummer und die Stadt oder
- ein Tag, ein Monat und ein Jahr

steht.

Da die Precision-Pattern auf einer Liste von Rohinformationen ausgeführt werden, steht am Ende auch eine Liste von Feininformationen als Ergebnis zur Verfügung. Wie mit dieser Liste weiterverfahen wird, wird in diesem Abschnitt noch nicht diskutiert, sondern ist Thema in einem der Folgeabschnitte über die Zusammenführung von Ereignisgruppen.

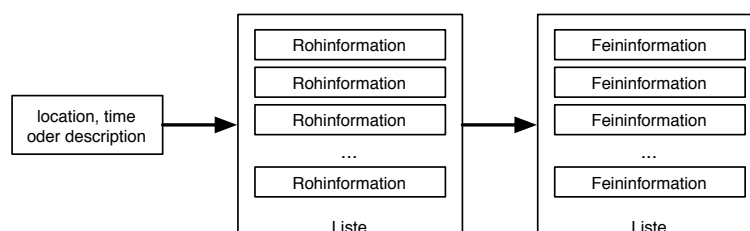


Abbildung 5.2: Ergebnis der Extraktion-Pattern vom Typ *time*, *location* oder *description*

Follow-Link: Ein Extraction-Pattern vom Typ *follow-link* extrahiert durch Anwendung des Raw- und des Precision-Patterns einen *Uniform Resource Locator* (URL). Diese URL stellt aber nicht die Information dar, die wirklich später genutzt wird, denn sie ist nicht Teil der Ereignisgruppe. Stattdessen wird mit Hilfe der URL die dahinter liegende Seite geladen.

Diese Seite stellt wiederum ein Processing-Subject dar, welches in weiteren Schritten mit Hilfe von Extraction-Pattern verarbeitet werden kann. Abbildung 5.3 verdeutlicht die Schritte, die ein *follow-link* Pattern nach sich zieht.

Group-Relation: Ein Extraction-Pattern vom Typ *group-relation* ist der einzige Typ von den Extraction-Pattern bei dem das Precision-Pattern keine Anwendung findet. Die Aufgabe des Patterns ist es, Bereiche innerhalb der Semistruktur vor-zuselektieren, die mehrere Elemente der Ereignisgruppen beinhalten. Es soll also die Relation zwischen den Elementen bewahren und nicht gezielt und präzise Information extrahieren.

Aus diesem Grund liefert eine *group-relation* als Ergebnis eine Zeichenkette zurück, die einen Teil des Textes darstellt, auf den das Pattern angewendet wurde. Diese Zeichenkette wird wiederum in ein Processing-Subject umgewandelt und ist Grundlage für die Ausführung weiterer Verarbeitungsschritte. Eine grobe Skizzierung des Ablaufes ist in Abbildung 5.4 zu sehen.

Abbildung 5.5 zeigt ein Beispiel für ein fertiges Extraction-Pattern vom Typ *time*, welches alle in dem Text enthaltenen Zeitangaben extrahieren würde. Dafür würde zuerst das Raw-Pattern ausgeführt werden und auf jedem Element der resultierenden Ergebnisliste das Precision-Pattern.

5.3.2 Zusammenführung von Ereignisgruppen

Die eben vorgestellten Extraction-Pattern liefern im Normalfall eine Liste von Ergebnissen, deren Elemente aufgrund der Ausführung der Raw- bzw. Precision-Pattern entstehen. Auf Seite 23 wurden die unterschiedlichen Präsentationsarten von Ereignissen erläutert. Hierbei wurden ebenfalls die Probleme

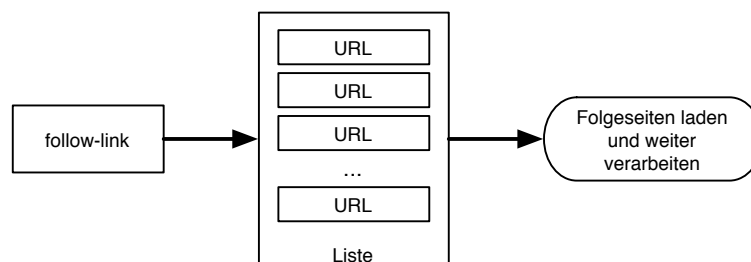


Abbildung 5.3: Ergebnis der Extraktion-Pattern vom Typ *follow-link*

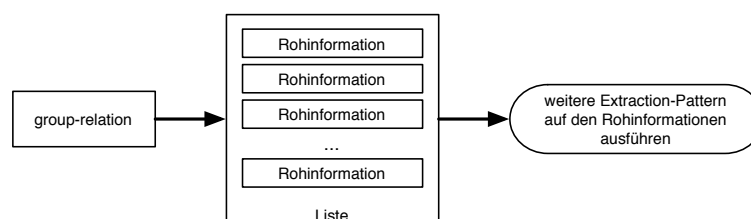


Abbildung 5.4: Ergebnis der Extraktion-Pattern vom Typ *group-relation*

bei der Erkennung der Gruppenelement-Relationen diskutiert. Dieser Problematik muss nun auch bei diesen Listen Beachtung geschenkt werden.

So werden bei einer nicht in der Semistruktur vorhandenen Gruppierung die jeweiligen Bestandteile der Gruppe mit Hilfe der Extraction-Pattern einzeln gesucht und in einem zweiten Schritt zu den Ereignisgruppen zusammengeführt. Diese Zusammenführung setzt voraus, dass alle Elemente in der gleichen Anzahl vorhanden sind, da sonst eine eindeutige Zuordnung nicht möglich ist. Abbildung 5.6 veranschaulicht die Schritte einer Gruppenzusammenführung.

Ein anderer Fall tritt dann auf, wenn die Beziehung zwischen den Gruppen-Bestandteilen anhand der

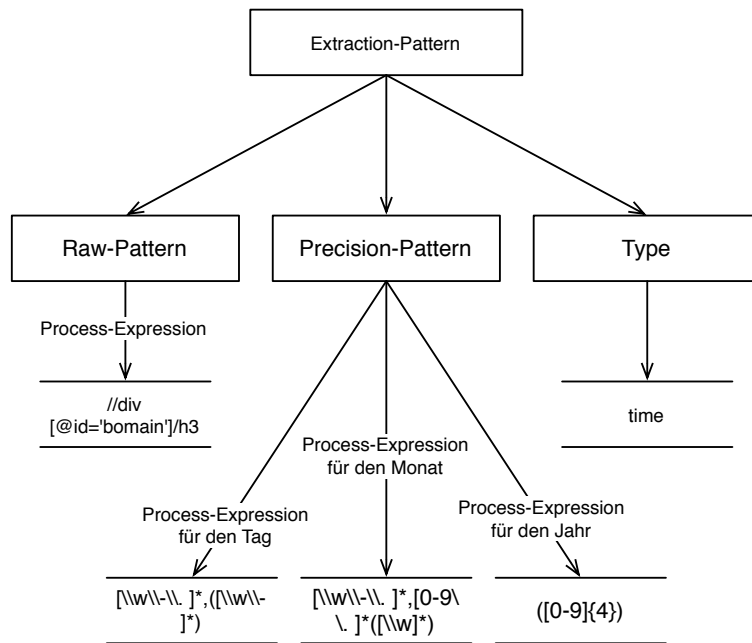


Abbildung 5.5: Extraction-Pattern zur Extraktion von Zeitangaben mittels regulären Ausdrücken

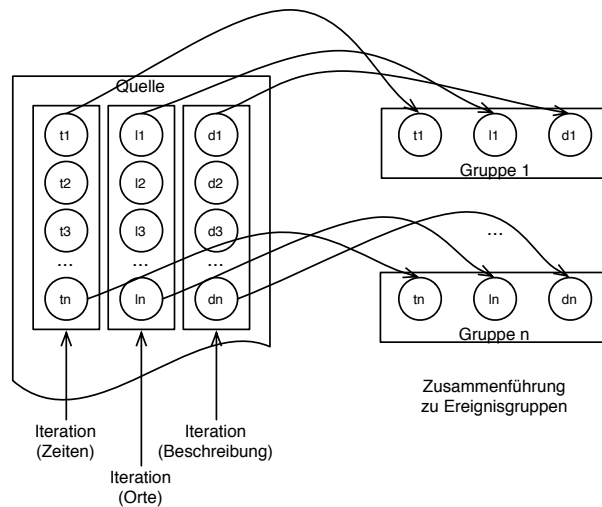


Abbildung 5.6: Zusammenführung von Gruppen-Elementen (time, location, description) zu Ereignisgruppen

Semistruktur erkennbar ist – beispielsweise anhand von Delimitern (siehe 3.4.2). Hier ist kein zweiter Schritt der Zusammenführung notwendig, wie die nächste Abbildung 5.7 zeigt.

Stattdessen werden bei dieser Art der Verarbeitung die einzelnen Blöcke durchlaufen, die durch Anwendung eines Patterns vom Typ group-relation entstanden sind. Innerhalb der Blöcke werden die Gruppen-Elemente durch Anwendung weiterer Extraction-Pattern extrahiert.

Falls der besondere Fall auftritt, dass nur für einige Teile der Ereignisgruppe durch Relationen erkennbar sind, muss anders verfahren werden. Hier werden zuerst die einzeln stehen Gruppen-Elemente mittels eines Extraction-Patterns extrahiert (Typ location, time oder description). Danach wird dann ein group-relation Pattern ausgeführt.

Eine Zusammenführung beider Liste ist erreichbar, falls beide die gleiche Länge haben und somit eine Zuordnung möglich ist.

Bemerkung: Die Zusammenführung gleichlanger Liste setzt natürlich voraus, dass die Reihenfolge der einzelnen Elemente in dem Textdokument gleich ist. Das heißt, falls die Zeiten in der Quelle hintereinander in einer festen Sortierung (z.B. jüngste Zeitpunkte zuerst) auftreten, müssen auch die Ortsangaben hintereinander auftreten und zwar passend zu den Zeitangaben (z.B. Ortsangaben passend zu den jüngsten Zeitpunkten zuerst).

Zudem setzt diese Zusammenführung voraus, dass mehrere Pattern hintereinander ausgeführt werden können. Dieser Vorgang wird noch detaillierter in 5.4.1 beschrieben.

5.3.3 Natürlich-sprachliche Methoden

Wie bereits in Kapitel 3 erwähnt wurde, kommt es in einigen Quellen vor, dass Teile der Ereignisgruppen innerhalb von natürlich-sprachlichen Text platziert sind. Da dort keine Struktur vorhanden ist, an der sich der Wrapper orientieren könnte, müssen hier Methoden des NLP eingesetzt werden.

Dabei ist es vor allen Dingen interessant, dass Ortsinformationen extrahiert werden können. Zeitinformationen könnten später von der Applikationslogik automatisch ergänzt werden (z.B. durch den Zeitpunkt des Auffindens) und als Beschreibung könnte beispielsweise der komplette Freitext dienen oder ein Textausschnitt, der sich im Bereich der Ortsangabe befindet.

Um nun beispielsweise natürlich-sprachliche Methoden zu integrieren, müsste das Extraction-Pattern erweitert werden. Zusätzlich zum Pattern-Typ würde ein weiterer Bestandteil (z.B. ein Flag)

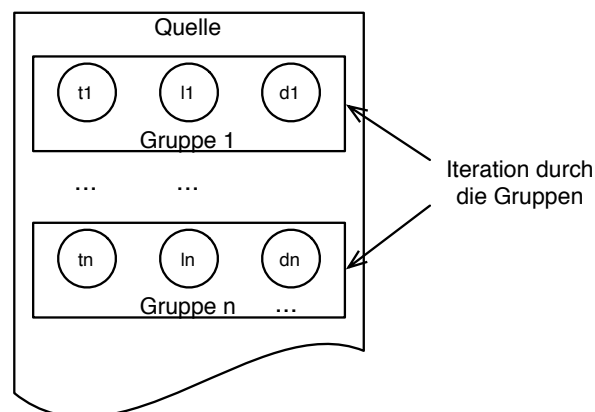


Abbildung 5.7: Iteration durch Ereignisgruppen in einer Quelle

hinzukommen, der Auskunft darüber gibt, ob ein NLP-Einsatz (hier: NER-Tagging) gewünscht ist. Raw- und Precision-Pattern bleiben bestehen und auch der Typ ist noch vorhanden und wird benutzt.

Falls das Flag gesetzt ist, wird zuerst das Raw-Pattern ausgeführt und extrahiert den Freitext. Auf diesem Text wird nun das NER-Tagging durchgeführt, welches einen markierten Text zurückliefert. Auf diesem Text wird dann darauf das Precision-Pattern angewendet, um die relevanten Teile mit Hilfe der Markierungen zu extrahieren (das Precision-Pattern muss dabei die NER-spezifischen Markierungen nutzen).

Problematisch wird die Weiterverarbeitung aber, falls die im Freitext enthaltenen Ortsangaben unvollständig sind.

Unvollständige Ortsangaben

In vielen natürlich-sprachlichen Texten kommt es vor, dass nur ungenaue bzw. unvollständige Ortsangaben zur Verfügung stehen. Beispielsweise enthält der Text

Frau Müller wurde am 21.03. in München überfallen.

Beispiel 5.1: Ungenaue Ortsangabe

die Ortsangabe „München“, die auch von einem NER-Programm erkannt werden und markiert werden würde. Jedoch ist diese Ortsangabe sehr ungenau formuliert, da sie eine ganze Stadt abdeckt und nicht ein kleines, bestimmbares Gebiet. Für die spätere Integration ins track-u System werden aber möglichst genaue Angaben benötigt und mit einer ganzen Stadt könnte nicht wirklich gezielt eine Zone ausgewählt werden. Aus diesem Grund ist ein NLP-Ansatz oft nicht sinnvoll und führt zum dem kompletten Verwerfen des Ereignisses.

5.4 Navigation in der Quelle

Neben der Präsentation der Informationen innerhalb einer Seite, ist die Verteilung ein weiterer wichtiger Aspekt der beachtet werden muss und im Kapitel 3 schon diskutiert wurde (siehe Seite 26). Hierbei wurden unterschiedliche Typen herausgearbeitet, die im System ebenfalls unterschiedlich behandelt werden müssen.

Die Feststellung, um welchen Typus es sich bei einer Quelle handelt, ist die Aufgabe des Knowledge Engineers, der dieses dem System bei der Quellen-Definition mitteilt. Diese „Mitteilung“ erfolgt dabei durch die Reihenfolge der Extraction-Pattern, die bei einer **Verkettung** festgelegt werden. Diese Verkettung soll im folgenden beschrieben werden, um im restlichen Abschnitt auf die Möglichkeiten bei der Navigation einzugehen.

5.4.1 Verkettung von Extraction Pattern

Einzel betrachtet liefern die Extraction Pattern nur bis zu einem gewissen Grad Informationen. Beispielsweise würde ein Extraction Pattern vom Typ location nach der Ausführung eine Liste von Ortsangaben liefern, nicht weniger aber auch nicht mehr.

Wird hingegen eine Verkettung (im weiteren wird die resultierende Kette als **Extraction-Chain** bezeichnet) von Pattern des Typs location, time und description vorgenommen, extrahiert jedes dieser Pattern eine Liste und das System fusioniert die Elemente der Liste zu einer Liste von Ereignisgruppen (siehe

Zusammenfassung von Gruppen in 5.3.2). Diese Art von Verkettung ist die einfachste und könnte z.B. auf „One-Level One Page“-Quellen zum Einsatz kommen.

Abbildung 5.8 veranschaulicht den Vorgang. Es wird allerdings wie immer vorausgesetzt, dass alle Listen die gleiche Anzahl von Elementen haben, da ansonsten eine Zuordnung nicht möglich ist.

Bevor im weiteren einige komplexere Beispiele aufgeführt werden, soll kurz zusammengefasst werden, welche allgemeinen Regeln bei einer Verkettung von Extraction-Pattern auftreten:

- Pattern, die nach einem Pattern vom Typ group-relation kommen, werden immer auf den Rohinformationen ausgeführt, die durch das Raw-Pattern der group-relation entstanden sind.
- Auf Ergebnissen vom Typ location, time und description werden keine weiteren Extraction-Pattern ausgeführt.
- Pattern, die nach einem follow-link kommen, werden auf der resultierenden Seite bzw. dem Processing-Subject der Link-URL ausgeführt
- Am Ende der Kette werden alle in den Listen time, location und description gespeicherten Informationen zu Ereignisgruppen zusammengefasst

Beispiel: Verkettung mit Gruppen-Relation

Abbildung 5.9 zeigt eine Verkettung mit 4 Extraction-Pattern. Das erste Pattern erzeugt dabei eine Liste von Relations-Gruppen. Auf den Elementen dieser Liste – genauer gesagt, den Rohinformationen – werden die folgenden Extraction-Pattern ausgeführt. In diesem Fall sind das die Pattern, die zur Extraktion der Ereignisgruppen-Informationen notwendig sind. Jedes Pattern extrahiert dabei aus dem gerade aktuellen Element der Relations-Gruppen-Liste die jeweilige Information. Dabei werden diese in einer

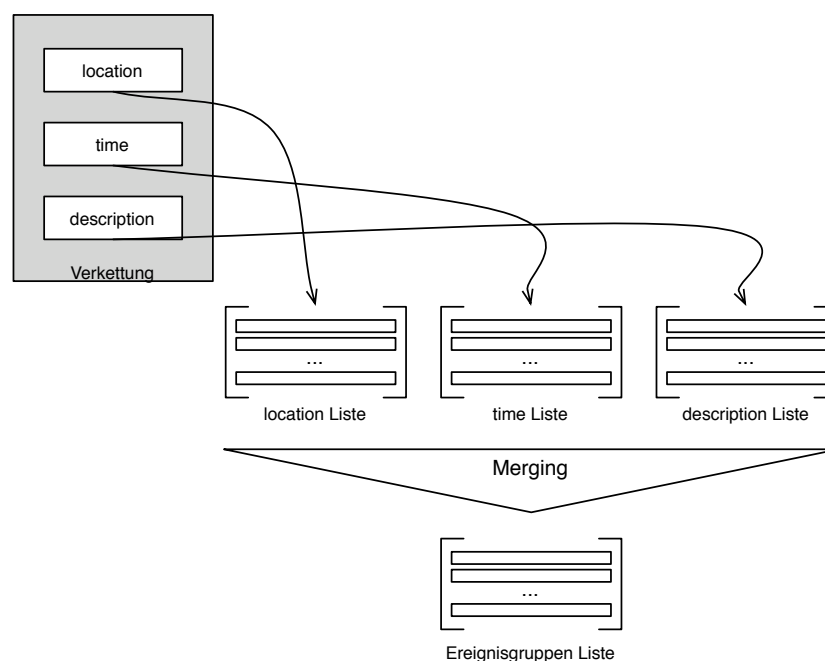


Abbildung 5.8: Beispiel: Einfache Extraction-Pattern Verkettung und anschließendes Merging

internen Datenstruktur gesammelt. Am Ende wird ein Merging durchgeführt.

Der Vorteil bei diesem Beispiel ist der, dass die Verbindungen zwischen time, location und description anhand der Gruppen-Relation schon definiert ist. Somit können auch fehlende Elemente ein Merging nicht behindern.

Ein komplexeres Beispiel wird im nächsten Abschnitt behandelt, nachdem auf die unterschiedlichen Verteilungstypen eingegangen wurden ist.

5.4.2 Vorgehen bei den unterschiedlichen Verteilungstypen

Nachdem nun die Möglichkeit der Verkettung vorgestellt wurden, soll nun auf die unterschiedlichen Navigationsmöglichkeiten eingegangen werden. Dieses geschieht anhand der unterschiedlichen Verteilungen, die in 3.4.2 vorgestellt wurden.

One-Level One Page

Diese Arten von Quellen sind direkt erreichbar und es ist keine gesonderte Navigation notwendig. Die ganze Verarbeitung findet an einer Stelle in der Quelle statt. Der Großteil der vorherigen Beispiele in diesem Abschnitt hat sich auf diesen Typus von Quelle bezogen.

One-Level Multi Page

Bei der Verarbeitung von „one-level multi page“-Quellen (Erinnerung: Quellen bei denen die Ereignisgruppen über mehrere Seiten verteilt sind, aber jede Gruppe vollständige Informationen enthält) ist

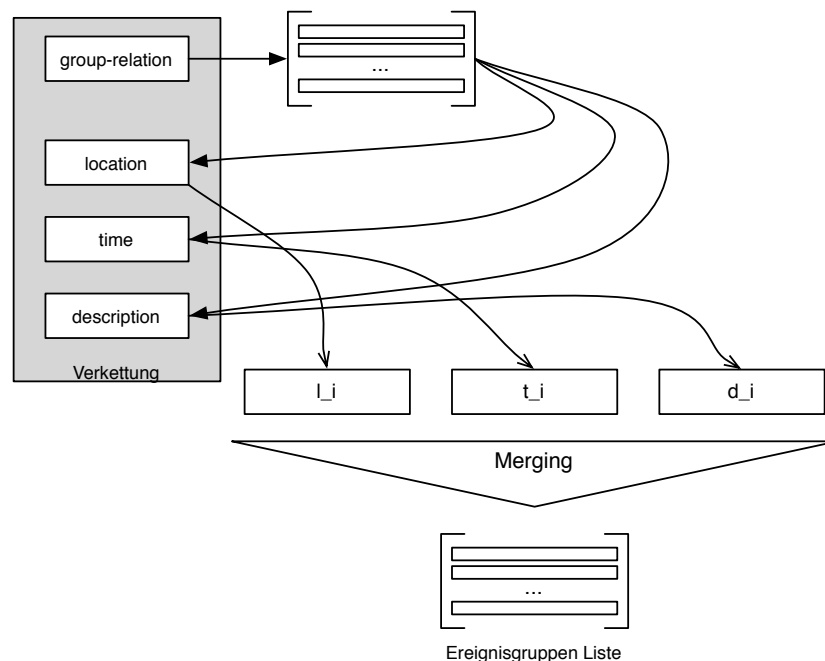


Abbildung 5.9: Beispiel: Verkettung von 4 Extraction Pattern

zu beachten, dass zusätzlich Hyperlinks verfolgt werden müssen, welche mit Hilfe eines follow-link-Patterns extrahiert werden. Die Hyperlinks treten dabei normalerweise in Form von Anchor-Elementen (z.B. HTML-Element `<a>`) in den einzelnen Seiten auf.

Ob es sich nun um eine „Multi Page“-Quelle handelt, erkennt der Algorithmus daran, ob das letzte Pattern in der Extraction-Chain ein follow-link Pattern ist. Falls dieses zutrifft, geht der Algorithmus folgendermassen vor:

1. Laden der Seite in ein Processing-Subject ps
2. Ausführen der Extraction-Chain bis auf das letzte Pattern auf dem Textdokument
3. Extraktion des Hyperlinks url durch Anwendung des letzten follow-link Patterns
4. Laden des nächsten Processing-Subjects ps' unter Anwendung von url
5. Wieder zu Schritt 2

Dieses wird so lange fortgeführt bis keine url mehr mit dem follow-link Pattern extrahierbar ist (siehe Abbildung 5.10).

Two-Level Pages

Bei Quellen mit einer *two-level pages*-Verteilung müssen ebenfalls Hyperlinks verfolgt werden. Dieses Mal dienen die Links dazu, die Ereignisgruppen zu vervollständigen, indem sie eine Detailseite aufrufen. Die Gruppen-Information ist also über zwei Seiten verteilt. Diese Art von Links werden im folgenden als „Detail-Links“ bezeichnet.

Die Aufgabe des System ist es nun, erst einmal jede Gruppe zu vervollständigen, bevor es sich der nächsten Gruppe zuwendet. Dadurch erfolgt nun bei jedem Iterations-Schritt durch die Gruppen, ein zusätzlicher Seiten-Request auf die Detailseite (siehe Abbildung 5.11).

Die Detailseite wird dabei ebenfalls wieder zu einem Processing-Subject. Auf ihr werden nun all die Pattern aus der Extraction-Chain ausgeführt, die nach dem follow-link kommen. Diese Teil-Kette wird

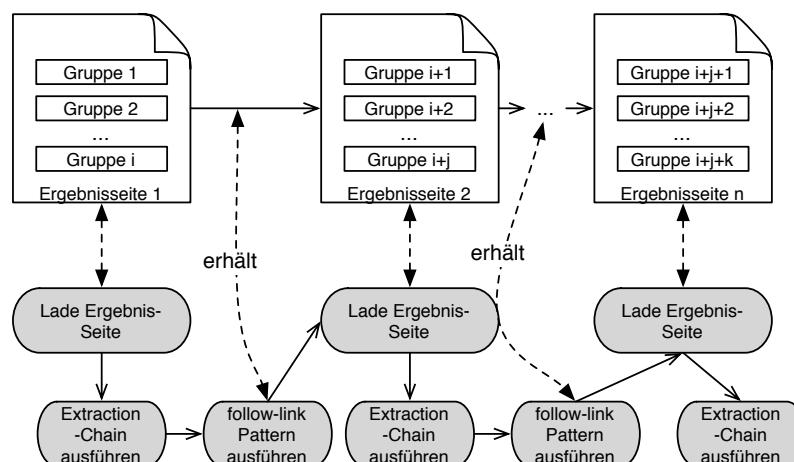


Abbildung 5.10: Verarbeitung einer *one-level multi page*-Quelle

auf jeder Detailseite aufgerufen.

Im Sonderfall der *two-level multi page*-Quelle müssen zusätzlich noch vor dieser Verarbeitung die Navigationsschritte für die „multi page“-Verarbeitung ausgeführt werden, wie es im letzten Abschnitt beschrieben wurde.

Multi-Level Pages

Der aufwendigste Verteilungsfall innerhalb einer Quelle ist die *multi-level pages*-Verteilung. Hier sind die Ereignisgruppen über mehrere Ergebnis- und mehrere Detailseiten verteilt, was das Extrahieren besonders erschwert. Auf jeder Detailseite sind somit auch wiederum Hyperlinks zu extrahieren, die zu weiteren Detailseiten führen. Das System muss also mehrere Seiten-Anfragen ausführen, um jede Ereignisgruppe zu vervollständigen.

Beispiel: Verkettung mit 2 follow-links

Das Beispiel in Abbildung 5.12 zeigt eine Verkettung bei der zwei Navigationsschritte erfolgen müssen (z.B. auf Detailseiten). Dabei wird mit einem location-Pattern begonnen, welches eine Liste der Ortsangaben extrahiert. Das darauf folgende follow-link Pattern ergibt ebenfalls eine Liste – eine Liste von Seiten auf denen die folgenden Pattern ausgeführt werden sollen. Das time-Pattern ist solch ein „Folge-Pattern“ und extrahiert für die i -te Seite die Zeit t_i und speichert diese in einer Liste.

Danach kommt zuerst das nächste follow-link Pattern in der gleichen Seite i an die Reihe und liefert eine weitere Seite i' auf der das description-Pattern ausgeführt wird, welches die extrahierten Beschreibungen ebenfalls in einer Liste sammelt. Am Ende werden die einzelnen Listen wieder unter der Voraussetzung zusammengefasst, dass alle die gleiche Länge aufweisen.

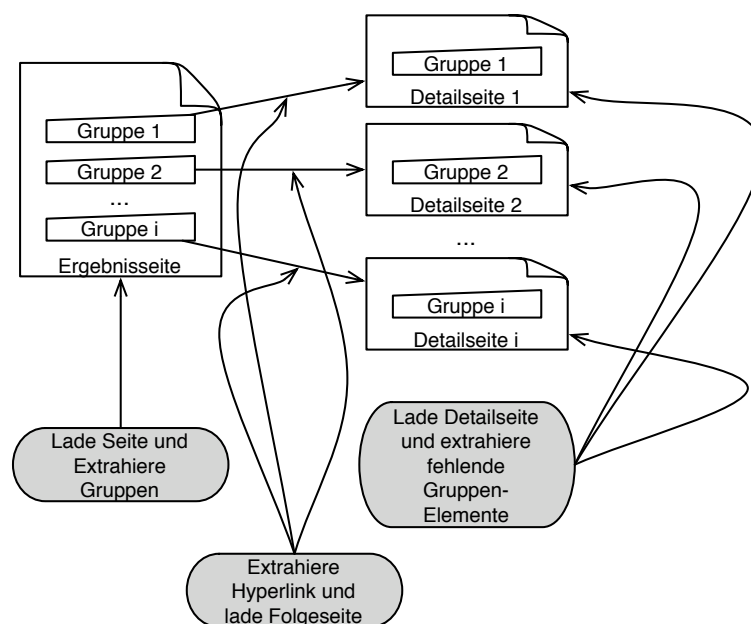


Abbildung 5.11: Verarbeitung einer *two-level pages*-Quelle

Probleme mit Änderungen an den Seiten

Da viele Informationsquellen kontinuierlich Änderungen unterliegen, beispielsweise durch Hinzukommen neuer Ereignisse usw., können teilweise Probleme auftreten.

Das folgende Beispiel soll dieses verdeutlichen:

- Der Wrapper lädt die Startseite einer Quelle. In diesem Textdokument befinden sich n Ortsangaben.
- Am Ende der Quelle existiert ein Link auf eine Detailseite, die alle Zeitangaben enthält. Heißt es existieren auf der Detailseite n Zeitangaben.
- Während der Extraktion der Ortsangaben wird vom Autor der Quelle ein neues Ereignis hinzugefügt. Es existieren nun also $n + 1$ Ortsangaben und $n + 1$ Zeitangaben. Letztere auf der Detailseite.
- Nachdem der Wrapper nun n Orte extrahiert hat (er bekommt ja nichts von der Änderung mit, da er auf dem vorher geladenen Processing-Subject arbeitet), folgt er dem follow-link und gelangt auf die Detailseite.
- Auf dieser Seite extrahiert er nun $n + 1$ Zeitangaben.

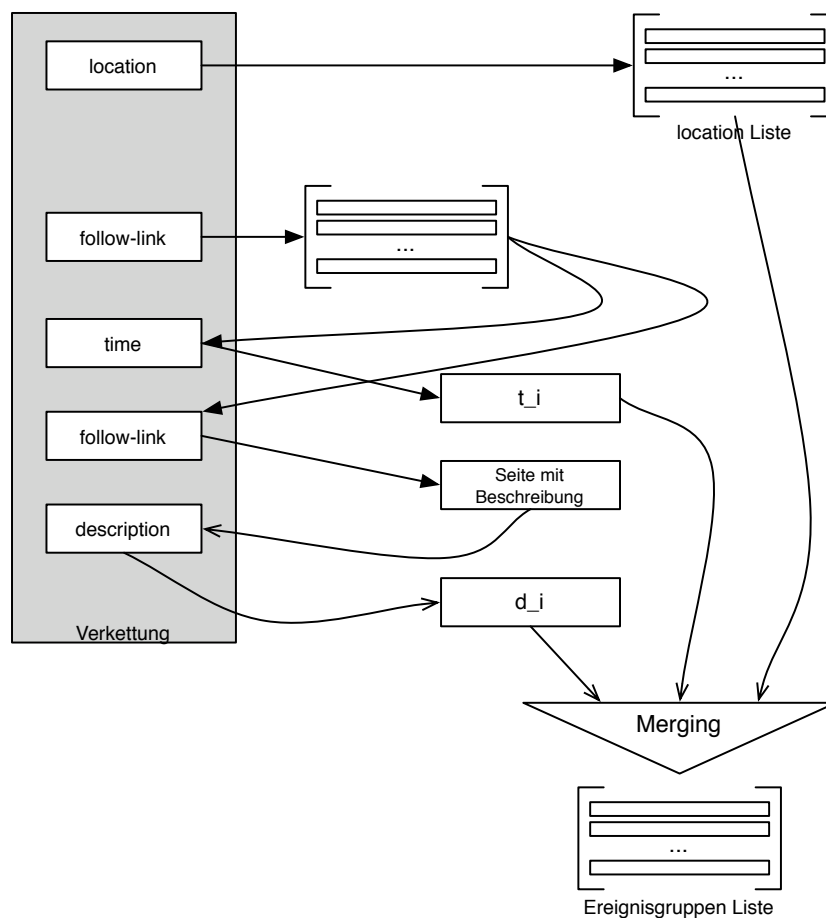


Abbildung 5.12: Beispiel: Verkettung mit 2 follow-links Extraction-Pattern

- Die von ihm geführten Orts- und Zeitlisten haben nun eine unterschiedliche Länge und ein Merging ist nicht mehr möglich.

Um dieses Problem zu lösen, müsste die Detailseite im selben Zeitpunkt geladen werden, in dem auch die Seite mit den Ortsangaben geladen wird. Auf diesen zwischengespeicherten Seiten würde dann die Extraktion erfolgreich ablaufen.

Für das weitere System wird aber angenommen, dass sich Quellen während der Extraktion nicht ändern und konstant bleiben. Für spätere Erweiterungen des Systems könnte aber solch ein Caching eingesetzt werden.

Ein weiteres Problem kann im Zusammenhang mit Hyperlinks auftreten wie im folgenden Abschnitt gezeigt werden soll.

Probleme mit Hyperlinks

Es ist den vorherigen Beispielen und Ausführungen davon ausgegangen worden, dass die Hyperlinks in Form von Anchor-Tags vorliegen und die darin enthaltene URL dadurch extrahierbar ist. In einigen Fällen kann es aber vorkommen, dass Verlinkungen beispielsweise über Javascript-Funktionen oder Formulare vorgenommen werden. Im Falle der Formulare kann ein Link extrahiert werden und mit der auch bei den Anchor-Elementen verwendeten Methodik weiterverarbeitet werden, falls das Formular einen GET-Request benötigt. Bei POST-Requests müssen andere Techniken verwendet werden, die im nächsten Abschnitt noch behandelt werden.

Falls die Navigation über Javascript erfolgt, ist eine größere Problematik gegeben, da diese u.U. keine URL enthalten, sondern mit internen Variablen arbeiten, die dann z.B. später auf URLs abgebildet werden.

Im Rahmen dieser Arbeit wird die Problematik zwar erwähnt, aber nicht näher behandelt, da sie in den untersuchten Quellen nicht vorgekommen ist. Nichtsdestotrotz könnten hier ebenfalls spätere Systemerweiterungen Techniken integrieren, die auch diese Art von Verlinkungen verarbeiten können.

5.5 Formularanfragen

Ergebnisseiten, die nur aufgrund einer Formularanfrage oder auch *Formular-Requests* erzeugt werden und nicht direkt per URL aufrufbar sind (also Teil des Deep Webs sind), stellen ein weiteres Hindernis bei der Erlangung aller Ereignisgruppen dar.

Da davon ausgegangen wird, dass solche Formulare am Anfang der Verarbeitung stehen, werden diese Requests ausgeführt, bevor die Informationsquelle in ein Processing-Subject geladen wird (siehe Abschnitt 5.2). Dazu muss der Knowledge Engineer die Eingabewerte für das Formular vorher definieren, damit auch die richtigen Ergebnisse erlangt werden. Abbildung 5.13 zeigt ein Beispiel anhand einer „one-level one page“-Quelle.

5.5.1 POST- und GET-Requests

Die Problematik mit POST-Requests wurde schon im Abschnitt über Hyperlinks erwähnt. Auch bei den Initial-Formularanfragen muss zwischen diesen beiden Typen unterschieden werden. Eine Möglichkeit dieses Problem zu umgehen, ist es, einen, wie in [Myl01] vorgeschlagenen, *GET-to-POST Proxy* zu integrieren, der GET Anfragen in POST-Anfragen umwandelt und diese dann an den Server sendet, wie in Abbildung 5.14 dargestellt wird.

5.6 Aktualisierte Systemstruktur

Aufgrund der Überlegungen, die in diesem Entwurfskapitel angestellt worden sind, ergibt sich nun die in Abbildung 5.15 dargestellte aktualisierte Struktur für das System. Hierbei wurde eine bidirektionale Verbindung (Empfangen von den Daten, aber Möglichkeit Anfragen zu senden bzw. in der Quelle zu navigieren) zwischen den Quellen und der Wrapper-Komponente integriert. Die Kommunikation läuft hierbei über HTTP (*Hypertext Transfer Protocol*) ab.

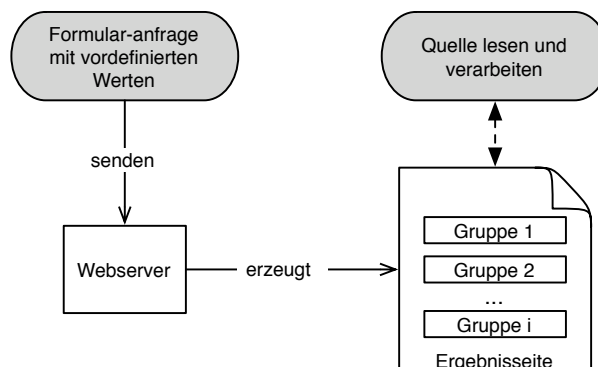


Abbildung 5.13: Integration einer Formularanfrage zur Erlangung der Ereignisgruppen

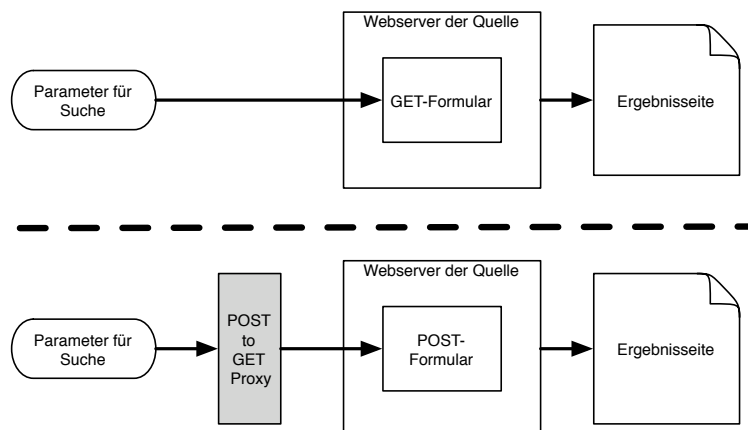


Abbildung 5.14: POST-to-GET Proxy

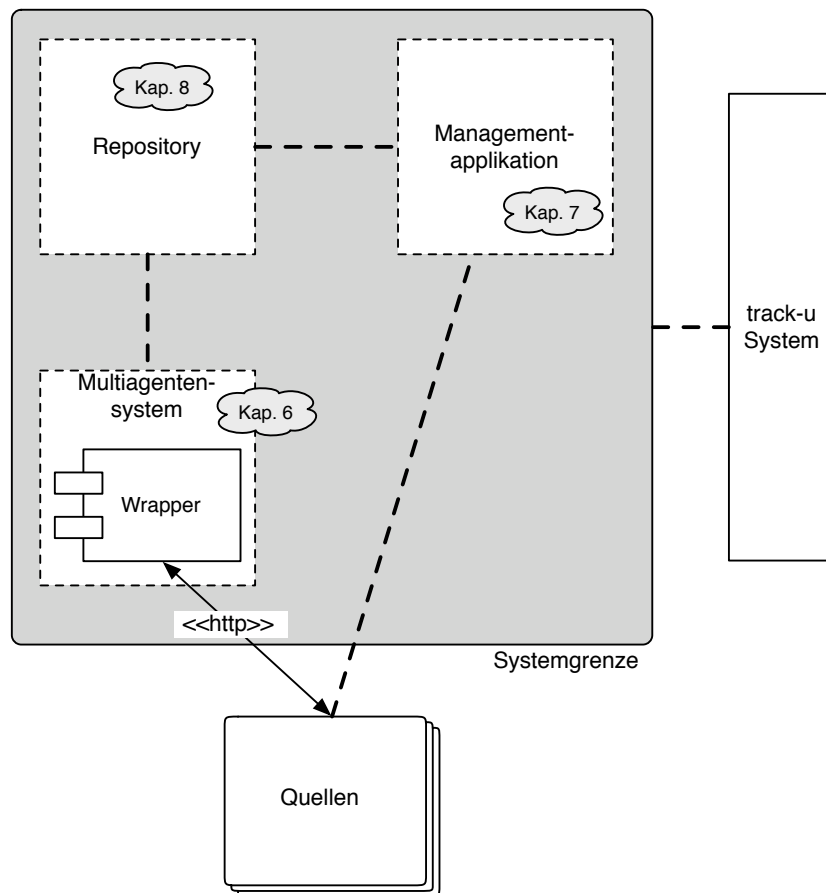


Abbildung 5.15: Systemstruktur nach IE-Entwurf

6

Entwurf des Multiagentensystems

Nachdem in Kapitel 5 die Informationsextraktions-Komponente entworfen wurde, wird sich nun in diesem Kapitel mit den Systemkomponenten befasst, die für die Ausführung und Koordination der Extraktion zuständig sind – den Agenten bzw. dem Multiagentensystem (MAS). Hierzu wird am Anfang eine kurze Einführung in den Begriff des „Agentensystems“ und dessen Bestandteile gegeben. Zudem werden einige Vorteile einer Agentenplattform aufgezeigt, bevor sich das restliche Kapitel mit dem eigentlichen Entwurf beschäftigt und am Ende die aktualisierte Systemstruktur vorgestellt wird.

6.1 Agentensystem

Die automatische „Überwachung“ der Informationsquellen erfolgt in diesem System unter Zuhilfenahme eines Multiagentensystems (MAS). Dabei sind die Agenten in dem MAS die ausführenden Einheiten. Aus diesem Grund werden zunächst einmal die Grundbegriffe dieser Umgebung vorgestellt.

6.1.1 Agent

Agenten bzw. „Software-Agenten“ sind Software-Komponenten, welche in dieser Thesis die Aufgabe der Kontrolle von Informationsquellen übernehmen. Da es für Software-Agenten noch keine einheitliche Definition gibt [PPW02], sollen charakteristische Merkmale herangezogen werden, die eine Einordnung dieser Komponenten sichtbar machen.

Da wäre zum einen die besonders im Kontext dieser Arbeit relevante Eigenschaft der *Autonomie*. Über diese sollte im Normalfall jeder Agent in einer gewissen Art und Weise verfügen, so dass er ohne menschliche Eingriffe oder dem Zutun anderer Systeme seine Ziele verfolgen und realisieren kann. Zudem sollte der Agent Kontrolle über sein eigenes Verhalten haben und dieses steuern können, also eine gewisse Entscheidungskompetenz aufweisen [Ses04, PPW02].

Unter eine etwas schwächere Variante der Autonomie fallen Agenten, die *semi-autonom* agieren. Diese Agenten sind teilweise auf menschliche Eingriffe bzw. Eingriffe von anderen System angewiesen. In dieser Thesis kommen eher semi-autonome Agenten zum Einsatz, da durch die Verwaltungsapplikation und die zentrale Datenbasis einige Abhängigkeiten bestehen, die ein komplett autonomes Verhalten nicht möglich machen.

Neben der Autonomie dienen noch einige weitere Kriterien der Charakterisierung von Agenten, die vor allem im Zusammenhang mit der Gruppe der *intelligenten Software-Agenten* benutzt werden. Dazu zählt unter anderem die *Interaktionsfähigkeit* des Agenten mit seinem Umfeld, wie beispielsweise

anderen Agenten. Diese Interaktion findet meist im Rahmen einer Kommunikation mittels strukturierter Nachrichten statt, wobei jede Nachricht eine Semantik mittels Sprechakten zugewiesen bekommt [BS98].

Ein weiteres Kriterium bzw. eine weitere Klassifikation für Agenten, ist beispielsweise die Eigenschaft der *Proaktivität*. Hierbei ist der Agent in der Lage, Eigeninitiative zur Erreichung seiner Ziele zu ergreifen. Bei Vorhandensein von *Reaktivität* hat der Agent die Fähigkeit, auf Veränderungen in seiner Umwelt selbstständig reagieren. Eine eingebaute *Lernfähigkeit* ermöglicht es einem Agenten zudem, anhand seiner Erfahrungen, sein Verhalten zu ändern und ggf. zu optimieren.

Auch Eigenschaften wie *Mobilität* und *Charakter* können auf einige Agenten zutreffen. Bei zuerst genannter Eigenschaft ist der Agent in der Lage, sich selbstständig auf andere Maschinen (z.B. Computer, mobile Endgeräte) zu transportieren. Ein Kriterium wie Charakter geht sogar soweit und weist dem Agenten eine glaubhafte Persönlichkeit und eine emotionale Seite zu [FG96].

6.1.2 Agentenplattform

Die eben erwähnten Agenten sind Software-Komponenten, die alleine gesehen, nicht ihren Aufgaben nachkommen können. Es wird immer eine Umgebung – eine spezielle *Agentenplattform* – benötigt, in der sie ausgeführt werden.

Diese Plattform stellt die technischen Voraussetzungen, sozusagen die Infrastruktur, zur Verfügung, um beispielsweise die Kommunikation zwischen einzelnen Agenten möglich zu machen. „Agentenplattform“ ist dabei ein Oberbegriff für alle Maschinen, Betriebssysteme, Management-Komponenten, Agenten und sonstige Software, die zur Umsetzung benötigt werden [FIP04].

Es existieren dabei verschiedene Standards für Agentenplattformen, wobei sich der FIPA-Standard (*Foundation for Intelligent Physical Agents*)¹ weitestgehend durchgesetzt hat. Andere Standards wie beispielsweise MASIF von der *Object Management Group* (OMG) werden zudem nicht so stark weiterentwickelt, wie es bei FIPA der Fall ist. Zusätzlich haben andere Standards auch häufig einen nicht so allgemeinen Ansatz wie FIPA, sondern sind auf spezielle Bereiche ausgerichtet. So bezieht sich MASIF z.B. eher auf Agentenplattformen für mobile Agenten [PPW02].

Aus diesem Grund wird im folgenden anhand von FIPA der Begriff der Agentenplattform vorgestellt.

FIPA Agentenplattform

Die Abbildung 6.1 zeigt das offizielle FIPA Referenzmodell einer Agentenplattform mit ihren logischen Komponenten

1. den Agenten,
2. dem *Agent Management System* (AMS),
3. dem *Directory Facilitator* (DF),
4. dem *Message Transport System* (MTS) und
5. der Agenten Plattform (AP) selbst.

Dabei muss beachtet werden, dass das Agent Management System, der Directory Facilitator und das Message Transport System – welches teilweise auch *Agent Communication Channel* (ACC) genannt

¹<http://www.fipa.org/>

wird – in einer Agentenplattform vorhanden sein muss, wenn diese als FIPA-konform gelten soll [VQC02].

Es müssen dabei aber nicht alle Komponenten auch wirklich genutzt werden, wie später auch noch im Zusammenhang mit dem DF erwähnt wird – aber sie müssen vorhanden sein.

Die **Agenten** und die **Agentenplattform** wurden in einem vorherigen Abschnitt schon beschrieben. Das FIPA Modell ergänzt hierbei nun vor allen Dingen die zentrale Komponente der Agenten noch um einige Eigenschaften, die noch nicht erwähnt worden sind. So wird definiert, dass die Kommunikation in der Plattform mittels der so genannten *Agent Communication Language* (ACL) zu erfolgen hat. Zudem hat jeder Agent über einen Eigentümer und eine eindeutige Kennung, den *Agent Identifier* (AID), in der AP zu verfügen [FIP04].

Das **Agent Management System**, welches nur einmal pro AP vorkommen kann, ist die zentrale Kontrollkomponente der Plattform. Jeder Agent muss sich hier registrieren, um eine AID zu erhalten. Das Management System verwaltet alle in der Plattform aktiven Agenten und bietet Zugriff auf diese Agenten-Listen, weswegen das AMS auch als die *white pages* der Plattform bezeichnet wird. Zudem regelt das AMS alle Zugriffe innerhalb des Systems und ist gleichzeitig für die Verwaltung der Kommunikation zu anderen Agentenplattformen zuständig.

Der **Directory Facilitator** ist eine Komponente, die optional verwendet werden kann. Sie erlaubt es, dass sich Agenten mit ihren jeweiligen Diensten (bzw. einer Beschreibung ihrer Dienste) bei ihr registrieren können. Hierdurch wird anderen Agenten ein Verzeichnis und die Möglichkeit geboten, über den DF gezielt nach Agenten zu suchen, die bestimmte Aufgaben ausführen können. Der DF wird dadurch auch als *yellow pages* bezeichnet.

Die Kontrolle, ob die beim DF gespeicherten Informationen über die Agenten richtig und aktuell sind, ist dabei nicht Aufgabe des Facilitators, sondern der Agenten selbst, die für die Pflege verantwortlich sind [FIP04].

Die letzte Komponente ist das **Message Transport System**. Dieses bietet eine Kommunikationsschnittstelle zwischen Agenten an, die sich in unterschiedlichen Agentenplattformen befinden. Der Austausch von Nachrichten erfolgt hierbei mittels der, schon oben erwähnten, Agent Communication Language.

6.1.3 Vorteile

Die Vorteile eines Agentensystems, vor allem im Kontext dieser Thesis, liegen in den technischen Gegebenheiten, die eine solche Plattform schon standardmässig mit sich bringt.

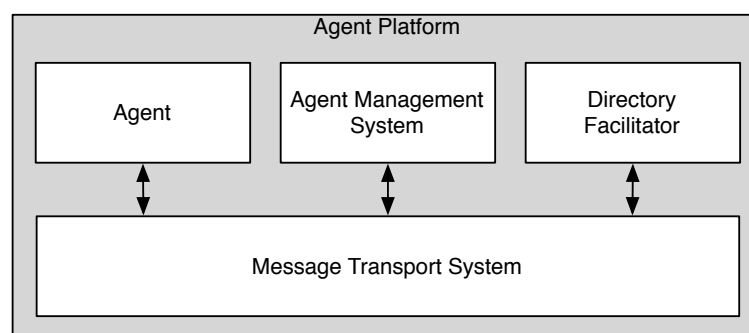


Abbildung 6.1: FIPA Referenzmodell für Agentenplattformen [FIP04]

So sind beispielsweise die eigenständig laufenden Software-Komponenten, welche mit komplett standardisierten Kommunikations-Funktionalitäten ausgestattet und über mehrere Rechner verteilbar sind, ein großer Vorteil. Dieses erlaubt es, ohne viel Aufwand, komplexe verteilte Anwendungen möglich zu machen [VQC02].

Zudem bietet ein MAS den Vorteil, dass bei Bedarf (z.B. weitere Informationsquellen überwachen) neue Agenten einfach zur Plattform hinzugefügt werden können und sofort einsatzbereit sind. Bei Verwendung alternativer Techniken ist dieses nicht so einfach möglich und erfordert entweder ein aufwendiges anfängliches Systemdesign oder eine ständige Nachbesserung an der Software.

Zusätzliche Komponenten wie die Verzeichnisdienste ermöglichen weitere Flexibilität z.B. für Systemerweiterungen bei denen auf Kommunikation zwischen den Agenten gesetzt wird.

Diese eben genannten Vorteile ermöglichen die Konzentrationen auf die Kernfunktionalität des Systems (die Informationsgewinnung), ohne dass sich viele Gedanken über die Infrastruktur gemacht werden muss.

6.2 Anforderungen an das MAS

Wie in Kapitel 5 gezeigt wurde, ist die IE-Komponente dafür zuständig, eine Informationsquelle zu laden und mit Hilfe einer Verkettung von Extraction-Pattern die Ereignisgruppen zu extrahieren und in einer Liste zurückzuliefern. Die Weiterverarbeitung dieser Liste fällt danach in den Zuständigkeitsbereich des Agentensystems, genauer gesagt dem Agenten, der für die jeweilige Quelle verantwortlich ist. Auch Fragen wie

- Wann soll eine Quelle extrahiert werden?
- Welche Quellen sollen extrahiert werden?
- Welche Ereignisse sind im Repository schon vorhanden, welche sollen neu hinzukommen und wie wird mit diesen umgegangen?

sind nicht Teil des Wrappers, sondern Teil der Applikationslogik in den Agenten.

Zusätzlich kommen noch einige andere Anforderungen zu jedem Agenten hinzu, wie das Geokodieren der Ortsangaben, das Aktualisieren agenten-spezifischer Informationen oder ggf. die Einflussnahme auf die Geo-Datenbank des track-u Systems.

Neben diesen, für jeden Agenten spezifischen Anforderungen, muss das Multiagentensystem im allgemeinen zusätzlich in der Lage sein, nach Ausfällen alle schon vorhandenen Agenten wieder herzustellen. Zudem müssen auch bei Bedarf einzelne Agenten erstellt werden können.

6.3 Informationsagent

Wenn in diesem System die Rede von „Agenten“ ist, ist damit immer eine ganz bestimmte Art von Agenten gemeint – die *Informationsagenten* oder auch *iAgents*. Alle Agenten, die im späteren System in der Plattform agieren, sind vom diesem Typ (Hinweis: Beim späteren Entwurf zur Geokodierung wird noch von einem anderen Typen die Rede sein, aber dieser kommt im System nicht zum Einsatz).

Jeder Informationsagent hat Kontrolle über die Ausführung der Informationsextraktion auf den von ihm überwachten Quellen. Dabei ist die IE-Komponente bzw. der Wrapper Bestandteil eines jeden Agenten, wie Abbildung 6.2 zeigt.

Agenten/Quellen-Verteilung

Damit ein Informationsagent Informationen extrahieren kann, ist es notwendig, dass ihm bekannt ist, welche Quellen er kontrollieren soll. Das heißt, es muss ein internes Wissen in jedem Agenten existieren, welche Quellen zu verarbeiten sind. Dieses Wissen wird in einem zentralen Repository gespeichert. Nach welchen Kriterien und welchem Algorithmus mehrere Quellen zwischen den Agenten zur Überwachung „aufgeteilt“ werden, ist nicht Aufgabe des MAS, sondern der Managementapplikation und wird in Kapitel 7 noch ausführlich behandelt. Stattdessen sollen an dieser Stelle die unterschiedlichen Verteilungsarten erwähnt werden, da diese für das weitere Verständnis von Bedeutung sind. So können entweder

1. mehrere Agenten existieren, wobei jeder einzelne Agent jeweils eine Quelle verwaltet oder
2. es existiert nur ein Agent, der alle Quellen verwaltet oder
3. es gibt mehrere Agenten, wovon jeder Agent für mehrere Quellen zuständig ist.

Die als letztes genannte Vorgehensweise erweist sich als am Sinnvollsten. Bei den anderen beiden Alternativen ist entweder eine Überlastung des Agenten zu befürchten (Möglichkeit 2) oder es ist eine Verschwendung von Ressourcen vorhanden, da jeder Agent auch Arbeitsspeicher benötigt und sich der Aufwand für nur eine Quelle pro Agent in den meisten Fällen nicht „lohnen“ würde (Möglichkeit 1). Jeder Informationsagent ist also für die Verwaltung von mehreren zugeteilten Quellen verantwortlich (Abbildung 6.3).

Bemerkung: Für die Anzahl der Quellen die pro Agent verwaltet werden können, wird in diesem System eine obere Grenze festgelegt (siehe ebenfalls Kapitel 7), so dass eine gleichmäßige Verteilung

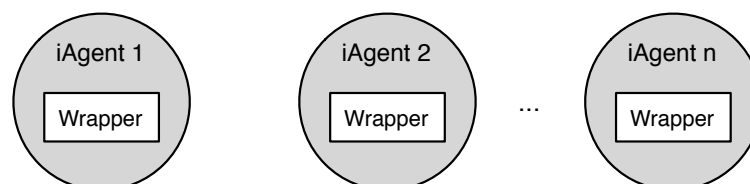


Abbildung 6.2: Informationsagenten mit integriertem Wrapper

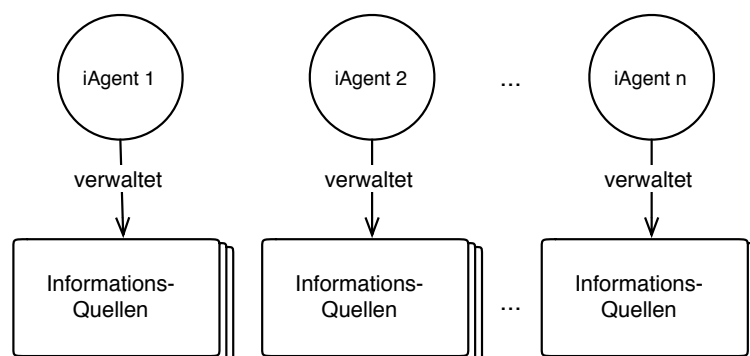


Abbildung 6.3: Agenten sind jeweils für die Verwaltung mehrerer Informationsquellen zuständig

gewährleistet ist.

Neben dieser einfachen Art der Verteilung sind natürlich auch andere Arten möglich, die nicht auf der bloßen Anzahl der Quellen basieren. Beispielsweise könnte auch der Auslastungsgrad der Agenten herangezogen werden, so dass Agenten mit wenig Arbeit bzw. Auslastung mehr Quellen zugeordnet bekommen.

In diesem System werden aber keine anderen Verteilungsarten berücksichtigt, so dass hier für spätere Systemerweiterungen weitere Möglichkeiten gegeben sind.

6.4 IE-Handling im Agenten

Am Anfang dieses Kapitels wurden die Anforderungen an die Agenten in der Plattform auf-, aber noch keine Lösungen dafür vorgestellt. In diesem Kapitel werden im folgenden die einzelnen Aufgaben diskutiert und Ansätze entwickelt.

6.4.1 Initiierung der IE

In einem der vorherigen Abschnitte wurde schon erwähnt, dass ein Agent für mehrere Informationsquellen zuständig ist. Für welche genau, wird dabei in einer zentralen Datenbasis – dem Repository – gespeichert, auf das jeder Agent Zugriff hat. Wann jede dieser Quellen von der IE-Komponente verarbeitet wird, wann also die Informationsextraktion initiiert wird, muss der Agent selbst entscheiden.

Dabei muss bedacht werden, dass es viele Quellen gibt, die unregelmässig von den Autoren der Quellen aktualisiert werden und dementsprechend lange über die gleichen Ereignisgruppen verfügen. Bei diesen Quellen ist es nicht sinnvoll, in kurzen Abständen immer wieder Informationsextraktionen durchzuführen.

Im Gegensatz dazu, ist es bei Quellen, die oft um neue Ereignisgruppen ergänzt werden, wichtig, dass die Extraktionen in kürzeren Abständen ausgeführt werden.

Aus diesem Grund wird für jede Quelle ein individueller Aktualisierungs-Rhythmus (in Form einer Zeitspanne) definiert, welcher vom Agenten genutzt wird. Dieser merkt sich dabei, wann die Quelle das letzte Mal an die IE-Komponente übergeben wurde und berechnet dann mittels der Zeitspanne, wann die nächste Extraktion durchzuführen ist (siehe Pseudo-Code 6.1).

```

DEF überwachte_quelle = <alle quellen des agenten>
FOR EACH überwachte_quelle DO
  DEF extraktions_alter = (aktuelle_zeit)-(überwachte_quelle::letztes_update)
  IF extraktions_alter > überwachte_quelle::aktualisierungs_zeitspanne THEN
    RUN information_extraction(überwachte_quelle)
    DEF überwachte_quelle::letztes_update = aktuelle_zeit
  ELSE
    // nichts muss aktualisiert werden
  FI
OD

```

Beispiel 6.1: Pseudo-Code für Zeitpunkt der Informationsextraktion im Agenten

6.4.2 Geokodierung von Ortsangaben

Die Extraktions-Komponente erhält als Eingabe eine Informationsquelle und liefert die extrahierten Ereignisgruppen zurück. Bestandteil einer jeden Gruppe ist unter anderem eine Ortsangabe, wie sie in

Abschnitt 3.3 definiert wurde. Diese Ortsangabe muss, bevor sie im track-u Projekt zum Einsatz kommen kann, in Geokoordinaten umgewandelt werden, da die internen Datenstrukturen, die bei track-u verwendet werden, dieses voraussetzen. Dieser Prozess der Umwandlung wird auch als *Geokodierung* bezeichnet [Wik07b].

Dabei werden im Rahmen dieses Systems einem Ereignis bzw. einer Ereignisgruppe geographische Koordinaten in Form von Breiten- und Längengraden zugewiesen. Zur Berechnung der Grade wird beispielsweise einer Geokodierungs-Applikation² eine Adresse mitgeteilt und die Applikation liefert die gewünschten Geo-Daten zurück, wie in Abbildung 6.4 vereinfacht dargestellt ist. Eine so kodierte Ereignisgruppe wird dementsprechend in der weiteren Arbeit als **Geo-Ereignisgruppe** bezeichnet.

Die Integration dieser Kodierung in das Agentensystem kann dabei auf unterschiedliche Weisen erfolgen, wobei die zwei folgenden näher untersucht werden sollen.

- Kodierung wird von eigens dafür vorgesehenen Agenten vorgenommen
- Jeder Agent verarbeitet anfallende Kodierungsaufgaben selbstständig

Geokodierungs-Agent

Wenn die Kodierung nicht auf Seiten der Informationsagenten stattfindet, sondern eigene Agenten dafür verantwortlich sind, bedeutet dies, dass im Agentensystem einer neuer Typus von Agent hinzukommt – der so genannte *Geokodierungs-Agent* oder auch *Geo-Agent*. Dieser Agent wird z.B. mittels des Directory Facilitators im System registriert, damit die Informationsagenten ihn finden können.

Ein Geo-Agent ist nun in der Lage, Anfragen – unkodierte Ortsangaben – der Informationsagenten entgegenzunehmen und führt dann intern die Kodierung dieser Ortsangaben durch (unter Zuhilfenahme externer Tools). Nach diesem Kodierungsschritt werden dem jeweiligen Informationsagenten die Ergebnisse mitgeteilt.

Ein Geo-Agent muss dabei intern eine Liste verwalten, die die Anfragen (Kodierungs-Aufträge) den einzelnen Informationsagenten eindeutig zuordnen kann. Zusätzlich muss eine Zuordnung zu den Ereignisgruppen erfolgen. Dazu sendet jeder Informationsagent per ACL-Nachricht eine eindeutige Gruppen-ID mit der Anfrage an den Geo-Agenten. Dieser kodiert die Adressinformation aus der Anfrage und sendet

²Diese Kodierungsapplikation ist dabei ein externer Service, der z.B. mit Hilfe einer XML-Schnittstelle angesprochen werden kann.

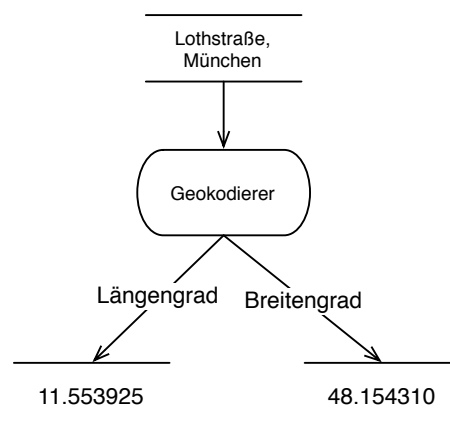


Abbildung 6.4: Beispiel einer Geokodierung

die Geokoordinaten inkl. der Gruppen-Id an den Agenten zurück. Dieser kann nun diese Koordinaten der unkodierten Ereignisgruppe zuordnen. Abbildung 6.5 stellt diesen Ablauf vereinfacht dar.

Problematisch bei diesem Ansatz ist das hohe Kommunikationsaufkommen auf Seiten des Geo-Agenten, falls viele Informationsagenten in der Plattform existieren. Zum Beispiel würden bei 100 Informationsagenten, wobei jeder Agent 10 Quellen verwaltet und aus jeder Quelle 10 Ereignisse pro Stunde extrahiert werden, 10000 Geokodierungs-Anfragen gesendet werden. Die Abbildung 6.6 veranschaulicht diese Problematik.

Aus diesem Grund müssten mehrere Geokodierungs-Agenten genutzt werden, die untereinander die

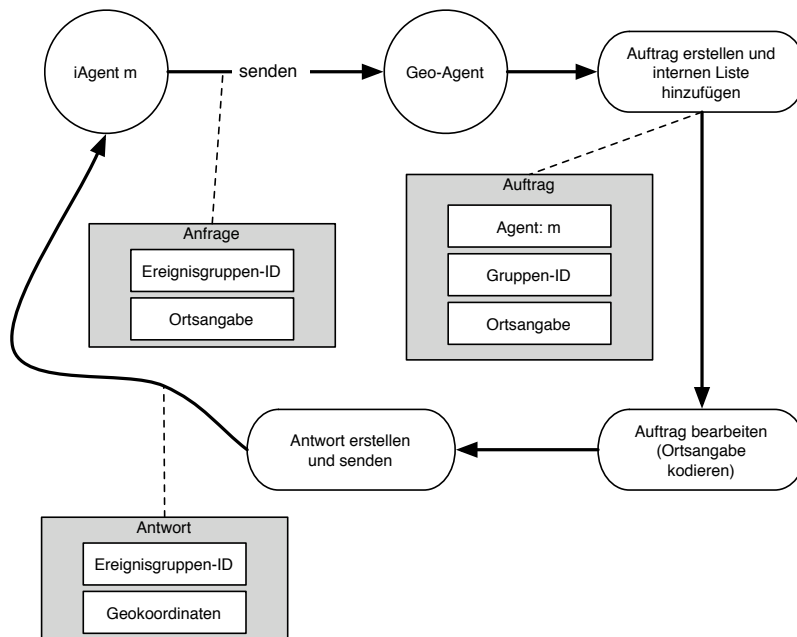


Abbildung 6.5: Ablauf der Geokodierung mit Geo-Agenten

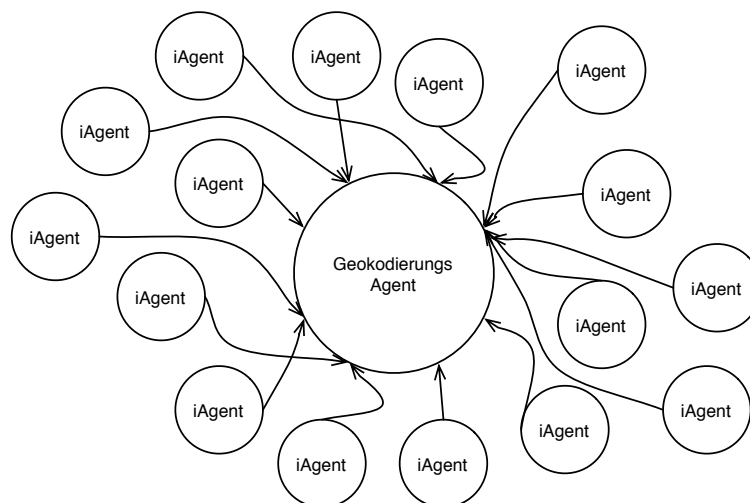


Abbildung 6.6: Erhöhte „Stau-Gefahr“ bei einem Geo-Agenten

Kodierungsaufträge aufteilen. Diese Aufteilung könnte beispielsweise so erfolgen, dass beim Start der Plattform nur ein einziger Geo-Agent existiert. Diesem Agenten wird ein maximales Limit an Aufträgen gegeben, die er gleichzeitig verarbeiten können soll.

Falls dieses Limit im Laufe der Ausführung überschritten wird, erzeugt der Geo-Agent selbstständig einen neuen weiteren Geo-Agenten. Diesem werden die „überschüssigen“ Aufträge sofort zugeteilt.

Damit die Informationsagenten im weiteren Verlauf einen „freien“ Geo-Agenten finden können, werden über den Directory Facilitator jeweils Informationen über den Auslastungsgrad der Geo-Agenten gespeichert. Somit können die Informationsagenten gezielt nach passenden freien Geo-Agenten suchen. Geo-Agenten, die dabei längere Zeit keine neuen Aufträge erhalten und dadurch nicht ausgelastet sind, können sich selbstständig aus der Plattform entfernen.

Probleme: Bei diesem Geo-Agenten-Ansatz, kommen neben der Komplexität bei der Umsetzung noch einige weitere Probleme hinzu. Beispielsweise kann nicht jede Adressinformation geokodiert werden, da sie unter Umständen ungenau oder fehlerhaft extrahiert wurde. Dieses würde aber erst auf Seiten eines Geo-Agenten auffallen, so dass die ganze Kommunikation zwischen den Agenten letztendlich „umsonst“ war.

Aus diesem Grund wird es für dieses System als sinnvoller angesehen, wenn die Geokodierung schon auf Seiten der Informationsagenten stattfindet.

Selbstständige Geokodierung durch die Informationsagenten

Bei diesem Ansatz, der auch letztendlich im System umgesetzt wird, verfügt jeder Informationsagent über eine eigene Geokodierungs-Komponente (diese nutzt intern wieder externe Dienste zur eigentlichen Kodierung). Der Agent kann nun nach einer durchgeführten Extraktion, jede Ortsangabe in den gefundenen Ereignisgruppen sofort kodieren und daraus eine Geo-Ereignisgruppe erstellen.

Zudem kann der Agent, falls eine Kodierung aufgrund unvollständiger bzw. fehlerhafter Ortsangaben nicht möglich ist, ohne viel Kommunikationsaufwand (nur Kommunikation mit dem externen Geokodierungsservice notwendig) die Ereignisgruppe verwerfen.

Abbildung 6.7 zeigt nun das aktualisierte Modell der Informationsagenten, in dem neben dem Wrapper jetzt die Geokodierungs-Komponente hinzugekommen ist.

6.4.3 Verhaltensweisen nach Extraktion

Nach der Geokodierung beeinflussen einige Faktoren die weitere Verarbeitung der Ereignisgruppen-Listen, die von der IE-Komponente geliefert werden. Diese Faktoren sind abhängig von Einstellungen, die in der Managementapplikation getroffen werden. Dabei geht es vor allen Dingen um die Fragen, welche Ereignisse im Repository gespeichert werden sollen und welche Ereignisse aus dem Repository wieder entfernt werden können.

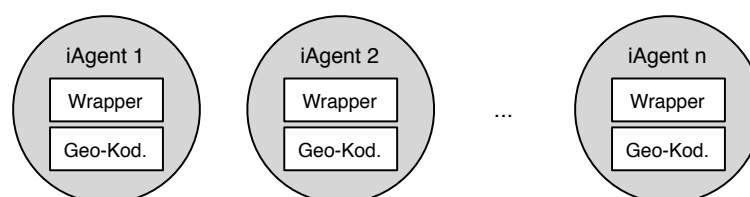


Abbildung 6.7: Informationsagent mit Wrapper- und Geokodierungs-Komponente

Haltbarkeit von Ereignissen

Bei der Erstellung³ einer Quelle in der Managementapplikation kann vom Knowledge Engineer (KE) angegeben werden, wie lange ein in dieser Quelle gefundenes Ereignis im Repository als „gültig“ geführt werden soll bzw. im Repository vorhanden sein soll bevor es gelöscht wird. Es handelt sich dabei also um eine Art **Haltbarkeitsdauer**, die in diesem System in Form von Tagen angegeben wird und aufgrund von Erfahrungswerten des KE für jede Quelle festgelegt wird.

Der Grund für diese Einstellung ist der, dass aus den Quellen keine Zeitspanne extrahiert wird, sondern höchstens nur ein Start-Zeitpunkt. Falls sogar dieser nicht Teil der Informationen in der Quelle ist, wird automatisch der Zeitpunkt (Tag) des Auffindens genommen und als *time* in der Ereignisgruppe gespeichert. Das Haltbarkeitsdatum eines Ereignisses berechnet sich demnach wie folgt:

$$\text{haltbarkeitsdatum} = \text{ereignis-zeitpunkt} + \text{haltbarkeitsdauer}$$

bzw.

$$\text{haltbarkeitsdatum} = \text{extraktions-zeitpunkt} + \text{haltbarkeitsdauer}$$

Bei jeder Durchführung einer Extraktion auf der Quelle überprüft der Agent gleichzeitig noch das Repository auf abgelaufene Ereignisse ($\text{haltbarkeitsdatum} < \text{aktuellesdatum}$) und löscht diese – bereinigt also die Datenbasis.

Neben einer expliziten Angabe einer Haltbarkeitsdauer kann auch noch ein anderes Vorgehen, zum Entfernen von Ereignissen aus dem Repository, vom Knowledge Engineer gewählt werden.

Löschen bei Aktualisierung

Eine andere Möglichkeit stellt das komplette Löschen aller aus einer Quellen stammenden Ereignisse dar, falls auf dieser Quelle eine aktuellere erfolgreiche Extraktion durchgeführt wurde. Dieses Vorgehen ergibt beispielsweise bei kleineren Quellen einen Sinn, wo das komplette extrahieren, aller in der Quelle vorhandenen Ereignisse, nicht zu viel Zeit beansprucht.

Nach dem kompletten Löschen werden alle in der Ereignisgruppen-Liste vorhandenen Orte geokodiert und es entsteht eine gleich große Liste an Geo-Ereignisgruppen, welche dann ins Repository übernommen werden kann.

Neue Ereignisse aufnehmen bei Aktualisierung

Wenn kein automatisches Löschen schon vorhandener Ereignisse gewünscht wird, kann bei einer Quelle die Einstellung gemacht werden, dass nur neue Ereignisse aufgenommen werden sollen. Die Problematik ist hierbei herauszufinden, welche Ereignisse neu und welche schon im Repository vorhanden sind. Diese Entscheidung kann dank der Eindeutigkeit der Geo-Ereignisgruppen getroffen werden. Das System betrachtet eine Geo-Ereignisgruppe durch die Kombination der Geokoordinaten, der Quelle und des Startzeitpunktes als eindeutig – bildet also hiermit einen Schlüssel.

Nachdem nun die IE-Komponente die Gruppen-Liste an den Agenten weitergegeben hat, werden so lange die darin befindlichen Ereignisgruppen geokodiert, bis n doppelte Schlüssel auftauchen. In diesem Fall wird die Geokodierung abgebrochen, da das System von nun an annimmt, dass die weiteren Ereignisse ebenfalls schon im Repository vorhanden sind.

Je größer der Wert von n vom Knowledge Engineer festgelegt wird, desto sicherer ist die Vermutung,

³Mit „Erstellen“ ist dabei nicht das Erzeugen der Quelle im Internet gemeint, sondern das Hinzufügen zum System

dass der Rest der Liste schon vorhanden ist.

Der Vorteil dieser Überprüfung ist der, dass bei großen Listen nicht unnötiges Geokodieren vorgenommen wird.

Bemerkung: Eine zusätzliche Erweiterungsmöglichkeit wäre die Integration dieser „Dubletten-Überprüfung“ auf Ebene des Wrappers. Hierdurch würde das System sich nicht nur weitere Geokodierungen sparen, sondern auch die Ausführung weiterer Extraction-Pattern.

Fehlgeschlagene Extraktion behandeln

Nach einer erfolgreichen Extraktion durch den Wrapper steht eine Liste von Ereignisgruppen zur Verfügung. Es kann aber auch vorkommen, dass Änderungen an der Quelle (z.B. ein neues Layout) den IE-Prozess behindert haben, indem Regeln nicht mehr anwendbar waren. In diesem Fall liefert der Wrapper eine leere Liste zurück.

Der Informationsagent überprüft nun anhand der Länge der zurückgelieferten Liste, ob ein Fehler aufgetreten ist oder nicht. Im Fehlerfall wird die Informationsquelle entsprechend in der Datenbank markiert, so dass der Knowledge Engineer diese später kontrollieren kann. Diese Markierung erfolgt, in dem die Id der Quelle, die Id der verwendeten Extraction-Chain und der Zeitpunkt an dem der Fehler aufgetreten ist, in die Datenbank geschrieben wird.

6.4.4 Berücksichtigung von Einflüssen auf Sicherheitszonen

Wie am Anfang der Thesis bereits erwähnt wurde, können extrahierte Informationen unter anderem auch zur Einflussnahme auf Sicherheits-Level von Zonen im track-u System verwendet werden. Ob ein negativer oder positiver Einfluss ausgeübt wird, wird ebenfalls in der Verwaltungsapplikation für jede Quelle definiert (z.B. durch ein Intervall $\{-1, 0, +1\}$) und wird in Kapitel 7 noch näher beschrieben.

Falls nun eine Ereignisgruppe gefunden wird und die entsprechenden Geokoordinaten in einer der im track-u System vorhandenen Zonen liegen, dann wirkt sich der vorher definierte Quellen-Einfluss auf die entsprechenden Zonen aus. Hierzu muss die Möglichkeit bestehen, dass die track-u Geo-Datenbank angefragt und geändert werden kann⁴.

6.4.5 Aktualisierung Ausführungszeitpunkt

In der Verwaltungsapplikation, die in Kapitel 7 vorgestellt wird, dienen einige Funktionen rein informativen Zwecken für den Benutzer. Beispielsweise kann dem Benutzer angezeigt werden, ob Agenten noch aktiv sind oder nicht. Nicht-aktive Agenten können z.B. durch Abstürze der Agentenplattform hervorgerufen werden.

Damit die Verwaltungsapplikation diese Informationen anbieten kann, benötigt sie vom Agenten den Zeitpunkt seiner letzten erfolgreichen Ausführung. Der Agent speichert also in einem regelmässigen Rhythmus einen aktuellen Zeitstempel in der Datenbasis ab. Auf Basis dieses Zeitpunktes werden eventuelle Ausfälle erkannt, wie in 7.6 noch gezeigt werden wird.

⁴Diese aktive Änderung der Geo-Datenbank wird in dieser Arbeit zwar erwähnt, aber nicht umgesetzt. Bietet aber Möglichkeiten für spätere Erweiterungen.

6.5 Zusatzaufgaben der Plattform

Neben den eben diskutierten Aufgaben der einzelnen Agenten, gibt es auch noch allgemeine Anforderungen, die vom Agentensystem erfüllt werden müssen.

6.5.1 Zustands-Wiederherstellung nach Ausfällen

Beim Einsatz des Systems in einer Produktivumgebung, verwaltet eine große Anzahl von Agenten eine noch größere Anzahl von Informationsquellen.

Da es hierbei vorkommen kann, dass Teile des Systems – in diesem Fall die Agentenplattform – ausfallen können, müssen Maßnahmen zur Zustands-Wiederherstellung getroffen werden. Das bedeutet, dass das Agentensystem in der Lage sein muss, bei einem Neustart der Plattform alle Agenten wiederherzustellen.

Zudem müssen auch, die von den Agenten verwalteten Quellen, wieder nach einem Neustart automatisch zugeordnet werden können.

Aus diesem Grund wird eine Liste aller Agenten und ebenfalls die Quellenzuordnung persistent gespeichert. Bei einer Aktivierung bzw. Re-Aktivierung der Plattform wird diese Liste durchlaufen und jeder Agent mit seiner eindeutigen AID wiederhergestellt. Nachdem der Agent in der Plattform wieder aktiv ist und läuft, kann er durch die vorgenommene Quellenzuordnung wieder auf den ursprünglichen Quellen arbeiten.

Bemerkung: Wie im Umsetzungskapitel noch gezeigt werden wird, übernimmt diesen Wiederherstellungsprozess die Komponente, die auch den XML-RPC-Server stellt.

6.5.2 Erstellung neuer Agenten

Wie in 6.3 beschrieben wurde, ist die Anzahl der Quellen pro Agent durch eine obere Grenze limitiert. Dadurch muss das Agentensystem in der Lage sein, neue Agenten bei Bedarf erstellen zu können.

Hierbei gibt es zum einen die Möglichkeit, dass die Agenten selbst neue Agenten erstellen können und zum anderen wäre ein Ansatz, dass die Erstellung von der Verwaltungsapplikation explizit initiiert wird. In dem System wird der letztere Ansatz gewählt, aber der Vollständigkeit halber sollen beide im folgenden diskutiert werden.

Erstellung durch Agenten

Falls die Erstellung neuer Agenten durch die Agenten selbst vorgenommen werden soll, muss jeder Informationsagent in regelmäßigen Abständen in der Datenbasis abfragen, welche Informationsquellen noch nicht einem Agenten zugeordnet wurden. Falls er solche „freien“ Quellen findet und er selbst noch nicht sein Quellen-Limit erreicht hat, ordnet er sich die möglichen freien Quellen zu.

Falls er die obere Grenze für Quellen erreicht hat, initiiert er eigenständig die Erstellung eines – vom Ursprungsagenten unabhängigen – neuen Agenten.

Dieser neue Informationsagent führt die selben Schritte aus und findet dementsprechend die Quellen, die der vorherige Agent nicht mehr aufnehmen konnte.

Diese Möglichkeit hat den Vorteil, dass die Verwaltungsapplikation entlastet wird und den Nachteil, dass die Komplexität der Informationsagenten ansteigt. Zudem kann es vorkommen, dass aufgrund der Parallelität der Agenten gleichzeitig eine freie Quelle gefunden und mehreren Agenten zugeteilt wird. Die

Lösung dieser Probleme ziehen aufwendigere Agenten nach sich. Die Agenten sollen jedoch möglichst einfache Einheiten bleiben und somit wird der Erstellung über die Managementapplikation Vorzug gegeben.

Erstellung über Managementapplikation

Die Entscheidung, ob ein neuer Agent benötigt wird, wird durch die Applikationslogik der Verwaltungsapplikation getroffen. Diese Logik wird in Kapitel 7 erläutert. An dieser Stelle soll nun darauf eingegangen werden, wie die Verwaltungsapplikation eine Neuerstellung eines Agenten initiieren kann.

Da Verwaltungsapplikation und Multiagentensystem u.U. auf unterschiedlichen physikalischen Geräten betrieben werden (unterschiedliche Server) ist eine Kommunikation zwischen beiden erforderlich. Diese Kommunikation erfolgt auf Basis von XML-RPC Aufrufen. Hierzu wird auf dem Server des Agentensystems zusätzlich ein XML-RPC-Server ausgeführt. Dieser kann von der Verwaltungsapplikation aufgerufen werden und aufgrund dessen wird in der Agentenplattform ein neuer Agent erstellt. Abbildung 6.8 zeigt diese Verbindung zwischen beiden Komponenten.

Dieser neue Agent erkennt anhand der Datenbasis, die er in regelmäßigen Abständen prüft, welche neue Quelle er überwachen soll. Diese Quellenzuordnung wurde dabei in einem vorherigen Schritt schon von der Verwaltungsapplikation getroffen.

6.6 Aktualisierte Systemstruktur

Durch den in diesem Kapitel entwickelten Aufbau des Multiagentensystems, ergibt sich die in Abbildung 6.9 aktualisierte Systemstruktur.

Es ist dabei der Server vorhanden, auf dem die Agentenplattform läuft. In der Plattform existieren die Informationsagenten, welche intern über die Wrapper-Komponente, sowie die Geokodierungskomponente verfügen. Letztere nutzt dabei einen externen Service, um die Abbildung von Adressdaten auf Geokoordinaten vorzunehmen.

Zudem ist der XML-RPC Server hinzugekommen, der mit der Managementapplikation kommuniziert. Zusätzlich ist eine Anbindung der Informationsagenten an das Repository eingefügt worden, über die die Agenten auf die notwendigen Daten zugreifen können, die für die Extraktion gebraucht werden.

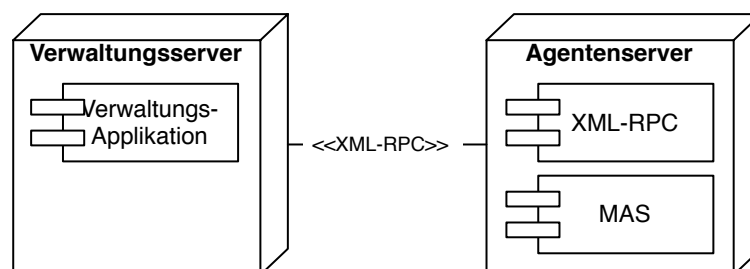


Abbildung 6.8: Verbindung zwischen Verwaltungsapplikation und MAS (inkl. XML-RPC)

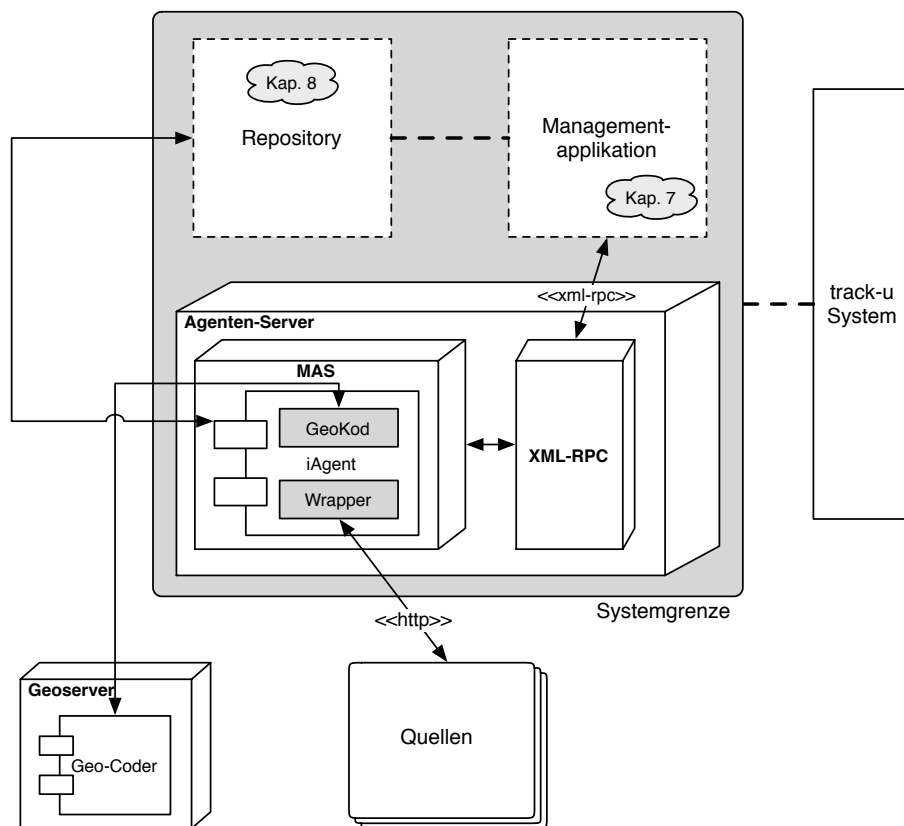


Abbildung 6.9: Aktualisierte Systemstruktur nach Entwurf des MAS

7

Entwurf der Managementapplikation

Die beiden vorherigen Entwurfskapitel haben die IE-Komponente und das Agentensystem vorgestellt. Beide Komponenten führen ihre Tätigkeiten im Hintergrund aus und sind nach Außen nicht direkt sichtbar. In diesem Kapitel wird nun mit der Management- bzw. der Verwaltungsapplikation die Schnittstelle zwischen Benutzer und System entworfen. Der Benutzer ist in diesem Fall vor allen Dingen der Knowledge Engineer, dessen Aufgabe es ist, die Regeln für den Extraktionsprozess zu definieren und diese Regeln einer Quelle anhand einer Extraction-Chain zuzuweisen. Zudem wurde in den vorherigen Kapiteln oft der informative Charakter der Verwaltungsapplikation erwähnt. Dieser Aspekt wird ebenfalls im vorliegenden Kapitel näher erläutert.

7.1 Anforderungen

Das in Kapitel 6 entworfene Agentensystem entzieht sich mehr oder weniger den Blicken der Benutzer und verrichtet seine Extraktions-Dienste selbständig im Hintergrund. Die einzige wirkliche Schnittstelle zum Benutzer – dem Knowledge Engineer – stellt die Managementapplikation dar. Diese muss einige Anforderungen erfüllen, die im folgenden aufgelistet werden.

- Erstellen der Extraktionsregeln
- Hinzufügen neuer Informationsquellen zum System unter Verwendung der definierten Extraktionsregeln
- Kommunikation mit der Agentenplattform via XML-RPC, um neue Agenten erstellen zu können
- Informative und administrative Aufgaben

Zusätzlich zu diesen Grundanforderungen muss das System den Knowledge Engineer durch eine benutzerfreundliche Schnittstelle bei Erfüllung seiner Tätigkeiten unterstützen.

Bevor nun in den Folgeabschnitten die Anforderungen noch im einzelnen detailliert behandelt werden, soll zunächst auf die Architektur der Applikation eingegangen werden.

7.2 Architektur

Es existieren unterschiedliche Möglichkeiten, die Verwaltungsapplikation umzusetzen und dem Benutzer zugänglich zu machen. In diesem Abschnitt sollen zwei Architekturen diskutiert werden. Zum einen die Umsetzung der Applikation

- als eine **Stand-Alone Anwendung** auf einem Rechner und zum anderen
- als Realisierung in Form einer **Webapplikation**.

Bei der ersten Möglichkeit wird auf dem Computer des Benutzers lokal eine Applikation ausgeführt. Diese stellt die Benutzerschnittstelle zur Verfügung und kann mit dem Multiagentensystem und mit der Datenbank kommunizieren, um beispielsweise Agenten zu erstellen oder Daten abzufragen.

Der Vorteil ist hierbei, dass die Benutzeroberfläche im Stil der verwendeten Betriebssystem-Oberfläche umsetzbar ist und vielfältige Möglichkeiten der Programmierung durch die API bietet.

Der Nachteil ist aber, dass für jeden neuen Benutzer der hinzukommt, auf dessen Computer die Applikation neu installiert und eingerichtet werden muss. Falls zudem noch Aktualisierungen an der Software vorgenommen werden, müssen alle Versionen auf den einzelnen Computern aktualisiert werden.

Ein weiterer Nachteil ist die Notwendigkeit einer Laufzeitumgebung, falls zur Umsetzung der Applikation eine plattformunabhängige Sprache wie z.B. Java gewählt wurde. Dieses zieht noch zusätzlichen Aufwand mit sich, da auch diese Umgebung teilweise aktualisiert werden muss.

Durch Techniken wie beispielsweise *Java Web Start* kann zwar eben genanntes Problem der Aktualisierung umgangen werden, da die Applikation bei jedem Start automatisch über das Internet prüft, ob eine aktuellere Version vorliegt und diese dann ggf. verwendet. Trotzdem bleibt auch hier das Problem der benötigten Laufzeitumgebung bestehen.

Auf diesen Gründen stellt die Umsetzung als Webapplikation eine gute Alternative dar. Hierbei läuft die ganze Applikationslogik auf einem Webserver ab, welcher auch die Aufgabe der Kommunikation mit dem Agentensystem bzw. der Datenbank übernimmt.

Als Client kann dabei jeder Webbrowser genutzt werden, der die Technologien unterstützt, die in der Webapplikation später eingesetzt werden (Javascript, Ajax usw.). Ein weiterer Vorteil ist hierbei, dass heutzutage nahezu jeder Computer standardmässig über einen Webbrowser verfügt, im Gegensatz zu einer passenden Laufzeitumgebung (siehe oben).

Da die Webapplikation übers Internet zentral erreichbar ist, kann sie zudem unabhängig vom Aufenthaltsort des Benutzers genutzt werden, solange ein internet-fähiger Computer zur Verfügung steht.

Zusätzlich bieten aktuelle Technologien wie beispielsweise Ajax die Möglichkeit, Webapplikationen mit immer mehr dynamischen Funktionalitäten auszustatten, die die Benutzeroberfläche der Applikation, der einer Stand-Alone-Anwendung, immer näher bringen und somit auch diesen Vorteil schwinden lassen.

Nach dieser Entscheidung für eine Architektur wird nun mit dem Entwurf der Umsetzung der einzelnen Anforderungen begonnen.

7.3 Erstellen der Extraktionsregeln

Das Erstellen, der für die Informationsextraktion benötigten Regeln, stellt den Hauptaufgabenbereich der Managementapplikation dar. Nachdem der Knowledge Engineer die Analysephase auf den Quellen durchgeführt hat, werden die einzelnen Bestandteile der Extraktionsketten (siehe 5.4.1) definiert. Dazu gehören im einzelnen

- die Process Expressions,
- die Precision- bzw. Raw-Pattern und

- die Extraction-Pattern.

Letztere werden danach zu einer Extraction-Chain zusammengesetzt.

Der Prozess der Regelerstellung bietet dabei zahlreiche Möglichkeiten, wie der Benutzer unterstützt werden kann. Im folgenden sollen einige dieser Möglichkeiten kurz vorgestellt werden.

7.3.1 Intuitive Benutzeroberfläche bei Verkettung

Der Vorteil der in dieser Arbeit entwickelten Informationsextraktion gegenüber anderen Systemen, ist die Möglichkeit der Verkettung einzelner kompakter Extraktionseinheiten. Diese Verkettung erlaubt es, einen Großteil der im Internet verfügbaren Informationsquellen zu extrahieren, ohne dass der grundlegende Extraktions-Algorithmus vom Knowledge Engineer angepasst werden muss.

Die Erstellung – besonders der Process Expressions – setzt dabei zwar Expertenwissen voraus, aber wenn dieses vorhanden ist, ist die Definition der kompletten Regeln relativ unkompliziert zu erreichen. Dabei kann dieser Prozess durch den Einsatz einer intuitiven Benutzeroberfläche noch zusätzlich vereinfacht werden. Dazu bietet es sich beispielsweise an, alle schon einmal erstellten Extraction-Pattern in einer Art *Pattern-Pool* zu sammeln, um sie danach wiederverwenden zu können. Dabei können beliebig viele Pattern aus dem Pool „gezogen“ werden, um sie danach dynamisch in die Liste zu integrieren (siehe Abb. 7.1).

7.3.2 Semi-automatische Erstellung regulärer Ausdrücke

Die eben beschriebene Verkettung von Extraction-Pattern setzt voraus, dass die Pattern bereits bestehen. Das heißt also, dass Precision- und Raw-Pattern existieren, die wiederum selbst Process-Expressions nutzen. Bei der Erstellung der dabei benötigten Ausdrücke (z.B. regulär oder XPath) kann der Knowledge Engineer noch zusätzlich unterstützt werden.

So kann z.B. die gewünschte Quelle während der Regelerstellung geladen werden und der Knowledge Engineer selektiert die Bereiche im Dokument, die extrahiert werden sollen. Die Applikation könnte dabei dem Benutzer an der selektierten Position umliegende Struktur-Elemente anzeigen und ihn somit bei der Erstellung der Ausdrücke unterstützen.

Eine fortschrittlichere Hilfe wäre zudem die automatische Erstellung passender Ausdrücke. Es würde

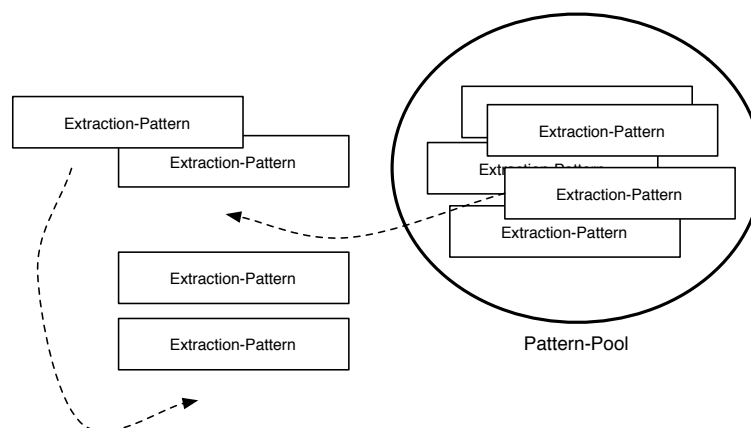


Abbildung 7.1: Flexible Zusammenstellung von Extraktionsketten

dabei eine Art semi-automatische Ausdruckserstellung erfolgen, bei der das System die Delimiter automatisch auswählt, die den gewünschten Bereich eindeutig im Dokument identifizieren. Auf Basis dieser würden danach die Process-Expressions erstellt werden.

7.3.3 Vorschau der Extraktion

Eine weitere Unterstützung ist die Ausführung der vom Knowledge Engineer definierten Ausdrücke innerhalb der Webapplikation auf dem geladenen Quell-Dokument. Der Knowledge Engineer schreibt dabei einen Ausdruck und dieser wird direkt auf dem Text ausgeführt, wobei die selektierten Bereiche hervorgehoben werden. Dadurch kann der Benutzer seinen Ausdruck überprüfen und ggf. modifizieren.

7.4 Hinzufügen neuer Informationsquellen

Die Erstellung der Extraktionsregeln stellt, wie schon erwähnt wurde, den aufwendigsten Teil der Verwaltungsapplikation dar. Das eigentliche Hinzufügen der Informationsquelle zum System benötigt stattdessen nicht so viele Einzelschritte und es müssen nur folgende Pflichtdaten angegeben werden:

- Die **Uniform Resource Identifier** (URI) der Quelle. Dieser ist im Falle dieses Systems immer ein *Uniform Resource Locator* (URL), der entweder eine Webseite oder einen Newsfeed identifiziert.
- Eine kurze **Beschreibung der Quelle**. Diese kann später genutzt werden, um beispielsweise den track-u Nutzern die Herkunft eines Ereignisses anzuzeigen (z.B. „Die folgende Information stammt von der Polizei Hamburg: ...“).
- Den **Rhythmus** in dem die Überprüfung der Quelle vorgenommen werden soll (siehe dazu auch 6.4.1). Diese Zeitspanne wird dabei in Minuten angegeben.
- Den **Effekt**, den Ereignisse aus dieser Quelle auf Zonen des track-u Systems haben. Mögliche Werte werden in Tabelle 7.1 beschrieben.

Eingabewert	Bedeutung
0	Keinen Einfluss auf das Zonenlevel
-1	Negativen Einfluss auf das Level
+1	Positiven Einfluss auf das Level

Tabelle 7.1: Mögliche Einfluss-Werte auf Sicherheitslevel einer Zone

Wie schon in 6.4.4 erwähnt wurde, ist es die Aufgabe des Agenten, abhängig von dem Effekt, Änderungen an der Geo-Datenbank vorzunehmen. In dieser Arbeit wird dieser Bereich aber nicht umgesetzt. Zudem ist es bei vielen Quellen auch nicht möglich einen positiven oder negativen Effekt anzugeben. Dieses ist vor allen Dingen bei Informationsquellen der Fall, die viele unterschiedliche Arten von Ereignissen anbieten. Hier kann nicht pauschal gesagt werden, dass alle Ereignisse positive oder negative Einflüsse haben. In diesem Fall sollte für die Quelle 0 als Einflussfaktor gewählt werden.

- Die **Gültigkeitsdauer** der gefundenen Ereignisse (siehe 6.4.3). Dabei wird die „Haltbarkeit“ in Tagen angegeben.

- Die **Verhaltensweise nach der Extraktion**. Hier wird, wie in 6.4.3 beschrieben wurde, eingestellt, wie alte Ereignisse im Repository behandelt werden sollen.
- Angabe, ob am Anfang eine **Formular-Anfrage** durchgeführt werden soll. Falls ja, Definition der benötigten Parameter (siehe 5.5)
- Angabe, ob das **Wurzel-Dokument** in der Quelle als DOM-Baum oder Zeichenkette geladen werden soll.

Optional kann eine Quelle auch noch mit Schlüsselwörtern – so genannten **Tags** – versehen werden (siehe Abbildung 7.2). Im Gegensatz zur oben genannten natürlich-sprachlichen Beschreibung, entsprechen Tags eher einer Form von Metadaten¹ und bieten somit Zusatzinformationen zu den Quellen.

Durch das Hinzufügen gleichnamiger Tags zu mehreren Quellen kann somit auch eine automatische Kategorisierung vorgenommen werden. An das System können dadurch Anfragen gestellt werden wie beispielsweise:

- *Zeige alle Ereignisse, die in Quellen mit den Tags „Polizei“ und „München“ aufgetreten sind*
- *berücksichtige nur Zonen-Effekte aus Quellen mit Tags „Bombenwarnungen“*

7.5 Kommunikation mit der Agentenplattform

In 6.5.2 wurde beschrieben, dass der Server auf dem die Agentenplattform abläuft, auch einen XML-RPC-Service zur Verfügung stellt, um darüber die Erstellung neuer Agenten anstossen zu können. Die Entscheidung, ob ein neuer Agent erstellt wird und welche Informationsquellen dieser überwachen soll, wird in der Managementapplikation getroffen.

Dazu wird, nachdem die Informationsquelle gespeichert wurde (siehe 7.4), überprüft, ob ein Agent zur Verfügung steht, der noch weitere Quellen aufnehmen kann. Um dieses zu überprüfen, wird zu jedem Agenten in der Datenbank die Anzahl der von ihm überwachten Quellen gespeichert und mit einem maximalen Limit verglichen. Stehen Agenten zur Verfügung deren Anzahl kleiner als dieses Limit ist, so wird die neue Quelle einem dieser Agenten zugewiesen.

¹Metadaten sind Daten, die andere Daten näher beschreiben. Das grundlegende Konzept von Metadaten kann beispielsweise in einer Bibliothek gefunden werden, wobei ein Bibliothekskatalog Metadaten über vorhandene Bücher enthält und somit ein schnelles Auffinden dieser erleichtert wird [MM04].

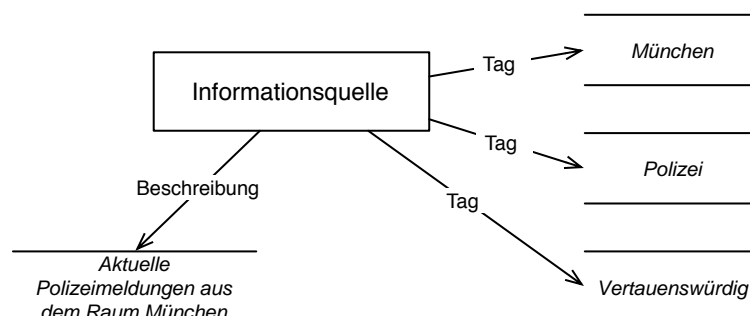


Abbildung 7.2: Informationsquelle mit einer Beschreibung und drei Tags, die Metadaten liefern.

Im anderen Fall, bei dem kein freier Agent gefunden wird, speichert das System die Daten für einen neuen Agenten in der Datenbank (Name des Agenten, Anzahl der überwachten Quellen = 1 und Erstellungszeitpunkt) und erhält daraufhin eine neue eindeutige Agenten-Id². Diese Id wird der neuen Informationsquelle zugewiesen, da der Agent nun für diese zuständig ist.

Bis zu diesem Zeitpunkt hat die Agentenplattform weder Kenntnisse von der neuen Informationsquelle noch vom neuen Agenten. Dieses ändert sich nun, da die Managementapplikation einen XML-RPC-Aufruf an die XML-RPC-Schnittstelle des Agentenservers sendet, bei dem die Id des neuen Agenten übermittelt wird.

Aufgrund dieser Id kann das MAS nun einen neuen Agenten erstellen und durch die Zuweisung der Quelle zum Agenten auch mit der Informationsgewinnung beginnen (siehe Abb. 7.3).

7.6 Informative und administrative Aufgaben

Neben den oben genannten grundlegenden Funktionen, erfüllt die Verwaltungsapplikation auch informative Aufgaben und stellt dem Benutzer einige Möglichkeiten der Überwachung der extrahierten Informationen zur Verfügung. Zudem können auch Daten, die im Repository gespeichert wurden, vom Benutzer administriert werden.

Zu den informativen Aufgaben zählen dabei

- das Anzeigen extrahierter Ereignisse,
- die Auflistung aller verwalteten Informationsquellen und
- Informationen zu den vom System verwalteten Agenten.

So werden beispielsweise alle extrahierten Ereignisse in einer Landkarte angezeigt, mit der zu jedem Ereignis die extrahierten Informationen abgerufen werden können. Dieses dient vor allem zur Kontrolle und Erkennung von Fehlern.

Bei der Auflistung der Quellen kann überprüft werden, welche Quellen von welchem Agenten überwacht werden und wann diese Quellen das letzte Mal extrahiert wurden.

Bei den Informationen, die zu den Agenten angezeigt werden, handelt es sich um eine Liste der Agenten, die in der Plattform aktiv sind. Dabei wird zu jedem Agenten angezeigt, zu welchem Zeitpunkt t er

²Diese Agenten-Id ist nicht mit der AID zu verwechseln, die von der Agenten-Plattform vergeben wird. Es handelt sich hierbei um eine intern verwendete Id.

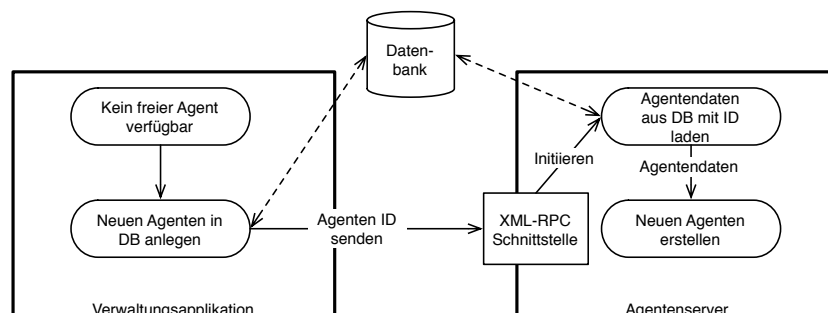


Abbildung 7.3: Neuen Agenten über Verwaltungsapplikation erstellen

das letzte Mal ausgeführt worden ist (Hinweis: jeder Agent speichert automatisch in regelmässigen Abständen die aktuelle Zeit).

Mit Hilfe des aktuellen Zeitstempels t' kann somit die Differenz zwischen beiden Zeitpunkten errechnet werden ($t' - t = \Delta t$) und mit einer Toleranz-Zeitspanne verglichen werden. Die Differenz Δt darf dabei diese Zeitspanne nicht überschreiten. Falls dennoch eine Überschreitung auftritt, gilt der Agent nicht mehr als aktiv und muss vom Benutzer kontrolliert werden, der daraufhin weitere manuelle Schritte ausführen kann (z.B. Überprüfung der Agentenplattform).

Natürlich kann auch eine automatische Fehlerbehandlung erfolgen. Beispielsweise könnte der Agent gelöscht und ein neuer erstellt werden, der die Informationsquellen des alten Agenten dann übernimmt. In diesem System wird jedoch keine automatische Behandlung vorgenommen.

Da das Repository über eine große Anzahl von Daten verfügt (Ereignisse, Regeln, Agenten usw.) und diese teilweise auch fehlerhaft sein können, bietet die Verwaltungsapplikation die Möglichkeit, die Daten zu administrieren. Dieses ist wichtig, da das track-u System aus dieser Datenbasis später Informationen zieht und sie dem Endkunden präsentiert. Falls nun bei der Extraktion beispielsweise eine offensichtlich fehlerhafte Beschreibung extrahiert wurde, kann diese manuell nachbearbeitet werden. Auch können Ereignisse gelöscht oder abgeändert werden, falls dieses notwendig sein sollte.

7.7 Aktualisierte Systemstruktur

Die in diesem Kapitel angestellten Überlegungen zum Entwurf der Verwaltungsapplikation führen zu der aktualisierten Systemstruktur wie sie in Abbildung 7.4 dargestellt wird.

Hierbei ist nun der Webserver integriert worden, auf dem die Verwaltungsapplikation in Form einer Webapplikation läuft. Zudem ist die Verbindung zum Repository hergestellt worden, in welchem die Daten gespeichert und aus dem die Daten für die informativen Aufgaben geladen werden.

Zusätzlich ist zwischen der Webapplikation und den Informationsquellen eine Verbindung entstanden. Diese erlaubt es, bei der Erstellung der Extraktionsregeln, das Quellen-Dokument in die Applikation zu laden und mit der Dokumenten-Struktur zu arbeiten.

Ebenfalls wurde die XML-RPC-Verbindung an die Webapplikation angeschlossen, damit die Kommunikation mit der Agentenplattform möglich ist.

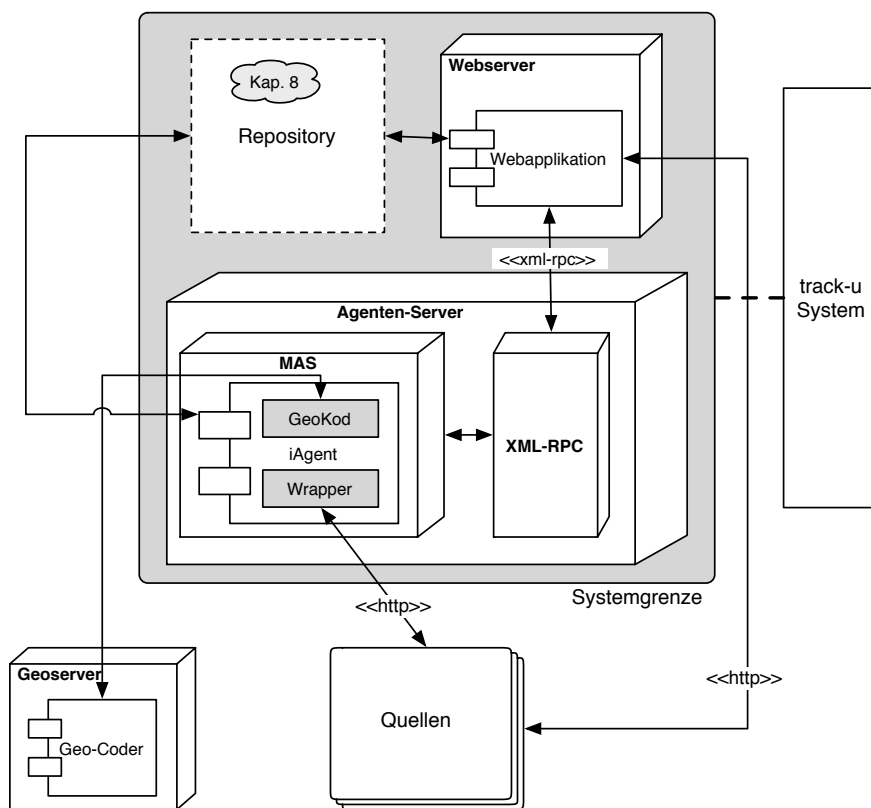


Abbildung 7.4: Aktualisierte Systemstruktur nach Entwurf der Managementapplikation

8

Repository-Design

Für die Informationsextraktion werden zahlreiche Daten benötigt und es fallen während der Extraktion Daten an. All diese Daten werden in einem Datenspeicher – dem Repository – persistent gespeichert.

Dazu zählen die extrahierten Ereignisse, Einstellungen zu Informationsquellen und Informationsagenten sowie die ganzen Extraktionsregeln, die zu den Quellen definiert wurden. Zudem dient das Repository auch dazu, dass das track-u System aus ihm die notwendigen Ereignisdaten abfragen kann, stellt also die Schnittstelle zwischen beiden Systemen dar.

Als Plattform für das Repository dient dabei eine relationale Datenbank, die entweder mit auf dem Webserver abläuft oder auf dem Agentenserver untergebracht ist.

Das folgende Kapitel beschäftigt sich nun mit dem Design des Repositories und stellt dessen Aufbau dar.

8.1 Struktur

Bevor mit der Beschreibung der einzelnen Einheiten begonnen wird, soll zunächst einmal die komplette Datenbankstruktur aufgezeigt werden, um deren Aufbau zu verdeutlichen. In Abbildung 8.1 sind die Einheiten und ihre Verbindungen untereinander dargestellt. Dabei wurden aufgrund der Übersichtlichkeit keine Datentypen für die Attribute mit aufgenommen.

Für das weitere Kapitel wird nun eine Aufteilung vorgenommen, welche die logisch eng verbundenen Einheiten für die späteren Beschreibungen zusammenfasst. Zu diesen Einheiten zählen

- die **Extraktionsregeln**,
- die **Informationsquellen** inkl. ihrer Tags, zugewiesenen Agenten, eventuellen Formularanfragen und Fehler-Protokolle und
- die extrahierten **Ereignisse**.

8.2 Extraktionsregeln

Für die persistente Speicherung der Extraktionsregeln sind die Tabellen `process expressions`, `precision patterns`, `raw patterns`, `extraction patterns`, `extraction chain links` und `extraction chains` zuständig.

Dabei stellen die Process-Expressions die kleinste Einheit bei der Zusammensetzung der Regeln dar und werden in der gleichnamigen Tabelle `process expressions` gespeichert. Hier wird entweder ein

regulärer Ausdruck oder ein XPath-Ausdruck gespeichert. Das Feld `reg exp group nr` gibt dabei die im regulären Ausdruck zu selektierende Gruppe an. Zudem wird ein Name gespeichert, der aber nur für die Verwaltungsapplikation benötigt wird, damit der Benutzer bei einer späteren Wiederverwendung des Ausdrucks diesen schneller auffinden kann.

Die beiden größeren Einheiten, Precision- und Raw-Pattern werden unter `precision patterns` bzw. `raw patterns` gespeichert. Neben einer Id und einem Namen enthalten sie dabei Verweise auf Process-Expressions, welche mittels der jeweiligen Process-Expressions-Ids realisiert werden. In `precision patterns` ist für jede mögliche Process-Expression dabei ein Feld vorhanden. Welches davon später im System vom Algorithmus genutzt wird, entscheidet dabei der `pattern type`,

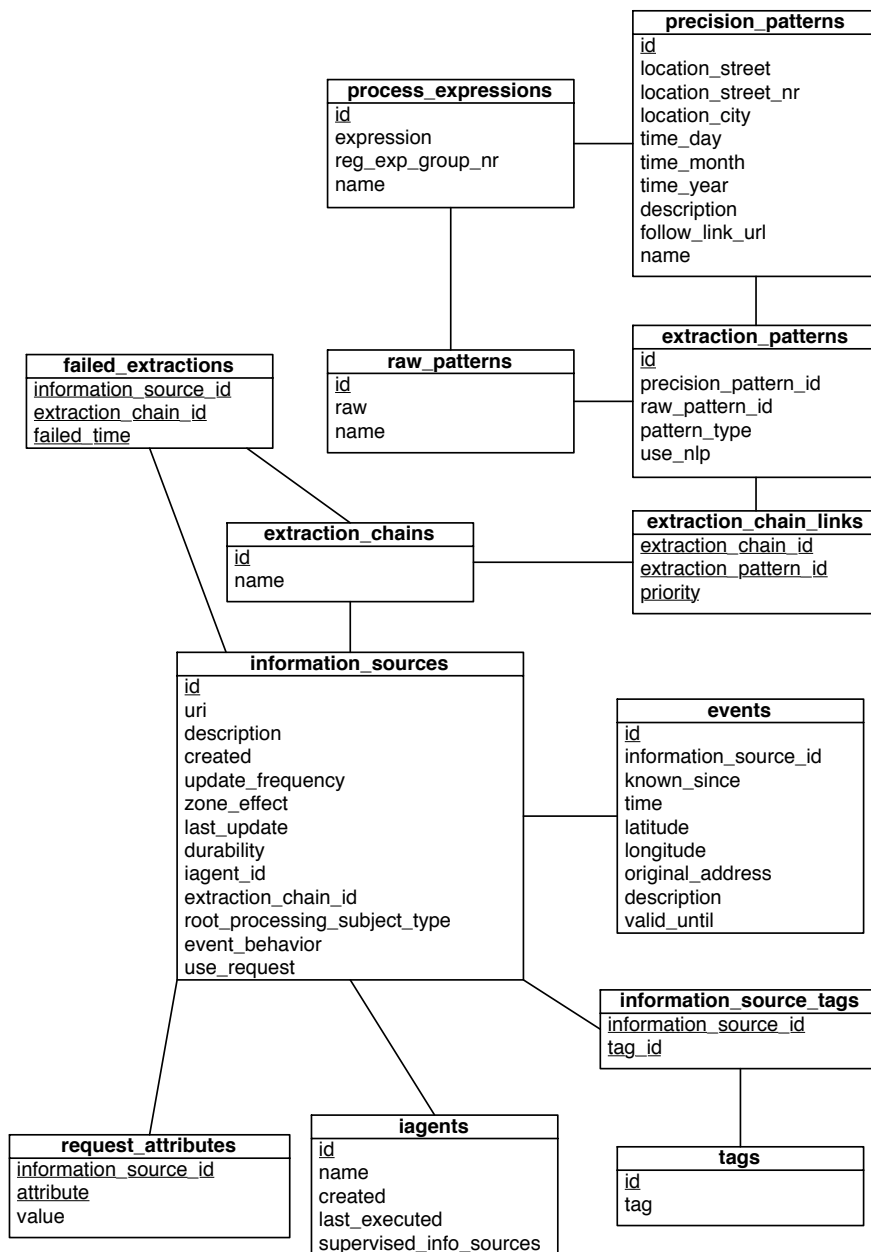


Abbildung 8.1: Datenbankmodell für das Repository

der in der Tabelle der Extraction-Pattern (`extraction patterns`) gespeichert ist. Diese Tabelle enthält zusätzlich zu diesem Typ zwei Verweise auf das jeweilige Raw- bzw. Precision-Pattern. Hierbei kann es auch vorkommen, dass nur ein Raw-Pattern vorhanden ist. Dieses ist beispielsweise bei Pattern vom Typ `group-relation` der Fall (siehe dazu Abschnitt 5.3.1). Ebenfalls die Angabe, ob die NLP-Technik zum Einsatz kommen soll, ist in `extraction patterns` durch `use nlp` vorhanden (z.B. „T“ falls NLP eingesetzt werden soll).

Zum Schluss existieren noch die Tabellen `extraction chain links` und `extraction chains`. Diese sind für die Speicherung der Extraction-Chains verantwortlich. `Extraction chains` enthält dabei die Id und den Namen der Verkettung. Der Name ist dabei wieder nur für die Managementapplikation interessant, um den Benutzer eine Hilfe zu geben.

Die einzelnen „Kettenglieder“, welche letztendlich Extraction-Pattern sind, werden in `extraction chain links` abgespeichert. Dabei wird für jedes Pattern noch eine Priorität mittels `priority` festgelegt, die später über die Reihenfolge der Ausführung entscheidet. Je höher die Priorität, desto früher wird das Extraction-Pattern auf der Quelle ausgeführt.

8.3 Informationsquellen

Nach den Extraktionsregeln ist die Speicherung der Informationsquellen die zweite große logische Einheit. Dabei wird in diesem Bereich auch die Speicherung der Agenten, eventueller Formularanfragen, fehlgeschlagener Extraktionen und der Tags miteinbezogen, da diese sinngemäß nahe beieinander liegen.

Die Stammdaten einer Informationsquelle werden in der Tabelle `information sources` gespeichert. Hierzu zählen u.a. neben einer Beschreibung und der URL,

- der Rhythmus in dem die Extraktion durchgeführt wird (`update frequency`),
- der Effekt den extrahierte Ereignisse auf den Sicherheitslevel der Zonen haben (`zone effect`),
- die Haltbarkeitsdauer von Ereignissen (`durability`) oder
- das Verhalten nach der Extraktion (`event behavior`), welches angibt, ob alte Ereignisse gelöscht werden soll usw.

Zusätzlich hierzu wird auch die Id der Extraction-Chain gespeichert, die mittels des Wrappers auf die jeweilige Quelle angewendet werden soll (`extraction chain id`).

Welche Art von Processing-Subject beim ersten Laden genutzt werden soll, wird durch die Angabe in `root processing subject type` definiert.

Falls das Laden dieser ersten Quellenseite nur mit Hilfe einer Formularanfrage erfolgen kann, werden zunächst einmal in der Tabelle `request attributes` die Parameter und die passenden Werte gespeichert, die für den erfolgreichen Aufruf des Formulars notwendig sind. Ob diese „Request-Daten“ auch später vom System genutzt werden, wird über einen booleschen Wert im Feld `use request` der Tabelle `information sources` definiert.

Für die Zuweisung eines Agenten zur Quelle dient das Feld `iagent id`, welches auf einen Eintrag in `iagents` verweist. Hier werden Name (`name`), letzter Ausführungszeitpunkt (`last executed`) und die Anzahl der vom Agenten überwachten Quellen gespeichert (`supervised info sources`).

Bei der Erstellung einer Informationsquelle können, wie in 7.4 beschrieben wurde, zusätzliche Metadaten in Form von Schlüsselwörter (Tags) der Quelle zugewiesen werden. Die einzelnen Wörter werden dabei in `tags` gespeichert. Die Zuweisung zur Quelle erfolgt über die Tabelle `information source tags`, so dass eine Quelle beliebig viele Tags erhalten kann.

Die Speicherung der Tags in einer eigenen Tabelle erlaubt es zum einen, dass Schlüsselwörter für mehrere Quellen verwendet werden können und dass Änderungen an diesen Tags sich auf alle Quellen auswirken, die dieses Tag nutzen.

In 6.4.3 wurde beschrieben, welche Daten im Falle einer fehlerhaften Extraktion protokolliert werden sollen. Zum einen war das die Id der Informationsquelle, die Id der verwendeten Extraction-Chain und der aktuelle Zeitstempel. All diese Daten werden in der Tabelle `failed extractions` gespeichert.

8.4 Ereignisse

Vom Wrapper extrahierte und geokodierte Ereignisse stellen das Endprodukt der Informationsextraktion dar und werden in der Tabelle `events` gespeichert. Dabei ist bei jedem Ereignis die Quelle angegeben, aus der es stammt (`information source id`). Zusätzlich dazu wird der Zeitpunkt gespeichert an dem das Ereignis extrahiert wurde (`known since`).

Die eigentlichen Informationen, die zur Ereignisgruppe gehören (siehe 3.3), sind in den Feldern

- `time` (gibt den Zeitpunkt des Ereignisses an)
- `latitude` und `longitude` (geben Breiten- bzw. Längengrad des Ortes an)
- `original address` (enthält die vom Wrapper extrahierte Adresse)
- `description` (Beschreibung des Ereignisses)
- `valid until` (Zeitpunkt bis zu dem das Ereignis gültig ist)

gespeichert. Die `original address` wird dabei nur zur Sicherheit gespeichert, da die Adresse natürlich auch anhand der Geokoordinaten bestimmt werden kann.

8.5 Aktualisierte Systemstruktur

Mit dem Repository ist nun die letzte Komponente im Gesamtsystem entworfen worden. Schaubild 8.2 zeigt nun die aktualisierte Version, wobei nun die relationale Datenbank hinzugefügt wurde.

Dabei ist diese in einem eigenen Datenbankserver integriert worden, um die mögliche Unabhängigkeit von den anderen Servern darzustellen. Natürlich kann das Repository auch auf dem Webserver oder dem Agentenserver installiert werden, wie schon am Anfang des Kapitels erwähnt wurde.

Zudem ist die Verbindung zwischen track-u System und Repository eingefügt worden, da die Eskalationslogik von track-u Zugriff auf die `events` haben muss.

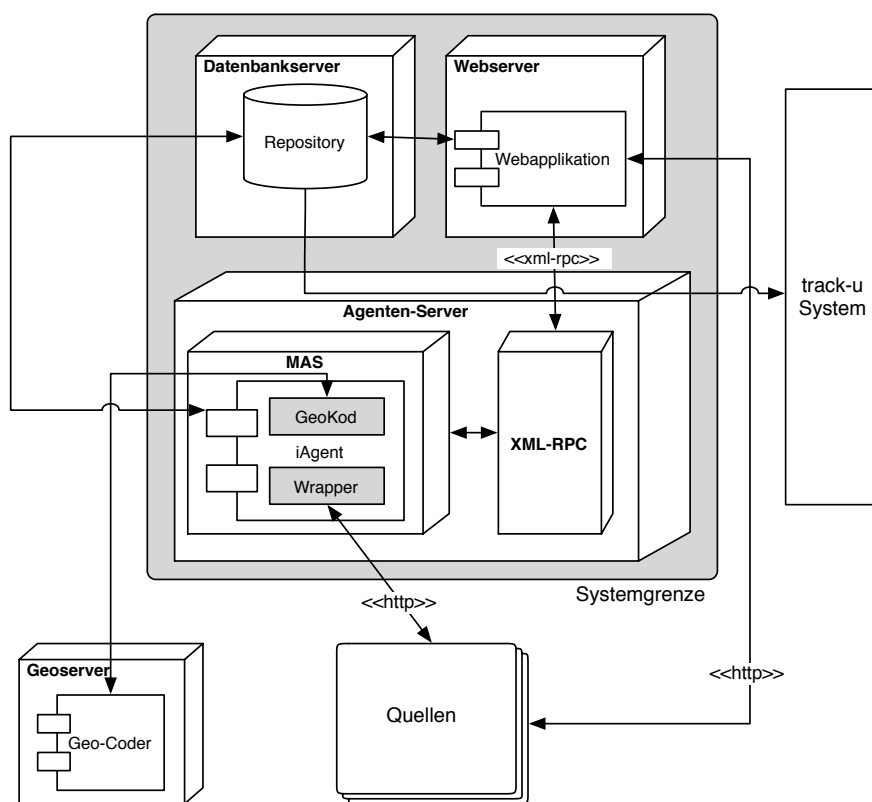


Abbildung 8.2: Komplette Systemstruktur nach Repository-Entwurf

9

Systemumsetzung

Die vorherige Arbeit hatte zum Ziel, theoretisch in die unterschiedlichen Bereiche des Systems einzuführen und die einzelnen Teilkomponenten vorzustellen bzw. zu entwerfen. Hierbei wurden zwar Architektur und Vorgehensweisen erläutert, aber es wurde nicht auf die wirkliche Umsetzung dieser Komponenten eingegangen.

Das vorliegende Kapitel soll nun diesen Umsetzungsschritt vollziehen und die Implementierung des Systems vorstellen. Dazu werden am Anfang einige der Technologien kurz beschrieben, die in der Umsetzung Anwendung finden werden. Das restliche Kapitel widmet sich dann den einzelnen Schnittstellen, Klassenstrukturen und beschreibt im einzelnen die Abläufe.

Grundsätzlich muss bei diesem Kapitel beachtet werden, dass aufgrund des Umfangs nicht alle in den Entwurfskapiteln besprochenen Funktionen auch wirklich in dem System im Rahmen dieser Thesis umgesetzt werden. Die Implementierung hat daher eher einen prototypischen Charakter und muss als ein „Proof-Of-Concept“ angesehen werden.

9.1 Verwendete Technologien

Bei dem System handelt es sich um ein verteiltes System, das aus unterschiedlichen Komponenten besteht. Die Bestandteile sind zum einen das Multiagentensystem (MAS), das auf einem eigenen Server ausgeführt wird. Hinzu kommt auf dem gleichen Server noch eine XML-RPC-Komponente für die Anfragen von der Verwaltungsapplikation. Diese Applikation hingegen wird auf einem Webserver ausgeführt.

Das MAS wird dabei unter Verwendung des *Java Agent Development Frameworks* (JADE) umgesetzt, die Implementierung erfolgt dementsprechend durch die Programmiersprache Java. Java wird ebenfalls bei der Umsetzung des XML-RPC-Servers verwendet. Hier wird der frei verfügbare Server *Apache XML-RPC* eingesetzt.

Die Realisierung der Verwaltungsapplikation wird, wie bei dem Entwurf schon festgelegt wurde, als Webapplikation erfolgen. Zur Umsetzung wird hierbei das MVC-Framework *Ruby on Rails* verwendet. Im folgenden werden diese Haupttechnologien kurz erläutert, um einen besseren Überblick über das spätere Gesamtsystem zu ermöglichen.

9.1.1 JADE Agentenframework

JADE¹ ist eine Middleware, mit der sich FIPA-konforme Multiagentensysteme in der Programmiersprache Java realisieren lassen. JADE besteht dabei aus einer Reihe von Klassen, die zur Entwicklung

¹<http://jade.tilab.com/>

benötigt werden und einer Laufzeitumgebung, um die Agenten letztendlich auf dem Rechner ausführen zu können [BCPR03].

Zudem bietet JADE die Möglichkeit, das MAS über mehrere Host-Rechner zu verteilen, wobei auf jedem Rechner eine JVM (*Java Virtual Machine*) laufen muss, um die Agenten in so genannten *Containern* aufnehmen zu können. Auf genau einem Host läuft dabei neben einer RMI-Registry, die intern von JADE benötigt wird, der *Main-Container*, der das Agent Management System (AMS) bzw. den Directory Facilitator (DF) ausführt (siehe dazu 6.1.2). Alle anderen Agenten-Container sind mit diesem Main-Container verbunden und bilden zusammen die gesamte Agenten-Umgebung. Abbildung 9.1 zeigt eine Beispiel-Verteilung über drei Host-Rechner.

Neben diesem Grundaufbau bietet JADE einige Methoden an, die bei der Entwicklung der Agenten genutzt werden können. Eine dieser Möglichkeiten soll im folgenden kurz vorgestellt werden. Für weitere Informationen zu den speziellen Eigenschaften der JADE-Plattform wird auf [BCTR06] verwiesen.

Behaviour

Eine für diese Arbeit besonders interessante Eigenschaft der JADE-Agenten, die auch später im System eingesetzt wird, sind so genannte *Behaviours*, die durch die *Behaviour*-Klasse bzw. deren Unterklassen umgesetzt werden. Jede Aufgabe, die ein Agent erfüllen soll, kann einem *Behaviour* zugewiesen werden und wird dann während der Laufzeit abhängig von der verwendeten *Behaviour*-Klasse ausgeführt.

Eine dieser „Verhaltens-Klassen“ ist das *TickerBehaviour*, welches in den Informationsagenten genutzt wird und es erlaubt, dass eine Aufgabe in regelmässigen Zeitabständen ausgeführt wird.

9.1.2 XML-RPC

Wie in den Entwurfskapiteln zum Multiagentensystem und zur Verwaltungsapplikation festgelegt wurde, erfolgt die Kommunikation zwischen diesen beiden Komponenten mittels XML-RPC-Aufrufen. Im folgenden wird über XML-RPC im allgemeinen und über den bei der Umsetzung auf Seiten des MAS verwendeten *Apache XML-RPC* Server ein kurzer Überblick gegeben.

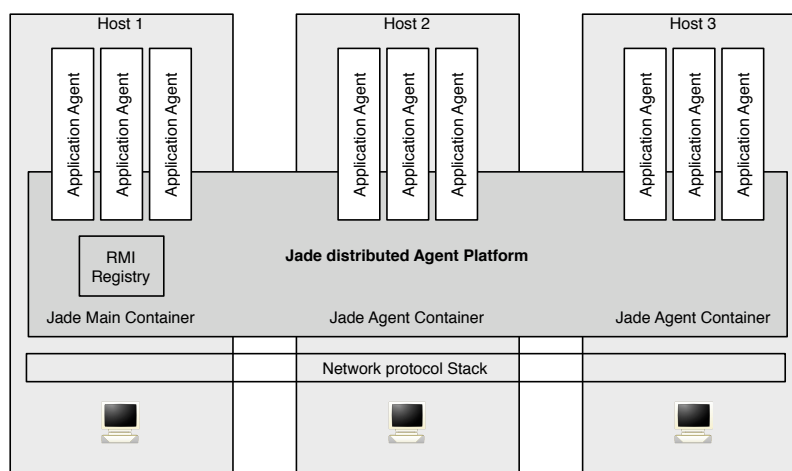


Abbildung 9.1: Ein über mehrere Hosts verteiltes Agentensystem mit JADE [BCTR06]

Grundlagen

XML-RPC ist ein auf XML basierendes Protokoll für *Remote Procedure Calls* (RPC). Es werden dabei über HTTP (*Hypertext Transfer Protocol*) Nachrichten zwischen Rechnern über ein Netzwerk ausgetauscht, um entfernte Funktionsaufrufe zu ermöglichen. Die Rumpfe (*body*) einer XML-RPC Anfragenachricht sowie der Antwortnachricht sind dabei durch ein XML-Format realisiert [Win99].

Im Gegensatz zu anderen Protokollen wie SOAP, welches ebenfalls ein XML-Format ist, ist XML-RPC ziemlich einfach gehalten, was den Vorteil bei diesem Protokoll ausmacht. So zeigt Beispiel 9.1 den Inhalt einer XML-RPC-Anfrage und 9.2 einer passenden XML-RPC-Antwort.

```
<?xml version="1.0" ?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
    <param>
      <value><i4>41</i4</value>
    </param>
  </params>
</methodCall>
```

Beispiel 9.1: Body einer XML-RPC-Anfrage [Win99]

```
<?xml version="1.0" ?>
<methodResponse>
  <params>
    <param>
      <value><string>South Dakota</string</value>
    </param>
  </params>
</methodResponse>
```

Beispiel 9.2: Body einer XML-RPC-Antwort [Win99]

XML-RPC unterstützt als Parameter für die Funktionen dabei unterschiedliche Datentypen wie beispielsweise Strings, Daten oder Zahlen.

Apache XML-RPC

Da es sich bei XML-RPC nur um ein Protokoll handelt, bedarf es einer Implementierung, um dieses Protokoll auch nutzen zu können. *Apache XML-RPC*² ist solch eine Implementierung in der Sprache Java und wird in dieser Thesis auf Seiten des Agenten-Servers eingesetzt.

Dabei wird nur die server-seitige Komponente von der Apache XML-RPC Implementierung verwendet, da in dieser Arbeit auf Seiten des Agentenservers Anfragen nur entgegen genommen werden müssen. Der XML-RPC Client wird innerhalb der Webapplikation ein- und durch Ruby umgesetzt.

Um nun einen XML-RPC Server zu starten, reicht es, auf Seiten des Agentenservers folgendes auszuführen:

```
WebServer server = new WebServer(8080);
server.addHandler("$default", new XmlRpcServer());
server.start();
```

Beispiel 9.3: Apache XML-RPC Server auf Port 8080 starten

²<http://ws.apache.org/xmlrpc/>

Hierbei wird der Server auf dem Port 8080 gestartet und nimmt dort die Methoden-Aufrufe entgegen, die dann in der Klasse `XmlRpcServer` aufgerufen werden.

Detaillierter soll an dieser Stelle auch nicht auf Apache XML-RPC eingegangen werden, da dieses für die weitere Arbeit nicht von Bedeutung ist und andere Quellen wie [Fou06] hierzu einen viel besseren Überblick liefern können.

9.1.3 Ruby on Rails Web-Framework

*Ruby on Rails*³ (RoR) ist ein seit dem Jahr 2004 existierendes MVC-Framework (*Model-View-Controller*), welches zur Entwicklung von Webapplikationen genutzt wird und auch in dieser Arbeit auf Seiten der Verwaltungsapplikation zum Einsatz kommt. Als Programmiersprache wird hierbei die objektorientierte Sprache *Ruby*⁴ verwendet.

Im folgenden werden die Teile von RoR vorgestellt, die auch in der Arbeit genutzt werden. Dabei wird sich bei der Beschreibung vor allem an den Ausführungen in [Tho05] und [TH06] orientiert.

Active Record Models

Das Modell, als der erste Bestandteil des MVC-Patterns, wird in RoR durch Klassen umgesetzt, die vom einem so genannten `ActiveRecord` abgeleitet werden, genauer gesagt `ActiveRecord::Base`. Ein Active Record ermöglicht dabei eine Abbildung – ein Mapping – des Modells auf eine relationale Datenbank, übernimmt also ein *Object-Relational Mapping* (ORM). Dabei werden Klassen auf Tabellen, Objekte auf Tabellenzeilen und Objekt-Attribute auf Spalten abgebildet.

Jeder Modell-Klasse stehen durch das `ActiveRecord` zusätzlich noch zahlreiche Methoden zur Verfügung, die dabei direkt auf den Daten in der Datenbank ausgeführt werden können.

Das bedeutet für die spätere Implementierung, dass eine in Ruby on Rails angelegte Klasse die z.B. `Event` heißt und vom `ActiveRecord` abgeleitet wurde, direkt auf eine Tabelle `events` in der Datenbank abgebildet wird⁵. Wird nun beispielsweise die Methode `Event.find(1)` ausgeführt, geht automatisch eine Anfrage an die Datenbank-Tabelle, die ein Ereignis mit der Id 1 lädt (z.B. `select * from events where id = 1`).

Der Vorteil ist hier also, dass sich beim Anlegen der Modell-Klassen keine Gedanken gemacht werden muss, wie der Zugriff auf die Datenbank aussehen soll. Es ist ausreichend, wenn die Tabellen und Klassen den Namenskonventionen entsprechen, da das Framework die restliche Arbeit übernimmt.

Controllers und Views

Neben dem Modell stellen Controller und Views die beiden anderen Bestandteile des MVC-Patterns dar. Ein Controller ist in RoR dabei eine Klasse, die von `ApplicationController` abgeleitet wird. Alle Methoden die auf diesem Controller ausgeführt werden können – die so genannten *Actions* – befinden sich in dieser Klasse und werden durch das Schlüsselwort `def` definiert.

Der Name des Controllers und der verwendeten Methode bzw. Action definieren dabei auch, welche Templates standardmässig zur Präsentation genutzt werden⁶. Diese Templates bilden dabei den View im MVC-Pattern und können entweder HTML erzeugen (RHTML-Templates) oder XML (RXML-Templates).

³<http://www.rubyonrails.org/>

⁴<http://www.ruby-lang.org>

⁵Die Konventionen von Ruby on Rails setzen voraus, dass der Name der jeweiligen Tabelle der Plural des Klassennamens ist

⁶Es ist auch möglich gezielt andere Templates auszuwählen

Nachdem die eingesetzten Technologien in den Grundzügen beschrieben worden sind, wird im weiteren auf die Umsetzung der einzelnen Systembestandteile eingegangen.

9.2 IE-Kernkomponente

In Kapitel 5 wurde der Wrapper bzw. die IE-Komponente entworfen und im Rahmen dessen der Extraktions-Algorithmus entwickelt. Zudem wurden die unterschiedlichen Datenstrukturen vorgestellt, die in der Umsetzung Anwendung finden werden.

Der folgende Abschnitt geht nun auf diese Implementierung des Wrappers ein, wobei hier ebenfalls die Programmiersprache Java verwendet wird, da die Extraktionskomponente innerhalb der Software-Agenten ausgeführt wird.

In Abbildung 9.2 ist die Klassenstruktur der IE-Komponente dargestellt und im folgenden wird anhand dieser Klassen die Beschreibung der Komponente vorgenommen. Dabei wird zwischen den Klassen unterschieden, die nur die Datenstrukturen repräsentieren und Klassen die ausführende Tätigkeiten übernehmen.

9.2.1 Datenobjekte

Die Datenobjekte enthalten keine wirkliche Applikationslogik, sondern dienen zur Speicherung von Anfragen, Ergebnissen und anderen Strukturen. Sie entsprechen dabei eher dem Konzept einer *Java Bean*, können aber teilweise über Methoden zur Aufbereitung der in ihnen gespeicherten Daten verfügen.

ProcessingSubject Ein Processing-Subject stellt im Wrapper das Objekt dar, auf dem Extraktionsschritte ausgeführt werden können (siehe 5.2). In der Implementierung wird es dabei durch die gleichnamige Klasse `ProcessingSubject` repräsentiert. In dieser wird entweder ein DOM-Objekt oder eine Zeichenkette gespeichert.

ProcessExpression Die `ProcessExpression`-Klasse speichert einen Ausdruck, der später von einem Prozessor für reguläre bzw. XPath-Ausdrücke verarbeitet werden kann. Bei regulären Ausdrücken wird dabei neben dem eigentlichen Ausdruck noch ein Index gespeichert, der später eine bestimmte Gruppe in dem regulären Ausdruck selektiert.

Ob es sich bei der `ProcessExpression` um einen XPath-Ausdruck bzw. regulären Ausdruck handelt, entscheidet dabei das Setzen des Indexes für den regulären Ausdruck. Das heißt, dass beim Gebrauch von regulären Ausdrücken immer mit Gruppen gearbeitet werden muss.

PrecisionPattern, RawPattern `Raw`- und `PrecisionPattern` nutzen die eben vorgestellten `ProcessExpressions`, um die einzelnen Ausdrücke für die jeweiligen Informationen zu speichern. Das `RawPattern` enthält dabei nur einen Ausdruck zur Selektion der späteren Blöcke (*raw*), während das `PrecisionPattern` hingegen über insgesamt 8 `ProcessExpressions` verfügt, die später, abhängig vom Extraktionstyp, genutzt werden können (`locationStreet`, `locationStreetNr`, `locationCity` usw.)

ExtractionPattern Das `ExtractionPattern` ist die Haupteinheit, die später zur Extraktion genutzt wird (siehe 5.3.1). In jedem `ExtractionPattern` wird dabei ein `RawPattern`- und ein `PrecisionPattern`-Objekt gespeichert. Zusätzlich wird noch der Typ des Extraction-Patterns gespeichert, der später bei der Verarbeitung für Entscheidungen im Algorithmus genutzt wird und Werte wie beispielsweise *location*, *time*, *description* oder *group-relation* annehmen kann.

Die Angabe, ob auf der resultierenden Rohinformation noch eine NLP-Methode angewendet werden soll, wird durch den booleschen Wert `useNLP` ebenfalls im `Extraction-Pattern` definiert.

ExtractionChain Die `ExtractionChain` stellt die Verkettung von `Extraction-Pattern` dar und enthält dementsprechend eine Liste von `ExtracionPattern`-Objekten. Zusätzlich ist die Kette über `isLastChainlinkFollowLink` in der Lage zu bestimmen, ob das letzte `Extraction-Pattern` vom Typ `follow-link` ist. Diese Überprüfung wird vom Extraktions-Algorithmus genutzt, um Multi-Pages verarbeiten zu können (siehe dazu Seite 46).

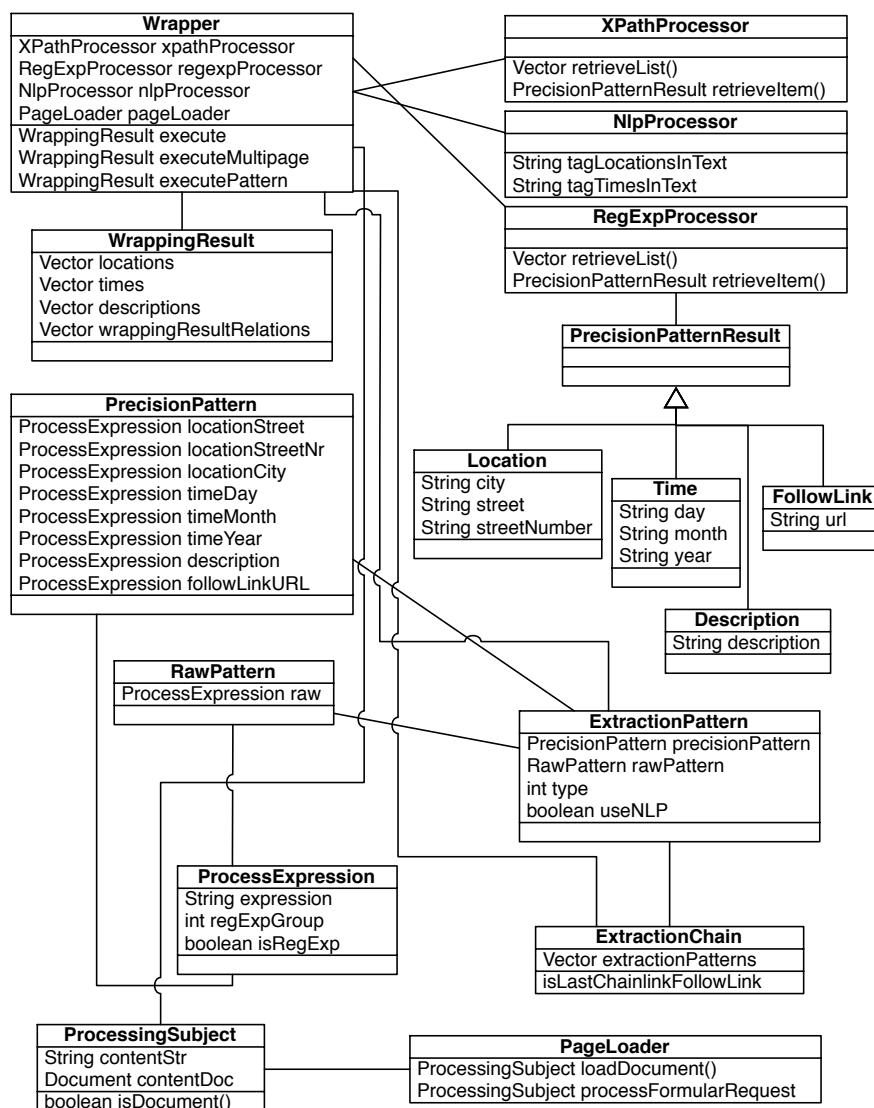


Abbildung 9.2: Klassen für die Wrapper-Umsetzung

PrecisionPatternResult, Time, Location, Description, FollowLink Diese Datenobjekte stellen das Ergebnis dar, welches durch die Anwendung eines Precision-Patterns entsteht. Dabei stellt `PrecisionPatternResult` die Oberklasse der Informations-Klassen `Location`, `Time`, `Description` und `FollowLink` dar.

Jede dieser Unterklasse enthält Zeichenketten-Variablen, in denen bei der Extraktion die relevanten Informationen gespeichert werden. Bei `Time` beispielsweise `day`, `month`, `year`, bei `Location` dementsprechend `city`, `street` und `streetNumber`.

`PrecisionPatternResult`-Objekte werden dabei vom `RegExpProcessor` als Ergebnis zurückgeliefert, wie im Abschnitt über die ausführenden Komponenten noch dargestellt wird.

WrappingResult Objekte vom Typ `WrappingResult` stellen die komplexeste Datenstruktur in der IE-Komponente dar und werden vom `Wrapper` als Ergebnis einer Extraktion zurückgeliefert. In ihr werden die Ergebnisse der Extraktion gespeichert, also die Ereignisgruppen. Dabei existieren zum einen Listen für die gefundenen Orte (`locations`), Zeitpunkte (`times`) und Beschreibungen (`descriptions`).

Zusätzlich dazu werden bei erkennbaren Gruppen-Relationen (siehe auch 3.4.2) diese in einer weiteren Liste gespeichert (`wrappingResultRelations`). Eine Gruppen-Relation ist dabei wiederum ein `WrappingResult`. Das bedeutet, dass Elemente der `wrappingResultRelations`-Liste `WrappingResults` sind.

Die Methode `getUngeocodedEvents`, die ebenfalls Bestandteil vom `WrappingResult` ist, liefert eine Liste von `UngeocodedEvents` zurück. Dabei ist ein `UngeocodedEvent`, wie noch im Folgeabschnitt 9.3 gezeigt werden wird, die Kombination aus `Location`, `Time` und `Description`.

`GetUngeocodedEvents` stellt dabei intern die Ereignisse aus den `locations`-, `times`- und `descriptions`-Listen zusammen. Dazu wird überprüft, ob die drei Listen die gleiche Länge haben. Falls ja, können sie zusammengefasst werden. Falls nur die `times`-Liste sich in der Länge unterscheidet, wird diese Liste verworfen und als `Time` die aktuelle Zeit genommen (Erinnerung: Es ist ja keine eindeutige Zuordnung der Zeit möglich, falls die Listen nicht gleich lang sind).

Der Fall bei dem das `WrappingResult` Gruppen-Relationen enthält, wird zunächst überprüft, ob auch neben `wrappingResultRelations` die `locations`-, `times`- und `descriptions`-Listen über Elemente verfügen. Falls dieses nicht der Fall ist, wird durch `wrappingResultRelations` iteriert und die Ereignisse aus den darin enthaltenen `WrappingResults` gebildet. Falls jedoch Elemente in den anderen Listen vorkommen, wird wiederum anhand der Länge überprüft, ob ein Merging vorgenommen werden kann (eine gleiche Länge lässt dieses zu).

9.2.2 Ausführende Komponenten

In den ausführenden Komponenten sind die Algorithmen umgesetzt, die zur Extraktion im `Wrapper` benutzt werden und im Entwurf entwickelt worden sind. Dabei arbeiten diese Komponenten auf Basis der Datenobjekte, die im letzten Abschnitt präsentiert wurden.

PageLoader Die Aufgabe des Ladens von Textdokumenten (Webseiten, Newsfeeds) wird vom `PageLoader` übernommen. Der `PageLoader` wird dabei nicht nur beim anfänglichen Aufrufen der Quellen-Startseite genutzt, sondern auch bei der Navigation mit Hilfe der `follow-links`.

Das Laden erfolgt über die Funktion `loadDocument`, welche neben der URL der Seite einen zusätzlichen Parameter erwartete, der den Typ der Quelle definiert (XML-Dokument oder nicht-valides Dokument).

Die Funktion verwendet dabei intern, abhängig vom Typ, entweder einen `javax.xml.parsers.DocumentBuilder` oder `java.io.InputStreamReader` zum Laden und liefert ein Objekt vom Typ `ProcessingSubject` zurück, welches als Eingabe für weitere Verarbeitungen dient.

RegExpProcessor, XPathProcessor `RegExpProcessor` und `XPathProcessor` sind Prozessoren, die reguläre Ausdrücke (RA) bzw. XPath-Ausdrücke auf Daten ausführen können. Intern wird dazu zur Verarbeitung von XPath *Xalan-Java*⁷ verwendet und für reguläre Ausdrücke, der von Java zur Verfügung gestellte `java.util.regex.Matcher`. Zur Verarbeitung von XPath müssen die Daten dabei als `org.w3c.dom.Document-Model` vorliegen. Der RA-Prozessor arbeitet dagegen auf normalen Zeichenketten.

Beide Klassen, `RegExpProcessor` sowie `XPathProcessor`, verfügen über eine `retrieveList`-Methode, die das `RawPattern` eines `ExtractionPattern` entweder auf einem `Document` ausführt oder auf einer Zeichenkette.

`RetrieveList` liefert danach eine Liste von Zeichenketten zurück, wobei die Zeichenketten bzw. Strings die späteren Rohinformationen repräsentieren.

Auch der XPath-Prozessor liefert Zeichenketten zurück, obwohl er natürlich auch Objekte wie beispielsweise vom Typ `org.w3c.dom.Node` liefern könnte. Da aber auf den Rohinformationen in dieser Arbeit `PrecisionPattern` als reguläre Ausdrücke ausgeführt werden, sind auch in diesem Fall Zeichenketten als Rückgabewerte gewählt worden.

Die Ausführung der `PrecisionPattern` übernimmt die Methode `retrieveItem`, die als Parameter eine Zeichenkette und ein `ExtractionPattern` erwartet. Diese Methode ist nur im `RegExpProcessor` implementiert, da – wie die Quellenanalyse ergeben hat – die Feininformationen mittels XPath nicht extrahierbar sind (dafür müssten sie in einer strukturierteren Form vorliegen). Natürlich kann der `XPathProcessor` bei späteren Systemerweiterungen noch um eine `retrieveItem`-Methode ergänzt werden, falls Quellen dieses dennoch erfordern sollten.

`RetrieveItem` liefert als Rückgabewert ein Objekt vom Typ `PrecisionPatternResult` zurück, genauer gesagt ein Objekt der Unterklassen von `PrecisionPatternResult`. Welche Klasse dabei genutzt wird, entscheidet sich innerhalb von `retrieveItem` anhand des `ExtractionPattern`-Typs.

NlpProcessor Der `NlpProcessor` enthält die beiden Methoden `tagLocationsInText` und `tagTimesInText`, welche Ortsbezeichnungen bzw. Zeitangaben im Text durch Anwendung eines Taggers speziell markieren können, damit diese später mit Hilfe eines `Precision`-Patterns selektiert werden können, wie in Abschnitt 5.3.3 beschrieben wurde⁸.

Wrapper Von den drei ausführenden Komponenten ist der `Wrapper` die komplexeste. Er hat die Aufgabe, auf Basis eines `ProcessingSubjects` eine `ExtractionChain` auszuführen und ein `WrappingResult` als Rückgabe zu liefern. Dabei nutzt er intern die drei Prozessoren `RegExpProcessor`, `XPathProcessor` und `NlpProcessor`.

Nach Außen stellt der `Wrapper` eine `execute`-Methode zur Verfügung, die später vom Informati-
onsagenten genutzt wird und die erste Seite der Quelle als `Processing-Subject` und eine `Extraction-Chain`

⁷<http://xml.apache.org/xalan-j/>

⁸Der `NlpProcessor` wurde in dieser Thesis nicht komplett implementiert, aber schon an den Stellen integriert an denen er eingesetzt wird.

erwartet.

Ob es sich bei der zu bearbeitenden Quelle um eine „Multi Page“-Quelle handelt, wird anhand des letzten Patterns in der Extraction-Chain entschieden (`isLastChainlinkFollowLink`). Falls dieses der Fall ist, wird `executeMultipage` aufgerufen⁹ im anderen Fall wird `executePattern` aufgerufen. Letztere erwartet als Parameter ein `ProcessingSubject` *ps*, eine `ExtractionChain` *ec* und ein `WrappingResult` *wr* und soll im folgenden erläutert werden.

Die `executePattern`-Methode überprüft direkt nach dem Aufruf zuerst einmal, ob die `ExtractionPattern`-Kette *ec* überhaupt Pattern enthält. Falls sie die Länge 0 hat, gibt die Methode das `WrappingResult` *wr* zurück, welches als Parameter übergeben wurde (dient zur Terminierung der späteren Rekursion). Bei vorhandenen Pattern wird das erste Pattern (*p*) von *ec* geholt und aus der Kette entfernt.

Nun wird aufgrund des Typs von *p* entschieden, wie weiter verfahren werden soll. Handelt es sich beispielsweise um ein `location`-Pattern und bei *ps* um ein `Document`, werden mittels des `XPath`-Prozessors die Rohinformationen extrahiert. Danach wird durch diese Liste von Rohinformationen iteriert und mittels der `retrieveItem`-Methode ein `Location`-Objekt extrahiert, welches dem `wrappingResult` hinzugefügt wird.

Falls *ps* eine Zeichenkette ist, wird anstelle des `XPath`-Prozessors die `RegExpProcessor`-Komponente genutzt, welche dann ebenfalls die Rohinformationen extrahiert.

An dieser Stelle kommt auch der `NlpProcessor` zum Einsatz. Dieser überprüft, bevor `retrieveItem` ausgeführt wird, ob das aktuelle `Extraction-Pattern` einen NLP-Ansatz verwenden soll (`useNLP`). Falls dieses der Fall ist, wird die Rohinformation durch den `NlpProcessor` bearbeitet. Die Methode, die dabei genutzt wird, ist abhängig vom aktuellen `Pattern`-Typ. Bei „`location`“ wird `tagLocationsInText` genutzt bei „`time`“ entsprechend `tagTimesInText`.

Nach diesem Schritt sind in den Rohinformationen die jeweiligen Angaben markiert und können in `retrieveItem` unter Verwendung des `Precision`-Patterns extrahiert werden.

Nachdem das `WrappingResult` *wr* um die extrahierten Informationen ergänzt wurde, wird *wr* als Wert nun der Rückgabewert eines erneuten Aufrufs von `executePattern` zugewiesen. Es findet also eine Rekursion statt. Als Parameter erhält `executePattern` dabei das aktuelle `ProcessingSubject` *ps* die Verkettung *ec* (Erinnerung: wurde am Anfang um ein Glied verkleinert) und das `WrappingResult` *wr*.

`Extraction-Pattern` vom Typ `time` oder `description` werden genauso behandelt, nur das dort `Time` bzw. `Description`-Objekte anstelle eines `Location`-Objekts genutzt werden.

Die komplizierteren Fälle sind dabei die, bei denen *p* den Typ `follow-link` oder `group-relation` hat, wie im folgenden gezeigt werden soll.

Im Falle eines `follow-links` wird mit Hilfe des `Extraction-Pattern` eine Liste von Links extrahiert. Nun gelten diese Links ja nicht als die gesuchte Information und werden dementsprechend nicht *wr* hinzugefügt. Stattdessen wird mittels des `PageLoaders` die jeweilige, durch die `Link-URL` identifizierbare, Detailseite geladen. Das Laden resultiert wiederum in einem `ProcessingSubject` *ps'*. Auf jedem *ps'* muss nun die restliche `Extraction-Chain` ausgeführt werden – heißt es findet hier wieder eine Rekursion statt.

Da die Kette reduziert wird, muss ein neues `ExtractionChain`-Objekt erstellt werden, welches die geklonte Liste der `Pattern` von *ec* enthält.

Ein `Pattern` vom Typ `group-relation` hat zum Ziel, gefundene Informationen in einer Relation „zusammenzuhalten“, damit einfacher die Ereignisgruppen zusammengestellt werden können.

Bei einer `group-relation` werden nun zuerst einmal die Blöcke extrahiert, die durch das `Pattern` identifiziert werden. Auf diesen Blöcken werden die weiteren `Pattern` aus der Verkettung *ec* ausgeführt. Zudem

⁹Aus Zeitgründen konnte diese Methode leider nicht implementiert werden.

wird das `WrappingResult wr` gekennzeichnet, dass sich die im folgenden gefundenen Informationen auf eine Gruppierung beziehen. Jede Iteration durch die Rohinformationen des `group-relation`-Pattern und die Ausführung der `ec` bilden dabei eine Gruppierung und werden in einem `WrappingResult`-Objekt in der `wrappingResultRelations`-Liste gespeichert.

Wie beim `follow-link` muss hier ebenfalls die Kette geklont werden, da sonst die einzelnen Pattern verloren gehen würden.

Die eben beschriebenen Klassen liefern noch keine geokodierten Ereignisse zurück und speichern auch noch nichts persistent ab. Obwohl diese Aufgabe streng genommen ebenfalls zum `Wrapping`-Prozess gehört, wird sie jetzt übersichtshalber in einem eigenen Abschnitt vorgestellt.

9.3 Geokodierung und Ereignisverwaltung

In Abbildung 9.3 sind die Klassen dargestellt, welche für die Geokodierung und die Verwaltung von Ereignissen benötigt werden.

Die beiden Ereignisklassen `UngeocodedEvent` und `Event`, speichern die extrahierte Ereignisinformation ab. Dabei beinhaltet `UngeocodedEvent` die Informationen, wie sie der Wrapper zurückliefert und wie es in dem Abschnitt über das `WrappingResult` schon erwähnt wurde.

`Event` hingegen verfügt über die geokodierte Ortsangabe in Form eines `GeoCoordinates`-Objektes und beinhaltet zudem selbst wiederum das original `UngeocodedEvent`, um darüber auf die restlichen Informationen wie die Zeit und die Beschreibung zugreifen zu können.

Der Schritt von einem `UngeocodedEvent` zu einem `Event` wird über den `EventManager` und seine Methode `geocodeEvents` vollzogen. Hierbei wird durch eine übergebene Liste von `UngeocodedEvents` iteriert und mittels des `Geocoders` die Kodierung jeder einzelnen Adresse aus dem `UngeocodedEvent` vorgenommen.

Der `Geocoder` ist dabei ein Interface, welches nur über die Methode `getGeoCoordinatesForAddress` verfügt. Diese Methode erwartet eine Adresse und liefert

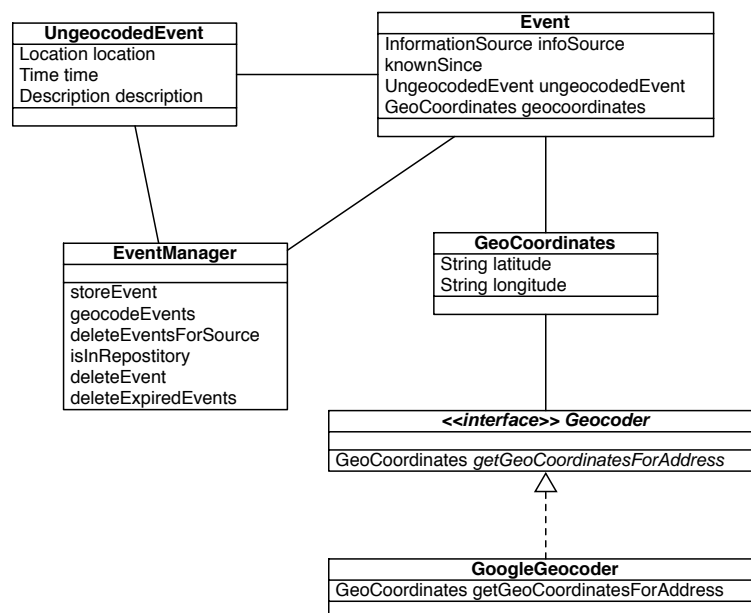


Abbildung 9.3: Klassen für die Geokodierung und das Ereignismanagement

ein `GeoCoordinates`-Objekt zurück, welches Längen- und den Breitengrade beinhaltet.

In dieser Arbeit wurde der Google-Geokodierungs-Service¹⁰ genutzt, weswegen die Schnittstelle durch eine Klasse `GoogleGeocoder` umgesetzt wurde. Dabei wird mittels eines URL-Aufrufs auf den Google-Geocoder zugegriffen, welcher wiederum die Kodierung als *Keyhole Markup Language* (KML) Dokument zurückliefert. Aus diesem XML-Dokument werden die beiden Geokoordinaten mit Hilfe eines XPath-Ausdrucks selektiert und dem `GeoCoordinates`-Objekt zugewiesen.

Zusätzlich zur Geokodierung auf den unkodierten Ereignissen, verfügt der `EventManager` noch über weitere Methoden, die später beim Informationsagenten eingesetzt werden, wie in Abschnitt 9.4 noch gezeigt werden wird.

Dazu zählt neben dem Speichern (`storeEvent`) und Löschen eines oder mehrerer Ereignisse (`deleteEventsForSource` bzw. `deleteEvent` oder `deleteExpiredEvents`) auch das Überprüfen mittels `isInRepository`, ob ein Ereignis im Repository schon vorhanden ist (Schlüssel aus Quelle-Id, Längen-, Breitengrad und Zeitpunkt).

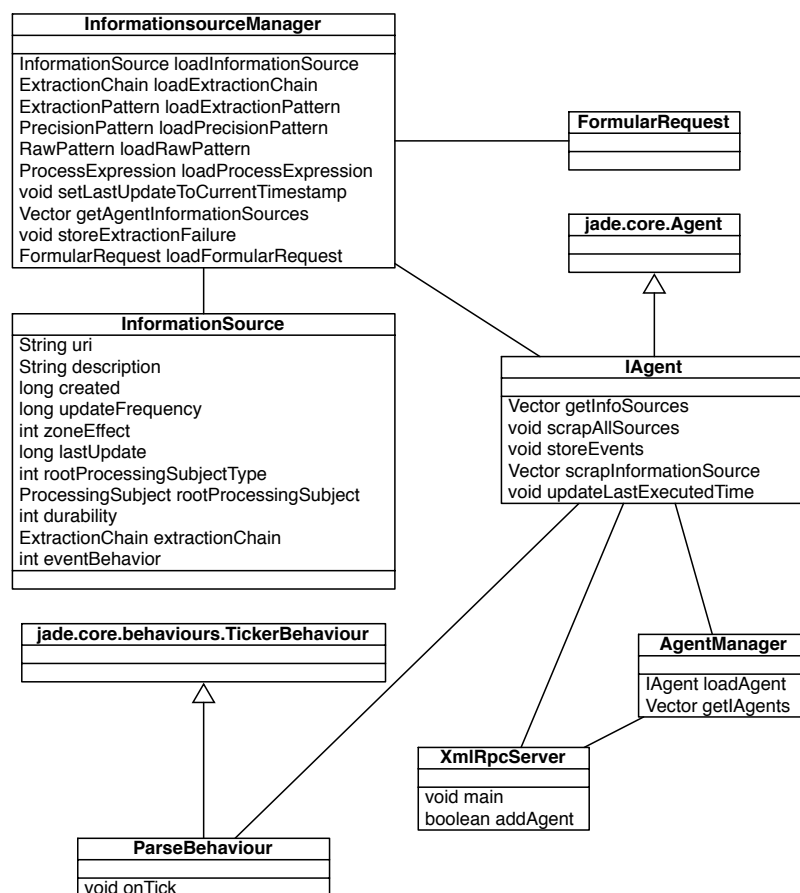


Abbildung 9.4: Klassen für die Agenten

9.4 Agenten, Quellen-Verwaltung und XML-RPC

Nachdem in den vorherigen Abschnitten die Informationsextraktion, Geokodierung und Ereignisverwaltung beschrieben worden ist, wird sich in diesem Abschnitt den Komponenten gewidmet, die für die Initiierung der Extraktion zuständig sind – den Software-Agenten.

Dabei übernehmen die Agenten auch die Aufgabe des Ladens der Pattern, die später im Wrapper genutzt werden. Aus diesem Grund werden auch die beiden Klassen `InformationSource` bzw. `InformationSourceManager` in diesem Abschnitt behandelt.

Zudem wird auch der XML-RPC-Server und seine Umsetzung dargestellt, da er direkt mit dem Agentensystem verbunden ist und für das Starten der Agenten verantwortlich ist.

In Abbildung 9.4 sind die Klassen im einzelnen aufgeführt.

9.4.1 Laden und Starten der Agenten

Wie eben erwähnt wurde, ist der XML-RPC-Server, genauer gesagt die Klasse `XmlRpcServer` für das Hinzufügen der Informationsagenten zur Agentenplattform zuständig. Dieses kann dabei bei zwei unterschiedlichen Ereignissen geschehen.

Zum einen, falls der Server einen XML-RPC Aufruf der Methode `addAgent` erhält. Dieser Aufruf findet von Seiten der Managementapplikation statt und erhält als einzigen Parameter die Id des Agenten. Anhand dieser Id wird über die Agenten-Managerklasse `AgentManager` der Agent mittels `loadAgent` geladen.

Die zweite Möglichkeit des Startens von Agenten findet statt, wenn der XML-RPC-Server gestartet wird. Hierbei werden alle in der Datenbank vorhandenen Agenten geladen. Der Grund hierfür ist der, dass der XML-RPC-Server die Aufgabe der Zustandswiederherstellung übernimmt (siehe 6.5.1). Nach Ausfall der Plattform können mit Hilfe eines Neustarts des XML-RPC-Servers alle Agenten wieder erzeugt werden.

Nachdem ein Agent als `IAgent`-Objekt geladen wurden, wobei ein `IAgent` direkt von Jades `jade.core.Agent` abgeleitet wird, muss er in einen `jade.wrapper.AgentContainer` geladen werden¹¹. Über `jade.core.Runtime` wird dabei mittels `createAgentContainer` ein neuer Container erstellt, in den durch Ausführung von `acceptNewAgent` der neue `IAgent` mit dem in der Datenbank vorhandenen Namen hinzugefügt wird.

An dieser Stelle befinden sich nun die Agenten in der Jade-Plattform und führen in regelmässigen Abständen den Wrapper auf den ihnen zugeteilten Informationsquellen aus. Dieser Prozess wird nun im darauf folgenden Abschnitt beschrieben.

9.4.2 Starten der Informationsextraktion

In 6.4.1 wurde beschrieben, wann die Extraktion vom Informationsagenten gestartet werden soll und welche Informationsquellen dabei einbezogen werden sollen. All diese Entscheidungen werden innerhalb des `IAgents` getroffen. Dazu erhält der Agent ein `ParseBehaviour` zugeordnet, welches eine Erweiterung von Jades `jade.core.behaviours.TickerBehaviour` ist und in regelmässigen Abständen die Methode `scrapAllSources` innerhalb des Agenten aufruft.

¹⁰Mehr Information dazu unter <http://www.google.com/apis/maps/documentation/>.

¹¹Es wird für jeden Agenten ein Container angelegt. Es ist natürlich auch möglich und sogar sinnvoller, dass auch hier eine Gleichverteilung von mehreren Agenten pro Container nach einem Muster vorgenommen wird. Dieses wurde in der Thesis aber nicht behandelt und bietet Möglichkeiten für zukünftige Erweiterungen.

Diese Methode lädt alle Informationsquellen, die der Agent überwachen soll. Dazu wird `getAgentInformationSources` über den `InformationSourceManager` ausgeführt, wobei eine Liste der Quellen zurückliefert wird, für die der jeweilige Agent zuständig ist. `getAgentInformationSources` nutzt dabei intern `loadInformationSource`, die wiederum überprüft, ob eine Quelle durch eine Formular-Anfrage geladen werden soll. Falls dieses Fall ist, werden über `loadFormularRequest` die notwendigen Parameter geladen und über `processFormularRequest` im `PageLoader` ausgeführt. Dieser liefert dann ein `ProcessingSubject` zurück, welches der Informationsquelle zugewiesen wird. Bei keiner erforderlichen Formularanfrage wird die Quelle standardmässig mittels `loadDocument` im `PageLoader` geladen.

Nachdem `getAgentInformationSources` die Liste von Informationsquellen geliefert hat, iteriert `ScrapAllSources` durch diese Liste und prüft, welche Quelle bereit zur Extraktion ist (falls der Zeitraum seit der letzten Extraktion größer oder gleich dem Aktualisierungs-Rhythmus ist). Dabei wird der Algorithmus aus Beispiel 6.1 verwendet.

Falls eine Quelle extrahiert werden kann, wird an dieser Stelle die Verbindung zum Wrapper hergestellt. Über diesen wird die `execute`-Methode aufgerufen und als Parameter das `ProcessingSubject` der Quelle und die `ExtractionChain` übergeben. Anhand der vom Wrapper zurückgelieferten Liste der `UngeocodedEvents` wird nun zuerst überprüft, ob ein möglicher Fehler vorliegt. Dieses wird anhand der Länge der Liste durchgeführt (siehe Seite 63). Bei einem Fehler werden durch den `InformationManager` unter Verwendung von `storeExtractionFailure` die Daten gespeichert, die für eine spätere Kontrolle durch den Benutzer notwendig sind (Quellen-Id, Extraction-Chain-Id, Zeitpunkt des Fehlers). Bei einer erfolgreichen Extraktion wird die nicht-leere Liste danach mit Hilfe des `EventManager` und der Methode `geocodeEvents` in eine Liste von geokodierten `Events` umgewandelt.

Im Agenten wird diese Liste in der Methode `storeEvents` weiterverarbeitet. Hier wird auch der Schritt des Löschens alter Ereignisse ausgeführt, wie er auf Seite 62 beschrieben und durch die Methode `deleteExpiredEvents` im `EventManager` realisiert wurde. Nach diesem Löschen wird anhand des Attributes `eventBehavior` der Klasse `InformationSource` überprüft, ob nach einer Extraktion alle Ereignisse aus dem Repository gelöscht werden sollen, die aus der aktuellen Quelle stammen. Bei Bedarf nimmt dieses der `EventManager` durch `deleteEventsForSource` vor.

Nach diesen Vorbereitungsschritten werden die extrahierten Ereignisse durchlaufen und geprüft, ob die geokodierten `Events` schon im Repository vorhanden sind. Durch die Methode `isInRepository` vom `EventManager` und unter Verwendung des eindeutigen Ereignis-Schlüssels wie er in Abschnitt 6.4.3 beschrieben wurde, wird diese Überprüfung vorgenommen und ggf. schon vorhandene Ereignisse gelöscht¹². Als letzter Schritt der `storeEvents`-Methode wird nun das neue Ereignis über den `EventManager` (`storeEvent`) in der Datenbank gespeichert.

9.5 Verwaltungssapplikation

In Kapitel 7 wurde die Managementapplikation entworfen und die Anforderungen definiert, die sie erfüllen muss. Eines der Ergebnisse war dabei, dass die Anwendung als Webapplikation umgesetzt werden soll, um größtmögliche Flexibilität zu bieten.

¹²Die in 6.4.3 ursprünglich beschriebene Methodik, bei der während der Extraktion die Überprüfung auf Dubletten vorgenommen wird, ist aufgrund Zeitmangels nicht umgesetzt worden. Stattdessen findet dieses nun – wie beschrieben – auf Seiten der Agenten statt.

Dieser Abschnitt widmet sich nun der Umsetzung der Webapplikation, wobei das in 9.1.3 vorgestellte Ruby on Rails Framework genutzt wird.

Es muss beachtet werden, dass die im folgenden beschriebene Implementierung nur die Grundanforderungen abdeckt und einen Prototypen zum Ziel hat. Es werden also nicht wirklich alle im Entwurf beschriebenen Funktionalitäten – vor allem im Bezug auf die Bedienoberfläche – umgesetzt.

In den Folgeabschnitten wird nun beschrieben, wie die Hauptfunktionalitäten

- Erstellen der Extraktionsregeln
- Hinzufügen einer neuen Informationsquelle zum System
- Überwachung bestehender Agenten, Quellen und Ereignisse

umgesetzt werden.

9.5.1 Erstellen der Extraktionsregeln

Für alle Funktionen, die mit der Erstellung von Regeln zu tun haben, ist der `PatternsController` zuständig. Hierbei ist für jede einzelne Komponente einer vollständigen Regel-Verkettung (`Process-Expression`, `Precision-Pattern`, `Raw-Pattern`, `Extraction-Pattern`, `Extraction-Chain`) eine eigene Action vorhanden.

Für die Erstellung einer `Process-Expression` übernimmt somit beispielsweise die Action `add new process expression` die Aufgabe der Speicherung in der Datenbank.

Der Controller verfügt insgesamt über die folgenden Aktionen zur Speicherung der Komponenten:

- `add new process expression`
- `add new precision pattern`
- `add new raw pattern`
- `add new extraction pattern`
- `add new extraction chain`

Zusätzlich dazu erlaubt `add pattern to chain` das Hinzufügen eines vorher erstellten `Extraction-Patterns` zu einer ebenfalls schon existierenden `Extraction-Chain` – sorgt also für die eigentliche Zusammenstellung der Verkettung.

Bei der Speicherung der Daten werden dabei einige Datenobjekte genutzt, die alle von der `ActiveRecord`-Klasse abgeleitet werden und somit eine einfache Anbindung an die relationale Datenbank – das `Repository` – ermöglichen.

Das Schreiben von Objekten vom Typ `ProcessExpression`, `PrecisionPattern`, `RawPattern`, `ExtractionPattern`, `ExtractionChainLink` und `ExtractionChain` läuft im `PatternsController` somit immer nach dem gleichen Muster ab.

Zuerst wird dabei ein neues der oben genannten Objekte mittels `new` erstellt, wobei mit Hilfe des `params`-Objekts die notwendigen Parameter für die Erstellung übergeben werden. Die Speicherung erfolgt danach durch den Aufruf von `save` auf dem jeweiligen Objekt.

Neben den Datenobjekten und dem Controller existieren noch einige RHTML-Seiten, die den View repräsentieren und Formulare zur Zusammenstellung der Regeln anbieten. Diese Zusammenstellung

erfolgt dabei über `select`-Listen. Jedes Datenobjekt, das solch eine Liste anbietet, verfügt aus diesem Grund über eine Methode `self.get all for selection list`, welche die in der Datenbank verfügbaren Objekte des jeweiligen Typs zusammenstellt (z.B. Liste von Precision-Pattern).

Zusätzlich zu den Listen stehen noch einige anderen Felder zur Verfügung (z.B. Name, Auswahl der Gruppe für reguläre Ausdrücke), die aber nicht näher betrachtet werden sollen. Stattdessen soll sich dem Hinzufügen neuer Informationsquellen zugewendet werden.

9.5.2 Hinzufügen neuer Informationsquellen

Das Erstellen der Extraktionsregeln, das im letzten Abschnitt beschrieben wurde, benötigt die meisten Einzelschritte und somit Aktionen im Controller.

Beim Hinzufügen einer neuen Informationsquelle hingegen, wird in dem dazugehörigen `SourcesController` nur eine Action benötigt, die eine Speicherung durchführt – `add new source`.

Hierbei werden Parameter wie Beschreibung, Zoneneinfluss, Gültigkeitsdauer aber auch die Id der passenden Extraction-Chain usw. wieder durch das `params`-Objekt geladen, einem neuen `InformationSource`-Objekt zugewiesen und gespeichert (`.save`).

Zusätzlich zu dieser Speicherung ist es auch die Aufgabe von `add new source` zu entscheiden, ob ein neuer Informationsagent angelegt werden muss oder nicht. Dazu wird über das Datenobjekt der Informationsagenten (`Iagent`) mittels der Methode `find` angefragt, ob Agenten zur Verfügung stehen, die noch nicht ihr Limit an überwachten Quellen erreicht haben. Dieses Limit wird als eine Konstante in der Webapplikation angelegt (`SOURCES PER AGENT`) und bei der Anfrage mit dem Feld `supervised info sources` überprüft, welches zu jedem Agenten in der Datenbank gespeichert wird (siehe 8.3). Das Auffinden eines freien Agenten erfolgt dabei wie in Beispiel 9.4 dargestellt.

```
available_agent = Iagent.find(
  :first,
  :conditions => "supervised_info_sources <#{SOURCES_PER_AGENT}")
```

Beispiel 9.4: Ermittlung eines freien Agenten in der Webapplikation

Falls nun also ein freier Agent gefunden wird, wird dieser dem neuen `InformationSource`-Objekt über das Datenbankfeld `iagent id` hinzugefügt. Sollte die Suche allerdings ergeben, dass kein Agent mehr verfügbar ist, wird über den `AgentController` die Aktion `create new agent` aufgerufen. Bei diesem Aufruf wird die Id der neu angelegten Informationsquelle übergeben, so dass später eine Zuordnung zwischen Agent und der richtigen Quelle erfolgen kann.

`Create new agent` erstellt nun zuerst ein neues `Iagent`-Objekt mit einem eindeutigen Namen (hier: „iAgent“ + Id des Agenten), speichert die aktuelle Zeit als Erstellungszeitpunkt ab (`created`) und setzt die Anzahl der überwachten Quellen des Agenten auf 1 (`supervised info sources`). Darauf folgt der wichtigste Schritt – die Kommunikation mit dem XML-RPC-Server, der auf dem Server der Agentenplattform läuft.

Dazu wird mittels `XMLRPC::Client.new` durch Angabe von Server-URL und Port ein neues XML-RPC-Objekt erstellt, auf dem dann der Aufruf der Methode `addAgent` mittels `call` durchgeführt wird. Als Parameter wird dabei die Id des neu angelegten `Iagent`-Objekts übergeben (siehe Beispiel 9.5).

```
server = XMLRPC::Client.new 'localhost', '/', 8080
status = server.call("addAgent", new_agent.id)
```

Beispiel 9.5: XML-RPC Aufruf über Ruby (hier: localhost und Port 8080)

Der Aufruf liefert einen booleschen Wert zurück, der Aufschluss über die fehlerfreie Ausführung des Aufrufs und der Erstellung des Agenten gibt.

9.5.3 Überwachung

Für die Überwachung bestehender Agenten, hinzugefügter Informationsquellen und schon extrahierter Ereignisse ist der `InformationController` zuständig. Er bietet Aktionen an, die die eben genannten Datenobjekte aus der Datenbank lesen und dem Benutzer zu informativen Zwecken präsentieren. In dieser prototypischen Implementierung der Managementapplikation werden dabei hauptsächlich einfache Tabellen angezeigt, die sich durch Verwendung von AJAX automatisch aktualisieren können. Bei der Anzeige der extrahierten Ereignisse wird zusätzlich eine Landkarte eingesetzt, auf der die Ereignisse angezeigt werden und ebenfalls automatisch aktualisiert werden.

Ereignisse in Tabellen Bei der Darstellung der extrahierten Ereignisse in einer Tabelle werden vom Controller zunächst einmal alle Ereignisse aus der Datenbank gelesen. Dieses geschieht durch die Ausführung von `Event.find(:all)`, wodurch eine Liste der Ereignisse zurückgegeben wird. Auf Template-Ebene wird nun durch diese `Event`-Objekte iteriert und die einzelnen Felder ausgegeben. Die automatische Aktualisierung der Tabelle erfolgt dabei durch die Hilfsfunktion (*helper function*) `periodically call remote`, die in einem bestimmten Rhythmus (`frequency`), die Ebene, in der die Tabelle eingebettet ist, neu lädt und mit Inhalt füllt¹³. Der Inhalt wird dabei von der Aktion `show events list` geliefert, die sich ebenfalls im `InformationController` befindet.

Darstellung in einer Landkarte Die Darstellung in einer Landkarte baut, grundsätzlich gesehen, auf den selben Funktionen auf, wie sie auch bei der Darstellung in einer Tabelle genutzt werden. Der Unterschied ist hierbei, dass mit einem externen Service zusammengearbeitet wird, der über eine API angesprochen werden kann und die Karte erzeugt. Im Falle dieser Thesis ist das der Google Maps Service bzw. die Google Maps API.

Da dieser Service zur Darstellung von Informationen innerhalb der Karte ein bestimmtes Datenformat erwartet, müssen die Ereignis-Daten entsprechend vorbereitet werden. Aus diesem Grund werden die `Event`-Objekte in einem RHTML-Template verarbeitet, welches ein XML-Dokument im passenden Google-Maps-Format erzeugt. Beispiel 9.6 zeigt die Erzeugung einer passenden XML-Datei.

```
<markers>
  <% for event in @events %>
    <marker lat='<%=event.latitude%>'
           lng='<%=event.longitude%>'
           info_source='<%=event.information_source.description%>'
           original_address='<%=event.original_address%>'
           description='<%=event.description%>' />
  <% end %>
</markers>
```

Beispiel 9.6: Erstellung der passenden Google-Maps-XML-Datei

Die so erzeugte XML-Datei dient danach als Eingabe für die von der API zur Verfügung gestellten Funktion `GDownloadUrl`, welche die einzelnen Markierungen der Ereignisse auf der Karte vornimmt. Auch hier findet eine automatische Aktualisierung statt, indem die Ebene, in der sich die Karte befindet, in regelmäßigen Abständen mit der Rückgabe aktualisiert wird, die die Action liefert, welche für die Darstellung der Karte zuständig ist (`show events map`).

¹³helper functions werden dabei von Ruby on Rails zur Verfügung gestellt.

Bemerkung: Es soll nochmal darauf hingewiesen werden, dass dieser Abschnitt über die Verwaltungsapplikation nur einen kurzen Überblick geben sollte, wie einige ausgewählte Funktionen grundsätzlich umgesetzt worden sind. Für detaillierte Informationen zu den einzelnen Teilen wird auf weiterführende Quellen wie z.B. die Google Maps API Dokumentation, Ruby on Rails Dokumentation oder [TH06, Tho05] verwiesen.

Zudem handelt es sich, wie schon erwähnt wurde, um eine prototypische Implementierung und bietet dadurch zahlreiche Möglichkeiten der Verbesserungen. Aus diesem Grund geht das folgende Kapitel nochmal auf diese Erweiterungen ein und wagt einen Ausblick auf zukünftige Weiterentwicklungen.

10

Ausblick und Erweiterungen

Das in dieser Arbeit konzipierte und umgesetzte System lässt aufgrund seines Umfangs viel „Spielraum“ für zukünftige Erweiterungen. Zum einen auf Seiten der Informationsagenten, deren Möglichkeiten noch viel mehr genutzt werden können. Zum anderen bei der Informationsextraktion, einem Forschungsgebiet, welches unzählige Ansätze bietet. Auch die Anbindung an das track-u System kann erweitert werden und zu guter letzt bietet die Managementapplikation vor allem im Bereich der Benutzeroberfläche viele Verbesserungsmöglichkeiten. Das vorliegende Kapitel soll nun einen kleinen Ausblick wagen und mögliche Erweiterungen vorstellen.

10.1 Informationsagenten

Der in dieser Arbeit entworfene System-Ansatz nutzt Agenten, welche relativ kleine Software-Einheiten bilden, dabei ihre Aufgabe erledigen, aber bei weitem nicht das Potential ausschöpfen, welches ein Multiagentensystem bietet. So wird beispielsweise keine Kommunikation unter den Agenten betrieben und es werden auch nicht Komponenten wie der Directory Facilitator (DF) eingesetzt.

Eine Möglichkeit der Erweiterung auf Seiten der Agenten wäre z.B. die gegenseitige **Unterstützung bei der Extraktion**.

Es kann unter Umständen vorkommen, dass ein Agent relativ lange nichts zu tun hat, weil er Quellen überwacht, die nur in großen Zeitabständen überprüft werden müssen. Während dieser Agent also „untätig“ ist, gibt es andere die pausenlos extrahieren. Eine Möglichkeit wäre nun, dass der weniger beschäftigte Agent dem DF mitteilt, dass er gerade verfügbar ist. Jeder Informationsagent könnte somit im Falle einer Überlastung beim DF nachfragen und passende Agenten zur „Hilfe anfordern“. Die Agenten würden nun miteinander kommunizieren und der beschäftigte Agent dem anderen die Quellen-Informationen mitteilen.

Eine andere Art der Erweiterung wäre z.B. wenn sich die Agenten gegenseitig **Mitteilungen über die laufenden Extraktionen** senden.

Es ist möglich, dass ein Ereignis aus mehreren Quellen extrahiert wird und somit später mehrfach im Repository vorhanden ist¹. Eine etwaige Mitteilung bietet dabei die Möglichkeit, dieses zu vermeiden, indem ein Agent nach einer Extraktion über ACL-Nachrichten den anderen Agenten mitteilt, welches Ereignis er gerade extrahiert hat. Die anderen Agenten können nun selbst schauen, ob sie ein Ereignis mit gleichem Ort, gleicher Zeit und „ähnlicher“ Beschreibung (z.B. Auftreten vieler gleicher Wörter) schon selbst extrahiert haben.

Problem ist natürlich hier der erhöhte Kommunikationsaufwand. Dieser könnte aber vermieden werden, wenn sich nur Agenten diese Nachricht schicken, die Quellen verarbeiten, die Ereignisse aus einem bestimmten Gebiet liefern. So müssten beispielsweise einem Agenten, der „Berliner Nachrichten“

¹Erinnerung: der eindeutige Schlüssel enthält ja auch die Quellen-Id.

überprüft, nicht Nachrichten gesendet werden, die sich auf den Raum München beziehen. Die Agenten müssten also noch über das Wissen verfügen, welche Gebiete verarbeitet werden.

Auch bei der **Geokodierung** könnten sich die Agenten gegenseitig unterstützen. Falls ein Agent z.B. beim Zugriff auf den externen Geokodierungs-Service (hier: Google-Geocoder) Probleme hat, wäre es möglich, dass er andere Agenten bittet, die Geokodierung für ihn zu übernehmen. So würde jeder Agent im Notfall die Aufgabe eines Geokodierungs-Agenten übernehmen, ähnlich dem der in 6.4.2 vorgestellt wurde.

Auf Seite 57 wurde definiert, dass jeder Agent für eine feste Anzahl von Informationsquellen zuständig ist. Zudem wurde auch eine **alternative Quellen-Verteilung** erwähnt, die den Auslastungsgrad der Agenten mit berücksichtigt. An dieser Stelle ist dadurch ebenfalls eine Erweiterung der Agenten möglich, bei der diese sich untereinander Quellen gegenseitig zuteilen können, abhängig von ihrer aktuellen Auslastung.

10.2 Informationsextraktion

Auf Seiten des Wrappers wurde in dieser Arbeit zum Großteil auf Expertenwissen bzw. auf die manuelle Erstellung von Extraktionsregel gebaut. Da der Bereich der Informationsextraktion aber ein riesiges Forschungsgebiet ist, welches zahlreiche Lösungen und Ansätze aufweist, sind für die IE-Komponente eine große Menge an Verbesserungen und Erweiterungen möglich.

Vor allem im Bereich der **Verarbeitung von natürlich-sprachlichen Texten** ist noch eine Menge Raum für Erweiterungen geboten. In der Thesis wurde dieser Bereich zwar erwähnt und auch beim Entwurf mitbedacht, aber nicht wirklich umgesetzt. Hier könnten entsprechende Methoden integriert werden, die vor allem Quellen verarbeiten können, bei denen die Informationen nicht über die Semistruktur zu erreichen sind.

Das vorliegende System setzt voraus, dass die zu überwachenden Informationsquellen manuell zum System hinzugefügt werden. Eine mögliche Erweiterung wäre es, ein **automatisches Auffinden von relevanten Informationsquellen** in das System zu integrieren und somit mehr in die Richtung Information Retrieval zu gehen. Neben dem Vorteil, dass der Knowledge Engineer entlastet wird und u.U. mehr Quellen einbezogen werden, als bei einem manuellen Vorgehen möglich wäre, gibt es hierbei allerdings einige Nachteile. Einer wäre beispielsweise der zu vermutende Qualitätsverlust, da u.U. viele Quellen einbezogen werden würden, die nicht wirklich korrekte Informationen liefern. Zudem müsste eine automatische Regelerstellung erfolgen, was ebenfalls keine triviale Aufgabe ist und somit auch relativ fehleranfällig sein kann.

In Abschnitt 5.4.2 wurden einige Probleme angesprochen, die bei der Extraktion im Zusammenhang mit Änderungen an der Quelle bzw. bei der Navigation auftreten können. Hier könnten entsprechende Erweiterungen beispielsweise ein **Caching der Seiten** vornehmen, bevor extrahiert wird. Im Bezug auf das **Hyperlink-Problem** könnten Methoden integriert werden, die die Extraktion aus Javascript-Funktionen möglich machen.

Bei großen Quellen wäre es zudem sinnvoll, eine Möglichkeit der **Zwischenspeicherung bereits extrahierter Ereignisse** zu integrieren und sich auch die aktuelle **Extraktions-Position** in der Quelle zu merken. Dieses könnte dann bei einem eventuellen Ausfall der Plattform genutzt werden, um an der Stelle fortzufahren, an der der Ausfall stattgefunden hat.

Weiterentwicklung des Web

Bei möglichen Erweiterungen der Informationsextraktion muss auch beachtet werden, dass mit der fortschreitenden Entwicklung des World Wide Web immer neue Möglichkeiten der IE hinzukommen und neuartige Ansätze entwickelt werden können.

Beispielsweise werden Informationsquellen immer mehr mit semantischen Daten versehen, die eine Extraktion erleichtern. So stellt z.B. die Einführung der so genannten **Microformats** eine Möglichkeit dar, um u.a. Geokoordinaten direkt in eine Webseite einzubinden [All07].

Dadurch könnte eine Informationsextraktion auf aufwendige Wrapper verzichten und gezielt Informationen auswählen.

Auch die Einbettung weiterer semantischer Daten, beispielsweise mit Hilfe von RDF, würde neue Wrapping-Ansätze nach sich ziehen.

10.3 Managementapplikation

Bei der Verwaltungsapplikation bieten sich Erweiterungen vor allen Dingen im Bereich der **Benutzeroberfläche** an. Wie schon in Abschnitt 7.2 erwähnt wurde, kann die Arbeit durch Einsatz von Techniken wie beispielsweise Ajax für den Knowledge Engineer komfortabler gemacht werden.

Auch bei der Erstellung der Regeln könnten Methoden eingesetzt werden, die den Benutzer unterstützen und den jetzigen manuellen Erstellungs-Prozess dem eines semi-automatischen näher bringen.

Zusätzlich ist auch bei den **administrativen Funktionen** einiges an Verbesserungen möglich. In dieser Arbeit wurde dieser Bereich bei der Umsetzung aus Umfangsgründen vernachlässigt. Beispielsweise könnten hier Funktionen zum Löschen von Quellen aus Agenten integriert werden.

Als weitere Ergänzung könnte in die Managementapplikation die **Möglichkeit des Neustartens aller Agenten** integriert werden. So müsste nicht der XML-RPC dafür genutzt werden. Beispielsweise könnte dabei eine neue zusätzliche Methode `restartAllAgents` in den XML-RPC-Server aufgenommen werden, die die JADE-Plattform runterfährt, neu startet und dann die Agenten wieder reaktiviert.

10.4 Anbindung an track-u

Die Anbindung an das track-u System erfolgt zur Zeit über die Datenbank, die der Eskalationslogik die notwendigen Daten zur Verfügung stellt. Das „Problem“ ist hier natürlich, dass bei einer Anbindung, der Entwickler auf Seiten des track-u Systems Kenntnisse der Datenbankstruktur haben muss und bei Änderungen seinen Programmcode entsprechend abändern muss.

Eine bessere Alternative der Anbindung wäre eine **wohldefinierte Schnittstelle**, die nach Außen zur Verfügung gestellt wird. Beispielsweise eine XML-Schnittstelle die per URL aufgerufen und angefragt werden kann und ein XML-Dokument mit den Ergebnissen zurückliefert.

So könnte das track-u System hierüber anfragen, ob Ereignisse innerhalb eines bestimmte geographischen Bereichs vorgefallen sind. Das System nimmt die Anfrage entgegen, wandelt sie in eine Datenbank-Anfrage auf den gespeicherten Ereignissen um und liefert die Antwort zurück.

Die Struktur von der Anfrage- und Antwortnachricht würde immer gleich sein und bei Änderungen auf Datenbankebene müsste sich das track-u System keine Gedanken darüber machen.

Zusätzlich zu dieser Schnittstelle, die für Anfragen der Eskalationslogik genutzt wird, bietet auch der Bereich der **Zonen-Einflussnahme** durch die Agenten, wie in 6.4.4 beschrieben wurde, einige Erweiterungsmöglichkeiten. Dieser Bereich wurde zwar in der Arbeit besprochen und analysiert, aber nicht umgesetzt. Hier könnten die Agenten dementsprechend erweitert werden, so dass sie prüfen können, ob ein Ereignis in einer bestimmten Zone der Geo-Datenbank vorkommt und diese abhängig vom jeweiligen Einfluss-Effekt ändern.

Hier muss ebenfalls bedacht werden, dass diese Änderungen auch wieder vom Agenten rückgängig gemacht werden müssen, falls das jeweilige Ereignis seine Haltbarkeit überschritten hat.

11

Zusammenfassung und Probleme

Mit diesem Kapitel wird nun der inhaltliche Teil der Arbeit abgeschlossen und ein zusammenfassender Überblick der Ergebnisse gegeben. Dabei wird das System und dessen Integration in das track-u Projekt nochmals kurz beschrieben und am Ende einige Probleme aufgezeigt, die während der Arbeit aufgetreten sind. Ergänzend werden noch mögliche Problemstellungen bei einem späteren Betrieb diskutiert.

11.1 Zusammenfassung

Die vorliegende Arbeit hatte zum Ziel, ein System zu entwerfen und umzusetzen, welches Ereignis-Informationen aus Internet-Quellen extrahieren und für die spätere Integration ins track-u System aufbereiten kann. Dabei sollte ein Multiagentensystem als unterstützende Plattform eingesetzt werden.

Nach einer ausführlichen Untersuchung bestehender Konzepte und Methoden der Informationsextraktion, wurde auf Basis einer Analyse von Beispielquellen der Entwurf eines eigenen Wrappers für dieses System vorgenommen. Der **Wrapper** baut dabei auf Techniken wie regulären sowie XPath-Ausdrücken auf, berücksichtigt aber auch natürlich-sprachliche Methoden. Die Besonderheit stellen dabei **kleine Extraktionseinheiten** dar, die jeweils unterschiedliche Abläufe nach sich ziehen und deren Verkettung komplexe Extraktionsabfolgen möglich machen. Am Ende dieser so genannten **Extraction-Chain** steht dabei eine Liste extrahierter Ereignisgruppen.

Die Wrapper-Komponente initiiert dabei nicht eigenständig die Extraktion, sondern ist der Kontrolle der **Informationsagenten** (iAgent) unterworfen. Dieser in der Arbeit entwickelte Agententyp, kontrolliert wann Extraktionen durchgeführt werden und teilt dem Wrapper ebenfalls mit, welche Informationsquellen extrahiert werden sollen.

Seine Informationen bezieht der iAgent dabei aus einem zentralen **Repository**, welches alle notwendigen Daten enthält und auch zur Speicherung der extrahierten Ereignisse genutzt wird.

Eine weitere Aufgabe, die vor dieser Speicherung erfolgen muss, ist die **Geokodierung** der Adressinformationen, welche ebenfalls in den Aufgabenbereich der Informationsagenten fällt und unter Zuhilfenahme einer externen Geokodierungs-API realisiert wurde.

Agenten und Wrapper-Komponente verrichten ihre Arbeit selbstständig im Hintergrund und entziehen somit direkten „Blicken“ eines Benutzers. Aus diesem Grund wurde eine **Managementapplikation** konzipiert und prototypisch in Form einer Webapplikation entwickelt, die die Benutzerschnittstelle darstellt.

Der Benutzer, welcher meistens ein Knowledge Engineer ist, erstellt über diese Applikation die Extraktionsregeln bzw. -ketten. Zusätzlich dazu bietet die Applikation Möglichkeiten, Informationen über den aktuellen Status der Informationsagenten, der Informationsquellen und der extrahierten Ereignisse abzurufen. Bei letzteren wurde ebenfalls eine visuelle Darstellung in einer Landkarte realisiert.

Die Verbindung zum track-u System erfolgte in dieser Arbeit unter Nutzung des Repositories. Mögliche Schnittstellen und alternative Anbindungen wurden ebenfalls diskutiert.

11.2 Probleme

Der Umfang des Systems und seiner Komponenten hat in dieser Arbeit dazu geführt, dass einige Teile nur kurz angesprochen wurden und nicht umgesetzt werden konnten. Vor allen Dingen der Bereich der Informationsextraktion aus natürlich-sprachlichen Texten wurde zwar diskutiert, aber nicht im System umgesetzt. Auch die Managementapplikation bietet großen Spielraum für Erweiterungen vor allem im Bereich der graphischen Benutzeroberfläche. Diese Erweiterungen wurden bedacht und im Kapitel 10 wurde einige Vorschläge diesbezüglich gegeben.

Performanz Da Tests nur anhand einiger weniger Quellen durchgeführt wurden und vor allem für größere Skalierungstests die Zeit gefehlt hat, kann nicht wirklich viel über spätere Performanz- bzw. Skalierungs-Eigenschaften des Systems gesagt werden und wie es beispielsweise aussieht, wenn mehrere hundert Quellen mit einer großen Anzahl von Agenten überwacht werden sollen. Speziell für die JADE-Agentenplattform existieren jedoch andere Arbeiten wie [VQC02], die u.a. Auskunft über die Skalierbarkeit auf Seiten des MAS geben können. Natürlich muss in diesem System auch noch die IE-Komponente bei einer Analyse betrachtet werden, da bei großen Quellen ein hoher Wrapping-Aufwand auf sie zukommt.

ABBILDUNGSVERZEICHNIS

3.1	Standard-Verarbeitungsschritte eines IE-Systems zur Verarbeitung von natürlich-sprachlichen Texten nach [AI99]	12
3.2	<i>Ideale Extraktion</i> von Daten	16
3.3	Aufgabe eines Wrappers im Kontext dieser Arbeit	17
3.4	Verteilung von Informationen mittels Layern	25
3.5	<i>one-level one page</i> Verteilung	26
3.6	Beispiel einer <i>one-level multi page</i> -Quelle	27
3.7	Beispiel einer <i>two-level pages</i> -Quelle	27
3.8	Beispiel einer <i>multi-level pages</i> -Quelle mit zwei Detailseiten	28
3.9	Beispielauflistung von Ereignisgruppen mit <i>two-level multi pages</i> und Formular-Erreichbarkeit	28
3.10	Beispiel eines regulären Ausdrucks zur Selektion von Informationsblöcken in einem semistrukturellen Dokument	29
3.11	Beispiel von Roh- und Feininformation anhand von einer Ortsangabe und einem Zeitpunkt	32
4.1	Grundstruktur des Systems	35
5.1	XML-Validität prüfen und danach entsprechend Verarbeitungsmethoden wählen	38
5.2	Ergebnis der Extraktion-Pattern vom Typ <i>time, location</i> oder <i>description</i>	40
5.3	Ergebnis der Extraktion-Pattern vom Typ <i>follow-link</i>	41
5.4	Ergebnis der Extraktion-Pattern vom Typ <i>group-relation</i>	41
5.5	Extraction-Pattern zur Extraktion von Zeitangaben mittels regulären Ausdrücken	42
5.6	Zusammenführung von Gruppen-Elementen (<i>time, location, description</i>) zu Ereignisgruppen	42
5.7	Iteration durch Ereignisgruppen in einer Quelle	43
5.8	Beispiel: Einfache Extraction-Pattern Verkettung und anschließendes Merging	45
5.9	Beispiel: Verkettung von 4 Extraction Pattern	46
5.10	Verarbeitung einer <i>one-level multi page</i> -Quelle	47

5.11	Verarbeitung einer <i>two-level pages</i> -Quelle	48
5.12	Beispiel: Verkettung mit 2 follow-links Extraction-Pattern	49
5.13	Integration einer Formularanfrage zur Erlangung der Ereignisgruppen	51
5.14	POST-to-GET Proxy	51
5.15	Systemstruktur nach IE-Entwurf	52
6.1	FIPA Referenzmodell für Agentenplattformen [FIP04]	55
6.2	Informationsagenten mit integriertem Wrapper	57
6.3	Agenten sind jeweils für die Verwaltung mehrerer Informationsquellen zuständig	57
6.4	Beispiel einer Geokodierung	59
6.5	Ablauf der Geokodierung mit Geo-Agenten	60
6.6	Erhöhte „Stau-Gefahr“ bei einem Geo-Agenten	60
6.7	Informationsagent mit Wrapper- und Geokodierungs-Komponente	61
6.8	Verbindung zwischen Verwaltungsapplikation und MAS (inkl. XML-RPC)	65
6.9	Aktualisierte Systemstruktur nach Entwurf des MAS	66
7.1	Flexible Zusammenstellung von Extraktionsketten	69
7.2	Informationsquelle mit einer Beschreibung und drei Tags, die Metadaten liefern.	71
7.3	Neuen Agenten über Verwaltungsapplikation erstellen	72
7.4	Aktualisierte Systemstruktur nach Entwurf der Mangementapplikation	74
8.1	Datenbankmodell für das Repository	76
8.2	Komplette Systemstruktur nach Repository-Entwurf	79
9.1	Ein über mehrere Hosts verteiltes Agentensystem mit JADE [BCTR06]	81
9.2	Klassen für die Wrapper-Umsetzung	85
9.3	Klassen für die Geokodierung und das Ereignismanagement	89
9.4	Klassen für die Agenten	90

TABELLENVERZEICHNIS

3.1	Beispiel einer Ereignisgruppe	19
3.2	Kriterien der Quellenbewertung	20
3.3	Quellenanalyse: eventim.de	21
3.4	Quellenanalyse: Newsfeed der Polizei Hamburg	21
3.5	Quellenanalyse: Fahrplanänderungen der Münchner Verkehrsbetriebe	22
3.6	Quellenanalyse: Internetauftritt der „Süddeutschen Zeitung“	22
7.1	Mögliche Einfluss-Werte auf Sicherheitslevel einer Zone	70

BEISPIELE

3.1	Unstrukturierter Text	7
3.2	Strukturierter Text	7
3.3	Semi-strukturierter Text	7
3.4	Semi-strukturierter Text der intern die Informationen in einer Art Struktur auflistet	8
3.5	Tokenisierung unter Verwendung des Natural Language Toolkits [BL04]	12
3.6	POS-Tagger Ausgabe [Bra98]	13
3.7	Koreferenz	15
3.8	Eindeutig identifizierbare Beziehung zwischen Gruppen-Elementen	23
3.9	Nicht-eindeutig identifizierbare Beziehung zwischen Gruppen-Elementen	24
3.10	Nicht eindeutig identifizierbare Beziehung zwischen unvollständigen Gruppen-Elementen	24
3.11	Nicht eindeutig erkennbare Gruppen-Relationen durch Verteilung auf der Seite	25
3.12	Nicht eindeutig identifizierbare Beziehung zwischen Gruppen-Elementen aufgrund von unstrukturiertem Text	25
3.13	Selektion der Ereignisse in Abb. 3.10 durch einen regulären Ausdruck	30
3.14	Aufzählung von Ereignissen in XHTML	30
3.15	XPath-Ausdruck zur Selektion der Ereignisse von Beispiel 3.14	31
3.16	Regulärer Ausdruck zur Selektion des Ortes in Beispiel 3.14	31
5.1	Ungenaue Ortsangabe	44
6.1	Pseudo-Code für Zeitpunkt der Informationsextraktion im Agenten	58
9.1	Body einer XML-RPC-Anfrage [Win99]	82
9.2	Body einer XML-RPC-Antwort [Win99]	82
9.3	Apache XML-RPC Server auf Port 8080 starten	82
9.4	Ermittlung eines freien Agenten in der Webapplikation	94
9.5	XML-RPC Aufruf über Ruby (hier: localhost und Port 8080)	94
9.6	Erstellung der passenden Google-Maps-XML-Datei	95

LITERATURVERZEICHNIS

- [AI99] APPELT, Douglas E. ; ISRAEL, David J.: *Introduction to Information Extraction Technology: A Tutorial Prepared for IJCAI-99*. SRI International. <http://www.dfki.de/~neumann/esslli04/reader/overview/IJCAI99.pdf>. Version: 1999
- [All07] ALLSOPP, John: *Microformats. Empowering Your Markup for Web 2.0*. friendsof, 2007
- [BAJ⁺01] BUITELAAR, P. ; ALEXANDERSSON, J. ; JAEGER, T. ; LESCH, S. ; PFLEGER, N. ; RAILEANU, D. ; BERG, T. von d. ; KLCKNER, K. ; NEIS, H. ; SCHLARB, H.: *An Unsupervised Semantic Tagger Applied to German*. citeseer.ist.psu.edu/buitelaar01unsupervised.html. Version: 2001
- [BCPR03] BELLIFEMINE, F. ; CAIRE, G. ; POGGI, A. ; RIMASSA, G.: JADE - A White Paper. In: *TILAB - EXP in search of innovation 3* (2003)
- [BCTR06] BELLIFEMINE, Fabio ; CAIRE, Giovanni ; TRUCCO, Tiziana ; RIMASSA, Giovanni: *JADE Programmer's Guide*. <http://jade.tilab.com/doc/programmersguide.pdf>. Version: August 2006
- [BFG01a] BAUMGARTNER, Robert ; FLESCA, Sergio ; GOTTLOB, Georg: Declarative Information Extraction, Web Crawling, and Recursive Wrapping with Lixto. In: *Lecture Notes in Computer Science 2173* (2001). citeseer.ist.psu.edu/baumgartner01declarative.html
- [BFG01b] BAUMGARTNER, Robert ; FLESCA, Sergio ; GOTTLOB, Georg: Visual Web Information Extraction with Lixto. In: *The VLDB Journal*, 2001, 119-128
- [BL04] BIRD, Steven ; LOPER, Edward: NLTK: The Natural Language Toolkit. In: *Proceedings of the ACL demonstration session* Association for Computational Linguistics, 2004, 214-217
- [Bra98] BRANTS, Thorsten: *TnT - Statistical Part-of-Speech Tagging*. <http://www.coli.uni-saarland.de/~thorsten/tnt/>. Version: Oktober 1998, Abruf: 15.04.2007
- [Bre03] BREUEL, Thomas M.: Information Extraction from HTML Documents by Structural Matching. In: *Proceedings of the Second International Workshop on Web Document Analysis*, 2003
- [BS98] BORGHOFF, U. ; SCHLICHTER, J.: *Rechnergestützte Gruppenarbeit - Eine Einführung in Verteilte Anwendungen*. Bd. 2. Auflage. Springer Verlag, 1998
- [Car97] CARDIE, Claire: Empirical Methods in Information Extraction. In: *AI Magazine* 18 (1997), Nr. 4, 65-80. citeseer.ist.psu.edu/cardie97empirical.html

- [CBC97] CHIDLOVSKII, B. ; BORGHOFF, U. ; CHEVALIER, P.: Towards sophisticated wrapping of web-based information repositories. In: *Proceedings of 5th International RIAO Conference, 1997*, 123–35
- [CD99] CLARK, J. ; DEROSE, S.: *XML Path Language (XPath)*. <http://www.w3.org/TR/1999/REC-xpath-19991116>. Version: November 1999
- [CL96] COWIE, Jim ; LEHNERT, Wendy: Information Extraction. In: *Communications of the ACM* 39 (1996), Nr. 1, 80–91. citeseer.ist.psu.edu/cowie96information.html
- [CL01] CHANG, Chia-Hui ; LUI, Shao-Chen: *IEPAD: Information Extraction Based on Pattern Discovery*. citeseer.ist.psu.edu/chang01iepad.html. Version: 2001
- [Eik99] EIKVIL, Line: Information Extraction from World Wide Web - A Survey / Norwegian Computing Center. Version: 1999. <http://citeseer.ist.psu.edu/eikvil99information.html>. 1999 (945). – Forschungsbericht
- [FG96] FRANKLIN, S. ; GRAESSER, A.: Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In: *Intelligent Agents III. Agent Theories, Architectures and Languages (ATAL'96)* Bd. 1193. Berlin, Germany : Springer-Verlag, 1996
- [FIP04] FIPA: *FIPA Agent Management Specification*. <http://www.fipa.org/specs/fipa00023/SC00023K.pdf>. Version: 2004
- [Fou06] FOUNDATION, Apache S.: *About Apache XML-RPC*. <http://ws.apache.org/xmlrpc/>. Version: 2006
- [Gei03] GEIGER, Jörg: *Open-Source-Groupware. Überblick, Kategorisierung, Auswahl und Installation*, Technische Universität München, Diplomarbeit, 2003. <http://www11.informatik.tu-muenchen.de/publications/html/Geiger2003/diplom.html>
- [GS96] GRISHMAN, Ralph ; SUNDHEIM, Beth: Message Understanding Conference - 6: A Brief History. In: *Proceedings of the 16th International Conference on Computational Linguistics, 1996*
- [GW98] GAIZAUSKAS, R. ; WILKS, Y.: Information Extraction: Beyond Document Retrieval. In: *Journal of Documentation* 54 (1998), Nr. 1, 70–105. citeseer.ist.psu.edu/gaizauskas98information.html
- [HD98] HSU, Chun-Nan ; DUNG, Ming-Tzung: Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web. In: *Information Systems* 23 (1998), Nr. 8, 521–538. citeseer.ist.psu.edu/hsu98generating.html
- [HM03] HAROLD, E. R. ; MEANS, W.Scott: *XML in a nutshell*. Bd. 2. O'Reilly, 2003
- [KSNM03] KLEIN, D. ; SMARR, J. ; NGUYEN, H. ; MANNING, C.: *Named entity recognition with character-level models*. citeseer.ist.psu.edu/klein03named.html. Version: 2003
- [Kus00] KUSHMERICK, Nicholas: Wrapper induction: Efficiency and expressiveness. In: *Artificial Intelligence* 118 (2000), Nr. 1-2, 15-68. citeseer.ist.psu.edu/kushmerick00wrapper.html

- [KWD97] KUSHMERICK, Nickolas ; WELD, Daniel S. ; DOORENBOS, Robert B.: Wrapper Induction for Information Extraction. In: *Intl. Joint Conference on Artificial Intelligence (IJCAI)*, 1997, 729–737
- [MM04] MANOLA, F. ; MILLER, E.: *RDF Primer*. <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>. Version: February 2004
- [MTU⁺01] MAYNARD, D. ; TABLAN, V. ; URSU, C. ; CUNNINGHAM, H. ; WILKS, Y.: *Named Entity Recognition from Diverse Text Types*. citeseer.ist.psu.edu/maynard01named.html. Version: 2001
- [Myl01] MYLLYMAKI, Jussi: Effective Web data extraction with standard XML technologies. In: *World Wide Web*, 2001, 689-696
- [PPW02] PICHLER, J. ; PLÖSCH, R. ; WEINREICH, R.: MASIF und FIPA: Standards für Agenten. In: *Informatik Spektrum* Band 24 (2002), Nr. Heft 2, S. 91–100
- [QRS⁺95] QUASS, Dallon ; RAJARAMAN, Anand ; SAGIV, Yehoshua ; ULLMAN, Jeffrey D. ; WIDOM, Jennifer: Querying Semistructured Heterogeneous Information. In: *Deductive and Object-Oriented Databases*, 1995, 319-344
- [Rag] RAGGETT, D.: *Clean up your Web pages with HTML TIDY*. <http://www.w3.org/People/Raggett/tidy/>
- [Ril99] RILOFF, E.: *Information Extraction as a Stepping Stone toward Story Understanding*. citeseer.ist.psu.edu/riloff99information.html. Version: 1999
- [Ses04] SESTER, P.: Vertragsabschluss und Verbraucherschutz beim Einsatz von Software-Agenten. In: *Informatik Spektrum* Band 27 (2004), Nr. Heft 4, S. 311–322
- [Sod99] SODERLAND, Stephen: Learning Information Extraction Rules for Semi-Structured and Free Text. In: *Machine Learning* 34 (1999), Nr. 1-3, 233-272. citeseer.ist.psu.edu/soderland99learning.html
- [SS05] SINIAKOV, Peter ; SIEFKES, Christian: An Overview and Classification of Adaptive Approaches to Information Extraction. In: *Journal on Data Semantics IV* (2005), 172–212. <http://siniakov.de.vu/articles/overview-ie.pdf>. – LNCS 3730
- [TH06] THOMAS, Dave ; HANSSON, David H.: *Agile Web Development with Rails*. The Pragmatic Bookshelf, 2006
- [Tho05] THOMAS, Dave: *Programming Ruby*. The Pragmatic Bookshelf, 2005
- [VQC02] VITAGLIONE, G. ; QUARTA, F. ; CORTESE, E.: *Scalability and Performance of JADE Message Transport System*. citeseer.ist.psu.edu/vitaglione02scalability.html. Version: 2002
- [VR79] VAN RIJSBERGEN, C. J.: *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow, 1979 citeseer.ist.psu.edu/vanrijsbergen79information.html
- [Wik06] WIKIPEDIA: *Koreferenz — Wikipedia, Die freie Enzyklopaedie*. <http://de.wikipedia.org/w/index.php?title=Koreferenz&oldid=22619856>. Version: 2006. – [Online; Stand 13. April 2007]

- [Wik07a] WIKIPEDIA: *Deep Web* — *Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Deep_Web&oldid=121969955.
Version: 2007. – [Online; accessed 13-April-2007]
- [Wik07b] WIKIPEDIA: *Geocoding* — *Wikipedia, The Free Encyclopedia*. <http://en.wikipedia.org/w/index.php?title=Geocoding&oldid=120995044>.
Version: 2007. – [Online; accessed 13-April-2007]
- [Wik07c] WIKIPEDIA: *RSS* — *Wikipedia, The Free Encyclopedia*. <http://en.wikipedia.org/w/index.php?title=RSS&oldid=122214799>. Version: 2007. – [Online; accessed 14-April-2007]
- [Win99] WINER, Dave: *XML-RPC Specification*. <http://www.xmlrpc.com/spec>.
Version: Juni 1999
- [Yin06] YING, Jun: *Analysis and Comparison of Existent Information Extraction Methods*, Darmstadt University of Technology, Diplomarbeit, 2006