SEMANTISCHES TASKMODELL FÜR DIALOGE MIT TASKWECHSEL

Sven Reichel[1], Philipp Pfarherr[1], Ute Ehrlich[1], André Berton[1] und Michael Weber[2]
[1] Speech Dialogue Systems, Daimler AG, Ulm
[2] Institut für Medieninformatik, Universität Ulm
sven.reichel@daimler.com

Kurzfassung: Infotainmentsysteme im Fahrzeug haben heutzutage oftmals Zugang zum Internet und erweitern ihren Funktionsumfang durch Online-Anwendungen. Die Aufgaben, die ein Nutzer mit solch einem System ausführen möchte, sind nicht mehr nur an einzelne Anwendungen gerichtet, sondern involvieren mehrere. Sprachdialogsysteme ermöglichen eine kurze und effiziente Interaktion über Anwendungsgrenzen hinweg und sichern den Blick auf der Straße und die Hände am Lenkrad. In diesem Beitrag wird eine Methode vorgestellt, wie Multi-Anwendungsdialoge - mit Werteübergabe zwischen Anwendungen - effizient in einem Sprachdialogsystem implementiert werden können. Hierfür werden die einzelnen Anwendungen, mit ihren Ein- und Ausgabewerten, semantisch durch eine globale Ontologie beschrieben. Der Taskmanager kann auf dieser Basis die nächstmöglichen Aktionen vorhersagen und stellt entsprechende Information inklusive einer Historie dem Dialogmanager zur Verfügung. Eine Proof-of-Concept-Umsetzung zeigt, dass eine Integration in ein Sprachdialogsystem denkbar wäre, aber weiterer Forschungsbedarf hinsichtlich der Ontologie besteht.

1 Einleitung

Durch die starke Verbreitung von Smartphones setzen immer mehr Kunden voraus, auch im Fahrzeug vernetzt zu sein. Eine Bedienung des Smartphones lenkt von der Fahrsituation ab und ist daher in vielen Ländern hinter dem Steuer verboten. Trotz des Verbots verursachen abgelenkte Fahrer weiterhin Unfälle [6]. Aus diesem Grund bieten mehr und mehr Infotainmentsysteme Online-Funktionalitäten an. Wie beim Smartphone werden die einzelnen Funktionen solcher Systeme in Anwendungen gekapselt, die die Onboard-Funktionalitäten (z.B. Navigation oder Telefon) mit Offboard-Inhalten (z.B. Mercedes-Benz Apps wie News oder Restaurantsuche) erweitern. Allerdings sind die Aufgaben der Nutzer nicht immer auf eine Anwendung beschränkt. Angenommen der Fahrer sucht ein Restaurant zum Mittagessen und wählt eine Pizzeria aus. Wenn er dort anrufen möchte und die Restaurantanwendung nicht explizit mit der Telefonanwendung verknüpft ist, muss er die Anwendung wechseln und die Telefonnummer manuell eingeben. Bei einfach zu merkenden Daten mag dies während der Fahrt funktionieren. Sollte die Nummer allerdings länger sein, wird der Fahrer diese Aufgabe kaum ausführen können. Eine Verknüpfung von Anwendungen und ein Datenaustausch ist somit unerlässlich.

Gerade beim Autofahren ist es wichtig, kurze und effiziente Interaktionen zwischen Mensch und Maschine zu gewährleisten, um die Ablenkung des Fahrers so gering wie möglich zu halten. Ein **S**prach**d**ialog**s**ystem (SDS) bietet dem Nutzer die Möglichkeit, komplexe Aufgaben einfach auszuführen und ermöglicht ihm, den Blick auf der Straße sowie die Hände am Lenkrad zu behalten. Dies trägt entscheidend zur Fahrsicherheit bei [4]. Im Gegensatz zur grafisch/haptischen Bedienschnittstelle, die durch die begrenzte Bildschirmgröße meist hierarchisch strukturiert wird, bietet ein SDS die Chance viele verschiedene Äußerungen direkt zu verstehen. Dies er-

möglicht eine einheitliche, aufgabenorientierte Bedienschnittstelle, wie beschrieben in [9]. Aus Sicht des Nutzers muss er sich in einem Sprachdialog nicht an Anwendungsgrenzen halten, sondern kann mit anaphorischen Äußerungen, wie zum Beispiel „Ich möchte diese Pizzeria anrufen“, direkt sein Interaktionsziel erreichen.

Ein SDS, welches Dialoge über verschiedene Domänen hinweg ermöglicht, stellt hohe Anforderungen an die Entwickler. Anwendungssprünge, sowie der Datenaustausch, müssen definiert werden. Speziell bei vielen Anwendungen oder dynamisch erweiterbaren Systemen (z.B. durch Download neuer Anwendungen) steigt der Aufwand. Wir stellen in diesem Beitrag eine Methode vor, wie Multi-Domänen-Dialoge - mit Werteübergabe zwischen Anwendungen - effizient in einem Sprachdialogsystem implementiert werden können. Hierfür werden Anwendungen und ihre Ein- und Ausgabewerte semantisch beschrieben und zur Laufzeit dynamisch mit einer globalen Ontologie zu einem semantischen Taskmodell verbunden. Durch die Ein- und Ausgabewerte verschiedener Anwendungen bzw. Aktionen, ergeben sich mögliche Dialogpfade durch das Taskmodell.

Der restliche Teil dieses Beitrags ist folgendermaßen gegliedert: In Abschnitt 2 wird auf bestehende Multi-Domänen-SDS eingegangen. Daraufhin zeigen wir unser Konzept des semantischen Taskmanagers mit Ontologie, sowie die Anwendungsbeschreibung in Abschnitt 3. Die Implementierung in RDF und Jena, wie beschrieben in Kapitel 4, bildet die Basis für die Evaluierung (Kapitel 5). Wir schließen in Abschnitt 6 mit der Erkenntnis, dass ein semantischer Taskmanager Multi-Domänen-Dialoge ermöglicht.

2 Multi-Domänen Sprachdialogsysteme

Ein SDS kann in mehrere Module unterteilt werden, wie unter anderem von Carstensen et al. [1] beschrieben (siehe Abbildung 1). Über ein Analyse- und Interpretationsmodul wird die Äußerung des Nutzers erkannt und im Dialogmanager verarbeitet. Dieser stößt die vom Nutzer gewünschte Aktion über den Taskmanager an und veranlasst die Ausgabe über eine Präsentationsplanungs- und Generierungskomponente. Dieser Beitrag betrachtet hauptsächlich den Taskmanager, der mittels eines semantischen Netzes dynamisch Anwendungen im Backend verwaltet und passende Schnittstellen für den Dialogmanager bereitstellt, um Dialoge mit Taskwechsel zu ermöglichen.

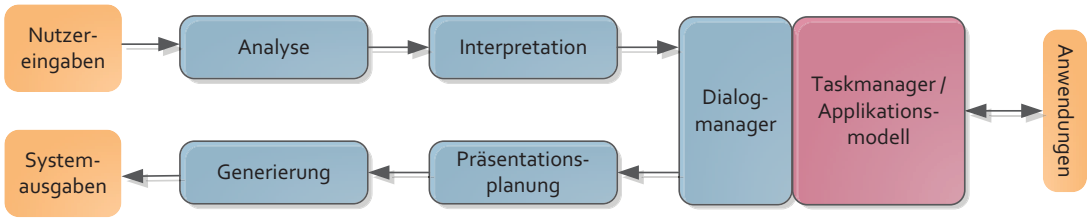

Abbildung 1 - Architektur eines Dialogsystems (nach Carstensen et al. [1])

Multi-Domänen SDS ermöglichen dem Nutzer, Dialoge mit mehreren Anwendungen zu führen. Diverse Systeme unterstützen dies, unter anderem SmartKom [10] und CHAT [13]. SmartKom bietet 50 verschiedene Funktionalitäten in 14 Anwendungen mit einer kohärenten Bedienschnittstelle und CHAT drei Anwendungen. Beide Systeme interpretieren die Nutzereingabe und bestimmen auf dieser Basis die zuständige Anwendung. Einen ähnlichen Ansatz verfolgt Lee et al. [3], indem erst jede Anwendung mit der Eingabe konfrontiert wird und diejenige, die die Anfrage am besten interpretieren kann, ausgeführt wird. Unser Ansatz geht nicht von der Nutzereingabe aus, sondern von den Anwendungen und kann dem Dialogmanager die wahrscheinlichsten, nächsten Aktionen zur Verfügung stellen. Dies erlaubt eine Optimierung des

Sprachverstehens, da eine Einschränkung je nach Kontext möglich wird.

Das Vereinigen von mehreren einzelnen Anwendungen bzw. Services zu komplexeren ist im Bereich der **s**ervice**o**rientierten **A**rchitektur (SOA) lange bekannt. Zum Beispiel nutzen Rao et al. [8] DAML-S zur semantischen Beschreibung von Services, die sie über eine lineare Logik zu einem Prozessmodell zusammenfügen. Eine globale Ontologie, wie in diesem Beitrag gezeigt, besteht allerdings nicht. Anders bei Zhang et al. [14], die zusätzlich eine Webservice-Ontologie nutzen, um mit einem Shortest-Path-Algorithmus eine Servicesequenz zu erstellen und damit eine Nutzeraufgabe zu erfüllen. Diese Arbeiten betrachten nicht die Integration in ein SDS, was wir in diesem Beitrag vorstellen.

3 Konzept eines semantischen Taskmanagers

Wie im vorherigen Abschnitt gezeigt, bildet ein Taskmanager die Schnittstelle zwischen dem Dialogmanager und verschiedenen Anwendungen. Er kapselt die Funktionalitäten der Anwendungen und bindet diese, inklusive ihrer Ein- und Ausgabewerte, in eine Ontologie ein. Eine Instanz dieser Ontologie bildet ein domänenübergreifendes semantisches Taskmodell, welches dem Dialogmanager abhängig vom Dialogkontext eine Aktionsliste und Wertehistorie zur Verfügung stellt. Bei entsprechenden Nutzeräußerungen wird die benötigte Aktion aufgerufen und von der jeweiligen Anwendung ausgeführt.

3.1 Domänenübergreifendes semantisches Taskmodell

Das semantische, anwendungsübergreifende Taskmodell ist ein Verbund aus einzelnen Taskmodellen der Anwendungen, die zur Laufzeit mit einer globalen Ontologie eine Instanz bilden. Dieser Abschnitt beschreibt die Ontologie, die Definition von Anwendungen und die Erzeugung der Instanz des Taskmodells.

3.1.1 Ontologiedefinition

Um ein zusammenhängendes Taskmodell und damit einen Informationsaustausch über Anwendungen hinweg zu ermöglichen, ist eine Modell nötig, das den Schluss von Verbindungen zwischen Anwendungen erlaubt. Die Ontologie definiert Objekte der realen Welt, mit welchen die Aktionen von Anwendungen und ihre zugehörigen Ein- und Ausgabewerte beschrieben werden können (siehe Abschnitt 3.1.2). Diese Werte können beliebige Objekte sein, was hohe Anforderungen an die Granularität des semantischen Netzes stellt [7]. Um ein zusammenhängendes Taskmodell zu gewährleisten, muss die Onotologie verbunden sein. Die **W**eb **O**ntology **L**anguage (OWL) sieht hier das Konzept `owl:Thing` vor, welches Oberklasse eines jeden Objekts ist.

3.1.2 Semantische Beschreibung von Anwendungen

Eine jede Anwendung kann mehrere Aktionen besitzen, die die Aufgaben, die ein Nutzer an das SDS stellt, ausführen. Hierbei ist jede Aktion als atomare Einheit zu sehen. Sie wird von einem externen Modul ausgeführt und ist nicht Teil des Taskmanagers. Zum Beispiel kann dies eine Webanbindung um Restaurants zu suchen, die Steuerung der Klimaanlage des Fahrzeugs oder aber ein Telefon sein. Jede Aktion kann hierbei Eingabewerte benötigen, wie zum Beispiel die Telefonnummer des Anzurufenden. Weiterhin kann eine Aktion Ausgabewerte zur Verfügung stellen, wie beispielsweise die Telefonnummer eines Restaurants.

Der Anwendungsmodellierer beschreibt die Eingabe- und Ausgabewerte der Aktion, sowie die Aktion selbst, indem er auf Objekte in der Ontologie verweist. Hierfür stehen die Relationen `hat_Eingabewert`, `hat_Ausgabewert` und `hat_Funktion` zur Verfügung. Dabei ist es wichtig, eine eindeutige Abbildung in die Ontologie zu gewährleisten, um einen Dialogfluss

über Anwendungen hinweg zu ermöglichen. Wie in Abschnitt 3.1.1 erwähnt, ist die Granularität der Ontologie kritisch. Bei einer zu groben Granularität kann es passieren, dass Aktionen und Werte existieren, die in der Ontologie nicht abgebildet werden können. Aus diesem Grund ist es dem Anwendungsmodellierer möglich, lokal in der Anwendung das semantische Netz so zu erweitern, bis eine Abbildung in die globale Ontologie stattfinden kann.

3.1.3 Erzeugung eines anwendungsübergreifenden Taskmodells

Im schnelllebigen Internetzeitalter ist es wichtig, auf aktuelle Trends reagieren zu können. Dies stellt die Anforderung, neue Anwendungen auch zur Laufzeit nachladen zu können. Um dem Folge zu leisten, wird das anwendungsübergreifende Taskmodell zur Laufzeit dynamisch aus den verfügbaren Anwendungen und der Ontologie zusammengesetzt. Abbildung 2 zeigt ein vereinfachtes Beispiel mit den beiden Aktionen *Restaurant suchen* und *Nummer anrufen* aus zwei verschiedenen Anwendungen. In der semantischen Beschreibung beider Aktionen ist definiert auf welche Objekte in der Ontologie referenziert wird. Im Beispiel definiert die *Restaurant suchen* Aktion eine *Telefonnummer* als Ausgabewert. Die Aktion *Nummer anrufen* verweist ebenso auf *Telefonnummer* und erwartet diesen Wert als Eingabe. Alle Anwendungen im Infotainmentsystem verweisen auf die Ontologie, so dass die Instanz des Modells die Gesamtfunktionalität des Infotainmentsystems ausdrückt und eine Informationsübergabe definiert.

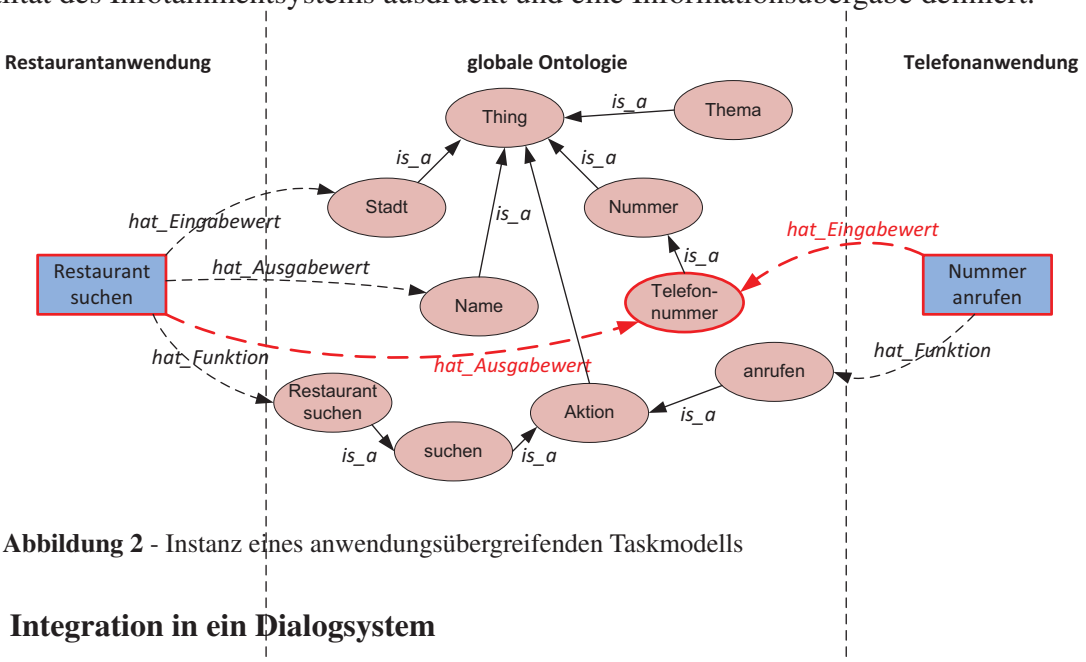

Abbildung 2 - Instanz eines anwendungsübergreifenden Taskmodells

3.2 Integration in ein Dialogsystem

Wie in Abschnitt 2 gezeigt, steht der Taskmanager zwischen dem Dialogmanager und den Anwendungen. Er verwaltet Schnittstellen zu Anwendungen und ermöglicht somit eine Ausführung von Aktionslogik. Die Schnittstelle zum Dialogmanager beschreibt einerseits eine Aktionsliste der im aktuellen Kontext ausführbaren Aktionen, und andererseits eine Wertehistorie.

3.2.1 Liste der im Kontext ausführbaren Aktionen

Die Aktionsliste wird dem Dialogmanager zur Verfügung gestellt und beinhaltet eine Auflistung der im aktuellen Kontext ausführbaren Aktionen, inklusive der erforderlichen und vorhandenen Werte, sowie einer Wahrscheinlichkeit zur Ausführung (siehe Tabelle 1). Die Kommunikation mit dem Nutzer wird durch den Dialogmanager gesteuert. Das heißt, dieser ermittelt die Aktion, die der Nutzer auszuführen wünscht, sowie die dafür notwendigen Eingabewerte.

Tabelle 1 zeigt das Beispiel einer Aktionsliste nach einer Restaurantsuche. Die Aktion *Restaurant suchen* aus Abbildung 2 wurde aufgerufen und es wurde ein Restaurant mit dem

Tabelle 1 - Aktionsliste nach einer Restaurantsuche

Aktion	**notwendige Eingabewerte**	**Übergangs-wahrscheinlichkeit**	**Vorhandene Eingabewerte**
Restaurant bewerten	Bewertung, Name	hoch	Name[Pizzeria Ulmia]
anrufen	Telefonnummer	niedrig	Telefonnummer[0731-229]
Hilfe abrufen	Thema	niedrig	

Namen[Pizzeria Ulmia] und der *Telefonnummer[0731-229]* gefunden. Angenommen die Restaurantanwendung hätte eine Aktion *Restaurant bewerten* mit *Namen* und *Bewertung* als Eingabeparameter. Dann wäre im aktuellen Kontext möglich, einerseits das Restaurant Ulmia zu bewerten, andererseits es anzurufen. Durch die vorhandenen Eingabewerte zu den Aktionen wird dem Dialogmanager bekannt gegeben, welche Werte noch zusätzlich durch den Nutzer eingeben werden müssen. Bei der Aktion *Restaurant bewerten* könnte dies zum Beispiel die *Bewertung* in Form einer Schulnote sein. Wohingegen die Aktion *anrufen* keine weiteren Werte benötigt und mit einer Äußerung „Rufe die Pizzeria an“ initiiert werden kann. Die Planung der Dialoge übernimmt der Dialogmanager.

Menschen agieren und kommunizieren oftmals innerhalb eines bestimmten Themengebiets. Dem tragen Anwendungen Rechnung, indem sie thematisch ähnliche Aktionen gruppieren. In der Aktionsliste sind somit Aktionen, die zur selben Anwendung wie die zuletzt ausgeführte Aktion gehören, mit einer hohen Übergangswahrscheinlichkeit gekennzeichnet. Aktionen von anderen Anwendungen, die über die Ontologie verbunden werden, erhalten eine niedrige Übergangswahrscheinlichkeit. Bisher wird lediglich die Wahrscheinlichkeit anhand der Aktionszugehörigkeit zwischen hoch und niedrig unterschieden. Dies soll dem Dialogmanager bei der Entscheidung helfen. Beim Anwendungswechsel sollte sicher gestellt werden, dass der Nutzer dies wirklich beabsichtigt, zum Beispiel über eine explizite Nachfrage.

Die Aktionsliste wird nach jeder Aktionsausführung aktualisiert, somit lassen sich sequentielle Aktionsfolgen mit zusammenhängendem Informationsfluss erreichen. Zusätzlich ist es möglich, Aktionen in der semantischen Beschreibung als statisch zu definieren, um sie dauerhaft in der Liste aufzuführen. In Tabelle 1 ist die Aktion *Hilfe abrufen* beispielsweise eine statische Aktion.

3.2.2 Wertehistorie

Die Aktionsliste enthält jeweils nur die Ausgabewerte der direkt zuvor ausgeführten Aktion (vorhandene Eingabewerte). Oftmals referenziert ein Nutzer sprachlich aber auch auf Diskursentitäten aus dem vorhergehenden Dialog [2]. Hierfür ist es also notwendig, dem Dialogmanager eine Historie aller Ein- und Ausgabewerte zur Verfügung zu stellen. Die Wertehistorie gibt Auskunft über den Wertetyp und den Betrag, sowie einen Zeitstempel, wann der Wert aufgenommen wurde. Ein Dialogmanager kann aufgrund dessen entscheiden, ob ein Wert noch gültig sein soll oder nicht.

Tabelle 2 - Beispiel einer Wertehistorie

Wertetyp	**Zeitstempel**	**Betrag**
Name	2013/01/31 17:05:15	Pizzeria Ulmia
Telefonnummer	2013/01/31 17:05:15	0731-229
Stadt	2013/01/31 17:05:12	Ulm
Thema	2013/01/31 17:03:31	Sprachbedienung

Tabelle 2 zeigt ein Beispiel einer Wertehistorie basierend auf dem Taskmodell aus Abbildung 2. Die Wertehistorie ist nach Zeitstempel sortiert.

3.3 Dialoge mit Taskwechsel

Die Aufgaben, die ein Nutzer an ein SDS stellt, können unter Umständen komplexer Natur sein und mehrere Aktionen involvieren. Offensichtliche Taskwechsel, wie z.B. das Anrufen eines Restaurants, sind bereits in aktuellen Infotainmentsystemen implementiert. Problematischer sind weniger naheliegende Dialoge, wie beispielsweise, zum Interpret des aktuell gespielten Musikstücks neue Schlagzeilen abzurufen. Durch den dynamischen Aufbau des Taskmodells definieren sich selbst komplexe Aktionssequenzen in unserem Taskmanager implizit.

Dialoge mit Taskwechsel definieren sich anhand der Ein- und Ausgabewerte, die semantisch über die Ontologie beschrieben werden. Wenn eine Aktion die Eingabewerte vom selben Typ erwartet wie die Ausgabewerte einer anderen Aktion, ist ein Dialog über diese Anwendungen möglich. Voraussetzung hierfür ist natürlich, dass dies vom Dialogmanager unterstützt wird. Ein fiktiver Dialogmanager könnte die Schnittstelle zum semantischen Taskmanager wie folgt nutzen (die eingerückten Zeilen bedienen die Schnittstelle zum Dialogmanager):

U: „Suche ein italienisches Restaurant in Ulm“
```
        IN: Restaurant suchen, Stadt[Ulm], Küche[italienisch]
        OUT: Name[Ulmia], Telefonnummer[0731-229], Adresse[Platz 1]
```
S: „Ich habe die Pizzeria Ulmia gefunden“
U: „Rufe die Pizzeria an“
```
        IN: anrufen
```
<Telefongespräch>
U: „Navigiere mich dorthin“
```
        IN: navigieren, Adresse[Platz 1]
```

Die jeweiligen Nutzeräußerungen müssen in die richtigen semantischen Werte interpretiert werden, was nicht Teil dieses Beitrags ist. Ebenso muss die Ausgabe des Taskmanagers in eine akustische Ausgabe für den Nutzer überführt werden.

4 Implementierung semantischer Taskmanager

Eine Proof-of-Concept-Implementierung soll zeigen, ob das Konzept, wie beschrieben in Abschnitt 3, in einem SDS umzusetzen ist. Hierbei soll nicht nur die technische Machbarkeit nachgewiesen werden, sondern auch die Systemantwortzeit. In der Mensch-Computer-Interaktion ist die Systemantwortzeit ein entscheidender Faktor bezüglich Produktiviät und Zufriedenheit [11]. Shneiderman definiert eine Antwortzeit unter einer Sekunde als zufriedenstellend. Heutzutage wird allerdings eine sofortige Systemantwort oftmals vorausgesetzt, weshalb wir an den Taskmanager die Anforderung stellen unter 50ms zu antworten. In der gesamten Prozessarchitektur des SDS, benötigen andere Module schließlich auch Zeit für die Verarbeitung.

Die semantische Beschreibung von Anwendungen und der Ontologie sollen einfach zu verarbeiten und zwischen Geräten austauschbar sein. Das **R**esource **D**escription **F**ramework (RDF) bietet die Möglichkeit, Beschreibungen mit Subjekt-Prädikat-Objekt in XML zu definieren. Durch die Erweiterbarkeit von Relationstypen lassen sich unsere Abbildungen in die Ontologie definieren. Zum Interpretieren, Instanziieren und Zusammensetzen von RDF-Dokumenten bietet sich die Java Bibliothek Jena [5] an.

Die Proof-of-Concept-Implementierung beinhaltet den semantischen Taskmanager in Java und eine Simulationsumgebung für einen funktionalen Test, sowie einen Leistungstest. Es wurden Anwendungen, wie eine Hotelanwendung, Suchanfrage und Wettervorhersage simuliert. Das nachfolgende Kapitel gibt Aufschluss über die Ergebnisse, die mit der Implementierung erzielt werden konnten.

5 Evaluation, Diskussion und Ausblick

Auf Basis der Proof-of-Concept-Implementierung soll die Machbarkeit und der Einsatz in einem SDS nachgewiesen werden. Hierfür dient die Simulationsumgebung, die ein dynamisches Nachladen von RDF-Dokumenten erlaubt und ein Überblick über das resultierende Taskmodell liefert. Zusätzlich werden Aktionsliste und Wertehistorie dargestellt. Es konnte gezeigt werden, dass die Umsetzung mit RDF und Jena prinzipiell möglich ist, bezüglich der Aktionsliste allerdings Schwächen zeigt. Durch die implizite Kontextdefinition über Ein- und Ausgabewerte ist es nur möglich, Aktionssequenzen auszuführen, die über diese Werte verbunden sind. Dies kann gelöst werden, indem alle Aktionen in der Aktionsliste aufgeführt werden und eine feinere Abstufung über die Übergangswahrscheinlichkeit vorgenommen wird. Z.B. über ein Modell, das andere Anwendungen oder kontextuelle Werte wie Nutzerpräferenzen berücksichtigt.

Wie bereits erwähnt, sind die Antwortzeiten im Mensch-Maschine-Dialog entscheidend für die Produktivität und Zufriedenheit der Nutzer. Somit ist zu testen, ob das vorgestellte Konzept eine zufriedenstellende Antwortzeit ermöglicht. Hierfür wurde der Taskmanager einem Leistungstest unterzogen. Drei Faktoren haben Einfluss auf die Antwortzeit: Anzahl der Ein-/Ausgabewerte, Aktionsausführung im Backend und die Größe des Taskmodells. Die Anzahl der Ein-/Ausgabewerte hat linearen Einfluss auf die Antwortzeit, da für jeden Wert eine Suche im semantischen Netz stattfindet und muss somit im Leistungstest nicht variiert werden. Die Aktionsausführung im Backend, wie zum Beispiel eine Webservice-Anfrage, ist mit oder ohne Taskmanager notwendig und muss daher nicht im Test berücksichtigt werden. Es muss somit lediglich die Größe des Taskmodells variiert werden, um eine Aussage zu treffen in wie weit sich das Konzept für Infotainmentsysteme mit mehreren Anwendungen einsetzen lassen würde. Hierfür wurden automatisch RDF-Dokumente generiert, die bis zu 25000 Aktionen beinhalten und über Ein-/Ausgabewerte verbunden werden können. Das Ergebnis der mittleren Antwortzeit über 25000 Durchgänge, sowie die Maxima sind Abbildung 3 zu entnehmen (ermittelt mit Intel Core i5, 3.2Ghz, 4GB, 64Bit Win7 Sp1). Im Mittel liegen alle Werte unter 10ms und sind somit für ein SDS tragbar. Sollte eine Antwort im Ausnahmefall bis zu einer halben Sekunde (15000 Aktionen, 448ms) dauern, ist dies ebenso akzeptabel.

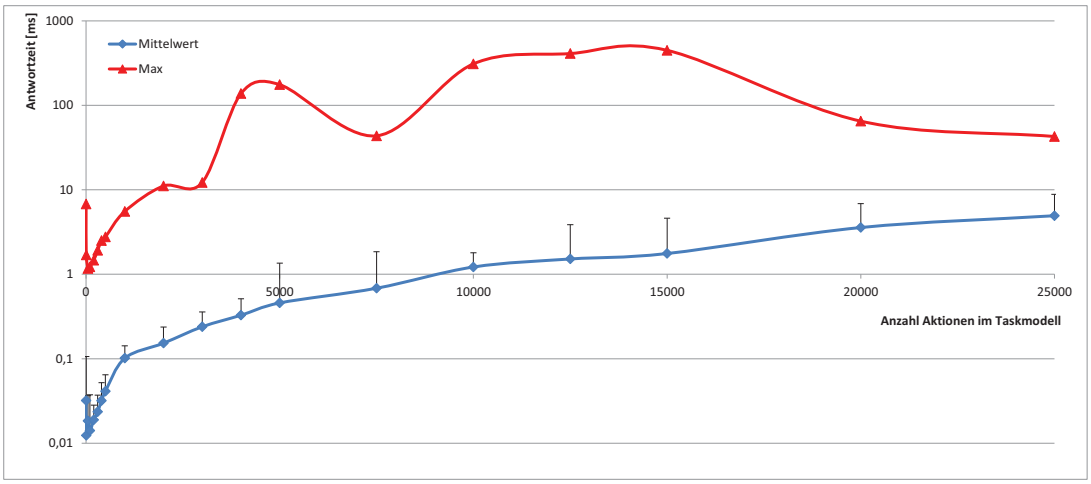

Abbildung 3 - Analyse der Antworzeit in Abhängigkeit von der Anzahl der Aktionen im Taskmodell

Im vorliegenden Beitrag zeigten wir, wie sich ein semantischer Taskmanager für Taskwechsel implementieren lässt. Derzeit erfolgt die Nutzung des semantischen Netzes nur sehr rudimentär, ähnlich einer Klassenhierarchie. In Zukunft müssen noch weitere Beziehungsarten untersucht und nicht nur ein eindeutiges Wertemapping implementiert werden, sondern auch Abbildungen über Inferenz im semantischen Netz. Dies mag einerseits über Abstraktions- und Konkreti-

sierungsrelationen geschehen, andererseits aber auch über semantische Ähnlichkeit (z.B. mit Tverskys Modell [12]). Weiterhin muss untersucht werden, wie das Taskmodell und die Implementierung sich mit großen, realen Ontologien verhält und wie fein granular diese sein müssen. Nach erfolgreicher Integration in ein SDS soll ein Nutzertest zeigen, ob ein Einsatz hinsichtlich Usability-Gesichtspunkten sinnvoll erscheint.

6 Zusammenfassung

Im vorliegenden Beitrag wurde ein semantischer Taskmanager vorgestellt, der Dialoge mit Taskwechsel ermöglicht. Durch die semantische Beschreibung von Anwendungen und einer globalen Ontologie wird ein Informationsaustausch zwischen Anwendungen möglich. Auf Basis von Ein- und Ausgabewerten wird eine Prädiktion der im Kontext sinnvollen Aktionen über eine Aktionsliste, sowie eine Historie an Werten, dem Dialogmanager zur Verfügung gestellt. Die Proof-of-Concept-Umsetzung zeigt, dass eine Antwortzeit von weniger als 10ms im Mittel möglich ist und daher ein Einsatz im SDS denkbar wäre, allerdings weitere Untersuchungen bezüglich der Granularität der Ontologie erforderlich sind. Weitergehend soll untersucht werden, wie Inferenzen im semantischen Taskmodell bei der Werteübergabe genutzt werden könnten, um ein Mapping über Relationen innerhalb der Ontologie zu ermöglichen.

Acknowledgments

The work presented here was funded by GetHomeSafe (EU 7th Framework STREP 288667).

Literatur

[1] CARSTENSEN, K.-U., C. EBERT, C. EBERT, S. JEKAT, R. KLABUNDE und H. LANGER: *Computerlinguistik und Sprachtechnologie*. Spektrum, Akad. Verl., 2010.

[2] FROMKIN, V., R. RODMAN und N. HYAMS: *An Introduction to Language*. Rosenberg, Michael, 2003.

[3] LEE, I., S. KIM, K. KIM, D. LEE, J. CHOI, S. RYU und G. LEE: *A Two-Step Approach for Efficient Domain Selection in Multi-Domain Dialog Systems*. In: MARIANI, J., S. ROSSET, M. GARNIER-RIZET und L. DEVILLERS (Hrsg.): *Natural Interaction with Robots, Knowbots and Smartphones*. Springer NY, 2014.

[4] MACIEJ, J. und M. VOLLRATH: *Comparison of manual vs. speech-based interaction with in-vehicle information systems*. Accident Analysis and Prevention, 41(5):924 – 930, 2009.

[5] MCBRIDE, B.: *Jena: Implementing the RDF Model and Syntax Specification*. In: *SemWeb*, 2001.

[6] NATIONAL HIGHWAY TRAFFIC SAFETY ADMINISTRATION: *Distracted Driving 2011*. Techn. Ber., 2013.

[7] PASSIN, T. B.: *Explorer's Guide to the Semantic Web*. Manning, Greenwich, 2004.

[8] RAO, J., P. KUNGAS und M. MATSKIN: *Logic-based Web services composition: from service description to process model*. In: *Web Services, 2004. Proceedings. IEEE International Conference on*, 2004.

[9] REICHEL, S., A. BERTON, U. EHRLICH und M. WEBER: *Semantische Anwendungsmodellierung und Nutzermodelle: Anforderungen an einen sprachlich zu bedienenden persönlichen Assistenten im dynamischen Systemumfeld*. In: *24. Konferenz zur Elektronischen Sprachsignalverarbeitung*, Bielefeld, 2013.

[10] REITHINGER, N., J. ALEXANDERSSON, T. BECKER, A. BLOCHER, R. ENGEL, M. LÖCKELT, J. MÜLLER, N. PFLEGER, P. POLLER, M. STREIT und V. TSCHERNOMAS: *SmartKom: Adaptive and Flexible Multimodal Access to Multiple Applications*. In: *Multimodal interfaces*, New York, 2003.

[11] SHNEIDERMAN, B.: *Response Time and Display Rate in Human Performance with Computers*. ACM Comput. Surv., 16(3), 1984.

[12] TVERSKY, A.: *Features of similarity*. In: *Psychological Review*, Bd. 84. 1977.

[13] WENG, F., B. YAN, Z. FENG, F. RATIU, M. RAYA, B. LATHROP, A. LIEN, S. VARGES, R. MISHRA, F. LIN, M. PURVER, H. BRATT, Y. MENG, S. PETERS, T. SCHEIDECK, B. RAGHUNATHAN und Z. ZHANG: *CHAT to your destination*. In: *In Proceedings of 8th SIGdial Workshop on Discourse and Dialogue*, 2007.

[14] ZHANG, R., I. B. ARPINAR und B. ALEMAN-MEZA: *Automatic Composition of Semantic Web Services.*. In: *ICWS*, Bd. 3, 2003.