Technische Universität Dresden
Fakultät Informatik
Institut für Systemarchitektur
Lehrstuhl Rechnernetze
Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Communardo Software GmbH
Dresden

Diplomarbeit

Personalisierte Filterung von Nachrichten aus semistrukturierten Quellen

zum
Erlangen des akademischen Grades

Diplom-Medieninformatiker

Eingereicht von:	Thomas Eixner Matrikelnummer 2959721
Verantwortlicher Hochschullehrer:	Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill
Betreuer TU Dresden:	Dipl.-Inf. Marius Feldmann
Betreuer Communardo:	Dipl. Wirt.-Inf. Dirk Röhrborn Dipl.-Inf. Torsten Lunze
Eingereicht am:	30. April 2009

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Diplomarbeit selbstständig ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Diese Arbeit ist bislang keiner anderen Prüfungsbehörde vorgelegt worden und auch nicht veröffentlicht worden.

Dresden 30.04.2009

Unterschrift

Inhaltsverzeichnis

1 Einleitung ... 1
1.1 Motivation ... 1
1.2 Ziele ... 1
1.3 Aufbau ... 2

2 Grundlagen ... 3
2.1 Personalisierung ... 3
2.1.1 Motivation ... 3
2.1.2 Vorgehensweise ... 4
2.1.3 Verfahren zur Personalisierung ... 5
2.1.4 Benutzermodellierung ... 8
2.1.5 Filtermechanismen ... 10
2.1.5.1 Inhaltsbasierendes Filtern ... 10
2.1.5.2 Kollaboratives Filtern ... 12
2.1.5.3 Item-based-Collaborative-Filtering ... 14
2.1.5.4 Probleme von Filtermechanismen ... 15
2.2 Publish / Subscribe Systeme ... 16
2.2.1 Distributor Aware Subscribtions ... 17
2.2.1.1 Microblogging ... 17
2.2.1.2 XMPP ... 23
2.2.2 User Client Aware Subscriptions ... 24
2.2.2.1 Web Feeds ... 25
2.2.2.2 Usenet ... 28
2.3 Consumer Aggregatoren ... 28
2.3.1 Feed Aggregatoren ... 28
2.3.2 Microblogging Aggregatoren ... 30
2.4 Enterprise Aggregatoren ... 31
2.4.1 Anforderungen ... 32
2.4.2 Attensa ... 33
2.4.3 Newsgator Social Sites ... 34
2.4.4 Zusammenfassung ... 36

3 Realisierung der Infrastruktur ... 37
3.1 Detailanalyse und Konzeption ... 37
3.1.1 Nutzergruppenanalyse ... 37
3.1.2 Anwendungsfälle ... 38
3.1.3 Datenfilterung und Datenhaltung ... 39
3.1.4 Internes Nachrichtenformat ... 42
3.1.4.1 RSS ... 42
3.1.4.2 Atom ... 43
3.1.4.3 Twitter ... 44
3.1.4.4 Communote ... 45
3.1.5 Infrastruktur ... 46

3.1.6 Zusammenfassung ... 48
3.2 Entwurf und Implementierung ... 49
3.2.1 Umgebungsmodell ... 49
3.2.2 Top-Level Architektur ... 49
3.2.3 Nutzermodell ... 50
3.2.4 Internes Nachrichtenformat ... 51
3.2.5 Datenhaltung ... 54
3.2.6 Filtermechanismus ... 55
3.2.7 Aufbau der Anwendung ... 60
3.2.7.1 Connectoren ... 60
3.2.7.2 Kommunikation mit externen Diensten ... 63
3.2.7.3 Dienste der Filterung ... 64
3.2.7.4 Frontend Schnittstelle ... 64
3.2.8 Automatische Metadatensuche ... 65
3.2.9 Implementierung der Architektur ... 66
3.2.9.1 Datendienste ... 68
3.2.9.2 Frontend Schnittstelle ... 69

4 Evaluierung ... 71
4.1 Ziele ... 71
4.2 Durchführung ... 72
4.2.1 Prüfung der Anforderungen ... 72
4.2.2 Evaluierung der Aggregation und Übersetzung ... 72
4.2.3 Evaluierung des Nutzermodells ... 73
4.2.4 Evaluierung der automatischen Metadatensuche ... 73
4.2.5 Evaluierung von Anwendungskonfigurationen ... 73
4.2.6 Evaluierung der Ähnlichkeitsanalyse ... 76
4.2.7 Skalierbarkeit und Leistungsfähigkeit ... 77
4.3 Auswertung ... 78
4.3.1 Bewertung der Anforderungen ... 78
4.3.2 Bewertung der Aggregation und Übersetzung ... 79
4.3.3 Bewertung des Nutzermodells ... 80
4.3.4 Bewertung der automatischen Metadatensuche ... 81
4.3.5 Bewertung von Anwendungskonfigurationen ... 81
4.3.6 Bewertung der Ähnlichkeitsanalyse ... 82
4.3.7 Bewertung der Skalierbarkeit und Leistungsfähigkeit ... 83

5 Verbesserungsmöglichkeiten ... 85
5.1 Nutzermodell und automatische Metadatensuche ... 85
5.2 Bewertungssystem ... 86
5.3 Filterung ... 87
5.4 Implementierung ... 87

6 Zusammenfassung und Fazit ... 89

A Anhang ... 91
A.1 Inhalt der DVD ... 91

A.2 Klassendiagramme ... 91
A.2.1 Entitäten ... 92
A.2.2 Servicelayer ... 96
A.2.3 Dienste für externe Zugriffe ... 98
A.2.4 Filterengine ... 98
A.2.5 Schnittstelle ... 99
A.2.5.1 API Value Objects ... 101
A.2.6 Anwendungskonfiguration ... 102
A.2.7 Deliciouskonfiguration ... 102
A.2.8 Sicherheitskomponente ... 103
A.2.9 Enummerations ... 103
A.3 Sequenzdiagramme ... 103
A.4 Diagramme der Evaluierung ... 104
A.5 Komplettübersicht der Architektur ... 106

Abbildungsverzeichnis ... 107

Tabellenverzeichnis ... 109

Literaturverzeichnis ... 110

Abkürzungsverzeichnis ... 115

1 Einleitung

1.1 Motivation

Im Laufe der Entwicklung des Webs entstand eine Vielzahl unterschiedlicher Möglichkeiten der Informationsdistribution. Dabei entwickelten sich verschiedene heterogene Nachrichtenquellen, die von Endnutzern gleichermaßen verwendet werden. Dieses vielfältige Angebot von Informationsquellen wirft jedoch eine Reihe neuer Probleme auf. Zum Einen resultiert die angebotene Menge an Informationen für viele Endnutzer in einer kaum überschaubaren Informationsflut, in der es viel Zeit erfordert, relevante bzw. interessante Informationen herauszufiltern. Zum Anderen führt die Verteilung der Informationsquellen auf voneinander unabhängige und heterogene Nachrichtenquellen zu einem erhöhten Zeitaufwand, jede abonnierte Quelle auf neue Informationen zu prüfen.
Darüber hinaus beschränken sich die Informationen, die einem Endnutzer unmittelbar zur Verfügung stehen, auf die von ihm abonnierten Inhalte. Relevante bzw. interessante Inhalte aus Quellen, die ihm unter Umständen nicht bekannt sind, stehen ihm folglich nicht ohne Weiteres zur Verfügung.
Derzeit existieren Anwendungen, die es Nutzern ermöglichen, Nachrichtenquellen eines spezifischen Formates zu aggregieren und in wenigen Fällen ebenfalls automatisiert zu filtern. Andere Anwendungen ermöglichen es Nutzern, Empfehlungen für interessante Inhalte eines spezifischen Nachrichtenformates auszusprechen. Eine Lösung, die sich aller dargelegten Probleme annimmt und aggregierte heterogene Nachrichtenquellen personalisiert filtert oder gar ein Empfehlungssystem beinhaltet, existiert derzeit jedoch nicht.

1.2 Ziele

Ziel dieser Arbeit ist es, Lösungsansätze für die genannten Probleme aufzuzeigen. Hierzu soll im Rahmen dieser Diplomarbeit ein Konzept für eine Infrastruktur entwickelt werden, die es ermöglicht, Nachrichten aus heterogenen semistrukturierten Quellen zu aggregieren und diese zu einem einheitlichen Nachrichtenstrom zusammenzufassen. Dabei kommt der Erweiterbarkeit der Infrastruktur hinsichtlich weiterer Nachrichtenformate eine besondere Bedeutung zu, um prinzipiell die Möglichkeit zu gewährleisten, weitere Formen von Nachrichtenquellen zu unterstützen.
Darüber hinaus ist die Entwicklung eines personalisierten Filter- und Empfehlungssystems für aggregierte Nachrichten ein weiteres Ziel dieser Arbeit. Hierzu soll erarbeitet werden, welche Möglichkeiten zur Erreichung dieses Ziels zur Verfügung stehen und welche Rahmenbedingungen und Funktionalitäten die Grundlage für derartige Systeme bilden.
Um die erarbeiteten Konzepte zu validieren und deren Verbesserungsmöglichkeiten aufzudecken, soll die konzipierte Infrastruktur prototypisch implementiert werden.

Zusammenfassend soll die schriftliche Ausarbeitung dieser Arbeit sowie die Implementierung der prototypischen Infrastruktur Antworten auf folgende Kernfragen des Themas liefern:

- Wie kann eine personalisierte Filterung sowie ein Empfehlungssystem für aggregierte Nachrichtenquellen umgesetzt werden?
- Wie kann eine effiziente Infrastruktur umgesetzt werden?
- Mit welchen gruppenbasierten Mechanismen kann ein Filterungs- bzw. Empfehlungssystem umgesetzt werden?
- Wie können Nutzer und deren Interessen in eine Datenstruktur abgebildet werden und welche Lösungsansätze sind für die dabei auftretenden Probleme denkbar?
- Welche Konzepte sollten im Rahmen zusätzlicher Arbeiten weiterentwickelt werden?

1.3 Aufbau

Grundlage für die Entwicklung des bereits beschriebenen Konzeptes bildet eine Analyse von bereits vorhandenen Verfahren zur Personalisierung von Anwendungen. Dazu werden zu Beginn der Arbeit in Kapitel 2 – Grundlagen - die verschiedenen Aspekte von Personalisierungsverfahren betrachtet und dokumentiert. Diese Analyse beinhaltet ebenfalls die Beschreibung von Möglichkeiten zum Aufbau von Benutzermodellen und die Beschreibung von bekannten inhaltsbasierenden sowie gruppenbasierenden Filtermechanismen. Anschließend werden verschiedene Nachrichtenformate, die für eine Filterung in Frage kommen näher erläutert und deren Funktionsweise und Aufbau beschrieben. Abschließend werden derzeitig verfügbare Aggregatoren analysiert, um einen Überblick über den aktuellen Stand der Technik in diesem Bereich zu geben

Im Rahmen von Kapitel 3 – Realisierung der Infrastruktur - wird der Softwareentwicklungsprozess der Infrastruktur dokumentiert. Dabei wird detailiert auf die erarbeiteten Konzepte zur Aggregation, Filterung und Empfehlung eingegangen. Darüber hinaus werden die Anforderungen, der Entwurf und die Implementierung beschrieben.

Darauffolgend werden die erarbeiteten Konzepte in Kapitel 4 einer Evaluierung unterzogen, um die Grundlage für die Beschreibung von Problemen und Verbesserungsmöglichkeiten zu legen.

Abschließend werden in Kapitel 5 – Verbesserungsmöglichkeiten - die durch die Evaluierung erkannten Probleme analysiert sowie weiterführende Ideen dokumentiert, um Empfehlungen für eine Weiterentwicklung der Infrastruktur zu geben.

2 Grundlagen

2.1 Personalisierung

2.1.1 Motivation

Personalisierung bzw. Adaptivität im Web tritt in vielen unterschiedlichen Ausprägungen auf. Dies beginnt bei der simplen Anpassung der verwendeten Sprache oder des Seitendesigns und zieht sich über die Anpassung von Inhalten bis zu vollständig personalisierbaren Portalen, in denen Nutzer selbst die Inhalte und Anwendungen des Portals auswählen und konfigurieren können. Im Rahmen dieser Arbeit ist dabei zu beachten, dass die Begriffe Personalisierung und Filterung immer in Anwendung auf Nachrichtenströme zu verstehen sind.

Ein hauptsächlicher Grund für die Personalisierung von Nachrichtenströmen ist die meist unüberschaubare Menge an Informationen, die für einen Nutzer verfügbar sind. Dieser Fakt macht es notwendig, geeignete Verfahren zu verwenden, die den Nutzer dabei unterstützen, die für ihn relevanten Inhalte aus der Gesamtmenge der Informationen herauszufiltern. Eine derartige personalisierte Filterung weist dabei folgende Vorteile auf:

- Steigerung der Attraktivität des Webangebots
- Zeitersparnis für den Nutzer bei der Suche nach relevanten Informationen
- Automatisierte Gruppierung zusammengehöriger Informationen aus ggf. verschiedenen Quellen
- Bildung von Grundlagen für neue Ansätze der Informationsvisualisierung
- Nutzung von Präferenzen anderer Nutzer eines Systems

Um ein solches Angebot bzw. Verfahren zu realisieren ist es notwendig, den Benutzer serverseitig auch über längere Zeiträume eindeutig zuordnen zu können und dessen gewünschte Konfiguration oder dessen Präferenzen zu speichern.

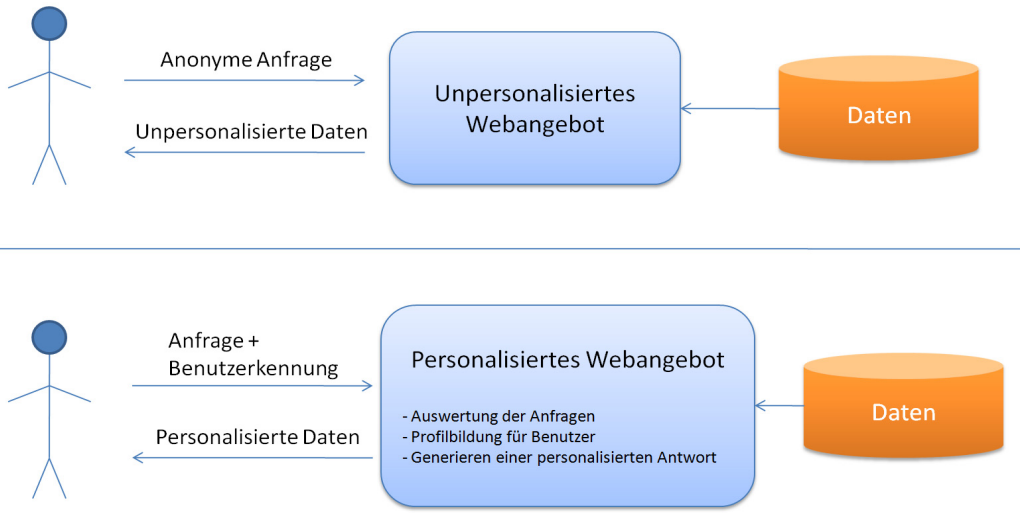

Abbildung 1: Personalisierte und unpersonalisierte Webangebote nach [Klossek+03]

Im Kontext dieser Arbeit kommt der Betrachtung von Informationsaggregatoren für Unternehmen eine große Bedeutung zu. Ziel solcher Aggregatoren ist es unter Anderem, aus der gegebenen Menge interner und externer Informationsflüsse die Informationen herauszufiltern, die für einen Nutzer relevant sind. Damit eine solche Funktion realisiert werden kann, muss dem Filtersystem zunächst eine entsprechende Datenbasis zur Verfügung gestellt werden, um die Interessen des Nutzers abzubilden. Hierfür existiert bereits eine Reihe von Möglichkeiten, um die dafür notwendigen Daten zu sammeln. Nachfolgend werden die Schritte, die für eine Personalisierung erforderlich sind vorgestellt und die wichtigsten Verfahren zur Realisierung eines solchen Angebots näher erläutert.

2.1.2 Vorgehensweise

Um ein personalisiertes Webangebot zu realisieren, muss eine Reihe von Schritten durchlaufen werden.

Im ersten Schritt muss die zu personalisierende Datenbasis näher beschrieben werden, um die Grundlage für eine spätere Filterung zu stellen. In den meisten Fällen werden die Ressourcen bzw. Inhalte, die das System verwaltet, bewertet oder mit Metainformationen versehen. Im zweiten Schritt müssen die Benutzer des Systems durch geeignete Verfahren in eine Form abgebildet werden, die für einen anschließenden Filterprozess verwertbar ist. Dieser Filterprozess muss mit entsprechenden Verfahren umgesetzt werden und liefert als Endprodukt die Ausgabe des Systems, die dem Nutzer präsentiert wird. Im letzten Schritt kann dem Nutzer die Möglichkeit geboten werden, dem System ein Feedback über die erhaltenen Inhalte zu geben. Dieses Feedback kann dazu genutzt werden, um die ausgelieferten Ressourcen zu bewerten oder das Profil des Nutzers zu verfeinern.

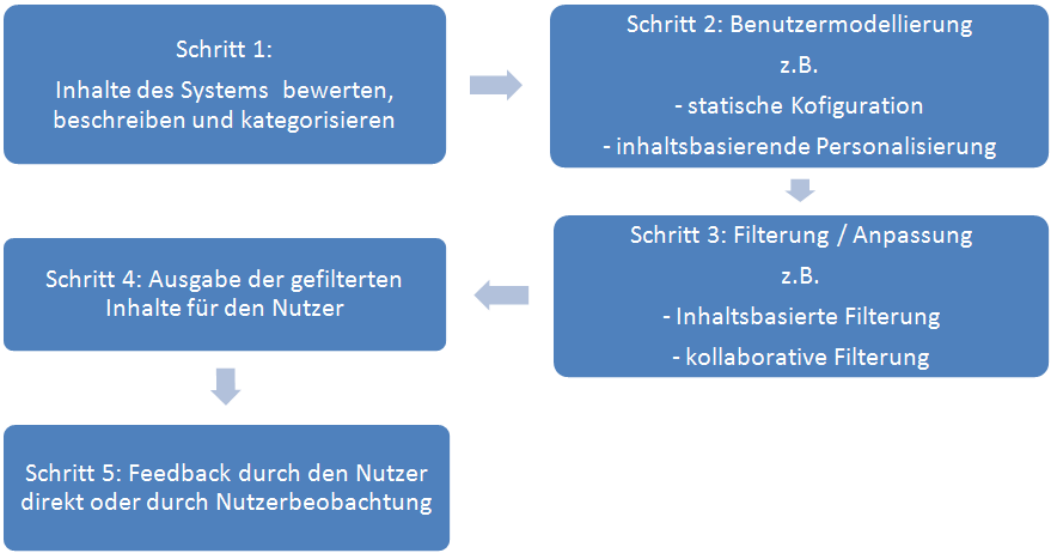

Abbildung 2: Schritte des Personalisierungsprozesses

In den nachfolgenden Abschnitten und Kapiteln werden die in Abbildung 2 dargestellten Schritte bzw. deren Verfahren näher erläutert. Dabei werden, aus Gründen der Vollständigkeit, auch allgemeine Verfahren zur Personalisierung dargelegt, die nicht zwangsläufig auf die Filterung von Inhalten abzielen (z.B. „statische Konfiguration“).

2.1.3 Verfahren zur Personalisierung

Im Rahmen der Analyse von Verfahren, die dazu genutzt werden können, eine Datenbasis für die Personalisierung zu gewinnen, wurden diese in zwei Kategorien, implizit und explizit, eingeteilt. Explizite Verfahren beruhen auf der Tatsache, dass ein Nutzer aktiv und wissentlich Daten über seine persönlichen Präferenzen preisgibt. Implizite Verfahren hingegen basieren darauf, dass das System die Aktivität des Nutzers analysiert und auf Grund dieser Daten entsprechend handelt oder nutzerbezogene Informationen aus einer anderen Quelle erhält und diese verwertet.

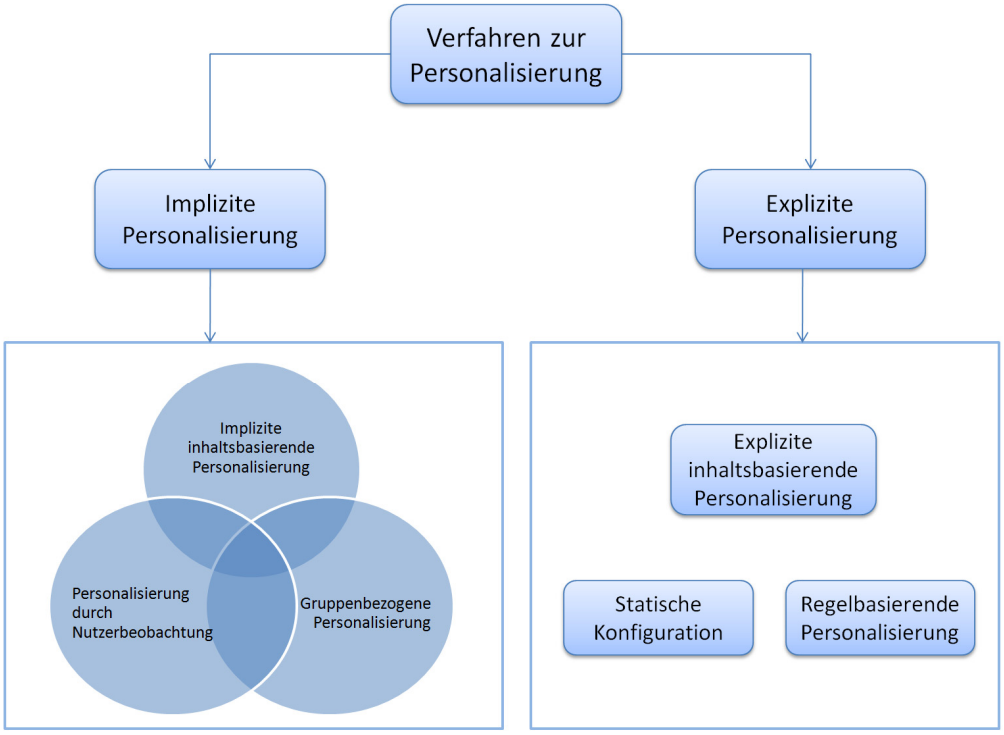

Abbildung 3: Verfahren zur Personalisierung

Statische Konfiguration

Eine sehr häufig eingesetzte Methode, um Nutzerdaten bzw. Nutzerpräferenzen zu gewinnen, ist die statische Konfiguration. Mit der expliziten Angabe von Konfigurationen werden häufige Elemente wie beispielsweise Sprache, Layout oder Interessengebiete personalisiert. Das heißt, dem Nutzer wird eine Reihe von Möglichkeiten geboten, die Ausgabe des Systems seinen Wünschen entsprechend zu variieren. Des Weiteren können über die Abfrage von nutzerbezogenen Daten, wie Wohnort, Alter und Geschlecht wichtige Informationen für eine automatisierte Personalisierung gewonnen werden. Zur statischen Konfiguration zählen ebenfalls personalisierbare Startseiten wie „iGoogle“, die die Möglichkeit bieten, Minianwendungen (Gadgets) in eine Seite zu integrieren. Dem Nutzer steht dabei frei, welche Gadgets mit welchen Einstellungen er in seine persönliche Seite integrieren möchte.

Regelbasierende Personalisierung

Eine regelbasierende Personalisierung bzw. Filterung lässt sich im Allgemeinen immer auf die Anwendung einer Regel zu einem Inhalt zurückführen. Nutzer können hierbei Regeln oder Parameter für Regeln definieren und sie einer Informationsquelle zuordnen. Üblicherweise hat eine solche Regel die folgende Form [Klossek+03]:

wenn (Bedingung ist wahr) dann Aktion

Typische Beispiele für Bedingungen sind die Angabe von Alter, Geschlecht, Wohnort oder Schlagwörtern. Anhand dieser Bedingungen werden

Informationsströme personalisiert und gegebenenfalls gefiltert. Selbstverständlich können auch mehrere Bedingungen verwendet werden, damit eine Aktion ausgeführt werden kann. Bezogen auf die Filterung von Informationsströmen ist es somit möglich, dass ein Nutzer mehrere Schlagwörter angibt, die seinen Interessenbereich näher spezifizieren. Diese Schlagwörter werden dazu verwendet, die für den Nutzer interessanten Inhalte aus dem Informationsstrom herauszufiltern. Ein praktisches Beispiel hierfür ist der Dienst „Yahoo Pipes“, der im Kapitel 2.3.1 näher beschrieben wird.

Gruppenbezogene Personalisierung
Gruppenbezogene Personalisierung [KEMPER+06] kommt häufig zur Anwendung, wenn die Nutzer eines Systems in verschiedene Gruppen mit unterschiedlichen Eigenschaften eingeteilt werden können. Das können beispielsweise Zugangsberechtigungen für verschiedene Bereiche, Optionen oder Features des Systems sein, die nicht zugangsberechtigten Nutzern von vorn herein nicht angezeigt werden. Nutzer werden in diesem Fall einer oder mehreren Gruppen zugeordnet, die mit den entsprechenden Berechtigungen versehen sind. Dies kann entweder durch externe Verzeichnisdienste (z.B. „LDAP“) geschehen oder anhand einer Zuordnung, die vom verwaltenden System ausgeführt wird, entschieden werden. Denkbar sind auch verschiedene Angebote in Online Shops, die für Stammkunden, Händler und normale Kunden unterschiedliche Preise ausweisen.

Personalisierung durch Nutzerbeobachtung
Nutzer hinterlassen bei der Benutzung von Webangeboten eine Vielzahl für die Personalisierung verwertbarer Informationen. Durch die Analyse von Logfiles oder durch geeignete Mechanismen zur Nutzerverfolgung können Daten über Navigationswege und Informationen, welche Elemente angeklickt wurden und für welchen Zeitraum sich der Nutzer bestimmte Seiten des Systems angesehen hat, gewonnen werden [RAHM+02]. Diese erhaltenen Informationen können wiederum für die dynamische Anpassung eines Systems verwendet werden. Denkbar sind Funktionen, die Nutzern zu den bereits angesehenen Dokumenten ähnliche Dokumente für die Betrachtung vorschlagen, wenn dieser Nutzer nach einem längeren Zeitraum das System wieder besucht. Zudem wird diese Art der Personalisierung häufig dazu genutzt, dem Nutzer eine Historie der bereits besuchten Dokumente in seiner aktuellen Sitzung anzuzeigen.

Inhaltsbasierende Personalisierung
Inhaltsbasierende Personalisierung kann nach zwei verschiedenen Prinzipien realisiert werden, „top down“ und „bottom up“ [Klossek+03]. Beim „top down“ Ansatz wird davon ausgegangen, dass Nutzer ihre Interessenbereiche auf einige Kategorien von Ressourcen einschränken. Diese Kategorien müssen von den Nutzern manuell ausgewählt werden, um eine spätere Filterung zu gewährleisten. Diese Tatsache führt allerdings häufig dazu, dass die Akzeptanz für dieses Verfahren bei den Nutzern vergleichsweise gering ausfällt. Zudem geben Nutzer ihre Interessenbereiche meist zu Beginn der Nutzung des Systems an. Mit der Zeit können sich diese Interessen jedoch ändern. Häufig werden in einem solchen Fall die

im System hinterlegten Daten nicht durch die Nutzer aktualisiert [Klossek+03], was bei einem Filterprozess auf Grundlage dieser Daten zu falschen Ergebnissen führt.

Der „bottom up“ Ansatz geht davon aus, dass der Nutzer einzelne Ressourcen bewertet und somit Auskunft über seine Präferenzen gibt. Dies kann entweder explizit, durch eine vom Nutzer wissentlich getroffene Bewertung der Ressource geschehen oder implizit durch eine vom System geschlussfolgerte Bewertung der Ressource für den Nutzer (vgl. Abbildung 3: Explizite und implizite inhaltsbasierende Personalisierung). Beispielsweise kann das einfache Ansehen von Detailinformationen dieser Ressource dazu verwendet werden, dass das System annimmt, dass diese Ressource für den Nutzer von Interesse ist. Darüber hinaus kann der Kauf von Artikeln in einem Online Shop ein Interesse für diesen Artikel bzw. für artverwandte Artikel implizieren. Diese vom System vorgenommene Art der Informationsgewinnung ist Teil der Kategorie „Implizite inhaltsbasierende Personalisierung“ (vgl. Abbildung 3) und weist durch die automatische Informationsgewinnung Schnittmengen mit der Kategorie „Personalisierung durch Nutzerbeobachtung“ auf.

Die gewonnen Daten werden häufig dazu genutzt, dem Nutzer Vorschläge für Ressourcen zu geben, die für ihn interessant sein könnten. Einen Einblick in die technische Vorgehensweise beim inhaltsbasierenden Filtern bietet der Abschnitt 2.1.5.1.

2.1.4 Benutzermodellierung

Im vorangegangenen Kapitel wurden Verfahren für die Ermittlung von Nutzerdaten erläutert. Um die mit Hilfe dieser Verfahren gewonnenen Daten nutzbar zu machen, müssen sie in ein Benutzermodell übertragen werden.

Dabei existieren verschiedene Arten von Informationen, die in einem Benutzerprofil hinterlegt werden können. Die folgende Auflistung gibt hierzu einen Überblick [Henze+06]:

- Persönliche Daten (Name, Alter, Telefonnummer, Beruf, Ausbildung, Sprache)
- Kontakte und Freunde (Adressbücher, Chat-Kontakte, Emailadressen)
- Zugriffsdaten (Login-Daten, abonnierte Dienste, Einstellungsparameter)
- Gerätedaten (Displayauflösung, Bandbreiten, vorhandene Software)
- Ortsdaten (Position, Bewegungsrichtung)
- Präferenzen (Einstellungsparameter, Interessengebiete)
- Richtlinien (Umgang mit persönlichen Daten, Umgang mit vertrauenswürdigen Quellen)
- Momentanes Ziel oder Aufgabe
- Wissensstand
- Browsing History (auch Bookmarks)
- Emotionale Befindlichkeit

Ein großes Problem der Benutzermodellierung stellt die Nutzerakzeptanz dar. Nicht jeder Nutzer ist bereit, seine Daten freiwillig und wahrheitsgemäß gegenüber dem

System Preis zu geben. Diese Tatsache kann zur Folge haben, dass Nutzer entweder falsche oder keine Angaben machen, was wiederum zu einer verminderten Qualität des Nutzermodells oder gar zum Fehlen der Datenbasis für einen späteren Filterprozess führen kann. Es ist daher in jedem Fall empfehlenswert, das Nutzerprofil soweit wie möglich durch automatisierte bzw. implizite Verfahren der Personalisierung zu füllen.

Darüber hinaus ist es in der Praxis notwendig, ein Benutzermodell so effizient wie möglich zu gestalten. Dazu werden häufig Stereotypen verwendet, um Nutzer in verschiedene Gruppen zu untergliedern. Diese Zuordnung erfolgt dabei aufgrund von vordefinierten Richtlinien [Meißner+07]. Erfüllt ein Nutzer eine Richtlinie, wird dieser den entsprechenden Stereotypen zugeordnet. Dabei ist nicht ausgeschlossen, dass ein Nutzer mehreren Stereotypen zugeordnet ist. Diese Form der Kategorisierung von Nutzern bietet den Vorteil einer effizienten Datenspeicherung und einer schnellen Adaption.

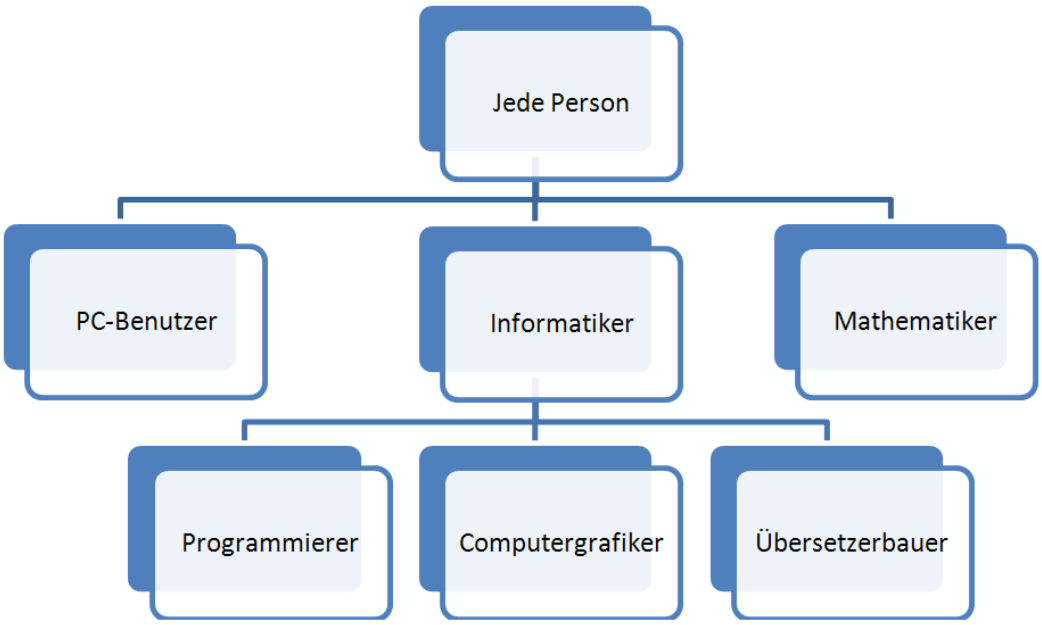

Abbildung 4: Beispiel einer Stereotypenhierarchie [Meißner+07]

Ist ein Benutzermodell erstellt und ist dieses in ein System integriert oder angebunden, kann es ebenfalls anhand anderer Merkmale klassifiziert werden. Die verschiedenen Ausprägungen sind nachfolgend aufgelistet [Meißner+07]:

- Long term / short term – Sessionübergreifend oder flüchtig
- Implizit / Explizit – Implizit: Benutzerinformationen sind Teil des Systems und können nur schwer verändert werden; Explizit: Benutzermodell ist vom Rest des Systems separiert
- Dynamisch / Statisch – Benutzermodell kann sich während der Sitzung ändern oder nicht

– Sichtbar / Unsichtbar – Benutzermodell ist für den Nutzer zugänglich und evtl. veränderbar oder nicht

Die Auswahl, welche Benutzermodellkategorie zur Anwendung kommt, hängt im Wesentlichen von den Anforderungen an das System ab bzw. welchen Zweck die Personalisierung und die Benutzermodellierung verfolgen. So ist das Benutzermodell des Onlineshops „Amazon“ als implizites, dynamisches und größtenteils unsichtbares „long term“-Benutzermodell einzuordnen. Dabei wurden diese Zuordnungen getroffen, da die Benutzerinformationen Teil des Systems sind, sich das Benutzermodell während einer Session verändern kann, der Nutzer lediglich seine personenbezogenen Daten einsehen kann, nicht aber die Informationen die für die inhaltsbasierende und kollaborative Filterung verwendet werden und sich die Personalisierung über einzelne Sitzungen hinaus erstreckt.

2.1.5 Filtermechanismen

In den vorangegangenen Abschnitten wurden Methoden zur Akquisition von Nutzerdaten bzw. Nutzermodellen vorgestellt. Mit diesen Daten als Grundlage können Verfahren zur Filterung von Inhalten umgesetzt werden. Die Motivation, ein solches System einzusetzen, ist relativ offensichtlich. Mit der immer größer werdenden Informationsflut, die in den meisten Fällen in ungefilterter Form nicht zu überschauen bzw. zu bewältigen ist, steigt auch die Nachfrage nach Möglichkeiten, diese Menge an Informationen durch automatische Verfahren einzugrenzen. Häufig werden derartige Filtermechanismen in Recommender-Systemen eingesetzt, um im Bereich des E-Commerce den Kunden möglichst präzise Vorschläge für interessante Produkte zu unterbreiten. Dabei wird versucht, anhand der Daten im Nutzerprofil eines Kunden auf den Interessenbereich des Kunden zu schließen, um ihm dann maßgeschneiderte Vorschläge für einzelne Produkte aus dem Gesamtangebot zu unterbreiten, die seinen Vorlieben entsprechen.
In den nachfolgenden Abschnitten werden einige Verfahren für eine solche automatisierte Filterung vorgestellt und die technischen Details näher erläutert.

2.1.5.1 Inhaltsbasierendes Filtern

Inhaltsbasierendes Filtern bzw. „Content-based-Filtering“ wird oft mit „Information-Retrieval“ bzw. „Information-Extraction“ in Verbindung gebracht. In beiden Bereichen werden Eigenschaften und Inhalte genutzt, um aus einer großen Menge von Ressourcen kleinere Mengen zu extrahieren. Dadurch lassen sich Erkenntnisse aus dem Bereich des „Information-Retrieval“ für das inhaltsbasierende Filtern nutzen [Lehm+04].

Beim inhaltsbasierenden Filtern wird versucht, in einer Menge von Ressourcen Ähnlichkeiten zwischen diesen festzustellen. Grundvoraussetzung hierfür ist eine möglichst genaue Beschreibung der einzelnen Ressourcen. Häufig wird dies durch gewichtete Schlüsselwörter, die den Ressourcen zugeordnet werden, realisiert (vgl. Abbildung 5):

Film	Drama	Liebe	Gewalt	Humor	Action
Sieben	5	1	10	1	5
Hannibal	4		9	2	3
Titanic	8	10	1		2

Abbildung 5: Klassifizierung von Ressourcen anhand von gewichteten Schlüsselwörtern [Lehm+04]

Die Ermittlung von Ähnlichkeiten kann mit zwei unterschiedlichen Ansätzen umgesetzt werden, „exact-match" und „best-match". Während beim „exact-match" nur Resultate geliefert werden, die der Anfrage exakt entsprechen, wird beim „best-match"-Verfahren versucht, die ähnlichsten Ressourcen zu finden. Diese werden dann nach dem ermittelten Ähnlichkeitsmaß sortiert und ausgegeben. Um diesen Ansatz zu verfolgen, existieren mehrere Verfahren für die Berechnung der Ähnlichkeit. Im Nachfolgenden werden zwei dieser Verfahren kurz erläutert.

Lineares Modell

Beim Verfahren des linearen Modells [Meißner+07] wird für jede Ressource durch eine Berechnungsvorschrift die Ähnlichkeit zur Referenzressource / Anfrage gebildet. Für die Berechnung der Ähnlichkeit wird die Summe von 1 bis zur Anzahl der Merkmale N aus dem Produkt des Gewichtes des Merkmals w_i und dem Wert für das Merkmal der Ressource $x_i(m)$, addiert mit der additiven Verschiebung b, errechnet:

$$r(m) = \sum_{i=1}^{N} w_i * x_i(m) + b$$

Vektorraum-Modell

Grundprinzip des Vektorraum-Modells ist es, Ressourcen und deren gewichtete Schlüsselwörter als Vektoren darzustellen. Dies geschieht in einem m- dimensionalen Raum, wobei m für die Anzahl der verwendeten Schlüsselwörter steht. Nutzerprofile bzw. die Anfragen, die sich aus den hinterlegten Daten im Nutzerprofil ergeben, werden ebenfalls als Vektoren in diesem m-dimensionalen Raum dargestellt. Die Ähnlichkeit zwischen einer Anfrage und einer Ressource ergibt sich aus dem Winkel zwischen den beiden Vektoren. Somit werden die Vektoren bei diesem Verfahren so gebildet, dass ein direkter Zusammenhang zwischen dem Winkel der Vektoren und der Relevanz entsteht.

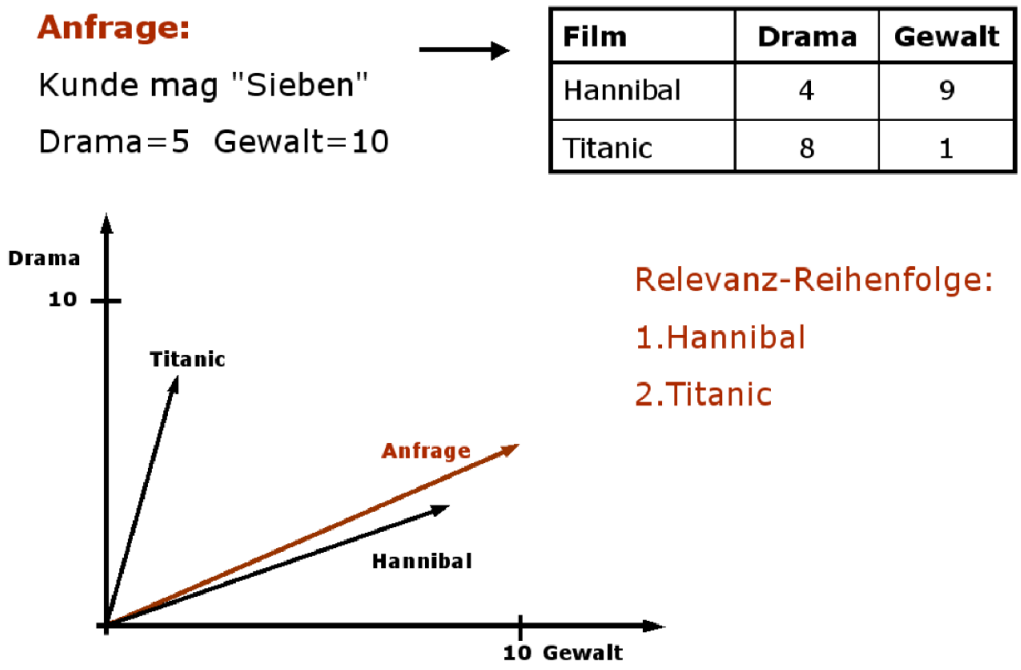

Abbildung 6: Beispiel Vektorraum-Modell [Lehm+04]

Die Berechnung des Ähnlichkeitsmaßes - u(c,s) - erfolgt dabei zwischen jeder Ressource (c) und jeder Anfrage (s) separat. Dabei wird folgende Berechnungsvorschrift [Tuzhilin] verwendet.

$$u(c,s) = \cos(\vec{w}_c, \vec{w}_s) = \frac{\vec{w}_c \cdot \vec{w}_s}{\|\vec{w}_c\|_2 \times \|\vec{w}_s\|_2} = \frac{\sum_{i=1}^{K} w_{i,c}\, w_{i,s}}{\sqrt{\sum_{i=1}^{K} w_{i,c}^2}\ \sqrt{\sum_{i=1}^{K} w_{i,s}^2}}$$

2.1.5.2 Kollaboratives Filtern

Im Gegensatz zum inhaltsbasierenden Filtern werden beim kollaborativen Filtern Ähnlichkeiten zwischen Nutzern gesucht. Dabei wird davon ausgegangen, dass Nutzer, die Schnittmengen bei ihren Präferenzen haben, ebenfalls noch nicht entdeckte Interessen in der Präferenzmenge des jeweilig anderen Nutzers haben.
In diesem Fall würde Nutzer A Vorschläge für Ressourcen aus der Interessenmenge von Nutzer B erhalten, die nicht Teil der Interessenschnittmenge von A und B sind. Dabei ist zu beachten, dass die Interessenschnittmenge möglichst groß sein sollte, um eine entsprechend hohe Ähnlichkeit abzuleiten.
Neben diesem relativ einfachen Prinzip zeichnet sich die kollaborative Filterung zudem durch einen relativ geringen Betreiberaufwand bei der Umsetzung eines solchen Systems aus. Die Nutzer des Systems stellen mit ihren Bewertungen der Ressourcen die notwendige Datenbasis für die Filterung. Die Errechnung von Ähnlichkeiten zwischen den Nutzern wird auf dieser Grundlage automatisch vom System vorgenommen.

Aus technischer Sicht werden hierzu zwei Verfahren zur Berechnung der Ähnlichkeiten unterschieden [Lehm+04]. Zum Einen das speicherbasierte Verfahren, das genutzt wird, wenn die Berechnung auf dem gesamten Datenbestand zur

Laufzeit ausgeführt werden soll und zum Anderen das modellbasierende Verfahren, das zum Einsatz kommt, wenn der Rechenaufwand für die Berechnung zur Laufzeit zu hoch ist.

Bei beiden Verfahren wird für die Darstellung der Nutzerprofile eine Datenmatrix V genutzt. Diese Matrix hat die Dimension $M \times N$, wobei M die Anzahl der Nutzer und N die Anzahl der Produkte im System ist [Lehm+04]. Die sich in dieser Matrix befindlichen Einträge $v_{i,j}$ repräsentieren jeweils eine Bewertung von einem Nutzer i zu einem Produkt j in einem Wertebereich von 0 bis t, wobei o ein noch nicht bewertetes Produkt symbolisiert.

$$V = (v_{i,j})_{M,N} = \begin{bmatrix} v_{1,1} & v_{1,j} & v_{1,N} \\ v_{i,1} & v_{i,j} & v_{i,N} \\ v_{M,1} & v_{M,j} & v_{M,N} \end{bmatrix} \in \{0 \dots t, o\}^{M \times N}$$

Speicherbasierende Verfahren

Wie bereits beschrieben, wird das speicherbasierende Verfahren zur Laufzeit angewendet und führt für jede Anfrage alle Operationen auf der gesamten Datenmatrix aus. Dabei wird zunächst die durchschnittlich abgegebene Bewertung des aktiven Nutzers ermittelt [Breese+98], wobei I_i die Anzahl der abgegebenen Bewertungen ist und $v_{i,j}$ die Einzelbewertung der jeweiligen Ressource aus der Datenmatrix widerspiegelt:

$$\bar{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j}$$

Der Wichtungswert $p_{a,j}$ des Nutzers a zur Ressource j ergibt sich dann aus der folgenden Berechnungsvorschrift [Breese+98]:

$$p_{a,j} = \bar{v}_a + k \sum_{i=1}^{n} w(a,i)(v_{i,j} - \bar{v}_i)$$

Bei der Bildung der Summe ist zu beachten, dass nur Nutzer in die Berechnung einfließen, von denen in der Datenmatrix Bewertungen vorliegen. Der Koeffizient w(a,i) gibt dabei die Ähnlichkeit des aktiven Nutzers a zu einem Nutzer i wieder und errechnet sich nach der Pearson Korrelation [Breese+98] wie folgt:

$$w(a,i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}}$$

Die Summen über j Ressourcen beziehen sich dabei nur auf Einträge in der Datenmatrix, bei denen die Nutzer a und i die Ressource j bewertet haben. Die folgende Abbildung 7 stellt neben einer exemplarischen Datenmatrix ebenfalls die zugehörige Ähnlichkeitstabelle dar:

	Shrek	IceAge	Rambo	Batman	StarWars
Nutzer1	5	-	1	2	5
Nutzer2	3	5	-	4	5
Nutzer3	3	2	4	4	3
Nutzer4	5	4	1	2	4
Nutzer5	4	3	2	-	5

	Nutzer2	Nutzer3	Nutzer4	Nutzer5
Nutzer1	0.00	-0.47	0.97	0.77
Nutzer2		0.85	-0.52	0.00
Nutzer3			-0.65	-0.18
Nutzer4				0.85

Abbildung 7: Datenmatrix und Ähnlichkeitstabelle [Lehm+04]

Ein solcher Algorithmus stellt naturgemäß hohe Anforderungen an die Rechenleistung des ausführenden Systems. Mit steigender Nutzer und Ressourcenzahl wird es immer schwieriger, ein solches Verfahren zur Laufzeit auszuführen. Deshalb wird bei höheren Nutzer- und Ressourcenzahlen dazu übergegangen, modellbasierende Verfahren zu verwenden.

Modellbasierende Verfahren

Der grundlegende Unterschied zwischen modellbasierenden Verfahren und speicherbasierenden Verfahren liegt darin, dass die Berechnungen im Fall der modellbasierenden Verfahren zur Erkennung von Ähnlichkeiten nicht online bzw. zur Laufzeit ausgeführt werden, sondern offline. Vorteil einer solchen Herangehensweise ist, dass bei Zugriffen auf das Modell zur Laufzeit keine Berechnungen auf der gesamten Datenmatrix ausgeführt werden müssen. Um dies zu realisieren wird zunächst eine Clusteranalyse durchgeführt [Lehm+04]. Dabei wird die Datenmatrix auf wenige Interpretationseinheiten bzw. Cluster reduziert. Diese werden so eingeteilt, dass in jedem Cluster eine bestimmte Gruppe von Nutzern abgelegt wird, die in ihren abgegebenen Bewertungen eine höchstmögliche Ähnlichkeit aufweisen. Ist die Einordnung der Nutzer in die jeweiligen Cluster abgeschlossen, werden für jeden Cluster repräsentative Nutzer ausgewählt. Das kann entweder ein realer Nutzer aus dem Cluster sein oder ein virtueller Nutzer, der anhand der Durchschnittswerte innerhalb des Clusters errechnet wird.

Wird eine Anfrage an das System gestellt, muss im Gegensatz zum speicherbasierenden Verfahren nicht die gesamte Datenmatrix zur Berechnung herangezogen werden, sondern nur noch die Repräsentanten der Cluster. Diese Tatsache birgt einen deutlichen Geschwindigkeitsvorteil gegenüber einer rein speicherbasierenden Lösung. Durch die Reduzierung auf ein Modell und die Bildung von Repräsentanten für verschiedene Cluster kommt es allerdings auch zu Informationsverlusten, die sich negativ auf die Qualität der Empfehlungen auswirken können. Darüber hinaus muss im Fall einer Neuanmeldung eines Nutzers oder einer abgegebenen Bewertung durch einen Nutzer das Modell auf seine Gültigkeit geprüft werden und gegebenenfalls neu berechnet werden.

2.1.5.3 Item-based-Collaborative-Filtering

Das „Item-based-Collaborative-Filtering“ verfolgt das Ziel, die beiden Verfahren inhaltsbasierendes und kollaboratives Filtern zu verbinden, um die Vorteile beider Verfahren zu nutzen und einige Nachteile zu beheben.

Das Verfahren untergliedert sich, ähnlich wie das modelbasierte kollaborative Filtern, in eine Online- und eine Offline-Phase. In der Offline-Phase werden hier jedoch nicht ähnliche Nutzer gesucht, sondern Ähnlichkeiten zwischen Ressourcen errechnet. Die ermittelten Ähnlichkeiten werden zu jeder Ressource in einer separaten „similar items table“ gespeichert. Zur Laufzeit werden diese Tabellen genutzt, um Ressourcen anzuzeigen, die der Nutzer neben der augenblicklich betrachteten Ressource interessant finden könnte. Für die Errechnung der Ähnlichkeit zweier Ressourcen werden bei diesem Verfahren zunächst alle Nutzer gesucht, die beide Ressourcen bewertet haben (vgl. Abbildung 8). Diese Bewertungen werden mit Hilfe eines geeigneten Verfahrens, wie beispielsweise dem Vektorraum-Modell, verglichen und der errechnete Ähnlichkeitswert wird in die „similar items table“ eingetragen.

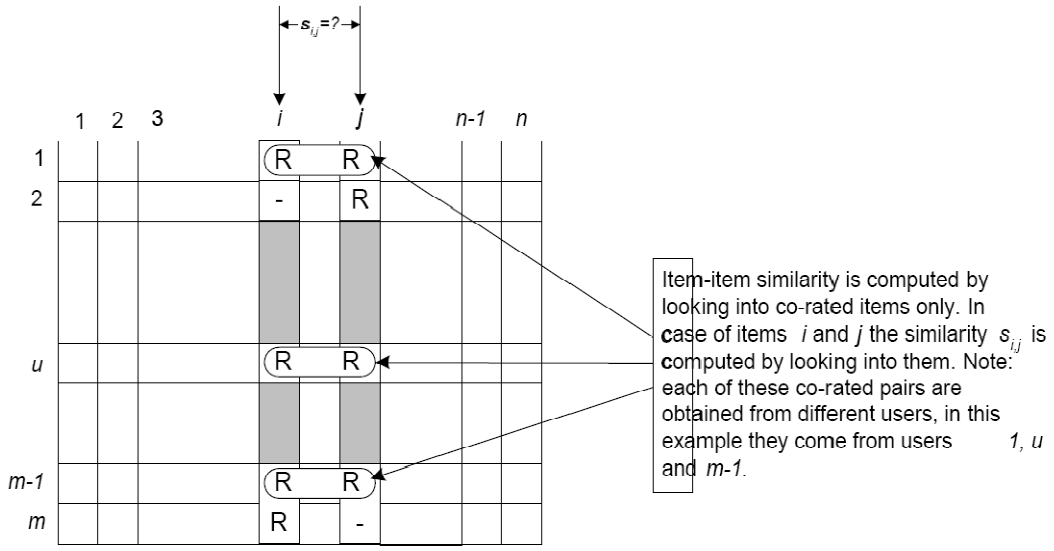

Abbildung 8: Isolation der durch zwei Nutzer bewerteten Ressourcen [Sarwar]

2.1.5.4 Probleme von Filtermechanismen

Wird in ein System, das kollaboratives Filtern verwendet, eine neue Ressource eingefügt, wird deutlich, dass dieses Verfahren eine grundlegende Schwäche hat. Diese neue Ressource hat üblicherweise keine oder nur wenige Bewertungen und wird aufgrund dieser Tatsache bei einem Filterprozess nicht oder nur sehr selten zur Ausgabemenge hinzugefügt. Dies hat wiederum zur Folge, dass Nutzer nicht auf die Ressource aufmerksam werden und demnach auch nicht in die Lage kommen, diese Ressource zu bewerten. Ein Lösungsansatz für dieses „New-Item“-Problem könnte z.B. die künstliche Bewertung neuer Ressourcen sein, um zu vermeiden, dass diese Ressourcen grundsätzlich herausgefiltert werden. Zudem könnte ein „Ageing-Faktor“ für neue Ressourcen in die Bewertung einbezogen werden, der zum Zeitpunkt der Aufnahme der Ressource in das System dafür sorgt, dass diese Ressource hoch bewertet wird. Im Laufe der Zeit könnte dieser Faktor kontinuierlich reduziert werden, so dass die Ressource wieder vollständig anhand der abgegebenen Bewertungen gemessen wird.

Ein anderes Problem, das sowohl das kollaborative als auch das inhaltsbasierte Filtern betrifft, ist das „New-User"-Problem [MELVILLE+02]. Ein Nutzer, der noch keine Bewertung abgegeben hat, ist nur schwer zu kategorisieren. Eine Personalisierung des Systems kann ebenso nur erfolgen, wenn die Präferenzen des Nutzers in irgendeiner Form bekannt und für das System zugänglich sind. Um dieses Problem zu umgehen, kann für den Zeitraum, in dem der neue Nutzer noch keine Bewertungen abgegeben hat, darauf zurückgegriffen werden, Empfehlungen zu geben, die im Durchschnitt am häufigsten an die Nutzer des Systems ausgegeben werden.

Werden die bereits vorgestellten Verfahren zur Filterung verglichen, ergeben sich unterschiedliche Vor- und Nachteile. Die nachfolgende Tabelle 1 gibt einen zusammenfassenden Überblick über die wichtigsten Vorzüge und Kritikpunkte:

	Vorteile	Nachteile
Inhaltsbasierendes Filtern	– Empfehlungen sind für den Betreiber leicht nachvollziehbar – Auch bei geringen Nutzerzahlen einsetzbar – In vielen Bereichen sind Kategorien schon gegeben	– Gut beschriebene Ressourcen sind Voraussetzung – Merkmalsextraktion oft schwierig – Keine Erfassung subjektiver Bewertungen
Kollaboratives Filtern	– Geringer Initialaufwand – Subjektive Bewertungen werden berücksichtigt – Flexibel einsetzbar – Hochwertige Empfehlungen (cross Genre)	– Minimum an Benutzerprofilen notwendig – Objekte müssen bewertet sein („new Item"- Problem) – Inhalte werden nicht berücksichtigt

Tabelle 1: Filtermechanismen Vor- und Nachteile [Meißner+07]

2.2 Publish / Subscribe Systeme

Die hier beschriebenen Dienste werden als „Publish-/Subscribe"- Angebote bezeichnet und können somit auch als solche kategorisiert werden. In der Forschung gibt es viele Ansätze wie z.B. von Patrick Eugster [Eugster+03], um solche Systeme zu klassifizieren. Eine sinnvolle Einteilung kann allerdings nur mit Blick auf die zu betrachtenden Verfahren gewählt werden. Deshalb wurde hier die Klassifizierung von Prof. Dr. Alexander Schill [Schill+08] verwendet, die „Publish / Subscribe"-Systeme in „Distributor Aware Subscriptions" und „User Client Aware Subscriptions" untergliedert. Sie ist darauf ausgerichtet, Systeme zu beschreiben und

einzuordnen, die für das publizieren und abonnieren von Informationen bzw. Nachrichten entwickelt wurden.

2.2.1 Distributor Aware Subscribtions

Eine sehr häufig auftretende Form von „Publish-/Subscribe"- Angeboten sind „Distributor Aware Subscriptions". Sie zeichnen sich durch folgende Merkmale [Schill+08] aus:

- Der Nutzer des Dienstes benötigt einen eindeutigen Identifier um ihn permanent im Netzwerk adressierbar zu machen
- Der Dienstanbieter speichert diesen Identifier für jeden Nutzer
- Wenn neue Informationen für den Nutzer vorliegen wird er automatisch darüber informiert und bekommt unter Umständen eine direkte Kopie dieser Information

Beispiele für solche Angebote sind Mailinglisten, Microblogging und „Instant Messaging"- Dienste.

2.2.1.1 Microblogging

Im Gegensatz zu den klassischen „Publish-/Subscribe"-Systemen stellt das Microblogging eine vergleichsweise neue Form der Kommunikation dar. Es bietet Nutzern eine einfachere und schnellere Form der Informationsweitergabe, als dies von anderen „Publish-/Subscribe"- Systemen bekannt ist. Dabei können Nutzer kurze Statusmeldungen über sich selbst oder allgemeine Informationen anderen Nutzern zur Verfügung stellen [SMITH+09]. Diese Statusmeldungen bzw. Nachrichten haben meist eine max. Länge von 140 Zeichen. Damit soll sichergestellt werden, dass eine Nachricht einen sehr hohen Informationsgehalt im Vergleich zu ihrer Länge aufweist und dass auf ausschweifende Umschreibungen verzichtet werden muss. Darüber hinaus führt diese Restriktion zu einer Senkung des Bandbreitenverbrauchs der Anbieter und zu einer effizienteren Verarbeitung der Inhalte.

Ein Nutzer eines Microblogging-Netzwerkes hat die Möglichkeit, sich für eine sog. „Follower-Liste" eines anderen Nutzers zu registrieren, um automatisch die aktuellen Mitteilungen dieses Nutzers zu empfangen. Gleichwohl hat dieser Nutzer auch seine eigene „Follower-Liste", in der sich Nutzer registrieren können, die ein Interesse an den Nachrichten des Nutzers haben. Durch dieses Konzept entstehen Netzwerke, die nicht allein auf einer privaten Verbindung beruhen, wie dies aus Angeboten wie beispielsweise „StudiVZ" oder „Facebook" bekannt ist. Häufig werden Microblogging-Netzwerke auch eingesetzt, um über Meinungen und Gedanken von Meinungsführern und fachlichen Koryphäen informiert zu werden. Diese Tatsache macht deutlich, dass Microblogging-Netzwerke für private und geschäftliche Zwecke eingesetzt werden können.

In der Literatur werden Microblogging-Netzwerke häufig als soziale Netzwerke bezeichnet. Dennoch unterscheiden sie sich in ihrer Ausrichtung von etablierten

Angeboten sozialer Netzwerke wie „Xing“ oder „StudiVZ“. Der Kerngedanke des Microblogging ist die Kommunikation und der Informationsaustausch, während für Netzwerke wie beispielsweise „StudiVZ“ das Bilden von sozialen Kontakten innerhalb des Netzwerkes die Grundlage für die vorgesehene Nutzung ist. Daher ist Microblogging weniger als soziales Netzwerk zu sehen, sondern mehr als Kommunikationsmittel zwischen Nutzern und deren „Follower“.

Folgt ein Nutzer einer größeren Anzahl von anderen Nutzern in einem Microblogging-Netzwerk, kann dies schnell zu einem unüberschaubaren Informationsfluss werden. Um dieser Informationsmenge Herr zu werden, werden in den meisten Microblogging-Netzwerken einzelne Schlüsselwörter in Nachrichten durch Hashtags mit einem vorgestellten „#“ Zeichen markiert, so dass eine Kategorisierung nach Schlüsselwörtern sowie eine einfache Filterung nach diesen „Tags“ möglich wird. Mit Hilfe von Hashtags ist es jedoch nicht möglich, komplexere semantische Ausdrücke zu formulieren. Wird beispielsweise das Wort „Paris“ als Hashtag ausgedrückt, ist es nicht möglich, zu unterscheiden, ob es sich um einen Personennamen oder um die Stadt Paris handelt. Um derartigen Problemen aus dem Weg zu gehen, wäre der Einsatz von semantischen Kategorien notwendig. Ein Beispiel hierfür liefert Alexandre Passant [Passant+08] und beschreibt, dass die Microblog-Nachricht:

„Visiting #Eifel_Tower in #Paris“

durch die Verwendung von vordefinierten Präfixen wie „geo“ (Geonames) [GEONAMES] und „dbp“ (DBpedia) [AUER+07] genauer ausgedrückt werden kann.

“Visiting #dbp:Eifel_Tower in #geo:Paris_France”

Somit kann die elektronische Maschinenverarbeitung deutlich erleichtert und die Wiederverwendbarkeit erhöht werden. Eine solche Nachricht könnte beispielsweise dynamisch in ein onlinebasiertes Kartensystem, wie etwa „Google Maps“, eingebunden werden, um die Attraktivität dieses Dienstes zu steigern oder um Verknüpfungen mit anderen Nachrichten, die die gleichen Tags verwenden, herzustellen.

Ein weiteres im Microblogging verwendetes Konzept ergibt sich aus der Vorgabe, dass Nachrichten nur eine maximale Länge von meist 140 Zeichen umfassen dürfen. Auf Grund dieser Restriktion und der Tatsache, dass URLs im Web oftmals deutlich länger als 24 Zeichen sind, wird die Länge von URLs in einer Nachricht minimiert. Hierfür wird jede URL, die einer Nachricht hinzugefügt wird, einem Mapping unterzogen, so dass die ursprüngliche URL auf eine URL der Form „http://tinyurl.com/xxxxx“ oder „http://ur1.ca/xxx“ abgebildet wird [TINYURL] [UR1]. Diese „kurze URL“ dient dem Zweck, bei einem Zugriff auf die ursprüngliche URL weiterzuleiten und stellt sicher, dass in Microblogging-Nachrichten URLs von beliebiger Länge verwendet werden können. Dabei ist

anzumerken, dass bis zum heutigen Zeitpunkt eine Reihe unterschiedlicher Anbieter zur Generierung von verkürzten URLs existiert.

Es gibt bereits eine Vielzahl von Microblogging-Netzwerken, die jedem Nutzer in der Regel kostenfrei zur Verfügung stehen. Als der bekannteste Vertreter gilt der Dienst „Twitter“, der seit März 2006 verfügbar ist. Neben ihm haben sich bereits weitere Dienste etabliert. Als Beispiele sind an dieser Stelle „Jaiku“, „Pownce“ oder „Friendfeed“ zu nennen. Für all diese Dienste stehen ebenfalls Client-Programme zur Verfügung, die eine Teilnahme am jeweiligen Netzwerk ermöglichen. In der Regel nutzen diese Programme eine vom Dienstanbieter zur Verfügung gestellte „API“, um Zugriff auf die Daten des Netzwerkes zu erhalten. Diese „API“ basiert bei „Twitter“ auf einem „REST“ (Representational State Transfer)-Interface und stellt Funktionen für die Verwendung und Steuerung eines „Twitter“-Accounts sowie Funktionen für die Suche im „Twitter“-Netzwerk bereit.

Um einen Eindruck von der Verbreitung und Wachstumsgeschwindigkeit des „Twitter“-Netzwerkes zu bekommen, wurde eine Studie [Java+07] zur Nutzung und Verbreitung von „Twitter“ erstellt. Dabei ist anzumerken, dass diese Studie in einem Zeitraum starken Wachstums erstellt wurde und lediglich auf die Akzeptanz derartiger Angebote hinweist, jedoch nicht auf das Wachstum von Microblogging-Diensten im Allgemeinen. Aufgrund der Tatsache, dass „Twitter“ selbst keine Angaben zu Nutzerzahlen oder anderen statistischen Daten macht, wurde die „Twitter-API“ genutzt, um anhand der maximalen Werte von „User-IDs“ und „Post-IDs“ diese Daten extern zu erheben. Betrachtet wurde ein zweimonatiger Zeitraum von Anfang April 2007 bis Ende Mai 2007, in dem kontinuierlich Daten erhoben wurden.

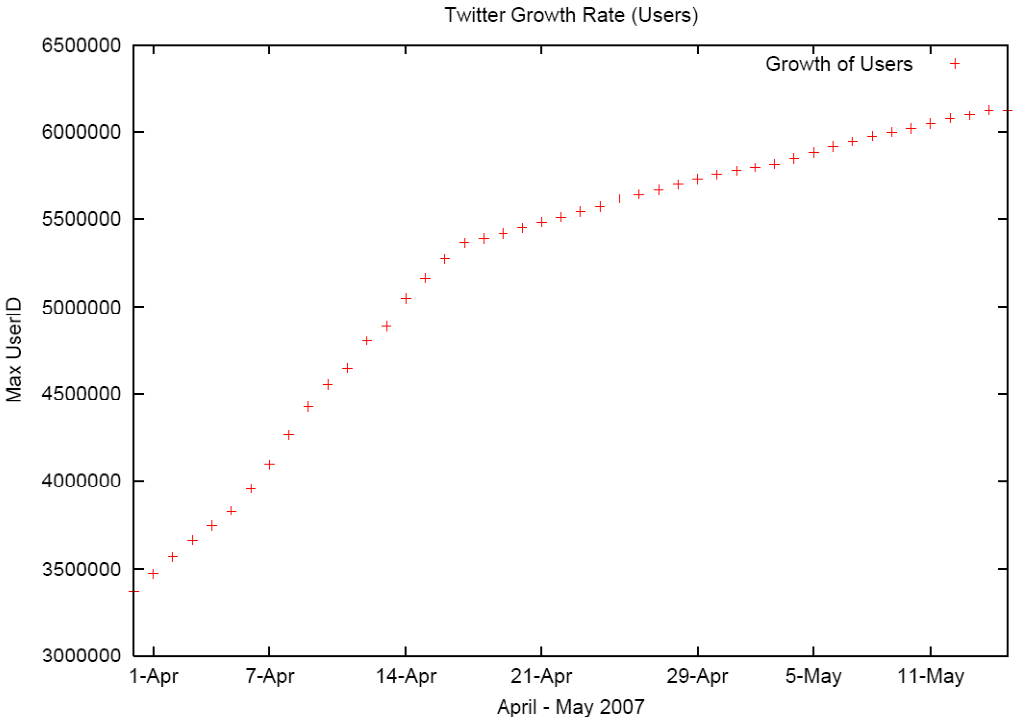

Abbildung 9: Wachstum „Twitter“-Nutzerzahlen April-Mai 2007 [Java+07]

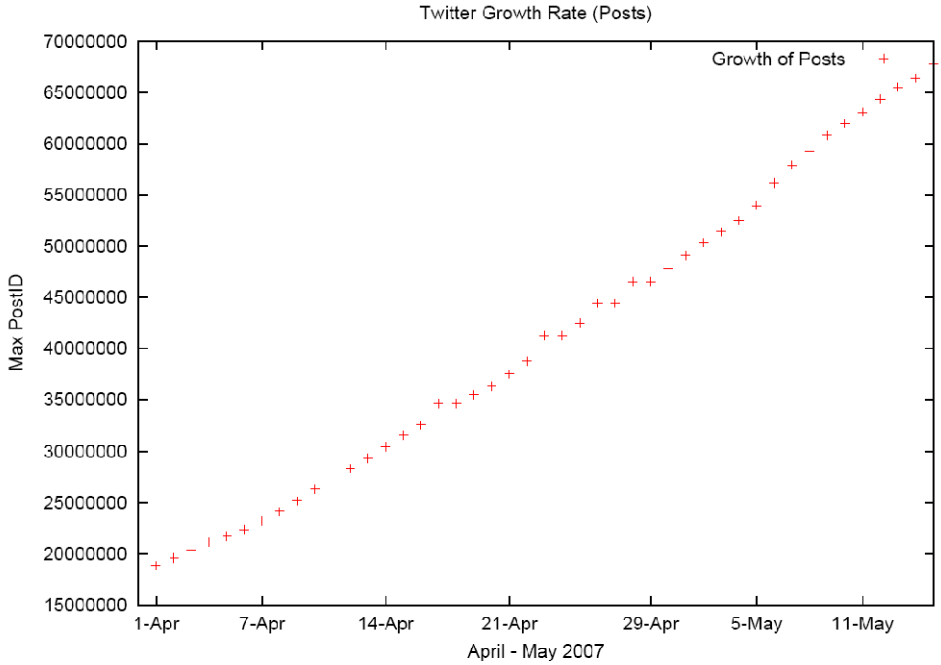

Abbildung 10: Wachstum „Twitter"-Posts April-Mai 2007 [Java+07]

Aus diesen Ergebnissen wird deutlich, mit welcher Geschwindigkeit sich das „Twitter"-Netzwerk ausdehnt und wie häufig diese Form der Kommunikation genutzt wird. Dabei ist es interessant zu analysieren, ob diese Nutzung geografisch begrenzt ist oder ob „Twitter" auf der Welt gleichverteilt genutzt wird. Auch hierzu wurden von der Studie [Java+07] Ergebnisse geliefert. Dabei wurde deutlich, dass „Twitter" zu einem großen Anteil in den USA, aber auch in Europa und Asien (hauptsächlich in Japan) genutzt wird. Um diese Ergebnisse zu ermitteln wurde eine Gruppe von 78000 Nutzern aus dem „Twitter"-Netzwerk betrachtet und dahingehend überprüft, ob sie Angaben zu ihrer Zeitzone sowie zu ihrem Aufenthaltsort gemacht haben. Die gewonnenen Daten wurden unter Zuhilfenahme der „Yahoo"-„Geocoding-API" analysiert und die Nutzer den entsprechenden Kontinenten bzw. Regionen zugeordnet.

Kontinent / Region	Anzahl der Nutzer
Nordamerika	21064 (27,7%)
Europa	7442 (9,8%)
Asien	6753 (8,9%)
Ozeanien	910 (1,2%)
Südamerika	816 (1,1%)
Afrika	120 (0,16%)
Andere	78 (0,1%)
Unbekannt	38994 (51,2 %)

Tabelle 2: „Twitter" Nutzerzahlen je Kontinent [Java+07]

Neben der proprietären Entwicklung des „Twitter"-Netzwerkes gibt es ebenfalls Lösungen auf „Open Source"- Basis. An dieser Stelle ist der „Open Microblogging

Standard“ zu nennen, der in der Microbloggingplattform „Laconica“ Anwendung findet und bereits von vielen Diensten genutzt wird [LACONICA].

Zudem ist Microblogging inzwischen nicht nur ein Thema für private Anwender. Auch für den Einsatz in Unternehmen gibt es inzwischen Lösungen. Ein Dienst, der sich auf die Nutzung in Unternehmen ausgerichtet hat, ist „yammer.com“. Dieser Dienst ist dabei nicht als Lösung für das Intranet konzipiert, sondern ähnelt in weiten Teilen den Diensten aus dem privaten Bereich. Der wesentliche Unterschied besteht darin, dass Nutzer je nach dem Domain-Teil ihrer Email-Adresse in separate Gruppen eingeteilt werden, die untereinander keine Zugriffsfunktionen haben. Auf diese Weise erhält jede Firma ein eigenes Microblogging-Netzwerk.

Exkurs: Communote

Eine weiterreichende Lösung für den unternehmensinternen Einsatz von Microblogging bietet die Firma „Communardo“ mit dem Produkt „Communote“. Ziel von „Communote“ ist dabei, einen Informationsaustausch zwischen Mitarbeitern über Mikroinhalte anzubieten, aber keine öffentliche Microblogging-Plattform für die Interaktion mit Kunden oder externen Personen zu stellen.

Motivation für die Entwicklung eines solchen Kommunikationsmittels sind die Defizite herkömmlicher Microblogging-Lösungen hinsichtlich eines unternehmensinternen Einsatzes. Eine „Enterprise Microblogging“- bzw. „Microsharing“- Lösung sollte weniger auf die Personen im Netzwerk zentriert sein, sondern auf die Themen, zu denen der Informationsaustausch erfolgen soll [Hauß+08]. In diesem Zusammenhang ist es fraglich, ob die Restriktion einer Nachricht auf eine maximale Länge von 140 Zeichen sinnvoll ist, da bei einer Kommunikation zu fachlichen Themen oft komplexe Zusammenhänge dargelegt werden müssen, die nur in den seltensten Fällen in 140 Zeichen ausgedrückt werden können. Zudem sollte eine solche Plattform auch eine Dokumentationsfunktion erfüllen, die es erlaubt, die Kommunikationshistorie persistent zu speichern, damit Entscheidungen und Gedanken über einen längeren Zeitraum protokolliert und zugehörige Dokumente archiviert werden können. Diese Inhalte sind oftmals nicht für die Öffentlichkeit bestimmt und stellen somit hohe Anforderungen an die Sicherheit und Vertraulichkeit der Plattform.

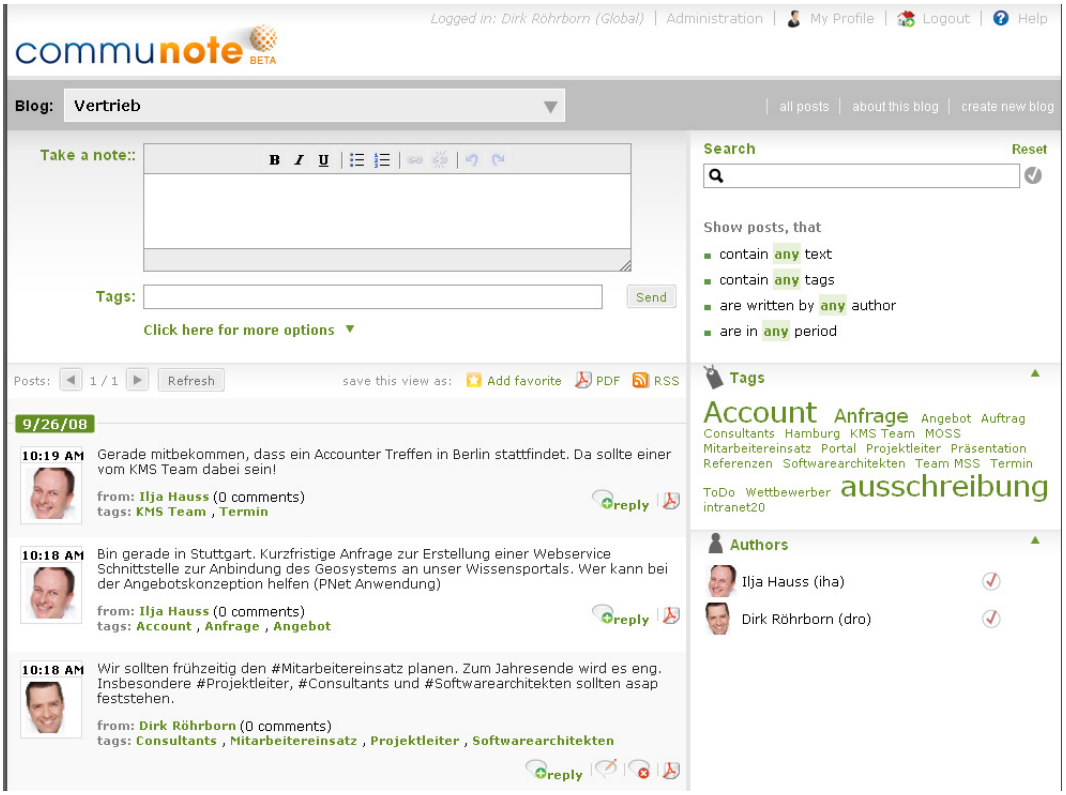

Abbildung 11: Screenshot Communote [Hauß+08]

„Communote“ greift diese Anforderungen auf und bietet somit eine vollständige Lösung für den Einsatz von Microblogging im Unternehmen. Erweitert werden diese Anforderungen durch eine gut strukturierte „AJAX“-Weboberfläche. Diese ermöglicht es, Nutzern mit Hilfe von Hashtags oder durch manuelles „Tagging“ in der Eingabemaske Schlüsselwörter zu markieren. Die angegebenen Schlüsselwörter werden in einer „Tag-Cloud“ aufgelistet und je nach der Häufigkeit ihres Auftretens in ihrer Schriftgröße variiert. Neben dem Hinzufügen von Schlüsselwörtern ist es ebenfalls möglich, die eingegebene Notiz an Personen innerhalb des Netzwerkes zu adressieren. Diese werden in einem solchen Fall über die hinterlegte Email-Adresse über diese Nachricht informiert. Darüber hinaus wird eine umfangreiche Suchfunktion angeboten, die die Inhalte der Plattform nach den gewünschten Schlüsselwörtern durchsucht. Dabei ist es möglich, die Suche auf einen Zeitraum einzugrenzen und nur Nachrichten anzuzeigen, die an die angemeldete Person gerichtet sind oder von einem bestimmten Autor verfasst wurden. Diese Filterkriterien lassen sich beliebig verbinden und variieren, um eine möglichst genaue Suchanfrage ausdrücken zu können. Darüber hinaus können die Ergebnisse einer Filtereinstellung als „RSS-Feed“ abonniert werden, um auch über einen längeren Zeitraum über relevante Einträge informiert zu werden.

Als zusätzliches Feature können die Blogs in „Communote“ auch per Email mit neuen „Posts“ gefüllt werden. Darüber hinaus wird derzeit auch an weiteren Zugriffsmöglichkeiten gearbeitet, die es in Zukunft ermöglichen sollen, via „XMPP“ und SMS auf die Blogs zuzugreifen. Eine Client Applikation für mobile Endgeräte, wie PDAs und Blackberrys, ist seit Anfang 2009 verfügbar.

Zusammenfassend ist festzustellen, dass „Communote“ eine durchdachte Microblogging- / Microsharing-Plattform für unternehmensinterne Zwecke bietet. Wünschenswert ist eine Erweiterung des Systems zu einer Kommunikationsplattform, die mehrere Nachrichtenströme aus verschiedenen Quellen bündelt und auf eine einheitliche Art und Weise präsentiert.

2.2.1.2 XMPP

Das „Extensible Messaging and Presence Protocol“ (XMPP) wurde mit der Zielstellung entwickelt, „Instant-Messaging-Systeme“ zu dezentralisieren, interoperabel, sicher und erweiterbar zu machen [XMPP]. Dabei ist ein XMPP-Netzwerk als System mehrerer verteilter Server zu sehen, die wiederum eine bestimmte Anzahl von Clients verwalten. Jeder Teilnehmer eines XMPP-Netzwerkes wird über einen „Jabber Identifier“ (JID) adressiert und hat die Möglichkeit, seinem Account mehrere Ressourcen hinzuzufügen. Dadurch wird es einem Nutzer ermöglicht, von seinem PC und auch von beispielsweise seinem mobilen Endgerät mit einer JID am Netzwerk teilzunehmen.

```
JID = [ user "@" ] domain [ "/" resource ]
```

Aufbau eines Jabber Identifiers [XMPP]

Nach der Registrierung bei einem Server wird dem Nutzer eine JID in der Form „nutzer@server.com“ zugewiesen. Verbindet sich der Nutzer unter Verwendung dieser JID mit dem Server, wird die JID um einen Ressourcenteil erweitert, um das verwendete Endgerät zu adressieren. Zu jedem Nutzer wird serverseitig eine Kontakteliste gespeichert. Dieses sog. „Roster“ enthält neben den Kontakten auch Präsenzinformationen über diese Kontakte.

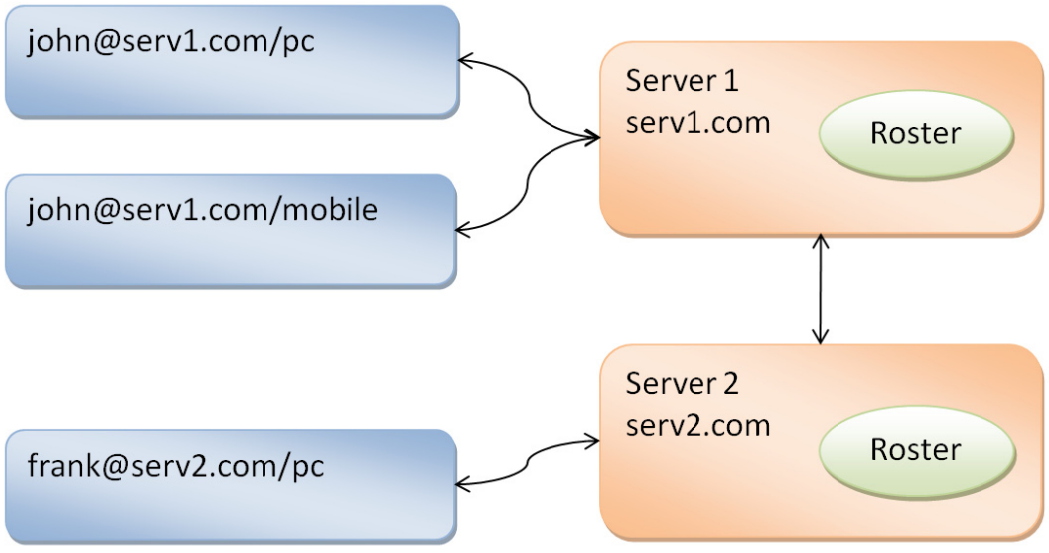

Abbildung 12: Aufbau eines XMPP-Netzwerkes [XMPP]

Nachdem eine „TCP“- Verbindung zwischen Client und Server aufgebaut wurde und innerhalb eines „XML-Streams“ Mechanismen zur Authentifikation und

Verschlüsselung abgearbeitet wurden, wird ein weiterer „XML-Stream“ geöffnet, der für die eigentliche Datenübertragung genutzt wird. Dieser „XML-Stream“ kapselt die Übertragung von „XML-Stanzas“, die sich in drei Typen untergliedern lassen [Schill+08]:

Message Stanzas
Ein „Message-Stanza“ beinhaltet eine Nachricht für einen bestimmten Empfänger:

```
<message to="frank@serv2.com" type="chat">
<body>Eine Nachricht für Frank</body>
<message>
```

Presence Stanzas
Ein „Presence-Stanza“ kapselt Präsenzinformationen über einen Nutzer. Diese werden vom Server verarbeitet und in das „Roster“ eingetragen:

```
<presence>
<show>dnd</show>
<status>Ich bin nicht erreichbar und möchte nicht gestört werden!</status>
</presence>
```

iq Stanzas
Ein „iq-Stanza“ wird benutzt, wenn bestimmte Verwaltungsinformationen angefordert oder gesetzt werden sollen:

```
<iq id="roster0" type="get">
<query xmlns="jabber:iq:roster"/>
</iq>
```

Diese Elemente werden je nach Bedarf in den „XML-Stream“ geschrieben und übertragen. Der „XML-Stream“ für die Datenübertragung bleibt dabei solange geöffnet, bis die Sitzung des Nutzers beendet wird. In diesem Fall wird der XML-Datenstrom mit dem schließenden Tag „</steam>“ geschlossen.

2.2.2 User Client Aware Subscriptions

Im Gegensatz zu den „Distributor Aware Subscriptions“ erfüllen „User Client Aware Subscriptions“ keine der unter 2.2.1 genannten Merkmale. Darüber hinaus definieren sie sich durch folgende Punkte [Schill+08]:

– Nutzer verwenden eine Software, in der sie jede Datenquelle angeben müssen, die sie abonnieren möchten

– Diese Software wird nicht automatisch über neue Informationen benachrichtigt, sondern überprüft in regelmäßigen Abständen, ob neue Inhalte in den angegebenen Datenquellen verfügbar sind

Beispiele für derartige Angebote sind „Web-Feeds“ und „Newsgroups/Usenet“.

2.2.2.1 Web Feeds

„Web Feeds“ sind im Allgemeinen XML-Dateien und beinhalten eine Übersicht bzw. Zusammenfassung der Inhalte einer Informationsquelle. Üblicherweise werden diese Dateien von einem Webserver bereitgestellt und dynamisch aus den Inhalten der Informationsquelle generiert. Anschließend werden diese Dateien mittels HTTP an den Client übermittelt und dort von einem „Feed-Reader“ gelesen (Abbildung 13).

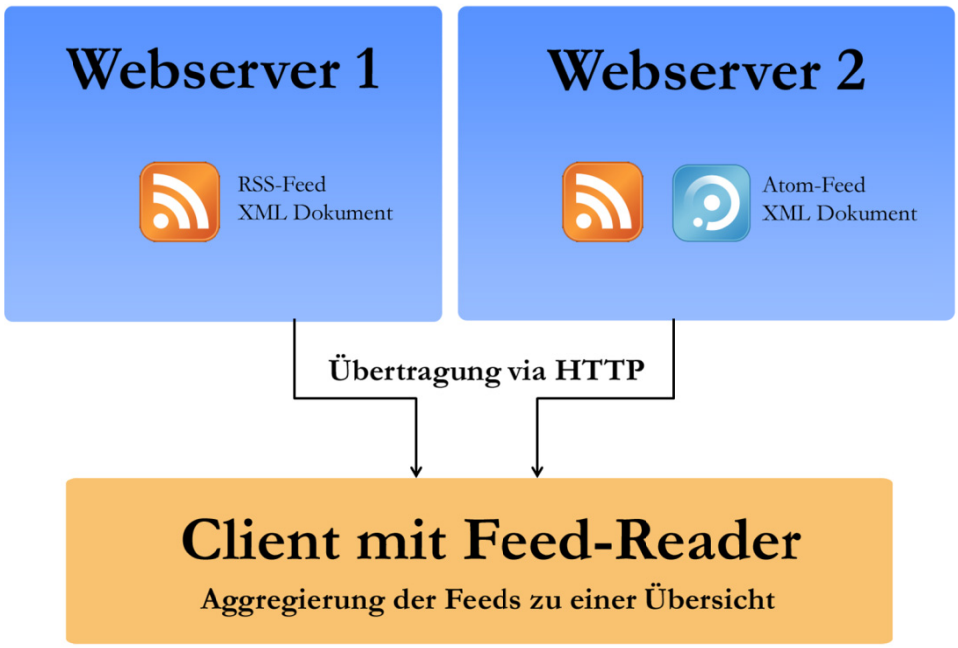

Abbildung 13: Bereitstellung und Übertragung von Feeds

„Feeds“ folgen einer vorgegebenen Struktur, die sich aus dem verwendeten „Feed-Standard“ ergibt. Am weitesten verbreitet ist hierbei der „RSS-Standard“, der sich in die Versionen 0.9, 0.9x, 1.0 und 2.0 untergliedern lässt. Dabei steht die Abkürzung „RSS“ in Verbindung mit jeder Versionsnummer für eine unterschiedliche Bezeichnung [RSS-Specification] [WIKI-01].

Standard	Name
RSS 0.9x	Rich Site Summary
RSS 0.9 & 1.0	RDF Site Summary
RSS 2.0	Really Simple Syndication

Tabelle 3: RSS-Standards

Diese Uneinheitlichkeit der Standards ist das Ergebnis paralleler Entwicklungen durch verschiedene Organisationen.
Die erste Form einer XML-basierten Webseitenzusammenfassung wurde bereits 1997 mit dem „Scripting News Format“ veröffentlicht. Mit Hilfe dieses Formates war es möglich, eine Informationsquelle durch beispielsweise Autorenangabe, Sprache und Publikationsdatum näher zu beschreiben [SCRIPTING-COM]. Darüber hinaus konnten mehrere Einträge eingefügt werden, die Nachrichten auf der referenzierten Seite repräsentierten und dadurch eine Übersicht über die auf der Seite verfügbaren Artikel lieferten. Dieses Format wurde 1999 von „Netscape“ aufgegriffen und zur RSS-Version 0.90 weiterentwickelt. Der wesentliche Unterschied bestand darin, dass RSS 0.90 die Angabe von Kanälen und deren untergeordneten Einträgen ermöglichte. Ebenfalls im Jahr 1999 übernahm die Firma „Userland“ die Entwicklung des RSS-Standards von „Netscape“. In den darauf folgenden Versionen von RSS wurde viel Wert auf die Möglichkeit gelegt, mehr Metadaten zu den einzelnen Elementen der Kanäle hinzufügen zu können. Heute hat sich RSS bis zur Version 2.0 weiterentwickelt, welche zugleich den Quasi-Standard im heutigen Web darstellt. Trotz dieser langjährigen Entwicklung und weiten Verbreitung, wird RSS oft aufgrund einiger Nachteile (vgl. Tabelle 4) bemängelt. Aus diesen Gründen wurde durch die „IETF“ ein weiterer Standard entwickelt und im Dezember 2005 als RFC-4287 mit dem Namen „Atom“ veröffentlicht [IETF]. Einige wesentliche Änderungen und somit auch die Vorteile des Atom-Standards gegenüber den RSS-Standards sind in der nachfolgenden Tabelle 4 aufgeführt:

Kriterium	RSS 2.0	Atom
Einbinden von XHTML Inhalten	Nur „escaped“ HTML; es ist nicht möglich den Inhalt eines Tags näher zu beschreiben	Textkonstrukte innerhalb von Tags können näher definiert werden. Bsp. type=“text“ oder type=“xhtml“
Podcasting	Möglich durch „enclosure“-Element. Jedoch nur ein Element pro „item“- Element erlaubt	Dateien können durch das „link“- Element referenziert werden. Auch mehrere Elemente möglich
Datenformate + Typen	Maximal eine externe Referenz pro Eintrag. Escaped HTML möglich.	Beliebig viele Referenzen auf externe Inhalte. Interner Inhalt kann reiner Text, escaped HTML, XHTML, anderes XML-Vokabular oder base64 kodierte Binärdaten sein
Digitale Signaturen und Verschlüsselung	Durch Signierung und Verschlüsselung von Webinhalten	Umfasst Regeln für XML-Verschlüsselung und digitaler XML-Signaturen

Tabelle 4: Vergleich RSS 2.0 und Atom [Schill+08] [MEIERT]

RSS und Atom sind für die gleichen Anwendungsgebiete ausgelegt, dennoch unterscheiden sie sich in einigen wesentlichen Punkten in Ihrer Struktur. Die Atom-Spezifikation beinhaltet 21 Elemente und weist gegenüber RSS 2.0 mit 30 Elementen eine niedrigere Anzahl auf. Diese Einsparung kann zum Einen auf die fehlende Umsetzung einiger RSS 2.0 Elemente in der Praxis zurückgeführt werden und zum Anderen werden in Atom die Funktionen einiger RSS 2.0 Elemente, wie beispielsweise des „language"-Tags auf andere Weise umgesetzt. Eine detailliertere Übersicht über die syntaktischen Unterschiede zwischen RSS und Atom gibt ein Auszug der Übersicht von [MEIERT]:

RSS 2.0	Atom 1.0	Kommentare
channel	feed	
title	title	
link	link	Atom definiert eine erweiterbare Familie von rel-Werten.
description	subtitle	
language	–	Atom verwendet das standardisierte xml:lang-Attribut.
copyright	rights	
managingEditor	author oder contributor	
pubDate	published (in einem Eintrag)	Atom besitzt kein Äquivalent auf Feed-Ebene.
lastBuildDate (in einem Kanal)	updated	RSS besitzt kein Äquivalent auf Element-Ebene.
category	category	
generator	generator	
image	logo	Atom empfiehlt ein Seitenverhältnis von 2:1.
item	entry	
author	author	
–	contributor	
description	summary und/oder content	Abhängig davon, ob der komplette Inhalt angeboten wird.
enclosure	–	Atom: rel="enclosure" auf link-Element.
guid	id	
source	–	Atom: rel="via" auf link-Element.
–	source	Container für Metadaten auf Feed-Ebene, um Aggregation zu fördern.

Tabelle 5: Auszug der Übersicht von [MEIERT]

Neben der Nutzung von RSS im privaten Bereich, kommt RSS auch in vielen Anwendungen in Unternehmen zum Einsatz. Einen genaueren Einblick in die Verwendung und Möglichkeiten von RSS im Unternehmen bietet Kapitel 2.4.

2.2.2.2 Usenet

Ein älterer Vertreter von „User Client Aware Subscriptions“ ist das „Usenet“, das trotz der sinkenden Bedeutung im heutigen Web durch die gebotene breite Wissensbasis in dieser Analyse eine Rolle spielt und hinsichtlich der Aggregation heterogener Nachrichtenquellen betrachtet werden muss.

„Usenet“ ist ein auf dem „Network News Transport Protocol“ (NNTP) aufbauendes, speziell für Diskussionsforen entworfenes System. Jeder Server, der „Usenet-Newsgroups“ anbietet, definiert dabei selbst die Themen der angebotenen Newsgroups. Diese Newsgroups werden hierarchisch angeordnet, so dass zu jedem Thema speziellere Unterkategorien gebildet werden können (z.B. „computer.java.web.jsf“).

Nutzer von Newsgroups melden sich bei einem entsprechenden Server an und abonnieren die Themen, die für sie interessant sind. Üblicherweise wird hierfür eine spezielle Clientsoftware benutzt, um den Zugriff auf die Inhalte des Newsgroup-Servers zu erleichtern. Neue Artikel können ebenfalls von Nutzern auf die Newsgroup-Server hochgeladen werden. Werden neue Artikel publiziert, verteilt der Server, der den Artikel empfangen hat, diesen an alle anderen Server, die die gleichen Newsgroups anbieten. Um diese Weiterleitung von neuen Artikeln zu gewährleisten, verfügt jeder Server über eine Tabelle, in der alle Nachbarserver mit ihren Newsgroups abgelegt sind. Wird ein Server gefunden, der die gleiche Newsgroup anbietet, wird dieser mittels des „IHAVE“- Kommandos und der Artikel-ID kontaktiert. Besitzt dieser Server diesen Artikel noch nicht, sendet er einen Statuscode zurück, um den Ursprungsserver aufzufordern, diesen Artikel zu übermitteln. Diese Prozedur wird nach Erhalt des Artikels auf diesem Server wiederholt, damit sich der Inhalt weiter verbreiten kann.

2.3 Consumer Aggregatoren

Mit der zunehmenden Verbreitung von Feeds und Microblogging-Netzwerken sowie deren Beliebtheit bei den Nutzern, wächst ebenfalls die Informationsmenge erheblich, die auf Nutzer einströmt, die viele Feeds abonniert haben und viele Personen in Microblogging-Netzwerken verfolgen. Um diese zum Teil sehr große Informationsmenge zu überschauen, ist es sinnvoll, die einzelnen Feeds mit Hilfe eines geeigneten Filters vor dem Lesen zu sortieren und gegebenenfalls offensichtlich uninteressante Dinge zu ignorieren. Die folgenden Abschnitte 2.3.1 und 2.3.2 geben einen Überblick über eine Auswahl der derzeitig verfügbaren Lösungen für den Consumer-Bereich.

2.3.1 Feed Aggregatoren

Um die eingangs beschriebene Informationsflut zu bewältigen, werden von den derzeit verfügbaren Lösungen hauptsächlich zwei Wege verfolgt. Entweder muss der Nutzer exakt definieren nach welchen Kriterien die von ihm abonnierten RSS-Feeds

gefiltert werden sollen oder der Nutzer wird dazu aufgefordert Artikel zu bewerten, um dem jeweiligen Dienst ein Feedback über seine Präferenzen zu geben. Mit Hilfe dieser gesammelten Daten wird ein Nutzerprofil erstellt und versucht, Feeds von vornherein als interessant oder uninteressant für den Nutzer zu bewerten.

Ein Dienst, der sich darauf spezialisiert hat Feeds durch die exakte Angabe von Kriterien wie Schlagwörtern, Datumsangaben oder Ortsangaben zu filtern, ist „Yahoo Pipes“ [YahooPipes]. Dem registrierten Nutzer wird hierbei eine AJAX-Oberfläche zur Verfügung gestellt, um Feed-Quellen, Filterelemente und Operatoren per „Drag and Drop“ auf der Arbeitsfläche zu positionieren (Abbildung 14):

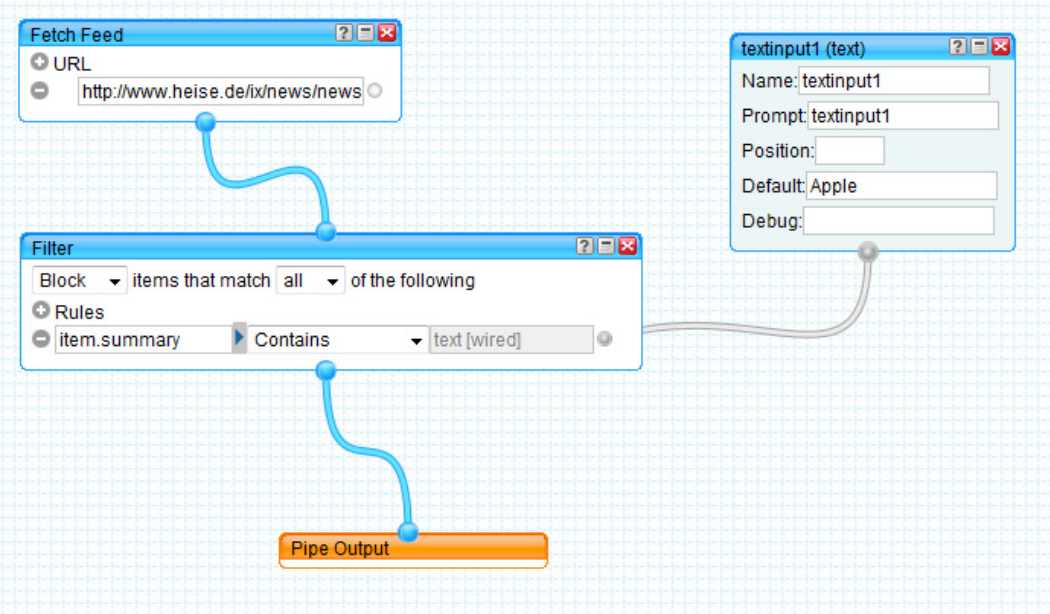

Abbildung 14: Yahoo Pipes Arbeitsfläche

Diese Elemente können konfiguriert und miteinander verbunden werden. So entsteht eine leicht überschaubare Kette des Filterprozesses, von der reinen Datenquelle über die angewendeten Filter, bis hin zum Ausgabeelement dieser Kette bzw. Pipe.

Technisch gesehen werden die Eingabeformate, wie beispielsweise RSS oder Atom, von der „Yahoo Pipes Engine“ von Fehlern im Markup bereinigt und in „UTF-8“ umgewandelt [Trevor]. Anschließend werden die in der Pipe angegebenen Filter und Operatoren auf die Quelldaten angewendet und in das gewählte Ausgabeformat umgewandelt.

Einen anderen weniger technischen Ansatz verfolgen andere Dienste, die sich auf das Aufbereiten von externen Feed-Angeboten spezialisiert haben. In diesen Diensten wird meist eine Weboberfläche bereitgestellt, in der registrierte Nutzer ihre abonnierten Feeds angeben können. Häufig kann an dieser Stelle auch eine „OPML“- Datei mit den Pfaden der Feed-Quellen importiert werden. Aus diesen Feed-Inhalten wird dann eine Übersicht erstellt, die mit zusätzlichen Steuerelementen versehen ist, damit Nutzer die RSS-/Atom- Inhalte bewerten können. Bei einigen Diensten wie Feeds 2.0 [Feeds2.0] ist lediglich eine Entscheidung des Nutzers zwischen interessant und uninteressant notwendig, um dem System ein Feedback

über die Relevanz des Inhaltes zu geben. Bei anderen Diensten wie „Feedhub“ [Feedhub] wird die Kategorisierung von interessant und uninteressant zusätzlich in je fünf Unterstufen eingeteilt, um ein möglichst genaues Feedback für die Adaption der Feedinhalte an die persönlichen Präferenzen des Nutzers zu erreichen. Die gewonnenen Informationen über die Interessen des Nutzers werden dann für die Sortierung der abonnierten Feeds verwendet. In den meisten Fällen sind diese Feeds vom Nutzer bereits verschiedenen Bereichen wie Politik oder Sport zugeordnet worden. Zusätzlich zu dieser Kategorisierung werden die Feeds, wie bereits beschrieben, nach ihrer errechneten Relevanz für den Nutzer sortiert, so dass Nutzer, die für sie interessanten Inhalte vorsortiert, bewertet und übersichtlich präsentiert bekommen.

Als Vertreter von Feed-Aggregatoren, die ein Filter- bzw. Bewertungssystem beinhalten, lässt sich auch der kanadische Dienst „AideRSS“ bzw. „PostRank“ zählen [AIDERSS]. Dieser Dienst bietet Nutzern die Möglichkeit, jedes „Web-Feed“- Format zu aggregieren und deren Inhalte über eine Weboberfläche anzuzeigen. Die angezeigten Feeds können Kategorien bzw. Themen zugeordnet werden, um mehrere Feeds gleichen Themas zusammenzufassen. Die Filter- bzw. Bewertungsfunktion basiert bei „AideRSS“ auf der Aggregation von externen Bewertungsindikatoren (z.B. „Posts“ bei „Twitter“, „Bookmarks“ bei „Delicious“ und Weitere) sowie auf der Anzahl der Klicks, die in der eigenen Oberfläche auf diesen Inhalt ausgeführt wurden. Anhand dieser Informationen wird ein „PostRank“ mit einem Wertebereich von eins bis zehn errechnet und den einzelnen Nachrichtenelementen zugeordnet.

2.3.2 Microblogging Aggregatoren

Mit der zunehmenden Nutzung von Microblogging-Netzwerken in den letzten Jahren hat auch die Anzahl der zur Verfügung stehenden Netzwerke zugenommen. Das hatte zu Folge, dass sich die Microblogging-Nutzer auf eine Vielzahl von Netzwerken verteilt haben, die in den seltensten Fällen eine native Unterstützung für die Vernetzung untereinander bieten. Dieser Fakt bildete die Grundlage für die Entwicklung von Microblogging-Aggregatoren, die es Nutzern ermöglichen, mehrere Netzwerke mit einer Client-Applikation zu nutzen.

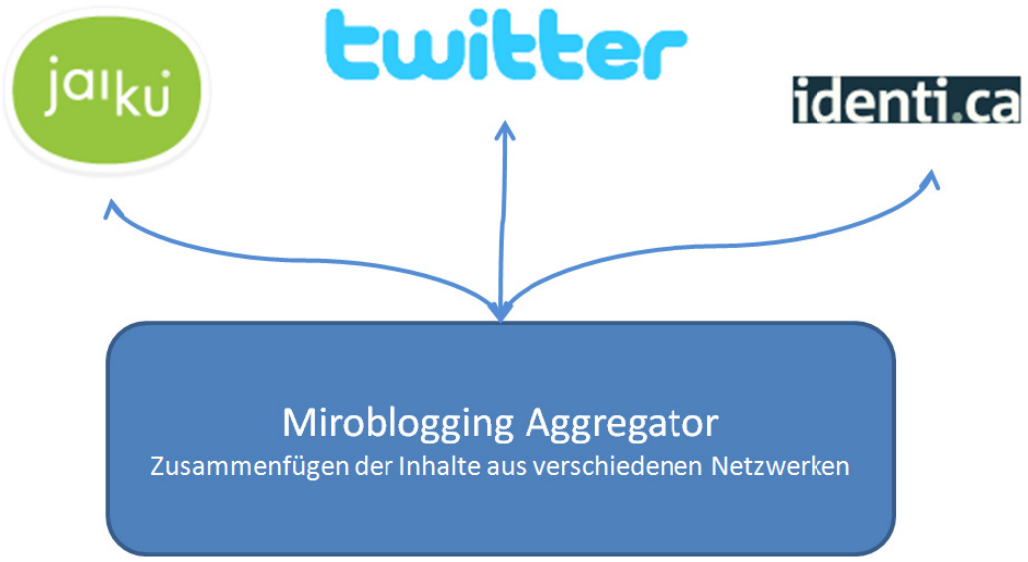

Abbildung 15: Konzept von Microblogging-Aggregatoren

In diesem Bereich häufig genutzte Aggregatoren sind beispielsweise „Twhirl“, „Gwibber“ und „HelloTxt“. Jedes dieser Produkte ist in der Lage, Nachrichten in die am weitesten verbreiteten Netzwerke zu publizieren und Nachrichten der verfolgten Personen aus diesen Netzwerken in der Clientapplikation anzuzeigen. Darüber hinaus unterstützen einige Lösungen wie beispielsweise „HelloTXT“ auch andere soziale Netzwerke wie „Facebook“ oder „LinkedIn“ und können Statusmeldungen auch in diese Netzwerke publizieren.

Mit Sicht auf die Personalisierung und Filterung der eingehenden Nachrichtenströme ist festzustellen, dass derzeit noch keine Ansätze vorhanden sind, die den Nutzer in ausreichender Form bei der Filterung der Nachrichten unterstützen. Programme wie „Tweetdeck“ bieten die Möglichkeit, Nutzer einzelnen Gruppen zuzuweisen, so dass eine Kategorisierung der empfangenen Nachrichten vorgenommen werden kann. Darüber hinaus ist die manuelle Suche nach Schlagwörtern oder Personen möglich. Allerdings fehlt derzeit noch die Möglichkeit, die eingehenden Nachrichten mit Hilfe eines Nutzermodells auf Basis von Nutzerpräferenzen automatisch zu filtern.

2.4 Enterprise Aggregatoren

Betrachtet man die Nutzung von Feeds und Microblogging-Netzwerken in Unternehmen, wird deutlich, dass für einen unternehmensinternen Einsatz dieser Informationskanäle andere Anforderungen an die verwendete Software gestellt werden, als dies im Consumer-Bereich der Fall ist. [YOUNG+07] beschreibt in seiner Studie, wie der Informationsfluss in Bezug auf Feeds in Unternehmen zu sehen ist.

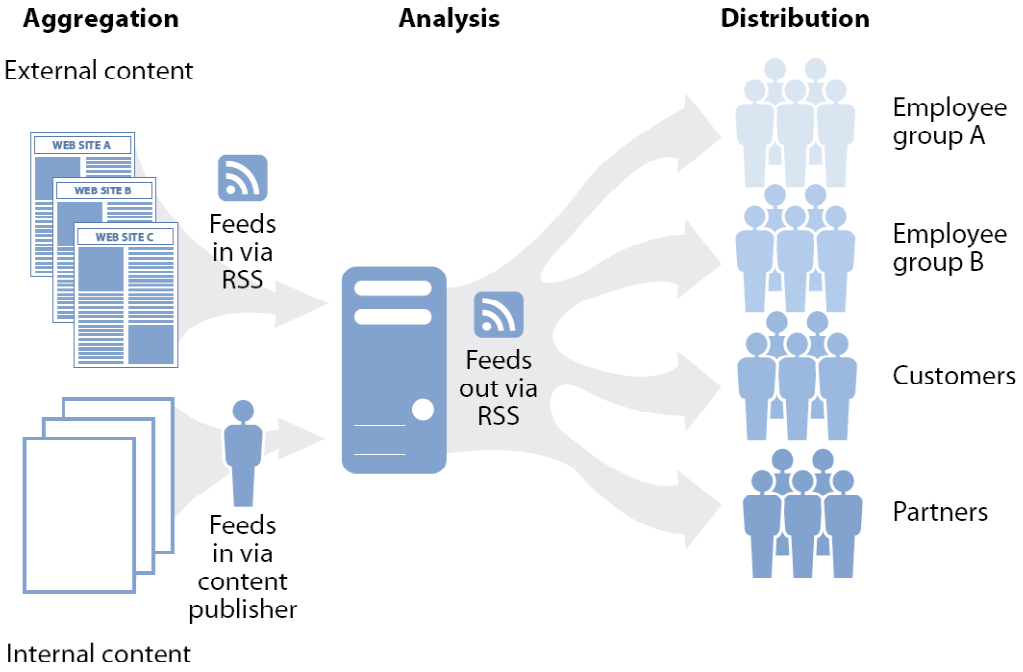

Abbildung 16: Informationsfluss am Beispiel von Feeds im Unternehmen [YOUNG+07]

Neben dem einfachen Aggregieren von Feed Inhalten aus verschiedenen internen und externen Quellen ergeben sich auch weitere Anforderungen für den effizienten und sicheren Einsatz von Feeds im Unternehmen.

2.4.1 Anforderungen

Die Umsetzung von Feed-Lösungen für Unternehmen wirft eine Reihe von Anforderungen auf, die bei gewöhnlichen Consumer-Produkten nur eine untergeordnete oder keine Rolle spielen. Für Unternehmen ist es wichtig, dass sich Software möglichst reibungslos in die bestehende IT-Infrastruktur und Softwareumgebung integrieren lässt und mit dieser vernetzt werden kann. Darüber hinaus spielen Sicherheits- und Kollaborationsaspekte eine entscheidende Rolle. Die nachfolgende Auflistung gibt einen Überblick über die wichtigsten Anforderungen an Enterprise-Feed-Lösungen:

- Hinter einer Firewall installierbar
- Externe Feed-Inhalte müssen hinsichtlich ihrer Sicherheit geprüft werden (bsp. JavaScript-Code entfernen)
- Einbindung bzw. Bereitstellung von Plugins für „Microsoft Outlook“, mobile Endgeräte sowie Bereitstellung eines Web-Interface
- Benutzerverwaltung mit Unterstützung für „Microsoft Active Directory“ bzw. „LDAP“
- Rechteverwaltung für geschützte Inhalte
- Filter- und Suchfunktionen für Feeds
- Adaptive Anpassung des Rankings von Feeds durch Analyse ihrer Nutzung

– Kollaborationsfunktionen wie „Markieren von wichtigen Artikeln“ oder „Weiterleiten von Empfehlungen an Kollegen“
– Gruppieren von Feeds in verschiedenen Kategorien
– Synchronisation und Konsistenzerhaltung bearbeiteter Daten [Schill+06]
– „Group Awareness“: Explizite Sichtbarkeit der Gruppenmitgliedschaft [Schill+06]

Nachfolgend werden die derzeit führenden Lösungen analysiert und hinsichtlich der bereits genannten Anforderungen überprüft, um eine Übersicht über den heutigen Stand der Technik zu bieten.

2.4.2 Attensa

Die Firma „Attensa“ bietet mit dem „Attensa Feed Server“ (AFS) und „Feed Reader“ eine einheitliche Lösung für die Verwendung von RSS im Unternehmen. Der AFS wird dabei hinter einer Firewall im unternehmensinternen Netz eingerichtet und verwaltet alle externen und internen Feed Inhalte, die den Nutzern im Unternehmen zur Verfügung stehen.

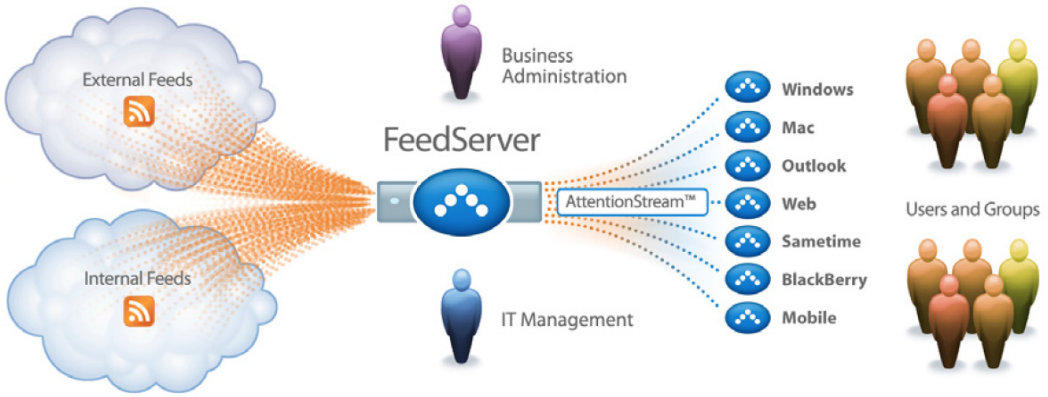

Abbildung 17: Attensa Feedserver [ATTENSA]

Um die Sicherheit bei externen Feeds zu gewährleisten, können sicherheitskritische Inhalte wie IFrames oder JavaScript sowohl durch die Server-Applikation als auch durch die Client-Applikation gefiltert werden. Darüber hinaus können Administratoren potentiell bösartige Anhänge bzw. Podcasts blockieren und Listen für gesperrte Feeds definieren.

Für die Verbindung von „Feed-Server“ und „Feed-Reader“ setzt „Attensa“ ein eigens für diesen Zweck entwickeltes Protokoll mit dem Namen „Attention Stream“ ein. Es stellt sicher, dass die Feed-Inhalte des Servers immer synchron mit den Inhalten auf den vom Nutzer verwendeten Endgeräten sind. Zusätzlich werden über den „Attention-Stream“ Daten über die Lesegewohnheiten des jeweiligen Nutzers an den Feed-Server gesendet und dort ausgewertet. Dies bietet zum Einen den Vorteil, dass nach einer Lernphase jedem Nutzer eine personalisierte Form der abonnierten Feeds angeboten werden kann, die nach Relevanz für den Nutzer sortiert ist. Darüber hinaus ist dies auch eine gute Möglichkeit, die Akzeptanz von Feeds

innerhalb eines Unternehmens festzustellen und eine Basis für kollaborative Erweiterungen des Systems bereit zu stellen. In der gegenwärtigen Version 2.6 ist eine solche Erweiterung allerdings noch nicht implementiert. Wünschenswert ist eine Funktion, die Ähnlichkeiten zwischen Nutzern feststellt und auf Basis dieser Erkenntnisse ähnlichen Nutzern Lesevorschläge unterbreitet. Derzeit ist es lediglich möglich, Artikel bzw. Inhalte manuell als Empfehlung an andere Nutzer weiterzuleiten.

Zusätzlich zu den bereits vorgestellten Funktionen bietet der AFS eine Nutzer- und Gruppenverwaltung unter Verwendung bereits bestehender „LDAP-Server", so dass eine einfache Integration in ein bestehendes Firmennetzwerk ermöglicht wird. Die Administration erfolgt dabei über eine rollenbasierte AJAX-Web-Oberfläche und bietet neben den Konfigurationsmöglichkeiten des Systems ebenfalls ein großes Angebot statistischer Übersichten zu Feed-Nutzungen und Lesezeiten einzelner Feeds.

Um die Funktionen des AFS zu nutzen, stehen eine Reihe von Zugriffsmöglichkeiten zur Verfügung. Neben einer Weboberfläche und dem Zugang über mobile Endgeräte wird ein kostenloses „Microsoft Outlook"- Plugin, mit dem Namen „Attensa Feed Reader" (AFR) angeboten. Dieses Plugin ermöglicht die vollständige Integration aller zur Verfügung stehenden Funktionen in die „MS Outlook"- Umgebung. Dies beinhaltet neben einer Übersicht der Feeds ebenfalls Funktionen, um direkt aus der Reader-Anwendung Blogbeiträge zu verfassen und einen integrierten „Media Player" für die Anzeige und das Abspielen von Podcast-Inhalten zu nutzen [Attensa].

2.4.3 Newsgator Social Sites

Im Unterschied zum „Attensa Feed Server" bietet „Newsgator" mit seinem Produkt „Newsgator Social Sites 2.0" (NGSS) neben der Bündelung von RSS-Inhalten ebenfalls die Integration in bestehende Systeme wie „Microsoft Sharepoint". Dabei ist NGSS als auf „ASP.net" basierende Webanwendung konzipiert, die neben einer umfangreichen Weboberfläche auch viele Zugriffsmöglichkeiten durch mobile Endgeräte oder einem „Microsoft Outlook"-Plugin bietet. NGSS verfolgt dabei konsequent einen kollaborativen Ansatz, um alle am System beteiligten Nutzer zu vernetzen. Im Einzelnen bedeutet dies, dass jedem Nutzer ein Kollegen- und Communitiesbereich zur Verfügung steht. Im Kollegenbereich werden beispielsweise durch das System gefundene, ähnliche Nutzer aufgelistet. Dies erfolgt auf Basis der von jedem einzelnen Nutzer abonnierten RSS-/Atom- Feeds, sowie der von den Nutzern am häufigsten verwendeten Tags [TECHNORATI]. So wird es Nutzern ermöglicht, Kollegen, die sich mit ähnlichen oder gleichen Themen beschäftigen, zu finden und Kontakt mit ihnen aufzunehmen oder durch die von ihnen erstellten Informationen (Beiträge, Bookmarks) ihr eigenes Wissen zu erweitern. Diese Verbindungen können vom System ebenfalls als Sozialer Graph angezeigt werden, um fachliche Kontakte innerhalb des Systems abzubilden (Abbildung 18):

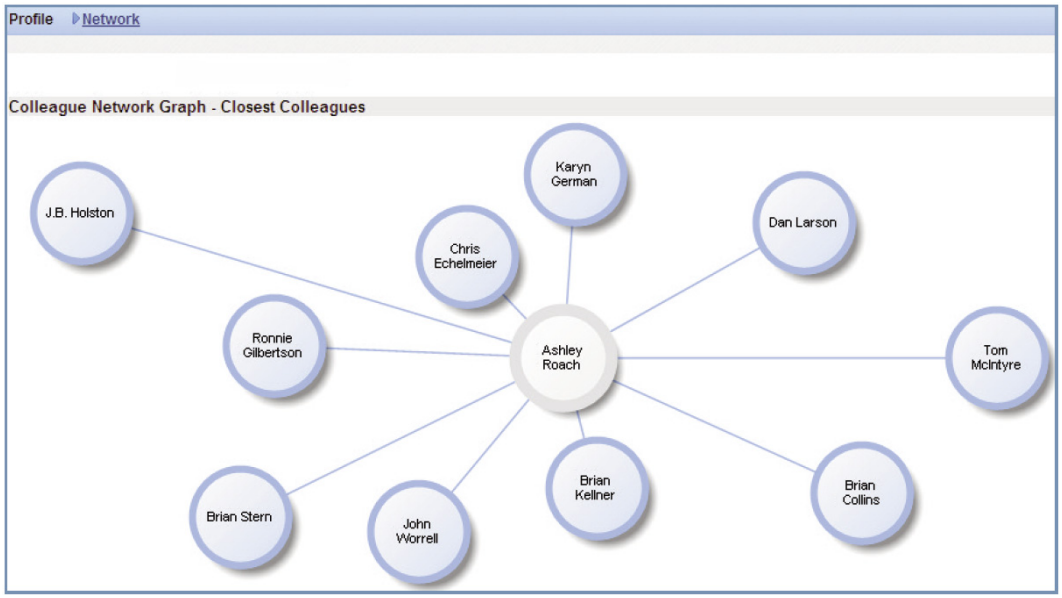

Abbildung 18: Social Graph [Newsgator]

Darüber hinaus bietet NGSS die Möglichkeit, dass Nutzer eigene, meist fachspezifische Communities gründen oder diesen beitreten können. Diese beinhalten ein Diskussionsforum sowie Rubriken für relevante Feed-Inhalte, Bookmarks zum Thema und Dokumente für die Community. Weiterhin werden Inhalte, die in angeschlossene Wikis oder Sharepoint-Server publiziert werden und die auf die der Community zugeordneten Tags passen, automatisch in dieser angezeigt oder referenziert.

Weiterhin erstellt NGSS für jeden registrierten Nutzer eine persönliche Seite, die individuell angepasst werden kann. Hierzu bietet Newsgator neben der Möglichkeit, persönliche Informationen (Aktivitäten, Tags, Bookmarks, automatisch erstellte Listen zur Feednutzung) anzugeben, auch Widgets zu verschiedenen Themen (beispielsweise ein National Geographic Widget) an, die in die persönliche Seite eines Nutzers aufgenommen werden können. Zudem wird zu den bereitgestellten Inhalten auf der persönlichen Seite oder in den Communities immer eine Tag-Cloud angezeigt, um die Nutzer über Trends und aktuell häufig benutzte Schlagwörter zu informieren.

Ein weiterer wichtiger Bestandteil der Weboberfläche des NGSS ist der integrierte Feed-Reader. Er bietet die Möglichkeit, Feeds mit Tags zu versehen, nach Relevanz zu sortieren oder nach Tags zu filtern. Dabei wird der Feed Reader genutzt, um dem System Rückkopplung über die Lesegewohnheiten des Nutzers zu geben und anhand dieser Daten eine Sortierung nach Relevanz der Feeds für den Nutzer zu ermöglichen.

Mit Blick auf die Integrationsfähigkeit von NGSS ist festzustellen, dass mit der Möglichkeit, vorhandene Dienste wie „Active Directory“ bzw. „LDAP“ oder „Microsoft Sharepoint“ zu nutzen, diese Anforderung an die Enterprise-RSS-Lösungen sehr gut erfüllt worden ist. Darüber hinaus lässt sich NGSS, wie auch der „Attensa Feed Server“, hinter einer Firewall installieren und verfügt über

Möglichkeiten, externe Inhalte hinsichtlich potentiell schädlicher Bestandteile zu filtern.

2.4.4 Zusammenfassung

In den vorangegangenen Abschnitten wurden die beiden führenden Enterprise-Feed-Lösungen hinsichtlich ihrer Leistungsmerkmale betrachtet. Dabei wurde deutlich, dass sich beide für den Einsatz im Unternehmen eignen und viele der eingangs gestellten Anforderungen erfüllen. Der wesentliche Unterschied liegt in der Ausrichtung der beiden Produkte. Während sich der „Attensa Feed Server" hauptsächlich auf die Verwaltung und Bereitstellung von Feed-Inhalten konzentriert, geht der Ansatz von „Newsgator Social Sites" noch weiter. Die Erweiterung eines Feed-Servers zu einem Punkt, an dem Inhalte aus verschieden Quellen bereitgestellt werden und in ein kollaborativ ausgerichtetes Netzwerk eingebunden werden, kann für Unternehmen eine sehr sinnvolle Art des Informationsmanagements sein.

Dennoch verfügt keines der betrachteten Produkte über die Möglichkeit, zusätzliche Informationskanäle wie beispielsweise Microblogging-Netzwerke oder „Usenet-Newsgroups" einzubinden. Wünschenswert ist eine offenere und leichter zu erweiternde Plattform für den unternehmensinternen Einsatz, mit der Mitarbeiter auf einfache und übersichtliche Weise ihre Informationskanäle bündeln und filtern können. Zudem sollte eine solche Plattform konsequent einen kollaborativen Ansatz verfolgen, so dass Gemeinsamkeiten unter den Nutzern der Plattform verwendet werden können, um die Filterung eingehender Nachrichtenströme effektiver zu gestalten.

3 Realisierung der Infrastruktur

Im Rahmen dieser Arbeit entsteht eine Infrastruktur, die es ermöglichen soll, verschiedene Nachrichtenquellen zu bündeln bzw. zu aggregieren. Darüber hinaus ist vorgesehen, dass dieser gebündelte Nachrichtenstrom personalisiert gefiltert wird. Das heißt, dass jeder Nutzer dieser prototypischen Anwendung eine personalisierte Sicht auf die von ihm abonnierten Nachrichtenquellen erhält. Um dies zu erreichen, sollen die Verfahren, Technologien und Nachrichtenformate, die im Kapitel 2 - Grundlagen vorgestellt wurden, eingesetzt werden. An dieser Stelle ist anzumerken, dass sich die Entwicklung ausschließlich auf die Konzeption eines Backends bezieht. Das Frontend für die Darstellung der gefilterten Inhalte ist Teil einer weiteren Diplomarbeit [Canbolat+09].

In den nachfolgenden Abschnitten werden die Phasen des Entwicklungsprozesses dokumentiert und die Infrastruktur aus technischer Sicht näher beschrieben. Dabei werden im Rahmen der Analyse kontinuierlich die erarbeiteten Anforderungen an die Infrastruktur dokumentiert.

3.1 Detailanalyse und Konzeption

3.1.1 Nutzergruppenanalyse

Die prototypische Anwendung wird mit der Vorgabe entwickelt, für den Einsatz in Unternehmen geeignet zu sein. Aus diesem Grund wurden die primären Nutzergruppen entsprechend gewählt und deren Interessen abgeleitet.

Nutzergruppe	Interessen
Administratoren / Geschäftsführung	– Statistiken erfassen, welche Themen für Mitarbeiter interessant sind – Kollaborativen Nutzen aus Mitarbeiteraktivität ziehen – Trends frühzeitig erkennen – Zeitersparnis für Mitarbeiter
Endnutzer	– Zeitersparnis und Erleichterung der Suche nach relevanten Nachrichteninhalten – Automatische Vorschläge für interessante Inhalte – Geringe Komplexität – Transparenter und nachvollziehbarer Filterprozess – Vertraulicher Umgang mit persönlichen Interessen und Informationen

Tabelle 6: Nutzergruppenanalyse

3.1.2 Anwendungsfälle

Wie in der Nutzergruppenanalyse (vgl. Kapitel 3.1.1) bereits dargestellt, gibt es zwei unterschiedliche Arten von Nutzern: Frontend-Nutzer (vgl. Abbildung 19) und Nutzer, die Zugriff auf die administrativen Funktionen des Systems haben (Abbildung 20). Für die Entwicklung des Prototyps und dessen Frontend Schnittstelle sind die nachfolgenden Anwendungsfälle der Abbildung 19 relevant:

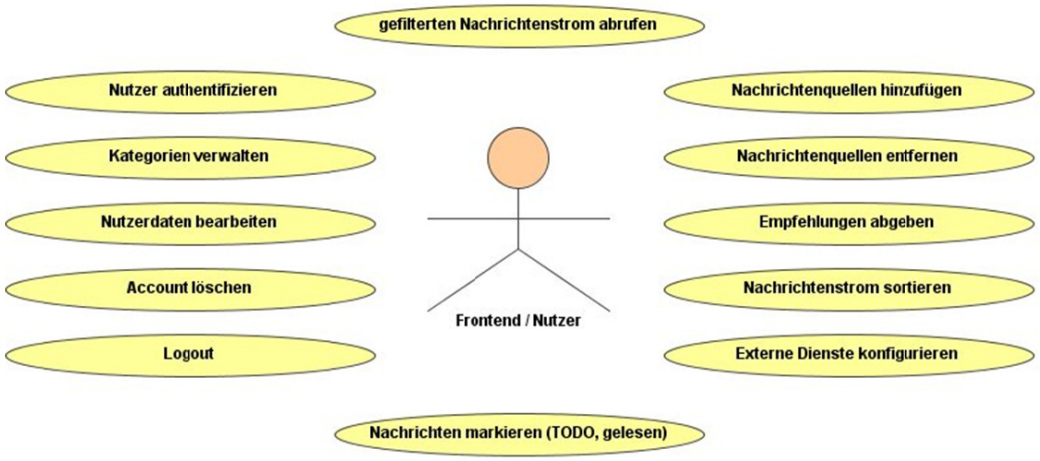

Abbildung 19: Anwendungsfalldiagramm Nutzer / Frontend

Im weiteren Verlauf dieses Kapitels werden kontinuierlich die Anforderungen, die durch diese Analyse an das Backend gestellt werden, dokumentiert und nummeriert.

Anforderung #1: Unterstützung benutzerdefinierter Kategorien (Definierbar nach Quellen und Schlüsselwörtern).

Anforderung #2: Unterstützung von personalisierten Metainformationen (z.B. Nachricht gelesen, verschieben auf ToDo-Liste).

Anforderung #3: Integration externer Dienste zur Akquise von Schlüsselwörtern und Nutzerprofilen.

Anforderung #4: Personalisiertes Bewertungssystem für Nachrichtenelemente.

Anforderung #5: Personalisierte Zusammenstellung von Nachrichtenquellen.

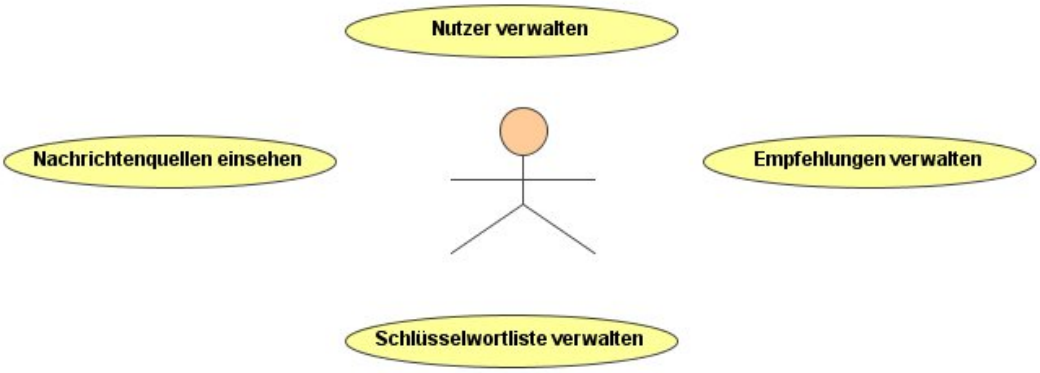

Abbildung 20: Anwendungsfalldiagramm Backend / Administrator

Anforderung #6: Oberfläche für administrative Funktionen zur Verfügung stellen (Einsehen der verwalteten Daten; Ändern der Parameter der Anwendungskonfiguration).

3.1.3 Datenfilterung und Datenhaltung

Ein zentrales Problem und zugleich Motivation für diese Arbeit ist die Tatsache, dass sich viele Nutzer von Online-Nachrichtenangeboten einer kaum überschaubaren Informationsmenge aus heterogenen Informationsquellen gegenüber sehen. Eine manuelle Filterung und Sortierung nimmt in den meisten Fällen viel Zeit in Anspruch, so dass die Unterstützung durch eine Software notwendig wird.
Daraus ergibt sich:

Anforderung #7: Aggregation heterogener semistrukturierter Quellen.

Eine Software, die diese Nachrichtenquellen aggregieren und personalisiert filtern soll, muss dabei eine Reihe von Problemen lösen:

Anforderung #8: Personalisierte Filterung von Nachrichten aus heterogenen semistrukturierten Quellen.

Es muss entschieden werden, ob die eingehenden Nachrichtenströme durch das System persistent gespeichert werden oder ob lediglich Referenzen auf die ursprünglichen Inhalte durch das System gehalten werden und diese Inhalte nur bei Bedarf (on demand) von der ursprünglichen Quelle geladen werden:

Anforderung #9: Vollständige Verfügbarkeit der Inhalte; aus Nutzersicht immer die aktuellen Inhalte der Quelle anzeigen.

Eine persistente Speicherung der Inhalte hat den Vorteil, dass die Operationen, die auf diesen Inhalten durch die verwendeten Filtermechanismen ausgeführt werden, in verhältnismäßig kurzer Zeit geschehen können, da sie nicht über eine Netzwerkverbindung aus der ursprünglichen Quelle geladen werden müssen. Ein Nachteil der persistenten Speicherung und zugleich ein Vorteil einer „on demand-

Lösung“ ist die Anforderung an den Speicherplatz, den das System mit der jeweiligen Lösung benötigt. Bei hohen Nutzerzahlen und vielen Abonnements der Nutzer steigt die zu verwaltende Datenmenge erheblich. Unter diesem Gesichtspunkt sind auch die Anforderungen an die „Quality of Service“ zu betrachten. Eine „on demand-Lösung“, die Inhalte aus vielen unterschiedlichen Nachrichtenquellen aggregiert, benötigt unter Umständen zu viel Zeit, um alle notwendigen Daten zu sammeln und zu verarbeiten. In diesem Zusammenhang könnte die Verwendung geeigneter Caching-Mechanismen in Betracht gezogen werden, um einen Mittelweg zwischen beiden Lösungen zu finden und die Vorteile beider Verfahren miteinander zu vereinen. Des Weiteren muss eine geeignete Zeitvorgabe für die Dauer der Speicherung der Inhalte oder deren Referenzen gefunden werden. Zudem ist es für Nachrichtenformate, die nativ keinen eindeutigen Identifier für Nachrichtenelemente beinhalten, erforderlich, geeignete Mechanismen zu finden, um Nachrichten aus diesen Quellen im Nachhinein wieder eindeutig zu identifizieren. Viele Formate unterstützen die Angabe eines eindeutigen Identifiers, allerdings kann es vorkommen, dass diese Möglichkeit von einigen Anbietern nicht unterstützt wird. An dieser Stelle müssen geeignete Verfahren gefunden werden, um diese fehlenden Angaben durch das System einzufügen, damit gelöschte oder veraltete Artikel, nachdem sie von der Quelle nicht mehr angeboten werden und die maximale Vorhaltedauer überschritten ist, ebenfalls aus dem System gelöscht werden können.
Daraus folgt:

Anforderung #10: Identifizierbarkeit einzelner Nachrichtenelemente über längeren Zeitraum (auch bei sich verändernder Quelle).

Neben der grundsätzlichen Entscheidung, wie hoch die maximale Vorhaltedauer von Inhalten sein soll, ist eine Unterscheidung nützlich, ob eine Information nur ein flüchtiges Interesse bzw. eine kurze Gültigkeit oder ein langlebiges Interesse besitzt. Unabhängig von der Quelle und dem genutzten Format kann die Gültigkeitsdauer einer Information sehr stark variieren. Diese Metainformation zu einer Nachricht lässt sich demnach nicht anhand eines Formates oder anderer messbarer Eigenschaften festlegen. Somit kann eine solche Kategorisierung von einem System nur schwer automatisch vorgenommen werden, weshalb der Einsatz von kollaborativen Filtern, die Nutzerbewertungen in den Filterprozess einfließen lassen, empfehlenswert ist:

Anforderung #11: Einsatz von kollaborativen Filtern, um subjektive Eigenschaften von Nachrichten zu nutzen.

Ein weiteres Problem stellen die unterschiedlichen semistrukturierten und strukturierten Quellen dar. Bevor ein Filtermechanismus entwickelt werden kann, muss entschieden werden, ob dieser für jedes einzelne Format separat entworfen wird oder ob ein universeller Mechanismus entwickelt werden kann, der ein systemeigenes Format als Eingabe akzeptiert und hochwertige Ergebnisse liefert. Der Vorteil eines universellen Mechanismus liegt zum Einen im Entwicklungsaufwand, da nur ein Mechanismus entwickelt werden muss. Zum Anderen würde er eine

spätere Erweiterung des Systems um ein weiteres Format vereinfachen, da in einem solchen Fall kein neuer Filtermechanismus entworfen werden müsste, sondern die vorhandenen Komponenten genutzt werden könnten. Voraussetzung und zugleich Nachteil eines universellen Filtermechanismus ist, dass als Eingabe ein einheitlicher, formatunabhängiger Nachrichtenstrom erforderlich ist. Hierzu müsste ein internes Nachrichtenformat entwickelt werden, das die unterschiedlichen Eigenschaften der Ausgangsformate möglichst verlustfrei und sinngemäß in das systemeigene Format überträgt. Zudem ist zu beachten, dass jede dieser Quellen unterschiedliche Meta-Informationen bereitstellt. Beispielsweise sehen einige Formate eine native Unterstützung für Tags vor, um den eigentlichen Inhalt mit Schlüsselwörtern näher zu beschreiben, wohingegen andere eine solche Funktionalität nicht bieten. Auch hier müssten geeignete Verfahren gefunden werden, um diese Unterschiede auf ein einheitliches Niveau zu normalisieren.
Demnach ergibt sich:

Anforderung #12: Unterstützung von Metainformationen zu aggregierten Inhalten.

Ein solches internes Format für die Inhalte der verschiedenen Quellen müsste darüber hinaus die Möglichkeit bieten, zusätzliche Meta-Informationen zu speichern, die durch die Nutzer oder das System selbst erstellt wurden und für den Filterprozess verwendet werden können. Dies können beispielsweise Bewertungen einzelner Inhalte oder zusätzliche Schlüsselwörter sein, die durch automatisierte Verfahren den Inhalten hinzugefügt werden:

Anforderung #13: Unterstützung von Mechanismen zur Anreicherung der Inhalte mit weiteren Metainformationen (z.B. Autotagging).

Um eine möglichst hochwertige Filterung der Inhalte vorzunehmen, sollten geeignete Filtermechanismen ausgewählt werden. Diese sollten mit Blick auf die zu Grunde liegenden Inhalte und deren Meta-Daten selektiert werden. Mitunter kann es sinnvoll sein, inhaltsbasierende und kollaborative Filter miteinander zu kombinieren, um objektive und subjektive Eigenschaften von Nachrichten in einem Filterprozess zu berücksichtigen:

Anforderung #14: Kombinieren von inhaltsbasierenden und kollaborativen Filtern.

Ein wichtiger Bestandteil eines solchen Systems ist das Nutzermodell. Durch dieses muss das Problem bewältigt werden, den Nutzer mit seinen Interessen möglichst genau in einer entsprechenden Datenstruktur abzubilden. Dabei muss sichergestellt sein, dass alle notwendigen Daten, die für den Filterprozess von Nutzerseite benötigt werden, aus dem Modell abrufbar sind:

Anforderung #15: Repräsentation des Nutzers bzw. dessen Interessen in einem Nutzermodell.

Um eine solche Funktionalität zu gewährleisten, muss genau analysiert werden, welche Daten manuell durch die Nutzer eingegeben werden müssen und welche Daten automatisch durch das System zusammengetragen werden können. Zudem sollte es dem Nutzer ermöglicht werden, seine Daten, die den kollaborativen Funktionen des Systems zur Verfügung stehen, zu schützen, damit die Privatsphäre jedes Nutzers in einem akzeptablen Maß gewährleistet wird. Im Einzelnen bedeutet dies, dass jeder Nutzer die Möglichkeit haben sollte, abonnierte Inhalte als privat zu definieren, um sie gegenüber anderen Nutzern der Anwendung zu verbergen. Diese Anforderung muss ebenfalls beim Entwurf eines Filtermechanismus beachtet werden:

Anforderung #16: Privatsphäreeinstellungen der Nutzer müssen im System konfigurierbar sein.

3.1.4 Internes Nachrichtenformat

Im Folgenden werden Nachrichtenformate auf ihre Eigenschaften hinsichtlich der Struktur der Inhalte und der mitgelieferten Meta-Informationen betrachtet. Ziel dieser Analyse ist es, Vorbetrachtungen für den Entwurf des Filtermechanismus sowie der Verarbeitung der eingehenden Nachrichtenströme durchzuführen. Dabei wird auch darauf eingegangen, ob es möglich ist, Informationen aus heterogenen semistrukturierten Quellen möglichst verlustfrei in eine einheitliche Repräsentation abzubilden. Zudem wird analysiert, inwieweit diese Nachrichtenformate Metadaten nativ unterstützen und ob dies von den Anbietern dieser Nachrichtenquellen genutzt wird. Dabei wird unter Anderem darauf eingegangen, ob und in welcher Weise es möglich ist, diese Nachrichten mit weiteren Metainformationen zu versehen.

Im Rahmen der Entwicklung des Prototyps werden die Formate „RSS“, „Atom“, „Twitter“ und die Struktur der Inhalte der Microbloggingplattform „Communote“ betrachtet. Diese Formate wurden ausgewählt, da sie sich in Aufbau und Verwendungszweck stark unterscheiden und somit der Anforderung, heterogene Informationsquellen zu aggregieren, gerecht werden. In den folgenden Abschnitten wird nicht der vollständige Funktionsumfang der Formate beschrieben, sondern eine Beschränkung auf die für eine Filterung und Aggregation relevanten Eigenschaften vorgenommen.

3.1.4.1 RSS

Da der Prototyp nur die gängigen RSS-Formate unterstützen sollte, beschränkt sich diese Analyse auf die Eigenschaften, die RSS 2.0 und 0.91 gemein haben.

Durch den RSS-Standard sind für jeden RSS-Channel die drei Elemente „title“, „description“ und „link“ vorgegeben. Darüber hinaus können weitere optionale Elemente für den RSS-Channel angegeben sein. Mit Blick auf eine Filterung von RSS-Feeds, sind die Inhalte der Elemente „language“, „pubDate“ und „lastBuildDate“ von hoher Relevanz.
Daraus folgt:

Anforderung #17: Nachrichtenquellen ist ein Titel zuzuweisen. Optional auch eine Beschreibung, Sprache sowie ein Link.

Die Träger für die Inhalte eines RSS-Channels sind „item"-Elemente, die in ihrer Anzahl nicht begrenzt sind. Diese „item"-Elemente bestehen aus mindestens einem „title"- bzw. einem „description"-Element. Zudem können diese über die optionalen Elemente „author", „category", „link", „enclosure" und „pubDate" mit Metainformationen versehen werden, die für eine Weiterverarbeitung in einem Filterprozess relevant sein können.

Des Weiteren zwingen die RSS-Standards die Anbieter von Feeds nicht zur Vergabe von eindeutigen Identifiern für jedes „item"-Element. Um ein solches „item"-Element auch nach einer Aktualisierung der Quelle wieder auffindbar zu machen, ist es notwendig einen geeigneten Mechanismus zu entwickeln, der diese Elemente eindeutig referenziert. Sollte sich die Position eines „item"-Elementes innerhalb eines Feeds ändern, beispielsweise durch das Hinzufügen neuer Elemente, ist mit einem solchen Mechanismus sichergestellt, dass alte Elemente von neuen unterscheidbar bleiben (siehe auch: Anforderung #10).

Viele Anbieter von RSS-Feeds nutzen die Möglichkeit, Elemente mit Metadaten zu beschreiben, nicht aus. Für eine Filterung dieser Inhalte ist es jedoch notwendig, dass diese Daten vorhanden sind. Aus diesen Gründen ist es denkbar und notwendig, die Inhalte dieser Elemente auf Schlüsselwörter zu durchsuchen, die dem System bekannt sind. Wünschenswert ist darüber hinaus eine Gewichtung der erkannten Schlüsselwörter, um auszudrücken, in welchem Maß das Schlüsselwort für den Inhalt relevant ist. Denkbar ist eine Berechnung der Term-Frequenz des Schlüsselwortes, bezogen auf das gesamte Element. Grundlage für eine derartige Berechnung ist, dass sich alle Feeds im Bezug auf Informationsdichte und Textlänge ähneln. Ist dies nicht der Fall, verfälscht dies das Ergebnis und die ermittelten Werte sind nicht vergleichbar. In der Praxis sind diese Bedingungen oft nicht gegeben. Es existieren Feeds, die Elemente beinhalten, die oftmals nur den Titel eines referenzierten Artikels kapseln. Andererseits werden RSS-Feeds oftmals auch dazu genutzt, komplette Artikel in Form von „escaped"-HTML zu publizieren. Aufgrund dieser Tatsachen ist eine Gewichtung der Schlüsselwörter innerhalb von RSS „item"-Elementen nicht empfehlenswert. Daraus ergibt sich:

Anforderung #18: Nachrichtenelementen sind Titel und Inhalt zuzuweisen. Optional: Autoren, Links und Publizierungsdatum.

3.1.4.2 Atom

Der strukturelle Aufbau von „Atom" ähnelt dem von RSS sehr stark. Das Atom-Äquivalent zum RSS-Element „channel" ist das „feed"-Element. Laut der Atom-Spezifikation [Atom] besteht der Zwang, dass die Elemente „title", „id" und „updated" in jedem „feed"-Element vorhanden sein müssen. Darüber hinaus ist festgelegt, dass ein „feed"-Element auch ein „author"-Element beinhalten muss, es sei denn, jedes der untergeordneten „entry"-Elemente weist ein „author"-Element

auf. Somit weist die Analyse des Atom-„feed“-Elements keine weiteren Anforderungen auf (vgl. Anforderung #17).

Der eigentliche Inhalt wird durch die „entry“ Elemente gekapselt. Diese Elemente müssen jeweils ein „title“-, „id“- und „updated“- Element enthalten. Zusätzlich ist ein „summary“-Element vorgeschrieben, es sei denn, das „entry“-Element beinhaltet ein „content“-Element, das „Base64“-kodierte Daten enthält oder über ein gefülltes „src“-Attribut verfügt. Mit Blick auf eine spätere Filterung sind die Elemente „category“, „published“ und „link“ von Bedeutung.

Die automatische Vergabe von Schlüsselwörtern und einer eventuellen Gewichtung kann an dieser Stelle analog zu RSS (vgl. Kapitel 3.1.4.1)betrachtet werden. Daraus folgen:

Anforderung #19: Beschreibung von Inhalten durch Schlüsselwörter.

und:

Anforderung #20: Format der Inhalte analysieren und separat speichern.

3.1.4.3 Twitter

Der Microblogging-Dienst „Twitter“ bietet für den Zugriff auf die Inhalte, die für einen Nutzer bereit stehen, mehrere Möglichkeiten. Zum Einen ist es möglich, über zugangsgeschützte RSS-Feeds auf die zur Verfügung stehenden Nachrichten zuzugreifen und zum Anderen bietet Twitter für den Zugriff auf die Inhalte des Netzwerks eine „REST API“ an.

Im Wesentlichen besteht eine „Twitter“-Nachricht aus den folgenden Bestandteilen [Twitter-API]:

- ID der Nachricht
- Absender der Nachricht
- Ggf. Adressat der Nachricht
- Text der Nachricht (max. 140 Zeichen Länge)
- Zeitpunkt der Erstellung
- Quelle mit der die Nachricht erstellt wurde (z.B. Weboberfläche, Art der Clientsoftware)

Für den Zugriff auf das „Twitter“- Netzwerk existieren bereits mehrere Bibliotheken für viele unterschiedliche Programmiersprachen. Diese ermöglichen Entwicklern einen einfachen und unkomplizierten Zugriff auf die Inhalte und Funktionen des Netzwerks bzw. der „REST API“.

Im Vergleich zu Web-Feeds sind „Twitter“- Nachrichten weniger strukturiert. So ist beispielsweise kein Titelelement für eine solche Nachricht vorgesehen. Mit Blick auf die mögliche Übersetzung in eine einheitliche Repräsentation müssen diese Elemente aus den vorhandenen Informationen generiert und abgeleitet werden:

Anforderung #21: Fehlende Elemente im Quellformat (z.B. Titel) aus Metainformationen generieren.

Da es sich bei Microblogging um personenzentrierte Netzwerke handelt, ist die Angabe des Autors im Titelelement des internen Nachrichtenformates empfehlenswert. Zusätzlich könnten in diesem Titel das Thema, um das es sich in der Nachricht handelt, angegeben werden. Diese Information kann beispielsweise aus enthaltenen Hashtags oder aus nachträglich erkannten Metainformationen gewonnen werden.

In vielen Fällen werden „Twitter"- Nachrichten auch genutzt, um Empfehlungen für externe Links zu verschiedenen Webseiten abzugeben. Diese Empfehlungen besitzen darüber hinaus keinen weiteren Inhalt, so dass eine separate Behandlung derartiger Nachrichten empfehlenswert ist:

Anforderung #22: Twitter Nachrichten, die ausschließlich einen Link beinhalten als Empfehlung für externen Inhalt verarbeiten.

3.1.4.4 Communote

Die Microblogging- und Microsharing- Lösung „Communote" bietet für Zugriffe auf die Daten der Plattform eine „JSON"- Schnittstelle sowie personalisierte RSS 2.0-Feeds an. Für die Nutzung dieser Schnittstelle ist es erforderlich, dass sich der jeweilige Client vor einem Datenaustausch gegenüber der Schnittstelle authentifiziert. Als Parameter ist hierbei die Angabe von LoginId, Passwort und Authentifizierungstyp (z.B. MD5 kodiertes Passwort) erforderlich. Nachdem eine erfolgreiche Authentifizierung durchgeführt wurde, können die Inhalte, auf die die authentifizierte Person Zugang hat, aus der Plattform abgerufen werden. Analog zur Authentifizierung gegenüber der „JSON"- Schnittstelle erfolgt der Zugriff auf die personalisierten RSS-Feeds. Lediglich das Format des Passworts wird hier im Klartext als „HTTP GET"- Parameter übergeben.

„Communote" verwaltet eine Reihe von Blogs, die verschiedenen Nutzern zugänglich sein können. Jeder Blog hat einen Titel und ist durch Schlüsselwörter beschrieben, die Aufschluss darüber geben, welche Themen innerhalb des Blogs diskutiert werden. Den Inhalt eines Blogs bilden Posts, die aus folgenden Strukturelementen bestehen:

- Autor
- Schlüsselwörter (wobei ein Schlüsselwort auch aus mehreren Wörtern bestehen kann)
- Referenzen auf andere Nutzer der Plattform (spezielle Hashtags mit vorangestelltem „@" anstatt „#")
- Referenz auf einen anderen Post
- Inhalt der Nachricht
- Referenzen auf angehängte Dateien
- Zeitstempel der Publikation

Daraus folgt:

Anforderung #23: Referenzen auf Personen als Metainformationen ansehen und gesondert verarbeiten (z.B. Direktnachrichten bei einer Filterung bevorzugen).

3.1.5 Infrastruktur

Eine der zentralen Anforderungen an die prototypische Anwendung ist die Fähigkeit, verschiedene Nachrichten aus semistrukturierten Quellen zu aggregieren und sie den Nutzern bzw. einer Frontend Schnittstelle in personalisierter, gefilterter Form anzubieten. Darüber hinaus muss die Anwendung erweiterbar für andere Nachrichtenformate bleiben und sicherstellen, dass zusätzliche Formate mit geringem Implementierungsaufwand unterstützt werden können.

Um die Personalisierung zu realisieren, muss die Anwendung über eine Nutzerverwaltung verfügen. Dabei sollen Nutzer über einen externen Nutzerverzeichnisdienst authentifiziert werden:

Anforderung #24: Unterstützung externer Nutzerverzeichnisdienste.

Die Anwendung selbst hält darüber hinaus für jeden Nutzer ein Profil, um zusätzliche Informationen zu speichern. Jedem Nutzer muss die Möglichkeit geboten werden, seine persönlichen Informationsquellen anzugeben. Für den Prototyp ist diesbezüglich die Unterstützung von RSS-Feeds, Atom-Feeds, Twitter-Accounts und Communote-Accounts vorgesehen. Optional kann der Prototyp auch eine Importfunktion für „OPML"- Dateien unterstützen.

Die angegebenen Informationsquellen müssen durch den Prototyp aggregiert werden und zu einem Nachrichtenstrom mit einheitlichen Nachrichtenelementen zusammengefasst werden. Dabei ist zu beachten, dass diese inhaltlich verlustfrei übersetzt werden:

Anforderung #25: Verlustfreie Verarbeitung eingehender Inhalte.

Darüber hinaus muss der Prototyp vorhandene Metainformationen aus den eingehenden Nachrichtenelementen auslesen und in einer internen Repräsentation der aggregierten Inhalte den Nachrichten zuordnen. Zudem muss die Anwendung jede eingehende Nachricht auf bereits bekannte Schlüsselwörter prüfen. Werden Schlüsselwörter gefunden, die noch nicht als Metainformation gekennzeichnet sind, müssen diese in der internen Repräsentation der Inhalte der jeweiligen Nachricht zugeordnet werden.

Da der Prototyp ebenfalls über kollaborative Funktionen verfügen muss, sollte es dem Nutzer möglich sein, Nachrichtenquellen als privat zu definieren. Dies soll den Anforderungen des Nutzers an seine Privatsphäre gerecht werden, damit diese Nachrichtenquellen von den kollaborativen Funktionen, wie der Empfehlungskomponente, abgegrenzt werden können.

Auf Basis dieser Anforderungen ist es notwendig, dass der Prototyp alle bekannten Schlüsselwörter verwaltet und in einer Schlüsselwortliste speichert. Jedes Schlüsselwort, das in einer eingehenden Nachricht gekennzeichnet ist und sich nicht in der Schlüsselwortliste befindet, muss dieser hinzugefügt werden, damit dieses in Nachrichten, die zu einem späteren Zeitpunkt in das System eingehen, wiederverwendet werden kann:

Anforderung #26: Zentrale Verwaltung bekannter Schlüsselwörter.

Eine weitere Kernanforderung der Anwendung ist die personalisierte Filterung der eingegangenen Nachrichten. Um dieser Anforderung gerecht zu werden, muss die Anwendung ein Nutzermodel beinhalten. Dieses Modell sollte auf gewichteten Schlüsselwörtern zu jedem Nutzer basieren. Dabei muss das initiale Nutzermodell über die Nutzung von externen Diensten konfigurierbar sein. Das heißt, es muss dem Nutzer möglich sein, dem System die Zugangsdaten für seinen „Delicious"-Account zu übergeben. Anhand der bei „Delicious" hinterlegten gewichteten Schlüsselwörter muss das System ein Startprofil für den Nutzer anlegen können. Darüber hinaus muss es dem Nutzer möglich sein, manuell Schlüsselwörter zu seinem Nutzerprofil hinzufügen zu können und vorhandene Schlüsselwörter in ihrer Bewertung zu manipulieren:

Anforderung #27: Manuelle Justierung der Nutzerprofile ermöglichen.

Im Interesse einer übersichtlichen Darstellung der aggregierten Inhalte, muss es dem Nutzer möglich sein, frei definierbare Kategorien anzulegen. Diesen Kategorien müssen sowohl Nachrichtenquellen zugeordnet werden können, als auch einzelnen Schlüsselwörtern, die beschreiben, welche Inhalte dieser Kategorie zugeordnet werden sollen. Der Prototyp muss diese Kategorien selbstständig mit den passenden Nachrichtenelementen füllen und abrufbar machen.

Die Filterung der eingehenden Nachrichten basiert auf den in den Nachrichtenelementen erkannten Schlüsselwörtern und ihren entsprechenden Werten im Nutzermodell.

Im Sinne einer übersichtlichen Gestaltung und einer einfachen Bedienung im Frontend muss es dem Nutzer möglich sein, einzelne Nachrichtenelemente als gelesen oder geschlossen zu markieren. Die Anwendung muss diesbezüglich zu jedem Nutzer speichern, welche Nachrichten bereits gelesen und welche bereits geschlossen bzw. verworfen wurden. Darüber hinaus muss es dem Nutzer möglich sein, Nachrichten, die er aus Zeitgründen oder Ähnlichem noch nicht lesen wollte oder konnte, in einer „Noch zu lesen"-Liste zu speichern.

Neben der Filterung von Inhalten muss das System ebenfalls über kollaborative Funktionen verfügen. Im Einzelnen bedeutet dies, dass Nutzer einzelne Nachrichtenelemente bewerten können. Diese Bewertungen werden einerseits im Filterprozess verwendet, andererseits bilden sie die Basis für das Empfehlungssystem der Anwendung. Durch die Anforderung des Empfehlungssystems muss der Prototyp in der Lage sein, einem Nutzer Empfehlungen für verschiedene Artikel auszusprechen:

Anforderung #28: Personalisierte Empfehlungen ermöglichen.

Dabei müssen auf Basis der Nutzerprofile ähnliche Nutzer gefunden werden, deren positive Bewertungen dem ursprünglichen Nutzer als Lesevorschlag unterbreitet werden.

Optional kann der Prototyp über die Funktionalität verfügen, Nutzern die Möglichkeit zu geben, Nachrichtenelemente explizit einem anderen Nutzer des Systems zu empfehlen.

Da der Prototyp als Backend-Anwendung konzipiert ist, muss er seine Funktionalitäten durch eine Schnittstelle der entsprechenden Frontend Anwendung bereitstellen. Diese Schnittstelle muss den Zugang zu allen erläuterten Funktionen bereitstellen.

Anforderung #29: Funktionalität über eine Schnittstelle für ein Frontend bereitstellen.

Zusätzlich zu einer Schnittstelle muss der Prototyp eine Weboberfläche für administrative Funktionen anbieten, die die Anwendungsfälle aus 3.1.2 abdeckt (Vergleich Anforderung #6).

3.1.6 Zusammenfassung

In den vorrangegangen Abschnitten wurden die Anforderungen an die zu entwickelnde Infrastruktur beschrieben. Auf Basis der Anforderungen #1 bis #29 lassen sich folgende Kernanforderungen zusammenfassen:

- KA #1: Aggregation heterogener semistrukturierter Informationsquellen
- KA #2: Personalisierte Filterung von Nachrichten auf Basis von inhaltsbasierenden und kollaborativen Filtern
- KA #3: Erweiterbarkeit der Infrastruktur hinsichtlich weiterer Nachrichtenformate
- KA #4: Realisierung eines personalisierten Empfehlungssystems für verwaltete Nachrichten
- KA #5: Automatische Generierung von Metainformationen zu Nachrichten (Zuordnung von Schlüsselwörtern zu Inhalten)

3.2 Entwurf und Implementierung

3.2.1 Umgebungsmodell

Die zu entwickelnde Infrastruktur aggregiert Informationsströme aus verschiedenen Quellen. Um dies zu gewährleisten, bietet der Prototyp für die Nachrichtenquellen „RSS“/ „Atom“, „Twitter“ sowie „Communote“ jeweils eine Schnittstelle. Diese Schnittstellen basieren auf einem „pull“-Prinzip, das in regelmäßigen Intervallen die durch die Nutzer des Systems angegebenen Quellen auf neue Inhalte überprüft. Die dadurch entstehenden Nachrichtenströme werden durch den Prototyp aggregiert, analysiert und auf Basis von Nutzerprofilen gefiltert. Ist dieser Prozess abgeschlossen, wird der personalisierte Nachrichtenstrom an die Ausgabeschnittstelle weitergegeben und kann von einem externen Client bzw. einem Frontend abgerufen werden. Die nachfolgende Abbildung 21 des Umgebungsmodells bzw. Kontextmodells [Aßmann+06] visualisiert die ein- und ausgehenden Datenströme:

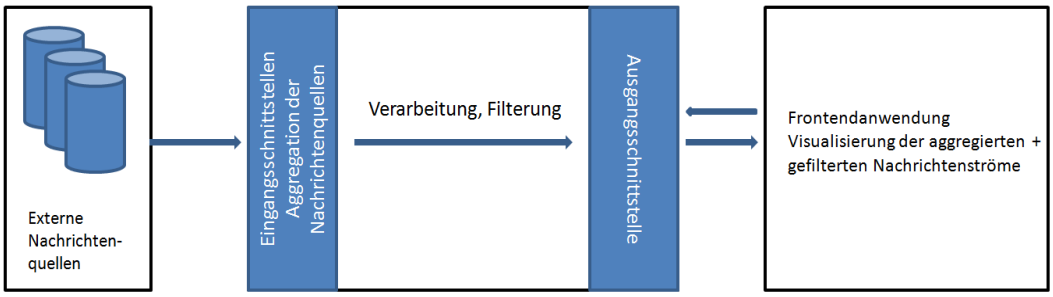

Abbildung 21: Übersicht des Umgebungsmodells

Die Kommunikation mit der Frontendanwendung basiert auf einem klassischen synchronen Interaktionsmodell. Die Frontendanwendung stellt Anfragen an die Ausgabeschnittstelle des Prototyps, welche nach der Bearbeitung durch die Anwendungslogik beantwortet werden. Darüber hinaus sendet das Frontend Rückmeldungen in Form von Anfragen, die mit einem oder mehreren Parametern versehen sind. Diese Parameter werden durch den Prototyp ausgewertet und verarbeitet.

3.2.2 Top-Level Architektur

Eine detailliertere Sicht auf die Architektur der Infrastruktur zeigt Abbildung 22 mit der Top-Level Architektur [Aßmann+06]. Die Komponente „Connectoren“ umfasst eine Reihe von Diensten, die Inhalte aus externen Nachrichtenquellen in periodischen Abständen auslesen. Die ausgelesenen Inhalte werden im Anschluss an die Komponente „Datenmanagement“ übergeben. Diese Komponente besteht ebenfalls aus einer Reihe von Diensten, die die Zugriffe auf die verschiedenen Entitäten des Systems realisieren und diese persistent in einer Datenbank speichern.

Die Komponente „Filterengine“ bedient sich der Inhalte aus den Komponenten „Datenmanagement“ und „Nutzermodell“. Sie realisiert die Filterung und Empfehlung von aggregierten Nachrichten. Dabei kann diese Komponente durch die Anwendungskonfiguration konfiguriert werden. Ist eine personalisierte Filterung mit Hilfe der „Filterengine“ durchgeführt, werden die durch das Frontend über die Ausgabeschnittstelle angeforderten Daten ausgeliefert:

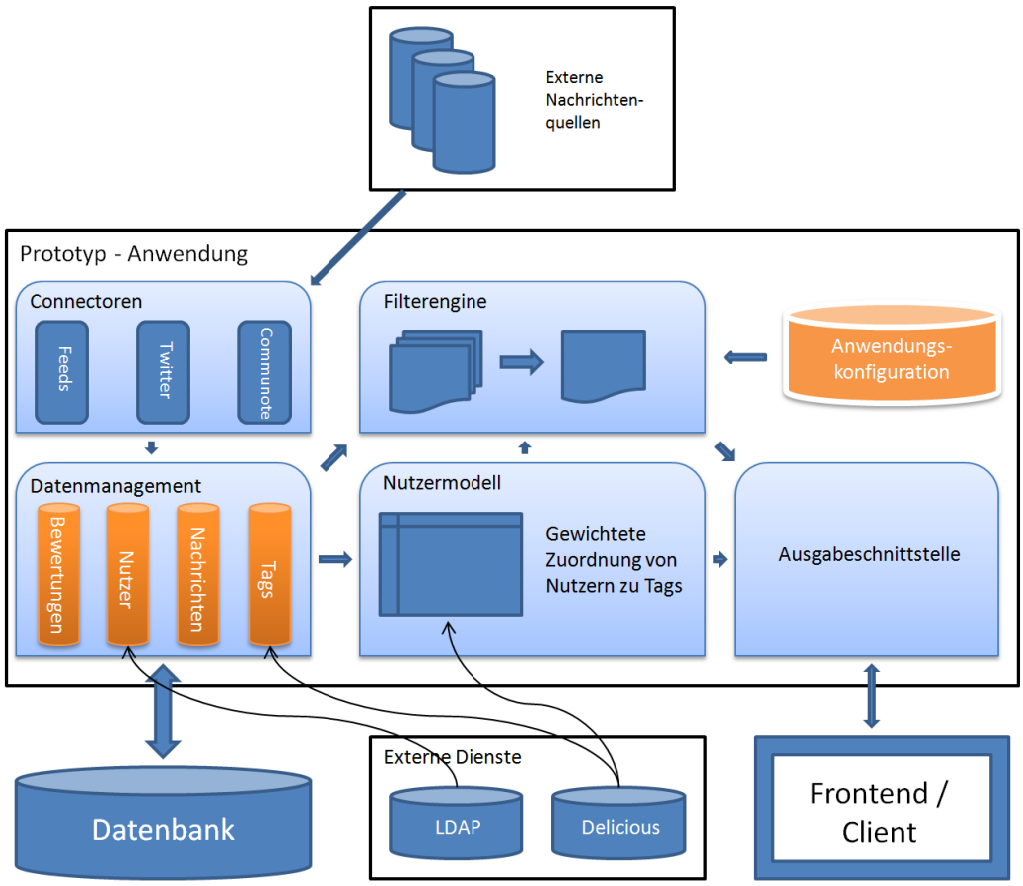

Abbildung 22: Top-Level Architektur

3.2.3 Nutzermodell

Die Grundlage für den Filterprozess bildet das Nutzermodell, das die Interessen aller Nutzer widerspiegelt. Dieses kann durch eine Matrix aus Nutzern i und Schlüsselwörtern j dargestellt werden, wobei die Werte der Matrix $v_{i,j}$ das Nutzerinteresse für ein Thema bzw. Schlüsselwort repräsentieren. Um das in Kapitel 2.1.5.4 beschriebene „New User“-Problem zu lösen, wird der „social-bookmarking“-Dienst „Delicious“ [DELICIOUS] genutzt. Verfügt der Nutzer über ein Profil in diesem Dienst, sollte er seine Zugangsdaten dem System übergeben. Dieser Dienst verwaltet unter Anderem die vom Nutzer häufig genutzten Schlüsselwörter und gibt Aufschluss über dessen Nutzerpräferenzen. Aus diesen Informationen kann ein Nutzerprofil abgeleitet werden, ohne dass der Nutzer manuell seine Präferenzen

angeben muss. Tabelle 7 gibt einen beispielhaften Überblick über ein derartiges Nutzermodell:

	Java	Microsoft	XML	MDA	MySQL
Nutzer 1	0,3	0,7	0,9	-	0,3
Nutzer 2	0,1	0,1	0,7	0,8	0,2
Nutzer 3	-	0,9	0,7	-	-
Nutzer 4	0,2	0,6	0,9	-	0,4

Tabelle 7: Beispieltabelle Nutzermodell

Dabei ist zu beachten, dass es sich hierbei um ein dynamisches Nutzermodell handelt (vgl. Kapitel 2.1.4). Durch explizite Bewertungsverfahren werden die Werte der Matrix aktualisiert. Auslöser für eine solche Aktualisierung ist beispielsweise eine vom Nutzer abgegebene Bewertung zu einem Artikel. Fällt diese positiv aus, wird der Wert für jedes im Artikel vorkommende Schlüsselwort in der Nutzermodellmatrix für den aktuellen Nutzer erhöht. Analog dazu werden bei einer negativen Bewertung die entsprechenden Werte gesenkt. Zusätzlich zu dieser automatischen Anpassung haben Nutzer die Möglichkeit, ihr Nutzerprofil aus dem Nutzermodell einzusehen und manuell anzupassen bzw. zu korrigieren. Diese Funktion ist notwendig, damit das System auf die Anforderungen von Nutzern eingehen kann, die ein sich häufig änderndes Interesse besitzen. Dies könnte zum Beispiel in einem Projektleiterszenario [CANBOLAT+09] der Fall sein.

3.2.4 Internes Nachrichtenformat

Im Rahmen des Entwurfs des Filtermechanismus wurde entschieden, dass die Filterung der eingehenden Nachrichtenströme nicht separat für jedes einzelne Format erfolgen soll, sondern auf Basis einer einheitlichen systemeigenen Repräsentation der eingegangenen Nachrichten. Diese Entscheidung wurde aus den folgenden Gründen getroffen:

- Vereinfachung des Entwurfs und der Implementierung des Filtermechanismus, da dieser nur für das interne Format umgesetzt werden muss
- Entwicklungsaufwand sinkt
- Erweiterbarkeit bezüglich der Unterstützung weiterer Nachrichtenformate wird verbessert und vereinfacht

Langfristig ist es denkbar, dass neben den für den Prototyp vorgesehenen Nachrichtenformaten auch weitere Formate integriert werden sollen. In einem solchen Fall müsste bei einer Erweiterung um ein Format lediglich ein Connector und ein Übersetzer für das entsprechende Format implementiert werden und nicht zusätzlich ein weiterer Filtermechanismus konzipiert werden. Entscheidend für die Wahl eines systemeigenen Formates ist jedoch die Tatsache, dass sich die meisten

betrachteten Formate ohne nennenswerte Informationsverluste in ein einheitliches Format überführen lassen.

Nachfolgend wird die Struktur des systemeigenen Nachrichtenformates erläutert. Nachdem die Inhalte aus den heterogenen externen Quellen aggregiert wurden, werden diese zunächst auf ein Java-Objektmodell übertragen. Dieses Objektmodell gliedert sich in die Entitäten „InformationSource“, „InformationElement“, “Author“, “Link“, „ElementTagRelation“ und „Tag“. Das Objekt „InformationSource“ repräsentiert dabei eine Nachrichtenquelle mit den entsprechenden Attributen. In diesen Attributen werden neben Zugangsdaten ebenfalls eine Reihe von Metadaten abgelegt, die bei der Analyse der Nachrichtenquelle durch das System gewonnen werden konnten.

Die ausgelesenen Inhalte einer Nachrichtenquelle werden durch die Entität „InformationElement“ repräsentiert. Zusätzlich zu Inhalt und Titel eines Artikels werden weitere Metainformationen gespeichert. Das Attribut „uid“ enthält einen universellen Identifier, mit dessen Hilfe der Artikel in der ursprünglichen Quelle eindeutig identifiziert werden kann. Dieser Identifier ist notwendig, um bereits bekannte von neuen Artikeln in den externen Quellen unterscheiden zu können. Die Entitäten „Author“ und „Link“ sind separat modelliert worden, um einen einfacheren und schnelleren Zugriff auf Querverweise zu erreichen. So können beispielsweise Artikel eines bestimmten Autors über eine einfache Relation abgefragt werden und über unterschiedliche Quellen hinweg nachvollzogen werden. Die Zuordnung von Nachrichtenelementen zu Schlüsselwörtern (Tags) erfolgt über die Entität „ElementTagRelation“. Diese Beziehung wurde auf diese Weise modelliert, um die Grundlage für spätere Erweiterungen bzw. Verbesserungen des Systems zu legen. Derzeit ist eine binäre Zuordnung von Schlüsselwörtern zu Nachrichten vorgesehen, die durch diese Relation abgebildet werden kann. Für eine spätere Weiterentwicklung ist es denkbar, dass nicht allein das reine Auftreten eines Schlüsselwortes registriert wird, sondern zusätzlich eine Gewichtung des Schlüsselwortes zu einem Inhalt errechnet wird. Um die Grundlage für eine persistente Speicherung dieser Gewichtungswerte zu legen, wurde in der Entität „ElementTagRelation“ das Attribut „value“ eingeführt. Mit Blick auf die Anforderungen an den Prototyp wird jedoch auf eine feingranulare Gewichtung von Schlüsselwörtern zu Nachrichten verzichtet.

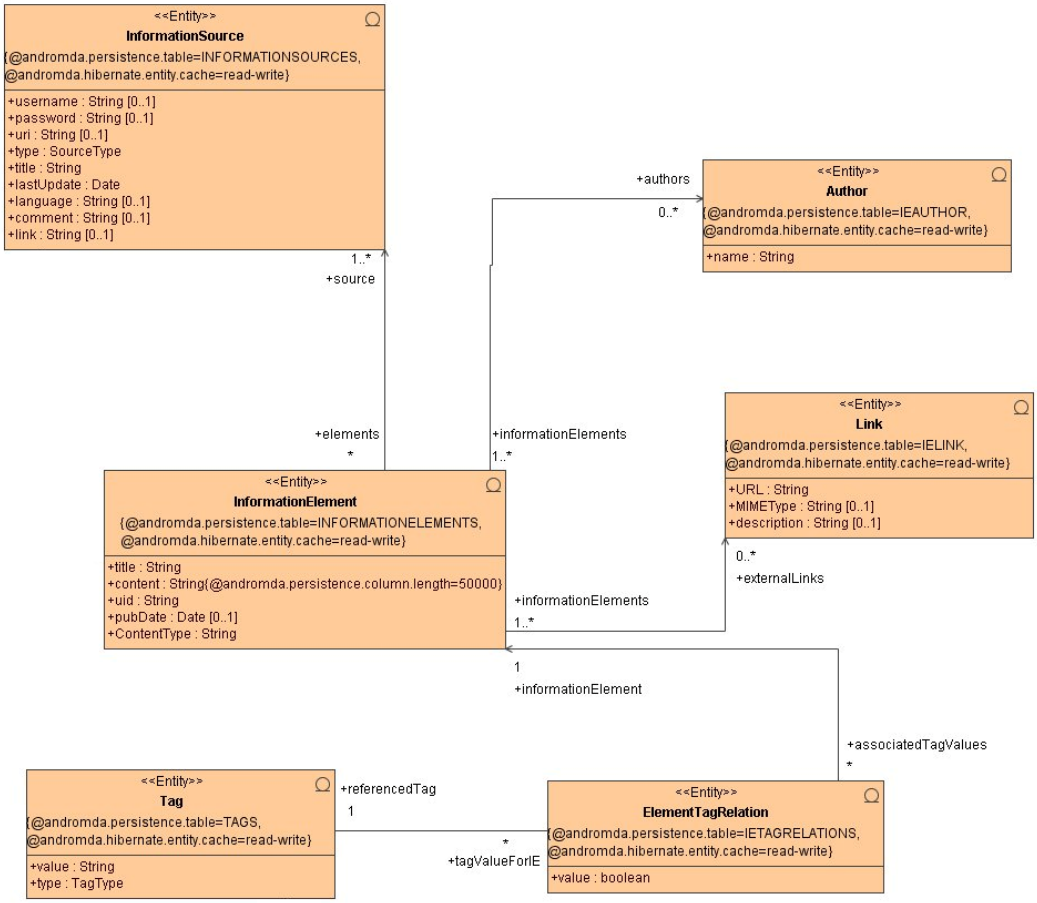

Abbildung 23: Auszug des Klassendiagrammes - Internes Nachrichtenformat

Aus Abbildung 23 wird ersichtlich, dass alle von den Nutzern abonnierten Nachrichtenquellen und deren Inhalte persistent gespeichert werden und nicht „on demand“ bei einem Zugriff aus der Quelle geladen werden. Die Entscheidung gegen eine „on demand“- Lösung wurde getroffen, da das System sehr häufig auf diese Daten zugreifen muss. Der entscheidende Nachteil einer „on demand“- Lösung ist, dass bei jedem Zugriff auf eine Nachrichtenquelle oder einen Artikel eine Verbindung zur externen Quelle aufgebaut und der Inhalt aus dieser Quelle geladen werden muss. Zusätzlich hätte ein solches Vorgehen zur Folge, dass die abgerufenen Nachrichten bei jedem Zugriff in das systemeigene Nachrichtenformat übersetzt werden müssten. Dieser Ablauf stellt somit hohe Anforderungen an die Bandbreite der Netzanbindung sowie an die Rechenleistung des Rechners, der die Übersetzung in das interne Nachrichtenformat ausführt. Aus den genannten Gründen wurde entschieden, die Inhalte der abonnierten Quellen persistent im System zu speichern, auch wenn dies bei steigender Anzahl der Nutzer und Abonnements hohe Anforderungen an die Speicherkapazität des Systems stellt.

3.2.5 Datenhaltung

Um die in den vorangegangen Abschnitten beschriebenen Funktionen zu realisieren, ist der Aufbau eines umfassenden Nutzermodells sowie eines detaillierten Nachrichtenmodells notwendig.
Dabei muss das System für einen Benutzer folgende Daten speichern:

- Identifier
- Name
- „LDAP"- Login
- Email-Adresse
- Referenz zu jeder Nachrichtenquelle mit ggf. Authentifizierungsinformationen
- Persönliche Schlagwörter mit Gewichtung
- Zugangsdaten zu externen Diensten (z.B. „Delicious")
- Eigene Kategorien
- Abgegebene Bewertungen
- Referenzen zu gelesenen und geschlossenen Nachrichten sowie zu Nachrichten, die vom Nutzer auf die „noch zu lesen"- Liste verschoben wurden

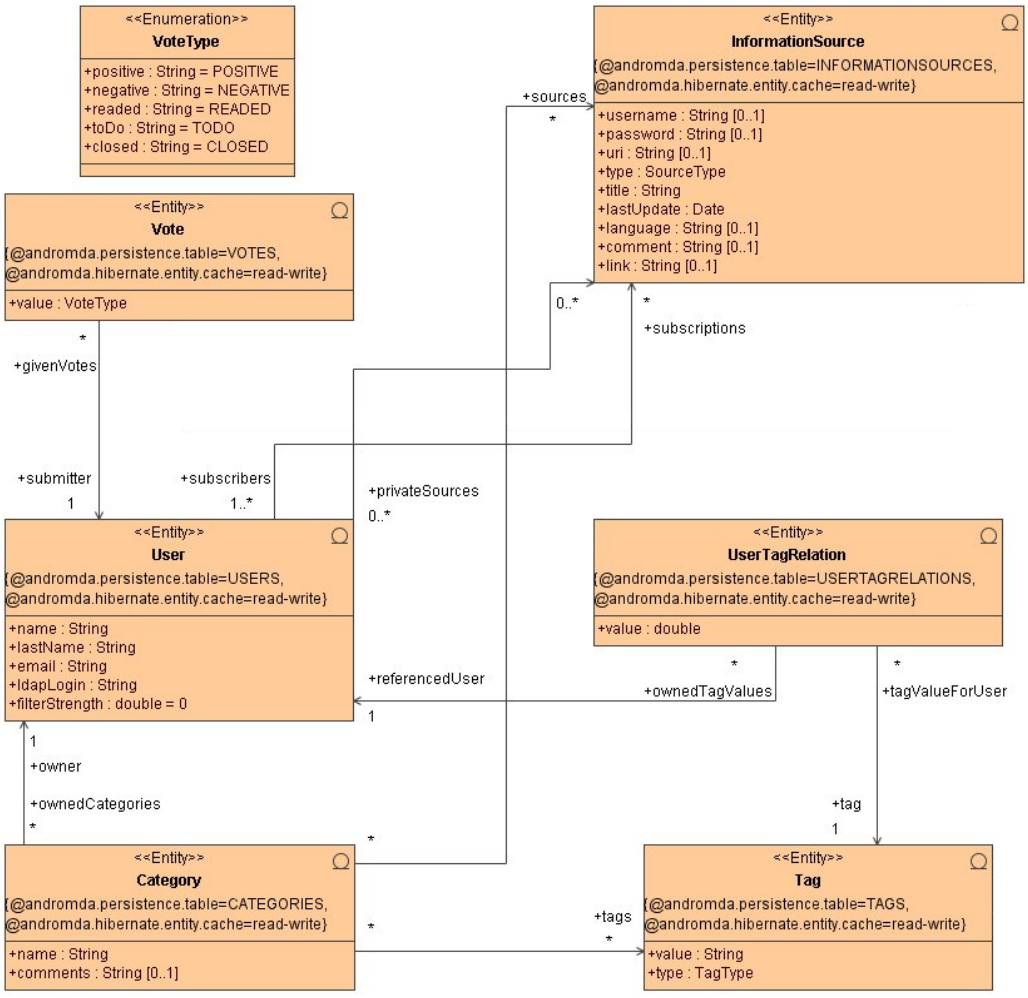

Abbildung 24: Auszug des Klassendiagrammes – Nutzer-Entitäten

Bezugnehmend auf die Klassifikation von Nutzermodellen aus Kapitel 2.1.4 kann das vorgesehene Nutzermodell für den Prototyp als ein weitgehend implizites, dynamisches und sichtbares „long-term“- Nutzermodell eingeordnet werden.
Darüber hinaus müssen zu jeder Nachrichtenquelle folgende Daten gespeichert werden:

- Identifier
- Typ der Nachrichtenquelle („Feed“, „Twitter“, „Communote“)
- Evtl. Login-Daten und Pfade zur Nachrichtenquelle
- Referenzen zu Nutzern, die die Nachrichtenquelle abonniert haben
- Zeitstempel der letzten Aktualisierung
- Referenzen zu allen untergeordneten Nachrichtenelementen
- Optional: Schlagwörter, die diese Nachrichtenquelle beschreiben
- Autor, Titel und Link
- Referenzen auf zugeordnete Nutzerkategorien

Aufbauend auf die Definition der Felder, die für jede Nachrichtenquelle angegeben sein müssen, ist es notwendig, folgende Daten für jede Nachricht zu speichern:

- Identifier
- Titel, Autor(en), Inhalt, Links
- Metainformationen (Tags, Publikationszeitpunkt)

3.2.6 Filtermechanismus

Für die Filterung von Nachrichtenströmen können einige der unter 2.1.5 vorgestellten Verfahren nur in begrenztem Umfang verwendet werden. So beruht beispielsweise das inhaltsbasierende Filtern auf der Tatsache, dass die einzelnen zu filternden Elemente bzw. Nachrichten nach verschiedenen Kriterien bewertet sein müssen. Eine solche Kategorisierung wird in der Regel durch Nutzer oder durch eine separat betriebene Redaktion ausgeführt und steht der Anforderung einer weitgehend automatisierten Filterung entgegen. Daher ergibt sich die Problematik, diese Ressourcen automatisch und korrekt zu bewerten und zu kategorisieren. Hierzu könnten vorhandene Tags genutzt werden. Im Detail bedeutet dies, dass durch das System und externe Dienste, wie beispielsweise „Delicious“ [DELICOIUS], „OpenCalais“ [OPENCALAIS] oder „Communote“ [COMMUNOTE], alle bekannten Schlagwörter bzw. Tags in einem „Tag Pool“ aufgezeichnet und abgelegt werden.
Die Inhalte von Quellen, die Tags nicht nativ unterstützen, sollten daher auf bekannte Tags aus dem „Tag-Pool“ durchsucht werden. Werden Übereinstimmungen gefunden, sollte der entsprechende Tag dem Inhalt zugeordnet werden, um ihn näher zu beschreiben und um ihn durch das System leichter filtern zu lassen. Anders als beim klassischen inhaltsbasierenden Filtern entsteht auf diese Weise eine Bewertung von Inhalten mit einem Wertebereich von {0,1}. Um diese Bewertungen genauer zu gestalten, kann dieses System erweitert werden. Möglich

wäre die Berechnung der Frequenz des Terms innerhalb des Dokuments, um die Wichtigkeit des Terms in Bezug auf das Dokument abzuleiten. So könnte es möglich sein, vollautomatisch Bewertungen für Inhalte zu erstellen, die in ihrer Ursprungsform weder kategorisiert, noch mit Metadaten versehen waren. Ist eine solche automatische Bewertung vorgenommen worden, könnte das Verfahren der inhaltsbasierten Filterung auf diese Quellen angewendet werden. Dabei ist jedoch zu beachten, dass der in Kapitel 2.1.5.1 beschriebene Algorithmus bzw. das Vektorraum-Modell nicht analog von Recommender-Systemen auf die Filterung von Nachrichtenströmen übernommen werden kann. Die Ursache hierfür liegt in der Berechnung von Ähnlichkeiten zwischen zwei Ressourcen. Während es bei Recommender-Systemen der Zielstellung entspricht, möglichst ähnliche Ressourcen innerhalb einer Gesamtmenge zu identifizieren, gilt diese Anforderung für die Berechnung zwischen Nutzern und Nachrichtenartikeln nur bedingt. Die nachfolgende Tabelle 8 zeigt einen Nutzer mit seinen entsprechenden Daten aus dem Nutzermodell, sowie zwei exemplarische Nachrichtenelemente mit den für sie ermittelten Werten:

Ressource	Java	XML	Microsoft	MDA
Nutzer #1	8	5	2	6
Artikel #1	6	-	2	-
Artikel #2	2	-	-	1

Tabelle 8: Schwächen des Vektorraum-Modells

Nach der Berechnungsvorschrift für das Vektorraum-Modell aus Kapitel 2.1.5 errechnet sich in diesem Fall eine sehr hohe Relevanz des Artikels #1 für den Nutzer #1, sowie eine sehr niedrige Relevanz des Artikels #2 für Nutzer #1. Hierbei ist festzustellen, dass dies für Artikel #2 nicht zwangsläufig den Tatsachen entsprechen muss. Nutzer #1 hat offenbar ein sehr hohes Interesse an der Programmiersprache „Java“ und an Artikeln die sich mit der „MDA“ befassen. Artikel #2 befasst sich mit genau diesen relevanten Themen, auch wenn die Schlagwörter, die auf diese Tatsache hinweisen, im Text nicht hochfrequent auftreten. Nach dem beschriebenen Vektorraummodell wird dieser Artikel als nicht relevant klassifiziert und gefiltert. Tatsächlich jedoch hat der Nutzer höchstwahrscheinlich ein Interesse an diesem Artikel, da er sich mit den für ihn relevanten Themen befasst. Aus dieser Analyse ergibt sich die Notwendigkeit, das Verfahren des inhaltsbasierenden Filterns hinsichtlich der beschriebenen Anwendungsfälle anzupassen, um eine möglichst hochwertige Filterung zu erzielen.

Zusätzlich zu den bereits beschriebenen Problemen des Vektorraum-Modells in Bezug auf eine Filterung von Nachrichten ergibt sich aus der vorangegangenen Analyse der Nachrichtenformate ein weiteres Problem. Grundlage für das fehlerfreie Arbeiten des Vektorraum-Modells ist die möglichst präzise Bewertung der Ressourcen bzw. der eingegangenen Nachrichten. Wie bereits beschrieben ist es denkbar, diese Nachrichten auch im Nachhinein mit Schlüsselwörtern zu beschreiben, jedoch müssen diese automatisch vergebenen Schlüsselwörter ebenfalls gewichtet werden, um ihre Wertigkeit für das jeweilige Dokument anzugeben. Die

automatische Ermittlung dieser Wertigkeiten muss dabei auf jede Art von Informationsquellen anwendbar sein und unabhängig von den Eigenschaften dieser Quellen vergleichbare Ergebnisse liefern. Während der Analyse der Nachrichtenformate „Twitter“, „RSS“/ „Atom“ und „Communote“ wurde deutlich, dass eine solche automatische Bewertung der enthaltenen Schlüsselwörter kaum möglich ist. Die Randbedingungen, wie Informationsdichte oder Textlänge, unterscheiden sich zwischen, sowie teilweise innerhalb dieser Quellen in hohem Maße, so dass es kaum möglich erscheint, diese Schlüsselwörter mit automatisch erzeugten Wertigkeiten zu beschreiben. Mitunter ist es schwierig, überhaupt eine Wertung für ein Schlüsselwort bezüglich einer Nachricht zu erstellen. Dies liegt unter anderem auch in der Tatsache begründet, dass eine solche Wertung sehr subjektiv ist und es in vielen Fällen nur schwer möglich ist, Nachrichten und deren Schlüsselwörter automatisch auf einer objektiven Skala zu beschreiben. Aus den genannten Gründen kommt für die Bewertung von Nachrichten nur das reine Auftreten von Schlüsselwörtern in Betracht. Das heißt, Nachrichten werden für jedes bekannte Schlüsselwort mit einer binären Bewertungsskala versehen, die Aussagen darüber liefert, ob der Inhalt der Nachricht durch dieses Schlüsselwort beschrieben werden kann oder nicht. Im Gegensatz zur Bewertung von Nachrichten sollte das Nutzerprofil, das aus einer Menge bewerteter Schlüsselwörter besteht, feingranularer angelegt werden. Diese feinere Untergliederung ist notwendig, um die unterschiedlichen Wertigkeiten der Nutzerpräferenzen möglichst genau voneinander abzugrenzen. Hierzu kann eine Bewertungsskala von 0 bis 1 angelegt werden, wobei 0 ein niedriges Interesse und 1 ein außerordentlich hohes Interesse an einem Schlüsselwort bzw. Thema widerspiegelt. Nachfolgend wird der aus den bereits genannten Erkenntnissen resultierende Filtermechanismus vorgestellt.
Wie eingangs beschrieben, sollten diese Schlüsselwörter den Nachrichten automatisch hinzugefügt werden. Das Produkt dieses Autotagging-Mechanismus ist eine Matrix, bestehend aus Nachrichten und Schlüsselwörtern, wobei die Werte der Matrizenelemente ausschließlich 0 oder 1 annehmen können. Die nachfolgende Tabelle 9 gibt eine beispielhafte Übersicht über eine derartige Matrix:

	Java	Microsoft	XML	MDA	MySQL
Nachricht 1	1	0	0	1	1
Nachricht 2	0	1	0	1	1
Nachricht 3	0	1	1	0	0

Tabelle 9: Beispieltabelle Bewertung von Nachrichten

Für die Berechnung der Relevanz einer Nachricht für einen Nutzer werden alle Schlüsselwörter, die in der Nachricht vorkommen und die eine Bewertung im Nutzerprofil des Nutzers besitzen, herangezogen. Andere Schlüsselwörter werden nicht betrachtet. Bezüglich der Relevanzberechnung von Nachricht 1 zu Nutzer 1 aus Kapitel 3.2.3 und Tabelle 7 bedeutet dies, dass die Schlüsselwörter „Java“ und „MySQL“ in die Berechnung einfließen. Das Schlüsselwort „MDA“ wird ignoriert, da vom Nutzer keine Bewertung für dieses Schlüsselwort vorliegt und davon ausgegangen wird, dass der Nutzer dieses nicht kennt. Die Berechnung erfolgt nach

folgender Vorschrift, wobei R die durchschnittliche Relevanz der Schlüsselwörter der Nachricht für den Nutzer angibt, t die Bewertung des Schlüsselwortes k im Nutzerprofil darstellt und n die Anzahl der herangezogenen Schlüsselwörter widerspiegelt:

$$\mathrm{R}_{\mathrm{Nutzer,Nachricht}} = \frac{\sum_{k=1}^{n} \mathrm{t_k}}{n}$$

Für Nutzer 1 aus Tabelle 7 ergeben sich für die in Tabelle 9 dargestellten Nachrichten folgende Relevanzwerte $R_{Nutzer,Nachricht}$.

$R_{1,1}$ = 0,3 $R_{1,2}$ = 0,5 $R_{1,3}$ = 0,8

Basierend auf dem Wertebereich des Nutzermodells ergibt sich für die errechneten Relevanzen ebenfalls ein Wertebereich von 0 für eine niedrige Relevanz bis 1 für eine hohe Relevanz.

Im Interesse, auch subjektive Bewertungen in die Filterung einfließen zu lassen, kommt mit dem kollaborativen Filtern ein weiteres Verfahren für die Filterung von Nachrichtenströmen in Betracht. Basis für dieses Verfahren bilden Bewertungen von Nutzern zu den jeweiligen Ressourcen bzw. Nachrichten. Demnach sollten Nutzer des Systems die Artikel, die sie gelesen haben bewerten, um dem System Rückmeldung über die Qualität des Inhalts aus ihrer Sicht zu geben. Dieses Verfahren kann ohne Weiteres auf Nachrichtenströme angewendet werden. Voraussetzung ist jedoch, dass durch das Frontend Rückmeldung über die gelesenen Inhalte gegeben wird.

Die erhaltenen Bewertungen könnten einerseits dazu genutzt werden, um Nutzern Empfehlungen über interessante Artikel zu geben, aber auch dafür verwendet werden, den bereits beschriebenen Filtermechanismus zu erweitern. Im Einzelnen bedeutet dies, dass nach der Berechnung der inhaltsbezogenen Relevanz einer Nachricht für einen Nutzer ähnliche Nutzer des Systems gesucht werden. Dabei ist zu beachten, dass für eine Ähnlichkeitsanalyse zwischen zwei Nutzern einige Vorbedingungen erfüllt sein müssen. Grundvoraussetzung für die Berechnung der Ähnlichkeit ist, dass die Schnittmenge der Schlüsselwörter der Nutzerprofile einen bestimmten Prozentsatz des gesamten jeweiligen Nutzerprofils besitzt. Damit wird verhindert, dass Ähnlichkeitsanalysen durchgeführt werden, wenn Nutzer nur eine sehr geringe Schnittmenge an Interessen besitzen. Für eine qualitativ hochwertige Filterung ist es notwendig, dass die Schnittmenge der Interessen möglichst groß ist. Ist diese Voraussetzung gegeben, kann eine Ähnlichkeitsanalyse durchgeführt werden. Hierzu steht eine Reihe von Algorithmen zur Verfügung, die in der Anwendungskonfiguration ausgewählt und konfiguriert werden können. Ist eine Ähnlichkeitsanalyse durchgeführt und sind ähnliche Nutzer gefunden worden, kann anschließend geprüft werden, ob diese bereits Bewertungen für Nachrichten abgegeben haben, die der Ausgangsnutzer abonniert hat. Wird eine solche Bewertung gefunden und fällt diese positiv aus, könnte das System schlussfolgern, dass es sich hierbei um eine besonders hochwertige Nachricht handelt und das Ergebnis der

Relevanzberechnung mit einem Bonus versehen. Für das bereits betrachtete Beispiel aus den Tabellen 7 und 9 bedeutet dies, dass das System errechnet, dass sich die Nutzer 1 und 4 ähnlich sind. Darauf aufbauend sollte das System prüfen, ob Nutzer 4 Bewertungen für die Nachrichten 1, 2 und 3 abgeben hat. Ist dies der Fall und ist beispielsweise die Bewertung von Nachricht 2 positiv, könnte die errechnete Relevanz $R_{1,2}$ mit einem Bonus versehen und erhöht werden. Für den umgekehrten Fall einer negativen Bewertung ist es darüber hinaus denkbar, die entsprechende Relevanz mit einer Strafe zu versehen und zu senken.

Im Detail wird die Berechnung der Werte der Boni (B) und Strafen (S) nach den folgenden Vorschriften durchgeführt:

$$\text{Bonus: } B = F_B * (1 - R)$$

Ein Bonus B errechnet sich aus dem Produkt eines Faktors F_B (Wertebereich 0 bis 1) zur Bestimmung der Bonusstärke, der in der Anwendungskonfiguration einstellbar ist, mit der Differenz von 1 und R, der errechneten Relevanz der Nachricht. Somit wird sichergestellt, dass keine statischen Boni, sondern dynamisch angepasste Boni anhand des vorhandenen Relevanzwertes vergeben werden.

Ähnlich der Berechnung von Boni erfolgt die Berechnung von Strafen im Fall von negativen Bewertungen für eine Nachricht.

$$\text{Strafe: } S = F_S * R$$

Eine Strafe S errechnet sich aus dem Produkt von einem Faktor F_S (Wertebereich 0 bis 1) zur Bestimmung der Strafenstärke, der in der Anwendungskonfiguration einstellbar ist und der bereits errechneten Relevanz R.
Alternativ könnte der Faktor F_S bzw. F_B dynamisch anhand des Ähnlichkeitswertes des Nutzers, der die entsprechende Bewertung abgegeben hat gewählt werden und nicht statisch in der Anwendungskonfiguration festgelegt werden.
Sowohl Boni als auch Strafen werden der Ausgangsrelevanz R angerechnet. Das heißt, Boni werden mit der Relevanz addiert und Strafen subtrahiert. Liegen mehrere positive Bewertungen von ähnlichen Nutzern vor, ist die Berechnung der neuen Relevanz als Iteration zu sehen. Im Detail bedeutet dies, dass nachdem ein Bonus zu einer Ausgangsrelevanz addiert wurde, das Ergebnis wiederum als Ausgansrelevanz für eine erneute Berechnung eines Bonus verwendet wird. Analog ist die Iteration bei mehreren negativen Bewertungen bzw. Strafen zu betrachten.
Sollten für eine Nachricht sowohl negative als auch positive Bewertungen vorliegen werden diese in ihrer Anzahl von einander abgezogen. Sollte beispielweise eine Nachricht von ähnlichen Nutzern drei positive und zwei negative Bewertungen aufweisen, fließt lediglich eine positive Bewertung in die Relevanzberechnung ein.
Neben der Verwendung von Empfehlungen zur Verbesserung des Ergebnisses des Filterprozesses kann diese Funktion auch dazu genutzt werden, Nutzern Empfehlungen für Nachrichtenelemente oder Nachrichtenquellen auszusprechen, die sie selbst nicht abonniert haben. Somit erhält der Nutzer auch Zugang zu

Nachrichtenquellen, die ihm noch nicht bekannt sind, aber dennoch von Interesse für ihn sein können.
Zusätzlich zu den beschriebenen Filter- und Empfehlungsverfahren wird es dem Nutzer ermöglicht, seine verschiedenen Nachrichtenquellen in frei definierbare Kategorien einzuordnen, so dass der aggregierte Nachrichtenstrom bei der Anzeige übersichtlich geordnet und sortiert werden kann. Zudem erscheint es sinnvoll, diese Kategorien mit Schlüsselwörtern zu versehen, so dass es möglich ist, einzelne Nachrichten aus verschiedenen Quellen, die ein bestimmtes Schlüsselwort beinhalten in einer Kategorie zu aggregieren.

3.2.7 Aufbau der Anwendung

Die zu entwickelnde Infrastruktur basiert auf einer 3-Schichtenarchitektur und untergliedert sich in die Ebenen Präsentationsschicht, Serviceschicht und Persistenzschicht. Die Entscheidung für eine solche Architektur basiert auf den Anforderungen an die Infrastruktur sowie auf den Vorteilen, die eine 3-Schichtenarchitektur aufweist. Durch die geringe Kopplung der einzelnen Schichten wird ein hoher Grad an Wiederverwendbarkeit und Erweiterbarkeit der einzelnen Schichten garantiert [BALZERT+99]. Mit Blick auf die Anforderungen der Anwendung müssen die Funktionen der Serviceschicht bzw. der Anwendungslogik für unterschiedliche Präsentationsimplementierungen verfügbar sein. Das heißt, diese Funktionen und Daten müssen für das von Serkan Canbolat [Canbolat+09] entwickelte Frontend sowie für die separate Administrationsoberfläche verfügbar sein. Daher ist die Trennung von Präsentations- und Serviceschicht ein wünschenswertes Ziel.

Zentraler Bestandteil der Anwendung ist die Serviceschicht, die durch ihre verschiedenen Dienste die Anwendungslogik kapselt. Neben Diensten zur Verwaltung der Entitäten des Systems, existieren ebenfalls Dienste für die Kommunikation mit externen Diensten und Nachrichtenquellen sowie für die Filterung der aggregierten Inhalte. Durch die Dienste zur Verwaltung der Entitäten wird die Verbindung zur Persistenzschicht realisiert. Ausschließlich diese Klassen haben Zugriff auf den Funktionsumfang der Persistenzschicht.

Dabei gewährleistet die Persistenzschicht die Zugriffe auf das zugrunde liegende Datenbanksystem und hält Funktionen für die Verwaltung der Datenbankinhalte und die Konvertierung von Objekten aus dem Anwendungsraum auf die Entitäten der Datenbank bereit.

3.2.7.1 Connectoren

Connectoren sind Dienste, die Inhalte aus externen Nachrichtenquellen wie Web-Feeds, „Twitter“ oder „Communote“ aggregieren und die Inhalte auf ein internes Format übertragen. Dabei existiert für jedes Nachrichtenformat ein separater Dienst, der diese Funktionalität gewährleistet. Durch das Konzept der Connectoren wird die einfache Erweiterbarkeit um weitere Nachrichtenformate sichergestellt. In einem solchen Fall muss lediglich ein formatspezifischer Connector implementiert werden.

Alle anderen Komponenten des Systems, wie beispielsweise das Objektmodell oder der Filtermechanismus, benötigen nur minimale Anpassungen, um ein weiteres Nachrichtenformat zu unterstützen. Wie bereits angedeutet unterstützt die Anwendung die Nachrichtenquellen Web-Feeds, „Twitter“ und „Communote“. Nachfolgend werden die Eigenschaften der entsprechenden Connectoren bzw. Serviceklassen erläutert.

FeedService

Die Klasse „FeedService“ realisiert das Auslesen und Übersetzen von Web-Feeds.
Im Detail wird einer Instanz des „FeedService“ in der Methode „loadFeed()“ ein Objekt vom Typ „InformationSource“ übergeben. Dieses Objekt beinhaltet die URL des Feeds, die zum Auslesen des Feeds benötigt wird. Im weiteren Verlauf der Verarbeitung wird jedes Nachrichtenelement des Feeds in die interne Repräsentation von Nachrichtenelementen übersetzt und auf bekannte Schlüsselwörter durchsucht, die beispielsweise direkt aus dem RSS-„category“-Element ausgelesen werden können. Darüber hinaus werden die Titel und Inhaltselemente auf Schlüsselwörter durchsucht, die dem System bereits bekannt sind. Werden bereits bekannte Schlüsselwörter gefunden, werden sie dem entsprechenden Inhalt zugeordnet. Zusätzlich wird der Web-Feed nach Autoren und Links durchsucht, die ebenfalls auf die entsprechenden Entitäten innerhalb des Systems abgebildet werden. Die nachfolgende Abbildung 25 zeigt das Sequenzdiagramm der Methode „loadFeed()“, die das Auslesen einer externen Feed-Quelle realisiert. Dieser Methode muss ein Objekt des Typs „InformationSource“ übergeben werden, welches die URI des Feed-Dokuments im Web beinhaltet.

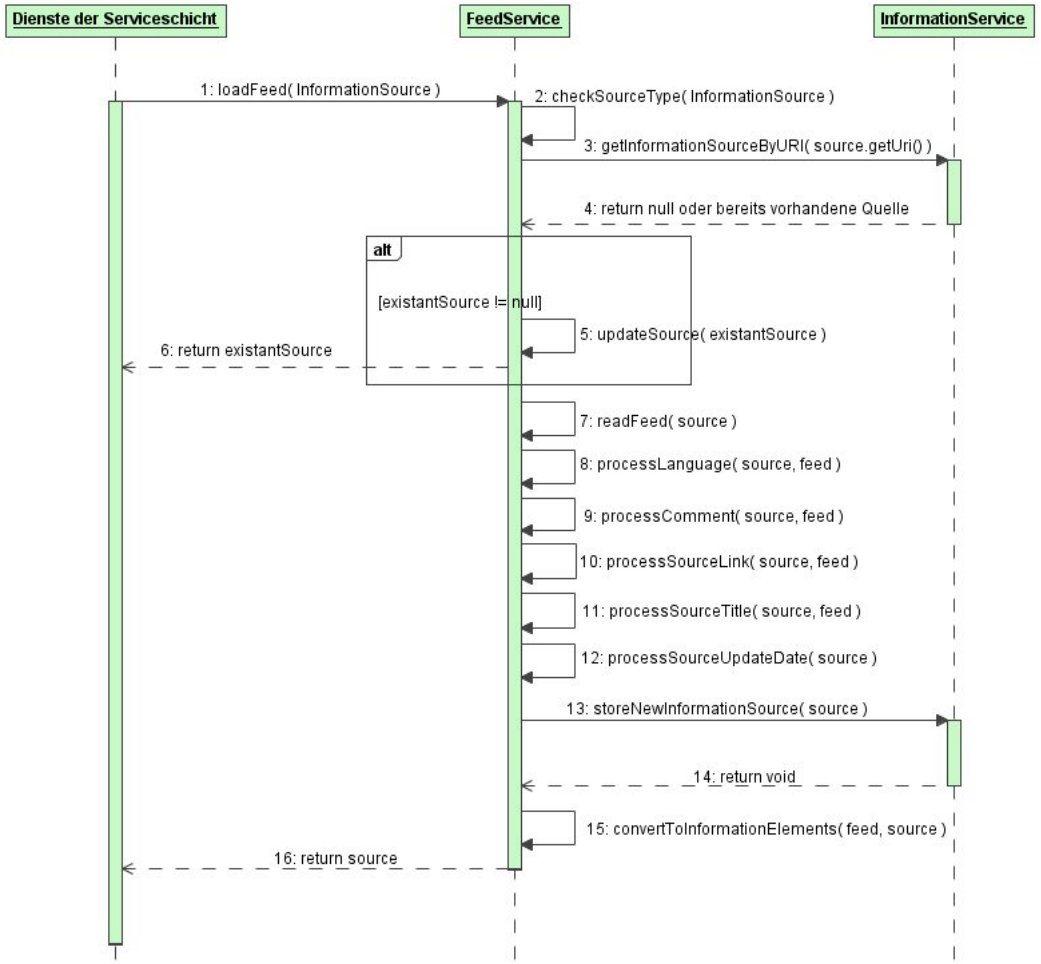

Abbildung 25: Sequenzdiagramm FeedService – Funktion loadFeed()

Für eine Beschreibung der Funktion „convertToInformationElements“, die die Konvertierung von Feed-Elementen in die interne Repräsentation kapselt, wird an dieser Stelle auf das Sequenzdiagramm (Abbildung 48) im Anhang A.3 dieser Arbeit verwiesen.

Im laufenden Betrieb der Anwendung ist es zudem notwendig, in regelmäßigen Intervallen die Quelle auf Änderungen bzw. Aktualisierungen zu überprüfen. Hierzu wird die Methode „updateSource“ aufgerufen und ihr ein Objekt „InformationSource“ als Parameter übergeben, um die zu überprüfende Quelle zu spezifizieren. Während der Verarbeitung werden, sofern vorhanden, neue Elemente ausgelesen und in die interne Repräsentation übersetzt. Elemente, die nicht länger Teil der Quelle sind, werden je nach Anwendungskonfiguration aus dem System entfernt oder bis zu ihrer maximalen Vorhaltedauer vom System gespeichert.

TwitterService

Mit Hilfe des Dienstes „TwitterService“ werden die Inhalte des Twitter-Netzwerkes zu einem bestimmten Twitter-Benutzerkonto ausgelesen. Ähnlich zu dem Vorgehen in der Klasse „FeedService“ wird hier einer Methode „loadAccount()“ ein Objekt „InformationSource“ übergeben. Dieses Objekt beinhaltet die notwendigen Zugangsdaten für das entsprechende Benutzerkonto. Während der Verarbeitung

werden alle Nachrichten von verfolgten Personen sowie Direktnachrichten für den Benutzer ausgelesen und in die interne Repräsentation übersetzt. Fehlende Felder, wie beispielsweise das Titelelement, werden durch die Nutzung von Metainformationen gefüllt. In diesem konkreten Fall wird analysiert, ob es sich bei einem Twitter-Post um eine normale Nachricht oder um eine Direktnachricht handelt. Das resultierende Titelelement besteht aus dem Indikator, um welchen Nachrichtentyp es sich handelt und dem vollen Namen des Absenders.
Aufgrund der in Kapitel 2.2.1.1 beschriebenen Besonderheit, dass Referenzen bzw. Links auf externe Webseiten häufig durch verkürzte URLs dargestellt werden, wird jeder erkannte Link auf eine Weiterleitung geprüft. Wird eine Weiterleitung erkannt, wird die reale Adresse des Links dem Informationselement als Metainformation hinzugefügt. Durch diesen Mechanismus ist es möglich, Konversationen bezüglich einer Webseite nachzuverfolgen und in einem Filter- oder Empfehlungsprozess auszuwerten.
Darüber hinaus werden alle angegebenen Twitterkonten, ebenso wie Web-Feeds, in regelmäßigen Abständen aktualisiert.

CommunoteService
Durch die Klasse „CommunoteService“ werden die Inhalte der Microblogging-Plattform „Communote“ mit Hilfe des von „Communote“ zur Verfügung gestellten personalisierten RSS 2.0 Feeds aggregiert. Der „Communote-RSS-Feed“ unterscheidet sich dabei in einigen Punkten von gewöhnlichen RSS-Feeds. Wie in Kapitel 3.1.4.4 beschrieben, ist eine Authentifizierung des jeweiligen Nutzers notwendig, um auf die Inhalte der Plattform Zugriff zu erhalten. Dies erfolgt über die Angabe von Nutzername und Passwort als „HTTP GET Parameter“ in dem URI des Feeds. Darüber hinaus verfügt Communote über eine Struktur aus Blogs mit untergeordneten Nachrichten, die bei einer verlustfreien Übersetzung in das interne Nachrichtenformat beachtet werden muss. Dies ist Aufgabe des unter 3.2.7 beschriebenen Mechanismus zur automatischen Suche nach Metadaten. Das Auslesen von Inhalten ähnelt dem von Feeds sehr stark. Daher wird diese Aufgabe durch den bereits beschriebenen „FeedService“ übernommen.

3.2.7.2 Kommunikation mit externen Diensten

Um das unter Kapitel 2.1.5.4 beschriebene „new User“-Problem zu lösen, wird für die initiale Benutzerprofilerstellung der „social-bookmarking“- Dienst „Delicious“ verwendet. Dabei werden die relevanten Inhalte, wie Schlüsselwörter und die Häufigkeit ihrer Verwendung, von einem spezifischen Benutzer ausgelesen. Jeder Nutzer hat somit die Möglichkeit, sein Nutzerprofil mit den Inhalten seines „Delicious“- Kontos zu aktualisieren. Dabei werden die in „Delicious“ vorhandenen Schlüsselwörter für den Nutzer importiert und anhand der Häufigkeit ihres Auftretens gewichtet. Ausgangspunkt für die Gewichtung ist der Wert des am seltensten verwendeten Schlüsselwortes. Dieser kann über die Administrationsoberfläche in der Anwendungskonfiguration eingestellt werden.

3.2.7.3 Dienste der Filterung

Die Hauptbestandteile der Filterkomponente der Anwendung stellen die zwei Klassen „ContentBasedFilter“ und „CollaborativeFilter“, die die Funktionalität eines inhaltsbasierenden sowie kollaborativen Filters gewährleisten. Diese Filterklassen lassen sich über die Anwendungskonfiguration mit verschiedenen Parametern konfigurieren, um die Filterung an die entsprechenden Anforderungen der Nutzung anpassen zu können.

Aufgabe des inhaltsbasierenden Filters ist die Berechnung von Relevanzwerten zu Nachrichten für einen bestimmten Nutzer. Um diese Funkionalität zu realisieren, werden dem Filter ein Nutzer sowie die zu analysierenden Informationselemente übergeben. Anhand des Nutzers wird dessen Profil aus dem Nutzermodell geladen, um es im Filterprozess für die Relevanzberechnung der Nachrichten zu verwenden. Die inhaltsbasierende Filterung stützt sich dabei auf die in Kapitel 3.2.6 vorgestellten Prinzipien und Berechnungsvorschriften.

Wurde die inhaltsbasierende Filterung abgeschlossen kann das Ergebnis der Relevanzberechnung durch die Verwendung des kollaborativen Filters noch verbessert werden. Voraussetzung hierfür ist, dass diese Funktionalität in der Anwendungskonfiguration aktiviert wurde. Ist dies der Fall, werden vom kollaborativen Filter zunächst ähnliche Nutzer gesucht. Hierfür stehen verschiedene Algorithmen zur Ähnlichkeitsberechnung zur Verfügung, die ebenfalls in der Anwendungskonfiguration ausgewählt werden können. Einem solchen Algorithmus werden zwei Nutzerprofile zur Ähnlichkeitsanalyse übergeben. Ergebnis der Berechnung ist ein Ähnlichkeitswert zwischen Null und Eins, wobei Eins die höchstmögliche Ähnlichkeit darstellt. Werden ähnliche Nutzer gefunden, die eine oder mehrere der relevanten Nachrichten bewertet haben, folgt die Vergabe von Boni bzw. Strafen für den ursprünglichen Relevanzwert der Nachricht.

Zusätzlich zu ihrer Bewertungs- bzw. Filterfunktion erfüllen die Klassen „ContentBasedFilter“ und „CollaborativeFilter“ ebenfalls die Funktion des Empfehlungssystems. Für sogenannte „Item-to-Item“-Empfehlungen sucht der inhaltsbasierende Filter nach Nachrichten, die mit den gleichen Schlüsselwörtern versehen sind und liefert diese als Ergebnis zurück.

Um nutzerbasierende Empfehlungen auszusprechen, implementiert die Klasse „CollaborativeFilter“ die Funktion „getRecommendations()“. Dieser Methode wird ein Objekt vom Typ „User“ übergeben, um Empfehlungen für diesen Nutzer zu erhalten. Während der Verarbeitung werden analog zur Filterung ähnliche Nutzer gesucht und geprüft, ob diese positive Bewertungen für Nachrichtenelemente abgegeben haben. Werden ähnliche Nutzer gefunden, werden die positiv bewerteten Nachrichten dieser Nutzer als Leseempfehlung für den ursprünglichen Nutzer zurückgegeben.

3.2.7.4 Frontend Schnittstelle

Um die Kommunikation mit dem durch Serkan Canbolat [Canbolat+09] entwickelten Frontend zu realisieren, bietet die prototypische Anwendung eine Schnittstelle auf Serviceebene an. Im Einzelnen bedeutet dies, dass die Anwendung

spezielle Serviceklassen bereitstellt, die den Zugriff auf die für das Frontend relevanten Funktionalitäten gewährleisten. Die nachfolgende Abbildung 26 ordnet die Schnittstelle in die Architektur der Anwendung ein.

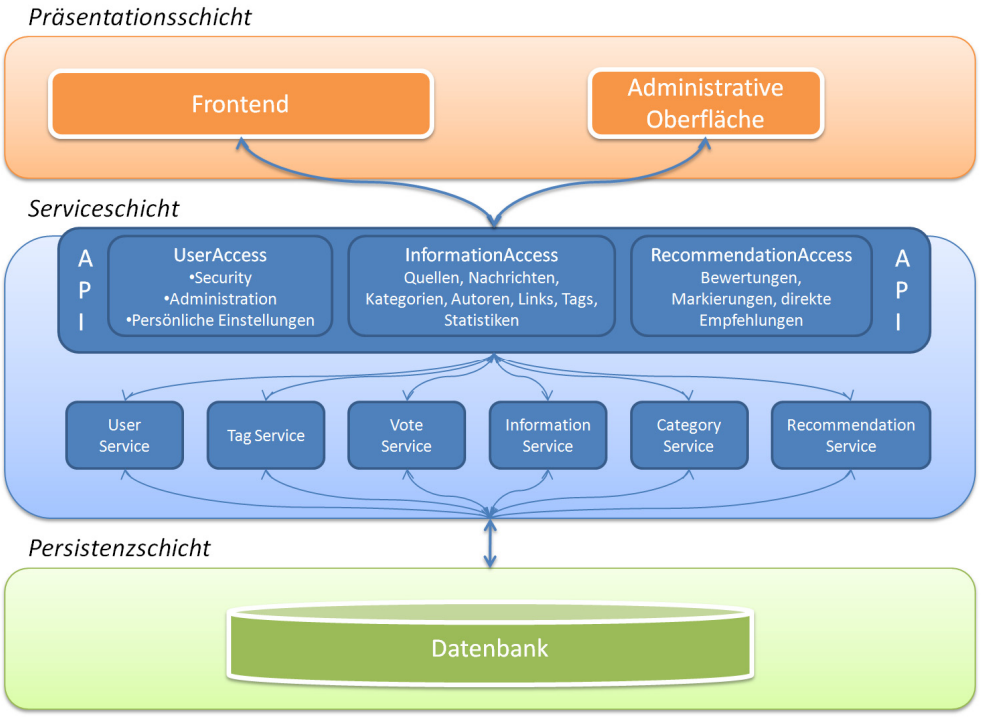

Abbildung 26: Einordnung der Frontend Schnittstelle

Die Schnittstelle ist aus Gründen der Übersichtlichkeit in drei Komponenten nach Funktionsbereichen aufgeteilt. Die Komponente „UserAccess“ dient dabei der Verwaltung von Nutzern und stellt die korrekte Abarbeitung von An- und Abmeldevorgängen sicher.

Kern der Schnittstelle bildet die Komponente „InformationAccess“. Diese kapselt Funktionen für den Zugriff auf Entitäten des Systems, die unmittelbar mit Informationsquellen und Informationselementen assoziiert sind.

Für den Zugriff auf das Bewertungs-, Markierungs- und Empfehlungssystem wird die Komponente „RecommendationAccess“ verwendet. Die Klassen dieser Komponente bieten Funktionen für den Zugriff auf inhaltsbasierende sowie kollaborative Empfehlungen. Darüber hinaus können über diese Komponente einzelne Nachrichtenelemente bewertet bzw. markiert werden.

3.2.8 Automatische Metadatensuche

Grundlage einer hochwertigen Filterung ist die möglichst genaue Beschreibung von Inhalten durch Metadaten, die bei einer Filterung verwendet werden können. Aufgrund der Tatsache, dass einige Nachrichtenquellen von den Autoren nur

unzureichend mit Metadaten versehen werden, verfügt der Prototyp über Mechanismen, die Inhalten automatisch Schlüsselwörter hinzufügen.

Ausgangspunkt für dieses Verfahren ist die Menge von Schlüsselwörtern, die durch Nutzerprofile oder nativ ausgelesene Schlüsselwörter der Informationsquellen im System zur Verfügung stehen. Jede eingehende Nachricht wird somit auf bereits bekannte Schlüsselwörter durchsucht und im Fall einer Übereinstimmung diesen Schlüsselwörtern zugeordnet.

Die Anwendung unterscheidet dabei verschiedene Typen von Schlüsselwörtern, um diese genauer zu spezifizieren und in einem späteren Filterprozess zielgerichtet einsetzten zu können. Die nachfolgende Tabelle 10 gibt eine Übersicht über die verschiedenen Typen von Schlüsselwörtern, die in der Klasse „TagType“ definiert sind:

Typ	Beschreibung
TagType.KEYWORD	Einfaches Schlüsselwort zur Inhaltsbeschreibung
TagType.TWITTER_USER	Identifier eines Twitternetzwerk-Benutzers
TagType.COMMUNOTE_USER	Identifier eines Communote-Benutzers
TagType.COMMUNOTE_BLOG	Titel eines Commuote Blogs

Tabelle 10: Typen von Schlüsselwörtern

Die Funktionalität dieses Mechanismus ist in der Klasse „TagService“ implementiert. In dieser Klasse existieren verschiedene Funktionen, um die Inhalte aus den heterogenen Nachrichtenquellen zu verarbeiten. Diese sind an die Eigenschaften und Spezialfälle dieser Nachrichtenformate angepasst und stellen sicher, dass die relevanten Informationen ordnungsgemäß ausgelesen werden.

3.2.9 Implementierung der Architektur

Die zu entwickelnde Infrastruktur wird mit Hilfe von „Model-Driven Software Development“ (MDSD) [WIKI-02] und Java erstellt. Hierzu wird das Opensource-Tool „AndroMDA“[1] verwendet, um Teile der Serviceschicht und die komplette Persistenzschicht in „UML 1.4“ zu modellieren und im Anschluss den entsprechenden Code zu generieren [AndroMDA].

Analog zum vorgesehenen Aufbau der Anwendung (vgl. Kapitel 3.2.6) verfolgt „AndroMDA“ einen 3-Schichten-Ansatz, der die Anwendung in die Ebenen „Presentation Layer“, „Business Layer“ und „Data Access Layer“ untergliedert (siehe Abbildung 27). Jede dieser Schichten kann mit unterschiedlichen Technologien realisiert werden. „AndroMDA“ hält für diesen Zweck sogenannte „Cartridges“ bereit, die die entsprechenden Teile des UML-Modells in die gewünschte Form überführen.

[1] AndroMDA Version 3.2 Lizenz: BSD license

Presentation Layer

In dem zu entwickelnden Prototyp wird darauf verzichtet, ein „Presentation Layer“ in „UML“ [UML] zu modellieren, da es sich bei der Anwendung fast ausschließlich um eine Backendanwendung handelt. Die Oberflächen, die den administrativen Bereich bilden, werden manuell erstellt und basieren auf „JavaServer Faces“ [JSF] bzw. „Apache MyFaces“[2] [MYFACES].

Business Layer

Der „Business Layer“ besteht im Wesentlichen aus seiner Reihe von Diensten bzw. Services, die mit Hilfe der „Spring Cartridge“ [3] generiert werden. Hauptaufgabe dieser Dienste ist es, Funktionen für das Management der Daten zu realisieren und die Anwendungslogik zu kapseln. Zu diesem Zweck kann jedem Dienst im „UML“-Modell eine Entität durch eine Abhängigkeit zugeordnet werden. Dies stellt sicher, dass der entsprechende Dienst Zugriff auf die Datenzugriffsobjekte („DataAccessObjects“) für diese Entität besitzt. Darüber hinaus ist vorgesehen, dass in diesen Diensten die Manipulation und die eigentliche Verarbeitung der Datenobjekte transaktionssicher abgearbeitet werden.

Dienste, die keiner Entität zugeordnet sind, werden ebenfalls generiert. In den meisten Fällen werden sie genutzt, um bestimmte Funktionen der Anwendung zu realisieren oder um mit externen Diensten zu kommunizieren.

Data Access Layer

Grundlage für den „Data Access Layer“ bildet die „Hibernate“[4]- Technologie [HIBERNATE]. Sie gewährleistet eine vollautomatische Umsetzung von Java-Objekten auf relationale Datenbanken. Die entsprechenden Voraussetzungen für einen solchen Vorgang, wie beispielsweise „Hibernate Mapping Files“ und „DataAccessObjects“, werden mit Hilfe der „Hibernate Cartridge“ vollständig generiert, sofern diese im zugehörigen „UML“- Modell korrekt definiert sind.

[2] Apache MyFaces Version 1.2.5 Lizenz: Apache Software License, Version 2.0

[3] Unter Nutzung des Spring Framework Version 2.0.6 Lizenz: Apache Software License, Version 2.0

[4] Hibernate, Version: 3.2.6 Lizenz: LGPL

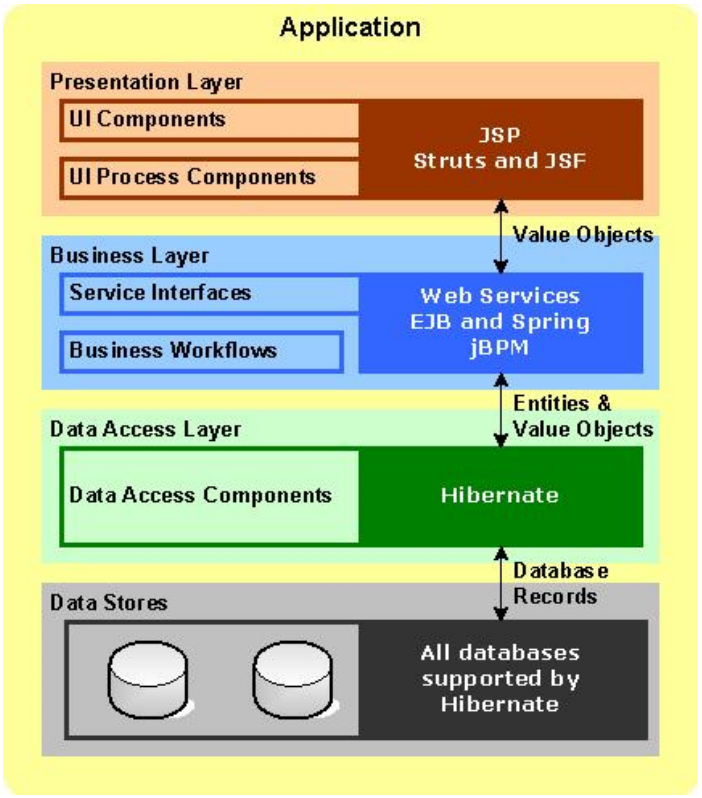

Abbildung 27: AndroMDA 3-Schichten-Architektur [AndroMDA]

3.2.9.1 Datendienste

Bei der Implementierung von Diensten, die auf externe Datenquellen zugreifen, werden offene Bibliotheken verwendet. Dies führt zu einer Reduzierung der Entwicklungszeit, da die Zugriffe auf diese externen Datendienste nicht eigenhändig implementiert werden müssen.

Für den Zugriff auf externe Feeds im „XML"- Format wird die offene Bibliothek „ROME"[5] [ROME] eingesetzt. Durch die Verwendung dieser Bibliothek wird sichergestellt, dass alle gängigen Feed-Formate, wie u.a. RSS 2.0, RSS 0.9 oder Atom 1.0, unterstützt werden. Darüber hinaus wird mit Hilfe von „ROME" der Inhalt von einzelnen Feed-Elementen hinsichtlich des Datenformats (z.B. Klartext oder „HTML") überprüft. Diese Informationen können später bei einer Präsentation durch eine Frontend-Anwendung ausgelesen und berücksichtigt werden.

Ähnlich dem Aggregieren von Feeds wird beim Auslesen von Inhalten aus dem „Twitter"- Netzwerk vorgegangen. Innerhalb der Klasse „TwitterService" wird die Bibliothek „Twitter4J"[6] [TWITTER4J] eingesetzt, um unter Zuhilfenahme der entsprechenden Zugangsdaten eine Verbindung zum „Twitter"- Netzwerk aufzubauen. Dabei werden jedoch nur Funktionen zum Auslesen von Inhalten genutzt, da für die Aggregation nur lesende Zugriffe notwendig sind. Für eine

[5] ROME Version 1.0 RC2 – Lizenz: Apache License 2.0
[6] Twitter4J Version 1.1.7 – Lizenz: BSD-style license

detailliertere Beschreibung des Vorgehens bei der Aggregation von „Feed"- und „Twitter"-Inhalten wird an dieser Stelle auf Kapitel 3.2.7.1 verwiesen.

Darüber hinaus wird auch beim Zugriff auf den „social-bookmarking"- Dienst „Delicious" eine offene Bibliothek verwendet. Um die relevanten Inhalte, wie Schlüsselwörter und die Häufigkeit ihrer Verwendung von einem spezifischen Benutzer auszulesen, wird in der Klasse „DeliciousService" die Bibliothek „del.icio.us Java API"[7] [DELI-API] verwendet.

3.2.9.2 Frontend Schnittstelle

Wie bereits in Kapitel 3.2.7.4 beschrieben, besteht die Frontend Schnittstelle aus mehreren Diensten der Serviceschicht. Nachfolgend wird die Implementierung der einzelnen Komponenten sowie deren Klassen beschrieben.

Die Funktionen der Serviceklassen sowie die Objekte, die als Träger der Daten fungieren, sind auf die Anwendungsfälle des Frontends zugeschnitten, um einen möglichst komfortablen Zugriff auf die Inhalte und Funktionen der Infrastruktur zu gewährleisten. Somit untergliedert sich die Komponente „UserAccess" in die Klassen „UserManager", die Funktionen für die Verwaltung der Nutzer bereitstellt und die Klasse „SecurityManager", die hauptsächlich Funktionen für die Abarbeitung von Anmelde- und Abmeldevorgängen kapselt. Für die Repräsentation von Nutzerentitäten innerhalb dieser Klassen wird das „Value Object" „UserVO" verwendet, das alle Attribute der zugrunde liegenden Entität kapselt.

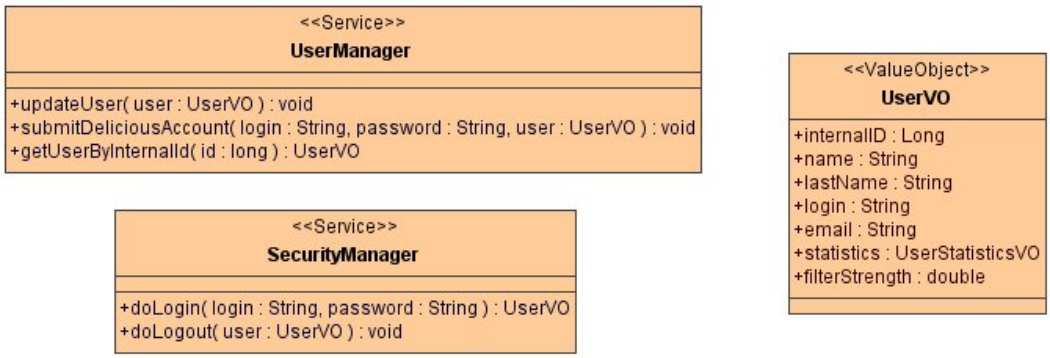

Abbildung 28: Klassendiagramm „UserAccess" und Auszug aus Klassendiagramm „API Value Objects"

Die zentrale Komponente der Schnittstelle bilden die Klassen die unter dem Punkt „InformationAccess" zusammengefasst sind. Diese Klassen stellen Funktionen bereit, um beispielsweise Nachrichtenströme abzurufen, Kategorien zu verwalten, Schlüsselwörter auszulesen sowie Autoren und Links auszulesen. Darüber hinaus wird durch die Klasse „StatisticsManager" die Möglichkeit geboten, Statistiken zu Nutzern und Kategorien zu erstellen.

Analog zur Komponente „UserAccess" werden an dieser Stelle ebenfalls „Value Objects" als Repräsentanten der Entitäten des Systems verwendet. Eine besondere Bedeutung hat hierbei das Objekt „NewsVO". Es kapselt neben den Attributen von Informationselementen (z.B. Inhalt oder Titel) ebenfalls die Relevanz der jeweiligen

[7] Delicious Java API Version 1.14 – Lizenz: BSD License

Nachricht für den entsprechenden Nutzer. Zudem beinhaltet es Informationen darüber, ob diese Nachricht bereits bewertet, gelesen, geschlossen oder als „ToDo“ markiert wurde.

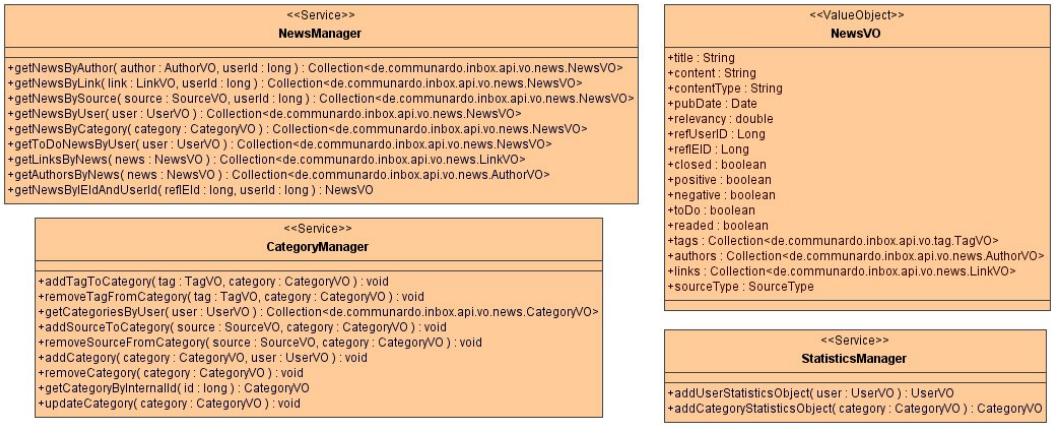

Abbildung 29: Auszug Klassendiagramm „InformationAccess“ und „API Value Objects“

Für den Zugriff auf das Bewertungs- und Empfehlungssystem der Infrastruktur wird die Komponente „RecommendationAccess“ genutzt. Diese bietet Funktionen, um Bewertungen und Markierungen von Nutzern zu Nachrichtenelementen an die Infrastruktur zu übermitteln. Darüber hinaus kann über die Funktionen der Klasse „RecommendationManager“ auf inhaltsbasierende Empfehlungen sowie kollaborative Empfehlungen zu einer Nachricht bzw. einem Nutzer zugegriffen werden.

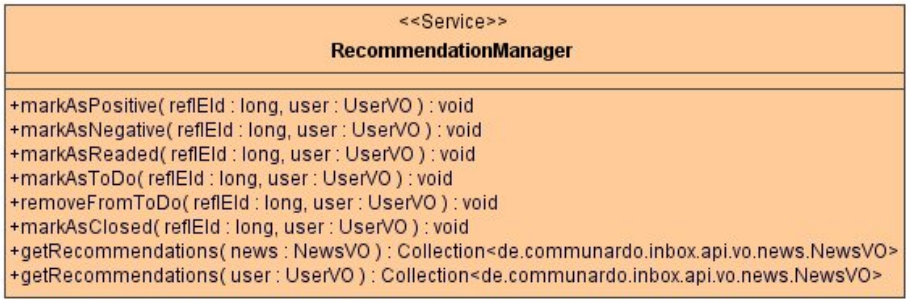

Abbildung 30: Klassendiagramm „RecommendationAccess“

Für eine Dokumentation der einzelnen Funktionen der Schnittstellenklassen wird an dieser Stelle auf den Anhang A.1 bzw. auf die Quellcodedokumentation („Javadoc“) auf der beiliegenden DVD verwiesen.

4 Evaluierung

Im vorhergehenden Kapitel 3 – Realisierung der Infrastruktur- wurden die im Rahmen dieser Diplomarbeit zu erarbeitenden Konzepte dargelegt und die prototypische Implementierung beschrieben. Anknüpfend an diese Erkenntnisse wird in diesem Kapitel eine Evaluierung bezüglich dieser Punkte durchgeführt. Hierzu werden zunächst die Ziele dieser Evaluierung bestimmt und die Durchführung dieser beschrieben. Abschließend werden die Ergebnisse in einer Auswertung dokumentiert.

4.1 Ziele

Im Rahmen der Evaluierung der erstellten Infrastruktur wird in den folgenden Abschnitten darauf eingegangen, auf welche Art und Weise die herausgestellten Kernanforderungen aus Kapitel 3.1.6 umgesetzt wurden. In diesem Zusammenhang soll erarbeitet werden, ob diese Ziele vollständig erreicht wurden bzw. unter welchen Voraussetzungen diese verwirklicht werden konnten.
Des Weiteren werden im Rahmen der Evaluation Prüfungen durchgeführt, um die korrekte Funktionsweise der prototypischen Implementierung sicherzustellen. In diesem Zusammenhang soll validiert werden, ob die Aggregation heterogener Nachrichtenquellen sowie die anschließende Übersetzung in ein internes Nachrichtenformat verlustfrei durchgeführt wird. Im Zuge dieser Prüfungen soll ebenfalls ermittelt werden, welche Schwächen die implementierte automatische Suche nach Metainformationen aufweist. Ziel dieser Prüfungen ist es, Ansätze für eine Weiterentwicklung der genannten Funktionen zu liefern.

Weitere Prüfungen verfolgen das Ziel, Aussagen über die Qualität des Nutzermodells zu treffen. Hierbei ist die initiale Benutzerprofilerstellung durch die Nutzung des „social bookmarking“-Dienstes „Delicious“ von besonderer Bedeutung. Die Tests dienen der Validierung des Konzepts, externe Dienste für die Abbildung von Nutzerinteressen zu verwenden. In diesem Kontext sollen die Voraussetzungen und Schwächen, die ein solches Vorgehen birgt, herausgestellt werden.

Ein weiterer Komplex der Evaluierung bezieht sich auf die Bestimmung von optimalen Konfigurationseinstellungen der Infrastruktur. Dabei soll verdeutlicht werden, welche Konfigurationsparameter bei bestimmten Rahmenbedingungen (z.B. Nutzeranzahl, Anzahl verwalteter Nachrichten) zu hochwertigen Ergebnissen der Filterung führen. Das Ziel dieser Evaluierung ist, einen Leitfaden für die Konfiguration von Filterparametern in Abhängigkeit der genannten Rahmenbedingungen zu erstellen. Im Zuge dieser Tests werden ebenfalls die von der Infrastruktur unterstützten Algorithmen zur Ähnlichkeitsanalyse gegenübergestellt. Hierbei soll ermittelt werden, welcher Algorithmus unter bestimmten Rahmenbedingungen zu repräsentativen Ergebnissen führt. Diese Ergebnisse sollen

eine Schussfolgerung ermöglichen, welcher Algorithmus das geeignetste Verfahren für die Auswertung des vorliegenden Nutzermodells ist.

Abschließend soll durch eine gesonderte Evaluierung die Effizienz des Systems herausgestellt werden. Hierzu soll eine Prüfung hinsichtlich der Skalierbarkeit und Leistungsfähigkeit durchgeführt werden.

4.2 Durchführung

4.2.1 Prüfung der Anforderungen

Im Zuge der Prüfung der Anforderungen wird die erstellte Implementierung der Infrastruktur mit Blick auf die herausgestellten Kernanforderungen betrachtet. Dabei wird darauf eingegangen, in welcher Art und Weise eine Anforderung erfüllt und in welchem Umfang diese umgesetzt wurde. In Fällen einer unvollständigen Umsetzung einer Anforderung wird dokumentiert, welcher Grad der Fertigstellung erreicht worden ist und welche Punkte zu einer vollständigen Umsetzung offen geblieben sind.

4.2.2 Evaluierung der Aggregation und Übersetzung

Im Rahmen der Evaluierung der Aggregation und Übersetzung von externen heterogenen Informationsquellen soll geprüft werden, ob die folgenden Nachrichtenformate anhand eines entsprechenden Beispiels verlustfrei in ein einheitliches internes Format überführt werden konnten:

- RSS 0.9 Feed – Heise Online News RSS 0.9 Feed[8]
- RSS 1.0 Feed – Semantic Web Feed[9]
- RSS 2.0 Feed – ReadWriteWeb Feed[10]
- Atom 1.0 Feed – Heise Online News Atom Feed[11]
- Twitter-Benutzerkonto mit Statusmeldungen und Direktnachrichten[12]
- Communote-Benutzerkonto via RSS 2.0 Feed[13]

Bei der Durchführung der Evaluation der Übersetzung in ein internes Nachrichtenformat sind die folgenden Kriterien von besonderer Relevanz:

[8] Heise Online News RSS 0.9 – http://www.heise.de/newsticker/heise.rdf Stand: 12.04.2009
[9] Semantic Web Feed RSS 1.0 – http://www.semantic-web.at/rss.php Stand: 12.04.2009
[10] ReadWriteWeb Feed RSS 2.0 – http://feeds2.feedburner.com/readwriteweb Stand: 12.04.2009
[11] Heise Online News Atom 1.0 Feed – http://www.heise.de/newsticker/heise-atom.xml Stand: 12.04.2009
[12] "Twitter"- Benutzerkonto INBOXTEST1 – Stand: 12.04.2009
[13] "Communote"- Benutzerkonto mit Login „tei" – Stand: 12.04.2009

– Korrekte Übersetzung der Elemente der Quelle auf die Attribute der Entitäten des internen Formates
– Auswertung des Inhaltstyps von Nachrichteninhalten
– Nicht auslesbare Elemente durch Nutzung von Metadaten füllen
– Korrekte Übersetzung von Metadaten (Datum der Publikation, Autoren und Links)
– Nachrichtenformatspezifische Metainformationen korrekt auslesen (z.B. referenzierte „Twitter"- Nutzer, „Communote"- Nutzer und Blogs)

4.2.3 Evaluierung des Nutzermodells

Im Rahmen der Evaluierung des Nutzermodells sollen die Funktionen der Infrastruktur bezüglich der Verwaltung von Nutzerprofilen geprüft werden. Darüber hinaus soll anhand zweier Beispielnutzer[14] die korrekte Funktionsweise der Benutzerprofilakquisition durch den „social bookmarking"-Dienst „Delicious" validiert werden.
Darüber hinaus werden die Funktionen des Systems zur automatischen Justierung von Nutzerprofilen durch explizite Bewertungsverfahren analysiert und geprüft, ob diese in jedem Fall zu den gewünschten Ergebnissen führen.

4.2.4 Evaluierung der automatischen Metadatensuche

Die Evaluierung der automatischen Metadatensuche ist in vielen Punkten eng mit der Evaluierung der Aggregation und Übersetzung verknüpft. Aufgrund dieser Tatsache wird diese Evaluation ebenfalls auf Grundlage der unter 4.2.2 genannten Nachrichtenformate und Quellen durchgeführt. Dabei wird analysiert, welche Voraussetzungen für eine erfolgreiche und korrekte Suche nach Schlüsselwörtern innerhalb einer Nachricht erfüllt sein müssen.

Im Rahmen dieser Evaluation wird ebenfalls das Verhalten der Infrastruktur unter den folgenden bewusst herbeigeführten Fehlern analysiert:

– Definition von falschen Schlüsselwörtern (z.B. „der", „die", „das", „Mikrosoft")
– Fehlende Metadaten (z.B. kein Publikationsdatum in der Quelle bzw. in den Nachrichtenelementen vorhanden)
– Mehrfachaufkommen von gleichen Schlüsselwörtern innerhalb eines Nachrichtenelementes

4.2.5 Evaluierung von Anwendungskonfigurationen

Das Filter- und Empfehlungssystem der Infrastruktur kann über die Anwendungskonfiguration angepasst werden. Mit Blick auf die Ergebnisse der

[14] "Delicious"- Account „skispeedrace" und „oppacher" Stand: 09.04.2009

Filterung werden verschiedene Konfigurationsbeispiele geprüft. Hierzu wird jeweils ein identischer Referenznutzer ausgewählt und Tests mit den in Tabelle 11 dargestellten Konfigurationen durchgeführt. Dabei ist zu beachten, dass diese Konfigurationen ausgewählt wurden, um die folgenden variablen Rahmenbedingungen zu simulieren:

- Unterschiedliche Nutzerzahlen
- Unterschiedliche Werte für die minimale Schnittmenge von Nutzerprofilen
- Unterschiedliche Ähnlichkeitsschwellenwerte
- Unterschiedliche Werte für kollaborative Strafen und Boni

Dabei ist zu beachten, dass die nachfolgendend aufgelisteten Rahmenbedingungen sowie Parameter der Anwendungskonfiguration für alle geprüften Konfigurationen die gleichen Werte besitzen und nicht variieren:

- Anzahl der Quellen des Referenznutzers / Anzahl der beinhalteten Nachrichtenelemente: 10 Quellen / 99 Elemente
- Anzahl der mit den Nachrichtenelementen verknüpften Schlüsselwörter: 48
- Minimale Relevanz von Nachrichten: 0,35
- Anpassungsfaktor für positive und negative Bewertungen im Nutzermodell: 0,15

Die nachfolgende Tabelle 11 gibt Aufschluss über die variablen Werte zwischen den einzelnen geprüften Konfigurationen:

	Konfiguration 1[15]	Konfiguration 2[16]	Konfiguration 3[17]	Konfiguration 4[18]	Konfiguration 5[19]	Konfiguration 6[20]
Anzahl der Nutzer des Systems	5	5	10	10	10	10
Anzahl positiver Bewertungen für Nachrichten des Referenznutzers von anderen Nutzern der Infrastruktur	14	14	32	32	32	32
Anzahl negativer Bewertungen für Nachrichten des Referenznutzers von anderen Nutzern der Infrastruktur	7	7	11	11	11	11
Minimale Schnittmenge der Nutzerprofile für die Ähnlichkeitsanalyse in %	60	30	60	30	60	60
Minimaler Ähnlichkeitswert zweier Nutzer	0,7	0,35	0,7	0,35	0,7	0,5
Faktor für kollaborative Boni	0,4	0,4	0,4	0,4	0,2	0,2
Faktor für kollaborative Strafen	0,4	0,4	0,4	0,4	0,2	0,2
Ähnlichkeitsalgorithmus	Vektorraum	Vektorraum	Vektorraum	Vektorraum	Vektorraum	Vektorraum

Tabelle 11: Evaluierungskonfigurationen der Infrastruktur

Während dieser Evaluierung werden die folgenden Ergebnisse dokumentiert:

– Anzahl der Nachrichtenelemente, die von kollaborativen Boni und Strafen betroffen sind
– Anzahl ähnlicher Nutzer sowie Anzahl ihrer Bewertungen für gefilterte Nachrichten
– Wert der kollaborativen Boni und Strafen zur relevantesten Nachricht

[15] Datensatz als SQL Backup Datei "conf1-2.sql" ist Bestandteil des Anhangs A.1 (DVD)
[16] Datensatz als SQL Backup Datei "conf1-2.sql" ist Bestandteil des Anhangs A.1 (DVD)
[17] Datensatz als SQL Backup Datei "conf3-6.sql" ist Bestandteil des Anhangs A.1 (DVD)
[18] Datensatz als SQL Backup Datei "conf3-6.sql" ist Bestandteil des Anhangs A.1 (DVD)
[19] Datensatz als SQL Backup Datei "conf3-6.sql" ist Bestandteil des Anhangs A.1 (DVD)
[20] Datensatz als SQL Backup Datei "conf3-6.sql" ist Bestandteil des Anhangs A.1 (DVD)

Für die Bewertung der kollaborativen Funktionen der Filterung sowie des Empfehlungssystems werden dabei die folgenden Nutzerprofile verwendet. Hierbei ist zu beachten, dass „Nutzer 1“ als Referenznutzer ausgewählt wird.

Nutzer / Schlüsselwort	1	2	3	4	5	6	7	8	9	10
Microsoft	0,75	0,7	0,5	0,2	0,6	-	0,2	-	-	0,67
Java	0,3	0,4	0,05	0,9	0,5	0,5	0,5	-	-	-
XML	0,8	0,75	0,55	1	-	-	0,5	-	0,34	0,92
XSLT	0,5	0,5	0,25	0,8	-	0,3	-	-	0,2	0,3
JSF	0,85	0,75	0,6	0,15	0,5	-	-	-	-	-
MDA	0,7	0,73	0,45	0,2	0,55	-	0,2	0,3	-	0,65
Hibernate	0,4	0,4	0,15	0,89	-	0,7	0,2	0,2	0,9	-
Spring	0,7	0,65	0,45	0,1	-	0,2	0,4	0,15	0,23	-
Apache	0,3	0,1	0,05	0,67	0,45	0,9	0,6	0,7	-	0,4
MyFaces	0,8	0,6	0,55	0,2	0,55	-	-	0,1	-	-
Microblogging	0,65	0,6	0,4	0,13	0,4	-	0,1	0,1	-	0,7
Twitter	0,5	0,4	0,25	0,2	-	-	0	-	-	0,55

Tabelle 12: Nutzerprofile der Evaluierungstests

Im Rahmen der Evaluierung des Empfehlungssystems wird der Algorithmus des Vektorraum-Modells zur Ähnlichkeitsanalyse verwendet. In der nachfolgenden Evaluierung der Ähnlichkeitsalgorithmen wird auch auf alle weiteren vom System unterstützten Algorithmen zur Ähnlichkeitsanalyse eingegangen.

4.2.6 Evaluierung der Ähnlichkeitsanalyse

Die Implementierung der Infrastruktur unterstützt drei verschiedene Algorithmen zur Ähnlichkeitsanalyse (angepasstes Vektorraum-Modell, Euklidische Distanz und Pearson-Korrelation) zwischen den Nutzern. Die Ergebnisse dieser Algorithmen sollen im Rahmen dieser Evaluierung gegenübergestellt werden. Als Datensatz für diese Evaluierung dienen die Nutzer bzw. deren Nutzerprofile aus Tabelle 12.
Darüber hinaus werden die folgenden Ergebnisse der Ähnlichkeitsberechnung für Nutzer 1 erwartet:

Nutzer	2	3	4	5	6	7	8	9	10
Erwartete Ähnlichkeit zu Nutzer 1	sehr hoch	mittel bzw. sehr hoch bei Pearson Korrelation	niedrig	hoch bis mittel	mittel	niedrig	niedrig	niedrig	sehr hoch

Tabelle 13: Erwartungswerte der Ähnlichkeitsanalyse

Während der Durchführung der Evaluierung werden alle errechneten Ähnlichkeitswerte jedes Algorithmus dokumentiert, um sie im Abschnitt 4.3.6 auszuwerten.

4.2.7 Skalierbarkeit und Leistungsfähigkeit

Zur Ermittlung von repräsentativen Werten wird die Implementierung vollständig auf einem System[21] ausgeführt. Dies beinhaltet den Webserver[22], die Anwendung sowie den Datenbank Server[23]. Dabei werden Messungen zu folgenden Punkten durchgeführt:

- Verhalten bei steigenden Eingabemengen im Bezug auf die Leistungsfähigkeit
- Dauer von Ähnlichkeitsberechnungen
- Dauer von Relevanzberechnungen
- Dauer von Datenbankzugriffen

Die Messungen erfolgen durch direkte Anweisungen in den betreffenden Methoden der ausführenden Dienste. Die zeitliche Abweichung, die durch die Verarbeitung der Messung entsteht, ist zu vernachlässigen. Um verschiedene Datenmengen innerhalb des Systems zu simulieren, werden die nachfolgend dargestellten Rahmenbedingungen eingesetzt, um fünf Testreihen durchzuführen:

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
Anzahl der anzuzeigenden Nachrichtenelemente	10	10	40	40	80	80
Anzahl der mit den Nachrichten verknüpften Schlüsselwörter	6	6	24	24	44	44
Anzahl der mit den Nachrichten verknüpften Links und Autoren	11	11	41	41	81	81
Anzahl ähnlicher Nutzer	4	8	4	8	4	8
Anzahl von Bewertungen ähnlicher Nutzer für abonnierte Nachrichtenelemente	14	21	19	32	24	39

Tabelle 14: Rahmenbedingungen der Leistungsfähigkeitsevaluierung

[21] Windows Vista Business 32bit; AMD Athlon X2 5600+ Prozessor, 3,19 GB verfügbarer Arbeitsspreicher

[22] Apache Tomcat Version 6.0.18

[23] MySQL Community Server Version 5.1.30

4.3 Auswertung

4.3.1 Bewertung der Anforderungen

KA #1: Aggregation heterogener semistrukturierter Informationsquellen:

Die Implementierung der Infrastruktur unterstützt die Aggregation von drei heterogenen Informationsquellen. Web-Feeds, das Microblogging-Netzwerk „Twitter“ sowie die Enterprise-Microblogging-Lösung „Communote“. Dabei werden alle durch einen Nutzer abonnierten Daten in ein internes Nachrichtenformat übersetzt und zu einem einheitlichen Nachrichtenstrom zusammengefasst. Die Konvertierung von externen Quellen auf eine einheitliche Repräsentation verläuft bezüglich der nachfolgend aufgelisteten Felder inhaltlich verlustfrei:

– Quellentitel, -URL, -sprache, -link
– Nachrichtentitel, -inhalt, -inhaltstyp, -publizierungsdatum, -autor, -link, -schlüsselwörter

KA #2: Personalisierte Filterung von Nachrichten auf Basis von inhaltsbasierenden und kollaborativen Filtern:

Die personalisierte Filterung der Anwendung erfolgt mit Hilfe von inhaltsbasierten und kollaborativen Verfahren. Inhaltsbasierende Verfahren basieren auf der Zuordnung von Schlüsselwörtern zu Nachrichten sowie dem Nutzermodell, das eine gewichtete Zuordnung von Schlüsselwörtern zu Nutzern darstellt. Dabei werden Relavanzwerte für Nachrichten anhand der verwendeten Schlüsselwörter berechnet. Zusätzlich kann diese Relevanzberechnung durch kollaborative Verfahren verfeinert werden. Hierzu werden ähnliche Nutzer gesucht, die bereits Bewertungen für Nachrichten abgegeben haben. Diese positiven oder negativen Bewertungen fließen anschließend in die Relevanzberechnung ein und bilden somit die Eindrücke anderer Nutzer in der Filterung ab.

KA #3: Erweiterbarkeit der Infrastruktur hinsichtlich weiterer Nachrichtenformate:

Im Entwurf der Infrastruktur wurde ein Konzept für die einfache Erweiterbarkeit der Infrastruktur entwickelt. Durch die Verwendung von separaten Connectoren für jedes unterstützte Nachrichtenformat wird die einfache Erweiterbarkeit der Infrastruktur sichergesellt. Jeder Connector ist in der Lage, die entsprechenden formatspezifischen Quellen auszulesen und deren Inhalte in die interne Repräsentation von Nachrichten zu übertragen.
Für den Fall einer Erweiterung um ein weiteres Nachrichtenformat, sind neben der Implementierung eines formatspezifischen Connectors nur minimale Änderungen an

den anderen Komponenten des Systems notwendig. Dies erhöht die Flexibilität des Systems und senkt die Entwicklungskosten für zukünftige Erweiterungen.

KA #4: Realisierung eines personalisierten Empfehlungssystems für verwaltete Nachrichten:

Neben einer personalisierten Filterung verfügt das System zudem über ein personalisiertes Empfehlungssystem, um sowohl inhaltsbasierte als auch kollaborative Empfehlungen bereitzustellen. Diese Funktionalitäten beziehen sich, im Gegensatz zu allen anderen Funktionen der Infrastruktur, nicht ausschließlich auf die abonnierten Inhalte des Referenznutzers. Mit Ausnahme von privaten Quellen können alle vom System verwalteten Nachrichten in das Empfehlungssystem einfließen und den Nutzern als Hinweise oder Leseempfehlungen unterbreitet werden.
Inhaltsbasierende Empfehlungen werden aufgrund von Ähnlichkeiten zwischen Nachrichten zusammengetragen. Grundlage für diese Empfehlungen sind Übereinstimmungen von Schlüsselwörtern einer Referenznachricht und anderen Nachrichten innerhalb der Infrastruktur.
Ähnlichkeiten zwischen Nutzern werden für kollaborative Empfehlungen genutzt. Auslöser für eine Empfehlung ist die positive Bewertung eines dem Referenznutzer ähnlichen Nutzers für eine Nachricht. Somit werden sämtliche positiven Bewertungen ähnlicher Nutzer als Empfehlungen verarbeitet.

KA #5: Automatische Generierung von Metainformationen zu Nachrichten (Zuordnung von Schlüsselwörtern zu Inhalten):

Die inhaltsbasierende Filterung von Nachrichten beruht auf der möglichst genauen Beschreibung von Inhalten durch Metainformationen bzw. Schlüsselwörter. Um hochwertige Ergebnisse bei der Filterung von Nachrichten zu erzielen, unterstützt die Infrastruktur ein Verfahren zur automatischen Suche und Zuordnung von bekannten Schlüsselwörtern zu Nachrichten. Dabei wird zwischen Schlüsselwörtern, Referenzen auf Personen sowie Referenzen auf „Communote“- Blogs unterschieden. Jede eingehende Nachricht wird nach dem System bereits bekannten Schlüsselwörtern durchsucht und im Fall von Übereinstimmungen diesen Schlüsselwörtern zugeordnet.

4.3.2 Bewertung der Aggregation und Übersetzung

Im Rahmen dieser Evaluation wurden heterogene Nachrichtenformate durch die Infrastruktur aggregiert. Dabei konnte festgestellt werden, dass alle Quellen korrekt erkannt und inhaltlich korrekt ausgelesen wurden. Lediglich beim Auslesen von Metainformationen von RSS-Feeds traten Ungenauigkeiten auf. Durch die Verwendung der Bibliothek „ROME“ zum Auslesen von Web-Feeds übertragen sich die Schwächen und Fehler dieser Bibliothek ebenfalls auf die Aggregationsfunktionalität der Infrastruktur.

Während der Aggregation des RSS 1.0 Feeds des Anbieters „semantic-web.at“ konnte das Publikationsdatum von einzelnen Nachrichtenelementen nicht korrekt ausgelesen werden, obwohl es im Quelldokument angegeben wurde. Während der Verarbeitung der Inhalte und Metainformationen wurde diese fehlende Metainformation mit dem Zeitpunkt des Auslesens ergänzt.

Alle anderen Inhalte, Metainformationen und Inhaltsformate konnten korrekt ausgelesen und in eine interne Repräsentation übertragen werden. Dies gilt ebenfalls für das Auslesen formatspezifischer Inhalte, wie beispielsweise „Twitter“- Nutzer, „Communote“- Nutzer und „Communote“- Blogs.

4.3.3 Bewertung des Nutzermodells

Bei der Akquise von Nutzermodellen durch die Nutzung des „social bookmarking“-Dienstes „Delicious“ wurden verschiedene „Delicious“- Benutzerkonten ausgelesen und in Nutzerprofile übersetzt. Dabei wurde deutlich, dass für eine hochqualitative Konvertierung bestimmte Voraussetzungen erfüllt sein müssen. Zum Einen müssen die jeweiligen Lesezeichen des „Delicious“- Benutzerkontos konsequent mit Schlüsselwörtern versehen sein. Zum Anderen müssen sich diese Schlüsselwörter auf Fachbegriffe bzw. domänenspezifische Terme beschränken. Wurden beispielsweise „falsche“ Schlüsselwörter wie „5.5“, „to“, „in“ oder „a“ angegeben, werden diese ebenfalls als Schlüsselwörter in das betreffende Nutzerprofil des Nutzermodells sowie in den zentralen Speicher für Schlüsselwörter der Anwendung übernommen. Dies führt unweigerlich zu einem fehlerhaften Verhalten der automatischen Suche nach Metainformationen sowie zu fehlerhaften und qualitativ minderwertigen Ergebnissen bei der Filterung.

Des Weitern wurde der Mechanismus zur Anpassung des Nutzermodells durch positive bzw. negative Bewertungen durch den Nutzer evaluiert. In Abhängigkeit der gewählten Anwendungskonfiguration werden die Werte der betreffenden Schlüsselwörter im Nutzermodell mehr oder weniger stark angehoben bzw. gesenkt. Ob die gewählten Werte in der Anwendungskonfiguration korrekt gewählt sind hängt im Wesentlichen vom Verhalten des Nutzers ab. Empfehlenswert sind hierbei niedrige Werte für Nutzer, die häufig Bewertungen abgeben sowie hohe Werte für Nutzer, die nur selten gelesene Nachrichten bewerten.

Darüber hinaus wurde bei der Evaluierung eine weitere Schwäche dieses Mechanismus deutlich. Diese tritt auf, wenn ein Nutzer eine Nachricht bewertet, die Schlüsselwörter enthält, die in seinem Nutzerprofil noch nicht verzeichnet sind, zudem aber ebenfalls Schlüsselwörter enthält, die in seinem Nutzerprofil mit hohen Interessenwerten versehen sind. Im Fall einer positiven Bewertung dieser Nachricht werden die bereits vorhandenen Werte des Nutzerprofils erhöht und die nicht vorhandenen Werte mit der in der Anwendungskonfiguration definierten initialen Relevanz versehen. Dieses Vorgehen hat in den meisten Fällen zur Folge, dass die errechnete Relevanz einer solchen Nachricht nach einer positiven Bewertung einen niedrigeren Wert aufweist als vor der positiven Bewertung. Empfehlenswert könnte an dieser Stelle die Verwendung des ursprünglichen Relevanzwertes für noch nicht

verzeichnete Schlüsselwörter im Nutzerprofil sein, so dass sichergestellt wird, dass sich der Relevanzwert der Nachricht nach einer positiven Bewertung erhöht.

4.3.4 Bewertung der automatischen Metadatensuche

Im Zuge der Evaluation der Aggregation und Übersetzung von externen Nachrichtenquellen wurden ebenfalls Prüfungen hinsichtlich der automatischen Metadatensuche durchgeführt. Dabei wurde das Verhalten der Infrastruktur unter den bereits in Abschnitt 3.2.4 genannten Fehlerfällen untersucht. Dabei wurde deutlich, dass keine weitere Prüfung hinsichtlich der Qualität bereits erkannter Schlüsselwörter durchgeführt wird. Dies bedeutet, dass fehlerhafte Schlüsselwörter aus der jeweiligen Quelle bzw. fehlerhafte Schlüsselwörter, die die Infrastruktur bereits verwaltet, ebenfalls durch das Verfahren der automatischen Suche nach Schlüsselwörtern verarbeitet werden. Empfehlenswert ist an dieser Stelle die Entwicklung einer automatischen Korrekturfunktion für Eingaben, wie diese bereits aus bestehenden Online-Angeboten wie beispielsweise der „Google“- Suche („Meinten Sie: “) bekannt ist.

Im Gegensatz zur Korrektur von fehlerhaften Schlüsselwörtern werden mehrfach verwendete Schlüsselwörter innerhalb einer Nachricht erkannt und dieser nur einmalig zugeordnet. Somit wird sichergestellt, dass keine Redundanzen bzw. überflüssigen Zuordnungen von Schlüsselwörtern hergestellt werden.

Darüber hinaus wurden Prüfungen hinsichtlich der Erkennung und Übersetzung von vorhandenen und fehlenden Metainformationen externer Quellen durchgeführt. Dabei konnte festgestellt werden, dass diese Funktionalität der Infrastruktur für die im Rahmen der Tests geprüften Quellen fehlerfrei arbeitet. Metainformationen, wie beispielsweise Autoren, Links, Publikationsdatum oder Format von Inhalten, konnten fehlerfrei erkannt werden.

Im Fall von fehlenden Metainformationen konnten diese für die folgenden Fälle aus anderen zur Verfügung stehenden Informationen gewonnen werden:

- Bei fehlendem Publikationsdatum wird das Datum der Aggregation und Übersetzung verwendet.
- Bei fehlenden Autoren und Links von einzelnen Nachrichtenelementen werden diese Informationen, sofern vorhanden, aus den Metainformationen der Quelle extrahiert und verwendet.

4.3.5 Bewertung von Anwendungskonfigurationen

Im Rahmen der Evaluierung von verschiedenen Anwendungskonfigurationen bzw. verschiedenen Einstellungen des inhaltsbasierenden sowie des kollaborativen Filters wurden die unterschiedlichen Anforderungen an die Einstellungsparameter des Filter- und Empfehlungssystems in Bezug auf sich ändernde Nutzer- und Bewertungszahlen deutlich. Im Zuge der Evaluierung konnte ein Anstieg von Empfehlungen für den Referenznutzer mit steigender Zahl von Nutzern bzw.

ähnlichen Nutzern im System nachgewiesen werden. Darüber hinaus konnte erwartungsgemäß die Anzahl der Empfehlungen durch stärkere Filtereinstellungen in der Anwendungskonfiguration gesenkt werden.

Mit dem Ziel, in jedem Zustand des Systems eine möglichst gleich große und überschaubare Menge Empfehlungen für Nutzer bereitzustellen, empfiehlt sich die adaptive Anpassung der Anwendungskonfiguration anhand aktueller Nutzer- und Bewertungszahlen des Systems.

Die gewählten Konfigurationen hatten ebenfalls Auswirkungen auf die Ergebnisse der Filterung, da die errechneten Relevanzwerte für Nachrichten zu einem variablen Teil aus Bewertungen ähnlicher Nutzer bestehen. Eine Empfehlung bezüglich einer Konfiguration dieser kollaborativen Anteile der Relevanzwerte von Nachrichten ist an dieser Stelle nicht möglich, da eine Aussage über deren Qualität nur durch eine Endnutzerevaluation erbracht werden kann.

Konfiguration / ermittelter Wert	**1**	**2**	**3**	**4**	**5**	**6**
Ausgegebene Empfehlungen	4	9	5	17	5	12
Anzahl ähnlicher Nutzer	1	4	2	9	2	7
Positive Bewertungen von ähnlichen Nutzern für Nachrichten des Referenznutzers	4	14	9	32	9	22
Negative Bewertungen von ähnlichen Nutzern für Nachrichten des Referenznutzers	0	7	2	11	2	11
Relevanzwert der wichtigsten Nachricht / Anteil von kollaborativen Boni und Strafen an diesem Wert	0,85 / 0,1	0,946 / 0,1946	0,91 / 0,16	0,980 / 0,23	0,84 / 0,09	0,8976 / 0,1476

Tabelle 15: Ergebnisse der Evaluation der Anwendungskonfigurationen

4.3.6 Bewertung der Ähnlichkeitsanalyse

Im Rahmen der Evaluation der Ähnlichkeitsanalyse wurden die Ergebnisse der Ähnlichkeitsberechnungen zu „Nutzer 1“ aus Tabelle 12 mit jedem der drei implementierten Ähnlichkeitsalgorithmen dokumentiert. Die nachfolgende Tabelle 16 gibt Aufschluss über die Ergebnisse dieser Evaluation:

Nutzer / Algorithmus	2	3	4	5	6	7	8	9	10
Vektorraum-Modell	0.835	0.545	0.542	0.690	0.584	0.477	0.357	0.578	0.969
Euklidische Distanz	0.899	0.75	0.500	0.774	0.605	0.611	0.507	0.560	0.894
Pearson Korrelation	0.950	1	0.226	0.710	0.098	0.299	0.127	0.195	0.915

Tabelle 16: Ergebnisse des Algorithmenvergleichs

Aus der dargestellten Tabelle 16 wird deutlich, dass der Algorithmus „Vektorraum-Modell“, gemessen an den Ausgangsdaten, zu vergleichsweise genauen und nachvollziehbaren Ergebnissen führt.

Der Algorithmus „Euklidische Distanz“ [SEGARAN+07] errechnet in sechs von neun Fällen höhere Ähnlichkeitswerte als der Algorithmus des „Vektorraum-Modells“. Darüber hinaus ist festzustellen, dass auch bei der Erwartung von niedrigen Ähnlichkeiten, wie beispielsweise im Fall der Berechnung zu Nutzer 8, der Algorithmus zu verhältnismäßig hohen Ergebnissen führt. Daher wird empfohlen, bei einer Verwendung der „Euklidischen Distanz“ die Anwendungskonfiguration anzupassen und den Wert für die minimale Ähnlichkeit zweier Nutzer entsprechend zu erhöhen, um auf diese Eigenschaft des Algorithmus zu reagieren.

Bei der Verwendung des Algorithmus der „Pearson-Korrelation“ wurde erwartet, dass durch die Einbeziehung der durchschnittlichen Werte der Nutzerprofile höhere Ähnlichkeiten bei Nutzern errechnet werden, die proportional höhere oder niedrigere Werte in ihrem Nutzerprofil aufweisen. Diese Erwartung konnte im Fall von Nutzer 3 erfüllt werden, jedoch führte die Berechnung bei anderen Nutzern zu Ergebnissen, die in keinster Weise den Erwartungen entsprach. Beispielsweise wurde zwischen dem Referenznutzer 1 und Nutzer 6 eine mittlere Ähnlichkeit erwartet. Errechnet wurde jedoch eine minimale Ähnlichkeit, die keinesfalls ein korrektes Ergebnis darstellt. Daher ist von einer Verwendung des Algorithmus „Pearson-Korrelation“ innerhalb der implementierten Infrastruktur abzusehen.

4.3.7 Bewertung der Skalierbarkeit und Leistungsfähigkeit

Im Laufe der Evaluierung der Skalierbarkeit und Leistungsfähigkeit wurden sechs Tests durchgeführt, um Aufschluss über die Steigerung von Lade- und Berechnungszeiten bei steigenden Nutzer- und Bewertungszahlen zu geben. Dabei wurde deutlich, dass die Berechnung von kollaborativen Boni und Strafen den Hauptteil der Gesamtladezeit eines Nachrichtenstroms ausmacht. Darüber hinaus konnte hinsichtlich der Skalierbarkeit der Infrastruktur festgestellt werden, dass sich die Gesamtladezeit eines kompletten Nachrichtenstroms bei einer Verdopplung der zu ladenden Nachrichtenelemente durchschnittlich um den Faktor 2,77 erhöht.

Zudem ist zu beachten, dass Zugriffe auf Relationen zwischen Entitäten unverhältnismäßig zeitaufwendig sind. Beispielsweise betrug die Dauer zum Laden

aller 80 Nachrichtenelemente des Nutzers in Test 6 276 Millisekunden. Die Zugriffe auf die Relationen zwischen diesen Nachrichtenelementen und das Auslesen der 44 Schlüsselwörter benötigten hingegen 387 Millisekunden.

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
Laden der Nachrichtenelemente	41 ms	39 ms	131 ms	141 ms	285 ms	276 ms
Berechnen der Relevanzwerte der Nachrichten	51 ms	50 ms	198 ms	193 ms	451 ms	508 ms
Laden von Autoren und Links der Nachrichtenelemente	27 ms	26 ms	81 ms	80 ms	150 ms	186 ms
Laden von Schlüsselwörtern der Nachrichtenelemente	35 ms	35 ms	143 ms	153 ms	350 ms	387 ms
Laden der Bewertungen der Nachrichtenelemente	30 ms	28 ms	105 ms	114 ms	268 ms	275 ms
Finden ähnlicher Nutzer	76 ms	72 ms	76 ms	82 ms	97 ms	101 ms
Berechnen von kollaborativen Boni und Strafen	93 ms	169 ms	330 ms	722 ms	885 ms	1633 ms
Gesamtzeit zum Laden des kompletten Nachrichtenstroms	367 ms	430 ms	1116 ms	1551 ms	2580 ms	3478 ms

Tabelle 17: Ergebnisse der Evaluierung der Skalierbarkeit und Leistungsfähigkeit

5 Verbesserungsmöglichkeiten

Während der Evaluierung der prototypischen Infrastruktur konnten einige Schwachstellen sowie Verbesserungsmöglichkeiten des Konzeptes und der Implementierung herausgestellt werden. Darüber hinaus ergaben sich während der Implementierungs- und Testphase weitere Ideen, die in einer Weiterentwicklung der Infrastruktur verwendet werden könnten. In den nachfolgenden Abschnitten werden diese Punkte aufgezeigt und lassen somit einen Ausblick auf zukünftige Entwicklungen zu.

5.1 Nutzermodell und automatische Metadatensuche

Im Rahmen der Evaluierung wurden bereits einige Verbesserungsmöglichkeiten des Nutzermodells dargelegt. In diesem Zusammenhang wurde die Prüfung auf korrekte Schlüsselwörter beim Import von „Delicious"- Nutzerprofilen genannt, um zu vermeiden, dass Schlüsselwörter der Form „a", „is" oder „to" in den Schlüsselwortspeicher der Infrastruktur übernommen werden. Dieses Problem besteht ebenfalls beim Auslesen von „Hashtags" aus „Twitter"-Nachrichten und „Communote"-Nachrichten sowie nativen Schlüsselwortangaben von Web-Feeds. Zur Lösung dieses Problems könnte die Einführung eines Verfahrens zur Analyse neuer potentieller Schlüsselwörter beitragen, mit dessen Hilfe es möglich ist Schlüsselwörter der bereits genannten Form herauszufiltern. Zudem ist die Nutzung von „Eingabe Korrektur"-Verfahren [GOOGLE-01], wie sie aus der „Google"-Suche [GOOGLE-02] bekannt sind, empfehlenswert, um nicht korrekte Schreibweisen von Wörtern zu korrigieren.

Darüber hinaus ist es denkbar, externe Dienste wie beispielsweise „OpenCalais" [OpenCalais] für die Suche nach Schlüsselwörtern einzusetzen. Dabei wird eine Wortkette via Webserviceaufruf an den externen Dienst übermittelt. Diese wird anhand semantischer Gesetzmäßigkeiten analysiert und auf Schlüsselwörter durchsucht. Werden Terme gefunden, die auf ein Schlüsselwort hinweisen, werden diese markiert und in der Antwort des Webserviceaufrufes zum Client übermittelt.

Zur Steigerung der Qualität des Nutzermodells ist es außerdem empfehlenswert, bestimmte Schlüsselwörter zu verknüpfen. In der Praxis beschränkt sich das Interesse eines Nutzers für ein Schlüsselwort oftmals auf den Fall, dass dieses Schlüsselwort in Kombination mit einem weiteren Schlüsselwort auftritt. Die Darstellung von Interessenwerten im Nutzermodell für Schlüsselwortkombinationen ist daher ratsam. Somit könnten Interessen der Form „C# und Webservices" bzw. „Java und Webservices" abgebildet werden, um auszudrücken, dass sich ein Nutzer nicht generell für das Thema „Webservices" interessiert, sondern nur in Verbindung mit einer bestimmten Programmiersprache.

Für die Bestimmung von Schlüsselwortkombinationen ist es denkbar, ebenfalls auf „social bookmarking"-Dienste wie „Delicious" zurückzugreifen. Die in diesen Profilen hinterlegten Schlüsselwörter sind an die jeweiligen Lesezeichen gebunden,

wobei davon ausgegangen werden kann, dass die Schlüsselwörter eines Lesezeichens in Beziehung zueinander stehen. Auf Basis dieser Annahme ist es prinzipiell möglich, Schlüsselwortkombinationen in den zugeordneten Nutzerprofilen der Infrastruktur zu bilden.

Eine weitere Verbesserungsmöglichkeit der Infrastruktur im Umgang mit Schlüsselwörtern stellt die Einführung von Synonymen und Namensräumen dar. Bezüglich der Bildung von Synonymen ist eine Definition von Relationen zwischen Schlüsselwörtern notwendig. Somit ist es möglich, Themen oder Begriffe, die durch unterschiedliche Schlüsselwörter dargestellt werden können, als Schlüsselwortgruppe abzubilden:

“Spring Security” = {“Spring Security”; “ACEGI”; “ACEGI Security”}

Beispiel einer Schlüsselwortgruppe bzw. Synonymabbildung

Des Weiteren ist die Definition von Namensräumen für Schlüsselwörter eine sinnvolle Erweiterung der Infrastruktur. Die vorliegende Implementierung ist nicht in der Lage, zwischen Schlüsselwörtern gleicher Schreibweise, die jedoch eine unterschiedliche Bedeutung haben, zu unterscheiden. Das heißt, Schlüsselwörter wie beispielsweise „Spring“ oder „Paris“ werden von der Anwendung ungeachtet ihrer Bedeutung verarbeitet. Somit wird keine Unterscheidung getroffen, ob es sich bei dem Wort „Spring“ um das Framework oder den Frühling, bzw. bei dem Wort „Paris“ und einen Stadtnamen oder einen Personennamen handelt.

5.2 Bewertungssystem

Das Bewertungssystem der Infrastruktur basiert in der vorliegenden Form ausschließlich auf expliziten Bewertungsverfahren. Das heißt, die Repräsentation von Nutzerinteressen im Nutzermodell wird nur im Fall einer explizit abgegebenen positiven oder negativen Bewertung für ein Nachrichtenelement angepasst. Hinsichtlich einer Verbesserung der Verfahrensweise zur Anpassung von Nutzerinteressen ist die Einführung von impliziten Bewertungsverfahren wünschenswert. Hierzu könnten Informationen durch eine Beobachtung des Nutzers beim Umgang mit der entsprechenden Frontendanwendung gesammelt werden. Diese Informationen, wie beispielsweise die Lesedauer von Nachrichtenelementen oder das Verschieben von Nachrichten auf die „noch zu lesen“-, „gelesen“- oder „geschlossen“- Liste, sind Indikatoren für das Interesse eines Nutzers an den Themen des betreffenden Nachrichtenelementes. Eine Auswertung dieser Informationen kann zur Verbesserung des Nutzermodells beitragen, insbesondere bei Nutzern, die von der Möglichkeit, explizite Bewertungen abzugeben, absehen.

5.3 Filterung

Die Filterung von Nachrichtenelementen eines Nutzers basiert zum Einen auf Bewertungen ähnlicher Nutzer zu den abonnierten Nachrichtenelementen und zum Anderen auf der Relevanzberechnung von den Nachrichten zugeordneten Schlüsselwörtern für den Nutzer. In diesem Verfahren werden Vorlieben für bestimmte Nachrichtenquellen nicht berücksichtigt. Daher besteht eine Verbesserungsmöglichkeit der Infrastruktur in der Möglichkeit, abonnierte Nachrichtenquellen mit Interessenwerten oder Prioritäten zu versehen. Dies bietet einerseits den Vorteil, dass die Filterung noch genauer und qualitativ hochwertiger ausgeführt werden könnte. Andererseits birgt diese Erweiterung einen Lösungsansatz, um Nachrichtenelemente, denen kein Schlüsselwort zugeordnet werden konnte, über die Quellenrelevanz genauer zu beschreiben.

Um eine solche Quellenrelevanz bzw. Priorität zu bestimmen ist es denkbar, zu ermitteln, ob die Nachrichtenelemente dieser Quelle oft oder weniger oft gelesen bzw. positiv bewertet wurden.

Des Weiteren könnte ebenfalls die Berechnung von Ähnlichkeiten zwischen Nutzern durch die Bestimmung von gemeinsam abonnierten Quellen bzw. ähnlichen Quellenrelevanzen erweitert werden. Diese Verfahrensweise ist insbesondere dann von Vorteil, wenn über die implementierte Ähnlichkeitsberechnung keine ähnlichen Nutzer gefunden werden können.

5.4 Implementierung

Die Implementierung eines Prototyps für eine Infrastruktur ist ein weiterer wesentlicher Bestandteil dieser Arbeit. Hierbei handelt es sich um eine Anwendung, die Verbesserungsmöglichkeiten offen lässt.

Durch die Evaluierung wurde deutlich, dass die Leistungsfähigkeit der Anwendung in einer weiteren Entwicklung verbessert werden sollte. Dabei können die Ergebnisse der Evaluation als Leitfaden für Nachbesserungen genutzt werden. Somit ist es empfehlenswert, verschiedene Verfahren zur Leistungssteigerung zu integrieren. Dabei erscheint es sinnvoll, das Nutzermodell einem „clustering“ zu unterziehen, wie dies in Abschnitt 2.1.5.2 unter „Modellbasierte Verfahren“ beschrieben wurde. Darüber hinaus sollten geeignete Zwischenspeicher eingerichtet werden, um bereits errechnete Relevanzwerte zu Nachrichtenelementen zu speichern, so dass aufwändige Relevanzberechnungen nicht redundant ausgeführt werden müssen.

6 Zusammenfassung und Fazit

Nach der Beschreibung von Grundlagen, Konzeption und Evaluierung wird in diesem Kapitel ein abschließendes Fazit bezüglich der vorliegenden Arbeit gezogen.

Während dieser Arbeit wurden in den verschiedenen Phasen der Entwicklung folgende Ergebnisse erzielt:

1. Analyse aktueller Nachrichtenaggregatoren für den Consumer- sowie den Enterprise- Bereich
2. Entwicklung und Implementierung eines Konzeptes für eine Infrastruktur zur Aggregation heterogener semistrukturierter Nachrichtenquellen
3. Konzeption und Implementierung eines Personalisierungskonzeptes bzw. eines Nutzermodells
4. Konzeption und Implementierung von Filter- und Empfehlungsverfahren für aggregierte Nachrichtenelemente
5. Evaluierung der implementierten Infrastruktur sowie Aufdeckung von Verbesserungsmöglichkeiten und Ansätzen für weiterführende Arbeiten

Analyse aktueller Nachrichtenaggregatoren für den Consumer- sowie den Enterprise- Bereich:
Während der Analyse bestehender Nachrichtenaggregatoren wurde deutlich, dass auf diesem Gebiet eine Vielzahl von Lösungen vorhanden ist. Darüber hinaus konnte festgestellt werden, dass diese Lösungen in den meisten Fällen nur ein spezifisches Nachrichtenformat unterstützen und somit nicht für die Aggregation heterogener Nachrichtenquellen geeignet sind. Zudem verfügen nur wenige Aggregatoren über eine personalisierte Filterung von Inhalten oder gar ein Empfehlungssystem für interessante Nachrichten.
Abschließend kann somit festgestellt werden, dass derzeit keine Lösung existiert, die sowohl heterogene Nachrichtenquellen aggregiert, als auch ein personalisiertes Filter- und Empfehlungssystem anbietet.

Entwicklung und Implementierung eines Konzeptes für eine Infrastruktur zur Aggregation heterogener semistrukturierter Nachrichtenquellen:
Im Rahmen dieser Arbeit wurde eine Infrastruktur entwickelt, mit deren Hilfe es möglich ist, heterogene semistrukturierte Nachrichtenquellen zu aggregieren. In der vorliegenden Implementierung wurde eine Unterstützung für die Nachrichtenquellen „Web-Feeds“, „Twitter“ und „Communote“ umgesetzt. Darüber hinaus ist die Infrastruktur so konzipiert, dass eine Erweiterung um andere Formate bzw. Nachrichtenquellen mit geringem Aufwand möglich ist.

Konzeption und Implementierung eines Personalisierungskonzeptes bzw. eines Nutzermodells:
Als Grundlage für die Entwicklung eines Filter- und Empfehlungssystems wurde ein Konzept zur Personalisierung des Angebots erstellt. Dabei handelt es sich im Wesentlichen um die Repräsentation von Nutzerinteressen in einer geeigneten Datenstruktur. Diese Datenstruktur bzw. dieses Nutzermodell basiert auf dem Konzept, dass Nutzer mit unterschiedlichen Interessenwerten für bestimmte Themen abgebildet werden können. Diese Themen wiederum können durch Schlüsselwörter ausgedrückt werden. Darüber hinaus umfasst dieses Konzept ebenfalls die Pflege der einzelnen Nutzerprofile, indem explizite Bewertungsverfahren verwendet werden, um die sich ändernden Interessen anzupassen und neue Interessen einfließen zu lassen.

Konzeption und Implementierung von Filter- und Empfehlungsverfahren für aggregierte Nachrichtenelemente:
Aufbauend auf dem erstellten Personalisierungskonzept wurden Verfahren zur Filterung und Empfehlung von Nachrichten konzipiert und implementiert. Dabei wurden sowohl inhaltsbasierte Verfahren genutzt, als auch kollaborative bzw. gruppenbasierte Verfahren. An dieser Stelle ist anzumerken, dass die entwickelte Filterung inhaltsbasierende und kollaborative Verfahren vereint, um eine möglichst hochqualitative und differenzierte Filterung zu erreichen.

Evaluierung der implementierten Infrastruktur sowie Aufdeckung von Verbesserungsmöglichkeiten und Ansätzen für weiterführende Arbeiten:
Abschießend wurden die entwickelten Konzepte evaluiert und Verbesserungsmöglichkeiten der Infrastruktur aufgezeigt, um Ansätze für weitere Arbeiten zu liefern. Während der Evaluierung wurden Schwachstellen der Implementierung und Verbesserungsmöglichkeiten des Konzeptes aufgedeckt und erarbeitet. Hauptsächlich bezogen sich diese Erkenntnisse auf die Leistungsfähigkeit und Skalierbarkeit der Implementierung sowie auf die Konzepte zur automatischen Metadatensuche und Schlüsselworterkennung.

Fazit:
Die vorliegende Arbeit zeigt Lösungsmöglichkeiten für die Umsetzung einer Anwendung zur personalisierten Filterung und Empfehlung von Nachrichten aus semistrukturierten Quellen auf. Dabei implementiert die entstandene Infrastruktur einen Großteil dieser Ansätze auf prototypische Art und Weise und gibt somit einen Eindruck, welche Möglichkeiten sich durch ein solches System ergeben. Dadurch bietet diese Arbeit einen Ausblick auf zukünftige Formen der Informationsbereitstellung, die es Endnutzern ermöglichen, die unüberschaubare Informationsflut ihrer verschiedenen Nachrichtenabonnements auf die persönlich relevantesten Inhalte zu reduzieren.

A Anhang

A.1 Inhalt der DVD

- /Diplomarbeit/PDF – Diese Arbeit im PDF Format
- /Diplomarbeit/Konfigurationen – SQL Dateien mit den referenzierten Evaluierungskonfigurationen
- /MDA – Projektordner der implementierten Infrastruktur inkl. des UML 1.4 Modells der Anwendung
- /WAR – Die implementierte Infrastruktur inkl. aller abhängigen Bibliotheken als *.war Datei
- /M2_REPO – Das Maven 2 Repository mit allen notwendigen Bibliotheken um ein „build“ des Projektordners auszuführen
- /JavaDoc – JavaDoc der Klassen der Anwendung
- /Quellen – Quellen der Arbeit, die als PDF vorliegen
- /Präsentationen – Präsentationen, die im Verlauf der Diplomarbeit vorgetragen wurden

A.2 Klassendiagramme

Nachfolgend werden Klassendiagramme, die nicht in den Text der Diplomarbeit eingebunden wurden, abgebildet. Dabei ist zu beachten, dass es sich hierbei um Klassen handelt, die mit Hilfe des „MDA“-Toolkits „AndroMDA“ generiert wurden. Die angegebenen Funktionen wurden in der Implementierung mit der entsprechenden Anwendungslogik gefüllt. „Classifier“-Methoden von Entitäten, die im „UML“-Modell unterstrichen sind, werden dabei in separate „Data Access Objects“ generiert und sind nicht Bestandteil der Entitätenklassen.

Darüber hinaus existieren weitere Klassen, die der Anwendung ohne die Nutzung von „AndroMDA“ hinzugefügt wurden. Diese sind jedoch nicht Teil der nachfolgenden Diagramme.

A.2.1 Entitäten

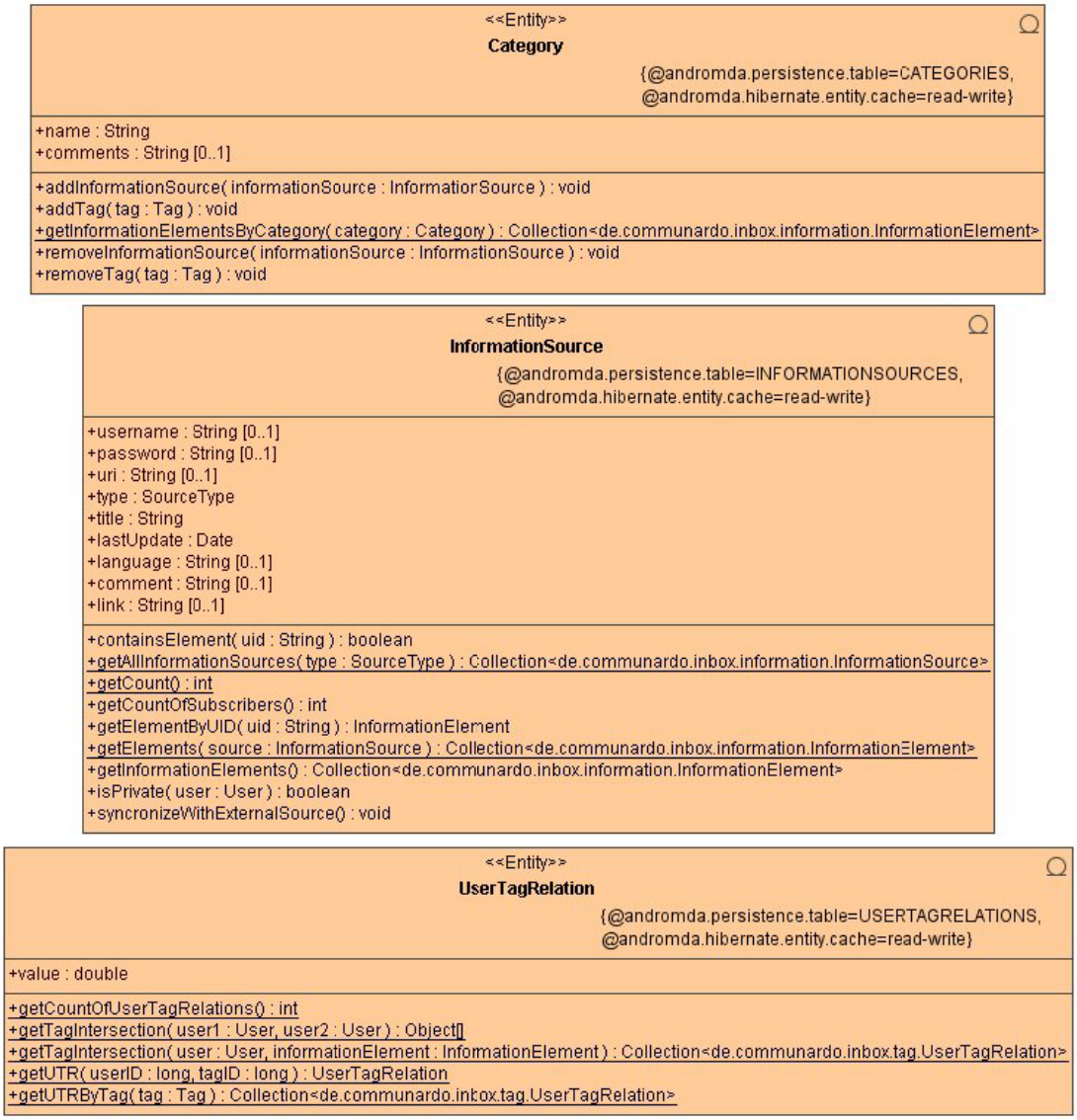

Abbildung 31: Klassendiagramm Entitäten (1)

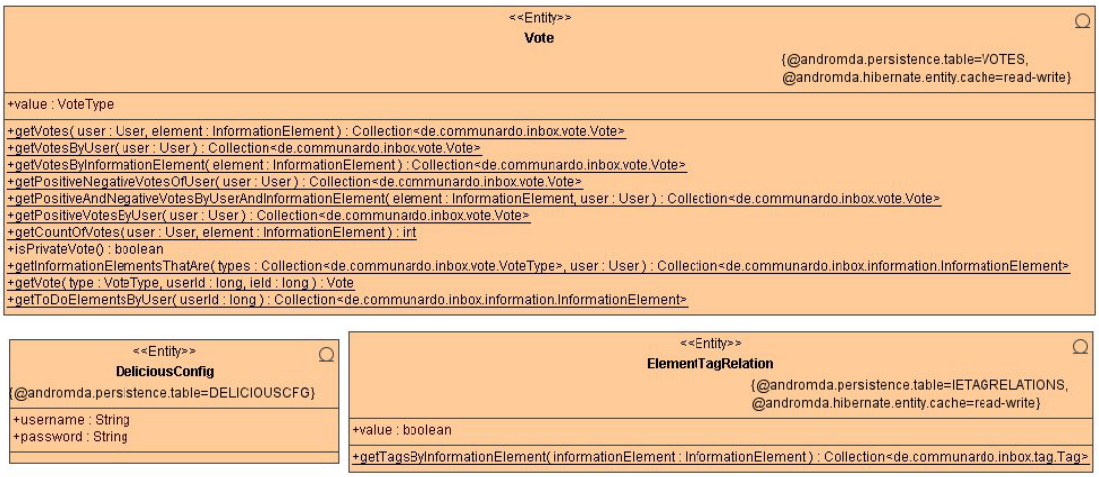

Abbildung 32: Klassendiagramm Entitäten (2)

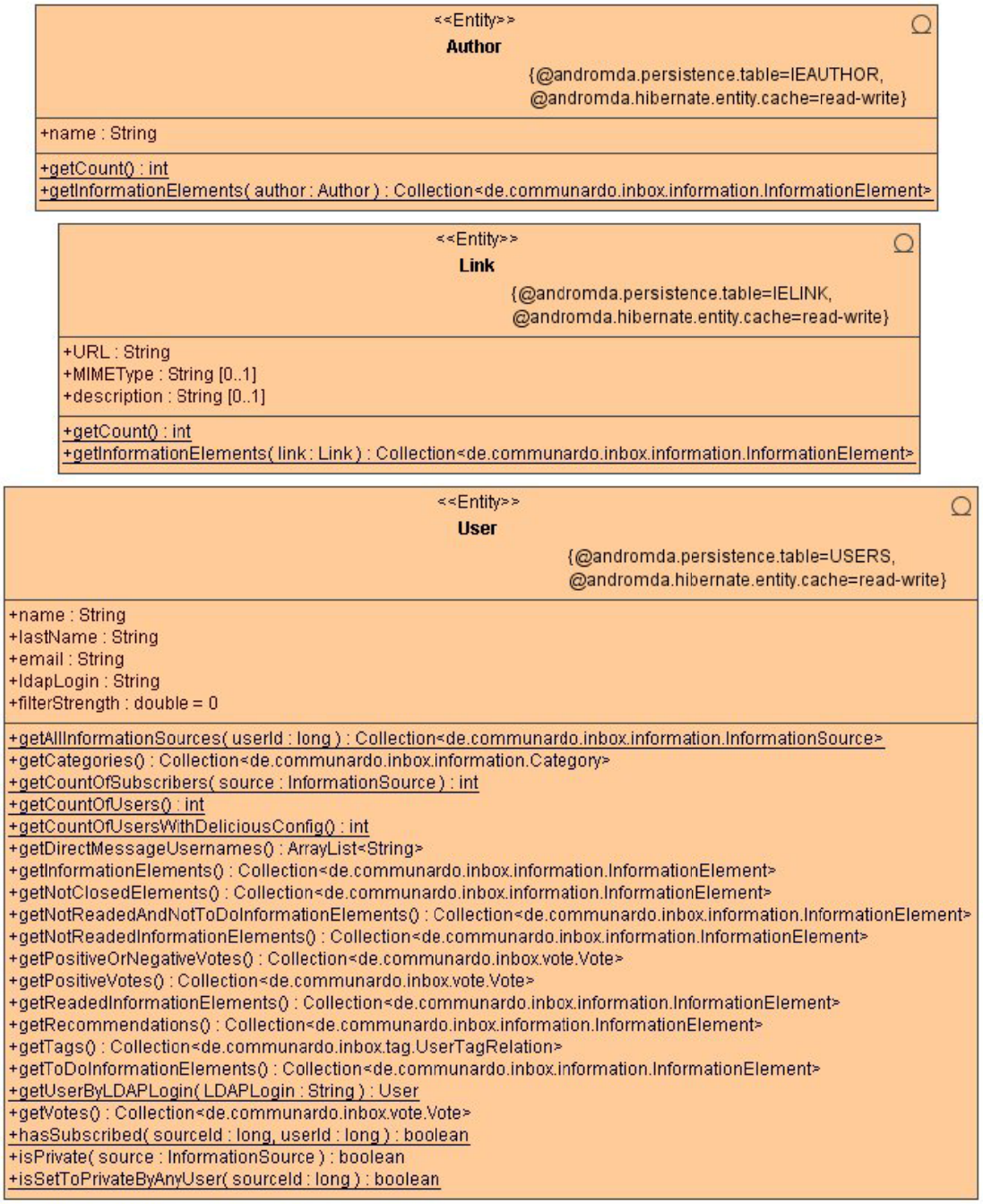

Abbildung 33: Klassendiagramm Entitäten (3)

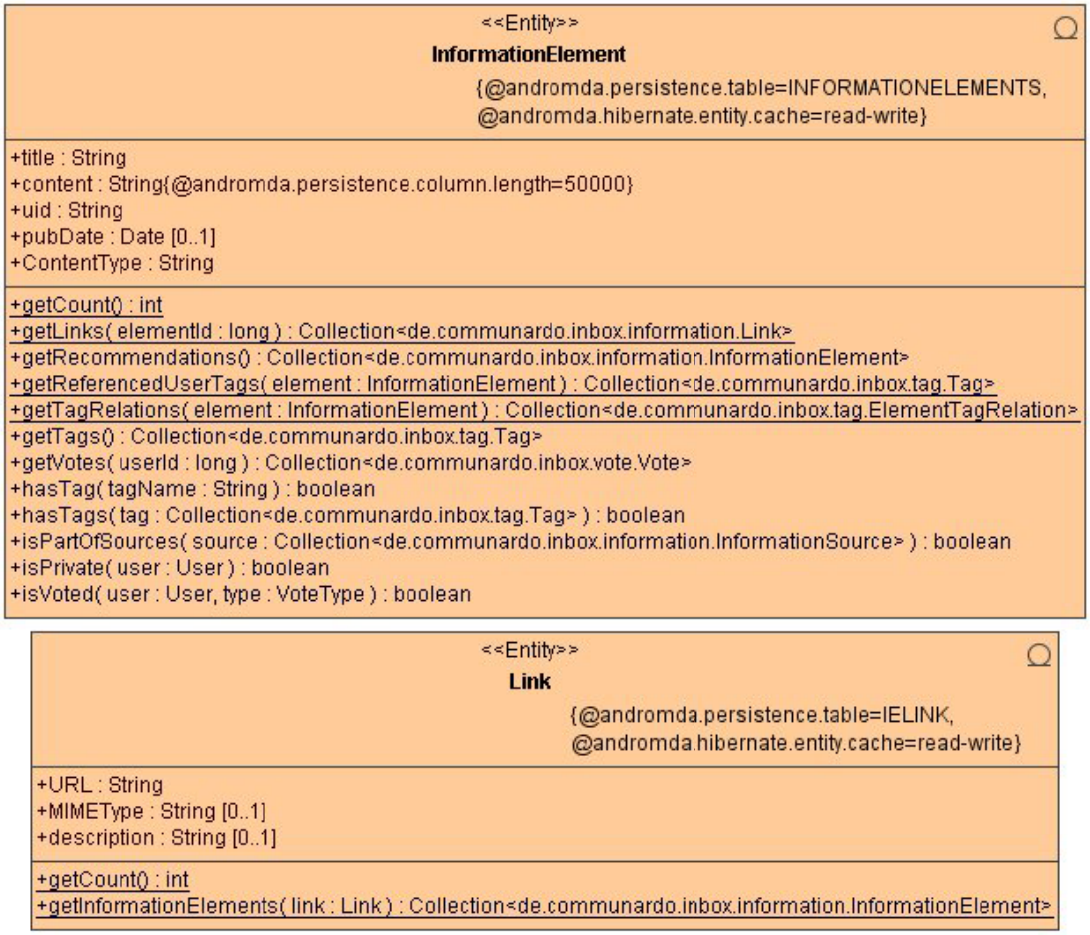

Abbildung 34: Klassendiagramm Entitäten (4)

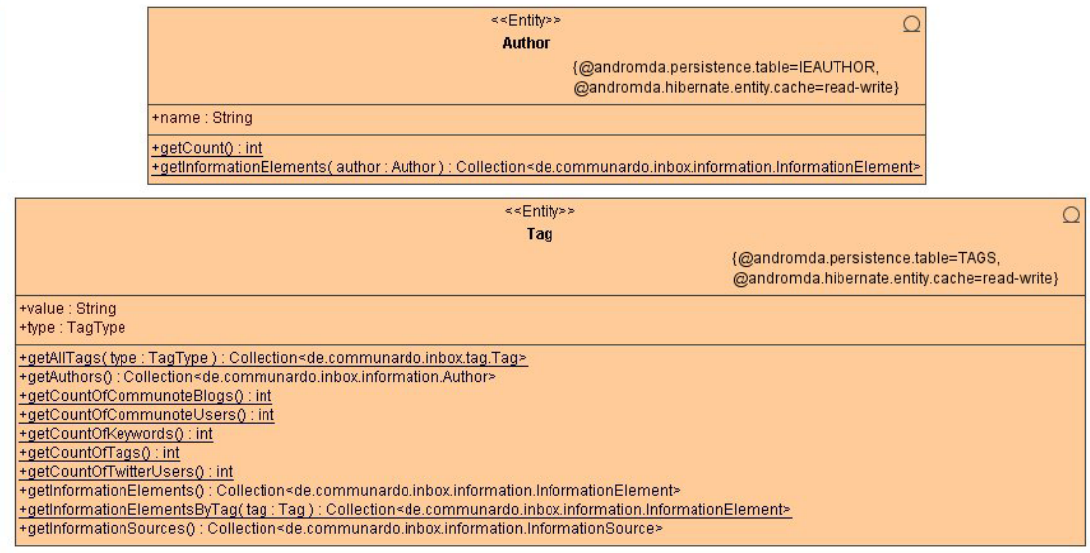

Abbildung 35: Klassendiagramm Entitäten (5)

A.2.2 Servicelayer

Abbildung 36: Klassendiagramm „Servicelayer“ (1)

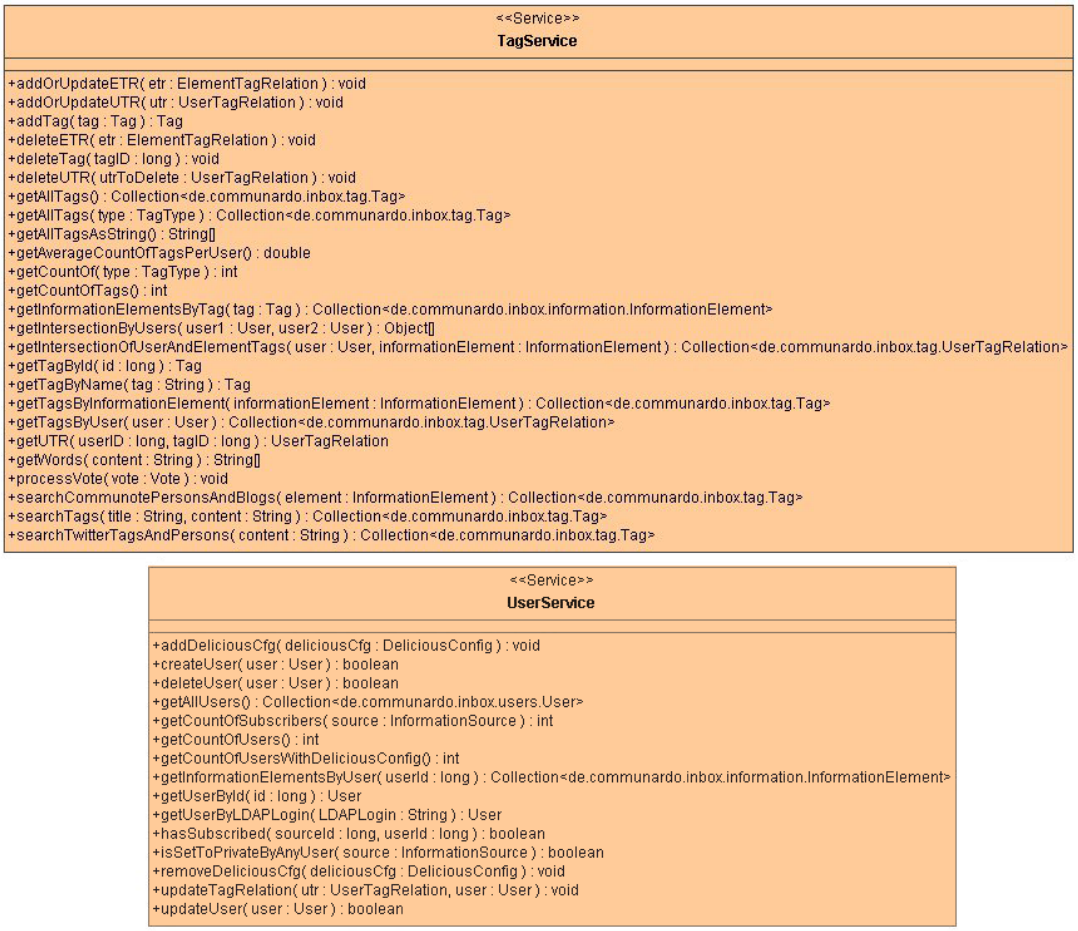

Abbildung 37: Klassendiagramm „Servicelayer“ (2)

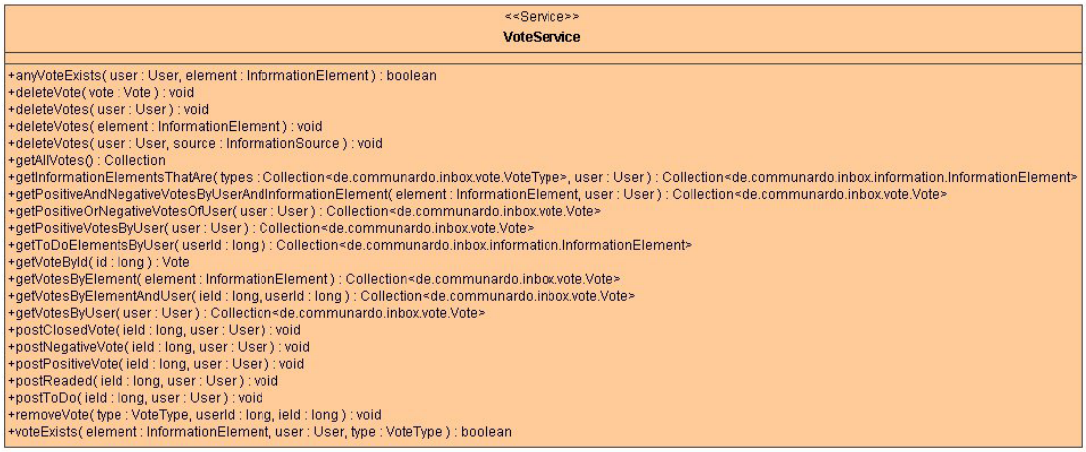

Abbildung 38: Klassendiagramm „Servicelayer“ (3)

A.2.3 Dienste für externe Zugriffe

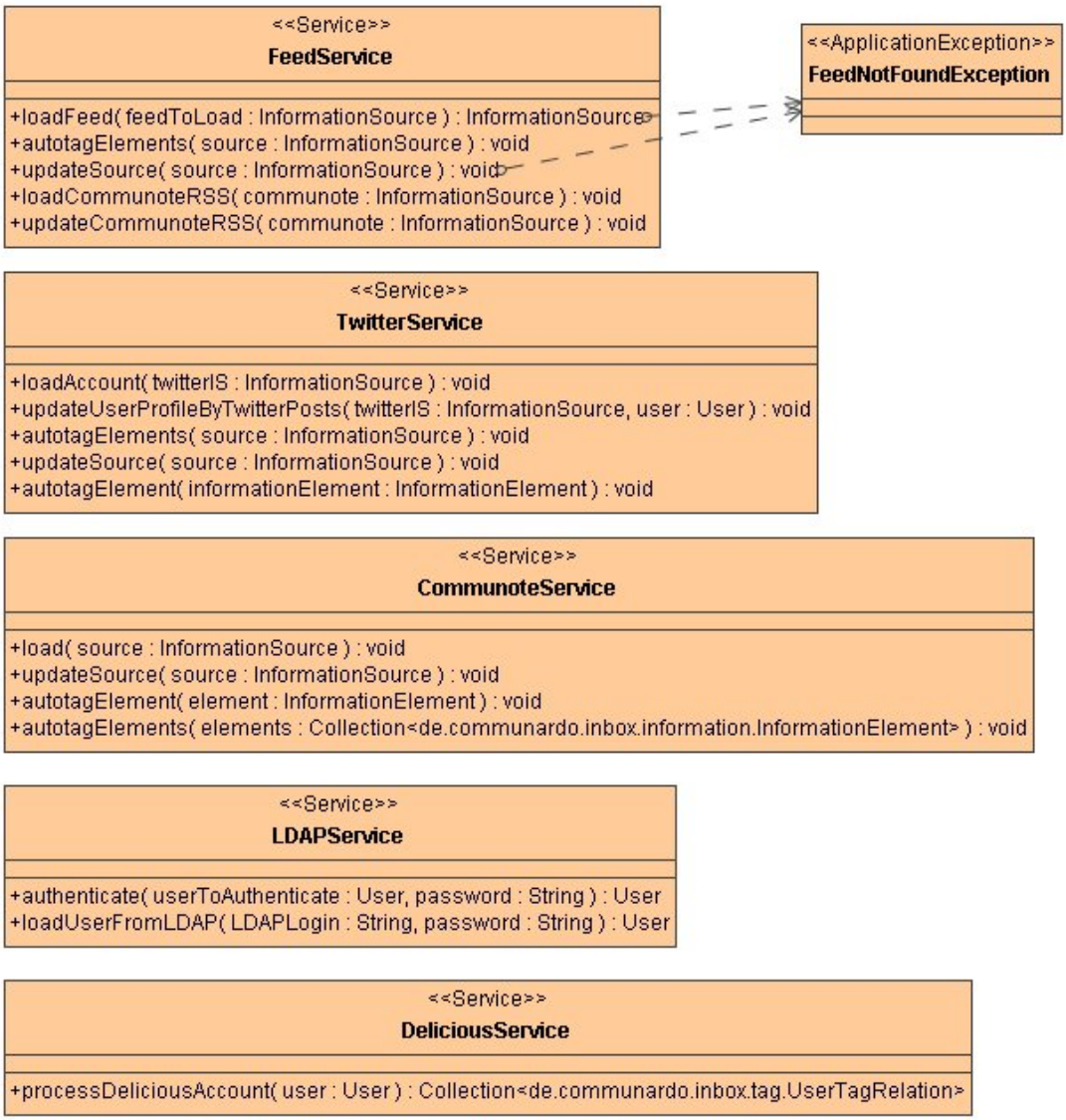

Abbildung 39: Klassendiagramm Dienste für externe Zugriffe

A.2.4 Filterengine

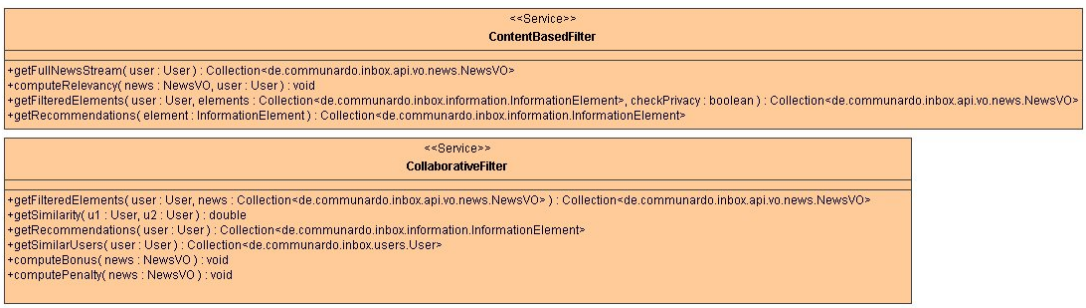

Abbildung 40: Klassendiagramm Filterengine

A.2.5 Schnittstelle

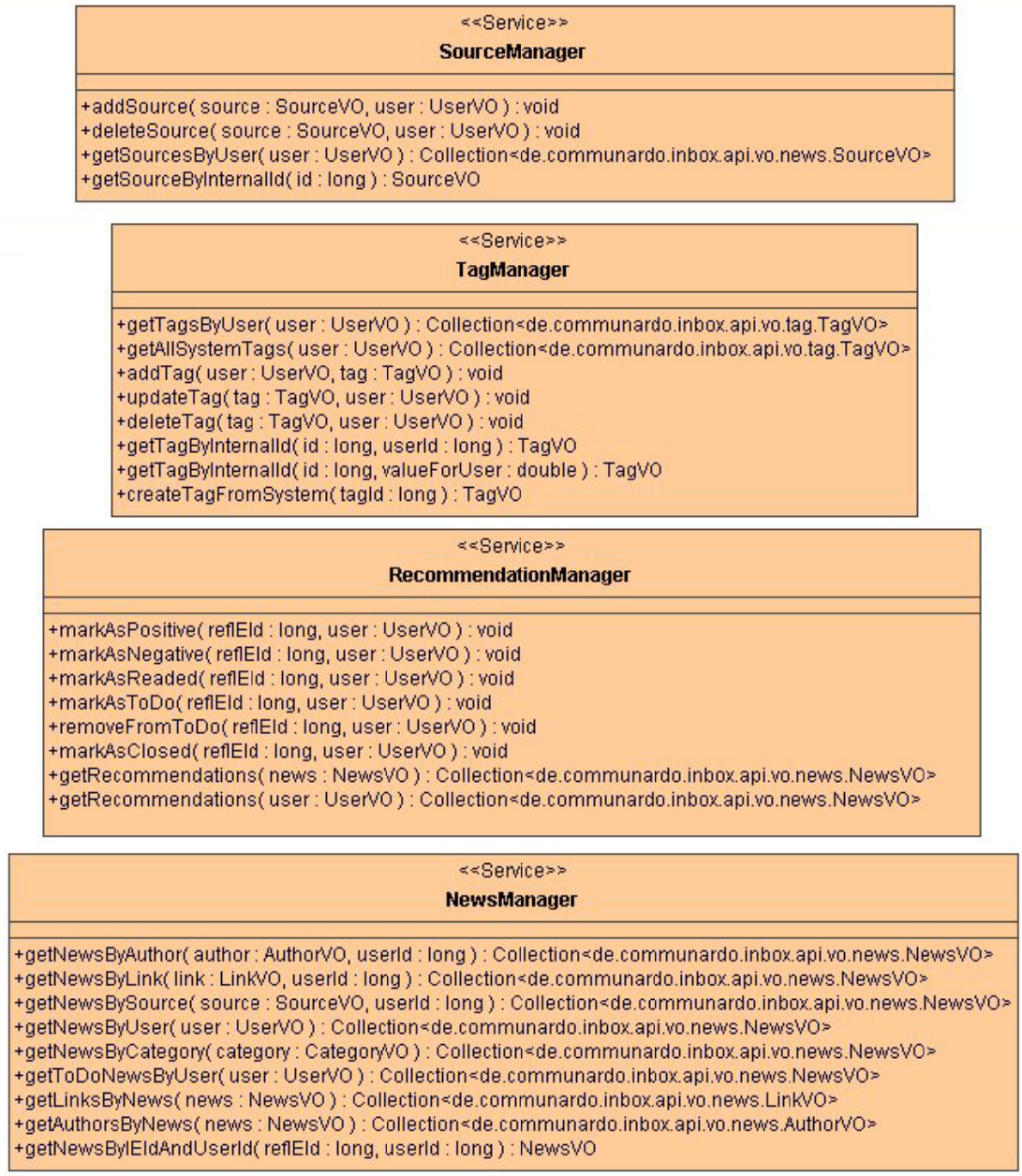

Abbildung 41: Klassendiagramm „API Service Classes“ (1)

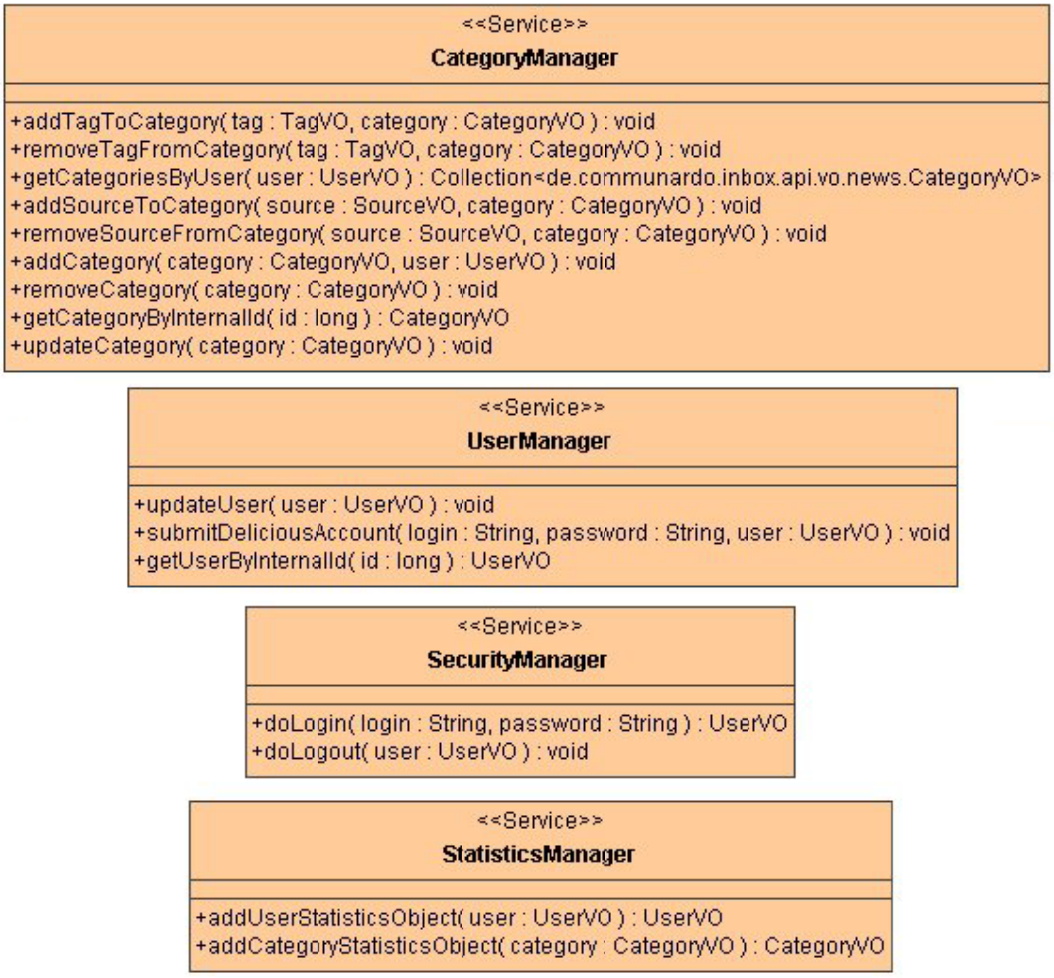

Abbildung 42: Klassendiagramm „API Service Classes“ (2)

A.2.5.1 API Value Objects

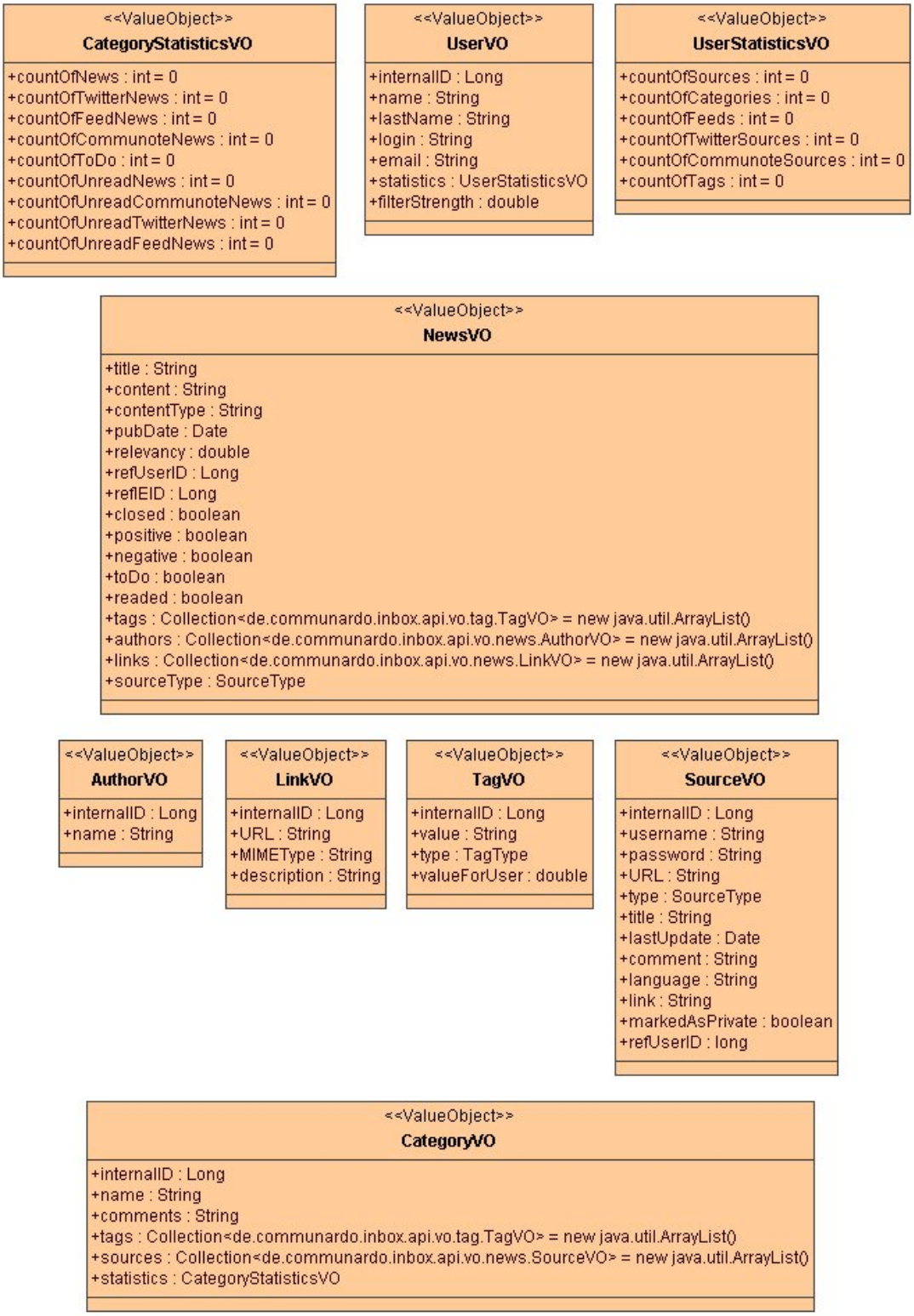

Abbildung 43: Klassendiagramm API Value Objects

A.2.6 Anwendungskonfiguration

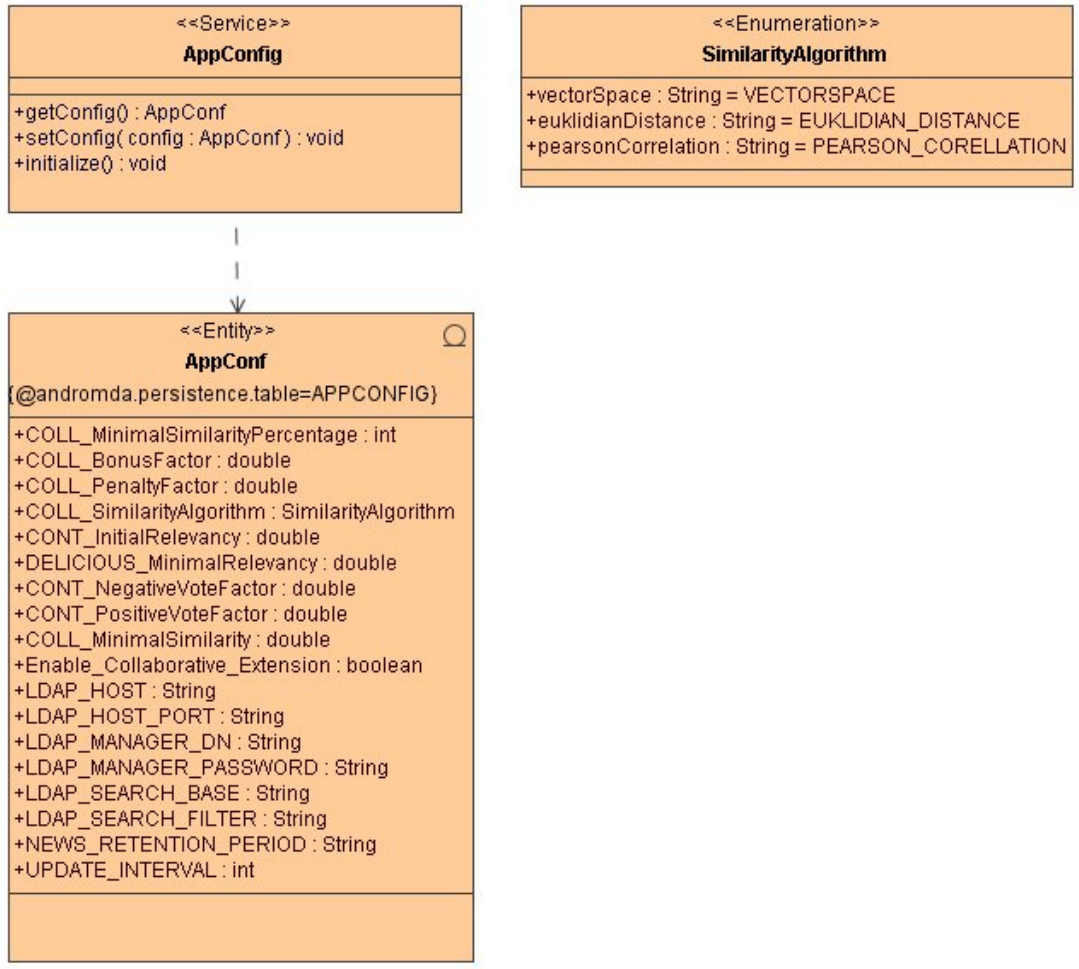

Abbildung 44: Klassendiagramm Anwendungskonfiguration

A.2.7 Deliciouskonfiguration

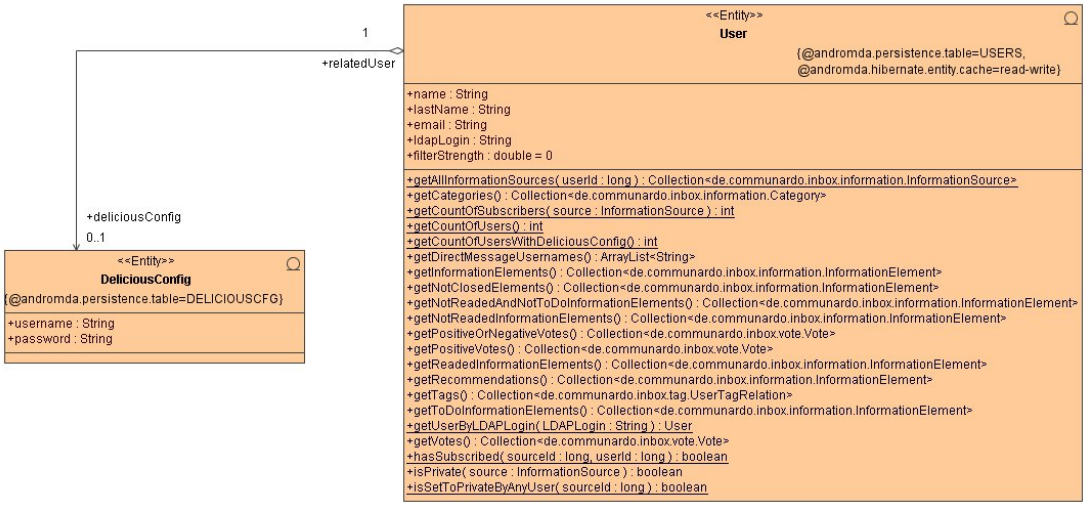

Abbildung 45: Klassendiagramm Beziehung User – Delicious Konfiguration

A.2.8 Sicherheitskomponente

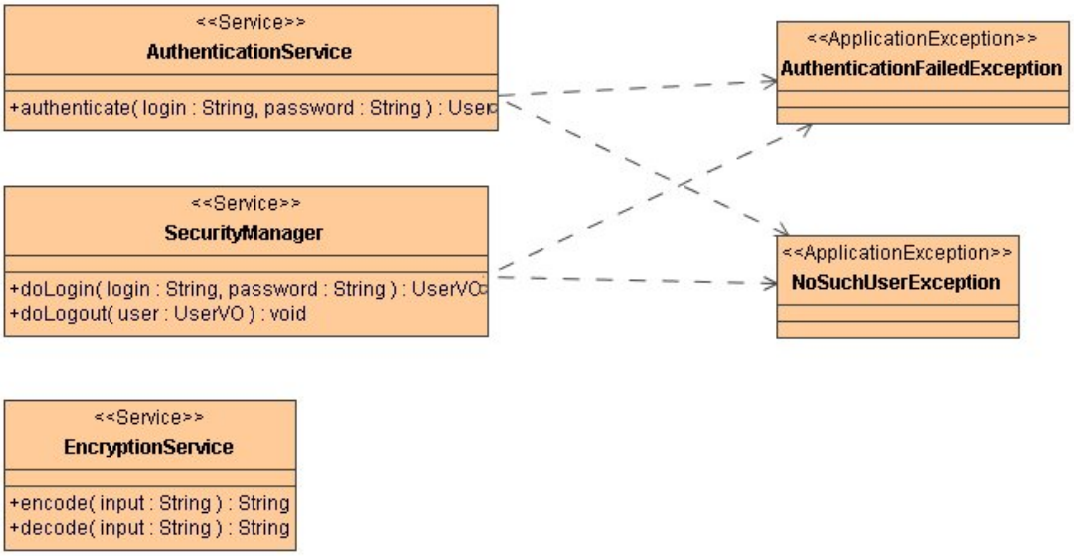

Abbildung 46: Klassendiagramm Sicherheitskomponente

A.2.9 Enummerations

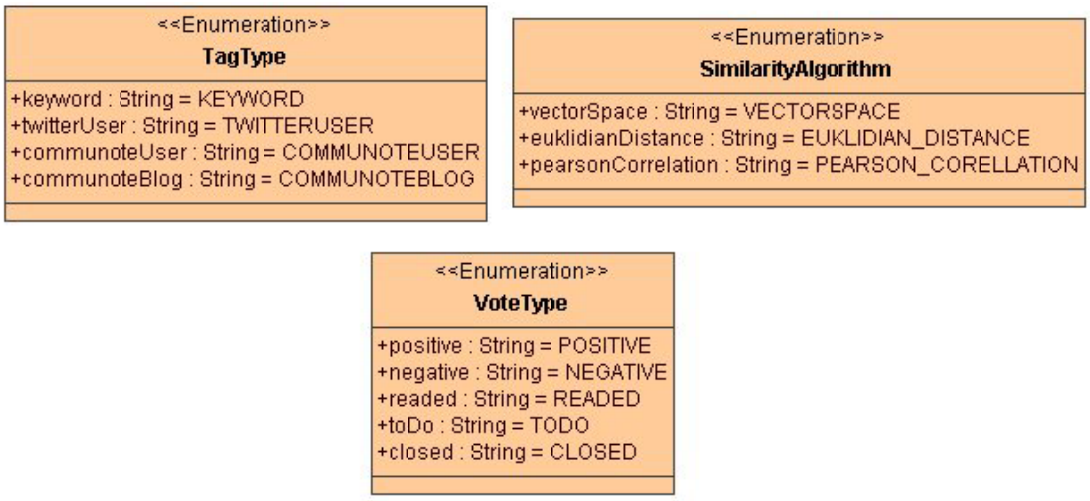

Abbildung 47: Klassendiagramm „Enummerations“

A.3 Sequenzdiagramme

Die folgende Abbildung 49 zeigt das Sequenzdiagramm der Funktion “FeedService.convertToInformationElements”, die innerhalb der Methode „FeedService.loadFeed()“ aufgerufen wird.

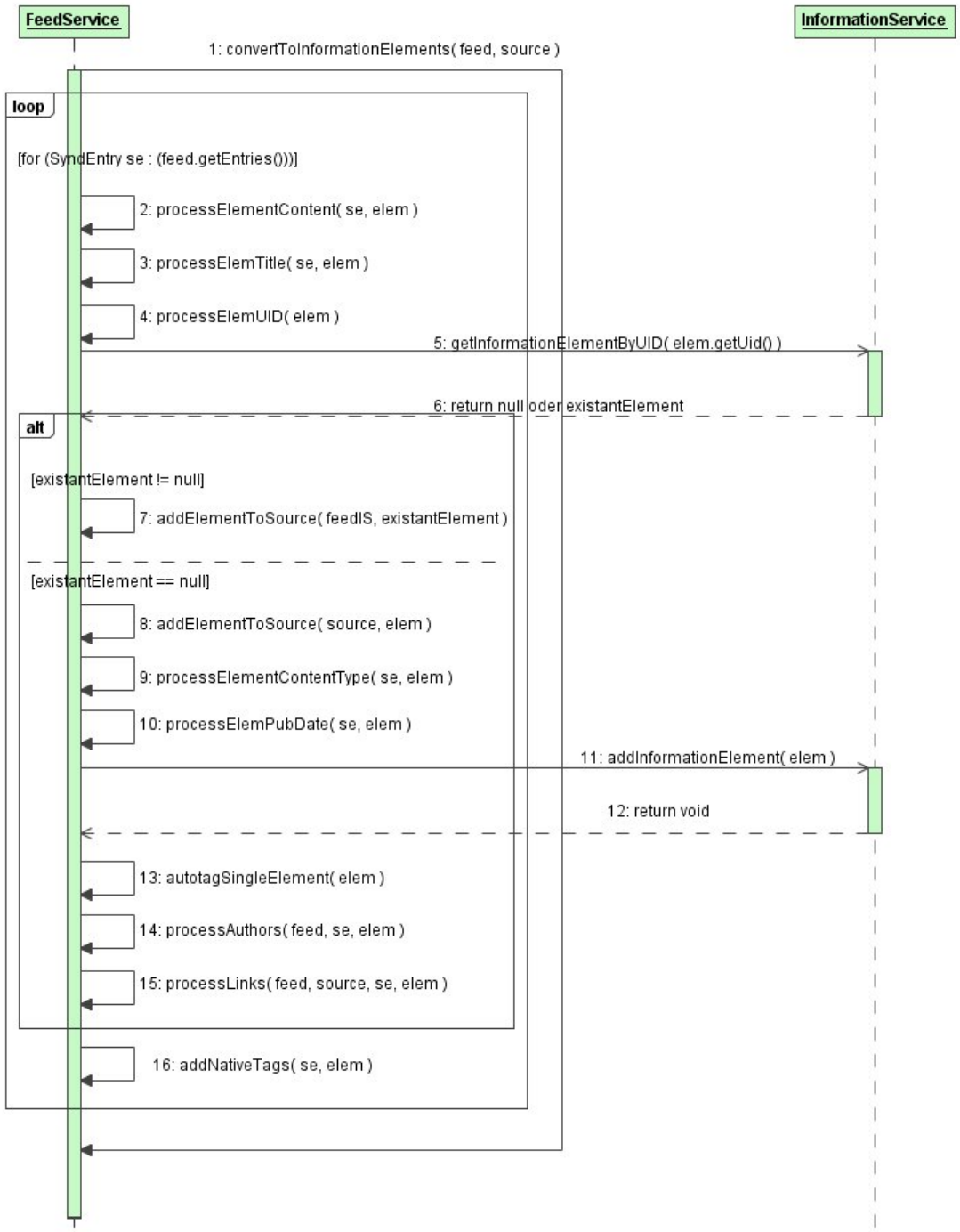

Abbildung 48: Sequenzdiagramm der Methode convertToInformationElements() der Klasse FeedService

A.4 Diagramme der Evaluierung

Nachfolgend werden die Diagramme der Evaluierung der Leistungsfähigkeit und Skalierbarkeit dargestellt, um diese Ergebnisse zu visualisieren. Dabei ist auf der Y-Achse die Zeit in Millisekunden dargestellt.

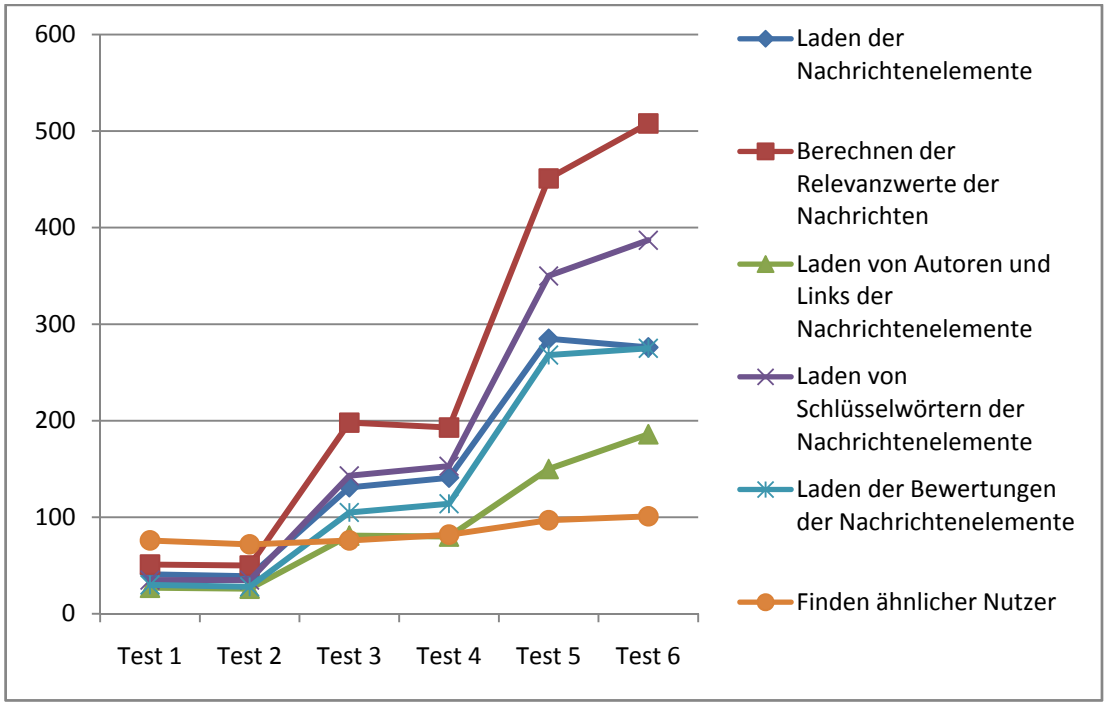

Abbildung 49: Ergebnisse der Evaluierung der Leistungsfähigkeit und Skalierbarkeit (1)

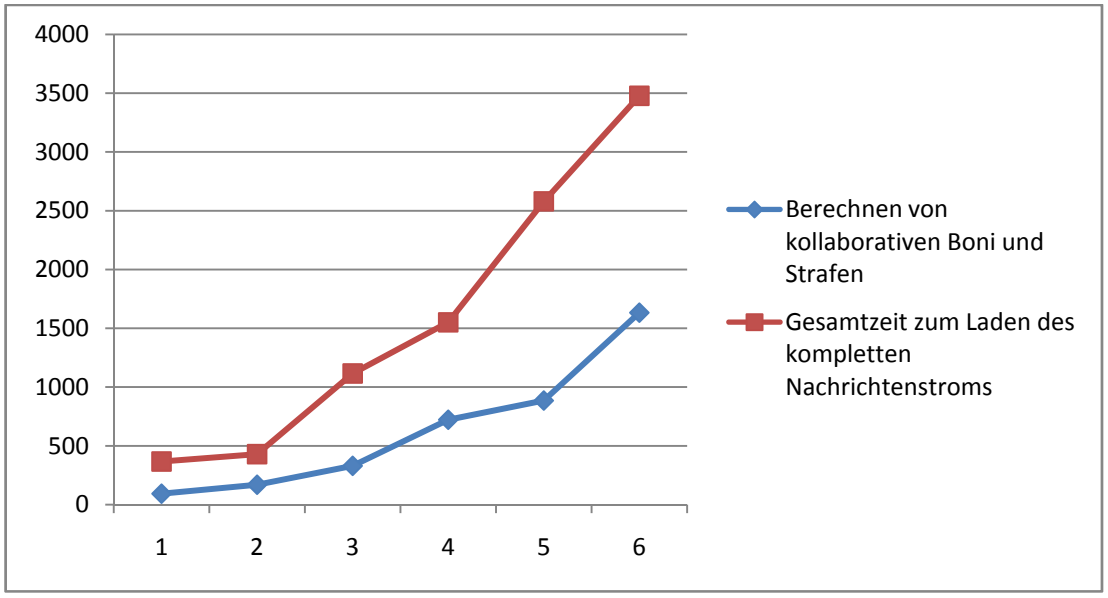

Abbildung 50: Ergebnisse der Evaluierung der Leistungsfähigkeit und Skalierbarkeit (2)

A.5 Komplettübersicht der Architektur

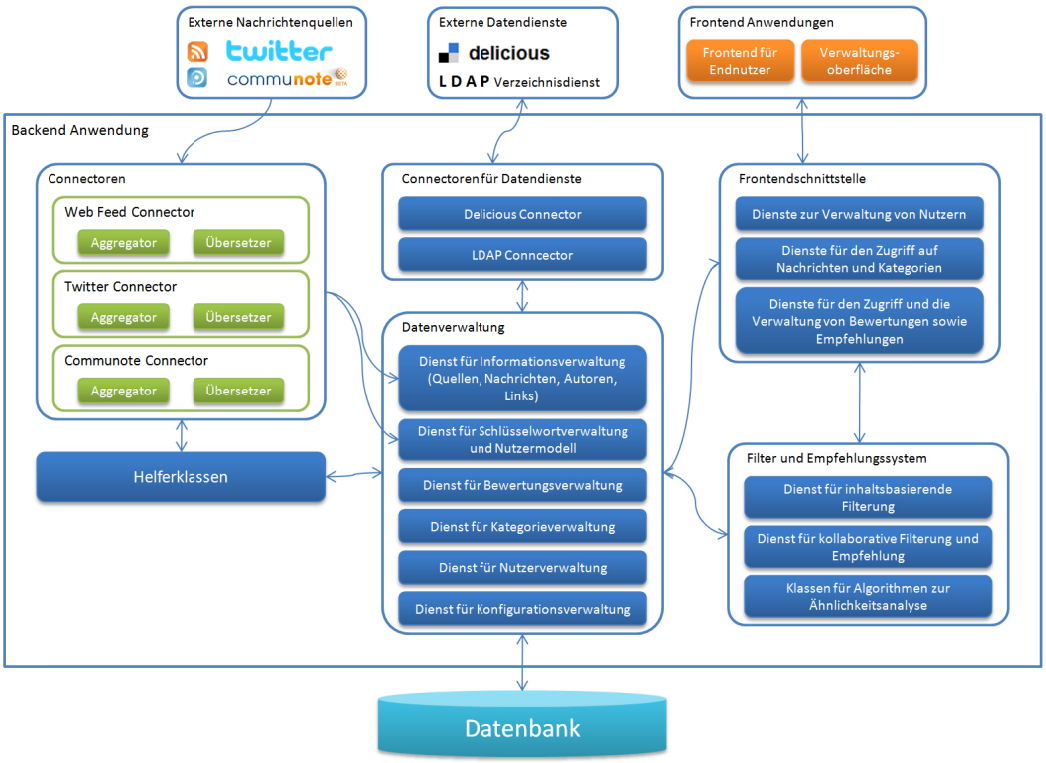

Abbildung 51: Komplettübersicht der Architektur

Abbildungsverzeichnis

Abbildung 1: Personalisierte und unpersonalisierte Webangebote nach [Klossek+03] 4
Abbildung 2: Schritte des Personalisierungsprozesses 5
Abbildung 3: Verfahren zur Personalisierung 6
Abbildung 4: Beispiel einer Stereotypenhierarchie [Meißner+07] 9
Abbildung 5: Klassifizierung von Ressourcen anhand von gewichteten Schlüsselwörtern [Lehm+04] 11
Abbildung 6: Beispiel Vektorraum-Modell [Lehm+04] 12
Abbildung 7: Datenmatrix und Ähnlichkeitstabelle [Lehm+04] 14
Abbildung 8: Isolation der durch zwei Nutzer bewerteten Ressourcen [Sarwar] 15
Abbildung 9: Wachstum „Twitter“-Nutzerzahlen April-Mai 2007 [Java+07] 19
Abbildung 10: Wachstum „Twitter“-Posts April-Mai 2007 [Java+07] 20
Abbildung 11: Screenshot Communote [Hauß+08] 22
Abbildung 12: Aufbau eines XMPP-Netzwerkes [XMPP] 23
Abbildung 13: Bereitstellung und Übertragung von Feeds 25
Abbildung 14: Yahoo Pipes Arbeitsfläche 29
Abbildung 15: Konzept von Microblogging-Aggregatoren 31
Abbildung 16: Informationsfluss am Beispiel von Feeds im Unternehmen [YOUNG+07] 32
Abbildung 17: Attensa Feedserver [ATTENSA] 33
Abbildung 18: Social Graph [Newsgator] 35
Abbildung 19: Anwendungsfalldiagramm Nutzer / Frontend 38
Abbildung 20: Anwendungsfalldiagramm Backend / Administrator 39
Abbildung 21: Übersicht des Umgebungsmodells 49
Abbildung 22: Top-Level Architektur 50
Abbildung 23: Auszug des Klassendiagrammes - Internes Nachrichtenformat 53
Abbildung 24: Auszug des Klassendiagrammes – Nutzer-Entitäten 54
Abbildung 25: Sequenzdiagramm FeedService – Funktion loadFeed() 62
Abbildung 26: Einordnung der Frontend Schnittstelle 65
Abbildung 27: AndroMDA 3-Schichten-Architektur [AndroMDA] 68
Abbildung 28: Klassendiagramm „UserAccess“ und Auszug aus Klassendiagramm „API Value Objects“ 69

Abbildung 29: Auszug Klassendiagramm „InformationAccess“ und „API Value Objects“ 70
Abbildung 30: Klassendiagramm „RecommendationAccess“ 70
Abbildung 31: Klassendiagramm Entitäten (1) 92
Abbildung 32: Klassendiagramm Entitäten (2) 93
Abbildung 33: Klassendiagramm Entitäten (3) 94
Abbildung 34: Klassendiagramm Entitäten (4) 95
Abbildung 35: Klassendiagramm Entitäten (5) 95
Abbildung 36: Klassendiagramm „Servicelayer“ (1) 96
Abbildung 37: Klassendiagramm „Servicelayer“ (2) 97
Abbildung 38: Klassendiagramm „Servicelayer“ (3) 97
Abbildung 39: Klassendiagramm Dienste für externe Zugriffe 98
Abbildung 40: Klassendiagramm Filterengine 98
Abbildung 41: Klassendiagramm „API Service Classes“ (1) 99
Abbildung 42: Klassendiagramm „API Service Classes“ (2) 100
Abbildung 43: Klassendiagramm API Value Objects 101
Abbildung 44: Klassendiagramm Anwendungskonfiguration 102
Abbildung 45: Klassendiagramm Beziehung User – Delicious Konfiguration 102
Abbildung 46: Klassendiagramm Sicherheitskomponente 103
Abbildung 47: Klassendiagramm „Enummerations“ 103
Abbildung 48: Sequenzdiagramm der Methode convertToInformationElements() der Klasse FeedService 104
Abbildung 49: Ergebnisse der Evaluierung der Leistungsfähigkeit und Skalierbarkeit (1) 105
Abbildung 50: Ergebnisse der Evaluierung der Leistungsfähigkeit und Skalierbarkeit (2) 105
Abbildung 51: Komplettübersicht der Architektur 106

Tabellenverzeichnis

Tabelle 1: Filtermechanismen Vor- und Nachteile [Meißner+07] 16
Tabelle 2: „Twitter“ Nutzerzahlen je Kontinent [Java+07] 20
Tabelle 3: RSS-Standards 25
Tabelle 4: Vergleich RSS 2.0 und Atom [Schill+08] [MEIERT] 26
Tabelle 5: Auszug der Übersicht von [MEIERT] 27
Tabelle 6: Nutzergruppenanalyse 37
Tabelle 7: Beispieltabelle Nutzermodell 51
Tabelle 8: Schwächen des Vektorraum-Modells 56
Tabelle 9: Beispieltabelle Bewertung von Nachrichten 57
Tabelle 10: Typen von Schlüsselwörtern 66
Tabelle 11: Evaluierungskonfigurationen der Infrastruktur 75
Tabelle 12: Nutzerprofile der Evaluierungstests 76
Tabelle 13: Erwartungswerte der Ähnlichkeitsanalyse 76
Tabelle 14: Rahmenbedingungen der Leistungsfähigkeitsevaluierung 77
Tabelle 15: Ergebnisse der Evaluation der Anwendungskonfigurationen 82
Tabelle 16: Ergebnisse des Algorithmenvergleichs 83
Tabelle 17: Ergebnisse der Evaluierung der Skalierbarkeit und Leistungsfähigkeit 84

Literaturverzeichnis

[AIDERSS] *http://www.aiderss.com*

[ANDROMDA] *http://galaxy.andromda.org/index.php?option=com_content&task=view&id=108&Itemid=89*

[ATOM] *http://www.atomenabled.org/developers/syndication/atom-format-spec.php*

[ATTENSA] *http://www.attensa.com*

[AUER+07] *Sören Auer, C. Bizer, G. Kobilarov, Jens Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A Nucleus for a Web of Open Data. 6th International Semantic Web Conference, Busan, Korea, 2007.*

[AßMANN+06] *Prof. Dr. U. Aßmann, Vorlesung Softwaretechnologie 2 SS2006, Technische Universität Dresden, Institut für Software- und Multimediatechnik*

[BALZERT+99] *Heide Balzert, Lehrbuch der Objektmodellierung, 1999*

[BONVANIE] *http://incisive.miramedia.net/online07/files/conferencing/9/SprulesBe01-Microsoft%20PowerPoint%20-%20A%20Bonvanie.ppt-00001.pdf*

[BREESE+98] *David Heckerman John S. Breese and Carl Kadie. Empirical analysis of redictive algorithms for collaborative filtering. In Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence, pages 43–52, July 1998.*

[CANBOLAT+09] *Serkan Canbolat, Web-basierte Visualisierung personalisierter Informationen aus semi-strukturierten Informationsquellen - Vergleichende Analyse. Konzeption und protoypische Realisierung, TU Dresden, 2009*

[COMMUNOTE] *http://www.communote.com*

[DELI-API] *http://sourceforge.net/projects/delicious-java/*

[DELICIOUS] *http://delicious.com/*

[DIGG] *http://digg.com/*

[EUGSTER+03] *Patrick Eugster, Pascal Felber, Rachid Guerraoui, and Anne-Marie Kermarrec, The Many Faces of Publish/Subscribe. ACM Computing Surveys, Number. 2, Volume 35, June 2003, pages 114-131, http://www.caip.rutgers.edu/~virajb/readinglist/facespublishsubscribe.pdf*

[FEEDHUB] *http://www.feedhub.com*

[FEEDS2.0] *http://www.feeds2.com*

[GEONAMES] *http://geonames.org*

[GOOGLE-01] *http://www.google.de/intl/de/help/features.html#spell*

[GOOGLE-02] *http://www.google.de/*

[HARVARD-LAW] *http://cyber.law.harvard.edu/rss/rssVersionHistory.html*

[HAUß+08] *Ilja Hauß, Enterprise Microblogging Präsentation, Barcamp Stuttgart, 27.09.2008*

[HENZE+06] *Prof. Dr. N. Henze, Personalisierung und Benutzermodellierung, Vorlesung an der Universität Hannover im Sommer-Semester 2006*

[HIBERNATE] *http://www.hibernate.org/*

[IETF] *http://tools.ietf.org/html/rfc4287*

[INFOWORLD] *http://www.infoworld.com/article/06/11/17/47TCrss_1.html*

[JAVA+07] *Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng, Why We Twitter: Understanding Microblogging Usage and Communities, Procedings of the Joint 9th WEBKDD and 1st SNA-KDD Workshop 2007, August 2007*

[JSF] *http://java.sun.com/javaee/javaserverfaces/*

[KEMPER+06] *Hans-Georg Kemper, Walid Mehanna, Carsten Unger, Business intelligence Grundlagen und praktische Anwendungen, 2. Auflage, 2006, Seite 136*

[KLOSSEK+03] *Martin Klossek, Optimierung der Personalisierung im Internet durch Kollaboratives Filtern, Johann Wolfgang Goethe-Universität Frankfurt am Main, 27.November 2003*

[KOBSA+04] *Alfred Kobsa, Adaptive Verfahren –Benutzermodellierung, University of California, Irvine, 2004*

[LACONICA] *http://laconi.ca/trac/wiki/ListOfServers*

[LEHM+04] *Anja Lehmann, Recommender Systems, TU Dresden, Hauptseminar Multimediatechnik, 2004*

[MEIERT] *http://meiert.com/de/publications/translations/intertwingly.net/rss-2.0-and-atom-1.0/*

[MEIßNER+07] *Prof.Dr-Ing. Klaus Meißner, Intelligente Multimediale User Interfaces, Vorlesung im WS06/07*

[MELVILLE+02] *Prem Melville, Raymond J. Mooney und Ramadass Nagarajan, Content-Boosted Collaborative Filtering for Improved Recommendations, Department of Computer Sciences University of Texas, Juli 2002, http://www.cs.utexas.edu/users/ml/papers/cbcf-aaai-02.pdf*

[MYFACES] *http://myfaces.apache.org/*

[NEWSGATOR] *http://www.newsgator.com*

[OPENCALIAS] *http://www.opencalais.com*

[PASSANT+08] *Alexandre Passant, Tuukka Hastrup, Uldis Bojars, John Breslin Microblogging: A Semantic and Distributed Approach, LaLIC, Université Paris-Sorbonne*

[RAHM+02] *Erhard Rahm, Web Usage Mining, Universität Leipzig http://dbs.uni-leipzig.de/files/abstr/wum-DB-Spektrum_02-02.pdf*

[RAMOS] *Juan Ramos, Using TF-IDF to Determine Word Relevance in Document Queries, Department of Computer Science, Rutgers University http://www.cs.rutgers.edu/~mlittman/courses/ml03/iCML03/papers/ramos.pdf*

[ROME] *https://rome.dev.java.net/*

[RSS-SPECIFICATION] *http://www.rss-specifications.com/history-rss.htm*

[SARWAR] *Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, ItemBased Collaborative Filtering Recommendation Algorithms, GroupLens Research Group/Army HPC Research Center Department of Computer Science and Engineering University of Minnesota, Minneapolis, MN 55455*

[SAXXESS+08] *http://www.saxxess.com/news/archiv/2008/sachsen/081208-communardo-communote-blog-microblogging-software.htm*

[SCHILL+06] *Vorlesung Bürpkommunikation SS 2006, Vorlesung 1 – Einführung, Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill*

[SCHILL+08] *Vorlesung Internet an Webapplications SS 2008, Vorlesung 7 – Subscription Services, Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill*

[SCHILL02+08] *Vorlesung Internet an Webapplications SS 2008, Vorlesung 10 – Chat Systems, Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill*

[SCRIPTING-COM] *http://www.scripting.com/davenet/1997/12/15/scriptingNewsInXML.html*

[SEGARAN+07] *Toby Segaran, Programming Collective Intelligence: Building Smart Web 2.0 Applications, 2007*

[SMITH+09] *Richard W. Smith und Gregory Priebe, „Twitter ‚Laconi.ca and Tumblr", Harford Community College, 2009 http://www.marylanddla.org/2009/presentations/T5_Smith.pdf*

[TECHNORATI] *http://technorati.com/videos/youtube.com%2Fwatch%3Fv%3DHGz8Csoh9cI*

[TINYURL] *http://tinyurl.com/*

[TREVOR] *Jonathan Trevor, http://www.slideshare.net/deimos/jonathan-trevor-*

yahoo-pipes/

[TUZHILIN] *Tuzhilin A Adomavicius G. Recommendation technologies: Survey of current methods and possible extensions*

[TWITTER] *http://twitter.com/help/aboutus*

[TWITTER4J] *http://yusuke.homeip.net/twitter4j/en/index.html*

[TWITTER-API] *http://apiwiki.twitter.com/*

[UML] *http://www.uml.org/*

[UR1] *http://ur1.ca/*

[WIKI-01] *http://en.wikipedia.org/wiki/RSS_(file_format)*

[WIKI-02] *http://de.wikipedia.org/wiki/Modellgetriebene_Softwareentwicklung*

[WOLLEN+04] *Frank Wollenweber, Kollaborative Nutzung des World Wide Webs, Universität Hamburg, Januar 2004*

[XML-COM] *http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html*

[XML-RSS-DE] *http://xml-rss.de/rss/die-geschichte-von-rss.htm*

[XMPP] *http://xmpp.org/*

[YAHOOPIPES] *http://pipes.yahoo.com*

[YOUNG+07] *G. Oliver Young, Bradford J. Holmes, Erica Driver, Charlene Li, Heidi Lo, and April Lawson, Enterprise RSS Tackles Information Worker Overload, 8.Mai 2007*

Alle Weblinks wurden zuletzt am 23.04.2009 auf Gültigkeit überprüft.

Abkürzungsverzeichnis

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface - Programmierschnittstelle
DVD	Digital Versatile Disc
HTTP	Hypertext Transfer Protocol
IT	Informationstechnik
JID	Jabber Identifier
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
MDA	Model driven architecture
MDSD	Model-Driven Software Development
NNTP	Network News Transfer Protocol
OPML	Outline Processor Markup Language
PDA	Personal Digital Assistent
PDF	Portable Document Format
REST	Representational State Transfer
SMS	Short Message Service
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTF-8	8-bit Unicode Transformation Format
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol