GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Abschlussbericht

OnToBau - Ontologie-basierte Prozessunterstützung im Bauwesen

FKZ 1731X09

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 1731X09 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

Projektlaufzeit

01.06.2009 - 31.08.2012

Projektleitung

Prof. Dr. Norbert Kuhn

Autoren

Norbert Kuhn

Stefan Richter

Markus Schwinn

Herausgeber

Institut für Softwaresysteme in Wirtschaft, Umwelt und Verwaltung

Hochschule Trier, Standort Birkenfeld

Umwelt-Campus Birkenfeld

Institut für Softwaresysteme in Wirtschaft, Umwelt und Verwaltung

Postfach 13 80

D-55761 Birkenfeld

Birkenfeld 2013

Inhaltsverzeichnis

ABBILDUNGSVERZEICHNIS

I KURZDARSTELLUNG DES VORHABENS

1 Einleitung

Die wachsende Bedeutung des Computers in den 70er Jahren und die damit verbundene Möglichkeit Informationen in digitaler Form zu verbreiten, wird als Umbruch in das heutige Informationszeitalter angesehen. Wie bei keiner anderen technologischen Revolution in der Geschichte der Menschheit breiteten sich die neuen Informationstechnologien binnen zwei Jahrzehnten über den gesamten Planeten aus [Cas01]. Erstmals war es möglich Informationen auf einfache Art und Weise zu erstellen, zu vervielfältigen und zu archivieren.

Anfang der 90er Jahre wurde ein weiterer Grundstein gelegt, der sich einige Jahre später zum Massenkommunikationsmedium der Neuzeit entwickeln sollte - das Internet. Die Reproduktion von Wissen, was Johannes Gutenberg Mitte des 15. Jahrhunderts durch die Erfindung des Buchdrucks ermöglichte, ist vergleichbar mit den Ideen von Tim-Berners Lee gut 450 Jahre später. Sowohl die Erfindung des Buchdrucks als auch die Entwicklung des Internets werden als Meilensteine in der Geschichte angesehen [Bod03].

Der weltweite Einzug des Internets in die privaten Haushalte der Menschen war ausschlaggebend für die inzwischen nicht mehr überschaubare Informationsflut, die alltäglich das World Wide Web wachsen lasst. Laut einer Studie [LV03] der Universität Berkeley aus dem Jahr 2003 betragt das Informationswachstum 800 Megabyte pro Mensch im Jahr. Bei einer damaligen Population von 6,3 Milliarden Menschen, entspricht dies einer jährlichen Zunahme an Informationen von 5 Exabyte. 92 % davon liegen in elektronischer Form vor. Im Vergleich dazu betragt der Zuwachs an Informationen in papiergebundener Form lediglich 0,0016 Exabyte.

Die Studie schätzte weiterhin die Gesamtgröße an Informationen, die im Internet verfügbar sind, auf ca. 92000 Terabyte. Von diesen Informationen seien aber lediglich ca. 170 Terabyte im so genannten *Surface Web* erreichbar. Das Surface Web bezeichnet den Teil des World Wide Webs, der über Suchmaschinen auffindbar ist. Der größte Informationsanteil befindet sich demnach u.a. in Datenbanken und Webseiten, die erst bei speziellen Anfragen dynamisch generiert werden (*Deep Web*) [Ber01]. Mitte 2008 veröffentlichte Google in ihrem offiziellem Blog, dass derzeit über eine Trillionen Webseiten von ihren Crawlern untersucht werden [AH08b]. Ein Studie aus dem Jahr 2008, durchgeführt am Institute of Computing Technology in China, untersuchte das Wachstum des World Wide Webs. Sie fanden Analogien zu dem von Moore 1965 aufgestellten Gesetz, dass sich die Anzahl an Transistoren in einem Hauptprozessor alle 18 Monate verdoppelt. Laut ihren empirischen Ergebnissen verdoppelt sich die Größe des World Wide Webs alle 5,35 Jahre und folgt einem ähnlichen exponentiellen Wachstumsmuster, welches Moore für sein Gesetz herangezogen hat [ZZY+08].

Den größten Informationsfluss im Internet stellen jedoch E-Mails dar, die in 2003 einen Anteil von 440000 Terabyte an Daten ausmachten [LV03]. Eine neue Studie der Radicati Group [RK09] besagt, dass im Jahr 2009 ungefähr 247 Milliarden E-Mails am Tag verschickt werden, deren Anzahl sich bis zum Jahr 2013 verdoppeln wird. Jedoch sind ca. 81 % der versandten E-Mails als Spam anzusehen. Eine Zahl die verdeutlicht, welchen Stellenwert das Filtern von Informationen alleine im elektronischen Postverkehr einnimmt. Dies wird ebenfalls von der Studie belegt, deren Schatzungen davon ausgehen, dass Unternehmen mit

1000 und mehr Mitarbeitern in den nächsten Jahren bis zu 1,8 Millionen Dollar pro Jahr in die Verarbeitung von E-Mails und Spam investieren müssen.

Jedoch nicht nur die Datenflut aus dem wachsenden E-Mailverkehr wird die Unternehmen in naher Zukunft vor neue Herausforderungen stellen. Eine Untersuchung der Gartner, Inc. von 2002 mit rund 300 Unternehmen ergab, dass 90 % der Befragten an einem Überfluss an Informationen leiden. Um diesen Massen an Informationen aus Internet, Intranet etc. entgegenzuwirken, mussten die Unternehmen in den nächsten Jahren 30 Milliarden Dollar für Informationsmanagement-Systeme ausgeben [Goa02]. Eine ähnliche Untersuchung der Basex Inc. ergab sechs Jahre später, dass die Folgen der Informationsüberflutung die U.S. Wirtschaft ca. 900 Milliarden Dollar pro Jahr kostet, hervorgerufen durch verringerte Produktivität der Wissensarbeiter, die im Schnitt 25 % ihrer Arbeitszeit mit dem Suchen nach Informationen verbringen [Spi08].

Um diese Überflutung an Informationen in Zukunft adäquat beherrschen zu können, befassen sich aktuelle Forschungen mit der Suche nach Technologien für den Einsatz in Informationsmanagement-Systemen. Ein wesentlicher Bestandteil dieser Forschungen beschäftigt sich mit der Informationsextraktion (IE). Das Ziel dieser Forschungsrichtung ist die Konzeptionierung und Realisierung von Verfahren, die aus freien Texten Informationen extrahieren und in eine strukturierte Form überführen können [Neu01].

Der Einsatz von IT-Technologien zur Unterstützung dieser Wissensnetzwerke kann entscheidende Verbesserungen für das Unternehmen und die Mitarbeiter bringen. Die Visualisierung der Netzwerke bietet die Möglichkeit kollektives Wissen anschaulich darzustellen und Zusammenhänge aufzuzeigen, die in klassischen Wissensabbildungen (tabellarisch, hierarchisch etc.) nicht erkannt werden, da die Querbezüge zwischen den einzelnen Informationsbeständen fehlen [Bai08]. Mittels entsprechender Abfragesprachen können gezielt Anfragen an das Wissensnetzwerk gestellt werden, die Wissen anhand ihrer Eigenschaften (z.B. *alle weiblichen Mitarbeiter mit Doktortitel*) oder dem vorliegen bestimmter Relationen (z.B. *alle Personen, die im Bauprojekt "Frankfurt 22“ beteiligt sind*) selektieren (vgl.[Rei10]). Während in größeren Unternehmen der Einzug von Wissensmanagementlösungen bereits weit verbreitet ist, wird dieses Thema in kleineren und mittleren Unternehmen jedoch immer noch mit Skepsis betrachtet [WA04].

Damit Wissen innerhalb eines Unternehmens mittels IKT gemanagt werden kann, muss zuvor Klarheit darüber bestehen, welches Wissen das Unternehmen bzw. verschiedene Prozesse benötigen und auf welche Art und Weise dieses aufgebaut und im Unternehmen hinterlegt ist. Diese Wissensstrategie kann sich dabei auf die Verarbeitung von internem oder externen Wissen fokussieren. Je nach Fokussierung und Kombinationsmöglichkeiten werden in der Literatur verschiedene Wissensstrategien identifiziert, auf die an dieser Stelle nicht weiter eingegangen wird (vgl. [OdP02], [AB06]).

Besonderheiten im Umfeld kleiner, mittelständischer Unternehmen ergeben sich im Bereich der Führungs- und Organisationsstrukturen (vgl. [WA04], [Fis06]): Über 90% der klein- und mittelständigen Unternehmen sind familiengeführt (vgl. [SW08]), womit die Unternehmensleitung in der Regel durch einen einzigen geschäftsführenden Unternehmer vertreten wird. Es besteht häufig eine flache Hierarchie, mit einem geringen Anteil an Leitungs- und Führungskräften. Entscheidungsprozesse sind somit weniger strukturiert und werden überwiegend direkt von der Unternehmensleitung getroffen, eine Delegation von Führungsentscheidungen findet nur in geringem Umfang statt. Die Bereitschaft sich mit

komplexen und spezialisierten Instrumenten auseinander zu setzen ist aufgrund Zeitmangel und fehlenden finanziellen und personellen Ressourcen kaum vorhanden.

Diese Besonderheiten stellen an die Einführung eines Wissensmanagementsystem in KMU´s zusätzliche Anforderungen, denen im Forschungsprojekt OnToBau eine besondere Aufmerksamkeit gewidmet wurde. Im folgenden Abschlussbericht wird das Forschungsprojekt und dessen Ergebnisse präsentiert.

2 Aufgabenstellung

Das Projektvorhaben war, Voraussetzungen zu schaffen, die es kleinen und mittelständigen Unternehmen im Bauwesen ermöglichen, fachspezifisches Wissen innerhalb ihres Unternehmens effektiv zu erfassen, zu archivieren und für unterschiedliche Prozesse unterstützend zu nutzen. Das zu entwickelnde Wissensarchiv soll den Mitarbeitern vor allem in wissensintensiven Prozessen (wie z.B. der Angebotserstellung) proaktiv Informationen zur Verfügung stellen. Zudem sollen Mitarbeiter die Möglichkeit haben, Wissen mit Hilfe verschiedener Suchanfragen zu recherchieren.

Vergleichbar mit dem Forschungsprojekt DYONIPOS, ist auch in OnToBau vorgesehen einen intelligenten Agenten zu entwickeln, der die Arbeit des Mitarbeiters verfolgt und bestimmte Verhaltensmuster bzw. Arbeitsschritte erlernt, mit Hilfe dieser Erkenntnisse ein persönliches Profil des Benutzers erstellt und ihm gegebenenfalls geeignete Informationsquellen automatisch zur Verfügung stellt.

Bezugnehmend auf die zuvor angesprochenen besonderen Gegebenheiten in KMU´s wurden in der Projektvorhabensbeschreibung an die Einführung eines derartigen Systems folgende Anforderungen gesetzt:

1. die elektronische Erfassung von analogen Dokumenten darf keinen (bzw. nur minimalen) Aufwand erfordern,
2. es dürfen keine laufenden Kosten entstehen, insbesondere keine zusätzlichen Personalkosten,
3. die Extraktion und Verknüpfung des Wissens aus den verschiedenen Informationsquellen soll automatisch erfolgen,
4. der Zugriff auf das extrahierten Wissen muss jederzeit möglich sein,
5. das extrahierte Wissen muss auf geeignete Weise mit den üblichen Unternehmensprozessen verknüpft werden.

3 Planung und Ablauf

In den folgenden Unterkapiteln werden die im Forschungsvorhaben geplanten Arbeitspakete vorgestellt. Die inhaltliche Bearbeitung und Erkenntnisse werden im Kapitel II ausführlich beschrieben.

3.1 *Arbeitspakete*

Folgende Arbeitspakte waren in dem Forschungsvorhaben vorgesehen:

<u>Arbeitspaket 1: Grundlagen</u>

1.1 Analyse Anwendungsdomäne (Absprachen/Interviews mit Nutzern und Anwendern)

Mit den Anwendern aus dem Baugewerbe werden strukturierte Interviews geführt. Dabei werden relevante Prozesse und dazu gehörige Dokumente identifiziert.

1.2 Auswahl der Referenzprozesse und Wissenskontexte

Gemeinsam mit den Partnern werden die Prozesse ausgewählt, die im Projekt exemplarisch bearbeitet werden sollen. Dabei sollen die Prozesse eine gewisse Allgemeingültigkeit, sowohl für die Anwendungspartner als auch allgemein für die Baubranche besitzen. Für die ausgewählten Prozesse werden die einzelnen Arbeitsschritte zusammen mit dem jeweils notwendigen Wissen und der Wissensquellen beschrieben. Dazu wird spezifiziert, welche Berechnungsschritte an den einzelnen Prozessschritten einzuplanen sind (z.B. Verifikation von Auftragsdaten, Rechnungspositionen, etc.)

1.3 Interaktionsmodell für die Benutzerschnittstelle

Die Funktionsweise und der Funktionsumfang der Benutzerschnittstelle werden beschrieben. Dies umfasst die Definition, wie und welche Vorschläge z. B. der intelligente Benutzerinteragent dem Benutzer macht. Es beschreibt aber auch, wie z.B. neue Information in das System eingefügt wird (automatisch oder nur nach Verifikation durch einen Benutzer oder einen Administrator), oder wer wie auf welche Information zugreifen kann.

Arbeitspaket 2: Aufbau der Ontologie

2.1 Untersuchung Beschreibungsformen für Ontologien

Verschiedene Beschreibungsformalismen für Ontologien sollen untersucht und evaluiert werden. Ein Kriterium dabei ist, wie effizient sowohl domänenbezogenes, statisches Wissen als auch prozessbezogenes, dynamisches Wissen dargestellt und bei der Prozesssteuerung genutzt werden kann. Darüber hinaus sind weitere Kriterien zu berücksichtigen, wie beispielsweise die Verständlichkeit für den Nutzer und Änderbarkeit durch ihn. In diesem Zusammenhang wird auch mit den Partnern untersucht werden, inwieweit eine grafische Repräsentationsmöglichkeit wichtig oder nützlich ist.

2.2 Aufbau einer Ontologie Bauwesen

Nach der Auswahl eines Beschreibungsformalismus wird eine Ontologie aufgebaut, die den Anwendungsbereich mindestens in den notwendigen Bereichen für die ausgewählten Dokumente abdeckt. Es wird sowohl statisches als auch dynamisches Wissen und insbesondere prozessbezogenes Wissen in der Ontologie abgelegt. Die Ontologie wird mit den Anwendungspartnern verifiziert. Die Integration der bestehenden Ontologie der Mindox GmbH wird gewährleistet.

2.3 Integration von Klassifikations- und Retrievalverfahren

Die Ablage neuer Dokumente und der Zugriff auf vorhandenes Wissen soll weitgehend automatisiert werden. Dazu ist zu beschreiben, wie aus neuen Dokumenten die notwendige Information bestimmt werden soll, die Basis für eine Klassifikation ist. In Abhängigkeit verschiedener Dokumentklassen sollen verschiedene Klassifikationsverfahren ansprechbar sein. Für das Retrieval sind ebenfalls verschiedene Ähnlichkeitsmaße auswählbar und können somit den Dokumenten zugeordnet werden. Hier wird untersucht, welche Maße als

Voreinstellung oder Default für bestimmte Wissenskategorien sinnvoll sind. Ein wesentliches Ziel innerhalb dieser Teilaufgabe besteht in der Automatisierung des Klassifikations- und Retrievalansatzes durch automatisches Lernen der Parameter, die diese Verfahren bestimmen. Das Lernen ist notwendig, wenn die Ontologie beispielsweise durch den Benutzer verändert wurde oder durch Feedback des Benutzers die Ergebnisse als nicht mehr ausreichend eingestuft werden.

Arbeitspaket 3: Prozessmodellierung

3.1 Untersuchung Prozessmodellierungsformalismen

Verschiedene Beschreibungsformalismen für Prozesse sollen untersucht und evaluiert werden. Ein Bewertungskriterium ist die wissensbasierte Steuerbarkeit durch den Rückgriff auf ontologisches Wissen. Darüber hinaus ist auch hier die Nachvollziehbarkeit und Änderbarkeit durch den Benutzer ein Untersuchungsgegenstand.

3.2 Modellierung der Referenzprozesse

Nach der Auswahl des Prozessmodellierungsformalismus werden die Referenzprozesse modelliert. Die Interaktionsmöglichkeiten werden skizziert. Die Modellierung wird getestet, d.h. die Modelle werden in Zusammenarbeit mit den Anwendungspartnern verifiziert.

3.3 Integration Intelligenter Benutzeragent

Der Intelligente Benutzeragent wird in die Prozessmodellierung integriert. Dabei planen wir, den Agenten als eine Komponente zu realisieren, die sich dem Benutzer gegenüber in etwa so verhält, wie „Karl Klammer“ aus der Office Welt von Microsoft („Karl Klammer“ erkennt z.B. wenn der Officenutzer einen Brief schreiben will und schlägt ihm dann seine Hilfe vor).

Unsere Benutzerassistenzkomponente soll als eigener Thread neben der „normalen“ Sachbearbeitung im Hintergrund laufen und dabei notwendige Informationen sammeln, um die Intention des Nutzers zu erkennen und zu entscheiden, ob er ihm gegebenenfalls Unterstützung anbieten soll. Darüber hinaus soll es möglich sein, dass der Benutzer aktiv Hilfe anfordert, indem er z.B. eine Taste wie „F1“ drückt oder einen Menüpunkt „Hilfe“ auswählt. Dann soll sich der Hilfedialog auch möglichst automatisch auf den aktuellen Kontext einstellen.

Eine weitere Möglichkeit der Einbindung kann über Schnittstellen in den Prozessschritten erfolgen. In Prozessbeschreibungen ist es normalerweise möglich, Berechnungsfunktionen in Bearbeitungsschritte einzubinden. Der Benutzerassistent soll dafür auch ansprechbar und einplanbar sein.

Arbeitspaket 4: Implementierung

4.1 Aufbau der Unternehmenswissensbasis

Es wird ein Datenbankmodell entworfen, mit dem das notwendige Wissen gespeichert werden kann. Dieses Wissen umfasst u.a. die Dokumente in ihrer bildhaften Form, zusammen mit Attributen, auf die sowohl Retrievalverfahren zurückgreifen sollen als auch

die prozesssteuernden Verfahren. Darüber hinaus soll auch die Ontologie in der Datenbank abgelegt und darüber bearbeitet werden können. Zur Implementierung des Modells planen wir aus Kostengründen den Einsatz lizenzfreier Produkte ein, z.B. Datenbanken wie MySQL oder PostGreSQL.

4.2 Implementierung des Intelligenten Benutzeragenten

Im ersten Schritt werden die Referenzprozesse, die mit den Partnern bestimmt wurden implementiert. Dazu werden gegebenenfalls ebenfalls lizenzfreie Werkzeuge genutzt. Für die Ablage der Dokumente werden die Komponenten der Mindox GmbH integriert und angepasst, damit auch die prozessbezogene Information mit berücksichtigt werden kann. Damit der Benutzeragent Vorschläge für die Nutzung vorhandenen Wissens unterbreiten kann, sind entsprechende Retrievalverfahren zu integrieren und gegebenenfalls zu konfigurieren.

Der Benutzer soll durch Feedbackverfahren diese Verfahren steuern und verbessern können. Dazu werden geeignete Lernverfahren integriert. Die Entwicklung soll evolutionär erfolgen, damit die Mitarbeiter der Partner den jeweiligen Entwicklungsstand begleitend testen und bewerten können.

4.3 Systemtest

Das System wird analog zur geplanten Konfiguration in der Firma aufgebaut und getestet. Diese Tests erfolgen gemeinsam mit den Mitarbeitern der Partnerunternehmen. Dabei streben wir insgesamt ein evolutionäres Vorgehensmodell an, bei dem wir das bestehende System zur Erfassung schrittweise weiter ausbauen, dabei jeweils Tests durchführen und Anwenderfeedback einholen, was dann in die weitere Entwicklung einfließt.

Arbeitspaket 5: Installation Prototyp

5.1 Aufbau der Konfiguration im Unternehmen

In Absprache mit den Geschäftsführern und ggf. den EDV-Leitern wird der Prototyp vor Ort physikalisch aufgebaut. Dazu werden die Systemkonfigurationen festgelegt (beteiligte Server, Aufbau der Erfassungsstationen, etc.). Firmen- und firmengrößenspezifische Aspekte werden untersucht und berücksichtigt.

5.2 Einführung im Unternehmen

Die Projektmitarbeiter begleiten den Start der Prototypen in den Firmen. Sie sind regelmäßig vor Ort um Probleme beseitigen zu können bzw. um Problemsituationen zu dokumentieren.

5.3 Anwenderschulung

Parallel zur Einführung wird eine Anwenderschulung durchgeführt. Deren Aufbau wird mit den Fachabteilungsleitern und den EDV-Leitern abgestimmt. Die Schulungen finden begleitend zur Nutzung an den Arbeitsplätzen statt. Die Schulungsmaßnahmen sind bei diesem Vorhaben von besonderer Wichtigkeit für die Akzeptanz des entwickelten Systems, da dadurch eventuelle Defizite in der Nutzung von EDV kompensiert und mit der Einführung

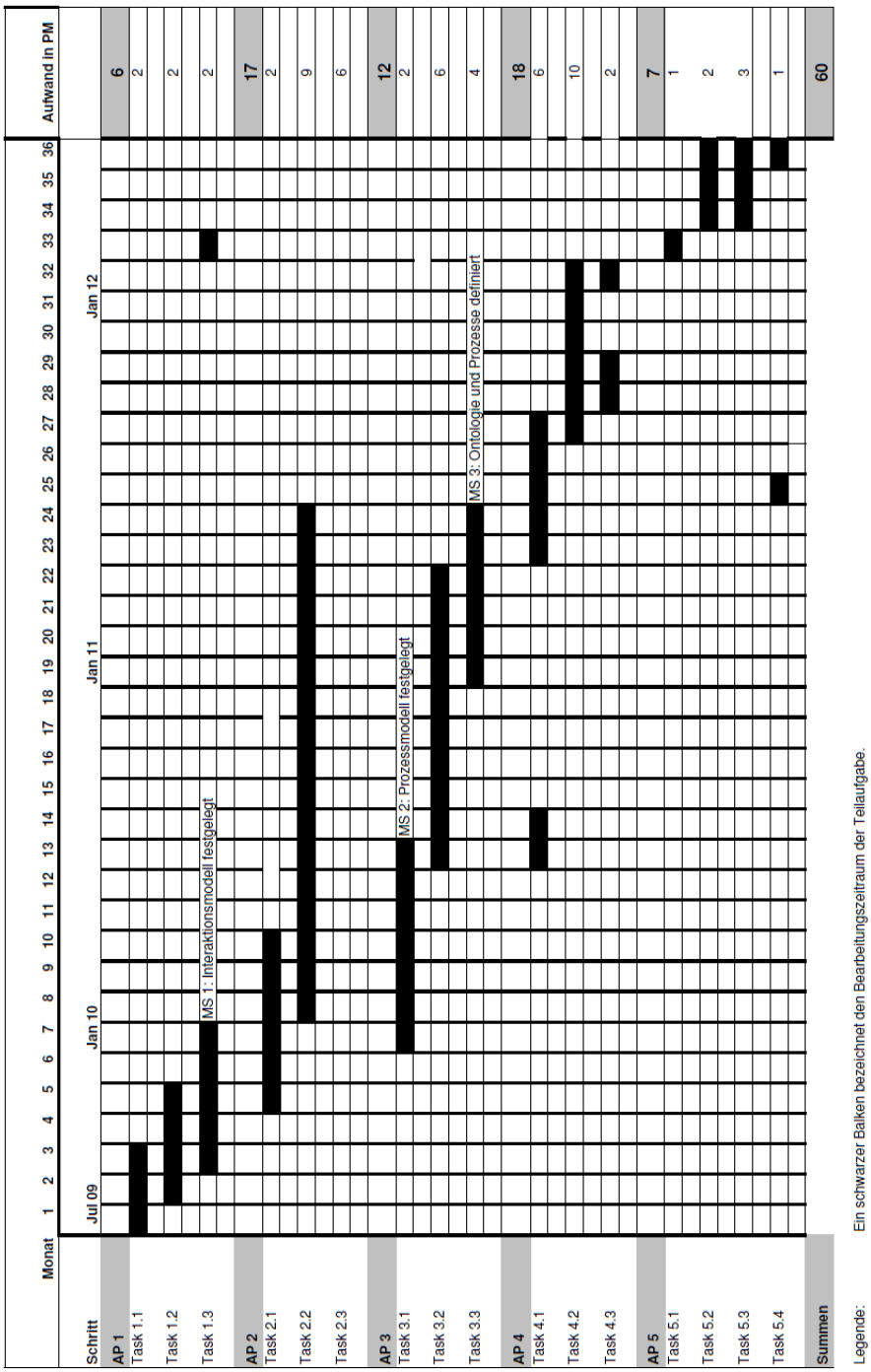
verbundene Ängste abgebaut werden können. Die hier gewonnenen Erkenntnisse sind für die weitere wirtschaftliche Verwertung ebenfalls von Belang.

5.4 Evaluation

Es finden Interviews bei den Projektpartnern zur Bewertung der Prototypen und dem damit verbundenen Nutzen statt. Ebenfalls bewertet werden der Projektverlauf und die Einzelergebnisse

3.2 *Zeitlicher Ablauf*

Der formale Projektbeginn war geplant für den 1. Juni 2009. Die Mitarbeiterstelle TV-L E12 konnte allerdings erst zum 1. September besetzt werden. Der Projektfortschritt geriet daher im Vergleich zum ursprünglichen Arbeitsplan etwas in Verzug. Die Erreichung des Meilensteins 1 verschob sich um etwa zwei Monate (Ende April anstatt wie ursprünglich geplant Ende Februar 2010). Das Arbeitspaket 1.3 „Interaktionsmodell für die Benutzerschnittstelle“ konnte deshalb nicht wie geplant im September 2009 begonnen werden, sondern erst im Dezember 2009. Die zweite Mitarbeiterstelle TV-L E11 wurde wie geplant zum 1. Juli 2010 besetzt. Aufgrund der verspäteten Besetzung geriet das Projekt im Hinblick auf den ursprünglichen Projektplan um etwa zwei bis drei Monate in Verzug. Zum Ausgleich wurde in 2011 die kostenneutrale Verlängerung der Projektlaufzeit bis Ende August 2012 beantragt und vom Projektträger genehmigt.



4 Ausgangslage, Stand von Wissenschaft und Technik

4.1 *Grundlagen*

Die Idee, Informationen aus Dokumenten durch linguistische Analysen in der Art zu reduzieren, dass man diese in strukturierter Form in einer Datenbank ablegen kann, ist bereits Ende der 50er publiziert worden [Har59]. In wissenschaftlichen Ausarbeitungen und Büchern wird immer wieder auf diese Pionierarbeit von Zellig Harris hingewiesen. Seine Forschungen auf dem Gebiet der Linguistik (*String Grammatiken*, *Baumadjunktions-*

Grammatiken, *Finite State Transducer*) sind noch heute grundlegender Bestandteil in vielen Bereichen der Informationsextraktion (IE) [Nev02].

Weltweites Interesse wurde der IE jedoch erst Ende der 80er Jahre durch die Einführung der *Message Understanding Conferences* (MUC) beigemessen [MUC91][MUC92] [MUC93][MUC95]. 1987 initiierte die amerikanische Verteidigungsbehörde *Defense Advanced Research Projects Agency* (DARPA) die erste MUC, mit dem Ziel IE-Systeme verschiedener Forscherteams gegeneinander antreten zu lassen. Die Systeme sollten zu einem vorgegebenem Themengebiet (z.B. Flottenoperationen in militärischen Meldungen) zuvor definierte Informationen extrahieren. Von 1987 bis 1997 wurden insgesamt sieben Forschungswettbewerbe (MUC-1 bis MUC-7), mit jeweils wechselnden Themengebieten (terroristische Aktivitäten, Führungswechsel in der Wirtschaft, Flugzeugabstürze etc.) durchgeführt. War es den Teilnehmern in der ersten MUC noch erlaubt, das Ausgabeformat der extrahierten Informationen selbst zu bestimmen, so wurde ab der zweiten MUC das Ausgabeformat in Form von zu füllenden Templates vorgegeben. Um eine Evaluation der verschiedenen Ergebnisse zu ermöglichen, wurde ein Maß für die Güte der extrahierten Informationen entwickelt. Die auf der MUC-2 hierfür eingeführten Maße *Recall* und *Precision*, werden auch heute noch zur Bewertung von IE-Systemen eingesetzt [GS96].

Sinnvolle Einsatzgebiete von IE-Systemen sind vielfältig. Ein Unternehmen ist zum Beispiel an den preislichen Entwicklungen der Produkte ihrer Konkurrenz interessiert und muss dazu hunderte von Webseiten kontinuierlich analysieren. Ein pharmazeutisches Unternehmen muss für die Entwicklung neuer Medikamente tausende von Artikeln heranziehen, um entsprechende Protein-Protein-Reaktionen vorhersagen zu können. Ein Verlag bietet zum Beispiel alle Gerichtsurteile zur Ansicht in seiner Online-Datenbank an. Dazu müssen Mitarbeiter täglich hunderte von Urteilen klassifizieren, um diese dann einem oder mehreren entsprechenden Rechtsgebieten zuteilen zu können. Der Geheimdienst verfolgt terroristische Aktivitäten weltweit. Dazu müssen hunderte von Nachrichten, E-Mails und Eilmeldungen bearbeitet werden.

All diese Beispiel haben einige Merkmale gemeinsam:

- es werden Informationen aus Dokumenten benötigt,
- diese Informationen liegen in unstrukturierter Form vor,
- die Anzahl der zu untersuchenden Dokumente ist groß.

Des Weiteren ist es für einen Mitarbeiter aufgrund der hohen Anzahl an Dokumenten nahezu unmöglich, die Informationen manuell zu extrahieren. Ein Computer kann diese Aufgabe jedoch auch nicht ohne Weiteres übernehmen, da die Informationen innerhalb der Dokumente nicht strukturiert repräsentiert sind. Genau an dieser Stelle setzt die IE an, indem algorithmische Lösungen für die oben angesprochenen Problemstellungen gesucht werden.

In folgenden wird die Architektur eines IE-Systems beschrieben und welche Aufgaben die einzelnen Module haben, auf deren Grundlagen auch das in diesem Forschungsprojekt entwickelte System basiert. Nach [Moe06] sind IE-Systeme typischerweise in zwei verschiedene Bereiche gegliedert, die unterschiedliche Phasen der Extraktion repräsentieren (vgl. Abbildung 1.1):

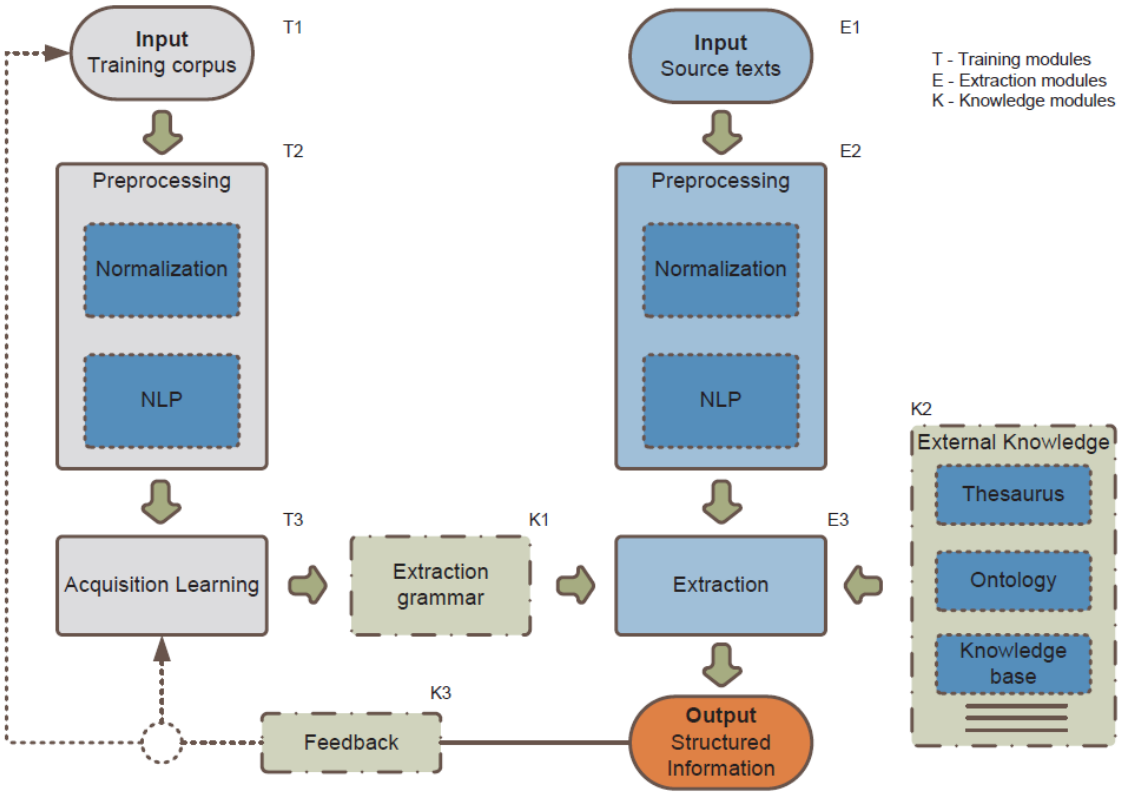


Abbildung 1: Aufbau eines Informationsextraktionssystem nach [Moe06]

• Trainingsphase:

In der Trainingsphase wird dem IE-System eine Sammlung von beispielhaften Dokumenten als Eingabe zur Verfügung gestellt. Dieser Trainingskorpus kann zuvor von einem Mitarbeiter mit entsprechenden Markierungen der relevanten Informationen versehen worden sein. Das IE-System setzt anschließend spezielle Lernverfahren ein, um aus dem annotierten Korpus geeignete Extraktionsgrammatiken zu bestimmen. Daneben gibt es auch Verfahren (unsupervised learning), bei denen keine Annotierung des Testkorpus notwendig ist. Werden in dem entwickelten IE-System keine Lernverfahren eingesetzt, so müssen die Extraktionsregeln von einem Entwickler manuell erstellt werden. In den meisten Fällen werden die Trainingstexte in mehreren Preprocessing-Schritten bearbeitet, um eine spätere Verarbeitung zu vereinfachen.

• Extraktionsphase:

In dieser Phase sollen Informationen aus bisher unbekannten Texten extrahiert werden. Auch hier wird der Text vorverarbeitet, bevor die eigentliche Extraktion erfolgt. Die Extraktionsregeln, die zuvor erlernt bzw. manuell erstellt wurden, dienen nun als Grundlage für die Informationsgewinnung. Oft wird dazu weiteres Wissen mit einbezogen, welches zum Beispiel in Form von Ontologien vorliegt. Die Extraktion liefert die gefundenen Informationen in einem strukturierten Format, welches von dem IE-System vorgegeben wird. Sehr häufig kommen dazu Templates zum Einsatz. Systeme, die mit Lernverfahren arbeiten, verwenden meistens Feedback-Mechanismen die in die Trainingsphase einfließen, um somit zusätzliches Wissen zur Verfügung zu stellen, welches die Extraktionsregeln weiter verbessern kann.

Das Ergebnis des IE-Systems wird meistens mit Hilfe von Templates als Ausgabeformat repräsentiert. Templates sind Informationseinheiten, die aus mehreren sogenannten Slots bestehen. Diese Slots beinhalten die einzelnen Attribute, die mit Hilfe der Extraktionsregeln gefüllt werden sollen. Dabei kann man diese Regeln in Single-Slot-Rules und Multi-Slot-Rules unterscheiden [NN06]. Während Single-Slot-Rules lediglich einen einzigen Slot mit Werten füllen, können Multi-Slot-Rules gleich mehrere parallel mit Informationen versehen. Templates repräsentieren häufig Elemente innerhalb einer Domäne (zum Beispiel Personen, Objekte etc.), können jedoch auch für abstraktere Begriffe wie Ereignisse (z.B. terroristische Anschläge [MUC91]) oder ein komplettes Szenario, welches sowohl Element-Templates und Ereignis-Templates enthält und ihre Relationen zueinander darstellt (vgl. [Chi98]), stehen. Templates können auf andere Templates verweisen, indem diese als Slot eingesetzt werden.

Zusammenfassend können in einem IE-Systems (ohne Training) drei verschiedene Module identifiziert werden:

- Preprocessing:
  In mehreren Schritten wird der Text für die anschließende Extraktion vorbereitet. In der Regel werden verschiedene Verfahren des Natural Language Processing (NLP) eingesetzt. Hierunter fallen morphologische Analysen, Tokenscanner, Eigennamenerkennung, Koreferenzauflösungen uvm. [Neu01]. In welchem Umfang ein Preproccesing der Texte notwendig ist, hängt stark von den zu extrahierenden Informationen ab. Deshalb bietet es sich häufig an, die einzelnen Preprocessing-Schritte modular aufzubauen, so dass diese ohne Aufwand in den Ablauf integriert werden können.

- Extraktionsregeln:
  Diese Regeln werden während der Extraktionsphase häufig durch Trigger ausgelöst und haben die Funktion, die Slots der jeweiligen Templates mit passenden Informationen zu füllen.

- Wissen:
  In diesem Modul wird das benötigte Wissen hinterlegt, welches während der Extraktion herangezogen werden kann. Dazu gehören die Templates, die mit Informationen gefüllt werden sollen, aber auch Ontologien, Thesauren etc., die domänen-spezifisches Wissen bereitstellen. Ontologien sind Netzwerke, die einen Wissensbereich beschreiben. Dies geschieht mit Hilfe einer festgelegten Terminologie sowie Relationen und Regeln zwischen den einzelnen Konzepten innerhalb des Netzes. Weiterführende Informationen zu Ontologien und wie diese der IE eingesetzt werden findet man u.a. in [Sch05], [Gru91] und [ZSS01].

4.2 *Verwandte Forschungsprojekte*

Projekte, die sich konkret mit der Informationsextraktion zur Prozessunterstützung im Bauwesen beschäftigen, konnten nicht recherchiert werden. Deshalb beschränkt sich die Auswahl der Projekte auf IE-Systeme, die in anderen Domänen angesiedelt sind. Zuerst wird das Projekt DYNIPOS vorgestellt, welches eine starke Ähnlichkeit zu den erklärten Zielen des OnToBau-Forschungsprojektes zeigt. DYNIPOS versucht die Prozessabläufe in öffentlichen Verwaltungseinrichtungen zu optimieren, indem Mitarbeitern proaktiv Wissen zur Verfügung gestellt wird.

Anschließend zeigt das ZENON-Projekt des Fraunhofer-Instituts, wie aktuelle IE-Systeme zur Unterstützung in militärischen Krisengebieten eingesetzt werden können. ZENON untersucht militärische Meldungen und extrahiert daraus navigierbare Entitäten- Aktivitäten-Netze.

Abschließend wir ein Projekt des Deutschen Forschungsinstituts für künstliche Intelligenz präsentiert. NEPOMUK ist die Realisierung eines semantischen Desktops zur Organisation von privaten und Unternehmensressourcen.

Das letzte Abschnitt gibt einen zusammenfassenden Überblick über alle vorgestellten Projekte und welche Ansätze in dieser Forschungsvorhaben motivierend waren.

4.2.1 DYONIPOS

DYONIPOS (DYnamic ONtology based Integrated Process OptimiSation) wurde von einem Konsortium u.a. bestehend aus dem Know-Center Graz und der Technischen Universität Graz durchgeführt und im Rahmen des Impulsprogramms FIT-IT zur Förderung anspruchsvoller IT-Innovationen und IT-Forschung in Österreich gefördert. Zielsetzung des Projektes ist die kontext-relevante, proaktive Bereitstellung von Informationen für den Wissensarbeiter in einem Unternehmen, um damit die Unterstützung wissensintensiver Geschäftsprozesse zu ermöglichen. Ausgangslage ist die Problematik, dass nicht formalisierbare Aufgaben innerhalb von Unternehmen stets Wissen zur Erfüllung der gestellten Anforderungen benötigen. Diese Menge an Informationen nimmt jedoch permanent zu, was zur Folge hat, dass der Wissensarbeiter oft nicht weiß, welches Wissen wo zur Verfügung steht [MR06]. Diese Problematik ist in Abbildung 2 schematisiert.

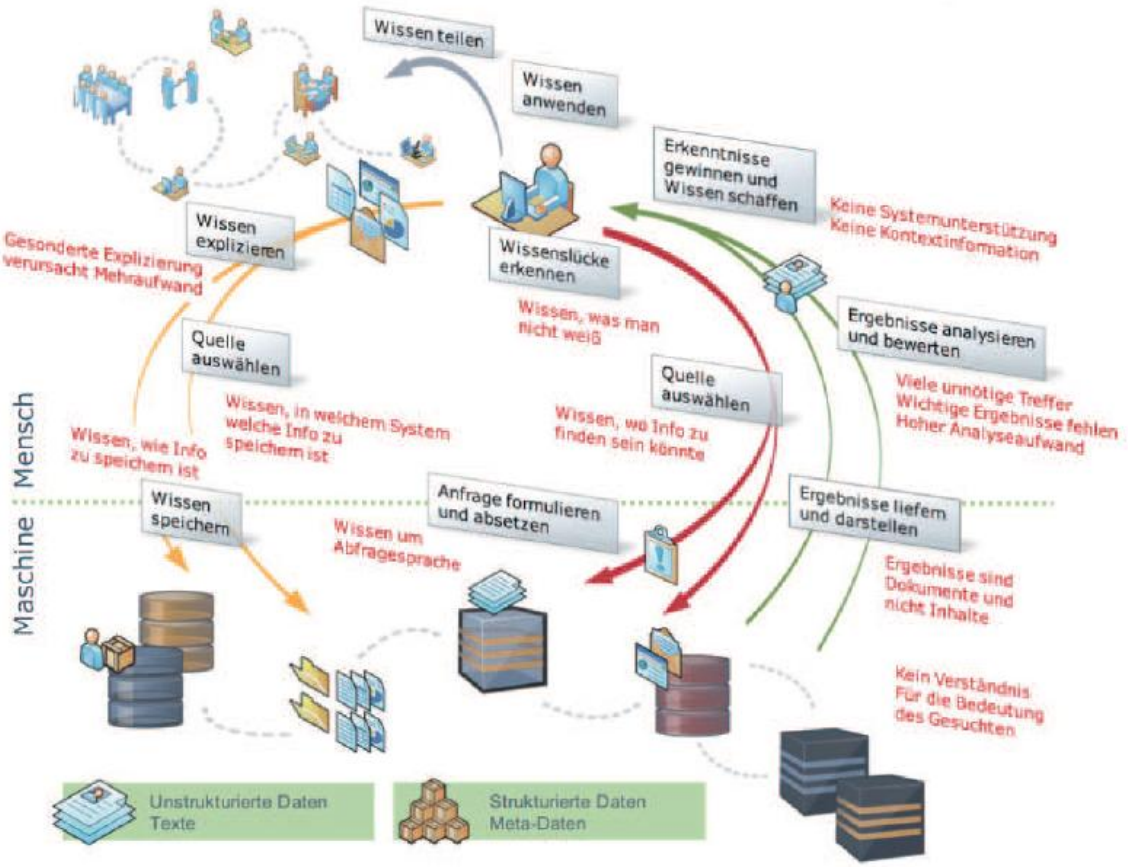


Abbildung 2: Zeitintensive Suche nach Informationen für einen Wissensarbeiter in einem Unternehmen [Dyo08]

DYONIPOS identifiziert den Informationsbedarf in dem bestehenden Kontext und recherchiert automatisch in einem zentral verfügbaren Wissensarchiv nach relevanten Informationen, wie z.B. Webseiten, Akten usw., die den Mitarbeiter in der aktuellen Situation unterstützen können. Damit DYNIPOS eine kontext-sensitive Bereitstellung von Wissen ermöglichen kann werden die Benutzerinteraktionen analysiert. Das System lernt dadurch welche Tätigkeiten der Benutzer durchführt und auf welche Informationsressourcen er zugreift. Aus diesen erlernten Interaktionen, Informationsbedürfnissen etc. wird eine persönliche Wissensbasis erstellt.

Der Wissensarbeiter wird durch einen persönlichen Agenten in seiner Arbeit unter stützt, indem dieser ihm proaktiv Informationen zur Verfügung stellt. Der Agent greift zur Laufzeit auf die hinterlegten Informationen der persönlichen Wissensbasis oder des zentralen Wissensarchivs zurück. Der Wissensarbeiter muss also nicht selbst nach Informationen suchen oder wissen, wo diese zu finden sind. Zusätzlich ist der Mitarbeiter aber in der Lage, gezielt über entsprechende Suchmasken nach Informationen zu suchen.

Zum Aufbau des Archivs sammelt der Agent Informationen auf dem Arbeitsplatz des Benutzers. Alternativ kann der Mitarbeiter jederzeit wichtige Ressourcen manuell an das Wissensarchiv schicken. Das Wissensarchiv integriert alle Wissensquellen des Unternehmens (z.B. Dateisysteme, Akten etc.) und analysiert diese unter semantischen Gesichtspunkten.

Abbildung 3 zeigt den gerade beschriebenen Ablauf. Im DYONIPOS-Projekt werden Techniken der formalen Wissensorganisation, semantischer Systeme und des Workflowmanagement eingesetzt. Die formale Wissensorgansation soll Verfahren und Konzepte zur Repräsentation der zentralen Wissensarchive (persönliches und unternehmensweites) und der Unternehmensprozesse liefern. Zur Formalisierung des Wissens werden Ontologien eingesetzt, hier baut DYONIPOS auf bereits existierende Standardsprachen wie RDF (Resource Description Framework) und OWL (Web Ontology Language) auf, erweitert diese aber um neue Sprachkonstrukte.

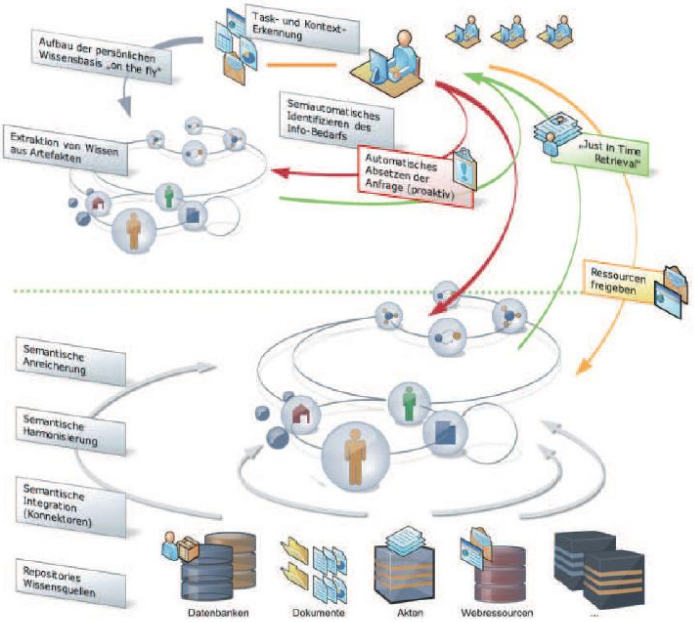


Abbildung 3: Integration des DYONIPOS-Systems in den Arbeitsablauf eines Wissensarbeiters [Dyo08]

Geschäftsprozesse in DYONIPOS werden in zwei Arten unterschieden: (1) Standardprozesse sind durch einen festen Arbeitsablauf gekennzeichnet und erfordern geringen dynamischen Zugriff auf Wissen. (2) Ad hoc Prozesse besitzen zwar eine definierte Ein- und Ausgabe, jedoch sind die Arbeitsschritte zum Erreichen der Aufgabe nicht standardisiert, sondern dem Wissensarbeiter überlassen. Beide Prozessarten werden in eigens entwickelten regelbasierten Wissensbasen modelliert. Diese Wissensbasen enthalten u.a. Informationen über Abhängigkeiten zwischen Prozessen, Rollen und Ressourcen [TRGL06].

DYONIPOS unterscheidet zwei Rollen mit unterschiedlichen Sichten auf die Prozesse. Process Engineers erstellen und erweitern die Standardprozesse des Unternehmens. Sie können des Weiteren die ad hoc Prozesse analysieren und als Grundlage für die Generierung von neuen Standardprozessen nehmen. Process Executers führen die Standardprozesse aus. Sobald jedoch ein Executer vom Standardprozess abweicht, erkennt DYONIPOS dies und erstellt automatisch neue ad hoc Prozesse. Abbildung 4 zeigt den Zusammenhang zwischen den Prozess- und Rollenarten innerhalb des Systems.

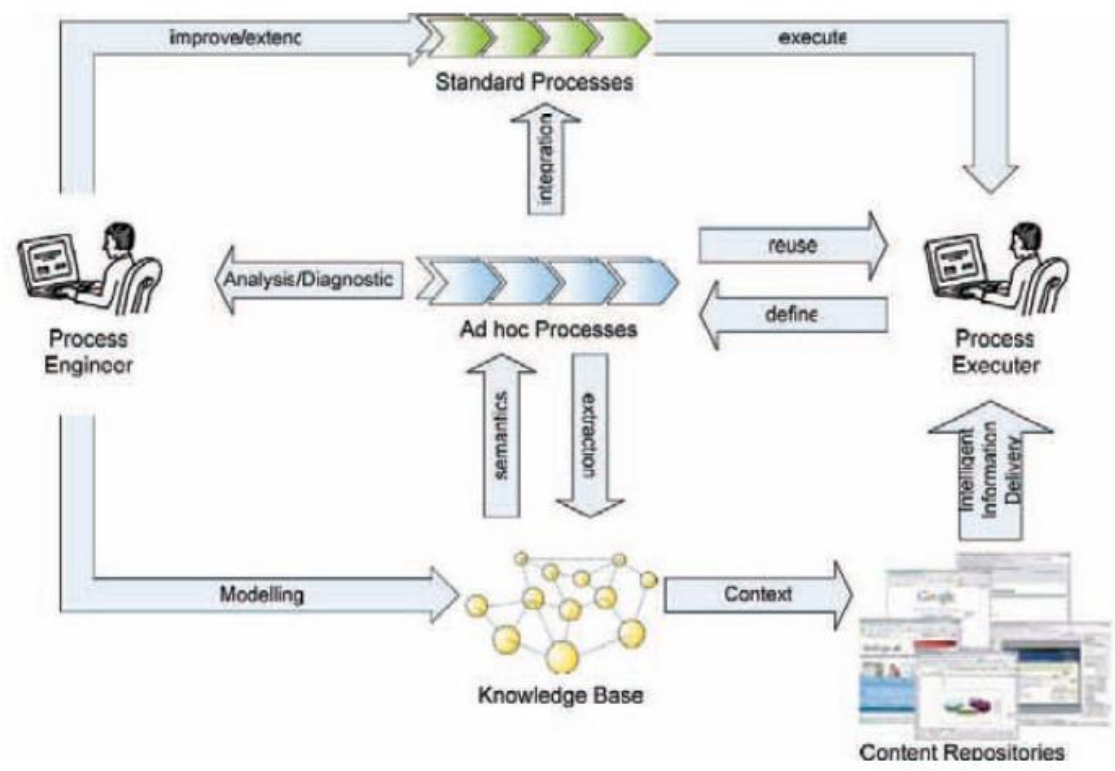


Abbildung 4: Prozesse innerhalb des DYONIPOS-Systems und ihre Beziehungen zueinander [TRGL06]

Zur Modellierung und Verbesserung von Prozessen unterstützt das System u.a. folgende Funktionalitäten:

- Vorschläge zur Workflow-Optimierung:
  Standardprozesse werden kontinuierlich analysiert. Abweichungen werden erkannt und dem Engineer berichtet, damit entsprechende Anpassungen durchgeführt werden können.
- Evaluation von Workflow-Änderungen:
  Wenn Prozesse geändert wurden, werden diese anschließend analysiert und mit Hilfe verschiedener Indikatoren kann eine Evaluation bezüglich des ursprünglichen Prozesses erfolgen.

- Visualisierung von Workflow-Veränderungen:
  Alle Änderungen innerhalb eines Workflows können mit einer Simulationssoftware dem Engineer helfen, Ineffizienzen und Abhängigkeiten zu anderen Prozessen zu ermitteln.

Das Projekt DYONIPOS wurde im Zeitraum März 2006 - Februar 2008 durchgeführt.

Laut Angaben auf der offiziellen Homepage befindet sich ein Prototyp derzeit in der IT-Abteilung des Bundesministeriums für Finanzen in Österreich im Piloteinsatz.

4.2.2 ZENON

Das Projekt ZENON wird aktuell am Fraunhofer Institut für Kommunikation, Informationsverarbeitung und Ergonomie (FKIE) durchgeführt. Ziel des Vorhabens ist die Entwicklung von Extraktionsverfahren zur inhaltlichen Analyse von militärischen Meldungen. Die veränderten Einsatzbereiche der Bundeswehr, zum Beispiel friedenserhaltende oder stabilisierende Operationen in Krisengebieten durchzuführen, führen zu Meldungen mit einem wesentlich höheren Informationsspektrum. Standen früher vor allem Gefechtsmeldungen im Vordergrund, so enthalten derzeitige Meldungen u.a. Beschreibungen über Konflikte zwischen Volksgruppen, Spannungen zwischen politischen Kräften usw. (vgl. [Hec05]).

Aussagen wie „A trifft B“ oder „C erschießt D“ enthalten Informationen über Aktivitäten und den beteiligten Entitäten. Diese Informationen werden extrahiert und zu einem Entitäten-Aktivitäten-Netzwerk, wie in Abbildung 5 zu sehen, zusammengefasst.

Dadurch können zum Beispiel Analysten der Bundeswehr bei militärischen Einsätzen unterstützt werden. Grundlage für die Realisierung des Projektes ZENON ist eine Sammlung von 4498 militärischen Meldungen (zumeist in englischer Sprache) aus dem KFOR-Einsatz (Kosovo Force) der Bundeswehr. Aus diesen Berichten wurden 800 Dokumente automatisch mit vorläufigen Annotationen versehen, die den KFOR-Corpus bilden.

Nach [Hec08] werden in den Korpus-Dokumenten folgende Annotationen vorgenommen:

- Original markups:
  Textstellen, die bereits strukturiert vorliegen (wie Empfänger, Betreff etc.) werden mit diesen Markierungen versehen.
- Token:
  Diese Annotationen enthalten die Wörter, die von dem Tokenizer bzw. Part-of-Speech Tagger (POS) geliefert werden. Hierunter fallen u.a. eine Klassifizierung der Zahlenwörter (Datum, Telefonnummer, Preis usw.) und das Zuordnen von Wörtern zu ihren grammatischen Kategorien (Verb, Adjektiv etc.).
- Gazetteer:
  Diese Markierungen zeichnen diejenigen Wörter aus, die über entsprechende Listen als Städte, Vornamen etc. identifiziert werden können.
  Sentence:
  Diese Annotationen markieren Sätze und Kommentare.
- Verb group:
  Kennzeichnet die verbalen Ausdrücke
- Named entities:
  Markiert die enthaltenen Entitäten innerhalb des Dokumentes.

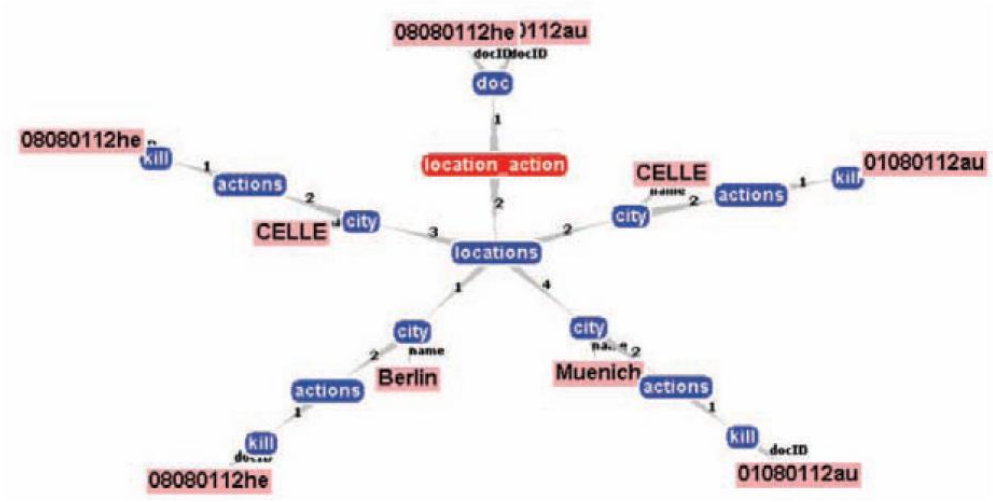


Abbildung 5: Entitäten-Aktivitäten-Netzwerk im Projekt ZENON

Nach der automatischen Annotierung wurden die 800 Dokumente manuell überprüft und entsprechende Korrekturen vorgenommen. Dieser Korpus dient zum Training und zur Optimierung des ZENON-Systems.

Das Projekt ZENON baut auf den Erkenntnissen und Ergebnissen des Vorgängerprojektes SOKRATES auf. In SOKRATES konnte die Verwendbarkeit von IE-Techniken zur inhaltlichen Analyse von deutschen Gefechtsmeldungen gezeigt werden. Die extrahierten Informationen wurden in Merkmalsstrukturen repräsentiert und mit Hilfe von Ontologien semantisch angereichert.

In SOKRATES werden shallow parsing Techniken eingesetzt. Dies bedeutet, dass lediglich ein partielles Verstehen des Satzes angestrebt wird, indem meist nur bestimmte Teile eines Satzes (chunks) verarbeitet werden. Die Verarbeitung erfolgt mit Hilfe von Transducern, dies sind endliche Automaten die zusätzlich eine Ausgabe bei der Bearbeitung von Eingabewörtern liefern. Diese Transducer können kaskadiert werden, indem die Ausgabe eines Transducer als Eingabe für den nachfolgenden Transducer dient.

SOKRATES verwendet das Tool SMES (Saarbrücken Message Extraction System) zur Generierung solcher Transducer. Dieses System wurde in Allegro Common Lisp entwickelt und unterstützt somit auch die komplette Funktionalität von Common Lisp. Weiterführende Informationen zum Aufbau und Arbeitsweise des SMES findet sich in [NBB+97].

Die folgende Abbildung 6 zeigt die Definition eines Transducers zur Ermittlung wer eine Gefechtsmeldung wann und wo geschickt hat. Die Bezeichnung des Transducers ist so-meldung-prolog. Der Transducer besteht aus einer sequentiellen Schaltung `(:conc)` von weiteren Transducern (z.B. `so-date-time date`). Das Ergebnis eines Transducers wird als Eingabe an den nächsten Transducer weitergegeben. Abschließend liefert der letzte Transducer sein Ergebnis an die morphologische Komponente `:morphix-punctuation`. Die extrahierten Informationen werden dann in der definierten Merkmalsstruktur `(:lisp (multi-acons :speaker meldender :time date :location standort))` ausgegeben.

```
(compile-regexp
'(:conc
        (:current-pos start)
        (:seek so-date-time date)
        (:seek so-meldender meldender)
        (:seek so-standort-meldender standort)
        (:morphix-punctuation ":")
        (:current-pos end)
)
:debug *debug*
:register-types '((:register start date meldender standort end))
:output-desc
'(:lisp (multi-acons :speaker meldender :time date :location standort))
:prefix T
:suffix nil
:name 'so-meldung-prolog
:compile *compile*
:write-to-file *trace-file*
)
```

Abbildung 6: Definition eines Transducers in der Programmiersprache LISP

4.2.3 NEPOMUK

NEPOMUK (Networked Environment for Personal Ontology-based Management of Unified Knowledge) war ein auf drei Jahre ausgelegtes EU-Projekt, das mit insgesamt 16 verschiedenen Partnern unter der Koordination des Deutschen Forschungsinstituts für künstliche Intelligenz (DFKI) in Kaiserslautern durchgeführt wurde. Ziel des Projektes war es, eine Lösung für die individuelle Wissenssuche, -verarbeitung und -austausch auf dem Desktop zu entwickeln.

Hierunter fällt u.a. die Beschreibung der auf dem Desktop befindlichen Informationen und deren Beziehungen untereinander. Zur Realisierung dieses Semantic Desktop werden Techniken aus dem Semantic Web eingesetzt, um die gefundenen Dateien im Computer mit semantischen Zusatzinformationen anzureichern.

Folgendes dargestellte Szenario wurde aus [RGSH08] entnommen und soll das Einsatzgebiet von NEPOMUK als Social Semantic Desktop (SSD) verdeutlichen: Die Benutzerin Claudia plant eine Dienstreise zu einem Projektmeeting in Belfast. Das Treffen geht über das anstehende CID-Projekt. Da Claudia das NEPOMUK-System auf ihrem Arbeitsplatzrechner verwendet, hat dieses bereits Informationen zu dem CID-Projekt aus E-Mails, Tabellenkalkulationen, Textdokumenten usw. erfasst und miteinander verknüpft. So wurde der Absender der E-Mail mit dem Adressbucheintrag, die E-Mail mit dem CID-Projekt und das anstehende

Projekttreffen mit einem Eintrag im Kalender in Beziehung gesetzt. Abbildung 2.14 zeigt schematisch das von NEPOMUK generierte semantische Netzwerk.

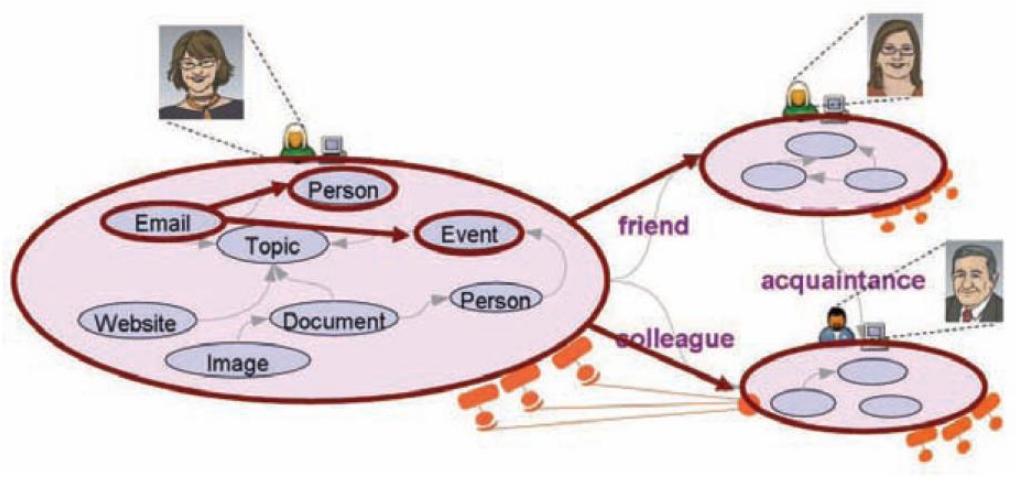


Abbildung 7: Anwendungsszenario für den Einsatz eines Social Semantic Desktop (Quelle: [RGSH08])

Claudia erstellt nun im Task Manager des Semantic Desktop eine neue Aufgabe mit der Beschreibung „Sitzung in Belfast wegen CID-Projekt“. Diese Beschreibung analysiert der Task Manager nach wichtigen Konzepten und erkennt „Sitzung“, „Belfast“ und „CIDProjekt“. Diese Konzepte sind in Wissensbasen des NEPOMUK-Systems hinterlegt und daher kann der Task Manager automatische Aktionen ausführen. Das Konzept „Sitzung“ erzeugt automatisch einen neuen Kalendereintrag. Auf Grund von hinterlegten Regeln, weiß der Task Manager, dass Dienstreisen genehmigt werden müssen. Demnach erzeugt der Task Manager automatisch einen neue Unteraufgabe (Subtask) für die Einholung dieser Genehmigung. Mit Hilfe von modelliertem Hintergrundwissen wird zusätzlich ein spezielles Formular für „Dienstreisen ins Ausland“ mit der neuen Unteraufgabe verknüpft. Bevor Claudia den Subtask an den Semantic Desktop der verantwortliche Person weiterleitet, sucht sie nach Dokumenten, die in Beziehung mit dem CID-Projekt stehen und verknüpft diese mit dem Subtask.

Zur Beschreibung der verschiedenen Ressourcen auf dem Desktop, zur Repräsentation von persönlichen Benutzermodellen und zur Erfassung von Metadaten werden in NEPOMUK drei verschiedene Modelle eingesetzt:

• Annotation Model (AM):

Dieses Modell erfasst allgemein Metadaten wie Autor, Erstelldatum usw. zu möglichen Ressourcen innerhalb des Systems.

• Information Element Model (IEM):

Dieses Modell beschreibt die Ressourcen (Dokumente, EMails etc.) innerhalb des Systems formal. Das IEM unterscheidet dabei sechs unterschiedliche Typen von Ressourcen: Dateien, Nachrichten, Kontakte, Kalendereinträge, Audio- und Bilddaten.

• Personal Information Model (PIM):

Dieses Modell erlaubt dem Benutzer seine eigene Wissensbasis aufzubauen. Es soll u.a. dem Benutzer die Organisation seines Social Semantic Desktop erleichtern. Dies kann zum Beispiel dadurch erreicht werden, dass der Benutzer eigene, neue Konzepte (wie in dem Szenario beschrieben) erstellen und selbstständig mit Informationen anreichern kann.

Alle drei Modelle sind in NEPOMUK mittels Ontologien realisiert worden. Das AM wird durch die NEPOMUK Annotation Ontology (NAO), das IEM durch die NEPOMUK Information Element Ontology (NIE) und das PIM durch die Personal Information Model Ontology (PIMO) repräsentiert. Abbildung 9 zeigt die drei verschiedenen Abstraktionslayer der eingesetzten Ontologien. Das Representational Layer enthält Ontologiesprachen auf denen die anderen Sprachen aufbauen. Neben den standardisierten Ontologiesprachen Resource Description Framework Schema (RDFS) und Web Ontology Language (OWL), ist hier die NEPOMUK Representation Language (NRL) definiert. Mit Hilfe dieser Ontologie werden die drei bereits genannten Ontologien zur Repräsentation der Modelle definiert.

Die NAO definiert u.a. Konstrukte mit denen die verschiedenen Ressourcen innerhalb des SSD miteinander verknüpft werden können. Ähnlich wie mit Hilfe des Dublin Core werden hierzu Metadaten über die Ressource erfasst. Abbildung 9 zeigt welche möglichen Annotationen vorgenommen werden können. Die drei Annotationen nao:isTopicOf, nao:hasTopic und nao:isRelated werden dazu verwendet, um Ressourcen untereinander in Relation zu setzen.

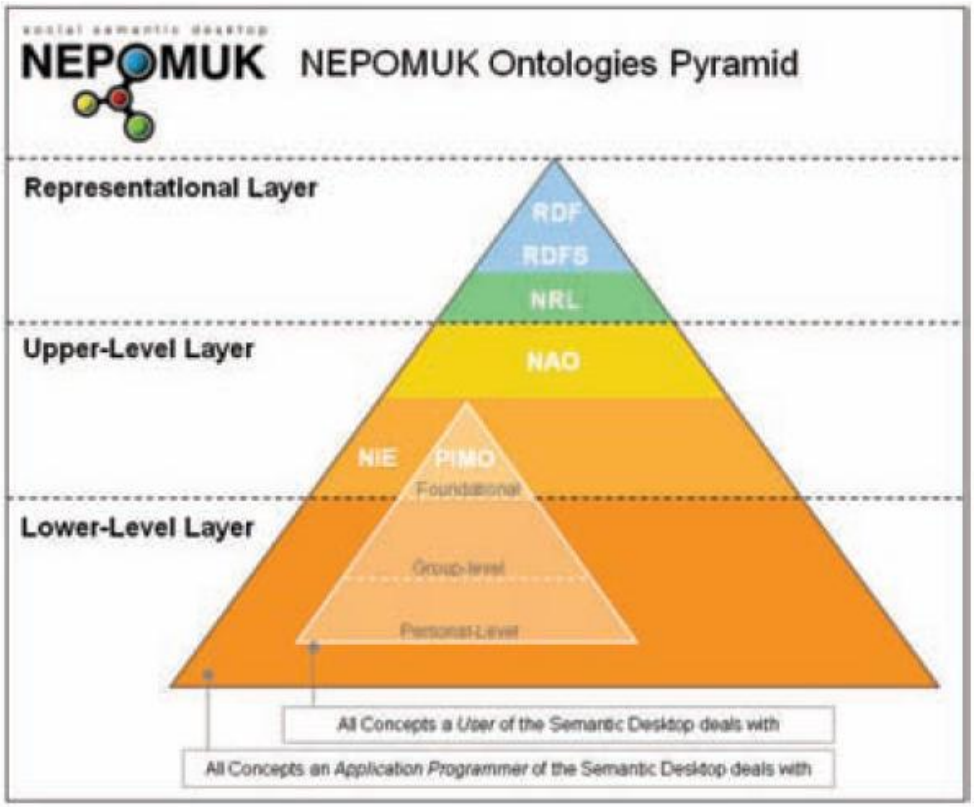


Abbildung 8: NEPOMUK setzt hierarchisch aufgebaute Ontologien ein, die in verschiedenen Layern organisiert sind (Quelle: [RGHS08]).

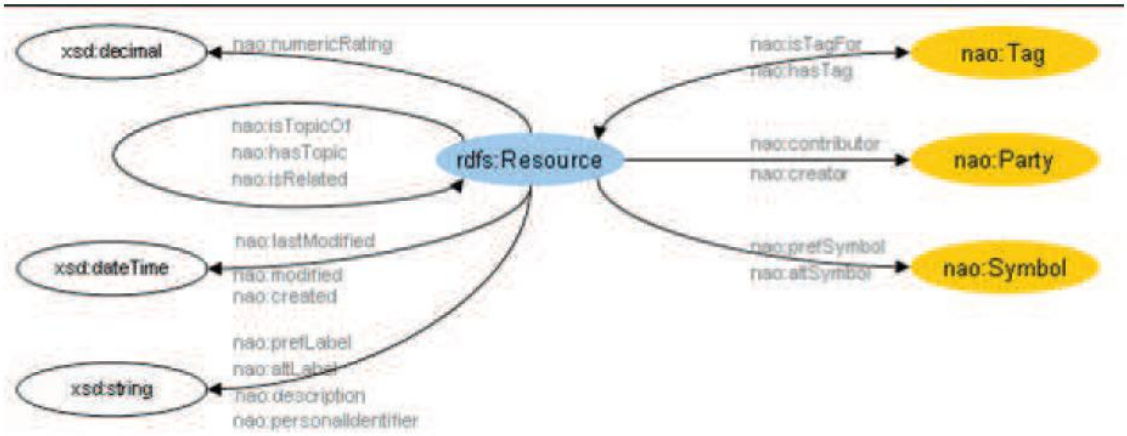


Abbildung 9: Mögliche Annotationsmöglichkeiten von Ressourcen in NEPOMUK (Quelle: [RGHS08]).

Die NIE beschreibt den Inhalt der Ressourcen und baut dabei auf standardisierten Formaten wie ID3 für Musikdateien, Ical für Kalendereinträge, EXIF für Bilddateien usw. auf. NIE besteht aus sieben Ontologien, von denen fünf in der Abbildung 10 dargestellt sind.

- NEPOMUK File Ontology (NFO):
  Beschreibt Dateisysteme, Ordner, Dateien und verschiedene elektronische Dokumente.

- NEPOMUK Contact Ontology (NCO):
  Beschreibt Kontaktinformationen aus einem Adressbuch und basiert auf der vom World Wide Web Consortium (W3C) spezifizierten VCARD-Ontology.

- NEPOMUK Message Ontology (NMO):
  Beschreibt EMails und Instant Messages (Nachrichten die z.B. über Chat-Programme wie ICQ verschickt werden).

- NEPOMUK Calendar Ontology (NCO):
  Beschreibt Kalendereinträge und basiert auf der vom W3C spezifizierten ICALOntology.

- NEPOMUK EXIF Ontology (NEXIF):
  Beschreibt Bilddateien und Videos und basiert auf dem EXIF-Standard.

- NEPOMUK ID3 Ontology (NID3):
  Beschreibt Audiodateien mit Hilfe von ID3 Metadaten.

Für ausführliche Informationen zu den eingesetzten Ontologien wird auf [RGSH08] und [GOR+07] verwiesen.

Zur Realisierung des Social Semantic Desktop kommt in NEPOMUK eine serviceorientierte Architektur (SOA) zum Einsatz. Die einzelnen Services stellen die Funktionalitäten des SSD zur Verfügungen und werden in zwei Kategorien unterschieden. In der ersten Kategorie (Semantic Desktop Services) werden Services zusammengefasst, die für die Realisierung des

lokalen Desktops notwendig sind. Darunter fallen alle Funktionen, die notwendig sind, um eine persönliche Wissensbasis aufzubauen, externe Applikationen (z.B. EMail- und Office-Programme) anzubinden, Ressourcen auf dem Desktop zu organisieren uvm. Die zweite Kategorie sieht Services vor, die für die sozialen Interaktionen (Social Services) zuständig sind. Darunter fällt zum Beispiel, dass Informationen zwischen den einzelnen SSDs ausgetauscht werden, Workflows modelliert werden können uvm.

Die Services werden in verschiedenen Schichten organisiert, die in Abbildung 10 zu sehen sind. Die Semantic Desktop Core Services stellen die grundlegenden Funktionen zum Auffinden, Verarbeiten, Speichern und Suchen von Ressourcen zur Verfügung. Des Weiteren wird die Kommunikation und Koordination der einzelnen Services untereinander übernommen. Die Helper Services stellen unterstützende Funktionen bereit, die von anderen Services in Anspruch genommen werden können. Die Extension Services stellen Funktionen zur Verfügung, die es Entwicklern erlauben eigene Erweiterungen des SSD zu implementieren. In der obersten Schicht befinden sich die Enduser Applications. Diese umfassen die von NEPOMUK bereitgestellten Applikationen, wie die PSEW-Oberfläche oder den Task Manager. Des Weiteren finden sich hier einige Plugins für bekannte Applikationen von MS Office, dem Firefox Webbrowser bis hin zum Thunderbird E-Mail-Client.

Diese Plugins ermöglichen dem SSD den Zugriff auf die darin enthaltenen Informationen. Die PIM Tools erlauben es dem Benutzer seine eigenen Wissensbasis zu erweitern, indem er neue Tasks erstellt und organisiert.

Auf die einzelnen Services wird an dieser Stelle nicht im Detail eingegangen, sondern auf [HVA09] und [RGSH08] verwiesen. Da ein Service für diese Arbeit jedoch von Interesse ist, soll dieser kurz vorgestellt werden.

Die Keyphrase Extraction ist eine Komponente der Text Analytics Services, die automatisch Schlüsselwörter aus Textdokumenten extrahieren soll. Dieser Service wird u.a. eingesetzt, wenn neue Ressourcen mit Metadaten angereichert werden sollen, um eine Auswahl an beschreibenden Schlüsselwörtern für das Dokument automatisch zu bestimmen. Die Architektur des Services ist in Abbildung 11 dargestellt und benötigt keine Trainingsphasen bevor er eingesetzt werden. Die Keyphrase Extraction ist einsetzbar für die Sprachen Englisch, Französisch und Deutsch und wurde in Java realisiert. Wie im Projekt ZENON kommt auch hier die GATE Toolbox zum Einsatz und erweitert diese um zusätzliche Plugins.

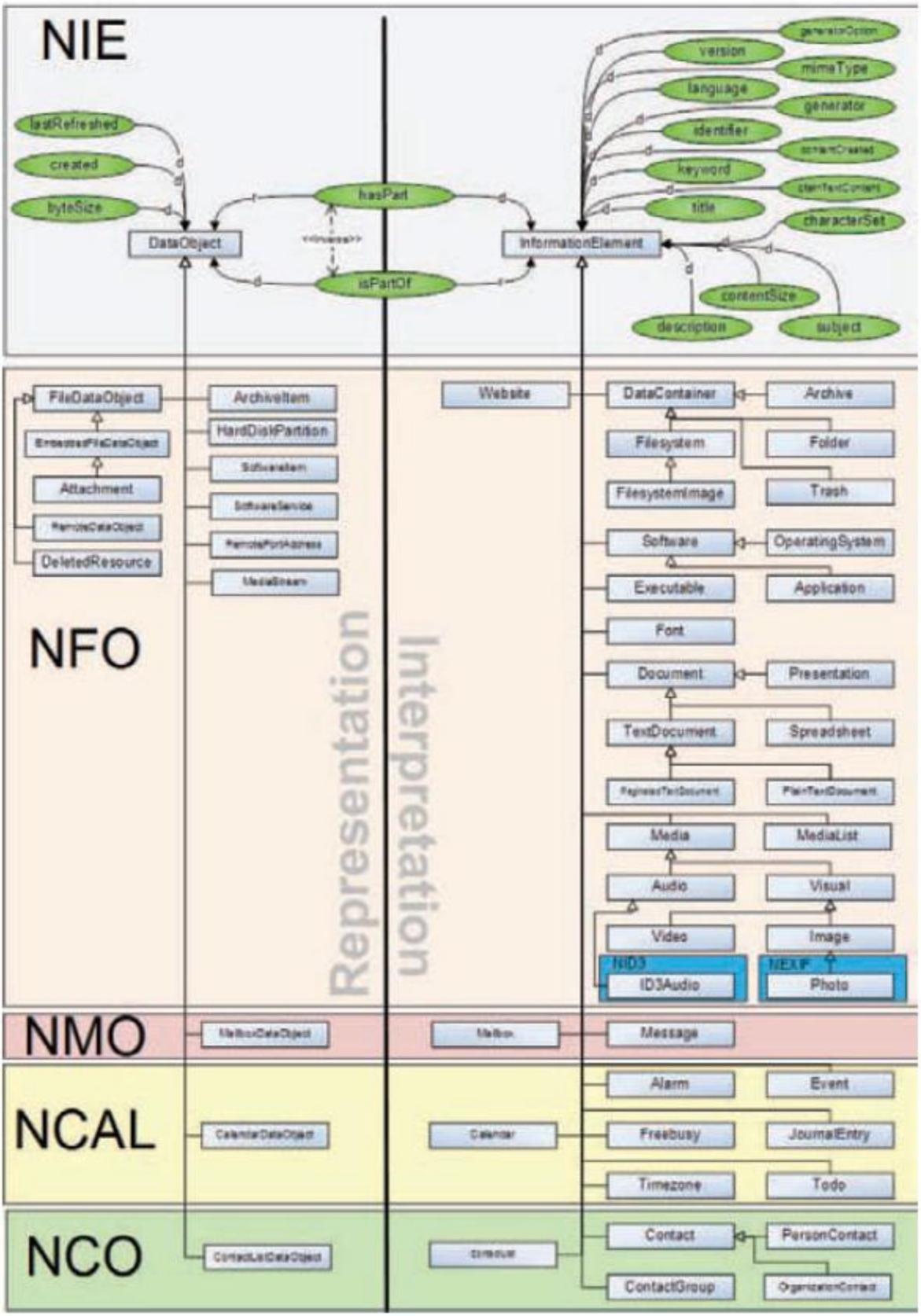


Abbildung 10: Aufbau des NEPOMUK Information Element Ontology Framework

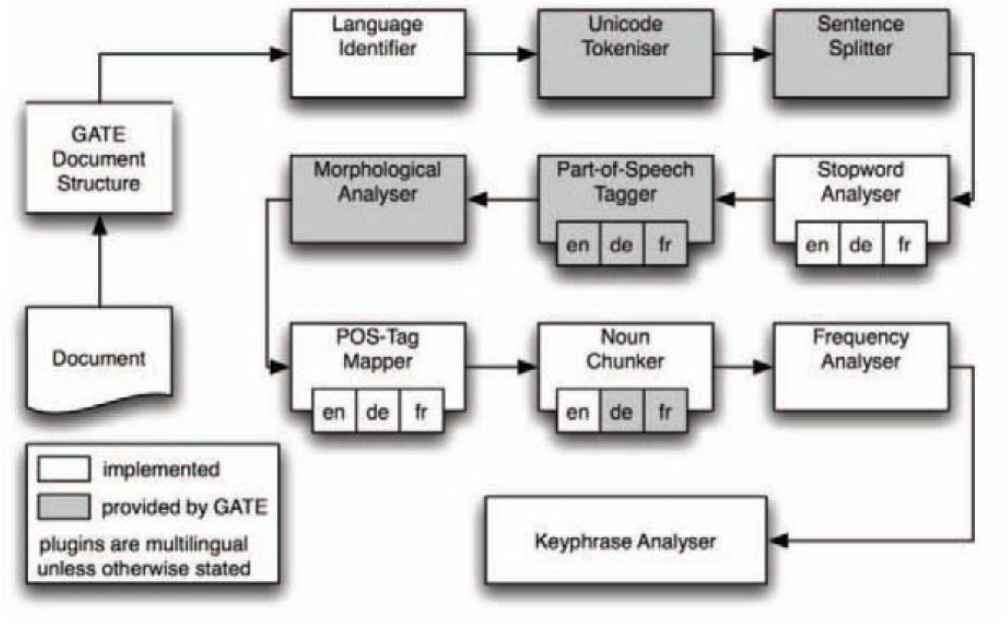


Abbildung 11: Keyphrase Extraction Architecture (Quelle: [HVA09]).

Um ein Dokument verarbeiten zu können, muss es in die GATE-Datenstruktur transformiert werden. Bevor eine Reihe von Textverarbeitungsmodulen folgen, wird die Sprache des Dokumentes identifiziert. Der Unicode Tokeniser und Sentence Splitter sind Module, die bereits von der GATE-Toolbox mitgeliefert werden und den Text zuerst in Wörter und anschließend in Sätze unterteilt. Der anschließende Stopword Analyser ist nicht Bestandteil von GATE und wurde selbst implementiert. Er entfernt Wörter die keine Relevanz für die Erfassung des Textinhaltes haben. Die nächsten beiden Module sind wieder Teil der GATE-Toolbox und führen ein Part-of-Speech Tagging und morphologische Analysen durch. Das Mapping der POS-Tags und die Gliederung des Textes in Nominalphrasen (Noun Chunker) sind NEPOMUK-Module. Der Frequency Analyser bestimmt mit Hilfe statistischer Verfahren die Signifikanz der einzelnen Wörter in Relation zu einem allgemeinem Referenz-Korpus. Mögliche Kandidaten werden mit Hilfe von Clustering-Verfahren in Gruppen zusammengefasst. Die wahrscheinlichsten Schlüsselwörter jedes Cluster werde als Ergebnis vorgeschlagen (vgl. [HVA09]).

Abschließend soll nun das zu Beginn des Kapitels angesprochene Szenario wiederaufgegriffen werden, um das Zusammenspiel der hier vorgestellten Komponenten des SSD zu verdeutlichen. In Abbildung 2.20 sind die ersten Schritte anhand eines Sequenzdiagramms dargestellt: Nachdem Claudia in der Oberfläche ihres SSD die Beschreibung für einen neuen Task angegeben hat, wird dieser von dem PIM Tool erfasst und an den Task Manager delegiert. Die Aufgabe des Task Managers ist die Analyse dieser Beschreibung und die Generierung von konkreten Tasks und eventuellen SubTasks. Dazu wird die Beschreibung von Claudia mit Hilfe der Text Analytics Services nach wichtigen Begriffen untersucht. Der angesprochene Service liefert als Ergebnis die Konzepte „Ereignis = Sitzung“, „Stadt = Belfast“ und „Projekt = CID“ zurück. Anhand dieser Informationen erstellt der Task Manager mit Hilfe eines weiteren Service zwei Subtasks, welche an den SSD von Claudia geschickt werden und in ihrer Oberfläche angezeigt werden.

Das Projekt wurde am 31.12.2008 abgeschlossen, wird aber in verschiedenen Open Source Projekten weitergeführt. Beispielsweise wird im Projekt NEPOMUK-Eclipse eine integrierte

Eclipse-Umgebung auf Basis der NEPOMUK-Architektur entwickelt und das Projekt NEPOMUK-KDE verfolgt ähnlich Ziele zur Integration von NEPOMUK in K Desktop Environments (KDE), einer Arbeitsumgebung die vorrangig in UNIX-Betriebssystemen zum Einsatz kommt.

4.2.4 Anknüpfung an die vorgestellten Forschungsprojekte

Das DYONIPOS-System wurde zur Unterstützung von Sachbearbeitern in Ministerien oder vergleichbaren öffentlichen Einrichtungen entwickelt. Die Zielsetzungen des Projektes ähneln dem Forschungsprojekt OnToBau vor allem in der proaktiven Unterstützung des Benutzers durch Bereitstellung geeigneter Dokumente sowie der Einführung eines intelligenten Agentensystems, welches die persönliche und unternehmensweite Wissensbasis mit Informationen anreichert. DYONIPOS verfolgt jedoch einen stark prozessorientierten Ansatz, der die Geschäftsprozesse in den Mittelpunkt stellt. Diese Vorgehensweise ist beim Vorliegen einer stark strukturierten Prozesslandschaft angebracht, in einer Umgebung wie dem Bauwesen sind die Prozesse jedoch eher als semistrukturiert zu bezeichnen. Deshalb verfolgt das OnToBau-Projekt einen eher dokumentenorientierten Ansatz, in dem die Bereitstellung von Wissen stärker an die Interaktionen des Benutzers als an vordefinierte Prozesse gekoppelt ist. Dennoch bietet DYONIPOS einige bereits angesprochene interessante Ansätze und der prozessorientierte Gedanke sollte nicht komplett vernachlässigt werden. Leider geben die Veröffentlichungen kaum Aufschluss über die konkreten Konzepte und Realisierungen, so dass lediglich Anregungen gegeben werden konnten.

Das Projekt ZENON extrahiert aus militärischen Gefechtsmeldungen Informationen, um diese anschließend in taktischen Karten zu visualisieren. Auch wenn ZENON wenig Gemeinsamkeiten mit dem Aufbau eines Wissensarchivs zur Optimierung der Angebotserstellung aufweisen kann, so ist der Aspekt der Visualisierung von Informationen auch für das OnToBau-Projekt von Interesse. Das Wissen innerhalb des Archivs soll dem Benutzer nicht nur proaktiv zur Verfügung stehen, sondern dazu auch gezielt eingesetzt werden können. Dies erfordert eine geeignete visuelle Aufbereitung und Darstellung. Des Weiteren sind die in ZENON verwendeten Transducer motivierend für einen ähnlichen Ansatz in dieser Arbeit. Ein Transducer ist ein endlicher Automat, der Eingaben verarbeitet und die Ausgabe an den nächsten Transducer weiterreicht. Der Aufbau des Preprocessing-Moduls orientiert sich ebenfalls an diesem Prinzip.

Abschließend wurde mit NEPOMUK ein System zur Realisierung eines Social Semantic Desktops vorgestellt. NEPOMUK konnte die meisten Inspirationen für diese Arbeit liefern, nicht zuletzt, da es zahlreiche Dokumentationen zu technischen Realisierungen des Projektvorhabens gab, die ausführlicher waren als die der anderen Projekte. NEPOMUK versucht die Ressourcen auf dem Desktop des Benutzers zu analysieren und zueinander in Beziehung zu setzen. Während der Arbeit analysiert NEPOMUK das Benutzerverhalten und generiert ein persönliches Benutzerprofil. Zudem kann der Benutzer jederzeit Tasks erstellen und Ressourcen mit diesen verknüpfen bzw. NEPOMUK versucht automatisch relevante Ressourcen einem Task hinzuzufügen. Diese Tasks können außerdem in den Workflow eines Unternehmens eingebettet werden, wie dies am Beispiel der Dienstreisegenehmigung gezeigt wurde. Zur Beschreibung der Ressourcen, des Benutzermodells, der Tasks usw. setzt NEPOMUK Ontologien, die auf den beiden Ontologiesprachen RDF und OWL basieren. In OnToBau spielt die Verarbeitung verschiedener Ressourcen innerhalb eines Unternehmens

eine zentrale Rolle. Zur Beschreibung dieser unterschiedlichen Ressourcen werden auch in dieser Arbeit Ontolgien eingesetzt, die durch die Ansätze in NEPOMUK motiviert sind.

Die detaillierten Realisierung werden im Kapitel Eingehende Darstellung genauer beschrieben.

5 Zusammenarbeit mit den Projektpartnern

5.1 *Mindox GmbH*

Die Mindox GmbH stellte eine Komponente zur Klassifikation von Dokumenten anhand einer vorgegebenen Ontologie bereit. Diese Komponente ist in der Lage, sowohl gescannte Dokumente als auch Dokumente aus Internetquellen zu klassifizieren. Im Rahmen des Projektes wurde die bereits vorhandene Ontologie weiter entwickelt, um darin auch vorgangsspezifisches Wissen abbilden zu können und um die Ontologie benutzerspezifisch aufbauen und erweitern zu können. Die von der Mindox GmbH eingesetzten Retrievalverfahren wurden erweitern, um den Anforderungen an das Forschungsvorhaben gerecht zu werden.

5.2 *Baufirmen*

Die Projektpartner aus dem Baugewerbe, die Firmen **Horn** GmbH und Co KG, **Dupré** Bau GmbH & Co KG und die **Schottler** GmbH, nutzen alle die Software der Mindox GmbH und die Sceye Kamera der **SilverCreations** AG, um Dokumente mit Bezug zu ihrem Geschäftsfeld digital zu speichern und bei Bedarf schnell darauf zugreifen zu können. Als ein wesentlicher Aspekt wird dabei übereinstimmend die Weitergabe des Wissens über die Existenz relevanter Information innerhalb des Unternehmens gesehen, hier insbesondere von den Inhabern oder Geschäftsführern an die Sachbearbeiter. Dieser Aufgabe wandte sich das Projekt zu.

Die drei genannten Firmen übernahmen die Rolle der prototypischen Anwender. Vor allem mit den Firmen **Schottler** und **Horn** wurden zusammen relevante Dokumente, Prozesse und Anwendungskontexte definiert. Dies umfasste die Analyse der bestehenden Prozesse, sowie deren Berücksichtigung für die Modellierung mit einer eventuellen neuen Definition von Abläufen und Zuständigkeiten. Dabei wurden insbesondere Prozesse mit einem generischen Charakter ausgewählt, die einen gewissen Allgemeinheitsgrad sowohl für die Prozesse der Partnerfirmen als auch für andere Firmen aus diesem Industriezweig haben.

In den Firmen arbeiten momentan die Geschäftsführer mit dem System von Prof. Bläsius zur Archivierung von Dokumenten. Als eine wesentliche Quelle relevanter Dokumente wurden vor allem Geschäftsdokumente genannt, die einen großen Anteil am alltäglichen Dokumentenaufkommen haben. Die automatische Verarbeitung und Verknüpfung dieser Dokumente in dem Unternehmen erachteten alle Beteiligten als eine enorme Erleichterung. Da Geschäftsdokumente im Baugewerbe überwiegend in Papierform ein- bzw. ausgehen, ist hier eine besondere Berücksichtigung der Dokumentenverarbeitung, -analyse und -archivierung notwendig gewesen.

Um den Einsatz des Systems weiter im Unternehmen auszubauen, wurde im nächsten Schritt ein Beispielprozess analysiert. Damit sollte eine Berücksichtigung wichtiger Dokumente bei der Sachbearbeitung gewährleistet werden. Eine genaue Ausgestaltung der Nutzung wurde zu Beginn des Projektes aufgesetzt und im Projektverlauf fortgeschrieben. Dazu standen

sowohl die jeweiligen Geschäftsführer als auch speziell benannte Sachbearbeiter in den Unternehmen zur Verfügung. In regelmäßigen Abständen fanden auf der Arbeitsebene Treffen zur gemeinsamen Diskussion des Projektstandes statt.

Eine prototypische Einführung in das System war aufgrund technische Komplexität in dem Forschungsvorhaben jedoch nicht wie geplant möglich. Nähere Erläuterungen dazu finden sich im Kapitel II2.4.3

5.3 *Silvercreations AG*

Zwischen dem ISS und der SilverCreations AG besteht schon seit einigen Jahren eine intensive Zusammenarbeit. Beide Partner haben ihren Sitz auf dem Campusgelände in Birkenfeld. Durch die räumliche Nähe können Soft- und Hardwareressourcen ohne großen Aufwand gemeinsam genutzt werden. Dazu zählen Rechner, Netzwerk, Kameras, und Erfassungsstationen für Dokumente, etc. Auch im Bereich der Software können so Lizenzen der SilverCreations AG, z.B. zur Bildbearbeitung oder zur Texterkennung, dem Projekt einfach zur Verfügung gestellt werden. Die beiden Geschäftsführer der SilverCreations AG sind Absolventen der Informatik an der Fachhochschule Trier und haben in der Vergangenheit wiederholt Lehraufträge im Studiengang Angewandte Informatik angeboten.

Zudem wurden mehrere Diplomarbeiten gemeinsam durchgeführt. Einige dieser Diplomanden haben danach eine feste Anstellung bei der SilverCreations AG erhalten. Im Rahmen des Projektbeitrags stellte die SilverCreations Lizenzen (z.B. die OCR-Engine (im Wert von ca. 4.500.-€), Runtimelizenzen (ca. 80.- € pro Rechner), Komponenten zum direkten Zugriff auf die Kamera) und Erfassungsgeräte (wie die Dokumentkamera Sceye) dem Projekt in ausreichender Anzahl zur Verfügung (laut UVP momentan 399.- €). Die Bereitstellung erfolgte entweder kostenlos oder auf Selbstkostenbasis, soweit sie z.B. für die Mitarbeiter der Partnerunternehmen bestimmt war. Weiter stellt sie Beratungsleistungen zur Verfügung, z.B. in der Integration der verschiedenen Anwendungssysteme und um den Aufbau und den Betrieb der Prototypen zu gewährleisten.

5.4 *Universität Trier*

Der Lehrstuhl von Herrn Prof. Dr. Bergmann an der Universität Trier forscht aktiv auf dem Gebiet Wissensmanagement & E-Business durch den Einsatz und die Weiterentwicklung von wissensbasierten Methoden, Internet-Technologien und Semantic Web. Herr Prof. Bergmann arbeitet darüber hinaus auch in der Thematik der „Ambient Intelligence“, die am ISS ebenfalls in verschiedenen Projekten thematisiert wird, u.a. im FHProfUnt Projekt GUIDO des Antragstellers. Für das aktuelle Projekt waren die Arbeiten von Herrn Prof. Dr. Bergmann im Bereich des Case-Based-Reasoning von besonderem Interesse. In diesem Themengebiet hat er verschiedene Projekte durchgeführt, auch mit industriellen Partnern. Dabei bestehen Berührungspunkte zu den fuzzy-basierten Retrievalverfahren, die am ISS entwickelt wurden. Diese wurden durch das gemeinsame Vorhaben vertieft.

Darüber hinaus konnte Prof. Bergmann als Doktorvater für Betreuung eines Projektmitarbeiters gewonnen werden. Durch diese Kooperation ist zudem eine übergreifende Kooperation der beiden Fachbereiche Informatik entstanden, die inzwischen in regelmäßige Doktoranden-Treffen gemündet ist. Eine weitere Kooperation bis hin zu einem gemeinsamen Doktoranden-Kollege sind derzeit geplant.

II EINGEHENDE DARSTELLUNG DER ERGEBNISSE DES VORHABENS

1 Ziel des Forschungsvorhaben

Ziel des Vorhabens war es für kleine und mittelständische Unternehmen Voraussetzungen zu schaffen, so dass diese Wissen in ihrem Unternehmen effektiv beschaffen, archivieren und recherchieren können. Daneben sollte ein persönlicher Agent den Mitarbeitern in den Unternehmen pro-aktiv relevantes Wissen zur Verfügung stellen.

2 Wissenschaftliche Ergebnisse

In diesem Kapitel werden die erzielten wissenschaftlichen Ergebnisse präsentiert. Dazu werden die einzelnen Teilergebnisse in den bereits vorgestellten Arbeitspaketen und -schritten näher beschrieben.

2.1 *Arbeitspaket 1: Grundlagen*

2.1.1 Analyse Anwendungsdomäne (Absprachen/Interviews mit Nutzern und Anwendern)

Zu Beginn des Forschungsvorhaben wurden mit den beteiligten Projektpartnern Interviews durchgeführt, um mögliche Schwachstellen in der Bearbeitung der Wissensprozesse aufzudecken und entsprechendes Optimierungspotenzial zu identifizieren. Beispielhaft soll hier die Firma Schottler näher vorgestellt werden, bei der sich vor allem der Prozess der Angebotserstellung als sehr wissensintensiv herausstellte.

Die Firma Schottler wurde 1886 als Schmiedebetrieb in Dörbach gegründet. 1920 wurde die Betriebspalette um den Industriezweig Wasserinstallation und im Jahre 1953 um den Zweig Zentralheizungsbau erweitert. Zudem verfügt die Firma über Kompetenzen auf dem Gebiet der Regenwassernutzung, sowie Wasser- und Abwassertechnik. Zwei Schwerpunkte von Schottler liegen im Heizungs- und Sanitärbereich. Die Mitarbeiter beraten Kunden auf den Gebieten der Solar- oder Erdwärmetechniken, Heiz- und Lüftungssysteme für Passiv- und Niedrigenergiehäuser oder helfen bei der Planung einer neuen Sanitäreinrichtung.

Als Anwendungsprozess zeigte sich die Angebotserstellung für Kunden der Firma Schottler als besonders geeignet. Im Folgenden wird dieser Prozess analysiert und Verbesserungsmöglichkeiten aufgezeigt. Ebenso werden dadurch auftretende Rahmenbedingungen für diese Forschungsvorhaben angesprochen.

2.1.2 Auswahl der Referenzprozesse und Wissenskontexte

Beschreibung des Referenzprozesses

Der komplette Prozess der Angebotserstellung ist in Abbildung 12 schematisiert. Ausgangspunkt ist ein persönliches Beratungsgespräch mit dem Kunden. Gemeinsam mit einem Sachbearbeiter werden dessen Vorstellungen und Wünsche geplant. Im Verlauf dieses

Gespräches werden verschiedene Produktkataloge herangezogen bzw. entsprechende

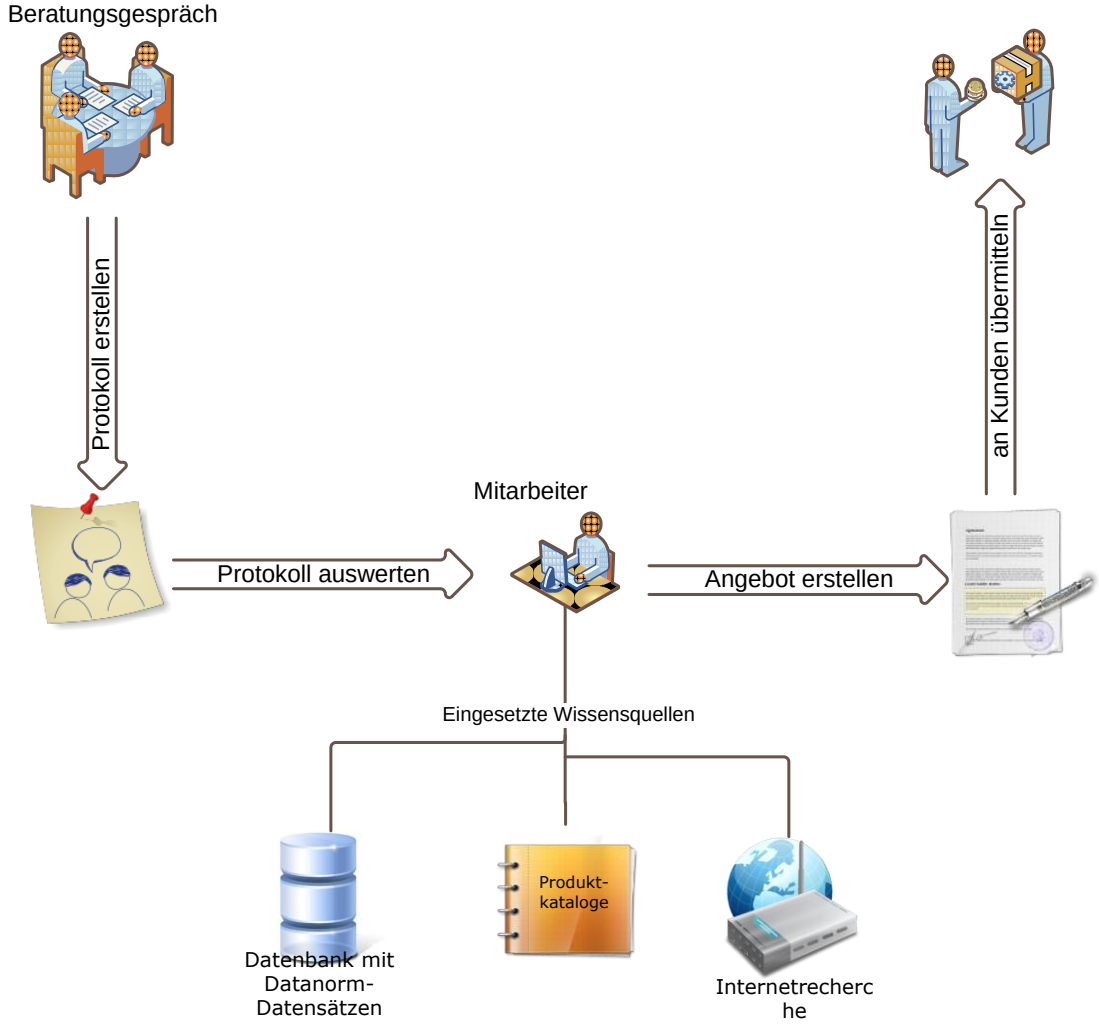


Abbildung 12: Ablauf einer Angebotserstellung bei der Firma Schottler

Ausstellungsstücke vor Ort ausgewählt. Der Sachbearbeiter führt während dieses Beratungsgesprächs ein Protokoll, welches in der Regel handschriftlich angefertigt wird. Hier werden alle notwendigen Informationen zu den von dem Kunden gewünschten Produkten festgehalten. Dieses Beratungsprotokoll verwendet der Sachbearbeiter anschließend, um ein Angebot für den Kunden zu erstellen. Die Firma Schottler verwendet eine spezielle Software zur Angebotserstellung. Die entsprechenden Produktpaletten werden über DATANORM-Datensätze in die Software eingepflegt. DATANORM ist ein Dateiformat für den Austausch von Artikelinformationen bzw. Stammdaten (wie z.B. Produktbeschreibungen, Preise, Rabattgruppen etc.) zwischen Herstellern, Lieferanten und Fachhändlern. Dieser Austausch gilt als Standardverfahren im Baugewerbe und wurde von dem DATANORM-Arbeitskreis herausgegeben. Die Weiterentwicklung wurde 1999 abgeschlossen. Abbildung 13 zeigt einen Ausschnitt aus einer Datei eines DATANORM-Datensatzes. Die jeweiligen Artikel sind zeilenweise abgespeichert. Die Informationen zu einem Artikel durch Trennzeichen (Semikolon) separiert.

```
V 190404Datanorm 2004                          gabo Systemtechnik GmbH                  Lieferprogramm  04/2004
T;N;GT-SRS;;1;;Schutzrohrschneider;2;;zum Abl„ngen von gabotherm Schutzrohr ;
T;N;GT-SRS;;3;;aus PE;;;;
E;N;FTDVA1;;1;;Fabrikat          : gabotherm;2;;Typ                : $$$$$$$$$$$$$$$$$$$$$;
E;N;FTDVA1;;3;;Dimension         : $$$$$$$$$$$$$$$$$$$;4;;Verpackungseinheit: $$$$$$$$$$$$$$$$$$$;
E;N;FTDVA1;;5;;Art.-Nr.          : $$$$$$$$;;;;
A;N;00032;20;Schutzrohrabschneider;GT-SRS 12-28;1;0;Stck;1383;001;080;;
B;N;00032;WERKZEUGE; ; ;0;0;0; ; ; ;0;1; ; ;
D;N;00032;1;T;GT-SRS;;2;E;FTDVA1;GT-SRS$12-28$1 Stk.$00032$;
T;N;GT-SE;;1;;Ersatzmesser f□r Schutzrohrschneider;;;;
E;N;FTVA1;;1;;Fabrikat          : gabotherm;2;;Typ                : $$$$$$$$$$$$$$$$$$$;
E;N;FTVA1;;3;;Verpackungseinheit: $$$$$$$$$$$$$$$$$$$;4;;Art.-Nr.          : $$$$$$;
A;N;00033;20;Ersatzmesser f□r Schutzrohrschneider;GT-SE;1;0;Stck;751;001;080;GT-SE;
B;N;00033;WERKZEUGE; ; ;0;0;0; ; ; ;0;1; ; ;
D;N;00033;1;T;GT-SE;;2;E;FTVA1;GT-SE$1 Paar$00033$;
T;N;GTF-FS;;1;;Ersatzteil-Set;2;;bestehend aus 2 Stk. Schonringen, ;
T;N;GTF-FS;;3;;Krallenring und O-Ring mit ;4;;Kunststoffmutter.;
A;N;00038;20;Ersatzteil-Set;GTF-Full-Set;1;0;SET;73;001;080;;
B;N;00038;gabo-Zubeh˝r; ; ;0;0;0; ; ; ;0;1; ; ;
D;N;00038;1;T;GTF-FS;;2;E;FTVA1;GTF-Full-Set$1 Set$00038$;
T;N;GTH-NRS;;1;;Nagel-Rundschelle;2;;zur Befestigung des gabotherm-Rohres;
T;N;GTH-NRS;;3;;12x1,3 mm auf Ziegelw„nden.;;;;
A;N;00039;20;Nagel-Rundschelle;GTH-NRS 12;1;0;Stck;11;002;040;;
B;N;00039;gabowall WHZ; ; ;0;0;0; ; ; ;0;0; ; ;
D;N;00039;1;T;GTH-NRS;;2;E;FTVA1;GTH-NRS 12$100 Stk.$00039$;
```

Abbildung 13: Ausschnitt aus einem DATANORM-Datensatz

Schwachstellenanalyse des Angebotsprozesses

Durch den Einsatz der Software zur Angebotserstellung ergeben sich verschiedene Problemstellungen: Zum einen müssen die Artikel- und Stammdaten von den Mitarbeitern konsequent gepflegt werden. Dies ist jedoch nach eigenen Angaben nicht immer der Fall. Zum anderen sind die Händler auf stets aktuelle DATANORM-Datensätze der Hersteller angewiesen. Jedoch liefern nicht alle Händler in regelmäßigen Abständen Updates bzw. kooperiert die Firma Schottler teilweise auch mit Herstellern, die keine DATANORM-Datensätze anbieten. Dies hat zur Folge, dass die ausgewählten Artikel in der Kalkulationssoftware unter Umständen veraltet bzw. überhaupt nicht vorhanden sind. Wenn ein solcher Fall eintritt, müssen die Produkte manuell aus entsprechenden Katalogen gesucht und in das System nachgetragen werden. Sollte kein aktueller Katalog vorhanden sein, wird der Mitarbeiter auf Informationen aus dem Internet zurückgreifen.

Die Mitarbeiter der Firma Schottler müssen also zur Erstellung ihre Angebote auf verschiedene Informationsquellen zurückgreifen, da die Artikelinformationen nicht zentral zur Verfügung stehen. Dies führt zu einem erheblich Mehraufwand, der u.a. darin begründet ist, dass zum Beispiel eine automatische Vervollständigung bei der Auswahl von Produkten nicht durchgeführt wird. Automatische Vervollständigung bedeutet, dass beispielsweise bei der Auswahl eines Waschbeckens notwendige Zubehörteile (wie Siphon, Montageschrauben etc.) in die Angebotserstellung mit aufgenommen werden, ohne dass der Mitarbeiter diese explizit auswählen muss. Die von Schottler eingesetzt Software kann dies leisten, jedoch nur wenn ein Produkt aus der Datenbank gewählt wurde. Wenn Produkte manuell aufgenommen werden, müssen notwendige Zubehörteile selbständig hinzugefügt werden. Für diesen Fall kann der Mitarbeiter ältere Angebote, die eine Ähnlichkeit mit der aktuellen Produktauswahl des Kunden aufweisen, heranziehen und daraus hilfreiche Informationen über notwendige Zubehörteile entnehmen. Diese Angebote werden jedoch papiergebunden archiviert, was ein Recherchieren erschwert.

Zusammengefasst konnten folgende Schwachstellen für diesen Geschäftsprozess identifiziert werden:

- die DATANORM-Datensätze müssen eigenständig gepflegt werden, was nicht immer erfolgt,

- die eingesetzte Software enthält somit u. U. nicht mehr aktuelle Produktdaten,
- bestimmte Produktdaten liegen in der Software überhaupt nicht vor,
- Mitarbeiter müssen in Produktkatalogen bzw. dem Internet fehlende Produkte nachschlagen,
- automatische Zubehörvervollständigung funktioniert nicht bei manueller Produkteingabe,
- die Suche nach ähnlichen, älteren Angeboten ist zeitaufwendig.

Optimierungspotenzial und mögliche Rahmenbedingungen

Eine Verbesserung dieses Prozesses kann durch eine teilweise Automatisierung der Angebotserstellung erreicht werden. Hierzu muss ein entsprechendes Archiv zur Verfügung stehen, welches das notwendige Wissen zur Verfügung stellt. Die manuelle Recherche der Mitarbeitern in Produktkatalogen, Internet und Aktenordnern kann somit im Idealfall entfallen.

Dieses Wissensarchiv wird aus den bereits angesprochenen Quellen (Produktdatenbank, Produktkataloge, Internet und archivierte Angebote) aufgebaut und ständig aktualisiert. Bei der Angebotserstellung werden die Informationen aus dem Wissensarchiv entnommen, wobei mittels geeigneter Darstellung von Relationen auch Zubehörteile automatisch ergänzt werden können. Aufgrund der vorgestellten Schwachstellen und Optimierungspotenziale ergaben sich für das Forschungsvorhaben folgende Rahmenbedingungen:

- Das Konzept des Extraktionssystem muss die Verarbeitung von Informationen aus verschiedenen Datenquellen vorsehen. Wegen der Vielfältigkeit an möglichen Datenquellen wird in dieser Arbeit die Extraktion aus Textdokumenten priorisiert. In diese Kategorie fallen u.a. die Produktkataloge, Internetseiten als auch archivierte Angebote und Rechnungen von früheren Bestellungen.
- Die prototypische Realisierung soll das Potenzial eines IE-Systems aufzeigen. Da sich vor allem die manuelle Recherche von Angeboten in den Aktenordnern als zeitintensiv erweist, soll der Prototyp Relationen zwischen den digitalisierten Angebots- und Rechnungsdokumenten erkennen und visualisieren.
- Spezielle Anforderungen an die einzusetzende Programmiersprache bestehen keine.

Die Firma Schottler setzt, ebenso wie die anderen beteiligten Unternehmen, Standardcomputer mit einem Windows-Betriebssystem ein. Da aus früheren Projekten jedoch bereits Arbeiten auf dem Gebiet der Dokumentenanalyse in der Programmiersprache LISP durchgeführt wurden und die Retrieval-Verfahren der Firma Mindox GmbH ebenfalls in dieser Sprache realisiert sind, bot sich diese Sprache an, da dadurch Module unter Umständen gegenseitig wiederverwendet werden können.

Angestrebte Verbesserungen des Referenzprozesses

In diesem Kapitel soll abschließend ein Konzept vorgestellt werden, wie die Angebotserstellung der Firma Schottler mit Hilfe eines IE-Systems und eines Wissensarchives optimiert werden könnte. Das Konzept ist in Abbildung 14 dargestellt. Im Vergleich zum derzeitigen Prozessablauf (vgl. Abbildung 12) wird die Erstellung des Angebotes von einer Software unterstützt. Ziel sollte es sein, dass die Software die Angebotserstellung weitestgehend vollautomatisch durchführt.

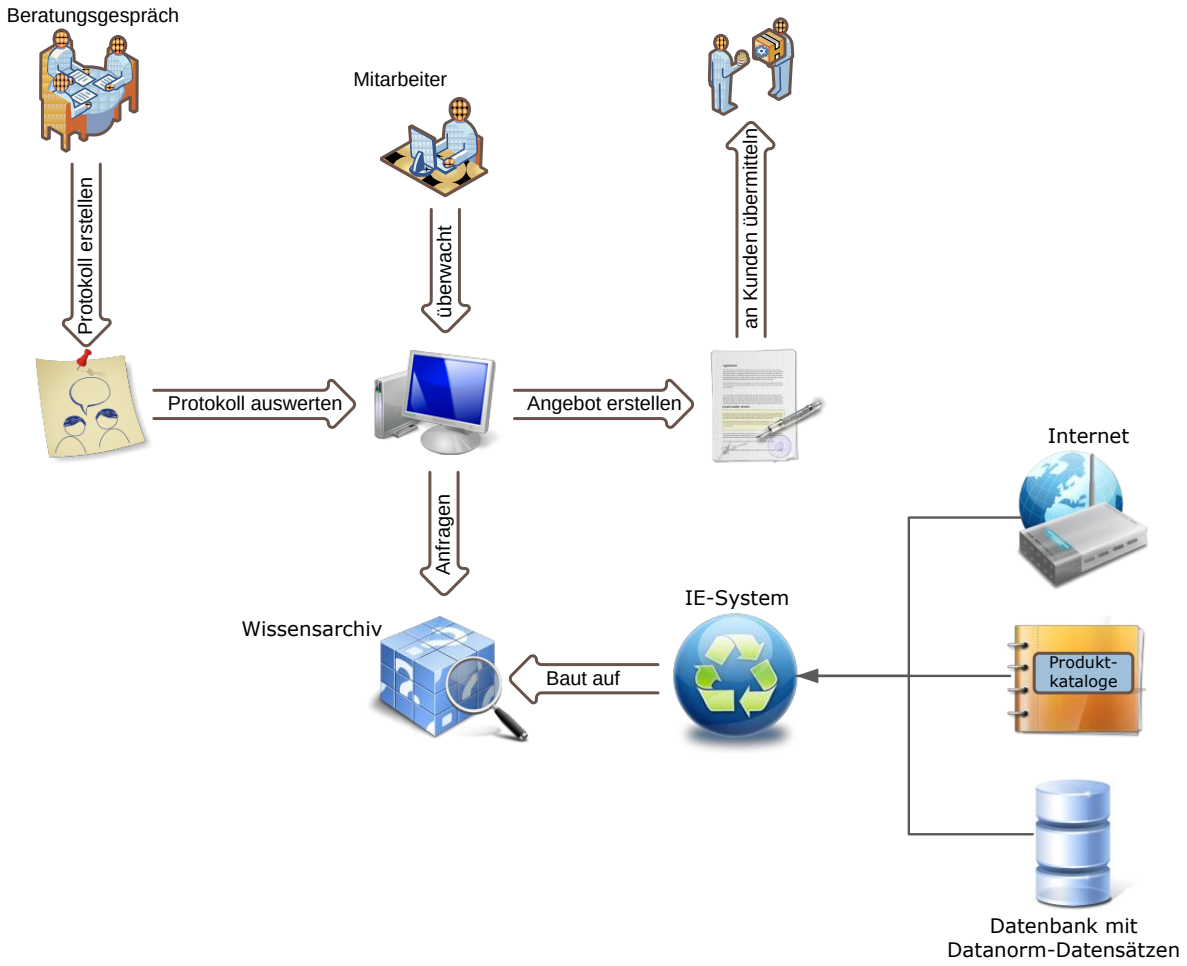


Abbildung 14: Mögliche Optimierung des Referenzprozesses

Am Anfang der Angebotserstellung steht nach wie vor das Beratungsgespräch mit dem Kunden. Während des Beratungsgesprächs wird ein Protokoll angefertigt, in dem die Wünsche und Vorstellungen des Kunden festgehalten werden. Eine handschriftliche Anfertigung des Protokolls würde Probleme bereiten, da die Software dieses einlesen und interpretieren müsste. Ein elektronisches Protokoll würde sich wesentlich besser eignen. Es war ebenfalls zu überlegen, ob die Software beim Beratungsgespräch zum Einsatz kommt und entsprechende Produktwünsche direkt in eine Eingabemaske eingetragen werden. Diese würde den Umweg über das Gesprächsprotokoll überflüssig machen.

Das Wissensarchiv der Firma Schottler wurde mit Hilfe des IE-System aus den Informationen aus Internet, Produktkatalogen und den Datenbankbeständen angereichert. Zuvor musste der Mitarbeiter manuell in diesen Informationsquellen nachschlagen, diese Aufgabe übernimmt nun die automatische Extraktion. Im Wissensarchiv werden die extrahierten Informationen mit Hilfe von Ontologien, die speziell auf den Bereich der Angebotserstellung ausgelegt sind, abgespeichert. Der Mitarbeiter kann Anfragen an das Wissensarchiv stellen, um dadurch ein optimales Angebot für den Kunden erstellen zu können. Konkrete Produkte (z.B. Kaldewei Mega Duo 6) werden im Archiv recherchiert und in das Angebot direkt aufgenommen.

Hat der Kunde keine konkrete Produktvorstellung, sondern nur Rahmenbedingungen angegeben (z.B. Badewanne max. 2 m lang und in Weiß) werden Produkte vorgeschlagen, die diese Anforderungen erfüllen. Zubehörteile, die zu einem Produkt zwingend dazu

gehören, sind in der Ontologie ebenfalls modelliert und werden automatisch dem Angebot hinzugefügt.

Da die Angebotserstellung ein sensibler Bereich ist, hat der Mitarbeiter jederzeit Einfluss auf die automatisch generierten Ergebnisse. Er kann diesen Prozess überwachen und abschließend notwendige Änderungen vornehmen. Dazu zählen zum Beispiel auch die Gewährung von kundenspezifischen Sonderrabatten usw., die von dem Wissensarchiv in der Form nicht berücksichtigt werden können.

1.3 Interaktionsmodell für die Benutzerschnittstelle

Die Funktionsweise und der Funktionsumfang der Benutzerschnittstelle werden (siehe Abbildung 15) in diesem Abschnitt beschrieben. Dies umfasst die Definition, wie und welche Vorschläge z. B. der intelligente Benutzerinteragent dem Benutzer macht. Es beschreibt aber auch, wie z.B. neue Informationen in das System eingefügt werden (automatisch oder nur nach Verifikation durch einen Benutzer oder einen Administrator), oder wie auf welche Information zugegriffen wird.

Für die Implementierung des intelligenten Benutzeragenten war es erforderlich, die Funktionsweise und den Funktionsumfang der Benutzerschnittstelle zu beschreiben. Dabei handelt es sich um die Möglichkeiten für den Benutzer neue, eingehende Informationen zu verarbeiten und dem System hinzuzufügen, als auch um die Vorschläge, die der Agent dem Benutzer für einen bestimmten Prozessschritt macht.

Letztendlich soll der Benutzer die neuen Informationen in Form von Dokumenten oder E-Mails mit möglichst wenig Interaktion ins System einbringen können. Dazu wurden Plugins für die beiden gängigen E-Mail Programme Outlook und Mozilla Thunderbird entwickelt, die es dem Nutzer ermöglichen eine ausgewählte E-Mail in das Wissensarchiv einzubringen. Sollte es sich hierbei um eine E-Mail mit einem angehängtem Dokument handeln, dann wird dieser Anhang als eigenes Objekt ins Wissensarchiv aufgenommen und dem E-Mail-Objekt im Wissensarchiv eine Verknüpfung auf das Dokument-Objekt mitgegeben.

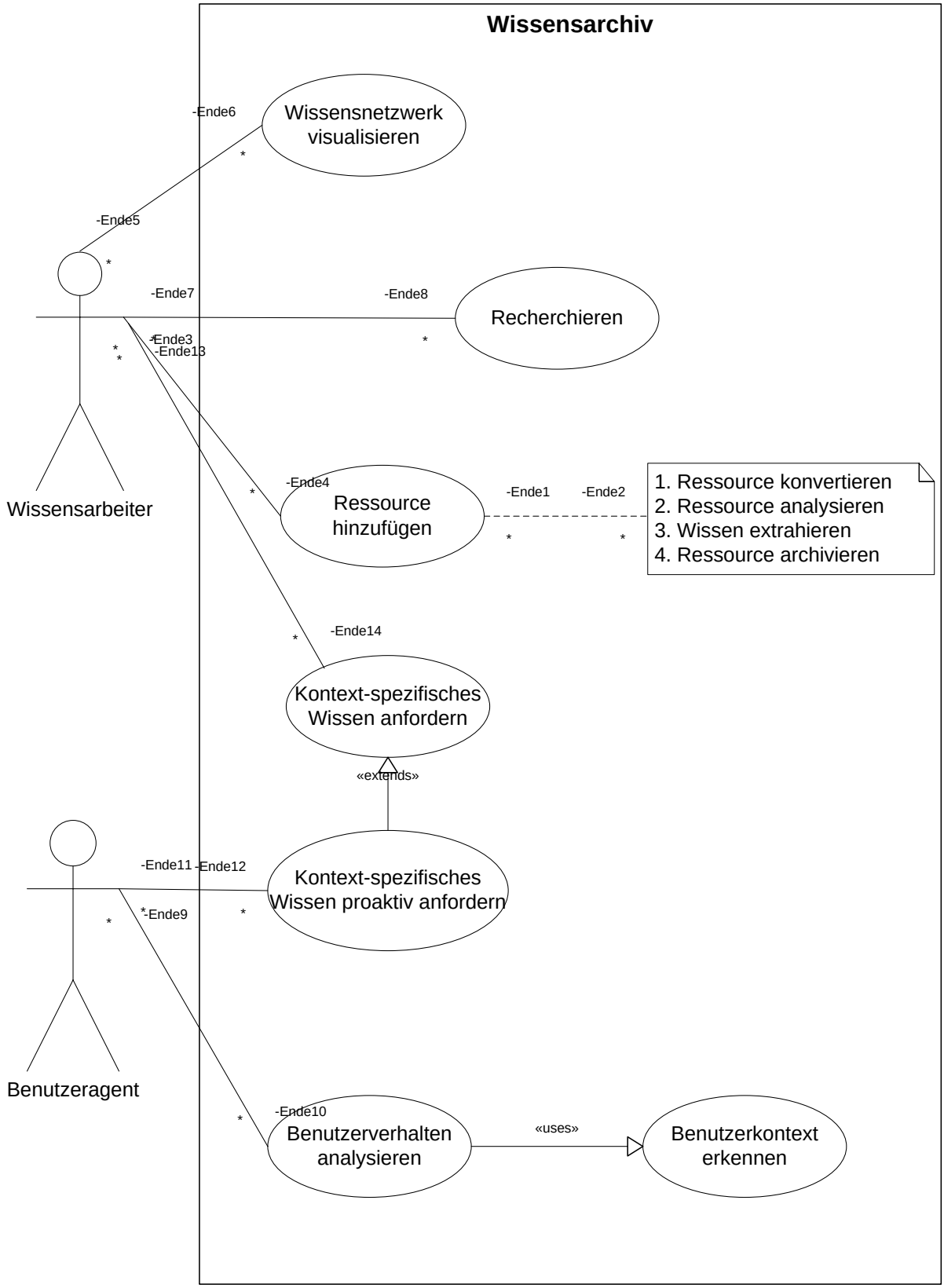


Abbildung 15: Interaktionsmodell der Benutzerschnittstelle

2.2 *Arbeitspaket 2: Aufbau der Ontologie*

2.2.1 Untersuchung Beschreibungsformen für Ontologien

Zur Unterstützung der Informationsextraktion werden häufig Ontologien eingesetzt, die zusätzliches Wissen bezüglich der relevanten Domäne zur Verfügung stellen. Der Begriff der Ontologie wird im folgenden Unterkapitel herausgearbeitet. In den letzten Jahren haben sich mehrere Ontologiesprachen entwickelt. Abschnitt 2.2.1.2 und Abschnitt 2.2.1.3 stellen die beiden bekanntesten Web-Ontologiesprachen Ressource Description Framework und Web Ontology Language vor. Im letzen Kapitel werden die Erkenntnisse zusammengefasst und die Entscheidung für die Auswahl der Web Ontology Language begründet.

2.2.1.1 Ontologien zur Wissensmodellierung

Der Begriff der Ontologie hat seinen Ursprung in der Philosophie und umschreibt ein Forschungsgebiet der Metaphysik, das sich mit der Natur und der Existenz des Seins beschäftigt. In der Informatik versteht man unter Ontologie eine Beschreibung eines Wissensbereich mit Hilfe einer festgelegten Terminologie, sowie Relationen und Regeln zwischen den definierten Begriffen [Sch05].

Ontologien werden in der Informatik schon seit Jahrzehnten in vielen Anwendungsbereichen (Sprachverarbeitung, Information Retrieval, Dokumentenanalyse, Semantic Web uvm.) eingesetzt. In allen Bereichen spielt der Austausch bzw. die Verarbeitung von Wissen eine zentrale Rolle. Die Kommunikation ist dabei das Mittel, mit der zwei oder mehrere Parteien - seien es Menschen oder Maschinen - ihr Wissen austauschen können. Um sicherzustellen dass die Parteien nicht „aneinander vorbeireden“, reicht es meistens nicht aus über ein gemeinsames Vokabular an Begriffen zu verfügen. Vielmehr muss sichergestellt sein, dass bestimmte Worte nicht nur bekannt, sondern auch überall die gleiche Bedeutung haben.

Im sogenannten semiotischen Dreieck wird der Zusammenhang zwischen Syntax und Semantik dargestellt. Erst wenn Syntax und Semantik in Beziehung zueinander gesetzt werden, ist es möglich terminologische Unklarheiten zu beheben. Inwieweit diese Beziehung hergestellt werden kann, hängt vom Hintergrundwissen, welches der jeweiligen Partei zur Verfügung steht, ab.

Unterhält man sich über den Begriff Golf, so ist der Kontext in dem man sich befindet, von Relevanz. Ist die Domäne der Kraftfahrzeuge aktuelles Gesprächsthema, so können Menschen auf Grund ihres Wissens ableiten, dass der Begriff Golf mit großer Wahrscheinlichkeit auf den Namen eines Automodells einer deutschen Firma zurück zu führen ist. Werden in der Kommunikation jedoch überwiegend Begriffe aus dem Sportbereich verwendet, so ist die Interpretation des Begriffs Golf in diesem Kontext eher auf die Sportart ausgerichtet.

Dieses Hintergrundwissen, welches Menschen implizit verwenden um logische Schlussfolgerungen ziehen zu können, muss für den Computer in einer Art modelliert werden, die er verarbeiten kann. Diese semantischen Modelle zur Wissensmodellierung erlauben eine formale und interpretierbare Repräsentation von Wissen. Es existieren verschiedene Modelle, die in Abbildung 16 dargestellt werden. Die Mächtigkeit und semantische Reichhaltigkeit der Ansätze nimmt mit steigender Reihenfolge zu.

- **Glossar**:
  Ist eine Liste von Begriffen mit zugehörigen Erklärungen. Es existieren keine Beziehungen zu anderen Begriffen.
- **Taxonomie**:
  Ist eine Hierarchie von Begriffen, die die Elemente in eine Beziehung Ober-/Unterbegriff setzt. Eine bekannte Anwendung, die das Prinzip der Taxonomie anwendet, ist das Dateisystem von Betriebssystemen.
- **Thesaurus**:
  Stammt aus dem Bibliothekarswesen und stellt ein Begriffsnetzwerk zu einer bestimmten Domäne zur Verfügung. Neben der Taxonomie existieren zusätzlich zwei weitere fest definierte Relationen: die Ähnlichkeits- und Synonymrelation.
- **Topic-Map**:
  Ist ein ISO-Standard auf XML-Basis. Eine Topic Map besteht aus Topics (abstrakten Begriffen), Assoziationen, Scopes (Gültigkeitsbereiche für die Topics) und zugeordneten Dokumenten, die sich außerhalb der Topic Map befinden. Die Assoziationen zwischen den Topics lassen sich selbst definieren.
- **Ontologie**:
  Die definierten Begriffe, häufig auch als Klassen oder Frames bezeichnet, werden mit Hilfe von Vererbungen und selbstdefinierten Relationen zu einem Begriffsnetzwerk verbunden. Jeder Klasse können zusätzlich Attribute (Slots) zugeordnet werden. Werden die Attribute einer Klasse mit konkreten Werten gefüllt, so spricht man von einer Instanz. Dies sind vergleichbare Paradigmen, wie sie aus der objekt-orientierten Programmierung bekannt sind. Darüber hinaus verfügen Ontologien über Inferenz- und Integritätsregeln, die es ermöglichen Schlussfolgerungen zu machen und deren Gültigkeit zu gewährleisten.

Nach [MNV02] kann man drei Arten von Ontologien unterscheiden (siehe Abbildung 17): (1) Die allgemeinen, bereichsübergreifenden Top Level Ontologies beschreiben lediglich sehr allgemeine Konzepte wie z.B. Zeit oder Raum, unabhängig von einer speziellen Domäne oder Problemstellung. Sie sind anwendbar auf jede in der Hierarchie tiefer liegende Ontologie. (2) Ein grundlegendes Vokabular zu einem allgemeinen Anwendungsgebiet (z.B. der Medizin), wird in den Upper Domain Ontologies definiert. (3) Letztlich werden in den Specific Domain Ontologies spezielle, auf einen Problembereich zugeschnittene, Ontologien definiert (z.B. Medikamente in der Krebsforschung), die in der Regel eine Upper Domain Ontology verwenden und diese um neue Konzepte erweitern.

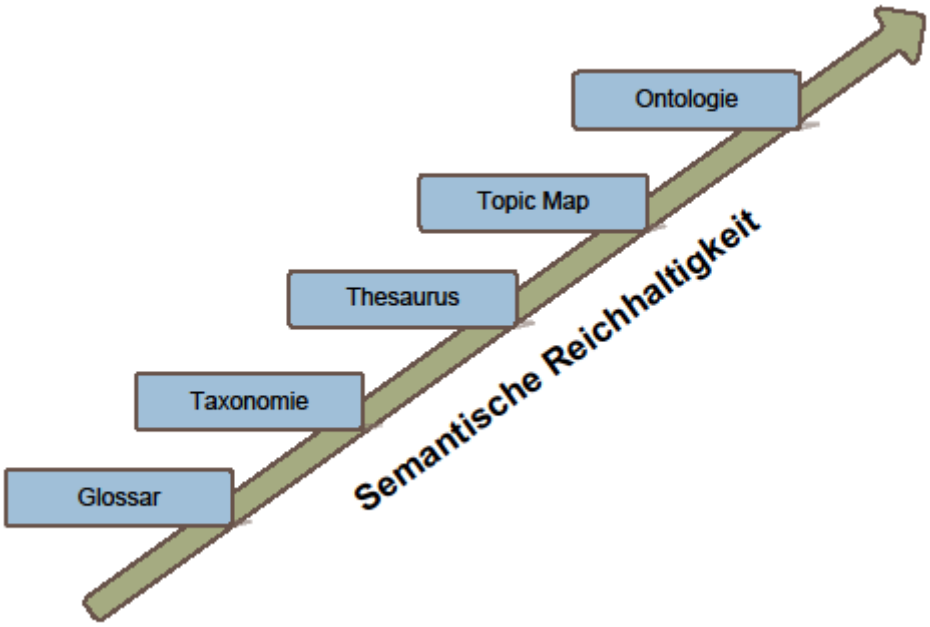


Abbildung 16: Semantische Modelle zur Wissensrepräsentation

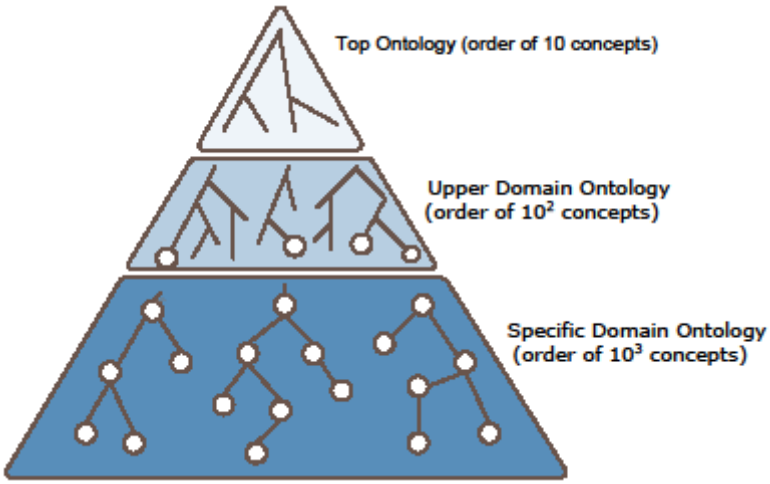


Abbildung 17: Kategorien von Ontologien nach [MNV02]

2.2.1.2 Resource Description Framework

Das Resource Description Framework (RDF) stellt Mittel zur Verfügung, um Metadaten zu modellieren. Metadaten werden häufig verwendet, um beschreibende Informationen über eine Ressource festzuhalten. RDF ist ein Standard des World Wide Web Consortium (W3C), der am 22.02.1999 verabschiedet wurde. Die Entwicklung von RDF zielt darauf ab, Informationen über Internetseiten - im Sinne des Semantic Web - maschinenlesbar bereit zu stellen.

RDF-Datenmodell

Das RDF-Datenmodell basiert auf RDF-Konstrukten die als Subjekt-Prädikat-Objekt Tripel dargestellt werden. Diese symbolisieren eine Beziehung zwischen zwei Ressourcen mittels

eines beschreibenden Prädikats (z.B. Merkel-regiert-Deutschland). Im Allgemeinen werden diese Beziehungen in folgender Notation definiert: `regiert(Merkel,Deutschland)` Das Subjekt S stellt die Ressource dar, über die nähere Informationen erfasst werden sollen, indem über ein Prädikat P eine Beziehung zu einem Objekt O hergestellt wird. Objekte können wiederrum Ressourcen oder Literale sein. So werden im folgendem Tripel `regiert(Merkel, Deutschland)` zwei Ressourcen in Beziehung gesetzt und im Tripel `gegründetAm(Deutschland, "23.05.1949")` eine Ressource und ein Literal.

Ressourcen müssen immer eindeutig zu identifizieren sein. Dies ist vor allem von Bedeutung, wenn verschiedene RDF-Ontologien miteinander kombiniert werden. Ansonsten könnten Ressourcen mit gleicher Bezeichnung nicht mehr unterschieden werden. Aus diesem Grund wird, vergleichbar mit dem Konzept der Namespaces aus der Extended Markup Language (XML), jede Ressource mit einem Universal Resource Identifier (URI) versehen.

RDF-Graphen (siehe Abbildung 18 (1)) bieten die Möglichkeit RDF-Konstrukte in visueller Form als gerichteten Graph darzustellen. Subjekte und Objekte werden durch Knoten repräsentiert und Kanten stehen für Prädikate. Daneben lassen sich RDF-Tripel in einer XML-basierten Syntax serialisieren. Dieses RDF/XML-Format (siehe Abbildung 18 (2)) ist die verbreiteste Form um RDF-Informationen auszutauschen [SET09]. Die Elemente `regiert` und `gegründetAm` gehören nicht zum Vokabular von RDF, sondern werden außerhalb - unter dem angegebenem Namespace - von RDF definiert.

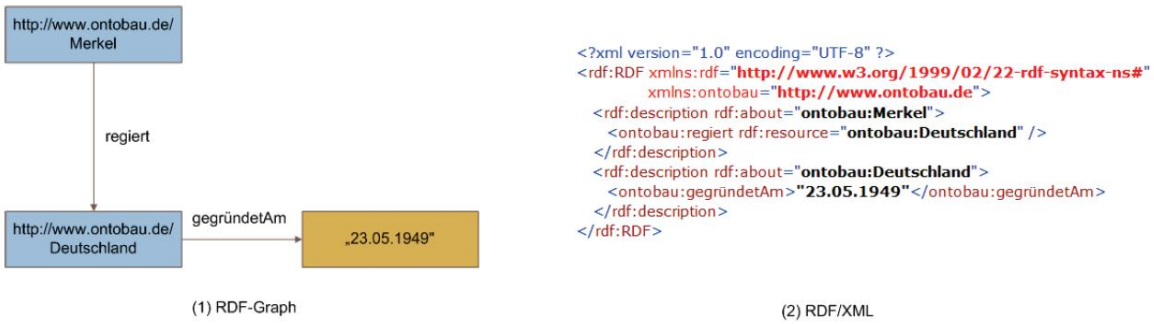


Abbildung 18: Verschiedene Serialisierungsmöglichkeiten von RDF

RDF-Schema

Da RDF zur Repräsentation von Metadaten gedacht ist, ist die Mächtigkeit zur Modellierung von Ontologien nicht ausreichend genug. Konzepte wie Klassen, Eigenschaften, Vererbung, Instanziierung etc. fehlen. Das RDF-Schema (RDFS) erweitert RDF um diese Konzepte, indem das dafür notwendige Vokabular zur Verfügung gestellt wird. Diese Erweiterungen ähneln stark den aus objekt-orientierten Programmiersprachen wie Java und C++ bekannten Spezifikationen. Abbildung 19: Das Kernvokabular von RDFS zur Spezifizierung von Klassen und Properties (Quelle: [BG98]). zeigt einen Auszug aus der RDFS-Spezifikation, von denen stellvertretend einige näher beschrieben werden:

- rdfs:Class
  Dient zur Spezifikation von Klassen. In Zusammenhang mit rdf:type können Instanzen einer Klasse erzeugt werden.

- rdf:Property
  Definiert eine Eigenschaft, die zwei Ressourcen über den Definitions- und Wertebereich der Eigenschaft in Beziehung setzt.
- rdfs:subClassOf
  Ist eine vordefinierte Property zur Definition einer Spezialisierungsbeziehung zweier Klassen. Werte- und Definitionsbereich ist rdfs:class.
- rdfs:range
  Ist eine vordefinierte Property zur Einschränkung des Wertebereichs einer Property. Die Werte von rdfs:range müssen Klassen oder Datentypen sein.
- rdfs:domain
  Analog zu rdfs:range, nur dass der Definitionsbereich der Property eingeschränkt wird.

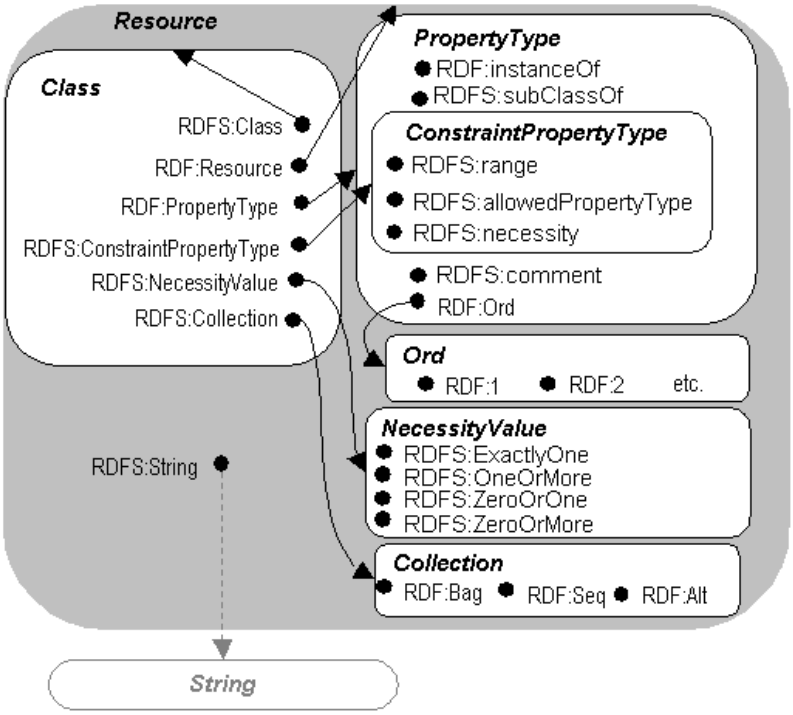


Abbildung 19: Das Kernvokabular von RDFS zur Spezifizierung von Klassen und Properties (Quelle: [BG98]).

Mit Hilfe des zusätzlichen Vokabulars von RDFS ist es möglich, das Beispiel aus Abbildung 18 (2) vollständig in einer RDFS-Ontologie zu spezifizieren (siehe Abbildung 20) . Dazu werden zwei Klassen zur Modellierung von Ländern und Regierungschefs definiert. Die Klasse Regierungschef wird um die Eigenschaft `regiert` erweitert und die Klasse Land entsprechend um eine Property `gegründetAm`. Beide Properties werden in ihrem Werte- und Definitionsbereich eingeschränkt. Zum Beispiel die Property regiert, sodass sie nur Instanzen der Klassen Land und Regierungschef in Beziehung setzen kann. Anschließend werden jeweils eine Instanz der Klasse Land und eine Instanz der Klasse Regierungschef erzeugt. In RDF existieren verschiedene Möglichkeiten der Instanziierung. Zwei dieser Varianten sind in dem Beispiel dargestellt. In der ersten Variante wird eine Instanz über eine Typreferenzierung mittels rdf:type erzeugt, während in der zweiten Variante eine Instanz direkt mittels des Klassenelements (hier Land) generiert wird.

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <rdfs:class rdf:ID="Regierungschef" />
    <rdfs:class rdf:ID="Land" />
  - <rdf:Property rdf:ID="regiert">                    >
      <rdfs:domain rdf:resource="#Regierungschef" /
      <rdfs:range rdf:resource="#Land" />
    </rdf:Property>
  - <rdf:Property rdf:ID="gegründetAm">
      <rdfs:domain rdf:resource="#Land" />
      <rdfs:range rdf:resource="rdfs:Literal" />
    </rdf:Property>
  - <rdf:description rdf:ID="Merkel">
      <rdf:type rdf:resource="#Regierungschef" />
      <regiert rdf:resource="#Deutschland" />
    </rdf:description>
  - <Land rdf:ID="Deutschland">
      <gegründetAm>23.05.1949</gegründetAm>
    </Land>
  </rdf:RDF>
```

Abbildung 20: Beispielontologie in RDFS zur Modellierung von Ländern und Regierungschefs

Weiterführende Informationen zum Thema Ontologiemodellierung mit RDF bzw. RDFS können u.a. in [SET09], [AH08a] und [Pow03] nachgelesen werden.

Schwächen von RDF

Durch die Erweiterungen des RDF-Schemas hat RDF zwar die notwendigen Konstrukte zur Verfügung, um logische Zusammenhänge zwischen Klassen, Instanzen und Beziehungen zu spezifizieren, was jedoch nicht ausreicht, um komplexere Zusammenhänge darzustellen. Aussagen wie „Bars und Restaurants gehören zur Gastronomie, haben aber keine gemeinsamen Eigenschaften“ oder Schlussfolgerungen wie „Wenn Klaus Vater von Hans ist, dann ist Hans Sohn von Klaus“ lassen sich mit RDF nicht abbilden.

Zusammenfassend können u.a. folgende Schwachstellen von RDF/RDFS genannt werden:

- Properties lassen sich in ihrem Werte- und Definitionsbereich einschränken. Diese Einschränkungen sind jedoch global gültig, sodass die gleiche Eigenschaft für verschiedene Klassen nicht definiert werden kann.
- Boolesche Kombinationen von mehreren Klassen bezüglich der Vereinigung, Schnittmenge und Komplement sind nicht möglich.
- Properties können nicht mit Kardinalitäten eingeschränkt werden
- Es können keine Eigenschaften von Properties wie Transitivität, Symmetrie, Invertierbarkeit oder Einzigartigkeit abgebildet werden.
- Es kann nicht angegeben werden, dass Klassen disjunkt sind.

2.2.1.3 Web Ontology Language

Die Web Ontology Language (OWL) basiert auf der RDF-Syntax und ist der Nachfolger der Ontologiesprachen Darpa Agent Markup Language (DAML) und Ontology Inference Layer (OIL). Im November 2002 bildete das W3C eine Arbeitsgruppe mit dem Ziel eine standardisierte Ontologiesprache zu entwickeln, die über die Mächtigkeit von RDF-Schema hinausgeht und die Bedürfnisse des Semantic Web erfüllt. Im Februar 2004 wurde OWL als W3C Empfehlung verabschiedet.

OWL-Sprachstufen

Aktuell verfügt OWL über drei verschiedene Sprachstufen, die in ihrer Komplexität zunehmen. Jede untere Stufe ist Teilsprache der nächsthöheren Stufe. OWL Lite ist die Sprache mit dem geringsten Umfang an Sprachkonstrukten. Boolesche Kombinationen und Kardinalitätsbeschränkungen von Properties sind in dieser Stufe u.a. nicht erlaubt. Dies ermöglicht ein effizientes Auswerten der modellierten Ontologien. Die nächsthöhere Sprache ist OWL DL. Das Akronym DL steht für Description Logic und bedeutet, dass die Anforderungen der Description Logic erfüllt sind. Somit gehört OWL DL zu einer entscheidbaren Untermenge der Prädikatenlogik. Die Sprachstufe verwendet alle Konzepte, jedoch sind Kardinalitäten von transitiven Properties und deren Inversen nicht erlaubt. Dies garantiert die rechnerische Vollständigkeit (alle möglichen Schlussfolgerungen können berechnet werden) und Entscheidbarkeit (alle Berechnungen von Schlussfolgerungen enden in endlicher Zeit), wobei die Komplexität in speziellen Fällen NP-vollständig sein kann. OWL Full bietet die uneingeschränkte Verwendung aller Konzepte von OWL, aber ohne Garantie für die Berechenbarkeit.

OWL-Sprachelemente

OWL baut auf der RDF/RDFS-Syntax auf und bezieht bereits definierte Elemente dieser Sprache weitestgehend mit ein. An dieser Stelle sollen überwiegend die Konstrukte vorgestellt werden, die die angesprochenen Schwächen von RDFS beheben:

- owl:disjointWith
  Ermöglicht die Aussage, dass zwei Klassen disjunkt sind.
- owl:equivalentClass
  Ermöglicht die Äquivalenz von zwei Klassen zu definieren.
- owl:unionOf
  Dadurch kann eine neue Klasse als Vereinigung von zwei bereits vorhandenen Klassen definiert werden. Analog dazu können auch owl:intersectionOf und owl:complementOf verwendet werden.
- owl:ObjectProperty
  Definiert eine Relation zwischen den Instanzen zweier Klassen.
- owl:DatatypeProperty
  Definiert eine Relation zwischen der Instanz einer Klasse und einem Datentyp. Da OWL keine eigenen Datentyp-Spezifikationen definiert, werden die Datentypen des XML-Schemas eingesetzt.
- owl:domain
  Definiert den Definitionsbereich einer Property. Im Gegensatz zu RDFS können jedoch für eine Property mehrere Definitionsbereiche angegeben werden. owl:range wird analog verwendet.
- owl:transitiveProperty
  Ermöglicht Aussagen über die transitiven Eigenschaften einer definierten Relation zu treffen. owl:symmetricProperty kann analog verwendet werden.
- owl:inverseOf
  Definiert eine Property als Inverse einer anderen Property. Dadurch wird ermöglicht, Folgerungen wie „Wenn Klaus Vater von Hans ist, dann ist Hans Sohn von Klaus“ zu modellieren.
- owl:functionalProperty
  Definiert eine Property, die für jede Instanz nur einmal auftreten darf.

- owl:maxCardinality
  Beschränkt den Wertebereich einer Property auf eine maximale Anzahl an unterschiedlichen Instanzen, zu denen eine Relation bestehen darf (analog owl:minCardinality und owl:Cardinality). Ohne eine Einschränkung der Kardinalität können beliebig viele Relationen bestehen. In OWL Lite sind Kardinalitäten nur mit den Werten 0 und 1 erlaubt.

Dies stellt nur einen Auszug aus den neuen Sprachkonstrukten dar, die OWL für die Modellierung von Ontologien zur Verfügung stellt. Jedoch sollte erkennbar sein, dass die Mächtigkeit von OWL die von RDFS übersteigt und dadurch die Modellierung komplexerer Ontologien erlaubt. Weiterführende Informationen können in [AH08a] und [HKRS07] gefunden werden.

2.2.1.4 Auswahl einer Ontologiesprache

Zur Modellierung der Ontologien in diesem Forschungsvorhaben standen verschiedene Ontologiesprachen zur Auswahl, von denen RDFS und OWL bereits in den vorherigen Kapiteln vorgestellt wurden. Zudem wäre es noch möglich gewesen, eine eigens entwickelte Ontologiesprache einzusetzen. Diese wurde in früheren Forschungsprojekten im Bereich der Informationsextraktion eingesetzt und an der Fachhochschule Trier in mehreren Studentenprojekten erstellt. Die Ontologiesprache orientiert sich an dem FRAMES-Konzept und wurde in LISP modelliert.

Der Einsatz der eigens entwickelten Ontologiesprache hätte den Vorteil, dass man selbst Kontrolle über die Mächtigkeit der Sprache hätte. Es könnten Anpassungen vorgenommen werden, die speziell den Anforderungen innerhalb des Forschungsprojektes gerecht werden. Sollten die Kernkomponenten des IE-Systems in der Programmiersprache LISP realisiert werden, so wäre die Interoperabilität ohne Anpassung gewährleistet. Die Nachteile beständen darin, dass man mögliche vorhandene Ontologien, Tools etc. nicht oder zumindest nur mit größerem Aufwand einsetzen könnte. In der Open Source Gemeinde existieren jedoch bereits zahlreiche Ontologien und Tools (wie z.B. Ontologie-Editoren, Reasoner uvm.), die überwiegend auf den hier vorgestellten Ontologiesprachen basieren. Die Vorteile, die sich durch den Einsatz einer standardisierten Ontologiesprache mit dazu passenden Tools ergeben, waren ausschlaggebend warum die Entwicklung einer eigenen Ontologiesprache zur Repräsentation des Wissensarchivs abgelehnt wurde.

Die Nachteile von RDFS wurden bereits in Abschnitt 2.2.1.2 angesprochen. Reichen diese Sprachkonstrukte zum Modellieren von Taxonomien und Thesauren noch aus, so können Ontologien damit lediglich im kleinen Rahmen erstellt werden. Ein Reasoning Support, um aus vorhandenen Informationen Schlussfolgerungen zu ziehen und damit neues Wissen abzuleiten, ist nur bedingt möglich. Dies liegt darin begründet, dass u.a. keine näheren Aussagen über Relationen erfasst werden können. Dadurch können keine ausreichenden Inferenzregeln modelliert werden, die ein logisches Schließen erfordert. Folgende Fakten als Prämisse „Alle Griechen sind sterblich“ und „Sokrates ist ein Grieche“ dargestellt, könnten in RDFS nicht zu der Konklusion führen, dass das Faktum „Sokrates ist sterblich“ abgleitet werden kann.

OWL behebt viele dieser Schwächen durch die Erweiterung der RDFS-Spezifikation um neue Sprachkonstrukte, die den Anforderungen der Beschreibungslogik gerecht werden. Beschränkt man sich auf OWL DL Ontologien, so ist ein effizientes Reasoning möglich. Die

o.g. Aussagen ließen sich in OWL wie in Abbildung 21 dargestellt modellieren. Zusätzlich stehen mit dem Fast Classification of Terminologies (FaCT) und der Karlsruhe Ontology management infrastructure (KAON2) bereits - zumindest im Bereich der Forschung und Lehre - kostenlose Reasoner für OWL zur Verfügung.

```
<owl:Class rdf:ID="Grieche">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue>
        <Sterblichkeit rdf:ID="Sterblich" />
      </owl:hasValue>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hat_Lebensspanne" />
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<Grieche rdf:ID="Sokrates" />
```

Abbildung 21: OWL-Beispiel für die Modellierung der Aussagen "Alle Griechen sind sterblich" und "Sokrates ist ein Grieche"

Dennoch bestehen auch in OWL Schwächen, die hier nachträglich noch angesprochen werden sollen:

- Werte-Einschränkung von Datentypen:
  In OWL ist es standardmäßig nicht möglich, numerische Datentypen in ihrem Wert einzuschränken. Eine Aussage wie „Ein Bachelorstudium hat maximal 8 Semester Regelstudienzeit“ ist nicht möglich.
- Arithmetische Zusammenhänge:
  In OWL können keine arithmetischen Zusammenhänge, wie zum Beispiel, dass die Property BruttoPreis dem Produkt von NettoPreis und Mehrwertsteuer entspricht, ausgedrückt werden.
- Prädikatenlogische Regeln:
  Außer den von OWL bereitgestellten Konstrukten, können keine Regeln beschrieben werden, die Aussagen wie „Wenn X Bruder von Y und X Vater Z, dann gilt auch Y Onkel von Z“, ermöglichen.

Diese angesprochenen Nachteile wurden jedoch bereits in den letzten Jahren durch Erweiterungen behoben. So kann mit Hilfe von OWL-EU (OWL with unary datatype Expressions) eine Einschränkung von Datentypen erreicht werden (vgl. [PH06]). Womit einer der Schwachstellen abgedeckt wäre.

Nach [BS03] ist das Erweitern einer Ontologiesprache um Aggregatfunktionen nicht möglich, wenn die Entscheidbarkeit von OWL DL garantiert werden soll. [Poh06] schlägt vor, dass man OWL um arithmetische Funktionen erweitert, die vor dem Reasoning ausgewertet werden, was die Entscheidbarkeit von OWL DL unberührt lässt. Dadurch wäre es möglich, auch in OWL mit arithmetischen Funktionen zu arbeiten.

Letztlich wurde mit der Semantic Web Rule Language (SWRL) eine Sprache entwickelt, die es ermöglicht Horn-Klauseln in OWL einzubinden. Abbildung 22 zeigt ein Beispiel einer SWRL-Regel, die die Aussage "‘Wenn X Bruder von Y und X Vater Z, dann gilt auch Y Onkel von Z“ modelliert.

```
<owlx:Rule>
  <owlx:antecedent>
    <owlx:individualPropertyAtom owlx:property="hasParent">
      <owlx:Variable owlx:name="x1" />
      <owlx:Variable owlx:name="x2" />
    </owlx:individualPropertyAtom>
    <owlx:individualPropertyAtom owlx:property="hasBrother">
      <owlx:Variable owlx:name="x2" />
      <owlx:Variable owlx:name="x3" />
    </owlx:individualPropertyAtom>
  </owlx:antecedent>
  <owlx:consequent>
    <owlx:individualPropertyAtom owlx:property="hasUncle">
      <owlx:Variable owlx:name="x1" />
      <owlx:Variable owlx:name="x3" />
    </owlx:individualPropertyAtom>
  </owlx:consequent>
</owlx:Rule>
```

Abbildung 22: Modellierung von Regeln mit SWRL

Jedoch ist OWL in Zusammenarbeit mit SWRL nicht mehr entscheidbar. Dieser Umstand lässt sich beheben, indem SWRL-Regeln die DL-safeness erfüllen. DL-Safeness wird erreicht, indem Regeln auf bekannte Individuen eingeschränkt werden. Die Aussage

$$\texttt{istBruder(X,Y)} \wedge \texttt{istVater(X,Z)} \rightarrow \texttt{istOnkel(Y,Z)}$$

muss erweitert werden zu

$$\texttt{istBruder(X,Y)} \wedge \texttt{istVater(X,Z)} \wedge \texttt{O(X)} \wedge \texttt{O(Y)} \wedge \texttt{O(Z)} \rightarrow \texttt{istOnkel(Y,Z)}$$

damit eine Entscheidbarkeit möglich ist und die Fakten O(X), O(Y) und O(Z) müssen für die Individuen in der Ontologie angelegt sein (vgl. [HKRS07]).

OWL ist von den derzeitigen Ontologiesprachen diejenige mit der man in der Lage ist Ontologien mit nahezu beliebiger semantischer Reichhaltigkeit zu modellieren. Durch die unterschiedlichen Sprachebenen, ist man in der Lage, anhand dieser bezüglich der Notwendigkeit einer Reasoning Unterstützung zu skalieren. Es existieren bereits einige Tools, die die Erstellung, Verarbeitung und das Reasoning unterstützen. Die oben genannten Schwachstellen von OWL können bei Bedarf durch entsprechende Erweiterungen behoben werden. Zudem wurde im Oktober 2009 der Nachfolger OWL 2 vom W3C als Standard verabschiedet. Letztlich basiert OWL auf einer XML-basierten Syntax, was die Möglichkeit bietet, dass OWL-Ontologien in beliebigen Programmiersprachen mit Hilfe entsprechender XML-Parser eingelesen werden können.

Diese Überlegungen führten dazu, dass OWL als Ontologiesprache gewählt wurde. Leider bietet OWL keine Möglichkeiten die angesprochenen Anforderungen zur Beschreibung von Klassifikationskonzepten und Templates abzubilden. OWL kann demnach nur zur Modellierung des Wissensarchivs eingesetzt werden.

Aufgrund der Aktualität war es nicht möglich, sich genauer mit den Neuerungen des Nachfolgers OWL 2 zu beschäftigen. Erste Tools wurden jedoch bereits für die Zusammenarbeit mit OWL 2 angepasst, so dass eine nähere Betrachtung der Sprache in naher Zukunft sinnvoll ist.

2.2.2 Aufbau einer Ontologie Bauwesen

Nach der Auswahl eines Beschreibungsformalismus wurden Ontologien aufgebaut, die den Anwendungsbereich mindestens in den notwendigen Bereichen für die ausgewählten Dokumente abdeckt. Hierzu wurden Taxonomien des Projektpartners Mindox GmbH zur Klassifikation herangezogen und in das Forschungsvorhaben integriert. Zusätzlich wurden OWL-Ontologien speziell für den Heizungs- und Sanitärbereich der Firma Schottler entwickelt und eine allgemeine Ontologie zur Repräsentation von beliebigen Dokumenten, die auf alle beteiligten Projektpartner zugeschnitten ist. Diese Ontologien werden in den nächsten Abschnitten näher beschrieben.

2.2.2.1 Taxonomien zur Klassifikation im Bauwesen

Aus der kompletten Taxonomie wurden dabei die Bereich „Bauwesen Allgemein - Bauwesentechnik“ (siehe Abbildung 23) und „Bauwesen - Fachbereiche Heizungs- und Sanitärtechnik“ (siehe Abbildung 24) ausgewählt, die auf die Anforderungen unserer Projektpartner am passendsten erschienen.

Die Taxonomien der Firma Mindox GmbH sind in einer eigens entwickelten Wissensrepräsentationssprache modelliert, auf die an dieser Stelle nicht weiter eingegangen werden kann. Aufgrund der starken Ähnlichkeit zur eingesetzten Realisierung des Gesamtsystems ließen sich diese jedoch ohne größeren Aufwand in das Gesamtsystem integrieren.

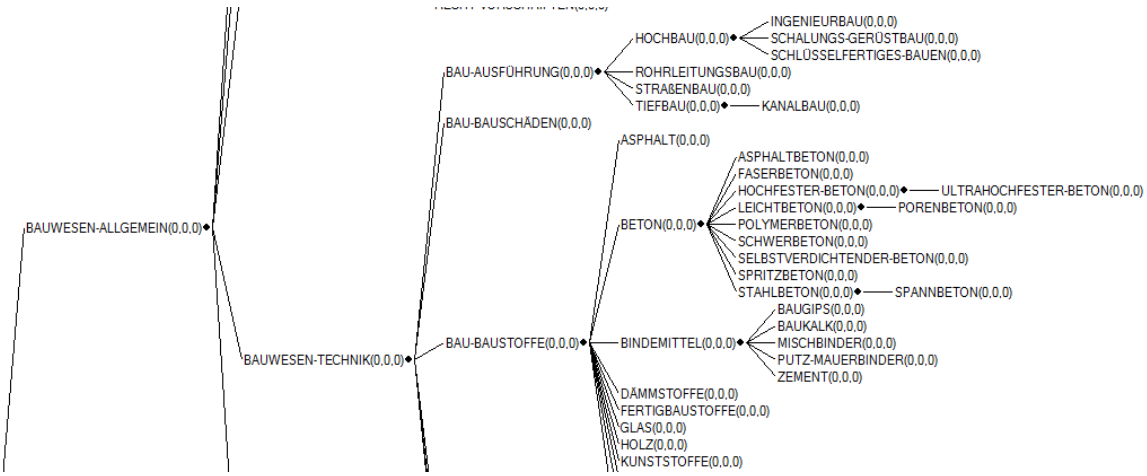


Abbildung 23: Taxonomie der Mindox GmbH zur Klassifikation im Bereich Bauwesen - Baustoffe

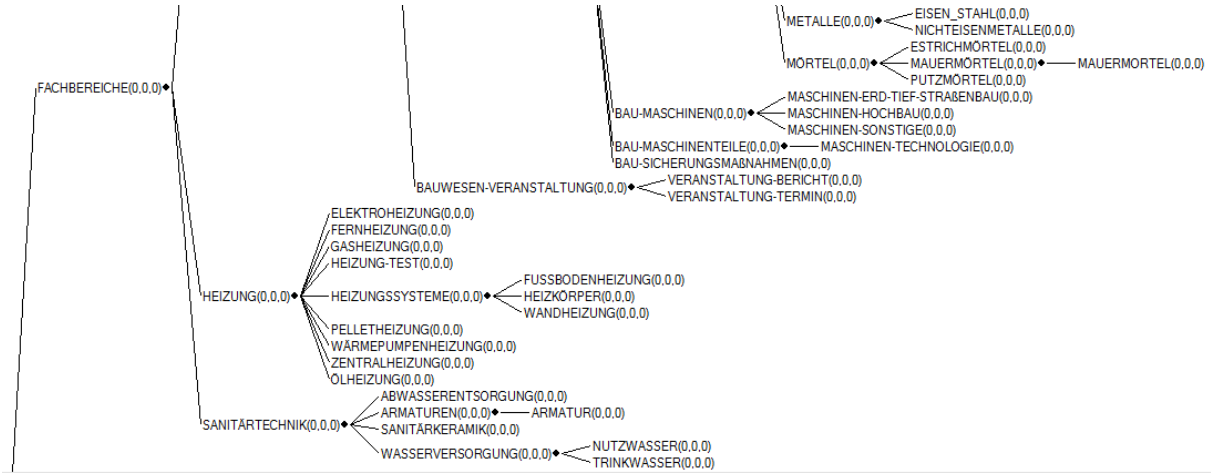


Abbildung 24: Taxonomie der Mindox GmbH zur Klassifikation im Bereich Bauwesen - Heizung- und Sanitärtechnik

2.2.2.2 Ontologie zur Repräsentation von Dokumenten

Nach [Pol85] kann Wissen innerhalb eines Unternehmens in zwei Formen vorliegen:

- Implizites Wissen:
  Wissen, dass an eine Person auf Grund ihrer Erfahrung, Kompetenz, Know-how usw. gebunden ist. Diese Form des Wissens lässt sich häufig sehr schwierig formal darstellen und bleibt damit anderen Personen unzugänglich.
- Explizites Wissen:
  Wissen, welches in einer dokumentierten Form vorliegt. Das heißt, dass das entsprechende Wissen formal darstellbar ist, weil der Wissende genau weiß, warum er über das entsprechende Wissen verfügt und dieses auch beschreiben kann. Explizites Wissen liegt zum Beispiel in Dokumenten, Datenbanken etc. innerhalb des Unternehmens vor.

Dieses Forschungsvorhaben hat sich ausschließlich mit der Frage beschäftigt, wie explizites Wissen innerhalb eines Unternehmens zentral bereitgestellt werden kann. Um implizites Wissen innerhalb eines Unternehmens in einem Wissensarchiv abzubilden, müsste dieses zuerst in explizites Wissen transformiert werden. Diese Externalisierung ist jedoch ein komplexer Prozess und häufig mit der Bereitschaft des Wissenden verbunden, das Wissen auch preiszugeben.

Dokumenttypen

Obwohl explizites Wissen bereits in einer formal beschriebenen Darstellung im Unternehmen vorliegt, gibt es dennoch viele verschiedene Darstellungsmöglichkeiten. Informationen können in papiergebundenen Dokumenten, in elektronischen Textdateien oder E-Mails vorliegen, aber auch das Adressbuch oder der Kalender eines Mitarbeiters kann relevante Informationen enthalten, ebenso wie Bild- oder Audiodateien. Diese Vielfältigkeit an Informationsquellen erschwert eine einheitliche Verarbeitung innerhalb des IE-Systems. Jede dieser Quellen hat ihre Besonderheiten, von denen einige näher betrachtet werden sollen:

- **Informationsmedium**:
  Informationen können in vielfältigen Informationsträgern vorhanden sein. Die häufigste Form ist die textuelle Darstellung. Daneben können Informationen aber auch visuell (Bilder, Videos, Grafiken etc.) oder akustisch (Audio-Dateien, etc.)

vorliegen. Während sich die klassische IE auf die Verarbeitung von textuellen Informationen konzentriert, soll das hier entwickelte IE-System ebenfalls die Verarbeitung von visuellen und auditiven Medien vorsehen.

- **Erscheinungsform**:
  Ein Dokument kann in analoger oder digitaler Form vorliegen. Um Informationen innerhalb eines Dokumentes maschinell verarbeiten zu können, muss zuerst eine Digitalisierung stattfinden. Im Dokumentenmanagement werden die beiden Begriffe Non-Coded Information (NCI) und Coded Information (CI) verwendet. Papierdokumente, aber auch deren eingescannten Abbilder, zählen zu den NCI-Dokumenten, da diese über keinerlei codierten Informationen verfügen. Erst durch den Einsatz einer Texterkennungssoftware wird ein NCI-Dokument in ein CI-Dokument überführt und kann innerhalb des IE-Systems weiterverarbeitet werden. CI-Dokumente enthalten die Informationen in einer codierten Form, sodass diese von einem Programm interpretiert und verarbeitet werden können (vgl. [JG99]).
- **Datenformat**:
  Darunter versteht man, in welchem Format die Daten innerhalb der Datei abgespeichert sind. Hier können hauptsächlich drei Speicherformate unterschieden werden. Interne Applikationsformate liegen in binärer Form vor und sind in der Regel nicht direkt von anderen Applikationen oder dem Benutzer lesbar (z.B. MS-Word, PDF etc.). Die Extraktion des Textes ist ohne eine entsprechende Decodierung des Applikationsformats nicht möglich. Externe Repräsentationsformate enthalten die Informationen bereits in einer dokumentierten Form und sind sowohl menschen- als auch maschinenlesbar. Häufig sind diese Formate standardisiert (wie z.B. XML, HTML etc.). Letztlich können Informationen als reiner Text, zum Beispiel in einer ASCII- oder Unicode-Codierung, vorliegen. Diese Dokumente enthalten außer Zeilenumbrüchen und Tabulatoren keine Formatierungsinformationen. Der Zugriff auf den Text ist ohne eine vorherige Decodierung möglich.
- **Metadaten**:
  Viele Dokumente verfügen über zusätzliche Informationen zur Beschreibung des Inhalts, des Autors etc. Diese Metadaten sind vor allem für nicht textuelle Dokumente wie Audio- oder Videodateien sehr hilfreich, da eine Informationsextraktion dieser Medien sich ansonsten als sehr schwierig erweist. Problematisch ist jedoch, dass zum einen bisher kein Standard von einem Normierungsgremium zur Beschreibung dieser Metadaten verabschiedet wurde und zum anderen diese Metadaten häufig nicht gepflegt werden. Während für MP3-Dateien die ID3-Tags den Status eines Quasi-Standards erreicht haben, gibt es für die Beschreibung von Bild- und Videodaten derzeit noch kein einheitliches Format. Einige Metadaten (z.B. EXIF 1) konzentrieren sich auf die Beschreibung von technischen Daten (z.B. Typ der digitalen Kamera, Beleuchtungszeit etc.), die für das Einbinden in ein Wissensarchiv von geringem Interesse sind. Andere Formate (z.B. IPTC 2, XMP 3 usw.) beschränken sich auf beschreibende Metadaten, wie den Autor oder die Kurzbeschreibung des Mediums. Diese Metadaten ermöglichen eine Verarbeitung von Medien, die keine oder nur sehr schwer zugängliche textuelle Informationen enthalten [HFW03]. Parallel zu diesen spezifischen Formaten existiert der Dublin Core (DC), ein von der DCMI (Dublin Core Metadata Initiative) verabschiedeter Standard zur Beschreibung beliebiger Dokumente. Der DC wurde jedoch ursprünglich für den Einsatz im Internet entwickelt und kann deshalb nur auf eine Bild- oder Videodatei referenzieren und ist nicht eigentlicher Bestandteil der Datei. Damit müssten immer

zwei Dateien verwaltet werden, weshalb sich der DC außerhalb des Internets nicht etablieren konnte.

- **Strukturierungsgrad der Informationen**:
  Dokumente werden generell in strukturierte, semi-strukturierte (manchmal auch als schwach strukturiert bezeichnet) und unstrukturierte Dokumente unterschieden. Jedoch besteht keine einheitliche Auffassung bezüglich der Definitionen, vor allem im Bereich der semi-strukturierten Dokumente finden sich unterschiedliche Aussagen. So verstehen [HMS03] unter strukturierten Informationen Daten, die in einer Datenbank oder einer Datei definiert sind und einem festgelegten Format unterliegen, was eine maschinelle Weiterverarbeitung erleichtert. Unstrukturierte Daten sind als Fließtext vorhanden und erfordern Techniken der natürlichsprachlichen Verarbeitung, da keinerlei Strukturinformationen enthalten sind. Schließlich sind semi-strukturierte Texte für [HMS03] Texte, die aus Datenbanken generiert wurden und dadurch teilweise ihre Struktur verloren haben. [Dir05] und [BF04] dagegen fassen alle Dokumente, „die eine Strukturierung in Felder enthalten ohne dass diese Strukturierung allerdings einheitlich erfolgt oder die Struktur des Dokuments explizit vom Autor vorgegeben wird [...]“, in der Kategorie der semi-strukturierten Texte zusammen. Darunter fallen insbesondere Dokumente, die mit Hilfe von Auszeichungssprachen (XML, HTML etc.) erstellt wurden. Diese letztere Auffassung der Strukturierungsgrade wird auch in der vorliegenden Arbeit verwendet.

Die gerade beschriebene Vielfältigkeit der Dokumenttypen resultiert in der Anforderung an das Gesamtsystem, dass möglichst viele unterschiedliche Dokumente verarbeitet werden müssen. Dabei muss folglich auf die o.a. Besonderheiten der einzelnen Dokumenttypen eingegangen werden. Abbildung 25 verdeutlicht dies in schematischer Form: Das IE-System bekommt als Eingabe ein beliebiges Dokument und versucht aus diesem Wissen zu extrahieren. Dieses Wissen wird anschließend in einem zentralen Archiv zur Verfügung gestellt.

Eine mögliche Variante zur Verarbeitung der unterschiedlichen Dokumenttypen könnte darin bestehen, dass verschiedene Extraktionspfade innerhalb des IE-Systems existieren. Ein Pfad zur Bearbeitung von E-Mails, ein weiterer für PDF-Dokumente usw. Diese Variante hätte den Vorteil, dass das IE-System leicht auf die Besonderheiten der einzelnen Dokumenttypen eingehen kann, da diese explizit auf den einzelnen Bearbeitungspfaden implementiert werden könnten (siehe Abbildung 3.2).

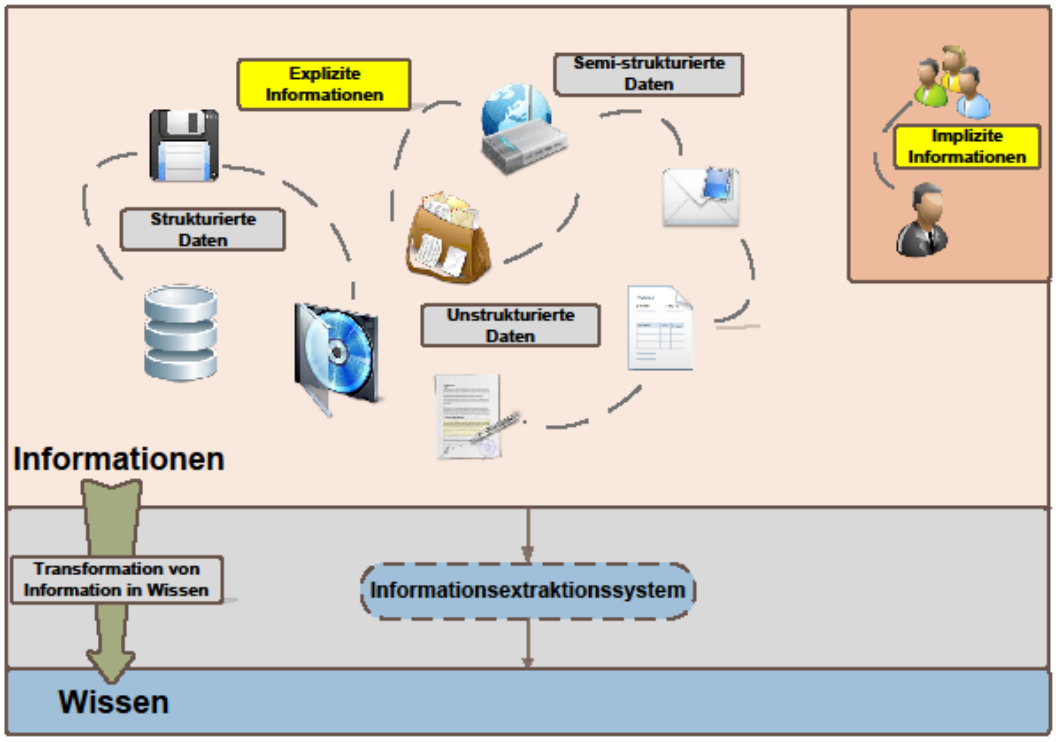


Abbildung 25: Zur Wissensverarbeitung müssen zahlreiche unterschiedliche Dokumente verarbeitet werden

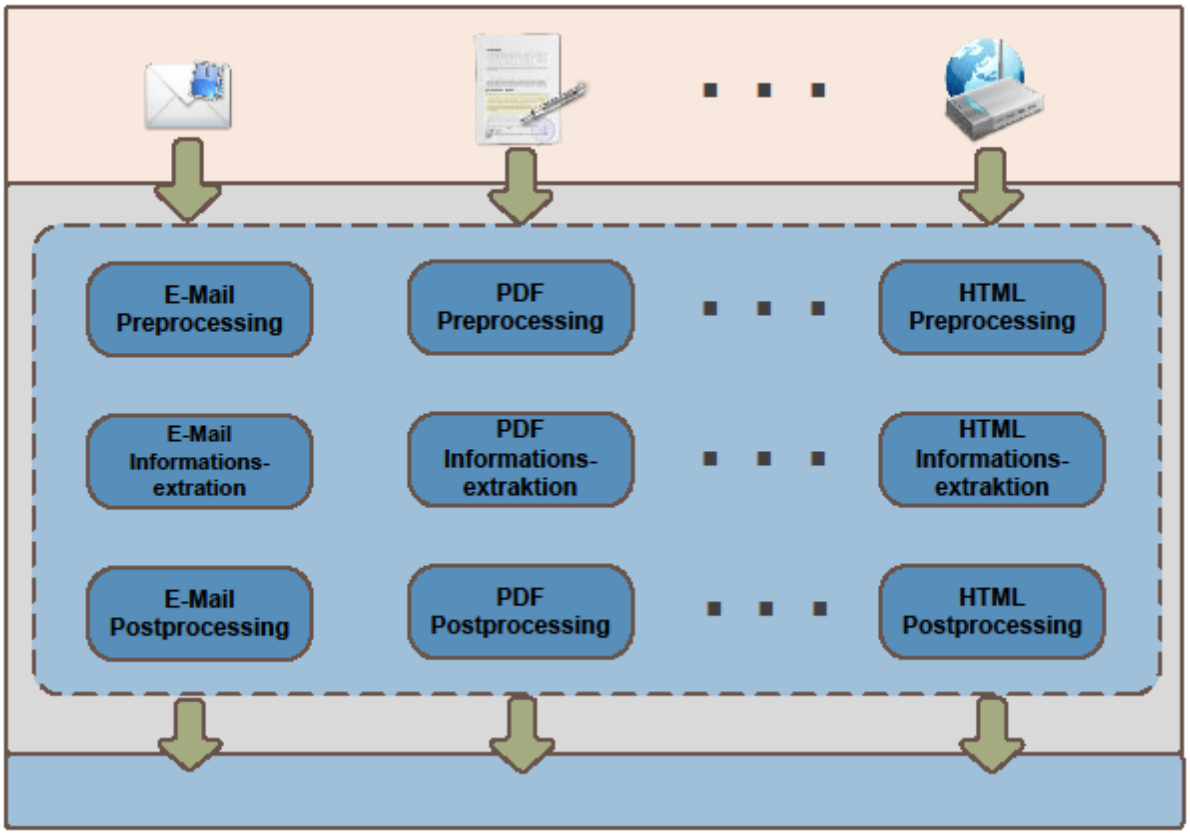


Abbildung 26: Getrennte Vorverarbeitung der unterschiedlichen Dokumenttypen

Erfahrungen aus den vorherigen Projekten Informationsextraktion aus dem Internet (INFOXI) und Intelligente Internetsuche (ISS) der Fachhochschule Trier zeigten jedoch, dass eine getrennte Verarbeitung von Dokumenttypen hohe Redundanzen zur Folge hat und damit die Übersichtlichkeit und Wartbarkeit des Programmcodes erschwert wird. In beiden Projekten

wurden HTML-Seiten, fehlerhafte HTML-Seiten und eingescannte Dokumente unterschiedlich behandelt. HTML-Dokumente wurden in eine eigens entwickelte interne Datenstruktur überführt, die auf der Hierarchie der HTML-Tags basierte.

Im Vordergrund standen die HTML-Strukturinformationen. HTML-Seiten, die in ihrem strukturellen Aufbau fehlerhaft waren, konnten nicht ohne weiteres in diese Datenrepräsentation transformiert werden. Wenn die Behebung der Strukturfehler nicht möglich war, wurden alle HTML-Tags gelöscht, so dass letzten Endes eine reine Textdatei übrig blieb. Zur Verarbeitung dieser Textdateien kam eine weitere Datenrepräsentation zum Einsatz. Letztlich existierte für eingescannte bzw. PDF-Dokumente eine weitere Variante. Die unterschiedlichen Varianten ermöglichten es, gezielt die zusätzlichen Angaben (HTML-Tags, Positionsangaben der OCR-Ergebnisse, etc.) zu nutzen.

Viele IE-Techniken kamen jedoch in der Verarbeitung aller drei Dokumenttypen zum Einsatz, konnten aber auf Grund der unterschiedlichen internen Repräsentationen nicht ohne Modifikationen wiederverwendet werden. Das hatte zur Folge, dass es häufig zu redundanten Lösungen kam. Anpassungen an den Extraktionstechniken mussten somit stets an verschiedenen Stellen innerhalb des Systems durchgeführt werden, was einen erheblichen Mehraufwand implizierte. Zudem trat der erhoffte Vorteil bezüglich des Mehrwertes der zusätzlichen Strukturinformationen nicht in dem gewünschten Maße ein. Viele Informationen wurden nicht benutzt, da zum Beispiel keine allgemeingültige Vorschrift existiert, an die sich die Autoren halten müssen. Eine Überschrift bzw. eine Tabelle wird in HTML-Seiten häufig nicht als solche eingesetzt, sondern zweckentfremdet zur Layoutgestaltung oder ähnlichem missbraucht.

Die o.g. Erfahrungen waren ausschlaggebend dafür, in diesem Projekt ein Konzept zu entwickeln, das eine höhere Wartbarkeit und Wiederverwendbarkeit garantiert. Eine individuelle Verarbeitung der einzelnen Dokumenttypen würde sich dafür als kontraproduktiv erweisen.

Betrachtet man die unterschiedlichen Informationsträger (im Folgenden auch öfters Informationseinheit genannt) innerhalb des Unternehmens, besteht die Hauptaufgabe des Forschungsprojektes darin, textuelle Informationen in Wissen (im Folgenden auch öfters Wissenseinheit genannt) zu überführen. Die verschiedenen IE-Techniken, die für diesen Prozess notwendig sind, basieren demnach auf der Verarbeitung von Text. Es würde also folglich ausreichen, dem IE-System lediglich die rein textuellen Informationen als Eingabe zur Verfügung zu stellen. Die Repräsentation dieser Informationen sollte für jeden Dokumenttyp gleich sein. Die Entwicklung einer allgemeinen Repräsentation von textuellen Informationen ist dafür Voraussetzung. Durch diese gemeinsame interne Datenrepräsentation wäre es möglich, die einzelnen IE-Techniken sowohl auf Texte innerhalb einer E-Mail, einer HTML-Seite oder einem eingescannten Dokument anzuwenden.

Eine mögliche Variante bestand darin, aus den verschiedenen Dokumenten den Text zu extrahieren und dem IE-System als reinen Fließtext zur Verfügung zu stellen. Gewisse strukturelle Informationen sind jedoch in allen Dokumenten vorhanden und sollten demnach auch erhalten bleiben. Einen Text kann man generell in Wörter, Sätze und Abschnitte unterteilen. Diese strukturelle Textgliederung lässt sich für alle Dokumenttypen bestimmen, da es in der deutschen Sprache eindeutige Zeichen zur Wort-, Satz- und Abschnittstrennung gibt, auch wenn einige Interpunktionszeichen wie der Punkt verschiedene Verwendungen finden.

Zur Repräsentation der textuellen Informationen wurde somit ein eigenes Repräsentationsformat verwendet. Um sowohl eine maschinen- als auch menschenlesbare Darstellung zu ermöglichen, bietet sich die Verwendung einer XML-basierten Formatierungssprache an. Die *Extended Markup Language* (XML) hat zudem den Vorteil, dass sie ein plattformunabhängiges Austauschformat für Daten ist. Da in dem Forschungsprojekt unterschiedliche Programmiersprachen zur Realisierung einzelner Module eingesetzt wurden, sprach dies ebenfalls für die XML-basierte Lösung. Zudem sind OWL-Ontologien ebenfalls in XML serialisiert, sodass eine einheitliches Repräsentationsformat für alle Bereiche im Forschungsvorhaben gewährleistet ist.

Ein Beispiel eines solchen Austauschformats für Dokumente ist das Open Document Format for Office Applications welches 2006 als Standard durch die Organization for the Advancement of Structured Information Standards (OASIS) verabschiedet wurde. Dieses Format basiert auf dem ursprünglichen Dateiformat von OpenOffice und findet inzwischen bereits in mehreren Textverarbeitungsprogrammen Verwendung (seit 2007 auch im Microsoft Office Paket). Abbildung 27 zeigt den Ausschnitt eines Textdokumentes im Open Document Format (ODF). Die Spezifikation des ODF unterteilt Texte in Überschriften <text:h> und Abschnitte <text:p>. Eine weitere Unterteilung in Sätze und Wörter ist nicht möglich. Des Weiteren enthält dieses Format zwar einige Konzepte, die auch in diesem Vorhaben von Interesse sind, dennoch wurde es primär als plattformunabhängiges und freies Austauschformat für Bürodokumente entwickelt und enthält nicht gezielt IE-relevante Informationen. Aus diesem Grund erwies sich die Entwicklung einer eigenen Repräsentationssprache, die speziell den Anforderungen des IE-Systems gerecht wird, als sinnvoll. Dabei wurde auf ähnliche Konzepte zurückgegriffen, wie diese im Projekt NEPOMUK zur Repräsentation eingesetzt wurde.

```
<?xml version="1.0" encoding="UTF-8"?>
<office:document-content xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0" xmlns:style="urn:oasis:names:tc:o
  <office:body>
    <office:text text:use-soft-page-breaks="true">
      <text:h text:style-name="Heading_20_1" text:outline-level="1">Wissenschaftliche und Technische Arbeitsziele</text:h>
      <text:h text:style-name="Heading_20_2" text:outline-level="2">Allgemeine Ziele</text:h>
      <text:p text:style-name="P2">
        Aufbauend auf die Betrachtung der o.g. Referenzmodelle soll in MUY BIEN ein angepasstes
        <text:span text:style-name="T1">Modell zur Gestaltung von Benutzerschnittstellen für Senioren</text:span>
        entwickelt werden. Dies beinhaltet eine Spezifikation der zu bearbeitenden Aufgaben und der verwendbaren Interakti
      </text:p>
      <text:p text:style-name="P2">
        Neben der Ausgestaltung eines solchen Modells sollen in Zusammenarbeit mit den Partnern und Nutzern auch Untersuch
      </text:p>
      <text:p text:style-name="P2">
        Besondere Beachtung soll der Eingabekanal Sprache erfahren. Hier streben wir eine <text:s/><text:span text:style-n
      </text:p>
      <text:p text:style-name="Text_20_body"/>
      <text:h text:style-name="Heading_20_2" text:outline-level="2">Vorgehensweise</text:h>
```

Abbildung 27: Auszug aus einem ODF-Dokument in XML-Repräsentation

Abbildung 28 stellt den erweiterten Konzeptentwurf mit der eingesetzten Repräsentationssprache dar. Es wird deutlich, dass diese als Schnittstelle zwischen den Informationseinheiten innerhalb des Unternehmens und dem IE-System dient. Damit die verschiedenen Dokumenttypen in die Repräsentationssprache transformiert werden können, musste für jeden Dokumenttyp ein Konverter implementiert werden. Das Konzept für die Dokumentenkonverter wird Abschnitt 2.2.3 näher vorgestellt.

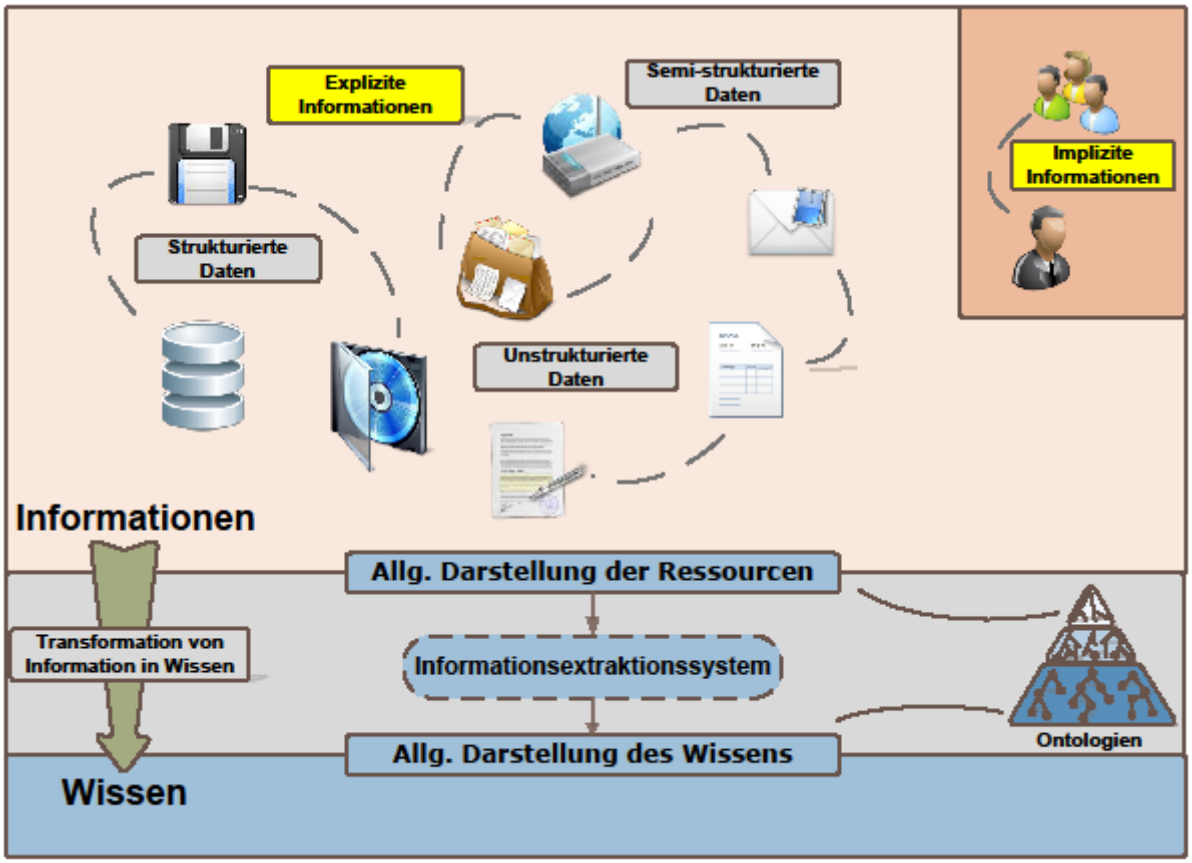


Abbildung 28: Konzept der Dokumentverarbeitung über eine allgemeine Repräsentationssprache für alle Dokumente

Dokumentklassifikation

Zur Entwicklung einer eigenen Repräsentationssprache für die Dokumente in unseren Partnerunternehmen wurde in einem ersten Schritt eine Taxonomie der unterschiedlichen Dokumenttypen erstellt, die wie die in Abschnitt 2.2.2.1 vorgestellten Taxonomien der Firma Mindox, zur Klassifikation dienen sollen.

Es wurde eine drei-schichtige Taxonomie (siehe Abbildung 29) gewählt:

- Top-Level:
  Enthält als einziges Element die InformationUnit als Wurzel. Sie stellt den höchsten Abstrahierungsgrad eines Dokumentes dar und dient als Container für die Elemente aus den darunter liegenden Ebenen. Die Repräsentationssprache für die InformationUnit enthält noch keine Element zur Darstellung von textuellen Informationen. Es können lediglich allgemeine Metadaten zu dem Dokument erfasst werden.
- Intermediate-Level:
  Auf dieser Ebene findet eine Klassifizierung der Dokumente in die Kategorien Media, Addressbook, Calendar und Textdocument statt. Im Intermediate-Level werden zusätzlich die Repräsentationssprachen für die einzelnen Kategorien definiert.
- Document-Level:
  Im Document-Level werden die jeweiligen Dokumenttypen konkret modelliert.

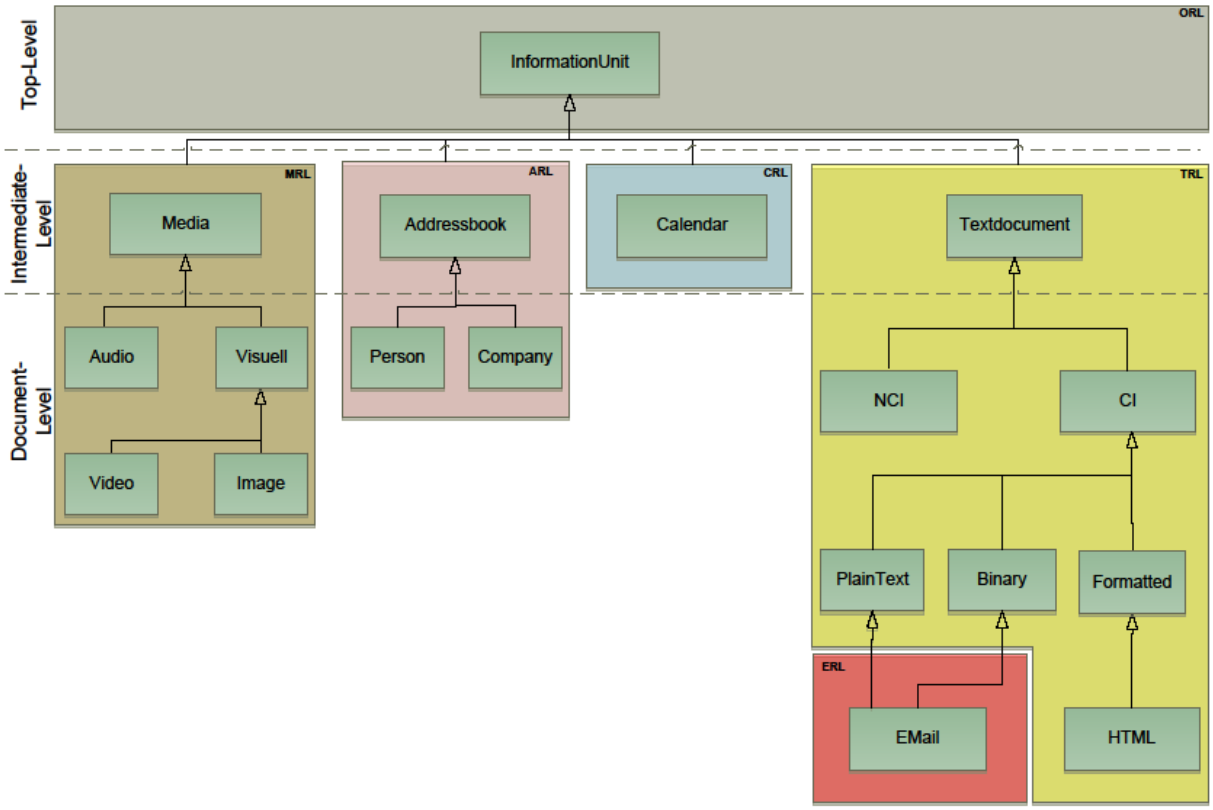


Abbildung 29: Taxonomie von Dokumenttypen

OnToBau Representation Language

Jede Kategorie in den ersten beiden Schichten, wird durch ein eigenes XML-Schema modelliert, die zusammen die OnToBau-Representation-Language (ORL) bilden.

Das zentrale Element stellt die InformationUnit dar. Diese enthält, unabhängig um welchen Typ es sich handelt, allgemeine Informationen über das Dokument. Der Dublin Core Standard wurde bereits angesprochen. Einige der darin enthaltenen Elemente, wie Informationen über den Autor, den Titel oder eine Beschreibung des Dokumentes, wurden in das entwickelte XML-Schema importiert und eingesetzt. Dadurch wird ermöglicht, dass zu jedem Dokument Metadaten erfasst werden können. Da diese Informationen derart allgemein sind, dass sie zu jedem beliebigen Dokumenttyp erhoben werden können, bildet die InformationUnit die Wurzel aller Dokumente.

Ein Auszug aus dem grafischen Schema der ORL ist in Abbildung 30 zu sehen. Es wird deutlich, dass eine InformationUnit aus genau drei Elementen aufgebaut ist. Das erste Element enthält die bereits angesprochenen Metadaten. Diese setzen sich aus einer Auswahl der Dublin Core Elemente und zusätzlichen selbst definierten Metadaten zusammen. Anschließend kann ein Element folgen, welches die extrahierten textuellen Informationen des Dokumentes enthält. Dafür wurde ebenfalls eine eigene Repräsentationssprache entwickelt, die im späteren Verlauf noch beschrieben wird. Abschließend muss genau ein Element aus der zweiten Schicht folgen. Dieses enthält spezifische Informationen zu dem Dokumenttyp.

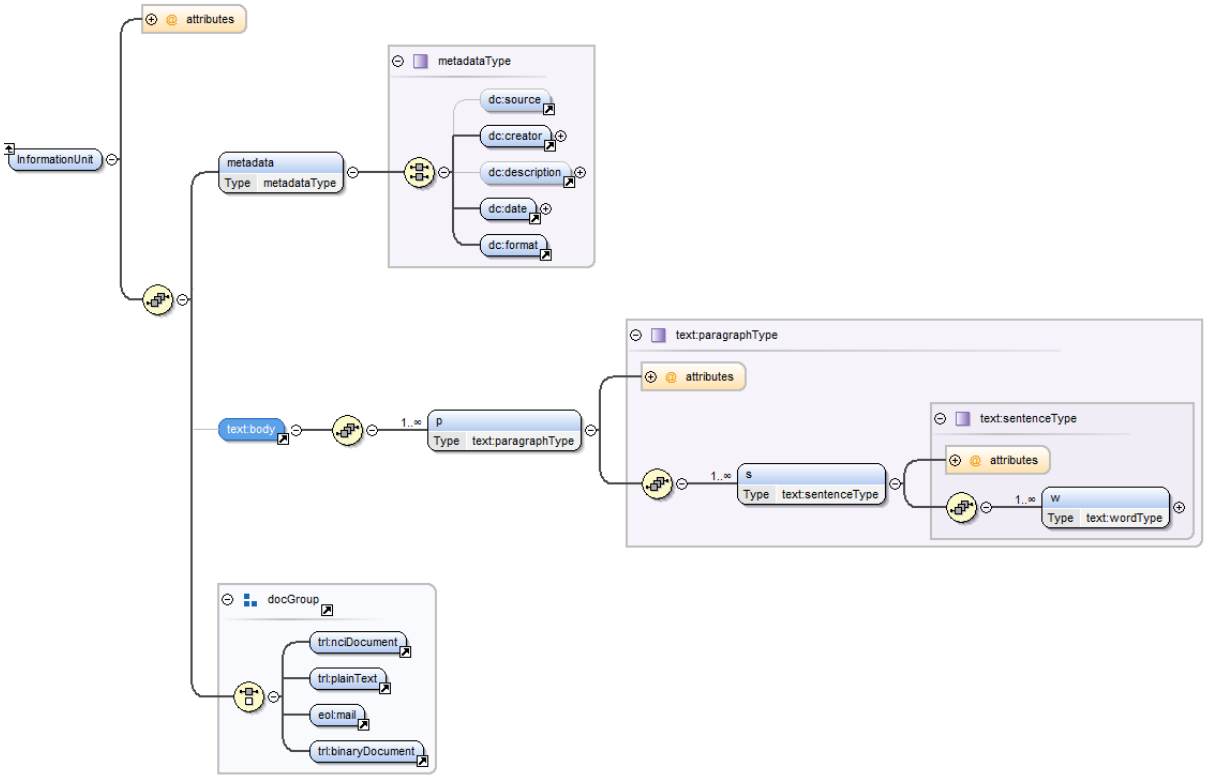


Abbildung 30: Auszug aus dem XML-Schema der ORL

Das IE-System bekommt die verschiedenen Dokumente in der einheitlichen Repräsentation einer InformationUnit als Eingabe geliefert. Dokumentenkonverter müssen zuvor eine Konvertierung in dieses Format vornehmen.

Die Metadaten können teilweise automatisch erstellt werden, da viele Dokumente diese bereits enthalten (z.B. enthalten PDF-Dokumente Informationen über Autor, Titel usw. oder MP3-Dateien enthalten ID3-Tags mit Interpret, Titel usw.). Dennoch wird eine manuelle Anreicherung der Dokumente um Metadaten zusätzlich erfolgen müssen. Vor allem für Dokumente die keine textuellen Informationen enthalten, wie Bilder oder Videos, sind Metadaten zum späteren Retrieval im Wissensarchiv von enormer Bedeutung. Eine Möglichkeit wäre, dass der Agent dem Benutzer die Möglichkeit anbietet, Metadaten zu erfassen, bevor das Dokument in das Wissensarchiv aufgenommen wird.

Für die IE sind die wesentlichen Informationen im Textelement enthalten. Sollten zu bestimmten Phasen in der Verarbeitung, weiterführende Informationen notwendig sein (z.B. ob der Satz ein Link auf der HTML-Seite ist, der Text aus der E-Mail eines bestimmten Absenders stammt etc.), so sind diese im dritten Element enthalten.

Die Trennung des Textes und der dokumentspezifischen Informationen hat den großen Vorteil, dass durch diese Modularität eine hohe Erweiterbarkeit garantiert wird. Müssen zu einem späteren Zeitpunkt weitere Dokumenttypen durch das IE-System verarbeitet werden, so muss die ORL lediglich um eine weitere Repräsentationssprache erweitert werden. Die Modellierung des Textes bleibt davon unberührt.

Dokumenttyp-Repräsentation

Die zweite Schicht klassifiziert die Dokumente innerhalb eines Unternehmens in vier unterschiedliche Kategorien. Jede dieser Kategorien wird durch eine eigene Repräsentationssprache modelliert. In diesem Vorhaben wurden die folgenden Repräsentationssprachen vorgesehen:

- Addressbook-Representation-Language (ARL):
  Das Adressbuch eines Mitarbeiters enthält relevante Informationen über interne wie auch externe Personen und zu anderen Unternehmen. Die Einträge eines Adressbuchs in das Wissensarchiv aufzunehmen, erweist sich deshalb als sinnvoll. Dadurch wird u.a. ermöglicht, dass Relationen zwischen Dokumenten und Personen bzw. Unternehmen gesetzt werden können.
- Calendar-Representation-Language (CRL):
  Der Kalender eines Mitarbeiters enthält Informationen über vergangene und anstehende Ereignisse. Viele Ereignisse, wie zum Beispiel ein wichtiger Kundentermin, können mit Dokumenten (z.B. Angebot) in Verbindung stehen. Durch die Aufnahme dieser Ereignisse in das Wissensarchiv wäre es möglich, Relationen zwischen den Informationen im Kalender und vorhandenen Dokumenten zu bestimmen.
- Textdocument-Representation-Language (TRL):
  Diese Kategorie ist ohne Frage der wichtigste Wissensspeicher eines Unternehmens. Deshalb erfährt die Modellierung der Textdokumente in dieser Arbeit eine besondere Aufmerksamkeit. Alle Dokumente in einem Unternehmen die hauptsächlich textuelle Informationen enthalten, werden in dieser Kategorie zusammengefasst. Im vorherigen Abschnitt wurden die verschiedenen Ausprägungen von Textdokumenten schon aufgezeigt.

Diese unterschiedlichen Typen werden in der dritten Schicht genauer behandelt.

- Media-Representation-Language (MRL):
  Die letzte Kategorie enthält alle medialen Dokumente wie Video- oder Audiodateien. Diese Dokumente haben in den meisten Unternehmen eine eher untergeordnete Rolle, vor allem da eine automatische IE für diese Dokumente sehr schwierig ist. Eine mögliche Lösung, um solche Dokumente dennoch in das Wissensarchiv aufzunehmen, wird im späteren Verlauf dieses Kapitels aufgezeigt.
- EMail-Representation-Language (ERL):
  Diese Repräsentationssprache bildet eine Ausnahme, da sie nicht zur zweiten Schicht gehört. E-Mails ordnen sich innerhalb der Taxonomie der Klasse der Textdokumente unter. Da E-Mails jedoch, im Vergleich zu allen anderen Dokumenten in dieser Kategorie, über spezielle Zusatzinformationen (Absender, Empfänger, Betreff usw.) verfügen, wurde die Einführung einer eigenen Repräsentationssprache als sinnvoll erachtet.

Für jede dieser fünf Kategorien existiert ein eigenes XML-Schema, welches genau festlegt welche dokumenttypischen Informationen modelliert sind. So enthält die TRL zum Beispiel eine spezielle Repräsentation für NCI- und CI-Dokumente und die MRL eigene Informationen über Spiellänge des Mediums oder eingesetzte Videocodecs, usw.

Die konkreten Dokumenttypen werden in der dritten Schicht modelliert und stellen die Blätter innerhalb der Taxonomie dar. Zur besseren Gliederung und um Vererbung zu realisieren, wurden auch in der dritten Schicht einige Dokumenttypen in Oberklassen zusammengefasst. Stellvertretend sollen hier einige der Klassen aus der dritten Schicht vorgestellt werden:

- Binary Documents:
  Alle Dokumente, die in einem binären Dateiformat vorliegen, fallen unter diese Kategorie. Dies sind in der Regel Applikationsdokumente wie PDFs, Word-Dateien etc. Elemente aus dieser Klasse enthalten u.a. Informationen über das Applikationsformat, welches Programm zur Decodierung verwendet wurde usw.
- Formatted Documents:
  Hierunter fallen alle Dokumente, die in einer Repräsentationssprache wie HTML oder XML vorliegen. HTML-Seiten enthalten neben den reinen textuellen Informationen zusätzlich Angaben bezüglich Formatierung und Layout. DesWeiteren wird in HTML-Seiten häufig auf andere Dokumente verwiesen. Dies können u.a. Bilder, downloadbare Dateien etc. sein. Die TRL bietet Möglichkeiten, diese Zusatzinformationen zu erfassen.
- Person: Adressbücher enthalten in der Regel zwei Arten von Einträgen. Zum einen zu Personen - mit den üblichen Angaben zu Adresse, E-Mails und Telefonnummern - oder zu Firmen. Beide Kategorien sind von Interesse, wenn Relationen zwischen Dokumenten und entsprechenden Verfassern, Absendern, beteiligten Personen usw. hergestellt werden sollen.

Jedes Dokument, das von dem IE-System verarbeitet werden soll, muss einer der Kategorien der dritten Schicht zugeordnet werden. XML bietet einfache Vererbungsmechanismen, die aus objekt-orientierten Programmiersprachen bekannt sind. So erweitern alle Unterklassen, die vorherigen Klassen um neue Elemente und Attribute oder schränken diese ein. Zusätzlich gibt das XML-Schema genau vor, welche Angaben vorgeschrieben und welche optional sind. Der Entwickler eines Dokumentenkonverter ist an diese Vorgaben gebunden. Abbildung 31 den Inhalt einer ORL-Datei mit den Informationen aus einer E-Mail.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<InformationUnit id="otb-3-209912141512"
    <!-- Namespaces aus Platzgründen entfernt --> >
    <metadata>
        <dc:creator>Markus Schwinn</dc:creator>
        <dc:date>2009-12-14</dc:date>
        <dc:format>text/email</dc:format>
    </metadata>
    <text:body>
        <text:p id="1">
            <text:s id="11">
                <text:w id="111">Sehr</text:w>
                <text:w id="112">geehrte</text:w>
                <text:w id="113">Damen</text:w>
                <text:w id="114">und</text:w>
                <text:w id="115">Herren,</text:w>
            </text:s>
        </text:p>
        <text:p id="2">
    </text:body>
    <erl:mail>
        <erl:sender>schwinn@fh-trier.de</erl:sender>
        <erl:recipient>max@mobile.de</erl:recipient>
        <erl:sendDate>2009-12-10T12:00:00</erl:sendDate>
        <erl:recievedDate>2002-12-10T12:05:35</erl:recievedDate>
        <erl:attachment>
             <erl:attachmentItem>otb-5-20091208T123412</erl:attachmentItem>
        </erl:attachment>
    </erl:mail>
</InformationUnit>
```

Abbildung 31: Beispielhafte Repräsentation einer Email in der ORL

Textrepräsentationssprache

Die Repräsentation des Textes ist für das IE-System der wichtigste Bestandteil innerhalb der ORL. Damit das Textelement unabhängig vom Dokumententyp ist, wurde eine Modellierung gewählt, die universell einsetzbar ist. Aus diesem Grund ist eine Trennung von Inhalt und Layout notwendig, da je nachdem welcher Dokumenttyp vorliegt, mehr bzw. weniger Layoutinformationen vorhanden sind. Die Text-Representation-Language (TXRL) setzt deshalb nur die strukturellen Informationen um, die für alle Dokumente erfasst werden können. Dies sind im Konkreten Abschnitte, Sätze und Wörter, in die jeder Text unterteilt werden kann. Papierdokumente nehmen hier eine Sonderrolle ein. Diese müssen zur Verarbeitung erst digitalisiert werden.

Diese Digitalisierung erfolgt durch eine Texterkennungssoftware. Diese liefert als Ergebnis keine Informationen über Leerzeichen oder Zeilenumbrüchen. Diese Zusammenhänge müssen aufgrund der Positionsangaben der einzelnen Wörter selbst bestimmt werden. Dies kann dazu führen, dass die Segmentierung eines elektronischen Dokumentes und des analogen Gegenstückes nicht dieselben Ergebnisse liefert. Die folgenden beiden Abbildungen sollen diese Problematik verdeutlichen:

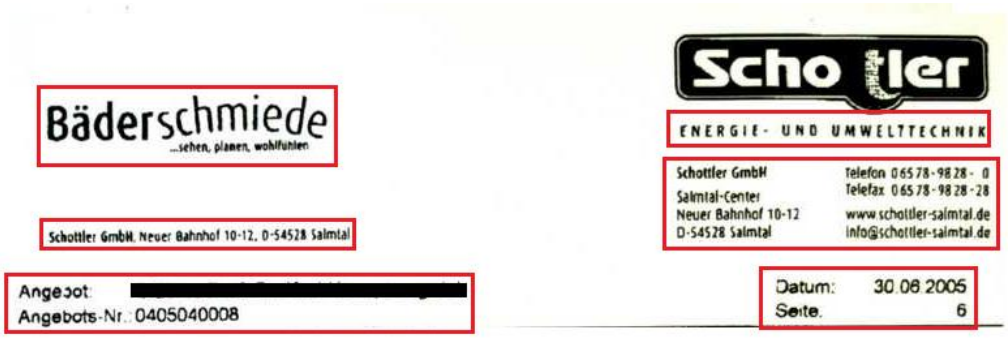


Abbildung 32: Segmentierung des Textes durch eine OCR-Software

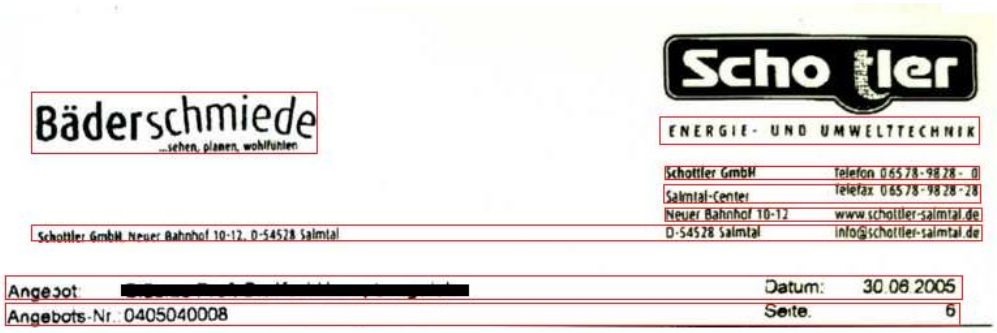


Abbildung 33: Segmentierung des Textes anhand der Zeilenumbrüche

Abbildung 32 zeigt die Segmentierung des eingescannten Dokumentes in Abschnitte, die von der OCR-Software durchgeführt wurde. Da der Zeilenumbruch nicht zur Segmentierung herangezogen werden kann, werden stattdessen zusammenhängende Textblöcke als Abschnitte klassifiziert. Das elektronische Original enthält alle Steuerzeichen, die der Benutzer zur Strukturierung verwendet hat. Dadurch ist es möglich die Segmentierung

anhand der vorhandenen Zeilenumbrüche durchzuführen. Dies führt jedoch zu einer anderen Strukturierung des Textes, wie in Abbildung 33 zu sehen ist.

Obwohl es wünschenswert wäre, dass unabhängig des Informationsträgers (analog oder digital) die Segmentierung desselben Textes konsistente Ergebnisse liefert, konnte für dieses Problem keine generelle Lösung gefunden werden. Da die Transformation eines digitalen Dokumentes in ein analoges verlustbehaftet ist und diese Informationen bei einer anschließenden Digitalisierung nicht wieder hergestellt werden können, kann die Segmentierung nicht in beiden Fällen auf die gleiche Art und Weise erfolgen. Die Struktur des Textes muss im analogen Fall anhand der Positionsangaben rekonstruiert werden, was zu Inkonsistenzen führen kann. Daneben birgt die Digitalisierung von Textdokumenten noch weitere Probleme in sich. So kommt es häufig zu fehlerhaften Erkennungen von Zeichen. Diese können durch verschiedenen Verfahren zwar teilweise korrigiert werden, eine fehlerfreie Einzelzeichenerkennung ist jedoch kaum zu erwarten und erschwert die anschließende Textverarbeitung. Dennoch liegen in vielen Unternehmen Informationen in papiergebundener Form vor (vgl. [Sap05]), so dass eine Behandlung dieser Dokumente nicht vernachlässigt werden konnte.

Bei Dokumente, die über zusätzliche Layoutinformationen verfügen (z.B. HTMLSeiten), können diese im dritten Element der ORL hinterlegt. Dieses Element wird für jeden Dokumenttyp eigens definiert und kann die dokumentenspezifischen Informationen (z.B. verwendete Schriftgrößen und -arten, welche Teile des Textes HTML-Überschriften oder Links sind, welche Anhänge die E-Mail hat usw.) aufnehmen. Sollten während der Extraktionsverarbeitung diese Informationen von Vorteil sein, kann der Algorithmus jederzeit auf diese Daten zugreifen. Extraktionsmodule, die lediglich auf rein strukturellen Informationen basieren, können dagegen alle Dokumente auf die gleiche Art und Weise verarbeiten.

Damit ein eindeutige Zuordnung zwischen Inhalt und Layout möglich ist, verfügt jeder Abschnitt, jeder Satz und jedes Wort über eine eindeutige ID. Werden im dritten Element Layoutinformationen angegeben, so referenzieren diese immer auf eine oder mehrere dieser IDs.

2.2.2.3 Ontologie fürs Bauwesen

Nach Wahl der Ontologiesprache wurde damit begonnen sowohl die domänenspezifische Bauwesen-Ontologie, als auch eine allgemeine Ontologie zur Beschreibung der Relationen zwischen den verschiedenen Dokumenten, wie E-Mails und Anhängen, Angebot und Rechnung, etc. aufzubauen. Abbildung 34 zeigt die Taxonomie aus der allgemeinen Ontologie zur Repräsentation von Geschäftsdokumenten. In Abbildung 35 ist eine grafische Darstellung der Relationen von ausgewählten Instanzen innerhalb der Bauwesen-Ontologie zu sehen.

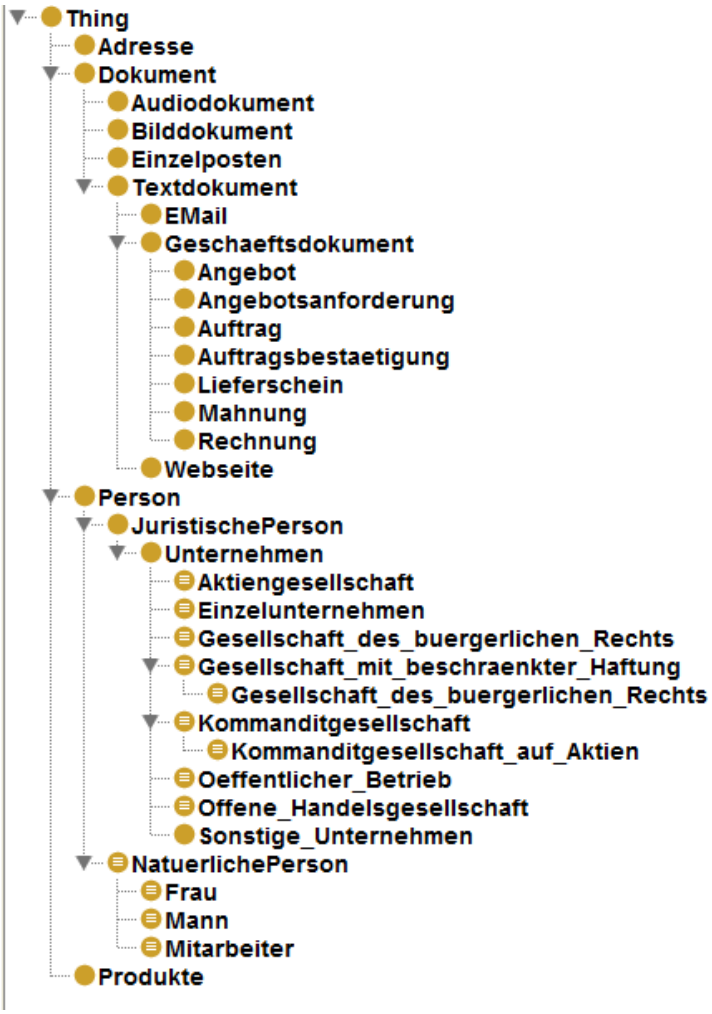


Abbildung 34: Auszug aus dem Domänen-Ontologie zur semantischen Repräsentation von Dokumenten und Personen

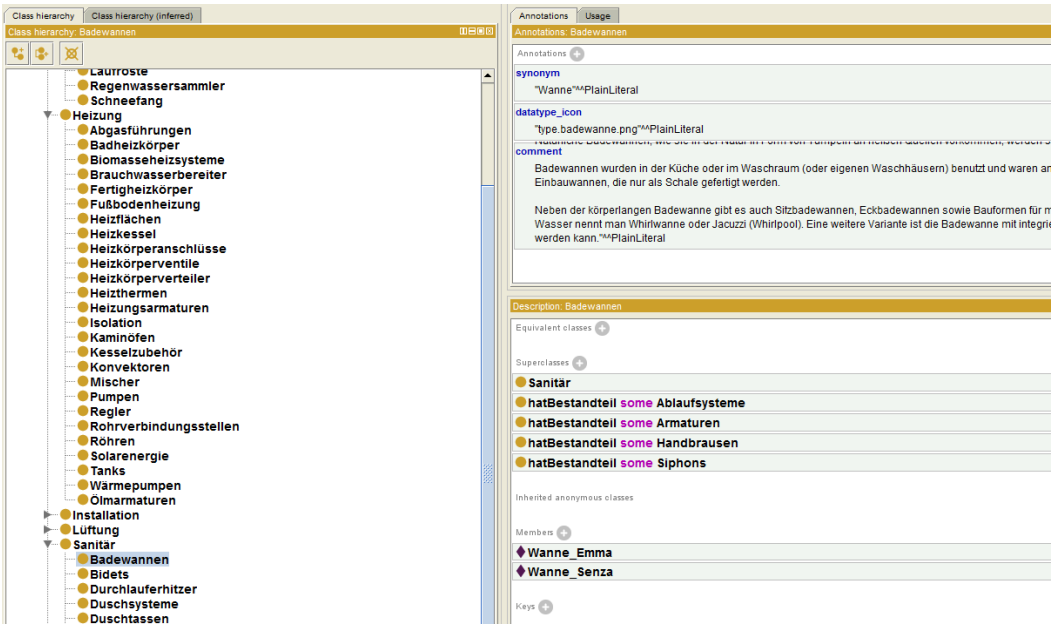


Abbildung 35: Ausschnitt aus der Domänen-Ontologie für die Bereiche Heizung und Sanitär

2.2.3 Integration von Klassifikations- und Retrievalverfahren

Die Ablage neuer Dokumente und der Zugriff auf vorhandenes Wissen soll weitgehend automatisiert werden. Dazu ist zu beschreiben, wie aus neuen Dokumenten die notwendige Information bestimmt werden soll, die Basis für eine Klassifikation ist. In Abhängigkeit verschiedener Dokumentklassen sollen verschiedene Klassifikationsverfahren ansprechbar sein. Für das Retrieval sind ebenfalls verschiedene Ähnlichkeitsmaße auswählbar und können somit den Dokumenten zugeordnet werden. Hier wird untersucht, welche Maße als Voreinstellung oder Default für bestimmte Wissenskategorien sinnvoll sind. Ein wesentliches Ziel innerhalb dieser Teilaufgabe bestand in der Automatisierung des Klassifikations- und Retrievalansatzes.

Bevor eine Klassifikation und Extraktion der relevanten Informationen durchgeführt werden kann, müssen die Dokumente vorverarbeitet werden. Die im vorherigen Abschnitt vorgestellte OnToBau Representation Language dient dabei als Datenformat für alle anschließenden Verfahren. Im Folgenden soll das Konzept der Vorverarbeitung (Preprocessing) näher beschrieben werden.

2.2.3.1 Preprocessing von Dokumenten

Abbildung 36 zeigt einen beispielhaften Ablauf des Preprocessings: Die einzelnen Module werden sequentiell ausgeführt, wobei jedes Modul als Eingabe die Ausgabe des vorherigen bekommt. Am Ende des Preprocessings erfolgt die Transformation des Textes in die Datenstruktur des IE-Systems. Diese ist stark von der eingesetzten Programmiersprache abhängig. Da diese zum Zeitpunkt dieser Arbeit noch nicht fest stand, erweist es sich als geeignet, die Transformation als eigenes Modul zu realisieren, da somit ein einfaches Austauschen ermöglicht wird.

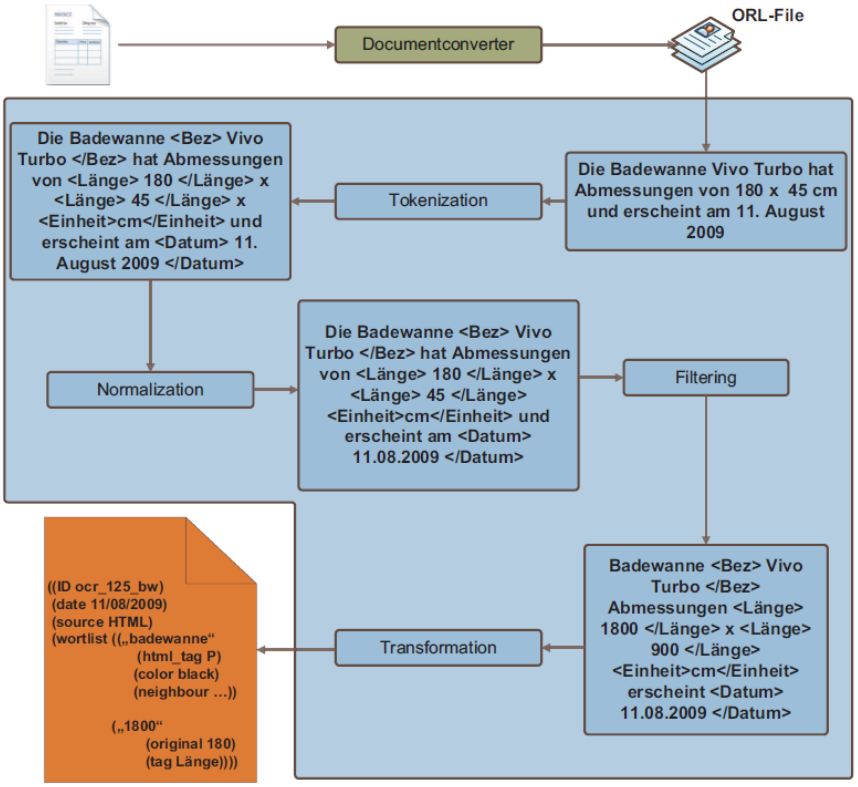


Abbildung 36: Beispielhafter Ablauf eines Preprocessings

Als Austauschformat zwischen den einzelnen Modulen wird ebenfalls ein XML-basiertes Dateiformat eingesetzt. Hier bietet es sich an, dass bereits definierte Format der OnToBau-Representation-Language um Elemente zu erweitern, die den vorverarbeiteten Text repräsentieren. Um eine möglichst hohe Erweiterbarkeit zu garantieren, sollten die Preprocessing-Schritte in eigenständigen Modulen definiert werden. Da bestimmte Module nicht zu einer beliebigen Zeit ausgeführt werden können, sondern abhängig von den Ergebnissen vorheriger Module sein können, definiert jedes Modul, welche Vorbedingungen erfüllt sein müssen, damit dieses ausgeführt werden kann. So wäre denkbar, dass Modul B erst ausgeführt werden kann, wenn zuvor Modul A den Text verarbeitet hat. Durch diese sequentielle Schaltung der einzelnen Module entsteht eine Pipeline. Da es denkbar ist, dass Module verschieden miteinander kombiniert werden können, ist es möglich spezialisierte Pipelines zu erstellen (zum Beispiel eine Pipeline die ein schnelles Preprocessing ermöglicht, eine Pipeline die spezielle Module für HTML-Seiten enthält usw.), die unterschiedliche Aufgaben erfüllen.

Zur Realisierung des Preprocessings wird eine erweiterte Variante des Pipes-and-Filters-Entwurfsmuster eingesetzt. Dieses Entwurfsmuster teilt das Architektursystem in sequentiell zu verarbeitende Teilschritte auf [BMR+96]. Die einzelnen Phasen werden Filter genannt, der Austausch der Daten zwischen den einzelnen Filtern erfolgt über Pipes. Diese können Daten zudem puffern, falls der folgende Filter für eine Datenaufnahme noch nicht bereit ist. Abbildung 37 zeigt den schematischen Ablauf eines Compilers in einer Pipes-and-Filters-Systemarchitektur.

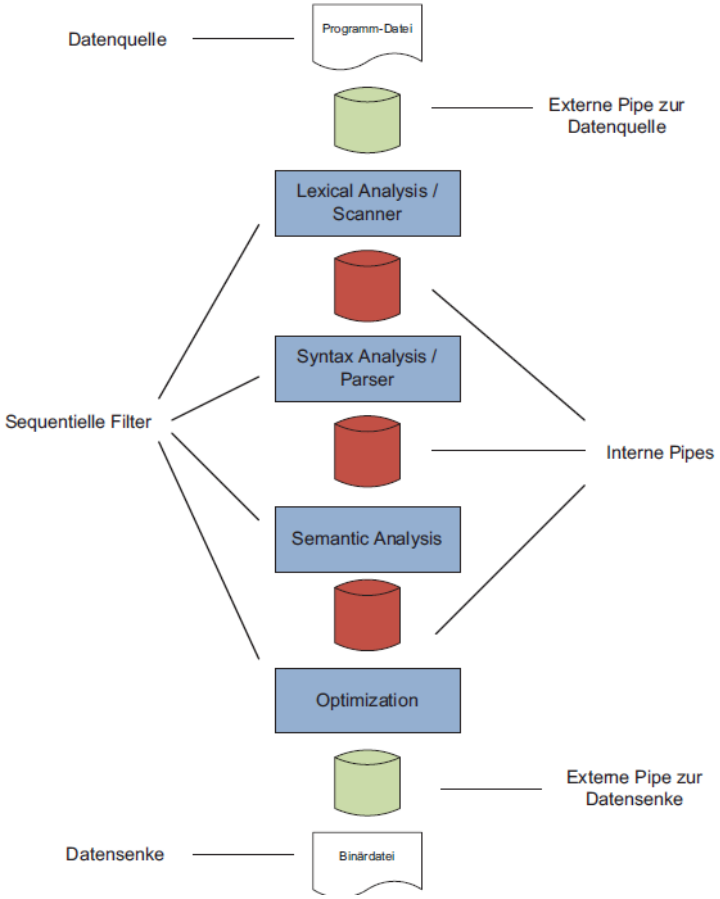


Abbildung 37: Das Pipes-and-Filters-Entwurfsmuster am Beispiel eines Compilers

Die Aufgabe eines Filter ist spezialisiert auf die kontinuierliche Verarbeitung der Eingabedaten in entsprechenden Ausgabedaten. Diese Verarbeitung der Daten kann zum Beispiel Aktivitäten wie das Anreichern von neuen Informationen, das Transformieren oder das Verfeinern von bestehenden Informationen umfassen. Durch Neukombinieren der Filter lassen sich Anwendungen mit unterschiedlichen Verhalten erzeugen. Die Eingabe in das Architektursystem erfolgt über Datenquellen (z.B. Dateien) und die Ausgabe in eine Datensenke (z.B. Datenbank, Datei, Bildschirm usw.).

Filter können unterschieden werden in aktive und passive Filter. Ein passiver Filter muss durch den Aufruf einer Funktion aktiviert werden, was entweder dadurch erfolgen kann, dass der Vorgänger Daten an den nächsten Filter weiterreicht (Push-Mechanismus) oder dass der Nachfolger Daten bei dem Vorgänger anfordert (Pull-Mechanismus) (vgl. [BMR+96] und [VBdT95]). Ein aktiver Filter dagegen startet die Verarbeitung eigenständig, vergleichbar mit einem Prozess der ständig die Pipeline überwacht.

Bei der Realisierung dieses Entwurfsmusters kann man zwei Varianten bezüglich der Verwendung der Pipes unterscheiden. Variante 1 ist in Abbildung 38 dargestellt und zeigt eine Pipeline, in der lediglich Push-Mechanismen zum Einsatz kommen. Jeder Filter übergibt nach Beendigung seiner Verarbeitung die Daten an den nächsten Filter und aktiviert diesen dadurch. Die Übergabe der Daten erfolgt durch einen direkten Aufruf einer entsprechenden Methode (write) des folgenden Filters. Explizite Pipes existieren in dieser Variante nicht.

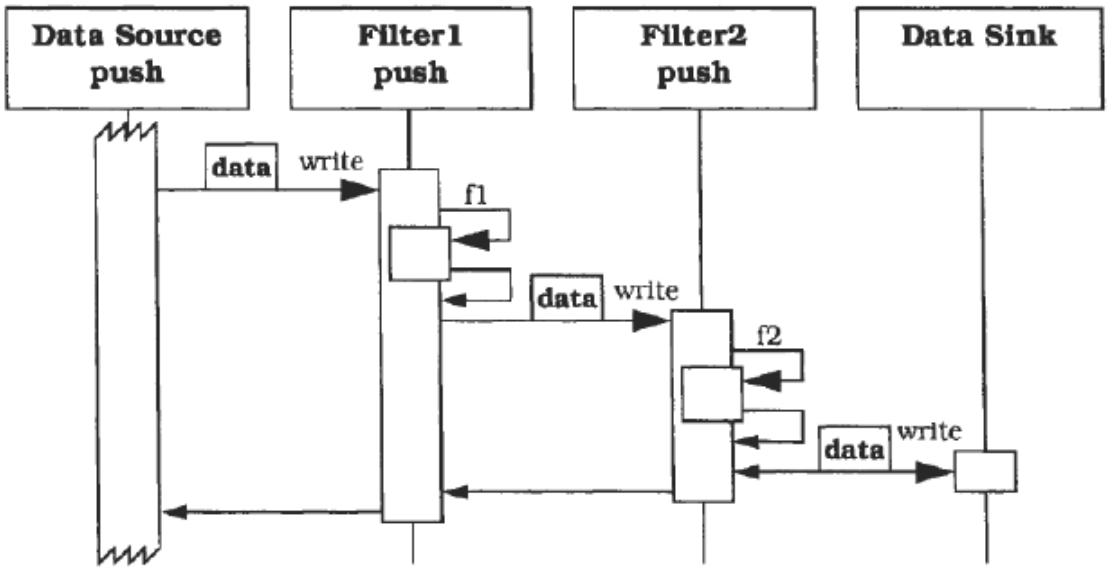


Abbildung 38: Variante 1 verwendet keine expliziten Pipe-Komponenten

Nachteil der Variante 1 ist, dass durch einen expliziten Aufruf von Methoden zur Datenübergabe die Erweiterbarkeit des Systems erschwert wird, da sich neue Filter nicht ohne Anpassung an bestehenden Filtern einfügen lassen. Des Weiteren wird ein Neukombinieren der einzelnen Filter verhindert. Dieser Nachteil könnte durch die Vorgabe eines Interfaces, in dem die Methode zur Datenübergabe definiert ist, behoben werden. Dieses Interface müsste von jedem Filter implementiert werden und ein Austauschen der Filter ermöglichen. Pipes haben aber zudem die Aufgabe Daten zu puffern, falls der Zielfilter gerade aktiv ist und keine weiteren Daten aufnehmen kann. Diese Pufferung müsste

entsprechend in den Filtern realisiert werden, was zur Folge hat, dass man Filter implementiert, die die Aufgaben einer Pipe übernehmen.

Wenn die oben angesprochenen Nachteile in dem zu implementierenden System von Bedeutung sind, dann sollte Variante 2 zum Einsatz kommen, die in Abbildung 39 dargestellt ist. Zwischen den jeweiligen Filtern werden Pipes zur Weitergabe der Daten an den nächsten Filter eingesetzt, die ebenfalls über die Möglichkeit der Pufferung verfügen. In diesem Beispiel liest Filter 1 Daten aus der Datenquelle ein und verarbeitet diese.

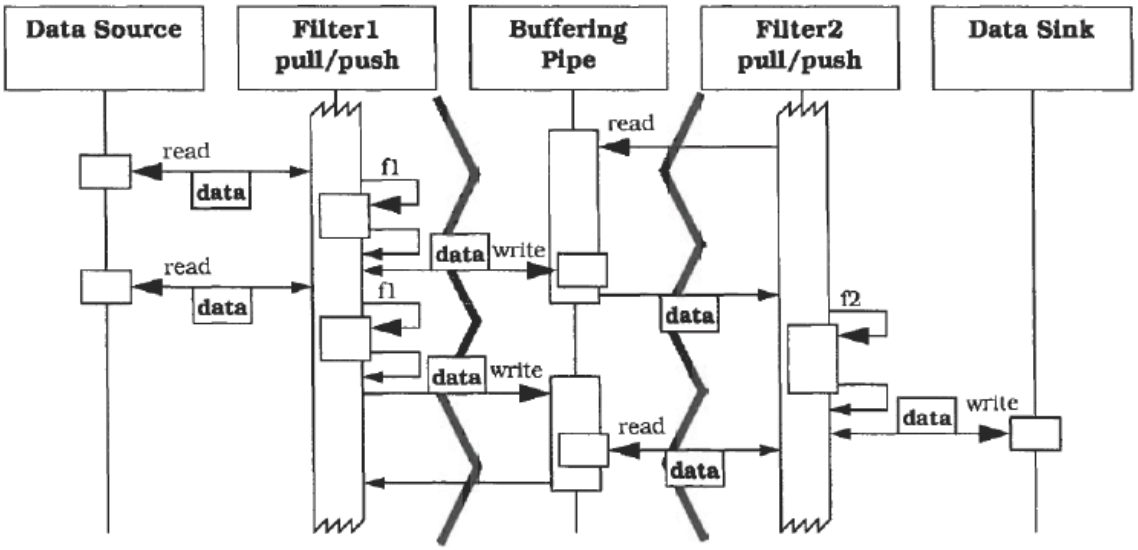


Abbildung 39: Variante 2 verwendet Pipe-Komponenten, über die die Daten an den nächsten Filter weitergegeben werden.

Anschließend schreibt er die Daten in die Pipe. Filter 2 ist ein aktiver Filter und hatte bereits zu Beginn die Pipe nach Daten abgefragt. Da zu diesem Zeitpunkt noch keine Daten vorhanden waren, muss der Filter in einen Wartezustand gehen. Nachdem neue Daten in der Pipe vorhanden sind, werden diese von Filter 2 entnommen und nach der Verarbeitung in die Datensenke geschrieben.

Die Vorteile dieser Variante liegen in der losen Kopplung der einzelnen Filter untereinander. Da den Filtern nicht bekannt ist, welcher in der Pipeline als nächstes folgt, ist ein Verändern der Reihenfolge (auch zur Laufzeit) möglich. Um eine möglichst hohe Flexibilität des Systems zu erreichen, wird in dieser Arbeit eine Architektur in Anlehnung an die zweite Variante konzeptioniert.

Zusammenfassend können die Vorteile des Pipes-and-Filter-Entwurfsmusters wie folgt beschrieben werden:

- **Erweiterbarkeit**:
  Das Austauschen und Hinzufügen von neuen Filtern ist jederzeit möglich. Durch verschiedene Kombinationsmöglichkeiten der Filter können unterschiedliche Anwendungsfälle realisiert werden.
- **Wiederverwendbarkeit**:
  Durch das Aufteilen des Gesamtsystems in mehrere unterschiedliche Filter wird eine hohe Kohäsion erreicht. Das fördert die Überschaubarkeit des Programmcodes und ein Einsatz der Filter in unterschiedlichen Kontexten wird ermöglicht.

- **Variable Ein- und Ausgabe**:
  Da Filter für die Ein- und Ausgabe in das System zuständig sind, ist es möglich, unterschiedliche Datenquellen und -senken auch noch nachträglich dem System hinzuzufügen.

Ein Nachteil des Entwurfsmusters beruht auf der fehlenden Möglichkeit, Bedingungen an Filter bezüglich der sequentiellen Reihenfolge zu knüpfen. Die lose Kopplung der Filter ermöglicht einen hohen Grad an Flexibilität bezüglich der Kombinationsmöglichkeiten der Filter. Es ist jedoch denkbar, dass bestimmte Kombinationen keinen funktionalen Nutzen haben bzw. sogar zu einer Kombination führen, die nicht ausführbar ist. Auch wenn das Ziel darin besteht, unabhängige Filter zu implementieren, so lässt sich eine logische Reihenfolge der Filter kaum vermeiden. Dieses Problem lässt sich vernachlässigen, wenn die Pipelines von einem Entwickler vorher implementiert werden und dieser somit für eine korrekte Sequenz Sorge zu tragen hat. Eine dynamische Anpassung bzw. Neukombinierung der Filter, möglicherweise durch den Endanwender, erfordert jedoch von dem System entsprechende Überprüfungen bezüglich der Korrektheit. Zum Beispiel kann ein Segmentierungsfilter in der Pipeline erst dann seine Aufgabe erfüllen, wenn zuvor ein Filter den zu segmentierenden Text aus einer Datenquelle eingelesen hat. Dagegen kann es aber auch Filter geben (z.B. Filter die lediglich für Debugging-Zwecke in die Pipeline eingebaut werden), die keine Vorbedingungen an die zu verarbeitenden Daten setzen.

Das Pipes-and-Filters-Entwurfsmuster wurde dahingehend erweitert, dass eine Modellierung von Abhängigkeiten zwischen den Filtern möglich ist, die von einer Kontrollinstanz beim Erstellen und Ändern der Pipeline überprüft wird. Insgesamt können drei Abhängigkeiten unterschieden werden:

- **autonom**:
  Dies sind Filter die keinerlei Abhängigkeiten zu anderen Filtern haben. Diese können innerhalb der Pipeline völlig frei und unbeschränkt an beliebigen Positionen eingefügt werden.
- **direkte Abhängigkeit**:
  Diese Filter sind von einem Filter abhängig, der in der Pipeline als direkter Vorgänger vorhanden ist. Darunter fallen zum Beispiel Filter, die eine gemeinsame Aufgabe erfüllen, die jedoch auf einzelne Teilfilter aufgeteilt werden kann. Diese Filter können nur zusammen innerhalb einer Pipeline stehen.
- **sequenzielle Abhängigkeit**:
  Diese Filter haben keine direkte Abhängigkeit zu einem anderen Filter. Ein Filter F2, der eine sequenzielle Abhängigkeit zu einem Filter F1 besitzt, darf jedoch innerhalb des Datenflusses erst nach der Ausführung von F1 eingebaut werden.

Abbildung 40 zeigt das Klassendiagramm des Preprocessing-Moduls. Das Konzept realisiert das zuvor vorgestellte Pipes-and-Filter-Entwurfsmuster und erweitert dieses um zusätzliche Funktionalitäten. In dem Konzept stellen die Filter die zentrale Komponente dar. Die Klasse AbstractFilter gibt die Schnittstelle vor, die allen konkreten Filtern zugrunde liegt. Diese sieht vor, dass jeder Filter über einen Namen verfügt und die Methode execute implementieren muss. Da aktive Filter zum Einsatz kommen, müssen diese als eigenständige Prozesse realisiert werden. Abhängig von der Programmiersprache kann dies zum Beispiel durch das Erweitern einer Thread-Klasse erfolgen (im UML-Diagramm anhand der Programmiersprache Java dargestellt).

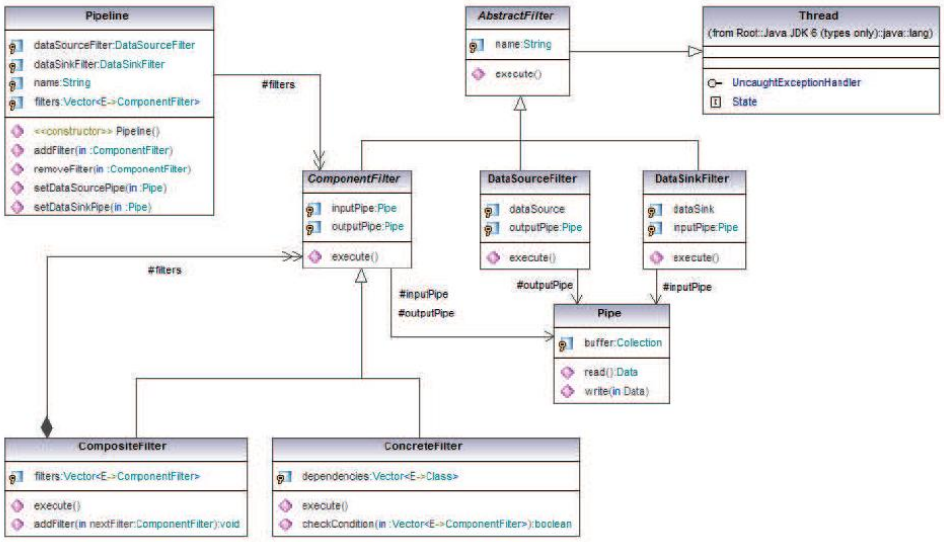


Abbildung 40: UML-Klassendiagramm des Preprocessing-Moduls in OnToBau

Ein spezieller Filter wird zum Verbinden der Pipeline mit der externen Datenquelle (DataSourceFilter) und der Datensenke (DataSinkFilter) eingesetzt. Diese beiden Filter unterscheiden sich von den übrigen Filtern dahingehend, dass sie über keine ein- bzw. ausgehende Pipeline verfügen, sondern entsprechend über eine Anbindung an die Datenquelle/-senke verfügen. Des Weiteren sind diese Filter für das Codieren der Daten in das interne Austauschformat bzw. für das Decodieren in das Format der Datensenke zuständig.

Filter innerhalb der Pipeline werden unter der abstrakten Klasse ComponentFilter zusammengefasst. Diese verfügen zusätzlich über eine ein- und ausgehende Pipe zum Empfangen und Versenden der Daten. Diejenigen Filter, die über eine sequenzielle Abhängigkeit zu einem oder mehreren anderen Filtern verfügen, werden mittels der Klasse ConcreteFilter realisiert. Diese Filterklasse verfügt zusätzlich über eine Eigenschaft dependencies, in der die Klassennamen anderer Filter gespeichert sind, die innerhalb der Pipeline vor diesem Filter stehen müssen. Diese Abhängigkeiten können entweder hart-codiert sein oder über eine externe Konfigurationsdatei gepflegt werden. Autonome Filter werden ebenfalls über die ConcreteFilter-Klasse realisiert, indem die dependencies-Eigenschaft leer bleibt. Letztlich werden zusammengesetzte Filter mit Hilfe des Composite-Entwurfsmusters ermöglicht. Dazu wurde eine weitere Klasse CompositeFilter eingefügt. Diese verfügt über eine Collection, in der weitere Filter enthalten sind. Das Composite-Entwurfsmuster ermöglicht es somit Filter, die aus anderen Filtern zusammengesetzt sind, zu realisieren.

Eine Pipe besteht aus einem Puffer, der als Collection realisiert werden kann. Zudem hat jede Pipe Methoden zum Lesen und Schreiben von Daten. Die Klasse Pipeline ist für die Kombination der einzelnen Filter zuständig. Sie stellt Methoden zum Hinzufügen und Entfernen von Filtern bereit und übernimmt die Validierung der Filterkombination.

Als Datenquelle sollen die, durch die Dokumentenkonverter erstellten, OnToBau-Representation-Language Dateien (ORL-Files) dienen. Die ORL-Files enthalten den extrahierten Text, der in mehreren Schritten durch die Filter verarbeitet werden soll. Da die ORL-Files in einer XML-basierten Sprache vorliegen, wird als internes Austauschformat eine

Document Object Model (DOM) Repräsentation eingesetzt. Das DOM ist die Spezifikation einer Schnittstelle, mit der es ermöglicht wird, auf die Inhalte von XML-Dokumenten zuzugreifen. Das DOM repräsentiert das XML-Dokument als hierarchische Baumstruktur, die aufgrund der Vater-Kind-Beziehungen der XML-Elemente aufgebaut wird. Viele Programmiersprachen bieten bereits vorgefertigte Bibliotheken zum Parsen von XML-Dokumenten, Manipulieren des DOM-Baumes etc. an.

Die Aufgabe des DataSourceFilters besteht somit darin, ORL-Files einzulesen und zu parsen. Die daraus resultierende Baumstruktur wird anschließend von Filter zu Filter weitergereicht und entsprechend verarbeitet. Der DataSinkFilter persistiert die Daten erneut als XML-Dokument.

In diesem Forschungsvorhaben wurden beispielhaft folgende Filter realisiert:

- **SegmentationFilter**:
  Unter Segmentierung versteht man die Zerlegung eines Textes in einzelne Segmente, die in einem bestimmten Sinne abgeschlossen sind (z.B. Wort - Satz - Abschnitt - Themenblock usw.). Obwohl in den westlichen Sprachen spezielle Zeichen zur Trennung eingesetzt werden (z.B. Leerzeichen, Interpunktionszeichen, Zeilenumbrüche usw.), ist die Segmentierung problematisch. Während die Erkennung von Wortgrenzen mit Hilfe des Leerzeichens eindeutig ist, so ist eine Satzgrenzenerkennung aufgrund der Ambiguität der Interpunktionszeichen schwierig. So wird der Punkt u.a. auch für die Kennzeichnung von Positionszahlen („13. Januar“) oder nach Abkürzungen („Prof. Dr. Meier“) eingesetzt. Die Satzgrenzenerkennung ist ein Klassifizierungsproblem, in dem für jedes Satzzeichen entschieden werden muss, ob es zu der Klasse der Satzendezeichen gehört oder nicht. Dazu werden Regeln definiert, die eine Disambiguierung ermöglichen (z.B. nachfolgendes Wort beginnt mit Großbuchstaben oder Präfix vorheriges Wort enthält einen Punkt). Unterstützend werden ebenfalls spezielle Listen mit gängigen Akronymen eingesetzt. Die Dokumentenkonverter liefern den extrahierten Text ohne bereits eine Segmentierung in Wörter, Sätze und Paragraphen durchgeführt zu haben. Deshalb sollte die Segmentierung in der Pipeline immer als erster Filter eingebaut sein, wenn es sich bei der zu verarbeitenden Ressource um ein Textdokument handelt.
- **StopwordFilter**:
  Stoppwörter sind innerhalb eines Dokumentes in der Regel keine Informationsträger, sondern erfüllen lediglich syntaktische Funktionen (wie z.B. Artikel, Präpositionen, Konjunktionen usw.). Die Eliminierung von Stoppwörtern kann den Dokumenteninhalt um 20%-30% reduzieren und führt damit zu einem geringeren Speicheraufwand und einer besseren Performance der anschließenden Module. Zur Eliminierung kommen meistens spezielle Stoppwortlisten zum Einsatz. Diese werden entsprechend manuell gepflegt, können jedoch auch automatisch erzeugt werden, wenn ein entsprechend großer Korpus an Beispieldokumenten zur Verfügung steht (vgl. [Net97]).
- **TokenizerFilter**:
  Die Tokenisierung ist eine lexikalische Analyse eines Textes und die dabei stattfindende Zerlegung in einzelne Tokens. Die Segmentierung gehört damit ebenfalls zur lexikalischen Analyse. Die Aufgabe des Tokenizer im Kontext dieser Arbeit soll dagegen eine Klassifizierung der Wörter in zuvor definierte Klassen vornehmen. Darunter fallen zum Beispiel Datumsangaben, Preise, Maßangaben,

Telefonnummern usw. Diese Markierung der einzelnen Wörter kann in den anschließenden IE-Prozessen direkt verwendet werden, ohne das eine nachträgliche Analyse der Wörter erfolgen muss. Da während des Preprocessings der komplette Text iteriert werden muss, bietet es sich an, diese Wortklassifizierung bereits in diesem Schritt durchzuführen.

- **NormalisationFilter**:
  Dieser Filter führt eine Normalisierung durch, damit anschließende Verarbeitungen und mögliche Vergleiche zwischen Dokumenten erleichtert werden. So werden zum Beispiel alle Datums-, Preis- und Maßangaben auf eine einheitliche Darstellung gebracht. Des Weiteren kann hierunter auch die Normalisierung auf Groß- bzw. Kleinschreibung erfolgen, wenn dies im entsprechenden Kontext von Vorteil sein kann.
- **MorphixFilter**:
  Der MorphixFilter gehört ebenfalls zu der Kategorie der Normalisierungsfilter und führt eine Stammformreduktion der Wörter durch. Dieses morphologische Verfahren wird auch als Stemming bezeichnet und führt die verschiedenen morphologischen Varianten eines Wortes auf den gemeinsamen Wortstamm zurück. Diese verschiedenen Varianten entstehen im Deutschen am häufigsten durch Flexion (z.B. Deklination und Konjugation), Komposition (Haus + Tür = Haustür) und Derivation (Erweitern eines Wortstamms um Suffixe oder Präfixe, z.B. frei + heit = Freiheit). Es existieren verschiedene Verfahren, die eine derartige morphologische Normalisierung ermöglichen. Lexikon-basierte Verfahren arbeiten mit einer Tabelle, die zu jedem Wort die entsprechenden Flexionen usw. verwaltet. Diese Verfahren sind sehr einfach, erfordern jedoch ein manuelles Pflegen des Lexikons. Daneben existieren algorithmenbasierte Verfahren, die mit Hilfe von morphologischen Regeln Wörter automatisch auf die Stammform zurückführen. In Sprachen mit morphologischen Besonderheiten (z.B. Stammformänderung von das Haus zu die Häuser) scheitern diese algorithmenbasierten Verfahren, da sich diese Besonderheiten nicht algorithmisch abbilden lassen. Deswegen kommen meistens Kombinationen aus lexikon- und algorithmen-basierten Verfahren zum Einsatz (vgl. [Fer03]).
- **DebuggingFilter**:
  DebuggingFilter können prinzipiell an einer beliebigen Stelle innerhalb der Pipeline eingebaut werden und zum Ausgeben des Datenstroms eingesetzt werden. Hier kann man sich unterschiedliche Varianten der Ausgabe vorstellen: von einfachen Ausgaben auf die Konsole bis hin zu detaillierten Log-Dateien. Des Weiteren wäre es denkbar, dass in einem späteren produktiven Einsatz, die Filter zur Überwachung der Pipeline eingesetzt werden könnten, um im Fehlerfall den Entwickler per E-Mail oder SMS zu benachrichtigen.
- **TransformationFilter**:
  Die Transformationsfilter sollen die verarbeitete Ressource in die interne Datenrepräsentation des IE-Systems überführen. Es wäre auch denkbar, dass bestimmte Ressourcen (z.B. Audiodateien) nicht durch das IE-System verarbeitet werden sollen bzw. können. Entsprechende Filter könnten diese Ressourcen direkt in das Wissensarchiv umleiten.

2.2.3.2 Extraktionsstrategien

Nachdem die Ressourcen mittels spezieller Dokumentenkonverter in ein allgemeines Datenmodell überführt wurden und in mehreren Preprocessing-Schritten auf die anschließende Extraktion vorbereitet wurden, wird in diesem Kapitel das Konzept der Kernkomponente des Extraktionssystems beschrieben, das Inferenzsystem. Der Ablauf der IE innerhalb des Systems ist in Abbildung 41 dargestellt. Die blauen Elemente stellen einzelne Module innerhalb des Inferenzsystems dar.

Bevor die verschiedenen IE-Strategien zum Einsatz kommen, muss für die aktuelle Ressource ermittelt werden, welche Informationen extrahiert werden sollen. Handelt es sich bei der Ressource um eine Rechnung, so sind andere Informationen von Interesse, als wenn eine Produktbeschreibung aus einem Onlinekatalog vorliegt. Während bei einer Rechnung Daten wie Rechnungsnummer, Rechnungssteller, Endbetrag, Einzelposten usw. extrahiert werden können, sind diese in einer Produktbeschreibung entweder nicht zu finden oder nicht von Interesse. Da die Entscheidung, welche Informationen extrahiert werden sollen, von dem IE-System weitestgehend automatisch getroffen werden soll, müssen zwei Voraussetzungen erfüllt sein. Das System muss erkennen, zu welchem Konzept (z.B. Rechnung, Produktbeschreibung, Fachartikel etc.) die Ressource gehört und innerhalb des Konzeptes muss hinterlegt sein, welche Informationen extrahiert werden sollen. Diese Aufgabe übernimmt das Modul Document Classification, welches in Kapitel 3.5.1 beschrieben wird.

Sollte eine automatische Dokumentenklassifikation nicht möglich sein, so muss unter Umständen eine manuelle Klassifikation durch den Benutzer erfolgen. In einem späteren Verlauf des Projektes wäre denkbar, dass an diesen Prozess eine Lernkomponente angeschlossen wird, die automatisch die Klassifikationsergebnisse für spätere Durchläufe verbessert.

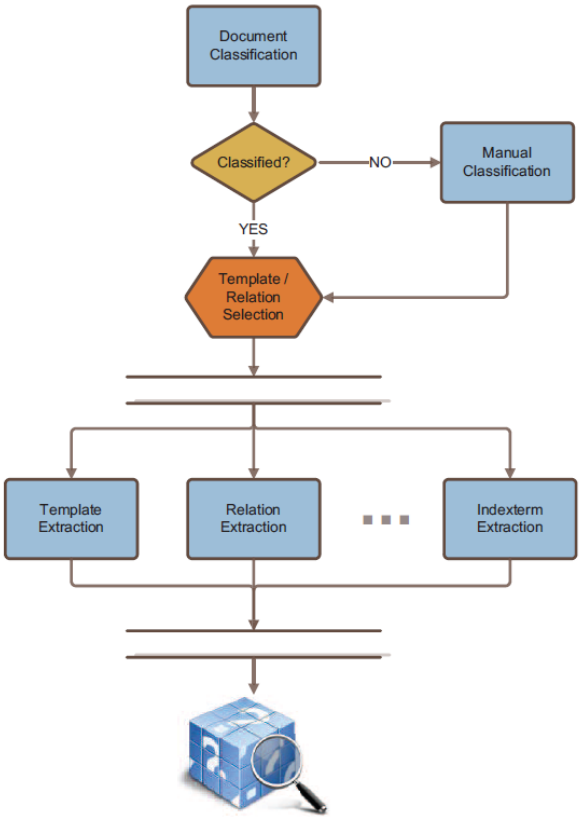


Abbildung 41: Ablaufdiagramm des Extraktionssystems

Nachdem die Klassifikation der Ressource eine oder mehrere Konzepte zugeordnet hat, erfolgt anschließend eine Auswahl der zu extrahierenden Informationen. Bei diesen Informationen handelt es sich zum Einen um explizite Informationen, die sich aus den Daten der Ressource ergeben und zum Anderen um implizite Informationen, die sich aus der Ermittlung von Beziehungen zu anderen, bereits im Wissensarchiv vorhanden Ressourcen ergeben. Welche expliziten Informationen extrahiert werden sollen, wird mit Hilfe von zu füllenden Templates definiert. Diese Templates sind den Konzepten zugeordnet, wobei zu einem Konzept mehrere Templates vorhanden sein können. Welche Relationen zu anderen Ressourcen bestehen können wird ebenfalls innerhalb der Konzepte modelliert. Kapitel 3.5.2 und Kapitel 3.5.3 behandeln die Template- und Relationsextraktion.

Nach Auswahl der zu extrahierenden Daten und Relationen folgen verschiedene Module, die unabhängig voneinander ausgeführt werden können und, abhängig von der Ressource, auch optional sein können. Die Architektur des Systems sollte ein einfaches Erweitern um neue Module ermöglichen. Die bereits angesprochenen Module zur Template- und Relationsextraktion können nur ausgeführt werden, wenn innerhalb des Wissensarchivs entsprechende Templates bzw. Regeln modelliert wurden. Für ein Information Retrieval ist die Ermittlung von Indextermen sinnvoll, was ebenfalls von einem eigenständigen Modul durchgeführt werden kann (siehe Kapitel 3.5.4).

Die Informationen, die von den verschiedenen Modulen gewonnen werden, werden in das Wissensarchiv integriert. Das Wissensarchiv ist dabei eine Kombinationen aus den definierten Ontologien, Taxonomien, Templates usw. und den extrahierten Informationen (i.d.R. in Form von gefüllten Templates). Die ermittelten Relationen zwischen den einzelnen Wissenseinheiten spannen letztlich ein Wissensnetzwerk auf.

2.2.3.3 Klassifikation von Dokumenten

Die Klassifikation der Ressource in eine domänen-spezifische Taxonomie stellt den Anfang innerhalb des Inferenzsystems dar. Erst nachdem die Ressource einem oder mehreren Konzepten zugeordnet werden konnte, lassen sich die anschließenden Module sinnvoll ausführen. Der erste Schritt besteht demnach darin eine Ontologie zu modellieren, die eine Sicht auf den Anwendungsbereich der Domäne darstellt. Die Konzepte werden taxonomisch aufgebaut und beinhalten die Kriterien, die eine Ressource erfüllen muss, um in dieses Konzept klassifiziert zu werden.

Viele Klassifikationsverfahren sind schlüsselwort-basiert und versuchen Dokumente anhand von Clustering-Verfahren den entsprechenden Konzepten zuzuordnen. Das heißt es werden Ähnlichkeitsmaße bestimmt und Dokumente mit einer hohen Ähnlichkeit zu einem Cluster zusammengefasst. Ein Cluster würde in diesem Kontext einem Konzept innerhalb der Ontologie entsprechen. Dokumente, die über einen hohen textuellen Anteil verfügen und gehäuft diskriminierende Schlüsselwörter verwenden, lassen sich mit Hilfe dieser Verfahren leicht klassifizieren. Problematisch sind Dokumente, die über eine geringe Anzahl dieser Schlüsselwörter verfügen. So verfügen Dokumente, die in das Konzept Rechnung fallen, über eine geringe Anzahl an signifikanten Schlüsselwörtern. Zudem lassen sich Rechnungen untereinander nicht mit Hilfe eines Clusteringverfahrens zusammenfassen, da es kaum Gemeinsamkeiten auf Wortebene gibt (es sei denn die Rechnungen sind von derselben Firma und es wurden ähnliche Produkte bestellt).

Aus diesem Grund sieht das Konzept vor, weitere Kriterien für die Klassifikation von Dokumenten einzuführen. Es werden folgende Kriterien unterschieden:

- Schlüsselwörter/-phrasen:
  Viele Dokumente enthalten Schlüsselwörter bzw. -phrasen, die Hinweise auf die mögliche Zugehörigkeit zu einem Konzept geben. Schlüsselwörter sind jedoch nicht immer ausreichend, wenn diese auf Grund geringer Häufigkeit innerhalb des Dokumentes keine ausreichende Evidenz liefern.
- Dokumentenstruktur:
  Die Untersuchung der Dokumentenstruktur ist aussagekräftiger. Schlüsselwörter kommen hier jedoch unterstützend zum Einsatz, wenn zum Beispiel numerische Angaben interpretiert werden sollen (z.B. als Rechnungsnummer). Unter der Dokumentenstruktur werden Merkmale wie das Vorkommen von Adressen, Preisangaben, Bankverbindungen usw. verstanden. Hierunter fallen aber auch abstraktere Merkmale wie das Verhältnis von Wörtern zu Bildern, die durchschnittliche Satzlänge usw.
- Regeln:
  Damit eine Ressource zu einem Konzept gehört, reicht es manchmal nicht aus, dass bestimmte Begriffe vorhanden sind oder die Dokumentenstruktur besondere Kriterien erfüllt. Ressourcen, die zum Beispiel zum Konzept Kleingeräteantrag gehören, dürfen lediglich über einen Gesamtbetrag von kleiner 15000 Euro verfügen. Regeln können zudem über boolesche Operatoren zu komplexeren Regeln zusammengesetzt werden.

Es fiel auf, dass unter Umständen während der Klassifikation bereits teilweise eine Extraktion von Informationen stattfand. Die hierbei gewonnenen Informationen können jedoch an die anschließenden Module zur Informationsextraktion weitergereicht werden, so dass ein doppelter Aufwand nicht erforderlich ist.

Die Sprache zur Modellierung der Konzepte sollte die Möglichkeit bieten, diese taxonomisch anzuordnen und die oben genannten Kriterien abbilden zu können. Da eine Ressource die Kriterien von verschiedenen Konzepten zugleich erfüllen kann, muss ein Bewertungssystem eingeführt werden, anhand dessen entschieden werden kann, zu welchem Konzept eine Ressource zugeordnet wird. Das Bewertungssystem sollte einfache Metriken verwenden, da Erfahrungen aus früheren Projekten gezeigt haben, dass manuelle Änderungen an den Bewertungen wesentlich einfacher nachvollziehbar sind. Aus diesem Grund wird ein additives Punktesystem vorgeschlagen. Ressourcen die Kriterien erfüllen erhalten Punkte, die sich zu einer Gesamtpunktzahl addieren. Zudem soll die Möglichkeit vorhanden sein, einzelne Kriterien stärker zu gewichten als andere. Ein Schwellwert entscheidet über die Klassifizierung einer Ressource zu einem Konzept.

Das Generieren der Konzepte und Kriterien kann durch einen Experten erfolgen und während der Testphase durch fortwährende Evaluierung stetig verbessert werden. Ein anderer Ansatz wäre, dass mit Hilfe einer Trainingsmenge an Beispieldokumenten und entsprechender Lernverfahren die Konzepte erstellt werden. So zeigt [KCL03] wie mittels eines naiven Bayes-Klassifikator Schlüsselwörter erlernt werden können und [MY07] klassifizieren Textdokumente mit Hilfe eines neuronalen Netzes.

Abbildung 42 zeigt beispielhaft die Modellierung eines Konzepts zur Klassifizierung einer Kleingeräterechnung. Schlüsselwörter und Phrasen werden aufgelistet und erhalten eine Punktezahl zugeordnet, die vergeben wird, wenn diese Schlüsselwörter bzw. Phrasen innerhalb der Ressource erscheinen. Hier wäre auch denkbar, dass man bestimmte Schlüsselwörter oder Phrasen stärker gewichtet als andere. Außerdem wäre eine

Unterscheidung in Pflicht- und optionale Wörter unter Umständen sinnvoll. Zur Modellierung von Phrasen könnten zusätzlich reguläre Ausdrücke zum Einsatz kommen. Viele Phrasen haben einen ähnlichen Aufbau und unterscheiden sich lediglich in einigen Worten (z.B. Die Arbeiten wurden von Februar bis Juni 2006 ausgeführt oder Der Auftrag

```
Concept KLEINGERÄTERECHNUNG

   :keys  „Rechnung“, „Mwst“, „Endbetrag“  :points 20

   :phrases   „Bitte zahlen Sie zum“  :points 50

   :rules
      :rule1   isFinalAmount(X) AND x < 500   :points 100
      :rule2   isAddresser(X) OR isBankAccount(Y)   :points 100

   :threshold 175
```

Abbildung 42: Deskriptive Beschreibung eines Konzeptes zur Klassifizierung

wurde von Mai - August 2006 bearbeitet). Für die Modellierung der Regeln wird eine prädikatenlogische Darstellung vorgeschlagen. Da die einzelnen Prädikate innerhalb des Inferenzsystems ausgewertet werden müssen, kann dadurch jedoch keine unabhängige Beschreibung der Konzepte von der Implementierung erreicht werden. Werden zur Modellierung von Regeln neue Prädikate notwendig, so müssen diese im Inferenzsystem implementiert werden. Zu jedem Konzept sollte ebenfalls ein Schwellwert definiert werden, der überschritten werden muss, damit eine Zuordnung zu diesem Konzept erfolgt.

Es ist nicht bekannt, ob eine applikationsunabhängige Modellierungssprache existiert, welche die oben genannten Anforderungen syntaktisch beschreiben kann. Es müsste demnach eine eigene deskriptive Sprache entwickelt werden. Diese Problemstellung lässt sich in dieser Arbeit jedoch nicht abschließend lösen, sollte jedoch Ausgangspunkt für weitere Untersuchungen und Arbeiten sein.

2.3 *Arbeitspaket 3: Prozessmodellierung*

2.3.1 Untersuchung Prozessmodellierungsformalismen

Verschiedene Beschreibungsformalismen für Prozesse wurden untersucht und evaluiert Ein Bewertungskriterium ist die wissensbasierte Steuerbarkeit durch den Rückgriff auf ontologisches Wissen. Darüber hinaus ist auch hier die Nachvollziehbarkeit und Änderbarkeit durch den Benutzer ein Untersuchungsgegenstand.

Festgestellt wurde, dass eine kaum überschaubare Vielfalt an Prozessmodellierungssprachen gibt. Nahezu jedes Modellierungswerkzeug verfügt über eine eigene, häufig nicht standardisierte Sprache.

Im Rahmen dieses Forschungsvorhaben wurden mehrere Modellierungssprachen untersucht, die in den folgenden Kapiteln kurz angeschnitten werden. Ausführliche Beschreibungen dieser Sprachen finden sich in zahlreichen Fachliteraturen zum Thema Geschäftsprozessmodellierung.

2.3.1.1 UML

Die Unified Modeling Language ist keine reine Geschäftsmodellierungssprache, sondern bietet eine Vielzahl an graphischen Notationen zur Beschreibung von Softwaresystemen. Aktivitätsdiagramme finden hierbei Anwendung zur Modellierung von Prozessen und Aktivitäten.

UML ist jedoch nur bedingt zur Geschäftsmodellierung geeignet, da

- keine Modellierungselemente zur Abbildung der Organisation vorhanden sind,
- keine Zuordnung von Akteuren zu Aktivitäten erfolgen kann,
- eine wissensbasierte Steuerbarkeit nicht ontologisches Wissen nicht möglich ist.

2.3.1.2 EPK

Die von im Rahmen des ARIS-Konzeptes der Firma IDS Scheer entwickelten Ereignisgesteuerten Prozessketten (EPK) sind das in Deutschland wohl weitverbeiteste Werkzeug zur Modellierung von Geschäftsprozessen.

Vorteile dieser Modellierungssprache sind deren Mächtigkeit, die die Modellierungen von Prozessen unter verschiedenen Aspekten ermöglicht und die weite Verbreitung. Ein großer Nachteil ist jedoch die eingeschränkte Nutzung dieser Modelle über den Anwendungsbereich der Geschäftsprozessmodellierung hinaus. Die Verknüpfung von EPKs mit ontologischem Wissen zur Ablaufsteuerung ist durch nicht ohne größere Modifikationen möglich. Im Forschungsvorhaben BauVoGrid wurde bereits ein verfolgt, der es ermöglichen soll EPK-basierte Prozesse in Ontologien abzubilden.

2.3.1.3 BPMN

Die Business Process Modeling Notation (BPMN) ist ebenfalls eine grafische Notation zur Modellierung von Geschäftsprozessen. Im Gegensatz zu vielen anderen Sprachen basiert BPMN nicht auf einem Modellierungstool bzw. Workflow Management System, sondern ist eine davon losgelöste Entwicklung der Business Process Management Initiative. Damit könnte BPMN sich auf lange Sicht erstmals zu einem Standard für die Prozessmodellierung durchsetzen.

Nachteile der Sprache sind ihre hohe Komplexität, die es einem Anwender nicht ohne weiteres ermöglicht diese zu erlernen. Die wissensbasierte Steuerbarkeit ist auch in dieser Sprache nicht möglich.

2.3.1.4 Fazit

Abschließend wurde in dem Forschungsvorhaben festgestellt, dass es bisher keine Geschäftsprozessmodellierungssprache gibt, die eine ontologie-basierte Modellierung ermöglicht und damit eine wissensbasierte Steuerung von sich aus ermöglicht. Forschungsvorhaben wie BauVoGrid verfolgen bereits Ansätze Geschäftsprozesse mittels Ontologien zu repräsentieren, was ein erste Schritte zu einer wissensbasierten Steuerung ermöglichen könnte.

Im Rahmen des OnToBau-Projektes konnte jedoch keine externe Lösung gefunden werden, die ohne einen erheblichen Modifikationsaufwand auf die Bedürfnisse einer wissensbasierten Steuerbarkeit möglich gewesen wären. Hier sehen wir für die Zukunft ein erhöhtes Forschungspotenzial.

2.3.2 Modellierung der Referenzprozesse

Da keine geeignete Prozessmodellierungssprache gefunden wurde, die den Anforderungen in OnToBau entsprochen hätten, wurde die wissensbasierte Steuerung der Prozesse nicht weiter verfolgt.

2.3.3 Integration Intelligenter Benutzeragent

Der intelligente Benutzeragent soll das Verhaltens des Mitarbeiters überwachen und ihm in bestimmen Situationen relevantes Wissen proaktiv zur Verfügung stellen. Hierzu war es in einem ersten Schritt erforderlich Möglichkeiten zu untersuchen, wie Benutzeraktionen überwachte werden können. Die hierzu erstellten Tools werden im folgenden Unterkapitel näher beschrieben.

2.3.3.1 Monitoring von Benutzerinterkationen

Viele Prozesse sind sehr wissensintensiv (z.B. die Angebotserstellung) und erfordern in verschiedenen Prozessschritten den Zugriff auf Ressourcen wie Produktkataloge, Datenbanken oder archivierte Geschäftsdokumente.

Häufig muss der Benutzer selbst nach diesen Informationen suchen, indem er entweder Suchanfragen an die Produktdatenbank stellt oder Aktenordner nach relevanten Dokumenten durchforstet. Eine Intention in OnToBau ist es, dass der Benutzer von dem System pro-aktiv mit hilfreichen Informationen zur Erfüllung seiner aktuellen Tätigkeit unterstützt wird, ähnlich wie man dies bereits aus Office-Produkten kennt.

Damit die pro-aktive Unterstützung stattfinden kann, ist jedoch vorausgesetzt, dass das System die aktuelle Intention des Benutzers erkennt (z.B. dass er gerade in Google nach einem Produkt sucht). Hierfür sind zwei Schritte notwendig:

- Es müssen Mechanismen entwickelt werden, die ein Monitoring der Benutzerinteraktionen mit der Systemumgebung ermöglichen.
- Aus den gesammelten Informationen muss eine Benutzerintention abgeleitet werden.

Das OnToBau-Projekt konnte eine praktische Realisierung aus zeitlichen Gründen lediglich für den ersten Schritt realisieren. Für Vorhersage der Benutzerintention konnten jedoch theoretische Vorarbeiten geleistet werden.

Für den ersten Schritt wurde ein System implementiert , mit Hilfe dessen die verschiedenen Benutzerinteraktionen mit der Systemumgebung überwacht und ausgewertet werden können (z.B. welche Applikation hat er geöffnet, welcher Suchbegriff wird gerade in Google eingegeben usw.). Die gesammelten Daten müssen in einer geeigneten Form aufbereitet und dem Lernverfahren zur Verfügung gestellt werden.

Das Ziel des Programms ist es, Informationen über die Aktionen eines Benutzers zu sammeln. Jedoch nur die, die für das Forschungsvorhaben wichtig sein könnten. Die gesammelten Informationen sollen dann in einem späteren Modul in Echtzeit analysiert werden.

Im OnToBau wurde das Sammeln von Informationen auf zwei Informationsquellen beschränkt:

- **Navigation im Internet**:
  Sucht ein Mitarbeiter mit Hilfe einer Suchmaschine im Internet nach wichtigen Daten, so müssen die gefundenen Webseiten oft mühselig durchsucht werden, bevor man die benötigten Informationen findet. Braucht ein anderer Mitarbeiter die gleichen Informationen, ist dieser gezwungen dasselbe tun. Um dies zu vermeiden, sollten nützliche Webseiten für Mitarbeiter gekennzeichnet werden, indem man z.B. Aufenthaltszeiten auf Webseiten analysiert.

- **Navigation im Dateisystem:**
  Navigation im Dateisystem kann oft eine mühselige Aufgabe sein. Sind die Dateien eines Benutzers schlecht organisiert, verliert er kostbare Zeit beim Suchen und somit wertvolle Arbeitszeit. Sind die Dateien dann auch noch im Netzwerk der Firma und werden von vielen Mitarbeitern verwendet, kann dies zu weiteren Problemen führen. So z.B. wenn eine Datei ohne Mitteilung umbenannt oder verschoben wurde.

Die Architektur des MonitoringTools ist in Abbildung 43 zu sehen. Das Programm aus drei Klassen und den erweiterbaren Modulen, die ihrerseits wieder aus mehreren Klassen bestehen. Jedes implementierte Modul muss von der abstrakten Klasse abgeleitet werden, damit die ermittelten Informationen korrekt an die Logger Klasse übermittelt werden können. In der Hauptklasse des Programms wird zuallererst kontrolliert, ob die benötigte Textdatei „Modul.txt“ existiert. In ihr stehen die Module die beim Programmstart gestartet werden sollen. Fehlt die Datei oder ist diese leer, kann die Anwendung nicht starten. Will man ein Modul starten, so schreibt man einfach die Namen untereinander in die Textdatei. Die jeweiligen Module werden dann falls gewollt, in einem neuen Thread gestartet.

Der Einfachheit halber wurde das Programm am Anfang als Windows Form Anwendung implementiert. Dies hat den Vorteil, dass der Code einfacher zu debuggen ist und man zusätzlich Windows Form Elemente zur Kontrolle, z.B. eine Listbox oder ein Datagrid, benutzen kann.

Die Vorteile der Verwendung von Threads sind**:**

- **Erreichbarkeit:**
  Wenn ein Prozess für eine bestimmte Aktion blockiert wird, so muss beim Einsatz von Threads nur ein Thread blockiert werden, während der Prozess ansonsten weiterarbeiten kann. Anwendungsbeispiele sind Webbrowser als Clients, die eine Anfrage an einen Server richten. Wenn in dem Beispiel nicht mit Threads gearbeitet würde, so würde dies bedeuten, dass der Webbrowser nur genau eine Anfrage bearbeiten kann. Erst wenn diese beendet ist, könnte die nächste Anfrage bearbeitet werden.
- **Effizienz:**
  Threads haben gegenüber herkömmlichen Prozessen den Vorteil, dass sie wesentlich effizienter arbeiten können, weil bei einem Wechsel der rechnenden Threads nur die Register ausgetauscht werden.

- **Ausnutzung der Mehrprozessorarchitektur:**
  Wenn eine Mehrprozessorarchitektur zur Verfügung steht, ist es grundsätzlich sinnvoll, dass mehrere Prozesse tatsächlich gleichzeitig rechnen.

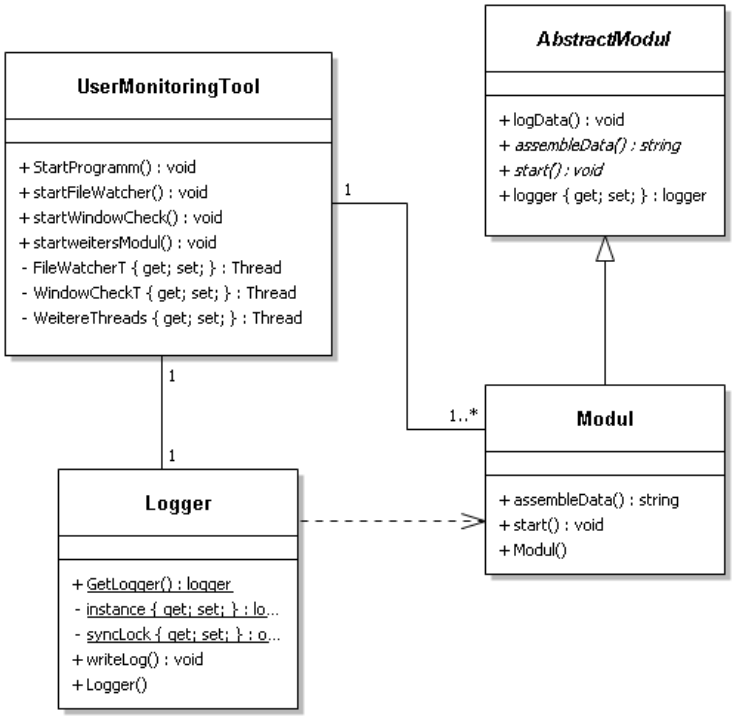


Abbildung 43: Architektur des MonitoringTools

Da das Programm aus mehreren unabhängig voneinander arbeitenden Modulen besteht, lag es nahe die Module in unterschiedlichen Threads laufen zu lassen. Nur so kann verhindert werden, dass falsche Informationen aufgrund von Ressourcen Engpässen geloggt werden. Zum Beispiel ist es beim Abfragen der aktiven Fensterinformationen unabdingbar, dass dies direkt geschieht und nicht erst nachdem ein anderer Prozess beendet wurde.

Um einen Thread zu starten, reicht es aus, eine neue Instanz des Moduls anzulegen und die Startmethode aufzurufen. Zusätzlich wird der Thread als Hintergrund-Thread deklariert. Dies bewirkt, dass dieser automatisch abgebrochen wird, falls das Hauptprogramm geschlossen wird oder abstürzt.[Mic11d]

Soll die Anwendung um ein weiteres Modul erweitert werden, so muss nur der Name des Moduls und der Hauptklasse bekannt sein. Damit ist es dann möglich die Modulabfrage im Programm so anzupassen, dass das Modul in einem zusätzlichen Thread gestartet werden kann.

Im Rahmen des Forschungsvorhabens wurden folgende Module zum Benutzermonitoring entwickelt:

- FileWatcher zur Überwachung von Dateien:
  Das Modul, mit dem Namen “FileWatcher“, dient der Überwachung von Änderungen im Dateisystem. Dieses Modul registriert wenn eine Datei umbenannt, kopiert, verschoben oder gelöscht wird. Das Öffnen von Dateien wird aufgrund von

gegebenen Einschränkungen nicht erkannt und wurde deshalb in einem separaten Modul implementiert.

- OpenWatcher zur Überwachung von Dateiöffnungen
  Das Modul „OpenWatcher“ (siehe Abbildung 44) ist dafür zuständig, das Öffnen von Dateien jeder Art zu überwachen. Ursprünglich sollte das Öffnen zusammen mit den anderen Dateiänderungen überwacht werden. Da sich die Implementierungen jedoch derart voneinander unterscheiden, wurde ein neues Modul erstellt.

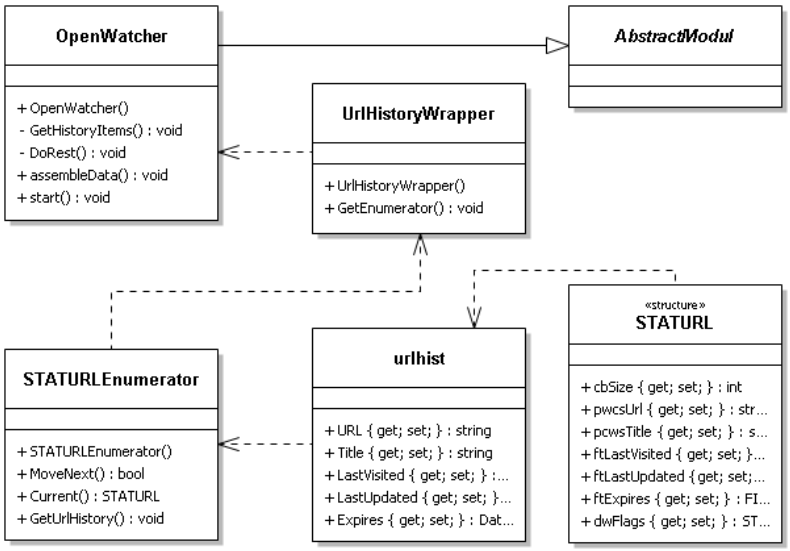


Abbildung 44: Architektur des OpenWatcher-Moduls

- WindowCheck zur Überwachung von Fensteraktivitäten
  Die Aufgabe des Moduls “WindowCheck“ ist es, zu analysieren wie ein Benutzer sich durch das Betriebssystem navigiert und welche Fenster er dafür benötigt. Mit ihm können z.B. häufig benutzte Ordner ausgemacht und schneller zugänglich gemacht werden.

- InternetCheck zur Überwachung von Internetaktivitäten
  Das Modul “InternetCheck“ (siehe Abbildung 45) ist dafür verantwortlich die besuchten Internetadressen zu loggen. Dies funktioniert mit dem Internet Explorer, Firefox und Chrome. Die Informationen können später benutzt werden, um das Surfverhalten einzelner Benutzer auszuwerten und nützliche Webseiten schneller zugänglich zu machen.

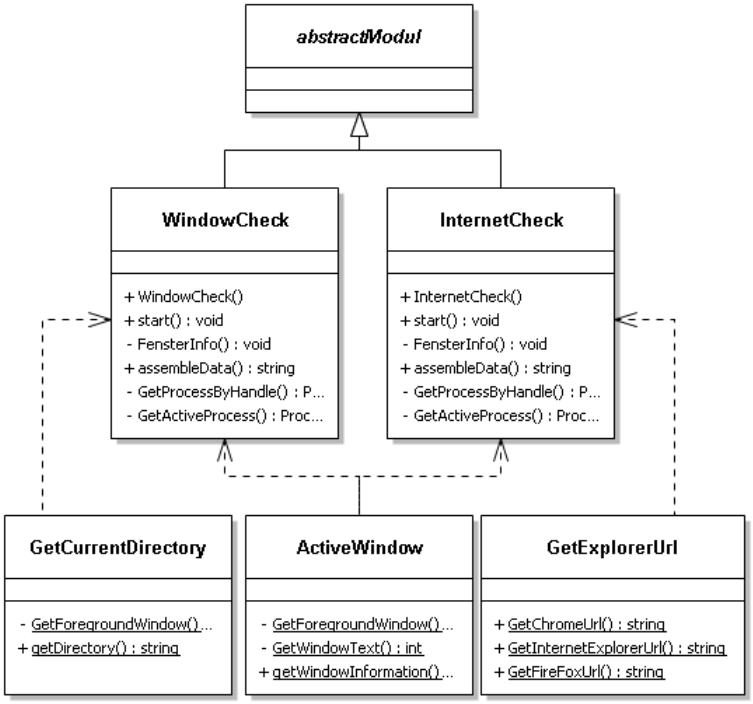


Abbildung 45: Architektur des WindowCheck und InternetCheck-Moduls

2.4 *Arbeitspaket 4: Implementierung*

2.4.1 Aufbau der Unternehmenswissensbasis

In diesem Arbeitspaket wurde die Unternehmenswissensbasis mit Hilfe von Ontologien aufgebaut. Die im Arbeitspaket 2.2.1 gewonnen Erkenntnisse bezüglich der verschiedenen Beschreibungsformalismen für Ontologien führten zu dem Ergebnisse, die Standardisierte Web Ontology Language (OWL) des W3C als Ontologiesprache zu wählen.

Insgesamt wurden zwei verschiedene Ontologien erstellt:

- **Allgemeine Ontologie zur Repräsentation von Geschäftsdokumenten**:
  Diese Ontologie beschreibt den Zusammenhang und Aufbau allgemeiner Geschäftsdokumente die in Unternehmen üblicherweise zur Anwendung kommen. da diese Ontologie unabhängig von der Anwendungsdomäne ist, ist ihre Wiederverwendbarkeit sehr hoch. Die folgende Abbildung 46 zeigt die hierarchische Gliederung der Ontologie.

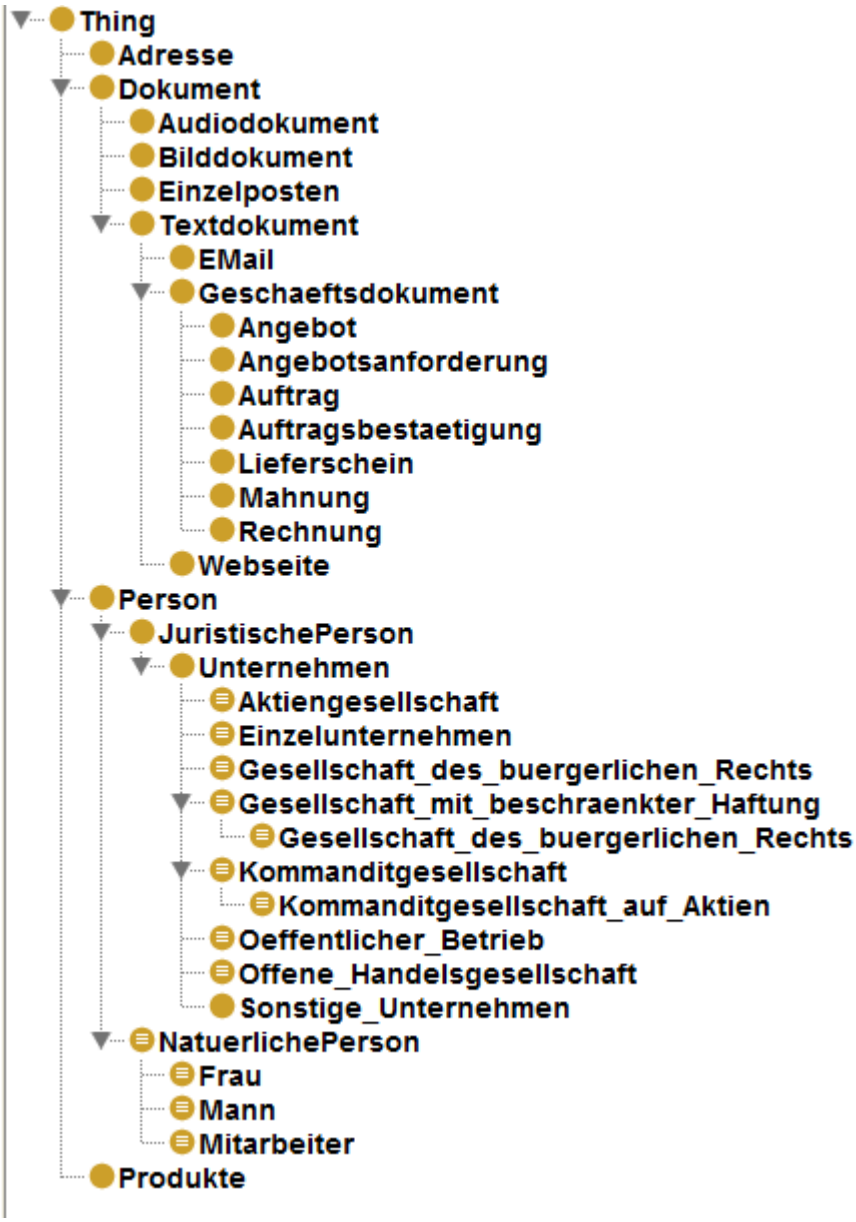


Abbildung 46: Taxonomischer Aufbau der Ontologie für Geschäftsdokumente

- **Fachspezifische Ontologie zum Bereich Bauwesen**:
  Daneben wurde zudem eine fachspezifische Ontologie erstellt, die die Anwendungsdomäne des Bauwesens modelliert. Zu diesem Zweck wurde sich auf den Teilbereich Heizung und Sanitär beschränkt, da der Projektpartner Schottler den Großteil der Dokumente zu diesem Bereich zu Verfügung gestellt hatte.
  Abbildung 47 zeigt einen Ausschnitt aus der fachspezifischen Ontologie. Initial wurde diese Ontologie von Hand mit konkreten Instanzen gefüllt.

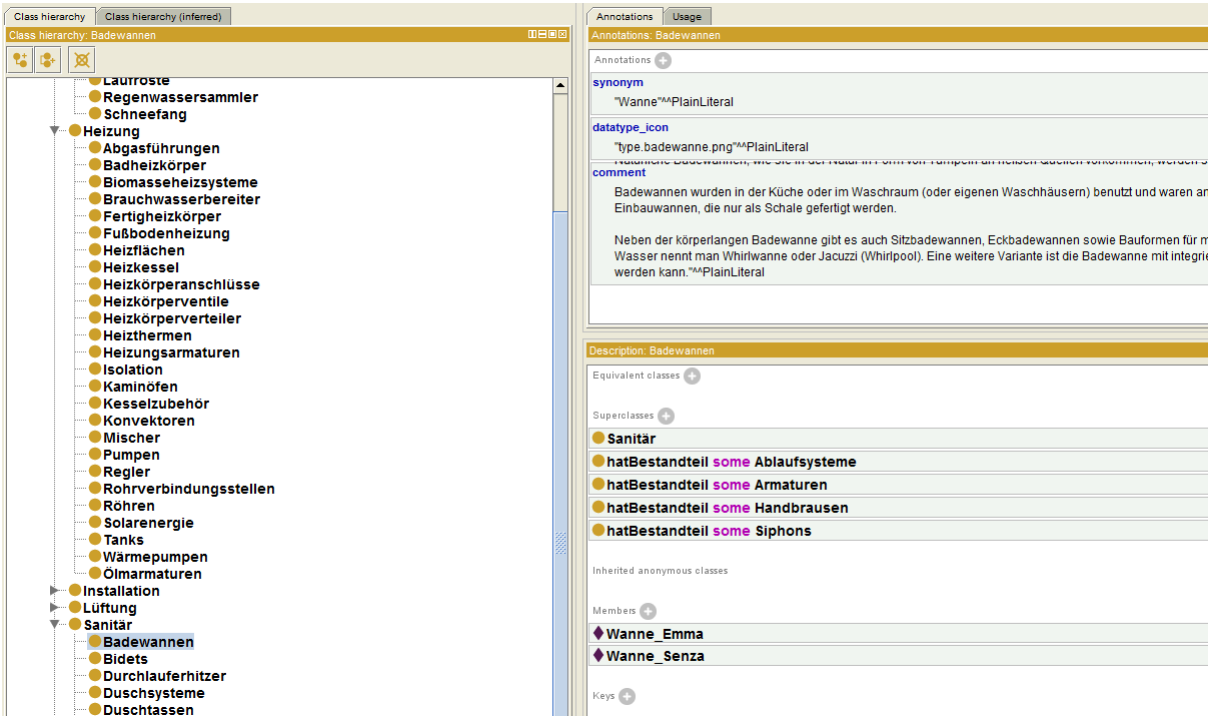


Abbildung 47: Fachspezifische Ontologie zum Bereich Heizung und Sanitär

2.4.2 Implementierung des OnToBau-Systems

In diesem Arbeitspaket wurde das OnToBau-System realisiert. Einzelne Komponenten wurden bereits in vorherigen Arbeitspaketen beschreiben. Zusammenfassend besteht das OnToBau-System aus den folgenden Komponenten:

- Dokumentkonverter (Vorverarbeitung):
  Diese Komponente ist für die Aufbereitung der unterschiedlichen Dokumenttypen zuständig, die in einem Unternehmen auftreten können. Sie garantieren, dass alle Dokumente, unabhängig ihres Typs, in ein einheitliches Datenformat transformiert werden. In Kapitel 2.2.3.1 wurde dieses Verfahren bereits beschrieben.

- Inferenzsystem:
  Diese Komponenten beinhaltet Komponenten zur Klassifikation und Extraktion von relevanten Informationen aus den Dokumenten. Hier kamen zum Teil Softwarekomponenten der Firma Mindox zum Einsatz oder wurden im Rahmen des Forschungsvorhabends selbst realisiert. Das Inferenzsystem erstellt mit Hilfe der konzeptuellen Modellierung innerhalb der Ontologien die Unternehmenswissensbasis. Die Klassifikations- und Extraktionsstrategien wurden bereits in den Kapiteln 2.2.3.2 und 2.2.3.3 beschrieben.

- Wissensbasis des Unternehmens (Ontologien):
  Die Ontologien stellen zum einen die notwendigen konzeptuellen Modelle für das Inferenzsystem zu Verfügung und enthalten zum anderen das eigentliche konkrete Faktenwissen des Unternehmens. Die Realisierung des Unternehmenswissen mittels Ontologien wurde in dem Kapitel 2.2 beschrieben.

- Front-End:
  Diese Komponente stellt die Benutzerschnittstelle zu dem OnToBau-System dar. Es besteht aus Visualisierungswerkzeugen, die die Unternehmenswissensbasis grafisch repräsentieren, einem Retrievalsystem zum Formulieren von Suchanfragen und dem Persönlichen Agenten, der das Benutzerverhalten überwacht und dem Benutzer die Möglichkeit bietet, neue Dokumenten in das OnToBau-System zu integrieren.

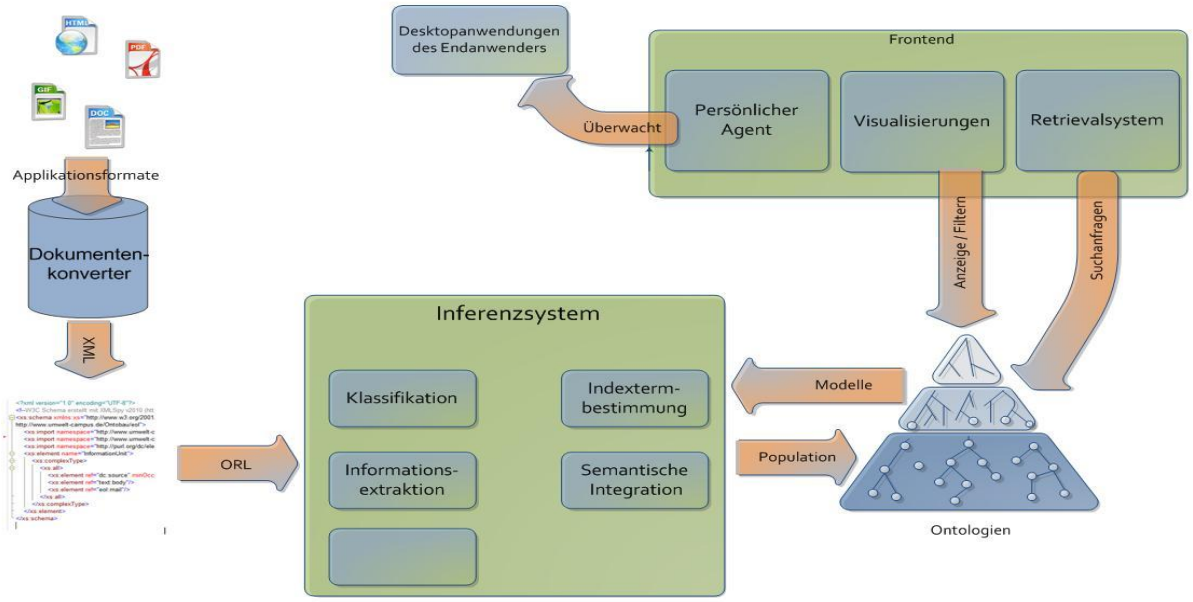


Abbildung 48: Überblick über das Gesamtsystem in OnToBau

2.4.3 Systemtest

Während der Implementierung des OnToBau-Systems stießen wir vor allem auf Probleme bei der Realisierung des persönlichen Agenten und dessen Funktionalität dem Mitarbeiter pro-aktiv Informationen zur Verfügung zu stellen.

Im Kapitel 2.3.3.1 wurde bereits ein Konzept zur Überwachung von Benutzeraktionen vorgestellt. Die vorgestellten Module konnten in unserem Testlabor und unseren Testsystemen zwar installiert werden, jedoch musst dafür einige Zugeständnisse gemacht werden. Um Benutzerverhalten zu überwachen, müssen Aktionen des Benutzers mit dem Betriebssystem protokolliert werden. Teilweise werden Tastaturanschläge mitgeschrieben, überwacht welche Internetseite geöffnet wurde usw. Damit die Software diese Informationen abgreifen kann, sind teilweise tiefen Eingriffe in die Betriebssystemumgebung notwendig. In einigen Fällen weisen die Monitoring-Tools ein ähnliches Verhalten auf wie Schadsoftware. Antivirenprogramme mussten auf unseren Testsystemen deaktiviert werden, damit die meisten Module einwandfrei funktionierten. Da die Realisierung der Module auf die spezielle Betriebssystemumgebung zugeschnitten ist, war ein Wechsel zwischen verschiedenen Betriebssystemen, teilweise nur zwischen verschiedenen Versionen nicht möglich.

Eine Integration in das Unternehmen erwies sich dadurch als erhebliches Problem, so dass das OnToBau-System lediglich auf den OnToBau-Testrechnern den Mitarbeitern unserer Projektpartner demonstriert werden konnte. Hinzu kamen arbeitsrechtliche Probleme, die die Einführung in das Unternehmen. Mitarbeiterüberwachung zur Optimierung von Geschäftsprozessen, wie im Falle von OnToBau, sind aus Lage des Arbeitnehmerdatenschutzes als problematisch anzusehen. Da das OnToBau-System nicht in der Lage ist zwischen geschäftlichen und privaten Benutzerverhalten zu unterscheiden, werden alle Aktionen des Mitarbeiters protokolliert und automatisch ausgewertet. Eine Verletzung des persönlichen Datenschutzes und der Privatsphäre können also nicht ausgeschlossen werden. Der prototypische Einsatz des OnToBau-Systems hätte also eine Zustimmung der Mitarbeiter vorausgesetzt. Welche negativen Auswirkungen die Überwachung von Mitarbeitern hat ist derzeit noch nicht ausreichend untersucht.

Da in dieser Phase des Projektes bereits ersichtlich war, dass es zu einer Einführung in das Unternehmen nicht kommen würde, wurde stattdessen auf andere Arbeitspakete im Forschungsvorhaben mehr Zeit investiert. So wurde zum Beispiel der Benutzeragent dahingehend erweitert, dass er dem Mitarbeiter die Möglichkeit bietet auf benutzerfreundliche Weise neue Dokumente in das Wissensarchiv zu integrieren und nach diesem Wissen mittels einen grafischen Anfrageeditors zu suchen. Diese Funktionalität war ursprünglich nicht vorgesehen wurde aber auf Grund der technischen Komplexität der Benutzerüberwachung in das Vorhaben aufgenommen.

2.4.4 Integration von Dokumenten in das Wissensarchiv

Für die Implementierung des intelligenten Benutzeragenten war es erforderlich, die Funktionsweise und den Funktionsumfang der Benutzerschnittstelle zu erweitern. Dabei handelt es sich um die Möglichkeiten für den Benutzer neue, eingehende Informationen zu verarbeiten und dem System hinzuzufügen.

Letztendlich soll der Benutzer die neuen Informationen in Form von Dokumenten oder E-Mails mit möglichst wenig Interaktion ins System einbringen können. Dazu wurden Plugins für die beiden gängigen E-Mail Programme Outlook und Mozilla Thunderbird entwickelt, die es dem Nutzer ermöglichen eine ausgewählte E-Mail in das Wissensarchiv einzubringen. Sollte es sich hierbei um eine E-Mail mit einem angehängtem Dokument handeln, dann wird dieser Anhang als eigenes Objekt ins Wissensarchiv aufgenommen und dem E-Mail-Objekt im Wissensarchiv eine Verknüpfung auf das Dokument-Objekt mitgegeben.

Weiterhin wurde mit der Implementierung des Benutzeragenten begonnen. Dieser kann auf einer kleinen Dropbox Dokumente per Drag&Drop entgegennehmen und sie ins Wissensarchiv aufnehmen. Außerdem kann er überwachen, ob mit einem gängigen Programm, z.B. Word, Excel, etc. ein Dokument abgespeichert wurde. Dieses wird anschließend analysiert und, falls es sich um ein unterstütztes Format handelt, wird es ins Archiv aufgenommen. In Abbildung 49 ist die geplante Schnittstelle des Agenten zu sehen, im linken Teil können Dokumente einfach mit der Maus abgelegt werden und erscheinen dann in der Liste mit ihrem Bearbeitungsstatus. Im rechten Teil sollen nach erfolgter Klassifikation die Ergebnisse angezeigt werden.

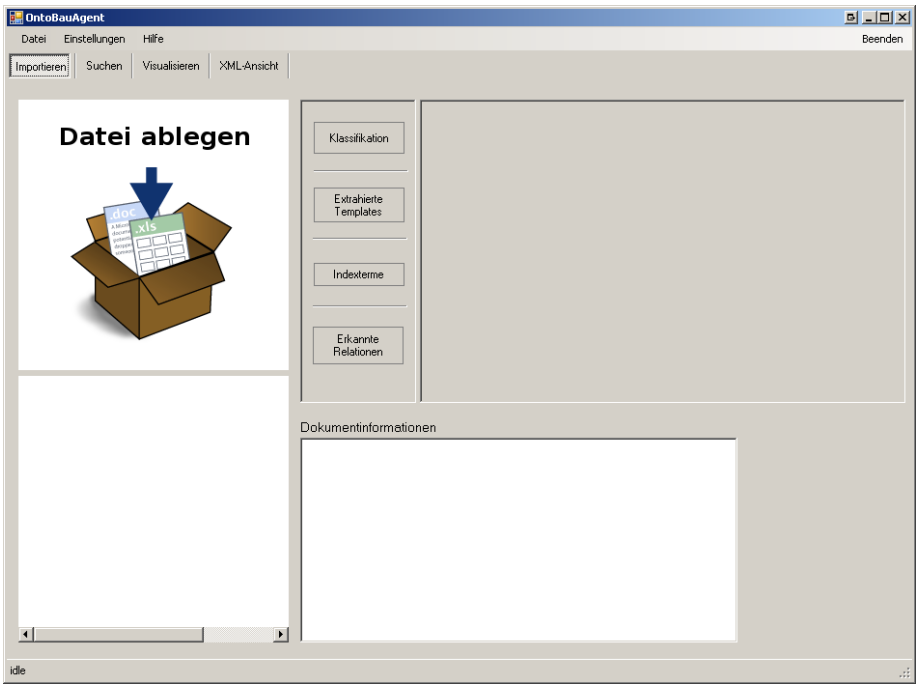
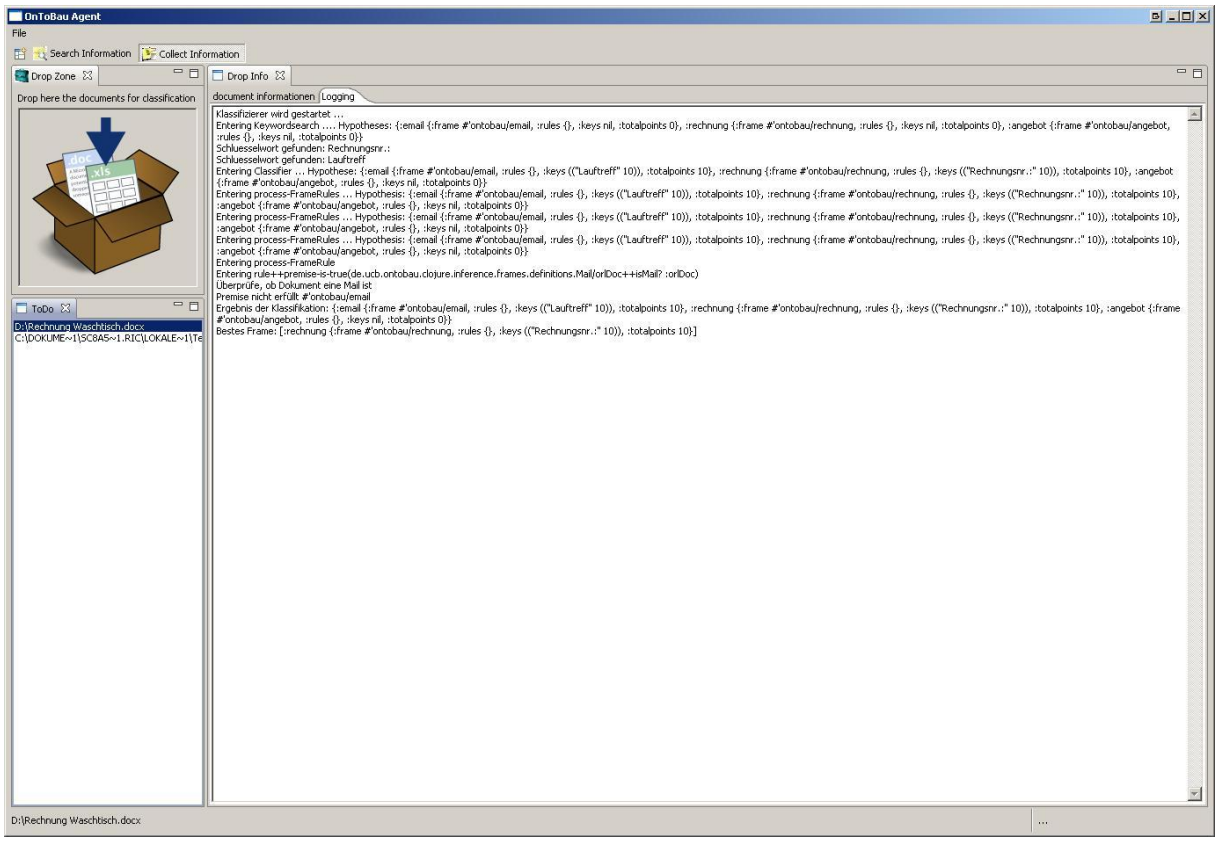


Abbildung 49: Oberfläche des persönlichen Agenten im OnToBau-System

Die Entwicklung des Benutzeragenten wurde mit dem Eclipse-RCP-Framework kontinuierlich weitergeführt. Das Hauptaugenmerk lag dabei auf der Modularität der Anwendung. Der Agent besteht aus drei Hauptmodulen, die teilweise noch unterteilt sind. Modul 1 ist für die Informations- (bzw. die Dokument-)verarbeitung zuständig. Es nimmt vom Wissensarbeiter unterstützte Dokumentdateien (z.B. Word, PDF) per Drag-and-Drop entgegen. Der Aufwand, um die Funktionsfähigkeit der entstandenen Plug-Ins für die Verarbeitung von E-Mails zu gewährleisten stellte sich als sehr hoch heraus. Mit jeder neuen Version von Mozilla Thunderbird musste das PlugIn angepasst werden. Deshalb wurde ein im Laufe des Vorhabens anderer Weg gewählt: der Benutzer zieht die zu archivierenden E-Mails analog zu simplen Dokumentdateien einfach aus dem Mailprogramm auf die Dropbox des Benutzeragenten. Wichtigster Punkt ist hierbei die Berücksichtigung von E-Mail-Attachments. Damit das Inferenz-Modul die Mail korrekt in die Ontologie einpflegen kann ist die folgende Reihenfolge an Arbeitsschritten notwendig:

1. *Attachments aus der Mail zu extrahieren*
2. *Attachments in das OnToBau-eigene XML-Format konvertieren*
3. *Attachments klassifizieren und in die Ontologie hinzufügen*
4. *E-Mail in das OnToBau-Format konvertieren*
5. *E-Mail klassifizieren*

Da die Attachments vor der E-Mail in die Ontologie hinzugefügt werden, wird sichergestellt, dass die Relationen „hatAnhang“ bzw. „istAnhangVon“ korrekt in die Ontologie übernommen werden.



```
Klassifizierer wird gestartet ...
Entering Keywordsearch .... Hypotheses: {:email {:frame #'ontobau/email, :rules {}, :keys nil, :totalpoints 0}, :rechnung {:frame #'ontobau/rechnung, :rules {}, :keys nil, :totalpoints 0}, :angebot {:frame #'ontobau/angebot,
:rules {}, :keys nil, :totalpoints 0}}
Schluesselwort gefunden: Rechnungsnr.:
Schluesselwort gefunden: Lauftreff
Entering Classifier ... Hypothese: {:email {:frame #'ontobau/email, :rules {}, :keys (("Lauftreff" 10)), :totalpoints 10}, :rechnung {:frame #'ontobau/rechnung, :rules {}, :keys (("Rechnungsnr.:" 10)), :totalpoints 10}, :angebot
{:frame #'ontobau/angebot, :rules {}, :keys nil, :totalpoints 0}}
Entering process-FrameRules ... Hypothesis: {:email {:frame #'ontobau/email, :rules {}, :keys (("Lauftreff" 10)), :totalpoints 10}, :rechnung {:frame #'ontobau/rechnung, :rules {}, :keys (("Rechnungsnr.:" 10)), :totalpoints 10},
:angebot {:frame #'ontobau/angebot, :rules {}, :keys nil, :totalpoints 0}}
Entering process-FrameRules ... Hypothesis: {:email {:frame #'ontobau/email, :rules {}, :keys (("Lauftreff" 10)), :totalpoints 10}, :rechnung {:frame #'ontobau/rechnung, :rules {}, :keys (("Rechnungsnr.:" 10)), :totalpoints 10},
:angebot {:frame #'ontobau/angebot, :rules {}, :keys nil, :totalpoints 0}}
Entering process-FrameRules ... Hypothesis: {:email {:frame #'ontobau/email, :rules {}, :keys (("Lauftreff" 10)), :totalpoints 10}, :rechnung {:frame #'ontobau/rechnung, :rules {}, :keys (("Rechnungsnr.:" 10)), :totalpoints 10},
:angebot {:frame #'ontobau/angebot, :rules {}, :keys nil, :totalpoints 0}}
Entering process-FrameRule
Entering rule++premise-is-true(de.ucb.ontobau.clojure.inference.frames.definitions.Mail/orlDoc++isMail? :orlDoc)
Überprüfe, ob Dokument eine Mail ist
Premise nicht erfüllt #'ontobau/email
Ergebnis der Klassifikation: {:email {:frame #'ontobau/email, :rules {}, :keys (("Lauftreff" 10)), :totalpoints 10}, :rechnung {:frame #'ontobau/rechnung, :rules {}, :keys (("Rechnungsnr.:" 10)), :totalpoints 10}, :angebot {:frame
#'ontobau/angebot, :rules {}, :keys nil, :totalpoints 0}}
Bestes Frame: [:rechnung {:frame #'ontobau/rechnung, :rules {}, :keys (("Rechnungsnr.:" 10)), :totalpoints 10}]
```

Abbildung 50: DropBox des Agenten mit Informationen aus den Inferenzmodulen

Das Inferenz-Modul (siehe Kapitel 2.2.3) ist das zweite Module im OnToBau-Agenten. Es klassifiziert anhand der vorliegenden Ontologie, um welche Art von Information es sich bei neu hinzugefügten Dokumenten/Informationsstücken handelt. Es kann z.B. erkennen, ob eine Rechnung, ein Lieferschein oder eine Email hinzugefügt wurde. Sollte es bei der Erkennung Unsicherheiten geben oder sogar zwei Klassen in Frage kommen, wird der Wissensarbeiter interaktiv in den Klassifizierungsprozess eingebunden und muss mittels eines Dialoges entscheiden, welche Dokumentenklasse die korrekte ist. Ist das Dokument klassifiziert, wird es zur Ontologie hinzugefügt. Dabei werden alle bekannten und benötigten Relationen zu weiteren Individuen in der Ontologie angelegt.

Das dritte Modul ist der grafische Suchanfrage-Editor der im folgenden Kapitel vorgestellt wird.

2.4.5 Grafisches Information Retrieval Modul

Da es sich als technisch zu komplex erwies, die Informationen im Wissensarchiv nur über den intelligenten Benutzeragenten erreichbar zu machen, wurden Ansätze untersucht, die es dem Benutzer erlauben selbst aktiv nach Informationen in den Wissensarchiv zu suchen. Um eine Anfrage an eine Ontologie zu formulieren gibt es einige mögliche Anfragesprachen wie z.B. SPARQL. Diese Anfragesprachen sind wie SQL-Anfragen aufgebaut und sehen folgendem Beispiel ähnlich:

```
SELECT ?invoice WHERE {
 ?invoice a (invoice and
  (hasInvoiceRecipient value
   MaxMustermann) and
   invDate some ?date and
   (hasProduct some
    (bathtub or wash-bowl))).
 filter(dateTime(?date) < dateTime
 ("2011-01-01T00:00:00Z")).
}
```

Diese Anfrage soll ausdrücken, dass eine Rechnung gesucht wird, deren Empfänger Max Mustermann ist, die vor dem 1.1.2011 ausgestellt wurde und auf der ein Waschbecken oder eine Badewanne zu finden sind. Es wird an diesem recht einfachen Beispiel ersichtlich, dass es für Laien absolut unpraktikabel ist auf diesem Weg nach Informationen zu suchen. Analog zu [Cat97] haben wir begonnen eine grafische Suche zu implementieren, welche dem Benutzer die etwas unübersichtliche Syntax abnimmt.

Der grafische Suchanfrage-Editor soll dem Umstand Rechnung tragen, dass ein technisch wenig versierter Mitarbeiter in einem KMU die Möglichkeit haben soll, im Wissensarchiv auf einfachem Weg nach Informationen suchen zu können.

Dazu wurde eine dreigeteilte Ansicht verwendet (siehe Abbildung 51): Auf der linken Seite befindet sich ein sogenanntes Treelist-Element, das alle Elemente der Ontologie hierarchisch anzeigt. Das umfasst alle in der Ontologie gespeicherten Konzepte und Relationen, aber auch alle Individuen. Der Wissensarbeiter kann in dieser Baumansicht dargestellte Elemente mit der Maus per Drag-and-Drop Vorgang auf den Editor „ziehen“. Hierbei kann der Nutzer hierarchisch übergeordnete Elemente ähnlich wie Platzhalter verwenden.

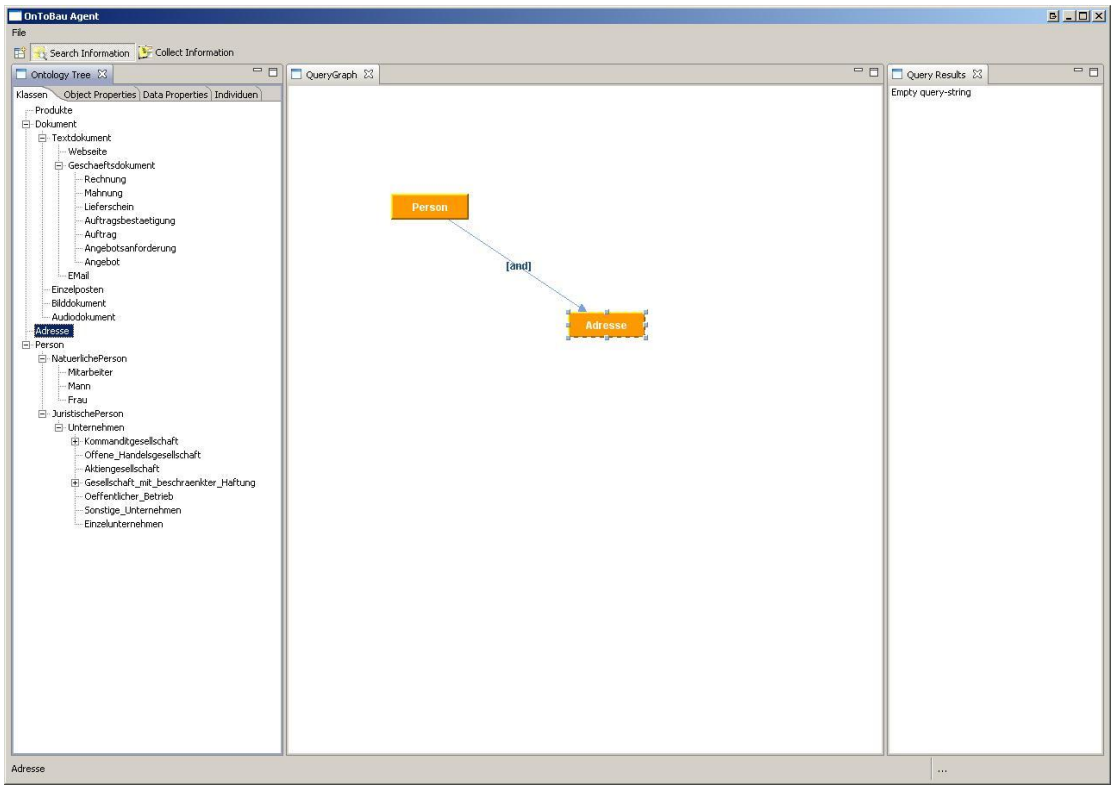


Abbildung 51: Grafische Suchanfrage im OnToBau-Agent

Sucht er z.B. nach einer Frau, welche für ein bestimmtes Unternehmen arbeitet und letzte Woche eine Rechnung geschickt hat, dann kann er statt dem Element *Frau* auch das Element *NatürlichePerson* oder auch *Person* verwenden. Dies ist möglich, da *Frau* ein Unterelement dieser anderen Elemente ist.

Prominent in der Mitte platziert ist der eigentliche Editor. Wie in Abbildung 51 zu sehen, wird hier die eigentliche grafische Suchanfrage dargestellt. Die mittels Drag-and-Drop auf den Editor gezogenen Elemente werden als gelblich-oranger Knoten dargestellt. Wird ein weiteres Element auf den Editor gelegt, dann wird automatisch eine Relation zum zuletzt gelegten Knoten gezogen. Wird das neue Element allerdings mit der Maus auf einem beliebigen Knoten abgelegt, dann wird die Relation zum gewählten Knoten gezogen. Die Relation kann dann per Klick geändert werden und kann die logische Werte annehmen, die bei einer Anfrage an eine Ontologie genutzt werden können.

Auf der rechten Seite befindet sich die „Live“ Ergebnisliste. Sobald sich aus den auf den Editor abgelegten Elementen eine logische Suchanfrage erstellen lässt, wird diese an die Ontologie angefragt. Das bedeutet, dass bei jeder Änderung, also Hinzufügen oder Löschen von Knoten, die jeweiligen Ergebnisse angepasst werden. Somit kann der Wissensarbeiter schon beim Erstellen der Suchanfrage die Ergebnisse sehen und anhand der Ergebnismenge entscheiden, ob es notwendig ist, die Anfrage noch weiter zu spezialisieren oder ob das gewünschte Ergebnis sich schon durch die vorhandene Suchanfrage finden lässt. Momentan passt die Ergebnisliste nur genau zur Suchanfrage. Somit sind (noch) keine unscharfen Suchen möglich. Denkbar wäre hier die Berücksichtigung von Teilbäumen der Suchanfrage (siehe Abbildung 52).

In dem hier dargestellten Beispiel besteht die Suchanfrage aus drei Teilbäumen. Erstens soll die Rechnung einen Empfänger mit bekanntem Nachnamen haben (hier blau dargestellt).

Zweitens soll die gesuchte Rechnung zeitlich vor einem bestimmten Datum[8.3.11] ausgestellt worden sein (grün). Und drittens soll die gesuchte Rechnung ein bestimmtes Produkt beinhalten (lila). Die hier gesuchte Rechnung soll vor dem 8. März 2011 ausgestellt worden sein, an einen Empfänger mit dem Nachnamen Schwinn gegangen sein und mindestens eine Badewanne oder ein Waschbecken enthalten. Wenn eine in der Ontologie gespeicherte Rechnung alle drei Bedingungen erfüllt dann passt sie zu 100% auf die Suche. Unter Umständen ist es aber auch erwünscht, Rechnungen zu finden, die nicht alle Bedingungen auf einmal erfüllen.

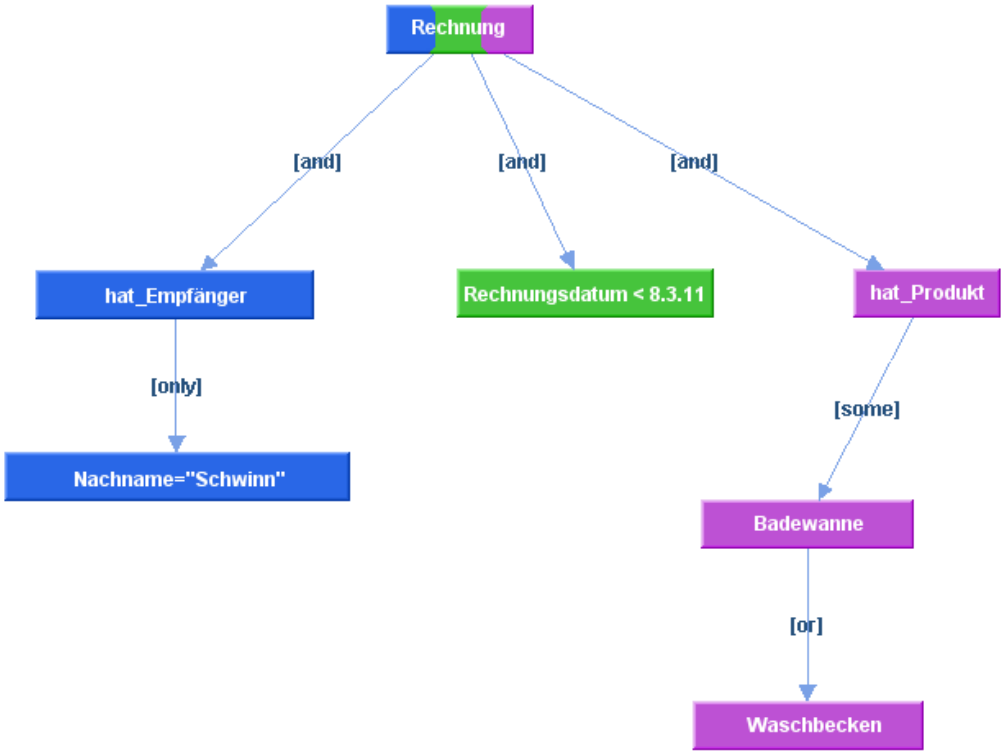


Abbildung 52: Suchanfragen bestehen aus einzelnen Teilanfragen

Falls der Wissensarbeiter sich z.B. beim Nachnamen oder beim Datum verschrieben hat oder sich nicht sicher ist, dann sollen natürlich auch diejenigen Rechnungen angezeigt werden, die möglicherweise nur zwei Bedingungen erfüllen und somit nur teilweise die Suchanfrage erfüllen. Das würde dann in der Ergebnisliste auf der rechten Seite dementsprechend gekennzeichnet werden. Vorteil eines solchen Vorgehens ist es, dass kleinere Abweichungen und kleine Fehler bei der Suchanfrage tolerant behandelt würden und der Nutzer im Idealfall trotzdem sein gewünschtes Ergebnis findet.

Bei der Integration des Editors in den Agenten hat sich die Entwicklung desselben mit dem Eclipse RCP-Framework als problematisch herausgestellt. Der grafische Editor wurde in einem studentischen Projekt seperat mit Java-Komponenten (AWT) entwickelt, welche nicht explizit zum RCP-Framework gehören und somit nicht (ohne weiteres) in dem Entwicklungsframework für den Agenten genutzt werden können. Das Eclipse RCP Framework verwendet ausschließlich SWT Komponenten. Um den grafischen Editor nicht komplett neu implementieren zu müssen wurde mithilfe einer SWT-AWT Bridge versucht den Editor einzupacken und ihn somit doch im RCP Framework nutzen zu können. Dabei

müssen allerdings Abstriche gemacht werden, so funktioniert der Rechtklick der Maus im Moment noch nicht und somit sind keine Kontextmenüs verfügbar. Das mindert den Bedienkomfort etwas.

2.5 *Arbeitspaket 5: Installation Prototyp*

2.5.1 Aufbau der Konfiguration im Unternehmen

Auf Grund der angesprochenen Probleme im Kapitel 2.4.3 wurde die Einrichtung und Konfiguration des OnToBau-Systems lediglich auf den Testrechnern durchgeführt. Sowohl die technischen Probleme, als auch die arbeitsrechtlichen Aspekte, standen einer unternehmenstauglichen Entwicklung eines Prototypen kontraproduktiv entgegen.

2.5.2 Einführung im Unternehmen

Statt eine Einführung im Unternehmen wurde das OnToBau-System auf Testrechner zur Verfügung gestellt. Mitarbeiter der Projektpartner konnten die Fortschritte des Forschungsvorhaben dadurch testen und uns entsprechendes Feedback in unserem evolutionären Entwicklungsprozess geben.

2.5.3 Anwenderschulung

Eine Anwenderschulung hat nicht stattgefunden, da der Prototyp nicht in den Unternehmen installiert wurde.

2.5.4 Evaluation

Während der kompletten Laufzeit des Forschungsvorhaben wurden in regelmäßigen Abständen Testsysteme den Projektpartnern zur Verfügung gestellt. Diese konnten zwar nicht in die IT-Infrastruktur des Unternehmens intergiert werden, sondern standen auf Testrechnern mit speziellen Betriebssystemumgebungen zur Verfügung. Feedback und daraus resultierende Evaluationen waren aber auch unter diesen Zugeständnissen möglich.

3 Voraussichtlicher Nutzen und Verwertbarkeit

3.1 *Wissenschaftliche Verwertung*

Im Rahmen des Forschungsvorhaben konnten sowohl wissenschaftliche als auch populärwissenschaftliche Veröffentlichungen vorgenommen werden. Eine Auflistung der Veröffentlichungen ist in Kapitel 4 zu finden.

Verwertung im Sinne der Qualifizierung wissenschaftlichen Nachwuchs konnte vor allem durch die in diesem Projekt gestartet Promotion eines Mitarbeiters erreicht werden. Die Zusammenarbeit mit der Universität Trier in diesem Forschungsvorhaben führte letztlich zu einer dauerhaft angestrebten Kooperation zwischen der Hochschule Trier und der Universität Trier. Dies zeigt sich inzwischen bereits durch regelmäßige gemeinsame Doktorandenkolloquien. Die Sicherstellung der Fortführung der kooperativen Promotion soll durch einen Antrag im Ingenieursnachwuchsprogramm gewährleistet werden.

Während des Forschungsvorhaben konnten zudem zahlreiche Abschluss- und Projektarbeiten durchgeführt werden. Im neu-akkreditierten Masterstudiengang wurde außerdem eine Lehrveranstaltung Wissensmanagement eingeführt, die wesentliche Inhalten aus dem Forschungsvorhaben enthält.

3.2 *Fachlicher Wissenstransfer*

Innerhalb des Projektes fanden regelmäßige Treffen und Workshops statt, die den aktuellen Stand kommunizierten. Auf zwei Exponatsausstellungen auf der CeBIT in den Jahren 2010 und 2011 konnten die aktuellen Forschungsergebnisse einem breiten Publikum präsentiert werden.

3.3 *Verwertung und Anschlussfähigkeit des Projekts*

Bereits während der Durchführung des Projektes entstand Kontakt zu Firmen und Institutionen aus anderen Branchen, die ein Interesse an den Arbeiten in diesem Vorhaben zeigten. So konnte das Interesse des Gemeinsamen Ausschuss für Elektronik im Bauwesen (GAEB) an unserem Vorhaben geweckt werden. Der GAEB hat es sich u.a. zur Aufgabe gemacht den Datenaustausch von Bauinformationen zu standardisieren. Ziel ist es dabei die effiziente Ausschreibung, Vergabe und Abrechnung von Bauleistungen zu ermöglichen.

Mehrere Treffen fanden zum Ideenaustausch statt und eine gemeinsame Bachelorarbeit wurde im Rahmen des Forschungsvorhaben durchgeführt. Eine gemeinsame Antragstellung zur Fortführung der Forschungsergebnisse ist in Planung.

Die Firma Phast [1]und flexible4science [2]zeigte ebenfalls Interesse an unseren Forschungsergebnissen. Beide sind in der pharmazeutischen Branche tätig und haben ein breit gefächertes Unternehmensportfolio. Die Firma Phast ist in der Entwicklung neuer Arzneiformen tätig und bietet hier Dienst von der Qualitätskontroller, über den Qualitätsservice bis hin zur Unterstützung bei der Entwicklung an. Im Rahmen dieser Tätigkeit fallen riesige Mengen an Dokumentationsberichten an, die bisher in dem Unternehmen in statischen Ordnerhierarchien abgelegt werden. Eine semantische Aufbereitung und Visualisierung dieser Informationen erachten die beiden Firmen als enorme Arbeitserleichterung.

Inwieweit die Erkenntnisse aus OnToBau auf die Pharmazie-Branche übertragen werden können soll in einem eigenen Forschungsvorhaben weiter untersucht werden.

4 Veröffentlichungen

4.1 *Veröffentlichungen der Projektergebnisse*

Schwinn, M., Kuhn, N., and Richter, S. **Ontology based knowledge management for SMEs**. In *Proceedings of the 2nd International Multi-Conference on Complexity, Informatics and Cybernetics: IMCIC 2011,* Orlando, Florida*,* volume 1, pages 151–155*.*

Schwinn, M., Kuhn, N., and Richter, S. **Ontology-Based Knowledge Management - Graphical Query Editor for OWL Ontologies**. In *Proceedings of the 13th International Conference on Enterprise Information Systems: ICEIS 2011,* Beijing, China, pages 235-240.

Schwinn, M., Kuhn, N., and Richter, S. **An Approach to a Graphical Query Editor - For Ontology-based Knowledge Management**. In *Proceedings of the International Conference on Knowledge Management and Information Sharing: KMIS 2011,* Paris, France, 2011, pages 436 - 441.

4.2 *Veröffentlichungen ohne Review*

Kuhn, Norbert und Schwinn, Markus. **Wissen, was ihr Unternehmen weiß**, In „intiativ - das Wirtschaftsmagazin“, Ess-Verlag, 01/2012

[1] http://www.phast.de
[2] http://www.flexible4science.de

4.3 *Integration in die Lehre*

Während des Forschungsvorhaben konnte der Transfer in die Lehre durch verschiedenen Veranstaltungen gewährleistet werden, die im Folgenden aufgelistet werden:

Wintersemester 2010/2011	Durchführung eines Seminars zum semantischen Wissensmanagement
Sommersemester 2011	Durchführung eines Seminars zum Thema Semantic Web
Wintersemester 2011/2012	Integration des Forschungsvorhabens in das Mastermodul „Anwendungen der künstlichen Intelligenz“ in Form von Projektarbeiten
Wintersemester 2012/2013	Durchführung eines Seminar zum Thema Wissensmanagement
	Aufnahme des Moduls „Wissensmanagement“ in das Curriculum des Masterstudiengangs Informatik

4.4 *Studentische Arbeiten*

Im Folgenden werden die in diesem Forschungsvorhaben durchgeführten studentischen Arbeiten aufgelistet:

4.4.1 Diplomarbeiten

Fabian Müller und Burhan Hayber, **Implementierung, Bewertung und Anwendung von maschinellen Lernverfahren zum automatischen Extrahieren von Instanzen einer Ontologie zum Aufbau einer A-Box,** Februar 2011, Universität Frankfurt

4.4.2 Bachelorarbeiten

Andreas Politz, **Grafisches Interface zur Darstellung und Manipulation der Relationen und Eigenschaften von (Bild)Objekten**, Februar 2010, FH Trier

Sebastian Stein, **Implementierung und Bewertung einer automatischen Fachwortextraktion und unscharfer Suche durch Clusteranalysen der Fachwörter**, Mai 2011, Umwelt-Campus Birkenfeld

Rene Huber, **Analyse von XML-Dokumenten auf Word-Basis zur Bestimmung von Text und Texteigenschaften**, Oktober 2011, FH Trier

Stefan Klein, **Visualisierung von semantischen Zusammenhängen mit Hilfe von Treemap Algorithmen**, Oktober 2011, FH Trier

Yvonne Hübner, **Qualitätsmanagement im Rahmen des standardisierten Datenaustauschs im Bauwesen**, November 2011, Umwelt-Campus Birkenfeld

4.4.3 Masterarbeiten

Markus Schwinn, **Automatische Informationsextraktion aus Dokumenten zum Aufbau eines Wissensarchivs als Grundlage für eine automatisierte Angebotserstellung**, Februar 2010, FH Trier

4.4.4 Projektarbeiten

Yvonne Hübner, **Implementierung von Dokumentkonvertern zur Transformation in die OnToBau Representation Language**, IP-Projekt, August 2010, Umwelt-Campus Birkenfeld

Sebastian Stein, **Implementierung einer Thunderbird Erweiterung zur Transformation von E-Mails in die OnToBau Representation Language**, Praktische Studienphase, August 2010, Umwelt-Campus Birkenfeld

Nima Pourjahedi, **Implementierung eines grafischen Editors zur Formulierung von SPARQL-Anfragen an Ontologien**, IP-Projekt, Oktober 2010, Umwelt-Campus Birkenfeld

Sebastian Stein, **Interpretation von E-Mailanhängen mittels Hidden-Markov-Ketten aus dem E-Mailtext**, Fachprojekt, Februar 2011, Umwelt-Campus Birkenfeld

Fabian Funk, **Implementierung eines Tools zur Rekonstruktion von OCR-Ergebnissen in eine textuelle Darstellung**, Fachprojekt, Juni 2011, Umwelt-Campus Birkenfeld

Stephan Gohlke, **Implementierung einer SWT / AWT-Bridge für den OnToBau Query Editor**, Master-Projektstudium, September 2011, Umwelt-Campus Birkenfeld

Ralph Schlesser, **Konzeptionierung und Implementierung eines Systems zum Monitoring von Benutzerinteraktionen**, IP-Projekt, Februar 2012, Umwelt-Campus Birkenfeld

4.5 *Sonstige Auftritte*

18.11.2009	Infostand an der Nacht der Wissenschaft am Umwelt-Campus Birkenfeld
02.03-2010 - 06.03.2010	Exponatsausstellung auf der CeBIT 2010
05.03.2010	Vortrag „Ontologie-basiertes Wissensmanagement in KMUs“ auf der CeBIT 2010
01.03-2011 - 05.03.2011	Exponatsausstellung auf der CeBIT 2011
01.03.2012	Präsentation des Forschungsvorhabens bei der Firma flexible4science

III LITERATURVERZEICHNIS

[AB06] A., Franken und A. Braganza: **Organizational forms and knowledge management: one size fits all?** In: International Journal of Knowledge Management Studies, Band 1, Seiten 18–37, 2006.

[AH08a] Dean Allemang and James Hendler. **Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL**. Morgan Kaufmann, May 2008.

[AH08b] Jesse Alpert and Nissan Hajaj. We knew the web was big..., 2008. http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html [24. August 2009].

[Bai08] Elisabeth Baier. **Semantische Technologien in Wissensmanagementlösungen. Einsatzpotenziale für den Mittelstand**. FAZIT-Schriftenreihe. Stiftung Baden-Württemberg, 2008.

[Ber01] Michael K. Bergman. **The deep web: Surfacing hidden value**. The Journal of Electronic Publishing, 7(1), 2001.

[BF04] A. Bratko and B. Filipič. **A Study of Approaches to Semi-structured Document Classification**. Technical Report IJS-DP-9015, Department of Intelligent Systems, Jožef Stefan Institute, Ljubljana, Slovenia, 2004.

[BG98] D. Brickley and R. Guha. **Resource Description Framework (RDF) Schemas**, W3C Working Draft. Technical report, W3C, http://www.w3.org/TR/1998/WD-rdf-schema-19980409, 1998.

[BMR+96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. **Pattern-Oriented Software Architecture** Volume 1: A System of Patterns. Wiley, 1 edition, August 1996.

[Bod03] Jochen Böder. **Das Internet und seine Folgen - Was Luthers Reformation und unsere Zukunft gemeinsam haben.**.., 2003. http://www.intelligenzia.de/Texte/Internet.pdf [18. August 2009].

[BS03] Franz Baader and Ulrike Sattler**. Description logics with aggregates and concrete domains**. Inf. Syst., 28(8):979–1004, 2003.

[Cas01] Manuel Castell. **Das Informationszeitalter Wirtschaft. Gesellschaft. Kultur**. Bd. 1: Die Netzwerkgesellschaft. Leske + Budrich Verlag, 2001.

[Cat97] Catarci, T. (1997). **Visual Query Systems for Databases: A Survey**. Journal of Visual Languages & Computing, 8(2):215–260.

[Chi98] Nancy Chinchor. **MUC-7 Information Extraction Task Definition (version 5.1).** In Proceedings of MUC-7 Grishman, R, 1998.

[Dir05] Lewandowski Dirk. **Web Information Retrieval : Technologien zur Informationssuche im Internet**. DGI, 2005. Auch als html Dokument vorhanden.

[Dyo08] DYONIPOS - **Kontextsensitive Unterstützung von Wissensprozessen. Ein Joint Venture zwischen Wissenschaft, Wirtschaft und Verwaltung**., 2008. http://www.m2n.at/_layout/_downl/m2n_folder_dyonipos.pdf [10. September 2009].

[Fis06] Fischer, Hajo: **Wissensmanagement für KMU - Was sind die Voraussetzungen?** online, Juli 2006.

[GS96] Ralph Grishman and Beth Sundheim. **Message Understanding Conference-6: a brief history**. In Proceedings of the 16th conference on Computational linguistics, pages 466–471, Morristown, NJ, USA, 1996. Association for Computational Linguistics.

[Goa02] Laurence Goasduff. **Gartner Says 90 Percent of Businesses Suffer from Information Overload**, 2002. http://www.gartner.com/5_about/press_releases/2002_05/pr20020507c.jsp [24. August 2009].

[GOR+07] Olaf Grebner, Ernie Ong, Uwe Riss, Marko Brunzel, Ansgar Bernardi, and Thomas Roth-Berghofer. **Task Management Model**. Delivarable, 2007.

[Gru91] Thomas R. Gruber. **The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases**. In KR, pages 601–602, 1991.

[Har59] Zellig Harris. **Linguistic Transformation for Information Retrieval**. In: Proceedings of the International Conference on Scientific Information, volume 2, pages 937–950, Washington DC, USA, 1959. National Academy of Sciences.

[Hec05] Matthias Hecking. **Domänenspezifische Informationsextraktion am Beispiel militärischer Meldungen**. In GI Jahrestagung (2), pages 109–113, 2005.

[Hec08] Matthias Hecking. System **ZENON - Semantic Analysis of Intelligence Reports**. In Proceedings of the LangTech 2008, 2008.

[HFW03] Joachim Hinrichs, Jürgen Friedrich, and Volker Wulf. **Zur Bedeutung des Nutzungskontextes im Dokumentenmanagement: Empirische Befunde und technische Lösungsansätze**. In Mensch & Computer, 2003.

[HKRS07] Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, and York Sure. **Semantic Web: Grundlagen**. Springer, 1 edition, October 2007.

[HMS03] M. Henzinger, R. Motwani, and C. Silverstein. **Challenges in Web Search Engines**, 2003.

[HVA09] Heiko Haller, Max Völkel, and Andreas Abdecker. **Integrated Knowledge Workbench**. Delivarable, 2009.

[JG99] H. Strack-Zimmermann J. Gulbins, M. Seyfried. **Dokumenten-Management**. Springer-Verlag, 2. edition, 1999.

[KCL03] Su Jeong Ko, Jun Hyeog Choi, and Jung Hyun Lee. **Bayesian web document classification through optimizing association word**. In IEA/AIE'2003: Proceedings of the 16th international conference on Developments in applied artificial intelligence, pages 565–574. Springer Springer Verlag Inc, 2003.

[LV03] Peter Lyman and Hal R. Varian. **How Much Information?**, 2003. http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/printable_report.pdf [17. August 2009].

[MNV02] Michele Missikoff, Roberto Navigli, and Paola Velardi. **The Usable Ontology: An Environment for Building and Assessing a Domain Ontology**. In ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web, pages 39–53, London, UK, 2002. Springer-Verlag.

[Moe06] Marie-Francine Moens. **Information Extraction: Algorithms and Prospects in a Retrieval Context** (The Information Retrieval Series). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[MR06] Josef Makolm and Doris Reisinger. **Forschungsprojekt DYONIPOS - Dynamische, intelligente Unterstützung der Wissensprozesse in der IT-Sektion des BMF**, 2006. http://www.m2n.at/_layout/_downl/Makolm-Reisinger%20-%20Wissensmanagement-VerwInnovZentrum%2020060620.pdf [10. September2009].

[MUC91] MUC3 '91: **Proceedings of the 3rd conference on Message understanding**, Morristown, NJ, USA, 1991. Association for Computational Linguistics.

[MUC92] MUC4 '92: **Proceedings of the 4th conference on Message understanding**, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[MUC93] MUC5 '93: **Proceedings of the 5th conference on Message understanding**, Morristown, NJ, USA, 1993. Association for Computational Linguistics.

[MUC95] MUC6 ’95: **Proceedings of the 6th conference on Message understanding**, Morristown, NJ, USA, 1995. Association for Computational Linguistics.

[MY07] Larry Manevitz and Malik Yousef. **One-class document classification via neural networks**. Neurocomput., 70:1466–1481, March 2007.

[NBB+97] Günter Neumann, Rolf Backofen, Judith Baur, Markus Becker, and Christian Braun. **An Information Extraction Core System for Real World German Text Processing**. In ANLP, pages 209–216, 1997.

[Net97] Thomas Netousek. **Verbesserung des Retrievalergebnisses durch Aufbau unscharfer Thesauri und Wortstammbildungen**. PhD thesis, Wirtschaftsuniversität Wien, Wien, 1997.

[Neu01] Günter Neumann. **Informationsextraktion**. In R. Klabunde, editor, Computerlinguistik und Sprachtechnologie - Eine Einführung, pages 448–455. Spektrum Akademischer Verlag, Heidelberg, 2001.

[Nev02] Bruce E. Nevin, editor. The Legacy of Zellig Harris. **Current Issues in Linguistic Theory** 228. John Benjamins, Philadelphia, 2002.

[OdP02] Pablos, P. Ordonez de: **Knowledge management and organizational learning: typologies of knowledge strategies in the Spanish manufacturing industry from 1995 to 1999**. In: Journal of Knowledge Management., Seiten 52–62, 2002.

[PH06] Jeff Z. Pan and Ian Horrocks. **OWL-Eu: adding customised datatypes into OWL**. Web Semantics, 4(1):29–39, 2006.

[Poh06] Alexander Pohl. **Modelling Examination Regulations in OWL**, Institute of Computer Science, LMU, Munich. Projektarbeit/project thesis, 2006.

[Pow03] Shelley Powers. **Practical RDF**. O’Reilly Media, Inc., 1st edition, August 2003.

[Rei10] Klaus Reichenberger. **Kompendium semantische Netze Konzepte, Technologie, Modellierung**, volume 1st Edition of X.media.press. Springer, 2010.

[RGSH08] Gerald Reif, Tudor Groza, Simon Scerri, and Siegfried Handschuh. **Final NEPOMUK Architecture**. Delivarable, 2008.

[RK09] Sara Radicati and Masha Khmartseva. **Email Statistics Report, 2009- 2013**, 2009. http://www.radicati.com/wp/wp-content/uploads/2009/05/email-stats-report-exec-summary.pdf [19. August 2009].

[Sch05] Markus Schwinn. **Einbeziehung von Ontologien in die Dokumentenanalyse**. Diplomarbeit, Fachhochschule Trier Fachbereich Angewandte Informatik, 2005.

[SET09] Toby Segaran, Colin Evans, and Jamie Taylor. **Programming the Semantic Web**. O'Reilly, Beijing, 2009.

[Spi08] Jonathan B. Spira. **Information Overload: Now $900 Billion - What is Your Organization´s Exposure?,** 2008. http://www.basexblog.com/2008/12/19/information-overload-now-900-billion-what-is-your-organizations\discretionary{-}{}{}-exposure/ [19. August 2009].

[SW08] Schröder, Michael und Peter Westerheide: **Wirtschaftliche und gesellschaftliche Bedeutung von Familienunternehmen.** Technischer Bericht, Zentrum für Europäische Wirtschaftsforschung GmbH, 2008.

[TRGL06] Klaus Tochtermann, Doris Reisinger, Michael Granitzer, and Stefanie Lindstaedt**. Integrating Ad Hoc Processes and Standard Processes in Public Administrations**. In Proceedings of the OCG eGovernment Conference, Linz (Austria),, 2006.

[VBdT95] Allan Vermeulen, Gabe Beged-dov, and Patrick Thompson. **The PipelineDesign Pattern**, 1995.

[WA04] K.Y. Wong and E. Aspinwall. **Characterizing knowledge management in the small business environment.** Journal of Knowledge Management, 8:44–61, 2004.

[ZSS01] Stephan Zelewski, Reinhard Schütte, and Jutta Siedentopf. **Ontologien zur Repräsentation von Domänen.** In Georg Schreyögg, editor, Wissen in Unternehmen: Konzepte, Maßnahmen, Methoden, pages 183–221. Erich Schmidt Verlag, 2001.

[ZZY+08] Guo-Qing Zhang, Guo-Qiang Zhang, Qing-Feng Yang, Su-Qi Cheng, and Tao Zhou**. Evolution of the internet and its cores.** New J. Phys., 10(12):123027+, December 2008.