Text Driven Modeling Architecture - Modellierung im Zeitalter Serviceorientierter Architekturen

Mai, André

kommfinanz GbR
Elsternweg 6
53179 Bonn
andre.mai@kommfinanz.de

Abstract: Mit Serviceorientierte Architekturen können wiederverwendbare Modellstrukturen entwickelt werden. Bei der Modellierung über Unternehmensgrenzen hinweg sind Missverständnisse vorprogrammiert, die durch eine Transformation von natürlichsprachlichen Texten in semiformale Modelle vermieden werden können. Im Beitrag werden existierende Möglichkeiten zur Extraktion von UML-Klassenmodellen aus Benutzeranforderungen, statistisch-linguistische Analysten und Heuristiken untersucht und zu einer neuen Methode kombiniert.

1. Herausforderung Serviceorientierte Architekturen

Unter Serviceorientierten Architekturen (SOA) werden Hard- und Softwarelandschaften sowie Organisationsformen verstanden, die durch eine Ausrichtung auf Dienste (engl. services) gekennzeichnet sind. Der Begriff Service wird insofern gleichbedeutend mit dem Begriff Dienst verwendet [WikiSOA]. Dienste sind insbesondere – aber nicht nur – Softwareleistungen, die einem Client von einem Server zur Verfügung gestellt werden. In der Mathematik und Informatik wird zwischen zustandslosen Diensten und Zustandsautomaten unterschieden, unter technische Services fallen RPC, CORBA, RMI, Web-Service [EiPi07, S. 13 f.]. Organisatorisch gesehen ist ein Service ein gekapselter (Teil-)Prozess mit definierten Eingangsparametern und vorgegebener Funktionalität.

Die technischen Argumente, die für eine SOA sprechen, sind lose Kopplung von Diensten durch standardisierte Kommunikationsstrukturen (z. B. [EiPi07, S. 13], [Herz05, S. 10], [WikiSOA]), wobei jedoch die Frage zu stellen ist, ob eine Kopplung von Diensten, die von Servern an verschiedenen Standorten angeboten werden, tatsächlich eine Verbesserung in der Performanz und Stabilität von Anwendungen bietet [EiPi07, S. 3].

Für die Modellierung von Geschäftsprozessen, bspw. mit ereignisgesteuerten Prozessketten (EPK, vgl. [Sche98]) bietet das Konzept der Serviceorientierung jedoch den Vorteil, während der Modellierung Kapselungen von Prozess-Schritten vorzunehmen, die an geeigneter Stelle wieder verwendet werden können. [BKnM08]

Bei der Definition von Services treten technische Probleme auf. Zum einen ist die Granularität der Dienste oft nicht klar ([EiPi07, S. 11], [Herz05, S. 13]). Hinzu kommen sämtliche Probleme, die mit verteilten Systemen einhergehen [EiPi07, Se. 3].

Konzeptuell wird deutlich, dass SOA sowohl auf der technischen Ebene in Form von Webservices und XML [Herz05, S. 11] als auch auf der organisatorischen Ebene ([EiPi07, S. 3]) diskutiert wird. Neben der uneinheitlichen Verwendung der Konzepte stellt sich insbesondere bei der Modellierung auch das Problem einer uneindeutigen Verwendung von Begriffen in Modellen ([Ortn00a], [Ortn00b]).

Um dieses Problem zu lösen, wird ein Ansatz vorgestellt, mit dem Inhalte aus Dokumenten, existierenden Modellen und vorhandener Software automatisch in Modelle transformiert werden. Somit gelingt dem Modellentwickler ein schneller Überblick über die verwendeten Begriffe und deren Strukturen und die uneindeutige Verwendung von Begriffen in Modellen wird vermieden. Durch dieses Vorgehen wird erreicht, dass Modellentwickler, die den Fachraum nicht so gut kennen, mit den relevanten Begriffen konfrontiert werden. Kennen sie deren Bedeutung, finden sie leichter Zugang zu den fremden Informationen.

Der Beitrag ist wie folgt aufgebaut: Kapitel 2 enthält Ansätze zur Normierung von Modellbegriffen. In Kapitel 3 werden mit beispielhaften Methoden des Information Retrievals und Heuristiken Wege zum Strukturieren von Informationen angegeben. Diese Ansätze werden im 4. Kapitel zu einer neuen Methode des Text Driven Modeling Architecture verbunden, mit der unstrukturierten Textinformationen in UML-Klassenmodelle überführt werden können. Die beispielhafte Modellierung eines kommunalen Informationssystems zeigt in Kapitel 5 eine Einsatzmöglichkeit der Methode demonstriert, ehe im 6. Kapitel ein Ausblick auf die automatische Extraktion von Servicekandidaten gegeben wird.

2. Normierung von Modellbegriffen

2.1. Methodisches Vorgehen

In Softwareentwicklungsprojekten müssen häufig zu Beginn Benutzeranforderungen erhoben und dokumentiert werden ([Part98]). Die Effizienz dieses Vorgehens und die Akzeptanz der entstehenden Interaktionsanwendungen lässt sich steigern, wenn die späteren Benutzer als Fachexperten in die Entwicklung einbezogen werden ([Wiem03]).

Erfahrungsgemäß sind Fachexperten nicht ausreichend in den Methoden des Software-Engineerings geschult. Sie stellen ihre Anforderungen mit Hilfe natürlichsprachlicher Dokumente (vgl. [Rupp01b], [Rupp03]) oder mit Hilfe von Prozessmodellen dar (vgl. [Sche98]), während im Software-Engineering die UML Anwendung findet (vgl. u. a. [OMG00, S. 2], [OMG01, S. 6], [Jec+04a, S. 323], [Jec+04b, S. 10]).

Das gemeinsame Verständnis der dem zu entwickelnden System zu Grunde liegenden Fachterminologie [Heye02, S. 1] hilft, die Fachexperten in das Projekt zu integrieren, und Missverständnisse bei der Interpretation von Anforderungen durch die Entwickler zu vermeiden. Sowohl die natürlichsprachliche Darstellung als auch UML-Diagramme haben ihre Berechtigung. Natürlichsprachliche Darstellungen bieten eine höhere Verständlichkeit für ungeübte Analysten. UML-Diagramme verfügen über eine höhere Präzision.

Aus diesem Grund wäre eine Kopplung beider Methoden denkbar. Der Einsatz mehrerer Methoden führt jedoch regelmäßig zu Konsistenzproblemen, wenn die jeweiligen Dokumente oder Modelle per Hand abgeglichen werden müssen (vgl. z. B. [RuSc04, S. 2]).

Um die Konsistenzprobleme zu vermeiden und den Aufwand zur Überführung eines Dokuments in ein Modell bzw. eines Prozessmodells in ein UML-Modell zu verringern, bietet sich die automatische Transformation der Anforderungen in Modelle an. Mit der Model Driven Architecture definiert die OMG Standards zur Transformation von Modellen in andere Modelle (vgl. [OMG00], [OMG01], [FeLo03], [Fran03]). Die Transformation von unstrukturierten Dokumenten in Modelle wird von der MDA jedoch nicht abgedeckt. Mit der natürlichsprachlichen Informationsbedarfsanalyse (NIBA) existiert ein Ansatz zur Generierung von UML-Modellen aus Dokumenten, der im Abschnitt 0 vorgestellt wird.

2.2. Natürlichsprachliches Parsing als Mittel zur Transformation von Benutzeranforderungen in semiformale Modelle

Das natürlichsprachliche Parsing basiert auf einer grammatikalischen Analyse der Sätze in einem Dokument. Jeder Satz wird in seine einzelnen Worte zerlegt, deren Beziehungen anhand der verwendeten Verben und ihrer Stelligkeit extrahiert werden [FlKM00], [FlMM00].

Im NIBA-Projekt – der Natürlichsprachlichen InformationsBedarfsAnalyse – wurde ein Parser und Mapper implementiert, mit deren Hilfe die entsprechenden Beziehungen gewonnen und in ein UML-Klassenstrukturdiagramm überführt werden können. [FlWe00] Ein Ausschnitt ist in Abbildung 1 dargestellt.

Um dieses Verfahren für die gesamte deutsche Sprache anzuwenden, ist die komplette Abbildung aller deutschen Verben und mit ihren jeweiligen Stelligkeiten notwendig. Unter Stelligkeiten wird die Anzahl der Subjekte und Objekte verstanden, die im Kontext eines Verbs auftauchen. [FlWe00] Geeignete Algorithmen zur generischen Extraktion dieser Beziehungen sind für deutsche Verben nicht vorhanden. Damit wird das Verfahren bei umfangreichen Texten oder Spezialkontexten sehr aufwändig, wenn das entsprechende Verzeichnis manuell gepflegt werden muss.

In Kapitel 3 werden ein Verfahren zur statistisch-linguistischen Analyse von unstrukturiertem Texten sowie eine Methode zur Überführung von Benutzeranforderungen in UML-Modelle vorgestellt.

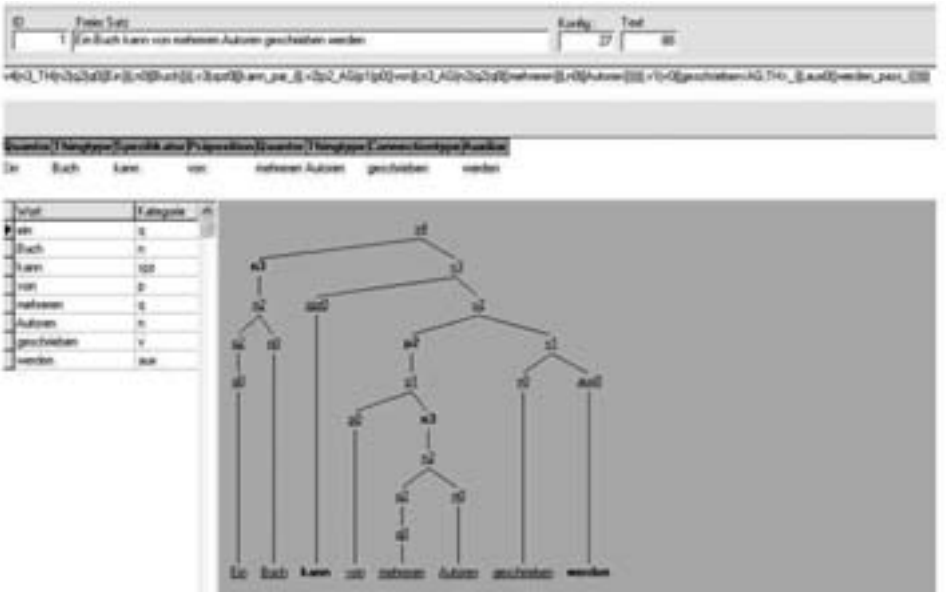

Abbildung 1: Natürlichsprachliches Parsing (aus [FlKM00])

3. Strukturierung von Benutzeranforderungen

3.1. Patternmatching und statistisch-linguistische Analysen

Der Ansatz des Pattern-Matchings beruht auf einer Matching-Operation `match[pattern, text]`, mit der entweder das Vorhandensein oder alle Positionen eines als Pattern definierten Strings in einem Text oder einem Dokument bestimmt werden [Sun90]. Es existieren verschiedene Verfahren zur Analyse von Texten, wie der Algorithmus von Boyer and Moore, das approximative Pattern-Matching, der Soundex-Algorithmus [Fuh96]. Ergebnis dieser Verfahren sind Begriff–Zitat- oder Begriff–Dokument-Beziehungen.

Als Beispiel für das Pattern-Matching soll der Ansatz der automatischen Gewinnung von Fachbegriffen dienen [Heye02], der in Form des Wortschatztools [Wort] implementiert wurde. Mit Hilfe des Wortschatztools lassen sich aus dem Deutschen Wortschatz[1] die häufigsten Worte, Kookkurrenzen[2] sowie linke und rechte Nachbarn aus einem Text extrahieren und die bestehenden Beziehungen grafisch aufbereiten (vgl. [Wort]).

[1] Im Deutschen Wortschatz werden "sorgfältig ausgewählte öffentlich zugängliche Quellen automatisch erhoben". [Wort, Hinweis unter einem Suchergebnis]

[2] Unter Kookkurrenzen wird das gemeinsame Auftreten von Worten in einem Dokument verstanden. Es wird angenommen, dass die beiden Worte interdependent sind, wenn sie häufig gemeinsam in Dokumenten auftreten. (vgl. http://de.wikipedia.org/Kookkurrenz, [Heye02, S. 4, wobei in der Quelle auf Kollokationen, also das unabhängige gemeinsame Auftreten von Wortformen, abgestellt wird)

Neben den Worten werden im Wortschatztool Zitate angezeigt, in denen die gesuchten Worte vorkommen. Außerdem enthält die zugrunde liegende Technologie die Möglichkeit zur Identifikation von Aussagen, die im Zusammenhang mit den Worten besonders häufig auftauchen.

Eine Technologie, wie sie im Wortschatztool vorliegt, bildet somit die Grundlage für die Extraktion von Begriffen und Zitaten aus Dokumenten.

3.2 Heuristiken

1998 wurde von Mario Jeckle ein Ansatz zur Strukturierung von Anforderungen aus natürlichsprachlichen Dokumenten vorgestellt ([Jeck98]). Dieser Ansatz basiert auf drei Heuristiken, die auf einen natürlichsprachlichen Fachtext angewendet werden:

1. "Mark every noun as candidate class which is not a meaningless expletive or part of a description clarifying some previous expressed information." [Jeck98, S. 3]

2. "Check every identified noun (candidate class) if it can receive a value expressible using the basic data types of the intended implementation language. If so the consider expressing the noun as attribute of a class." [Jeck98, S. 4]

3. "A verb within the textual specification linking two nouns (i. e. concepts) together could be an association within the analysis model. If one of the stated nouns is previously identified as attribute, the verb expresses the implicit interrelation to the containing class. Else all the related nouns were identified as candidate classes, they will be interrelated using a UML association." [Jeck98, S. 5]

Der Modellentwickler soll also zunächst alle Substantive im natürlichsprachlichen Fachtext als Klassenkandidaten markieren, dann bewerten, ob die bedeutungslos oder Teil einer erklärenden Wiederholung sind. Kann ein gefundene Klassenkandidat einem Basisdatentyp der zu implementierenden Programmiersprache zugeordnet werden, wird er als Attribut modelliert. Mit den Verben werden Beziehungen im Modell angelegt.

In [Jeck98] ist ein Beispiel angegeben, in dem mit Hilfe dieser Heuristiken aus einer Prozessbeschreibung ein Klassenmodell entwickelt wird. Es wird deutlich, dass die Identifikation der Objekt- und Attributkandidaten sowie ihrer Beziehungen intellektuelle Fähigkeiten bei der Anwendung der Heuristiken erforderlich sind, da sich die Entscheidungsfindung über die Bedeutungslosigkeit einzelner Worte, ihre Zuordnungsmöglichkeit zu einem Datentyp oder selbst die Zugehörigkeit von Substantiven zu Verben nicht oder nur mit großem Aufwand (s. o. 0) automatisieren lässt.

In [GeMa03] sind ebenfalls mit Syntaxmustern, Patternorientierter Prozessmodellierung und Sprachnormierung Heuristiken zur Extraktion von Modellinformationen aus natürlichsprachlichen Texten vorgestellt, die mangels Toolunterstützung ebenfalls nur in Modellierungsregeln festgelegt werden können und durch manuell angewandt werden.

4. Kopplung der Verfahren

Die in den vorangegangenen Abschnitten vorgestellten Verfahren zeichnen sich entweder durch hohen Aufwand (NIBA, vgl. 0, bzw. Heuristiken, vgl. 0) oder durch Unvollständigkeit (Wortschatz-Technologie, vgl. 0) aus. Koppelt man die Wortschatz-Technologie mit den Heuristiken, die Jeckle vorgestellt hat, zu einer neuen Methode, eröffnet sich die Möglichkeit zur automatischen Extraktion von kontextspezifischen Fachwissen, das bei der Modellierung von Fachräumen genutzt werden kann.

Nachfolgend soll der dem gekoppelten Verfahren zu Grunde liegende Algorithmus an einem Beispiel erläutert werden, ehe in einem Ausblick die noch zu beantwortenden Fragen aufgeführt werden.

Grundlage des Algorithmus' sind Textdokumente D des der Software zu Grunde liegenden Fachkontexts. Diese werden mit Hilfe der Wortschatz-Technologie in Worte im Singular W und Zitate Z zerlegt. Außerdem wird eine Liste der Trivialworte und Eigennamen T sowie die Liste der attribut- und werttypischen Worte A gebildet. Unter einem "attributtypischen" Wort sind zu verstehen, die von der Bezeichnung her auf ein Attribut schließen lassen, z. B. ID, Name, Bezeichnung, Größe, Farbe. "Werttypische" Worte stellen Werte, wie true, false, Millionen, DM, Euro, cm, dar. Die attribut- und werttypischen Worte sind manuell zu bilden bzw. zu verwalten und werden in den Modellierungsprojekten fortgeschrieben. Es gilt: $D \supset Z \supset W \supset A$, $W \supset T$.

Um alle Zitate zu finden, die ein bestimmtes Wort $w \in W$ enthalten, wird eine Abbildung Z' definiert: $Z': W \rightarrow Z: w \in W \rightarrow z' \in Z'(W) \mid w \in z'$.

Das Wort und seine Zitate werden in einem Tupel $B(W_B, Z'(W_B))$ gespeichert, welches gemäß dem semiotischen Dreieck der Sprachwissenschaft (vgl. [HeSi80, S. 236]) den Begriff B widerspiegelt.

Zu jedem Begriff wird die Häufigkeitsklasse H_B gespeichert (vgl. [Heye02, S. 3), mit der der Begriff im Kontext auftritt. Die häufigsten Begriffe, die keine Trivialworte T darstellen, werden relevante Begriffe B_r genannt. Es gilt: $\forall b \in B_r: H_B(W_b) > H_S \wedge W_b \notin T$, wobei H_S als signifikante Häufigkeit für Begriffe definiert ist.

Die Beziehungen zwischen Begriffen lassen sich durch die Stärke von Kookkurrenzen bestimmen (vgl. [Heye02, S. 4]). Die Stärke der Kookkurrenzen $Ko(B_1, B_2)$ zwischen den relevanten Begriffen lässt sich als Graph darstellen. Aus Übersichtlichkeitsgründen soll der Graph nur Kookkurrenzen enthalten, deren Häufigkeit einen bestimmten Wert H_K übersteigen. Die Menge der Kookkurrenzen mit signifikant hoher Häufigkeit stellt die Menge der relevanten Kookkurrenzen dar. $\forall k \in K_r(B1, B2): K(B1, B2) > H_K$.

In Anlehnung an die Heuristiken in [Jeck98] werden auf die Menge der relevanten Kookkurrenzen Heuristiken angewendet. Dazu werden die Menge der UML-Klassen Kl_B und UML-Attribute Att_B gebildet, wobei das tiefgestellte B die Bezeichnung der Klasse mit dem Begriff B andeutet. Die Heuristiken lauten:

1. Jeder Begriff, der relevante Kookkurrenzen zu mindestens zwei anderen Begriffen besitzt, wird Klassenkandidat.
   $\forall b \in W_B, B1, B2 \in W$:
   $\exists Ko(b, B1)>0 \wedge \exists Ko(b, B2) \wedge B1 \neq B2 \Rightarrow b \rightarrow Kl_b$

2. Jeder Begriff, der relevante Kookkurrenzen zu nur einem anderen Begriff besitzt, wird Attribut: $\forall b \in W_B, B1,B2 \in W$:
   $\forall Ko(B, B1) > 0 \wedge \forall Ko(B, B2) > 0 \Leftrightarrow B1 = B2 \Rightarrow b \rightarrow Att_b$

3. Ein Begriff, der ein "attribut-" bzw. "werttypisches" Wort enthält, wird Attribut.
   $\forall b \in W_B: b \in A \Rightarrow \exists Att_b$

Im Gegensatz zu Jeckles Heuristik 2 (vgl. Abschnitt 0) werden die Worte automatisch gegen eine Liste geprüft, in der attribut- bzw. werttypische Worte enthalten sind.

Während bei [Jeck98, S. 5] die UML-Assoziation mit Hilfe von Verben gebildet wird und somit entweder die Infrastruktur von NIBA (vgl. Abschnitt 0) oder intellektuelle Leistungen erfordert, wird im vorgestellten Ansatz die UML-Assoziation mit Hilfe der relevanten Kookkurrenzen gebildet, die automatisch ausgewertet werden können.

5. Beispiel: kommunales Ratsinformationssystem

Der vorgestellte Algorithmus soll nun am Beispiel eines Ratsinformationssystems dargestellt werden. Diese Informationssysteme dienen der Arbeit der kommunalen Parlamente und zum Austausch von Terminen, Agenden und Protokollen der Sitzungen der Parlamente und Ausschüsse.

Im Wortschatztool wurde eine Suche nach den Begriffen *Gemeinderat, Sitzung* und *Tagesordnung* durchgeführt. Die Suche lieferte die in Abbildung 2 enthaltenen Kookkurrenzen.

im (939), der (625), beschlossen (480), hat (392), Verwaltung (364), Bürgermeister (355), Mehrheit (326), Sitzung (258), Stadt (241), Stadtverwaltung (238), Stuttgarter (219), für (212), Oberbürgermeister (211), Der (203), zugestimmt (186), einstimmig (151), dem (146), beschloß (133), Esslinger (131), jetzt (126), Stimmen (120), Bürgerentscheid (119), vom (114), Beschluß (112), Grünen (107), Fraktionen (105), Antrag (104)

Abbildung 2: Kookkurrenzen zu den Worten Gemeinderat, Sitzung und Tagesordnung

Werden die Trivialworte T ausgeblendet, bleiben die in Abbildung 3 genannten Kookkurrenzen. Bei der Analyse der Worte wird deutlich, dass die Begriffe *beschlossen*, *beschloß* und *Beschluß* sowie die Begriffe *Bürgermeister* und *Oberbürgermeister* synonym verwendet werden.

beschlossen (480), Verwaltung (364), Bürgermeister (355), Mehrheit (326), Sitzung (258), Stadt (241), Stadtverwaltung (238), Oberbürgermeister (211), zugestimmt (186), beschloß (133), Bürgerentscheid (119), Beschluß (112), Fraktionen (105), Antrag (104)

Abbildung 3: Kookkurrenzen zu den Worten Gemeinderat, Sitzung und Tagesordnung

Ermittelt man die Stärke der Kookkurrenzen, ergibt sich bei einem Signifikanzlevel H_K=245 der in Abbildung 4 dargestellte Graph. Die Dicke der Kanten gibt die Stärke der Kookkurrenzen an.

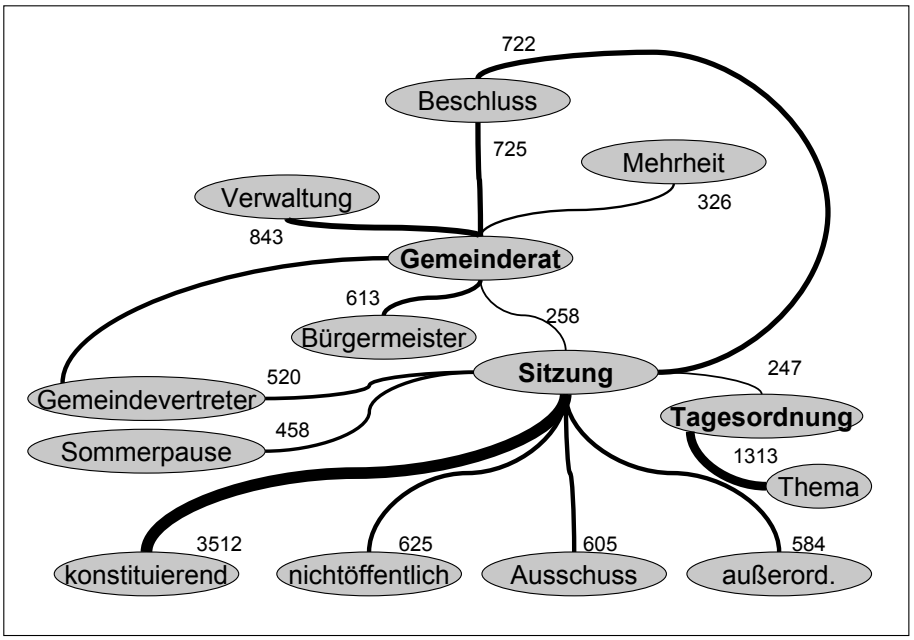

Abbildung 4: Graph über die signifikanten Kookkurrenzen

Aus Heuristik 1 folgt, dass alle Substantive im Graph Klassenkandidaten im UML-Klassenmodell werden. Da die Begriffe *konstituierend*, *nichtöffentlich*, *Ausschuss*, *außerordentlich*, *Thema*, *Mehrheit*, *Verwaltung*, *Bürgermeister* und *Sommerpause* nur Beziehungen zu einem anderen Begriff haben, werden nach Heuristik 2 Attribute. Somit lässt sich das in Abbildung 5 abgebildete UML-Modell generieren.

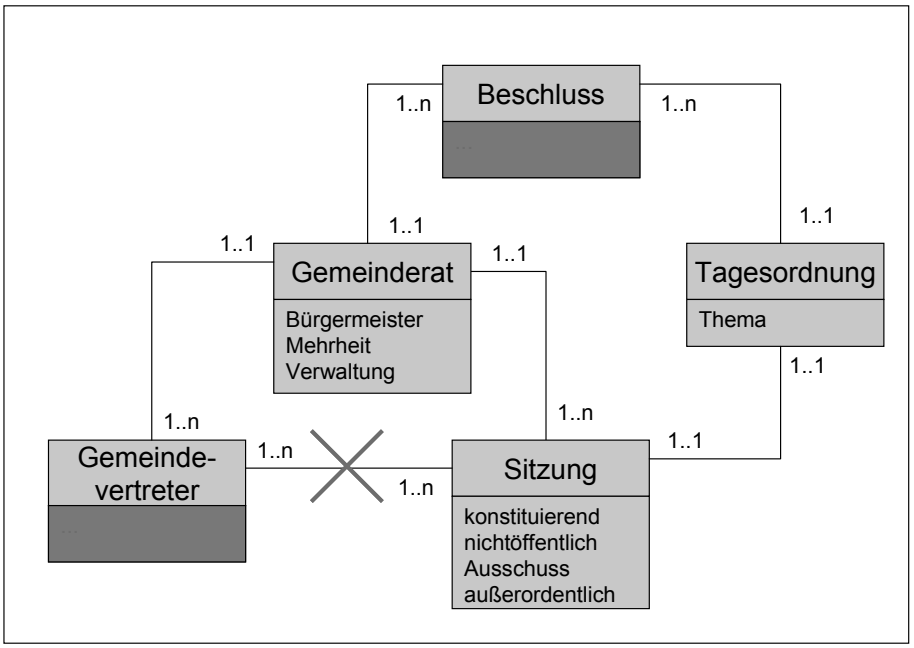

Abbildung 5: generiertes UML-Modell

Im generierten UML-Klassenmodell wurden die Klassen *Gemeinderat, Gemeindevertreter, Sitzung, Tagesordnung* und *Beschluss* generiert. Die Klassen *Gemeindevertreter* und *Beschluss* haben keine Attribute. Bei näherer Betrachtung ist die Beziehung zwischen den Klassen *Gemeindevertreter* und *Sitzung* nicht notwendig. Im Ratsinformationssystem werden Sitzungsprotokolle u. ä. Dokumente verwaltet. Die Anwesenheit einzelner Gemeindevertreter an konkreten Sitzungsterminen spielt in der Regel keine Rolle.

Die Diskussion mit Fachexperten ergibt, dass das Attribut *Bürgermeister* in der Klasse *Gemeinderat* seine Berechtigung hat, da der Bürgermeister sowohl Chef der Gemeindeverwaltung ist als auch den Vorsitz im Kommunalparlament hat und hier in der Rolle eines Gemeindeverteters auftritt. Das Attribut *Bürgermeister* weist somit auf den Gemeindevertreter, der den Vorsitz im Gemeinderat innehat.

Das aus dem Graph generierte UML-Modell bietet einen ersten Ansatz für ein Klassenmodell des zu entwickelnden Ratsinformationssystems. Die aufgetretenen Fehler, wie leere Klassen und überflüssige Beziehungen, sind nachvollziehbar, da als Grundlage für das Modell der Inhalt des Deutschen Wortschatzes, also öffentlich zugängliche Quellen, und keine Spezifikation aus dem entsprechenden Kontext verwendet wurde. Für ein Modell, dass den Anforderungen eines Kommunalpolitikers entspricht, müssen weitere Informationen, wie Zugangsberechtigungen, Sitzungsdokumente o. ä. mit abgebildet werden.

6. Stand des Projekts – Ausblick – Bezug zu SOA

Anhand eines Beispiels konnte der Ablauf des Algorithmus' demonstriert werden. Es verdeutlicht, dass theoretisch durch die Kopplung von Verfahren zur statistisch-linguistischen Analyse und Heuristiken UML-Klassendiagramme aus natürlichsprachlichen Dokumenten erzeugt werden können. Die Umsetzbarkeit dieses theoretischen Ansatzes ist durch eine protoypische Implementierung nachzuweisen.

Darüber hinaus wird zu zeigen sein, dass dieses Verfahren auch für umfangreiche Dokumente in akzeptabler Zeit Ergebnisse liefert, mit denen ein fachfremder Modellentwickler schnell einen Überblick über den Fachraum gewinnt und mit den Fachexperten die nötigen Änderungen durchführen kann. Insbesondere sind die Höhe der Signifikanzwerte H_S und H_K sowie die Gültigkeit der Heuristiken für Fachtexte zu prüfen.

Anschließend ist zu untersuchen, inwiefern sich Beziehungen zwischen Klassenkandidaten mit Hilfe der Zitate, in denen sie vorkommen, als Aggregationen oder Spezialisierungen identifizieren lassen. Gleiche Überlegungen sind für Verhaltensdiagramme zu stellen. Die Frage ist, ob aus sich Begriffsbeziehungen (dann auch zu anderen Wortarten, wie Verben und Adjektiven) oder Zitaten allgemeingültige oder kontextspezifische Aussagen zu Prozessabläufen ableiten lassen.

Für die Spezifikation von Services müsste das Verfahren und Heuristiken erweitert werden, mit denen auch Operationen der Klassen und deren Reihenfolge hintereinander ermittelt werden könnte. Aus den Operationen lassen sich dann mit Hilfe von UML-Activity-Diagrams oder auch Ereignisgesteuerten Prozessketten Abläufe definieren, die als Services gekapselt werden können. [GeMa03]

Hierfür bietet das Wortschatztool bereits eine Auswertung, bei der signifikante rechte und linke Nachbarn angezeigt werden können. [FlMM00] Außerdem könnten die Kookkurrenzen auf den Worttyp *Verb* beschränkt erste Ansätze für mögliche Operationen bilden.

Literaturverzeichnis

[BKnM08] Beverungen, Daniel/ Knackstedt, Ralf/Müller, Oliver: Entwicklung Serviceorienterte Architekturen zur Integration von Produktion und Dienstleistung – Eine Konzeptionsmethode und ihre Anwendung am Beispiel des Recyclings elektronischer Geräte
in Wirtschaftsinformatik, Heft 3/2008, S. 220-234

[Chom65] Chomsky Noam: Aspects of the Theory of Syntax, 1965.

[EiPi07] Eisele, Markus/Pizka, Markus: SOA² Report – State of the Art Service-Orientierter Architekturen, in: http://www.software-kompetenz.de, v. 19.04.2007

[FeLo03] Fettke, Peter/Loos, Peter: Model Driven Architecture (MDA).
in: Wirtschaftsinformatik. 45 (2003), Nr. 5, S. 555-559

[FeSi01] Ferstl, Otto K./Sinz, Elmar J.: Grundlagen der Wirtschaftsinformatik : Band 1. 4., überarb. und erw. Auflage. München : Oldenbourg, 2001

[FlKM00] Linguistische Aspekte des Projekts NIBA, Klagenfurt 2000.
[FlMM00] Fliedl, Günther/Mayerthaler, W/Mayr, Heinrich C. et. al.: The NIBA Workflow – From textual requirements specifications to UML-Schemata, Klagenfurt 2000.
[FlWe00] Fliedl, Günther/Werner, Günther: NIBA-TAG – A tool for analyzing and preparing german texts, 2000.
[Fran03] Frankel, David S.: Model Driven ArchitectureTM : Applying MDATM to Enterprise Computing. 1. Auflage, Indianapolis, Wiley Publishing, Inc. 2003
[Fuh96] Norbert Fuhr: Skriptum Information Retrieval, Lecture Notes on Information Retrieval, Universtität Dortmund, 1996
[GeMa03] Gerber, Stefan/Mai, André: Modellieren mit Begriffen - ein Vorgehensmodell zur Wiederverwendung von Ergebnissen der Analysephase im Design, in: Petrasch, Roland/Wiemers, Manuela/Kneuper, Ralf (Hrsg): Praxistauglichkeit von Vorgehensmodellen : 10. Workshop der Fachgruppe WI-VM der Gesellschaft für Informatik e. V. (GI), Aachen, Shaker 2003
[HeSi80] Herberger, Maximilian/Simon, Dieter: Wissenschaftstheorie für Juristen : Logik, Semiotik, Erfahrungswissenschaften, Frankfurt am Main, Metzner 1980
[Herz05] Herz, Richard Alan: Stichwort SOA – Serviceorientierte Architekturen, WIAI-Stammtisch 2.11.2005
[Heye02] Heyer, Gerhard/Quasthoff, Uwe et al.: Möglichkeiten und Verfahren zur autormatischen Gewinnung von Fachbegriffen aus Texten in: Bullinger, H.-J. (ed.) .Proc. Innovationsforum Content Management –Digitale Inhalte als Bausteine einer vernetzten Welt, Stuttgart 2002.
[Jeck98] Jeckle, Mario: On formalising OOA heuristics – Bridging the gap between analysis and design, 1998
[Jec+04a] Jeckle, Mario/Rupp, Chris/Zengler, Barbara et. al.: UML 2.0 – Neue Möglichkeiten und alte Probleme, in: Informatik Spektrum, Bd. 27 (2004), Heft 4, S. 323 – 331
[Jec+04b] Jeckle, Mario ; Rupp, Chris ; Zengler, Barbara et. al.: UML 2 glasklar. 1. Auflage. München. Carl Hanser, 2004
[JKSK00] Jungings, S./Kühn, H./Strobl, R./Karagiannis, D.: Ein Geschäftsprozessmanagement-Werkzeug der nächsten Generation- Adonis: Konzeption und Anwendung, Wirtschaftinformatik, 5, 2000.
[Oest04] Oestereich, Bernd: Objektorientierte Softwareentwicklung. Analyse und Design mit der UML 2.0. 3. Auflage, München : Oldenbourg, 2004.
[OMG00] OMG: Model Driven Architecture : White Paper Draft 3.2. ftp://ftp.omg.org/pub/docs/omg/00-11-05.pdf 25.01.2005 - OMG
[OMG01] OMG: Model Driven Architecture (MDA). http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01 25.01.2005 - OMG
[Ortn00a] Ortner, Erich: Wissensmanagement, Teil 1: Rekonstruktion des Anwendungswissens, in: informatik spektrum, vom 23.04.2000.
[Ortn00b] Ortner, Erich: Wissensmanagement, Teil 2: Systeme und Werkzeuge, in: informatik_spektrum vom 23.06.2000.
[Part98] Partsch, Helmuth: Requirements-Engineering systematisch : Modellbildung für softwaregestützte Systeme., o. Auflage., Berlin. Springer, 1998.
[Rump05] Rumpe, Bernhard: Agile Modellierung mit UML : Codegenerierung, Testfälle, Refactoring. 1. Auflage. Berlin : Springer, 2005
[Rupp01a] Requirements Engineering und Management, Professionelle, iterative Anforderungsanalyse für die Praxis, Hanser 2001; www.sophist.de
[Rupp01b] Requirements Engineering – der Einsatz einer natürlich-sprachlichen Methode bei der Ermittlung und Qualitätsprüfung von Anforderungen, in: Proceedings der Net.ObjectDays 2001, net.objectdays.org
[Rupp03] Rupp, Chris: Wie aus Wünschen Anforderungen werden. in: InformationWeek Nr. 22 vom 30. Oktober 2003, S. 24-25

[RuSc04] Rupp, Chris/Schimpfky, Nicole: Erfahrungsbericht über einen Spezifikationsprozess mit einem bunten Reigen an Methoden und Notationen, in: Softwaretechnik-Trends, Band 24, Heft 4, November 2004

[Sche98] Scheer, August-Wilhelm: ARIS – Vom Geschäftsprozess zum Anwendungssystem, Berlin, Heidelberg, New York, Tokio, Springer 1998.

[Sun90] D. Sunday: A Very Fast Substring Search Algorithm. Communications of the ACM (CACM), Volume 33, Number 8: S. 132-142 (1990)

[Wiem03] Wiemers, Manuela: Usability und Vorgehensmodelle, in: Petrasch, Roland/Wiemers, Manuela/Kneuper, Ralf (Hrsg): Praxistauglichkeit von Vorgehensmodellen – 10. Workshop der Fachgruppe WI-VM der Gesellschaft für Informatik e. V., Aachen, Shaker 2003

[WikiSOA] Begriff SOA in Wikipedia, abgerufen am 11.08.2008

[Wort] Wortschatztool der Universität Leipzig, http://www.wortschatz.uni-leipzig.de, zuletzt abgerufen am 27.06.06