Dynamische Anpassung bei der Generierung natürlicher Sprache

Abschlussarbeit Bachelor of Science

Ole Eichhorn
Matrikelnummer 6144849
Studiengang Mensch-Computer-Interaktion
Fachbereich Informatik
Universität Hamburg
9eichhor@informatik.uni-hamburg.de

28. Oktober 2013

Erstgutachter:
Prof. Dr. Wolfgang Menzel

Zweitgutachter:
Dr. Timo Baumann

Inhaltsverzeichnis

1 Einführung

Computer sind in unserem Alltag allgegenwärtig und durchdringen unser Leben an den unterschiedlichsten Stellen. In fast allen Fällen sollen sie uns dabei unsere Aufgaben überhaupt ermöglichen, erleichtern oder generell das Leben angenehmer gestalten. Auf der anderen Seite sorgt gerade diese Omnipräsenz für eine immer größer werdende Informationsflut, die viele Menschen täglich verarbeiten müssen. Es muss daher stetig weiter an der automatisierten Aufbereitung und Präsentation von Daten geforscht werden, um dieser Herausforderung gerecht zu werden. Wichtig ist dabei, dass die Kommunikation zwischen Mensch und Maschine möglichst einfach ist, also wenig Expertise erfordert und nicht der Mensch die Sprache des Computers zu sprechen braucht, sondern im Gegenteil der Computer dem Menschen im Optimalfall Informationen in natürlicher Sprache darbieten und diese Sprache auch verstehen kann. Der Weg bis zu diesem Optimalfall ist weit und noch keinesfalls bis zu seinem Ende beschritten worden.

In dieser Arbeit soll es darum gehen, ein konkretes, bereits existierendes Softwaresystem, dessen Aufgabe die adäquate Produktion natürlicher Sprache für eine bestimmte Aufgabe ist, zu verbessern. Das System VAVETaM (Lohmann, Kerzel und Habel 2012) ist eine Software, die sehbehinderten Nutzern mit Hilfe gesprochener Sprache bei der Erkundung von taktilen Landkarten assistiert. Die Sprachausgabe von VAVETaM ist in ihrer bestehenden Form in bestimmten Situationen unflexibel und wirkt dadurch unnatürlich. Zu erklären, was diese Unnatürlichkeit verursacht, was die dahinter stehenden Phänomene und die dazugehörigen technischen Abläufe sind und insbesondere wie eine technische Lösung an dem konkreten Beispiel aussehen kann, ist Inhalt dieser Arbeit. Dabei spielt das Prinzip der *inkrementellen Verarbeitung* eine zentrale Rolle, um produzierte Sprache beim Sprechen dynamisch anpassen zu können. Der Fokus liegt dabei auf der *Generierung* und weniger auf der *Synthese* natürlicher Sprache.

Die automatische Produktion gesprochener natürlicher Sprache durch ein Softwaresystem kann grob in die folgenden zwei Schritte unterteilt werden: zuerst wird eine textuelle Repräsentation der gewünschten Ausgabe durch eine Sprachgenerierungskomponente erstellt, die dann im nächsten Schritt durch ein Sprachsynthesemodul in gesprochene Sprache überführt und ausgegeben wird (vgl. Jurafsky und Martin 2009, S. 862; McTear 2004, S. 80). Eine natürlichsprachliche Äußerung wird dabei zunächst vollständig bis zu ihrer Repräsentation als geschriebener Text berechnet. Dieser Text wird dann als eine Eingabeeinheit für das Sprachsynthesemodul verwendet, was zur Folge hat, dass ein solches System ganze Äußerungen ausformulieren muss, bevor mit der verbalen Ausgabe begonnen werden kann und nach dem Beginn der Ausgabe an der gesamten Äußerung keine Änderungen mehr vorgenommen werden können.

Natürlichsprachliche Interfaces ermöglichen eine Kommunikation mit dem Computer, bei der der Mensch die auch ansonsten gewohnten Kommunikationswege nutzen kann und kein Experte für die Kommunikation mit einem Computer sein muss. Je menschlicher das Verhalten eines natürlichsprachlichen Interfaces ist, desto leichter ist es also für den Benutzer zu verwenden. Ein gravierender Unterschied zwischen menschlichen Sprechern und einem System wie oben beschrieben ist jedoch, dass Menschen ihre Äußerungen beim Sprechen an sich ändernde Bedingungen anpassen. Diese Bedingungen können veränderte Umstände in der Umgebung oder auch die Veränderung des Wissensstandes des Sprechenden sein. Menschen können darüber hinaus sogar anfangen, eine Äußerung auszusprechen bevor sie alle Details derselben mental ausformuliert haben.

Ein Softwaresystem wie oben beschrieben hat diese Fähigkeiten nicht, da eine Äußerung in ihrer vollen Ausformulierung vorliegen muss, bevor sie durch die Sprachsynthese weiter verarbeitet werden kann. Dies ist eine technisch bedingte Einschränkung, die mit der Top-Down-Analyse der textuellen Repräsentation einhergeht, die klassische Sprachsynthesemodule verwenden (vgl.

z.B. Taylor 2009, Kapitel 3.3; Liberman und Church 1992; Beutnagel u. a. 1999). Dass diese Einschränkung nicht notwendig ist, zeigten Baumann und Schlangen (2012) im Zusammenhang mit ihrer Software InproTK, indem sie den Prozess der Sprachsynthese inkrementalisierten, also wesentlich kleinschrittiger gestalteten, und somit eine technische Voraussetzung für eine dynamische Anpassung von gesprochener Sprache im Kontext von Softwaresystemen bereitstellten.

Ein schritthaltendes System wie InproTK muss in der Lage sein, einen Prozess anzustoßen und sogar Resultate zu produzieren, noch bevor seine Eingabe vollständig vorliegt. Dies wird erreicht, indem die Eingabe für einen Prozess in kleinere Teile, die sogenannten Inkremente, unterteilt wird. Dies hat hier den Effekt, dass eine Eingabe (wie die oben genannte Äußerung in Textform) nicht vollständig vorliegen muss, damit sie weiter verarbeitet werden kann. Schlangen und Skantze (2009) zitieren Levelt (1993) dahingehend, dass inkrementelle Systeme diejenigen sind, bei denen jede Verarbeitungskomponente durch die minimale Menge des für sie charakteristischen Inputs in Gang gesetzt wird. Diese Eigenschaft birgt allerdings nicht nur die Herausforderung, mit unvollständiger Information als Eingabe zu arbeiten, sondern auch die, das bisherige Resultat aufgrund von neuen Informationen anzupassen oder sogar gänzlich zu verwerfen.

Eine inkrementelle Sprachsynthese ohne eine Sprachgenerierungskomponente, die mit dieser interagieren kann ist im realen Anwendungskontext beschränkt auf Domänen, in denen keine Generierung notwendig ist, beispielsweise weil alle plausiblen Äußerungen des Systems vorkonfiguriert sind. Unter solchen Bedingungen ist der Vorteil einer inkrementellen Sprachsynthese aber selten vonnöten, weil dann auch alle möglichen Äußerungen bereits synthetisiert vorgehalten werden können. Um Inkrementalität in der Produktion natürlicher Sprache sinnvoll zu verwenden sollte eine inkrementelle Sprachsynthese also auch mit einer inkrementellen Generierungskomponente verbunden werden.

In dieser Arbeit wird anhand von Beispielen, die in Kapitel 2 vorgestellt werden, erläutert, welche Herausforderungen die Verbindung von inkrementeller Generierung und Synthese birgt. Nachdem in Kapitel 3 Grundlagen der Produktion natürlicher Sprache zusammengefasst wurden, wird in Kapitel 5 mit Rückbezug auf die Beispiele eine Architektur vorgestellt, die sich diesen Herausforderungen annimmt. Sie verbindet die beiden vorhandenen Softwaresysteme InproTK und VAVETaM, die in Kapitel 4 näher erläutert werden. Im Anschluss daran wird in Kapitel 6 ein Prototyp dieser erweiterten Architektur einem Beispieldurchlauf unterzogen und in Kapitel 7 Vorschläge für zukünftige Arbeit gegeben sowie die Übertragbarkeit der gewonnenen Erkenntnisse auf andere Domänen diskutiert.

2 Problembeschreibung

In diesem Kapitel wird anhand von drei Beispielen die Problematik umrissen, deren Lösung die Aufgabe der in Kapitel 5 vorgestellten inkrementellen Sprachgenerierungskomponente ist. Vorweg wird der Begriff der *taktilen Landkarte* geklärt, da die Beispiele und auch das in Abschnitt 4.1 vorgestellte VAVETaM sowie die darauf aufbauende Architektur, die in Kapitel 5 vorgestellt wird darauf Bezug nehmen.

Eine taktile Landkarte ist ein Ersatz für eine reguläre Landkarte, die blinden und sehbehinderten Menschen die Möglichkeit bietet, eine Umgebung auf ihr durch ihren Tastsinn zu erkunden. Eine übliche Repräsentation einer solchen taktilen Landkarte ist eine gestanzte Papierkarte, die optional mit Braille-Schrift angereichert werden kann. Eine andere Variante ist eine Reliefkarte, beispielsweise aus Metall, die eine Miniatur der abgebildeten Umgebung darstellt und dadurch geometrische Gegebenheiten aller drei Dimensionen darstellt. Der Tastsinn ist im Gegensatz zur

visuellen Wahrnehmung ein lokaler Sinn, weshalb eine taktile Landkarte nur sehr viel aufwändiger erkundet werden kann als eine reguläre visuelle Landkarte. Dieser Problematik kann durch eine multimodale Unterstützung entgegnet werden. Konkret kann dem Erkunder einer taktilen Landkarte zusätzlich verbale Assistenz geboten werden, die ihn über die Gegebenheiten aufklärt, die er wegen der Lokalität des Tastsinnes nicht erfahren kann.

Um mit den nachfolgenden drei Beispielen (verdeutlicht durch Abbildung 1) eine gute Analogie zu VAVETaM herzustellen sei nun angenommen, ein Mensch (im folgenden *Erkunder*) erkunde eine taktile Landkarte, die der gestanzten Papierkarte ähnelt, indem er mit seinem Finger die Kartenobjekte ertastet und ein weiterer Mensch (im folgenden *Assistent*) assistiere ihm dabei, indem er ihn beobachtet und Informationen zu den jeweils erkundeten Kartenobjekten sprachlich präsentiert. Der Erkunder bestimmt dabei die Geschwindigkeit und die Ziele seiner Exploration vollständig selbst. Die Beispiele:

1. Abbrechen: Der Assistent bricht das Aussprechen einer Äußerung ab. Dies geschieht zugunsten einer Anderen, die angesichts von veränderten Umständen wichtigere Informationen enthält. Die folgende Situation demonstriert dies: Der Erkunder erkundet eine Straße auf der Landkarte. Der Assistent sagt als erstes „Jetzt bist du auf der Berliner Straße.“ Direkt im Anschluss daran setzt er an, zu sagen „An ihr befinden sich der Lidl und die Hauptschule.“ Kurz nachdem der Assistent diesen zweiten Satz angefangen hat, beginnt der Erkunder jedoch mit der Erkundung eines der Gebäude am Straßenrand. Der Assistent bemerkt dies, bricht dann seinen Satz an einer geeigneten Stelle ab und gibt möglicherweise einen Laut des Zögerns von sich, um anschließend zeitnah passendere Informationen präsentieren zu können. Der gesamte Diskurs könnte dann in etwa so aussehen: „Jetzt bist du auf der Berliner Straße. An Ihr befinden sich ... äääh ... also das ist jetzt die Hauptschule.“

2. Verändern: Hier erkennt der Assistent, dass eine bereits begonnene Äußerung nicht mehr aktuell ist, diese sich aber an die aktuelle Situation anpassen lässt, ohne sie komplett zu verwerfen. Eine entsprechende Situation: Der Erkunder erkundet wiederum eine Straße auf der Landkarte. Der Assistent setzt an, zu sagen „Jetzt bist du mit dem Finger auf der Kennedy-Allee.“ Der Erkunder beginnt allerdings kurz nach dem Anfang der Äußerung, sich für eine Seitenstraße zu interessieren und ‚biegt in diese ein‘. Der Assistent bemerkt dies, zögert unter Umständen kurz und passt seinen Satz an die neue Situation an. Er sagt also insgesamt „Jetzt bist du mit dem Finger auf ... ähm ... auf der Clintonstraße.“

3. Vervollständigen: Es gibt darüber hinaus wie oben bereits angedeutet Fälle, in denen vorstellbar ist, dass der Assistent einen Satz bereits beginnt, noch bevor er alle Details zu seiner Formulierung kennt. Eine Beispielsituation hierfür: Der Erkunder beginnt eine Straße zu erkunden, dessen Name der Assistent nicht aus dem Kopf kennt. Er beginnt dennoch den Satz „Das ist ...“ , während er den Namen aus der Karte heraussucht bzw. abliest. Sobald er ihn gefunden hat beendet er den Satz: „... die Thomas-Mann-Straße.“ Zwischen den beiden Satzteilen entsteht dabei eventuell eine kurze Verzögerung in der Aussprache.

Das letzte Beispiel ist zwar prinzipiell interessant für inkrementelle Sprachgenerierung, steht aber nicht im Fokus dieser Arbeit, weil es sich in einem bedeutenden Punkt von den vorhergegangenen Beispielen unterscheidet: Während in den ersten beiden Beispielen die Anpassung einer Äußerung durch Umstände der Umwelt bedingt ist, ist sie im letzten Beispiel ein Resultat aus der Unwissenheit oder fehlenden Verarbeitungsgeschwindigkeit des Assistenten. VAVETaM kann jedoch in dieser Hinsicht als allwissend angesehen werden weil der Zeitaufwand zur Datenabfrage (wie beispielsweise das Nachschlagen eines Namens) vernachlässigbar ist. Situationen dieser Art treten also in dem hier behandelten Kontext nicht auf.

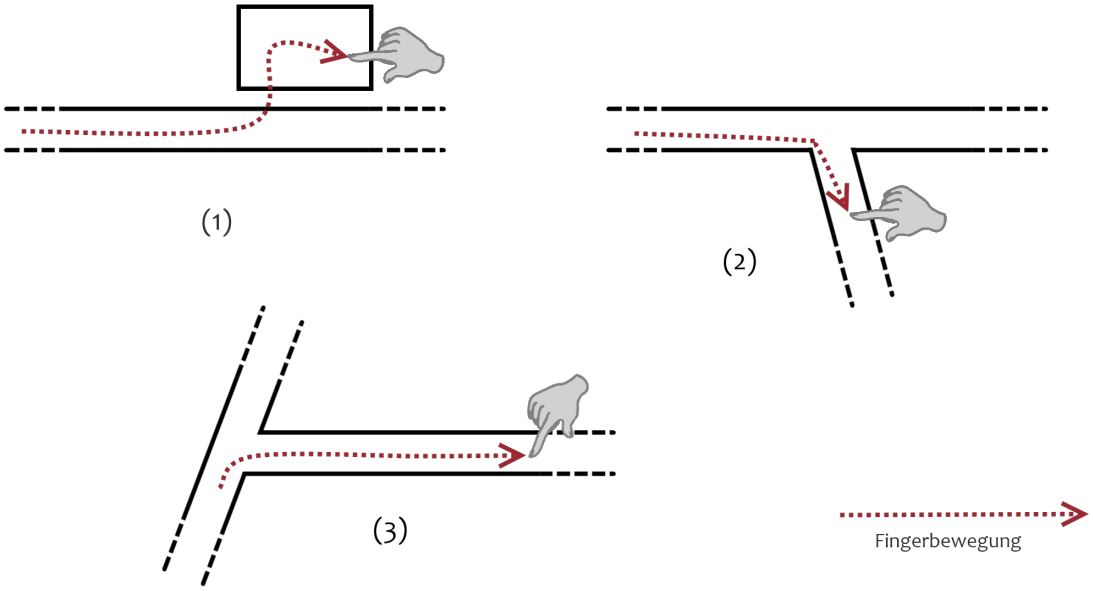


Abbildung 1: Veranschaulichung der Beispiele

Der weitere Inhalt dieser Arbeit wird auf die ersten beiden Beispiele Bezug nehmen, nachdem im folgenden Kapitel 3 Grundlagen und Begrifflichkeiten der Produktion natürlicher Sprache geklärt werden, die ebenfalls eine Rolle in der restlichen Arbeit spielen werden. Darüber hinaus wird im nächsten Kapitel ein Einblick in die vergangene und die aktuelle Forschung zu diesem Thema geboten.

3 Grundlagen der Produktion natürlicher Sprache

Dieses Kapitel legt eine Grundlage für das Verständnis und die Begrifflichkeiten in der Disziplin der Produktion natürlicher Sprache. Zunächst ordnet Abschnitt 3.1 die Produktion natürlicher Sprache in die Forschungslandschaft der natürlichsprachlichen Systeme ein und nennt die spezifischen Herausforderungen dieses Teilgebietes. Im darauf folgenden Abschnitt 3.2 werden Möglichkeiten zur Aufteilung der automatisierten Produktion natürlicher Sprache in kleinere Bestandteile aufgezeigt, die es ermöglichen die Komplexität der menschlichen Sprache zu systematisieren und natürliche Sprache überhaupt automatisiert zu produzieren. Anschließend werden im Abschnitt 3.3 aktuelle Forschungsansätze vorgestellt. Dabei liegt ein Schwerpunkt auf den sogenannten *repairs*, also Reparaturen von sprachlichen Äußerungen, denn diese sind ein Kernelement bei der dynamischen Anpassung gesprochener Sprache. Wie die aus diesem Kapitel gewonnenen Erkenntnisse für die Verbesserung des in Abschnitt 4.1 vorgestellten Systems VAVETaM hilfreich sein können zeigt sich in Kapitel 5.

3.1 Einordnung

Im Forschungsfeld der natürlichsprachlichen Systeme ist die Produktion natürlicher Sprache eines von zwei großen Teilgebieten, das andere ist die Erkennung natürlicher Sprache. Von außen betrachtet und auch nach klassischen Kommunikationstheorien (vgl. z.B. Shannon und Weaver 1949) scheinen diese beiden Teilgebiete in einem symmetrischen Verhältnis zu stehen, denn sie teilen sich jeweils auf in den eher technischen Aspekt des Überführens von Text in Sprache oder anders herum auf der einen Seite und den eher psycholinguistischen Aspekt der Semantik,

Pragmatik und Syntax, also im Falle der Produktion die Frage „Was soll wie gesagt werden?“ und im Falle der Erkennung die Frage „Was ist mit dem Gesagten gemeint?“ auf der anderen Seite. Im technischen Kontext der Computerlinguistik werfen die beiden Teilbereiche allerdings unterschiedliche Probleme auf.

Ein bestimmter Sachverhalt kann durch unzählige unterschiedliche Äußerungen dargestellt werden. „Die Tasse ist blau“ transportiert (nahezu) die gleiche Information wie „Die Tasse hat eine blaue Farbe“ oder „Blau ist die Tasse“. Die automatische Erkennung natürlicher Sprache steht also unter anderem vor der Herausforderung, dass nicht unbedingt das Gleiche gesagt wird, wenn das Gleiche gemeint ist und im Übrigen auch anders herum. „Das ist die Tür“ kann eine neutrale Identifikation eines Objektes sein oder auch eine verärgerte Aufforderung, den Raum zu verlassen.

Bei der Produktion natürlicher Sprache stellt sich unter anderem die sehr komplexe Frage, wie ein Sachverhalt am besten ausgedrückt werden kann. Diese Frage ist nahezu gar nicht, höchstens aber für sehr eingeschränkte Domänen beantwortbar, weil selbst die kleinste Änderung in einer Äußerung ihre Implikationen verändern kann. Zieht man die Kommunikationstheorie des *Vier-Seiten-Modells* von Schulz von Thun (2006) heran, so wird außerdem schnell klar, dass jegliche Antwort auf die oben gestellte Frage auch abhängig vom Adressaten einer Aussage, also dem Zuhörer ist.

In der zwischenmenschlichen Kommunikation hat ein Sprecher ein Modell seines Zuhörers im Sinn. Er hat möglicherweise eine Vorstellung, was sein Gegenüber weiß oder zumindest wahrnehmen kann und kann darauf referenzieren. Es gibt Konventionen über den Sprachgebrauch[1], die nötig sind um eine funktionierende Kommunikation überhaupt aufzubauen, oder anders gesagt, die bei Nicht-Einhaltung zu ernsten Kommunikationsschwierigkeiten führen.

Natürlich können hier nicht all diese Aspekte modelliert werden aber es wäre dennoch falsch zu glauben, sie seien deshalb nicht vorhanden. Sie werden implizit mitschwingen, denn „man kann nicht nicht Kommunizieren“, wie Watzlawick, Beavin und Jackson (1969, S. 53) es ausdrücken. Die Tatsache, dass hier ein Computer mit einem Menschen kommunizieren soll ist beispielsweise ein Beziehungsaspekt nach Schulz von Thun.

3.2 Bestandteile der Produktion natürlicher Sprache

Die Produktion natürlicher Sprache (im Weiteren abgekürzt durch NLP für *natrual language production*) lässt sich grob unterteilen in die Bereiche der Sprachgenerierung (im Weiteren abgekürzt durch NLG für *natrual language generation*) und der Sprachsynthese (im Weiteren abgekürzt durch TTS für *text-to-speech*). In dieser Arbeit wird ein besonderes Augenmerk auf die Generierung und ihre Verbindung zur Synthese gelegt.

Die NLG muss sich mit den im vorigen Abschnitt gestellten Fragen „Was soll gesagt werden?“ und „Wie soll es gesagt werden?“ auseinandersetzen. Reiter und Dale (2000) schlagen zur Beantwortung dieser Fragen die informelle Unterteilung des Generierungsprozesses in Dokumentplanung, Mikroplanung und Realisierung vor. Jedes dieser Module ist wieder in weitere Unteraufgaben unterteilt (für eine detailliertere Beschreibung siehe Reiter und Dale 2000, Abschnitt 3.3, S.47-59).

Unter Dokumentplanung verstehen Reiter und Dale die Bestimmung des Inhaltes und dessen grobe Strukturierung in Abschnitte und Absätze, sowie die sinnvolle rhetorische Verbindung

[1] Ein Beispiel für einen bewussten Bruch einer solchen Konvention als sprachliches Mittel ist die Verwendung von Ironie.

derselben. Die Mikroplanung beinhaltet die Generierung von referenzierenden Ausdrücken, die Lexikalisierung, also die Auswahl der passenden Wörter und Ausdrücke um den Inhalt angemessen darzustellen, und die Aggregation. Die Realisierung schließlich überführt die zuvor generierten Strukturen in eine sogenannte *Oberflächenform*, also eine tatsächliche textuelle Repräsentation.

NLG kann bis heute stets nur auf sehr eingeschränkten Domänen angewendet werden, da die Komplexität schon in einfachen Kontexten schnell unübersichtlich wird. Es ist auf der anderen Seite in manchen Systemen sogar ausreichend, vorgefertigte Satzstücke (sog. Templates) richtig zu kombinieren. Damit werden die Probleme, denen sich Reiter und Dale (2000) mit ihren Überlegungen stellen weitgehend umgangen.

Ein Beispiel für ein einfaches Template ist der Satz „Jetzt bist du auf X.“, in dem X durch ein Substantiv substituiert wird, das eine Straße oder einen Weg bezeichnet und einen Artikel im Dativ vorangestellt hat. Kennt man den Bezeichner und sein Geschlecht, so lässt sich der Satz in eine vollständige Oberflächenform überführen (etwa „Jetzt bist du auf der Bundesstraße“ oder „Jetzt bist du auf dem Amselweg“).

Die in Kapitel 5 vorgestellte Architektur folgt einem Template-Ansatz, der allerdings deutlich mehr Möglichkeiten bietet, als nur das Substituieren von Bezeichnern. Sie muss damit in der Lage sein, Äußerungen zu formulieren, die bei der Exploration einer taktilen Landkarte als nützliche Assistenz dienen. Wichtig ist deshalb nicht nur die Angemessenheit der produzierten Äußerungen, sondern angesichts schneller Explorationshandlungen ganz besonders das zeitliche Verhalten, was eine Kernmotivation für die Anwendung inkrementeller Sprachgenerierung in diesem Kontext ist.

3.3 Aktueller Stand der Forschung

In diesem Abschnitt wird die hier vorgestellte Arbeit mit der Forschung zur automatisierten Sprachproduktion der letzten Jahre in Verhältnis gesetzt. Es wird erklärt, welche der Forschungsergebnisse relevant für die vorgestellte Architektur sind, warum sie dies sind und warum andere Forschung eine nachrangige Rolle für das konkrete Beispiel spielt.

3.3.1 Reparaturen von sprachlichen Äußerungen

Ein zentraler Aspekt bei der dynamischen Anpassung natürlicher Sprache sind sogenannte *repairs*. Gemeint sind damit Korrekturen einer im Gange befindlichen sprachlichen Äußerung. Sie finden häufige Anwendung im täglichen zwischenmenschlichen Sprachgebrauch und erfahren dabei trotzdem wenig Beachtung. Zur Erläuterung stelle man sich hierbei vor, der Assistent ginge in Beispiel Zwei der Problembeschreibung auf die veränderte Situation ein, indem er seinen Satz ohne Zögern oder Wiederholung von Worten an die neue Sachlage anpasst und sagt „Jetzt bist du mit dem Finger auf der Clintonstraße.“ Da der Satz begonnen hat, als der Erkunder noch eine andere Straße erkundet hat könnte dieser sich nun unsicher darüber sein, auf welche der erkundeten Straßen sich die Assistenz bezieht. Erst das Zögern und die Wiederholung eines Teiles des Satzes versetzen den Erkunder in die Lage nachzuvollziehen, dass der Assistent die veränderte Situation in seiner Äußerung berücksichtigt.

Die im Beispiel dargestellte Konstellation aus Zögern und der anschließenden Wiederholung des Wortes „auf“, eine Form von Reparatur, ist also wesentlich mehr, als nur die Überbrückung einer Neuformulierung der Assistenz. Clark (2002) argumentiert, dass Diskontinuitäten im Redefluss keine Probleme beim Sprechen, sondern im Gegenteil deren Lösung darstellen. Sie signalisieren dem Zuhörer, dass eine Änderung in der Äußerung stattgefunden hat und markieren durch

gezielte sprachliche Mittel sogar sehr präzise, welcher Teil der Äußerung angepasst wurde. Diese Form der Reparaturen nennt sich *overt repair*, und schlägt sich im Gegensatz zur *hidden repair* in der gesprochenen Äußerung für den Zuhörer bemerkbar nieder. Eine *hidden* oder auch *covert repair* also eine verdeckte Reparatur ist laut Levelt (1983, S. 44) eine, bei der keine Morpheme geändert hinzugefügt oder entfernt werden. Im Weiteren wird es ausschließlich um offene Reparaturen gehen, da sie für die Sprachproduktion eine wesentlich größere Herausforderung sind, als verdeckte Reparaturen.

Levelt (1983) identifiziert drei Phasen bei der offenen Reparatur sprachlicher Äußerungen:

1. die Unterbrechung des Redeflusses,
2. das Signalisieren bzw. Ankündigen einer Änderung durch ein Zögern oder ein sogenannten *ending-term* wie etwa „äähm“ oder „also“ und schließlich
3. die Reparatur selbst.

Eine Reparatur wird auf eine Art und Weise eingeleitet, die den Zuhörer in die Lage versetzt, Die Korrektur nachzuvollziehen (eine sehr gute Graphik zur Erklärung der verschiedenen Konstituenten einer Reparatur findet sich in Levelt 1983, S. 45). In Kapitel 5 wird das Konzept der Reparaturen angewandt um dynamische Anpassungen von bereits begonnenen Äußerungen zu vollführen. Auf diese Weise kann der Zuhörer nachvollziehen, inwiefern das dort präsentierte Softwaresystem auf seine Aktionen eingeht.

3.3.2 Baumadjunktions-Grammatiken

Durchläuft man die verschiedenen von Reiter und Dale (2000) vorgeschlagenen Schritte der Generierung natürlicher Sprache, so ist eines der Probleme die Lexikalisierung, also die Determinierung der verwendeten Wörter und syntaktischen Konstrukte. Bei einer aufwendigen Generierung bietet es sich an, eine *Baumadjunktions-Grammatik (TAG für tree adjoining grammar)* zu verwenden. Sie operiert auf Syntaxbäumen und baut durch Ersetzung und Anhängen von Teilbäumen eine Repräsentation auf, die sich anschließend in eine lexikalische Oberflächenform überführen lässt.

Zuerst entworfen von Joshi, Levy und Takahashi (1975), ist die Gattung der Baumadjunktionsgrammatiken in der Chomsky-Hierarchie zwischen kontextfreier und kontextsensitiver Grammatik einzuordnen. Sie ist somit mächtiger als eine kontextfreie und leichter zu parsen als eine kontextsensitive Grammatik und wurde deshalb unter anderem von Kilger und Finkler (1995) und Herzog u. a. (1996) für die Generierung natürlicher Sprache verwendet. Joshi und Schabes (1991) geben eine genaue Beschreibung von TAG. Die hier vorgestellte erweiterte Architektur verwendet allerdings keine TAG, da fertige Templates eine Lexikalisierung überflüssig machen. Dies ist sicherlich nur deshalb sinnvoll, weil die Domäne recht überschaubar ist und dadurch Verarbeitungsschritte vermieden werden können. Zu einem späteren Zeitpunkt ließe sich die Verwendung einer TAG in die Architektur integrieren, sofern eine solche Lexikalisierung auch als einer der inkrementellen Verarbeitungsschritte realisiert werden kann.

3.3.3 Inkrementalität

Es gibt in der Forschung viele Ansätze, die Produktion natürlicher Sprache mit den Prinzipien der inkrementellen Verarbeitung zu verbessern. Dohsaka und Shimazu (1996) stellen ein System vor, das schon während des Lösens eines spezifischen Problems anfängt, sprachliche Äußerungen zu generieren, um ein adäquates zeitliches Verhalten zu gewährleisten. Sie beschreiben dabei, wie sie die inkrementelle Planung von Äußerungen mit Hilfe von Kommunikationszielen, deren

Oberflächenrealisierung und wiederholter Neuplanung umsetzen. Der Nutzen der inkrementellen Verarbeitung beschränkt sich bei ihnen auf die Produktion von Zögerungslauten, um unnatürliche wirkende Pausen zu vermeiden.

Kilger und Finkler (1995) präsentieren mit einem System namens VM-GEN einen weiteren Ansatz, natürlichsprachliche Äußerungen inkrementell zu generieren. Besonderes Augenmerk legen sie dabei auf ihre Beschreibung des Verarbeitungsprozesses von der schrittweise Verarbeitung von Input bis zur gleichsam schrittweise Produktion von Wörtern und Satzteilen und stellen dabei elf Designprinzipien für die Konzeption von inkrementellen Sprachgenerierungsmodulen auf. Sie kommen dabei aber auch zu dem Schluss, dass inkrementelle Verarbeitung scheinbar nicht optimal für die Produktion natürlicher Sprache verwendet werden kann (vgl. Kilger und Finkler 1995, S. 9).

VITRA ist ein System, das von Herzog u. a. (1996) entworfen wurde und für die automatische verbale Beschreibung visueller Perzeption konzipiert ist. Das Prinzip der inkrementellen Verarbeitung zieht sich auch bei ihnen durch den kompletten betrachteten Verarbeitungsprozess von der inkrementellen Szenefolgenanalyse bis zur inkrementellen Verbalisierung. Beispielhaft wird die automatische Beschreibung von Fußballspielszenen veranschaulicht und damit ein exzellent geeignetes Einsatzfeld für die Verwendung von inkrementeller Sprachproduktion aufgezeigt, nämlich das Kommentieren von simultan zur Sprache stattfindenden Ereignissen. Guhe, Habel und Tappe (2000) erläutern ebenfalls eine Lösung zu einem Problem aus der Domäne der Beschreibung von beobachteten Vorgängen, indem sie ein inkrementelles Verarbeitungsmodell zur Erkennung und Kommentierung von Ereignissen auf einem Flugfeld vorstellen. Ihr Schwerpunkt liegt dabei allerdings auf der inkrementellen Konzeptualiserung der Ereignisse.

All diesen Forschungsarbeiten ist gemein, dass sie sich mehr oder weniger mit den Problemen der Inkrementalität *innerhalb* der Disziplin der Generierung natürlicher Sprache auseinander setzen. Obwohl z.B. im Falle von VM-GEN eine solche Anbindung durchaus vorstellbar ist, wird die Interaktion mit einer inkrementellen Sprachsynthese, also die durchgängige Inkrementalisierung des gesamten Produktionsprozesses dabei vernachlässigt. Das bedeutet, dass zwar auf andauernden Input schon vor dessen Beendigung eine Verarbeitung und generierte Sprache als Output folgen kann, aber angefangene Äußerungen der Synthese nicht revidiert werden können. Zu diesem Schluss kommen auch Skantze und Hjalmarsson (2013, vgl. S.246), die ein Dialogsystem vorstellen, das vollständig von der Verarbeitung gesprochenen Sprache bis zur Produktion gesprochener Sprache auf inkrementeller Verarbeitung basiert. In einer spielerischen Anwendung namens DEAL können Benutzer mit dem System über den Preis von zum Kauf angebotenen virtuellen Waren verhandeln. Das System ist dabei wie auch die in Kapitel 5 beschriebene Architektur in der Lage, Reparaturen an den im Gange befindlichen sprachlichen Äußerungen zu vollziehen, indem es auf neuen Input durch eine vollständige Neuplanung reagiert, die neuen Resultate mit den Vorherigen vergleicht und die dabei geänderten Ausgabeinkremente ersetzt.

Buschmeier u. a. (2012) beschäftigen sich mit der Verbindung von inkrementeller Generierung und Synthese im Umfeld der Dialogsysteme. Wie auch die in Kapitel 5 vorgestellte Architektur verwenden sie InproTK für die inkrementelle Sprachsynthese. Für die inkrementelle Generierung sprachlicher Äußerungen greifen sie auf SPUD (siehe Stone u. a. 2003) zurück, ein Framework für die Mikroplanung mit Hilfe von *Kommunikationsintentionen*. SPUD ist zwar ursprünglich nicht für eine inkrementelle Verarbeitung konzipiert, einzelne für die Mikroplanung wichtige Funktionen können jedoch auch kleinschrittiger als vorgesehen verwendet werden. Dies machen sich Buschmeier und Kollegen zunutze, wobei bedingt durch die Arbeitsweise von SPUD nur eine Teil-Inkrementalisierung der Generierung umgesetzt werden kann. Sie erläutern darüber hinaus Adaptionsprinzipien, die eine Anbindung ihrer Generierungskomponente an InproTK ermöglichen.

Dies ist deshalb interessant, weil auch die erweiterte VAVETaM-Architektur eine Möglichkeit zur Adaption, also zur Anpassung der inkrementellen Eingabe in die Synthese anhand von neu Generierten Äußerungen umsetzt.

Ein weiterer interessanter Aspekt bei der inkrementellen Dialogverarbeitung ist die Frage, wann ein inkrementelles Modul optimalerweise seinen Verarbeitungsprozess anstoßen sollte, um einerseits eine möglichst kurze Verzögerung durch die Verarbeitung zu gewährleisten und andererseits zu häufige und aufwendige Neuberechnung zu vermeiden. Ein Modul kann mit nur einem Inkrement bereits eine Verarbeitung anstoßen, muss aber unter Umständen bereits beim nächsten eintreffenden Inkrement seine Berechnungen revidieren. Dethlefs u. a. (2012) wenden daher statistische Methoden des Maschinenlernens auf die Domäne der inkrementellen Dialogverarbeitung an, um die Anzahl der Neuplanungsprozesse und damit auch die Notwendigkeit der Anwendung von Reparaturen zu minimieren. Außerdem leidet die Qualität der berechneten Prosodie, also derjenigen lautlichen Eigenschaften der Sprache, die nicht direkt aus den einzelnen Phonemen erhoben werden kann, wenn aufgrund von zu kleinen Inkrementen, also zu wenig Eingabe zu wenig Kontext für die Berechnung zur Verfügung steht (für eine genauere Beschreibung dieses Phänomens siehe Baumann und Schlangen 2012, Abschnitt 4).

Einige der hier vorgestellten Forschungsthemen haben Erkenntnisse hervorgebracht, die von der erweiterten VAVETaM-Architektur in Kapitel 5 aufgegriffen werden. Doch zunächst werden im Kapitel 4 die Ausgangssysteme vorgestellt, auf denen die erweiterte Architektur basiert.

4 Beschreibung der Ausgangssysteme

Für die genauere Einordnung und ein Verständnis dafür, was die angestrebte inkrementelle Sprachgenerierungsarchitektur leisten muss folgt nun zuerst jeweils eine Beschreibung der beteiligten Systeme VAVETaM im Abschnitt 4.1 und InproTK im Abschnitt 4.2 bevor die darauf aufbauende erweiterte Architektur im Kapitel 5 diskutiert wird.

4.1 VAVETaM

Nachdem im folgenden Abschnitt 4.1.1 allgemeine Informationen zu VAVETaM gegeben werden, wird im Abschnitt 4.1.2 genauer auf die verwendeten Datentypen eingegangen. Wie diese bei Verarbeitungsabläufen zusammenspielen wird in Abschnitt 4.1.3 erläutert, bevor im Abschnitt 4.1.4 Einschränkungen und Probleme der bisherigen Konzeption von VAVETaM diskutiert werden.

4.1.1 Allgemeines

Die Abkürzung *VAVETaM* steht für *Verbally Assisted Virtual Environment Tactile Map*, was bereits eine erste Idee der Aufgabe dieses Systems geben kann. *Tactile Maps*, sind dabei allerdings nicht taktile Landkarten wie in der Problembeschreibung erwähnt. VAVETaM wählt zugunsten von Faktoren wie Flexibilität, Vervielfältigbarkeit und Transportierbarkeit einen anderen Ansatz. Grundlage hierfür ist ein haptisches Interface, ein dreidimensionales Zeigegerät mit *force feedback*, das die Möglichkeit bietet, virtuelle Objekte zu ertasten (für eine genauere Beschreibung siehe Lohmann, Kerzel und Habel 2012). Mit Hilfe eines solchen Gerätes lässt sich eine virtuelle dreidimensionale Landkarte ertasten, die als eine horizontale Fläche mit Vertiefungen repräsentiert wird. Die Vertiefungen stellen Kartenobjekte wie Straßen und Gebäude dar. Auf diese Weise kann ihnen mit dem haptischen Zeigegerät einfach gefolgt werden. Als Analogie zu der hierbei stattfindenden Interaktion kann man sich vorstellen, mit einem Stift Vertiefungen in einer

physischen Oberfläche nachzufahren.

Ein erwähnenswerter Unterschied zwischen diesem Konzept und der in der Problembeschreibung genannten taktilen Landkarte ist der, dass durch die Gegebenheiten des haptischen Interfaces eine Interaktion lediglich in einem Punkt stattfinden kann. Auf einer herkömmlichen taktilen Landkarte könnte auch mit mehreren Fingern oder der ganzen Handfläche simultan erkundet werden. Aus Gründen der besseren Vergleichbarkeit wurde deswegen in den Beispielen davon ausgegangen, dass der Erkunder die Karte mit einem einzelnen Finger erkundet.

Die eigentliche Leistung von VAVETaM liegt darin, zu den Explorationshandlungen des Erkunders in Echtzeit eine passende verbale Assistenz zu generieren. Es informiert dabei den Nutzer nicht nur darüber, welches Kartenobjekt er gerade erkundet, sondern auch wie es zu anderen Kartenobjekten und der gesamten Karte in Verhältnis steht, sofern VAVETaM genug Zeit für solche Ausführungen hat. Die Richtung und Geschwindigkeit der Exploration wird dabei allein vom Nutzer vorgegeben. Konkret wird also ein Assistent wie im Kapitel 2 imitiert.

4.1.2 Die verwendeten Datentypen und Wissensrepräsentation

Die Benutzerinteraktionen auf der virtuellen Karte werden von einer nicht weiter spezifizierten Einheit aufbereitet und zusammengefasst zu sogenannten *Map Exploratory Procedures* (`MEP`s). Sie repräsentieren Bewegungsabläufe, denen eine bestimmte Semantik zugeordnet werden kann. Ein Beispiel hierfür ist eine `TrackMEP`, die signalisiert, dass ein *Track*, also eine Straße oder etwas vergleichbares im Explorationsfokus des Erkundenden liegt.

MEPs haben eine Kategorie wie beispielsweise die oben erwähnte `TrackMEP` und können darüber hinaus mit Argumenten versehen werden. Sei `pt1` ein interner Bezeichner für die Kennedy-Allee, so signalisiert `TrackMEP(pt1)`, dass der Benutzer seinen Explorationsfokus auf die Kennedy-Allee verschoben hat. `TrackMEP(pt1)` wird so interpretiert, dass nun relevante Informationen zur Kennedy-Allee sprachlich präsentiert werden sollten. Ein sogenannter *Diskursplan* gibt dazu vor, welche Informationen präsentiert werden könnten.

Ein *Diskursplan* ist eine Auflistung von *Intentionen*, die ausdrücken, welche Art von Informationen in welcher Reihenfolge dargeboten werden sollen. Jede MEP-Kategorie ist mit einem Diskursplan assoziiert. Ein solcher Diskursplan beinhaltet zum Beispiel im Falle einer `TrackMEP` die folgenden Intentionen:

1. Identifikation des erkundeten Tracks.
2. Beschreibung der räumlichen Relationen zu anderen Kartenobjekten in der Nähe.
3. Beschreibung der Enden des Tracks.
4. Beschreibung von Kreuzungen, die der Track mit anderen hat.

Das Wissen über alle Kartenobjekte wird in einem Netz aus *referenziellen Objekten* repräsentiert, das auf dem Formalismus für referenzielle Netzwerke von Habel (1982) basiert. Jedes dieser Objekte steht für eine Karten-Entität und kann eine beliebige Anzahl an Prädikaten zugeordnet haben, die seine Relationen zu anderen referenziellen Objekten beschreiben. Hat also die Kennedy-Alle eine Kreuzung mit der Clintonstraße und sei diese Kreuzung ein eigenes referenzielles Objekt mit dem Bezeichner `ptco3` so ist dem referenziellen Objekt `pt1`, das die Kennedy-Allee repräsentiert ein Prädikat der Form `is_in_track_config(pt1, ptco3)` zugeordnet.

Existieren Informationen (in Form von passenden Prädikaten) für das fragliche referenzielle Objekt, die die jeweilige Intention erfüllen, so werden diese Informationen aggregiert und in *Präverbalen Äußerungen* (im Weiteren abgekürzt durch PVM für **P**re**v**erbal **M**essage) zusammengefasst. Aus jeder PVM kann dann eine natürlichsprachliche Äußerung konstruiert werden.

4.1.3 Verarbeitungsabläufe

Wird beispielsweise `trackMEP(pt1)` erkannt und der Diskursplan gibt vor, dass im Falle einer Straße Informationen über ihre Kreuzungen mit anderen Straßen präsentiert werden sollen, so werden diese Informationen aus dem referenziellen Netz bezogen, in PVMs verpackt und schließlich zu Aussagen formuliert. Im Beispiel aus Abschnitt 4.1.2 könnte daraus die Formulierung des Satzes „Die Kennedy-Allee hat eine Kreuzung mit der Clintonstraße“ resultieren. Ein solcher Satz wird nun lediglich noch von einer herkömmlichen TTS-Komponente in gesprochene Sprache umgewandelt und ausgegeben. Abbildung 2 veranschaulicht die Zusammenarbeit der verschiedenen Komponenten bei einem solchen Ablauf.

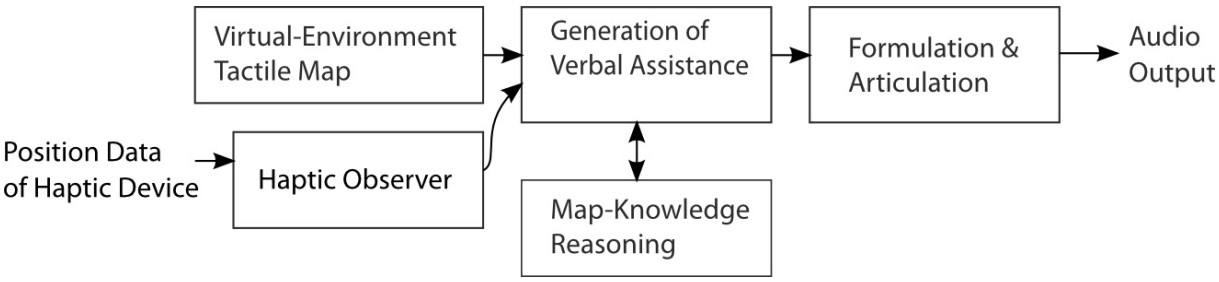


Abbildung 2: die Ursprüngliche Architektur von VAVETaM

Für die Formulierung wird ein Template-Ansatz verwendet, der einige Varitationen zulässt. Zum Beispiel

- muss nicht für jede Kreuzung eine eigener Äußerung ausformuliert werden, sondern eine Aggregation kann eine Äußerung wie etwa „Die Kennedy-Allee hat Kreuzungen mit der Clintonstraße, dem Roosevelt-Boulevard und dem Lincolnweg“ als Variation eines Templates zusammenstellen.
- gibt es Ansätze, eintönige Wiederholungen zu reduzieren. Kartenobjekte können durch eine der folgenden Äußerungen Identifiziert werden:
  - „Das ist X.“
  - „Dies ist X.“
  - „Hier ist X.“
  - „Jetzt bist du auf X.“

  wobei X jeweils durch einen Bezeichner ersetzt wird.

Außerdem verwendet VAVETaM ein minimales Zuhörermodell. Dieses besteht aus einem Diskursverlauf und erlaubt daher, unnötige Wiederholungen zu vermeiden. Äußerungen, die bereits in der jüngeren Vergangenheit gesagt wurden, werden dabei in ihrer Priorität niedriger eingestuft, als der Diskursplan es vorgibt und daher nachrangig Produziert.

4.1.4 Einschränkungen und Probleme

Die Architektur beinhaltet zwar die Verwendung eines haptischen 3D-Zeigegerätes, dessen Positionsdaten zu MEPs aufbereitet werden, jedoch wird bei dem von Lohmann, Eichhorn und

Baumann (2012) vorgestellten Prototypen vom Input durch ein echtes 3D-Zeigegerät abstrahiert, da der Fokus der Entwicklung zunächst auf der Kernkomponente lag. Der Input für den Prototypen wird stattdessen anhand von vorgefertigten Explorationsverlaufsplänen simuliert. Diese Pläne basieren auf detaillierten manuellen Transkriptionen von Video-Mitschnitten von Explorationen, die im Rahmen der Vorstudien zum VAVETaM-Projekt stattgefunden haben. Für die Transkription wurden Explorationen ausgewählt, die möglichst vielfältige und charakteristische Explorationssituationen beinhalten.

Abbildung 3 gibt einen genaueren Einblick in die Kernkomponente, die *Generation of Verbal Assistance* (GVA) der ursprünglichen Architektur. Die dort abgebildeten Verbindungen sind zwar teilweise bidirektional, aber die Verarbeitung erfolgt in diesem Aufbau stets in eine Richtung. Zeitliches Verhalten wird lediglich durch die Agenda als Warteschlange gewährleistet, sodass die Formulierung einer neuen Äußerung erst stattfindet, sobald die TTS-Komponente ihre aktuelle Äußerung beendet hat. Die einzige Ausnahme hierzu besteht für den GVA Controller in der Möglichkeit, mit Hilfe von *Agenda Operationen* noch nicht verarbeitete Äußerungen von der Agenda zurück zu ziehen, wenn diese angesichts neuen Inputs nicht mehr aktuell sind. Besonders wichtig ist hier die Anmerkung, dass eine Äußerung, hat sie die Agenda einmal in Richtung Formulierungskomponente verlassen, nicht mehr aufgehalten werden kann.

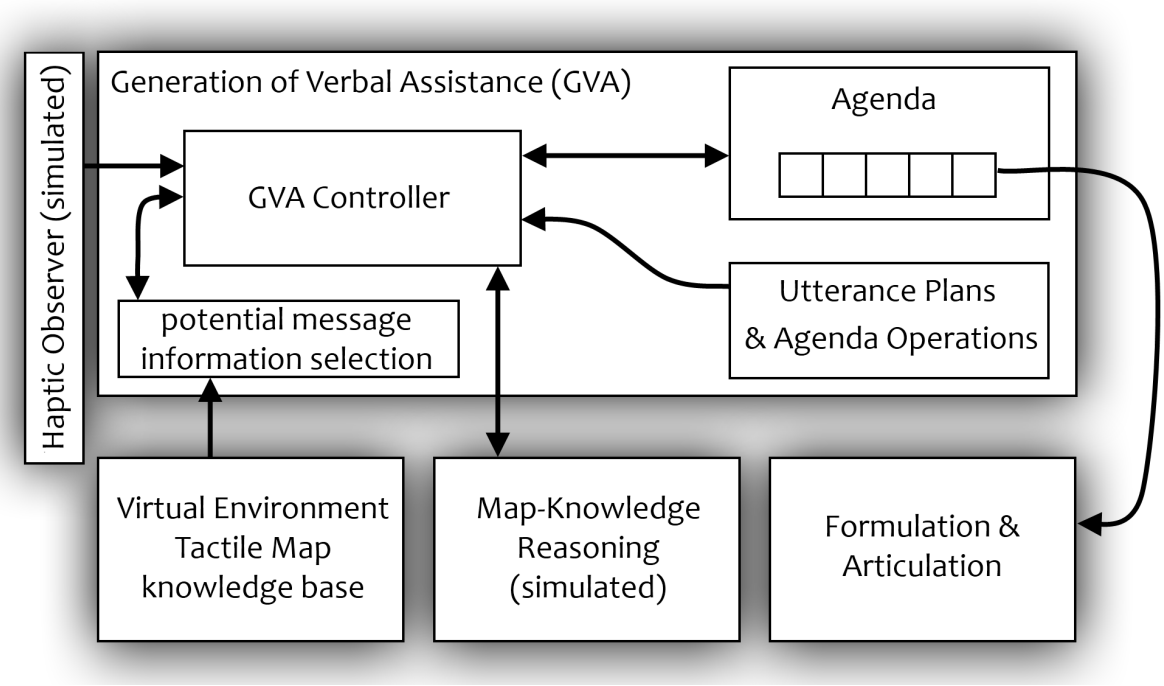


Abbildung 3: Die Kernkomponente von VAVETaM

Es stellte sich heraus, dass eines der größten Probleme von VAVETaM die Inflexibilität der Sprachausgabe angesichts schneller Explorationsbewegungen ist (siehe Lohmann, Eichhorn und Baumann 2012, Abschitt 7). Gemeint ist damit, dass VAVETaM bislang noch nicht in der Lage ist, eine Äußerung, dessen Aussprache bereits angefangen wurde, zu einem geeigneten Zeitpunkt abzubrechen, wie es der Assistent im ersten Beispiel in Kapitel 2 tut, geschweige denn eine solchen Äußerung dynamisch anzupassen wie im zweiten Beispiel dargestellt. Lohmann, Eichhorn und Baumann schlugen deshalb bereits die Verwendung inkrementell arbeitender Module vor.

4.2 InproTK

InproTK ist ein Softwarewerkzeug, das inkrementelle Verarbeitung von Dialogen ermöglicht (InproTK steht für ‚**In**cremental **pro**cessing **T**ool**k**it'). Es stellt diejenige inkrementelle TTS-Komponente bereit, die durch die im Kapitel 5 beschriebenen Architektur mit VAVETaM kombiniert wird.

InproTK folgt dem von Schlangen und Skantze (2009) vorgeschlagenen abstrakten Modell zur inkrementellen Dialogverarbeitung. Bei diesem Verarbeitungsmodell ist bezüglich der Dialogverarbeitung bewusst keine Direktionalität vorgegeben. Das hat zum Effekt, dass InproTK sowohl zur Spracherkennung als auch zur Sprachproduktion eingesetzt werden kann, was auch bereits im Rahmen von verschiedenen Projekten erprobt wurde (beispielsweise von Kennington, Kousidis und Schlangen 2013; Otsuka u. a. 2013; Baumann u. a. 2013).

Die wichtigsten Bestandteile dieses Modells sind *Incremental Units*, *Prozessoren* und *Puffer*, die in den folgenden Abschnitten näher beschrieben werden. Dabei wird auch verdeutlicht, welche Möglichkeiten InproTK auf diese Weise bietet.

4.2.1 Incremental Units

Incremental Units (IUs) sind die Inkremente von InproTK, die atomaren Prozesseinheiten, die die Ein- und Ausgabe eines Prozessors bilden. Sie lassen sich auf zwei verschiedene Arten miteinander assoziieren. Einerseits durch *same-level-links*, die typischerweise signalisieren, dass zwei IUs gleichen Ranges aufeinander folgen. Es können allerdings beliebige semantische Zusammenhänge wie beispielsweise Alternativen durch same-level-links modelliert werden. Auf der anderen Seite kann durch *grounded-in-links* eine hierarchische Strukturierung der IUs vorgenommen werden. *Grounded-in-links* werden verwendet, um durch einen Prozess entstehende IUs mit den Eingabe-IUs des Prozesses zu assoziieren. Damit kann nachvollzogen werden, welche IUs aus welchen anderen IUs resultieren.

Außerdem können sich Feedback-Zuhörer bei IUs registrieren, die über eine Veränderung des Verarbeitungsstatus informiert werden. Eine IU kann dabei die Zustände *upcoming*, *ongoing* und *completed* annehmen und durchläuft all diese in ihrem Lebenszyklus.

4.2.2 Prozessoren und Puffer

Prozessoren verarbeiten Eingabe-IUs durch ihren individuellen Prozess zu Ausgabe-IUs und übernehmen die Aufgabe, die IUs miteinander durch Links zu assoziieren. Jeder Prozessor hat dabei einen linken und einen rechten Puffer. Der linke Puffer dient als Lager für Eingabe-IUs und der rechte beinhaltet die Ausgabe-IUs. Aus einem funktionalen Blickwinkel betrachtet kann ein Prozessor als die funktionale Abbildung seines linken Puffers auf seinen Rechten gesehen werden. Prozessoren werden so aneinandergereiht, dass der rechte Puffer des einen Prozessors gleichzeitig der linke Puffer mindestens eines anderen Prozessors ist. Die so entstehende Verarbeitungskette bildet den gesamten inkrementellen Prozess ab.

Verändert sich der linke Puffer eines Prozessors, so wird der Prozessor in Gang gesetzt und damit dessen rechter Puffer neu berechnet. Wenn sich dabei ergibt, dass eine IU, die zuvor im rechten Puffer war nicht mehr aktuell ist, so wird diese zurückgezogen und mit Hilfe der grounded-in-links alle, die aus ihr durch nachfolgende Prozesse resultierten, sofern dies noch möglich ist.

Berechnete Resultate werden aufgrund ihrer vorläufigen Natur als *Hypothesen* betrachtet, die nicht sicher sind und durch neue Eingaben bestätigt (und ausgebaut) oder verworfen werden können.

4.2.3 Die verwendete Schnittstelle

InproTK bietet eine Softwareschnittstelle an, die es ermöglicht, relevante Operationen aus der Domäne der inkrementellen Verarbeitung zu verwenden. Dies sind insbesondere *add*, *revoke* und *commit*.

- *add* fügt eine IU an das Ende des linken Puffers eines Prozessors hinzu, wo diese auf ihre Verarbeitung wartet.
- *revoke* ist die Komplementäroperation zum *add*, die eine noch nicht weiter verarbeitete IU aus dem linken Puffer eines Prozessors zurückzieht.
- *commit* ist ein Signal für den Prozessor, dass eine bestimmte IU nicht mehr mit *revoke* entfernt werden wird und daher verlässlich hinsichtlich der Hypothesenbildung ist.

Andere Operationen lassen sich auf diese drei abbilden. So lässt sich etwa das Einfügen einer IU an einer bestimmten Position im Puffer als *revoke* von allen IU nach der gewünschten Position, *add* der neuen IU und anschließendem *add* der zuvor entfernten IUs realisieren.

Über diese Schnittstelle wird das inkrementelle Sprachsynthesemodul von InproTK im nun folgenden Kapitel 5 durch eine erweiterte Architektur von VAVETaM verwendet, um dynamische Anpassungen der produzierten Sprache zu ermöglichen.

5 Beschreibung der erweiterten Architektur

In diesem Kapitel wird beschrieben, wie in einer Erweiterung der Architektur von VAVETaM eine Anbindung an InproTK erfolgt ist. Hierfür wurden ausgehend von der in Kapitel 4.1 beschriebenen nicht-inkrementellen Architektur, die Lohmann, Kerzel und Habel (2012) vorschlagen einige Anpassungen vorgenommen, die eine sinnvolle Adaption an die von InproTK gebotene Schnittstelle ermöglichen. Diese erweiterte Architektur ermöglicht es VAVETaM, generierte Assistenz dynamisch anzupassen. Ein Prototyp dieser Architektur wurde wie bereits der erste Prototyp (vgl. Lohmann, Eichhorn und Baumann 2012) in Java implementiert. Ein Beispieldurchlauf dieses Prototypen wird in Kapitel 6 untersucht.

Aus Abbildung 2 ist ersichtlich, dass das Modul *Formulation & Articulation*, das für die Lexikalisierung und die Realisierung der Oberflächenform nach Reiter und Dale (2000) verantwortlich ist und das Modul *Audio Output*, die TTS-Komponente, in der ursprünglichen Architektur unidirektional miteinander verbunden sind. Vor allem diese Verbindung ist in der erweiterten Architektur wesentlich verändert und zu einer bidirektionalen Verbindung ausgebaut worden. Den Platz der TTS-Komponente in der ursprünglichen Architektur nimmt nun das inkrementelle Sprachsynthesemodul von InproTK ein.

Die Module *Haptic Observer*, *Virtual Environment Tactile Map knowledge base* und *potential message information selection* sind strukturell und funktionell nicht nennenswert verändert worden. Das Modul *Map-Knowledge Reasoning* ist im erweiterten Design zunächst noch nicht enthalten, lässt sich aber leicht in seiner vorherigen Form wieder integrieren.

In der erweiterten Architektur ist die Aufteilung der Module im Groben gleich geblieben. Geändert hat sich hingegen dass sie alle nebenläufig operieren. Sie erweitern ein abstraktes Modul, das an die Idee der Prozessoren von InproTK angelehnt ist. Wie sie besitzt auch dieser abstrakte *Pipeline-Prozessor* einen Datenpuffer, der die Eingabe zeitlich von der Verarbeitung entkoppelt. Dies ist für die Nebenläufigkeit der Module unerlässlich. Ein Pipeline-Prozessor kann eine beliebige Anzahl an Nachfolgern haben, die seine Ausgabe weiter verarbeiten. Abbildung 4 verdeutlicht das generelle Prinzip und Zusammenspiel solcher Komponenten.

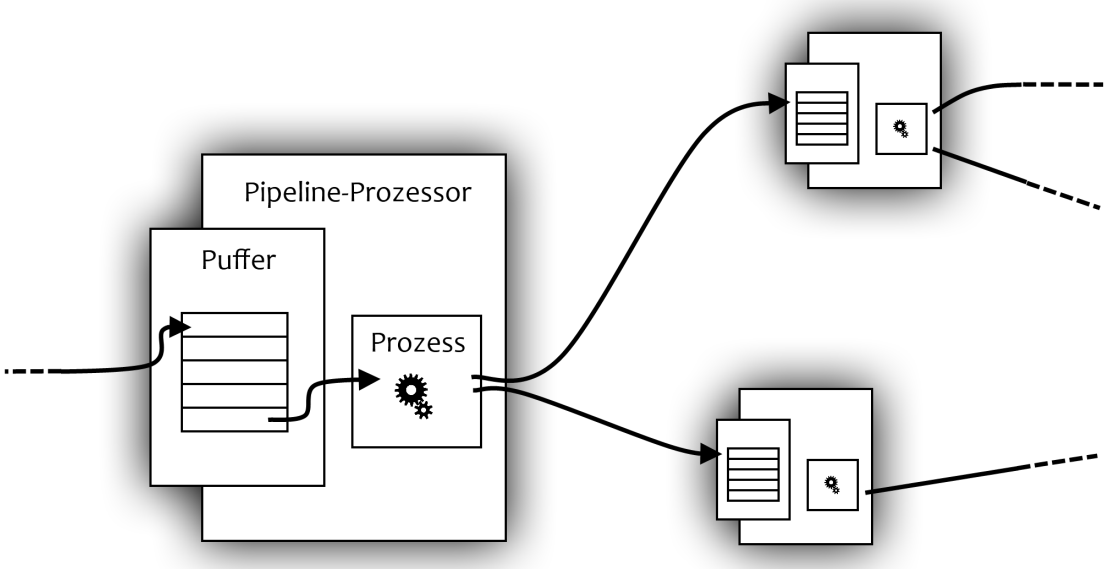


Abbildung 4: Pipeline-Prozessoren

Jedes so implementierte Modul muss lediglich den Verarbeitungsprozess von Eingabe zu Ausgabe, oder aus einem anderen Blickwinkel betrachtet die funktionale Abbildung beschreiben und braucht keine weitere individuelle zeitliche Steuerung oder Pufferverwaltung. Ist eine Eingabe im Puffer vorhanden und das Modul nicht beschäftigt, so wird der individuelle Verarbeitungsprozess mit der neuen Eingabe vom Modulgerüst angestoßen und dessen Ausgabe automatisch an alle Nachfolger weitergegeben, wobei in der hier vorgestellten Architektur jedes Modul maximal einen Nachfolger hat.

Basierend auf dem Pipeline-Prozessor wurde das Modul *Formulation & Articulation* von Grund auf neu implementiert. Der Pseudocode zur Hauptmethode der Formulierungskomponente im Anhang beschreibt den Ablauf in der neuen Implementation. Es enthält einen eigenständigen *Formulator* für jede *Intention*, die in einem *Diskursplan* enthalten sein kann. Dieser generiert mit Hilfe eines zentralen Wortschatzes an einzelnen Wörtern und Teilsätzen (den *Templates*) alle bekannten Formulierungen für einen Sachverhalt. Einem Formulator können eine oder mehrere PVMs (im Prototypen bisher nur bis zu zwei) als Eingabe übergeben werden und die resultierenden Formulierungen decken die Aussagen aus allen PVMs ab.

Für jede in VAVETaM erkannte MEP wird ein *Informationsaggregat* aufgebaut, eine Datenstruktur, die in den einzelnen Schritten nach und nach mit allen Informationen angereichert wird, die mit der MEP assoziiert sind. Dies beinhaltet

- die MEP selbst,
- den für sie selektierten Diskursplan mit seinen jeweiligen Intentionen,

- Referenzen zu den relevanten Kartenobjekten die Gegenstand des Diskurs werden sollen und Prädikate über deren Relationen zueinander, verpackt zu PVMs und
- die aus den PVMs resultierenden Formulierungsalternativen.

Trifft ein solches Informationsaggregat in der Formulierungskomponente ein, so enthält es bereits aufgrund voriger Verarbeitungsschritte die für die Situation als passend erachteten PVMs. Folgender Algorithmus wird dann von der Formulierungskomponente vollzogen:

1. Es wird überprüft, ob zum Zeitpunkt des Eintreffens eines neuen Informationsaggregates ein anderes noch nicht vollständig verarbeitet wurde, das heißt ob gerade etwas gesagt wird.
2. ist dies der Fall, so ist eine dynamische Anpassung angebracht. Es geschieht folgendes:
   (a) Sowohl die PVMs des aktuellen als auch des neu eingetroffenen Informationsaggregates werden für einen neuen Formulierungsprozess als Eingabe verwendet.
   (b) Der Formulator setzt aus allen ihm bekannten Templates fertige textuelle Äußerungen zusammen, deren Inhalt beide PVMs abdeckt.
   (c) Der Wortlaut der so generierten Formulierungen wird mit dem der aktuellen Äußerung abgeglichen.
   (d) Es werden diejenigen Formulierungen heraus gefiltert, die nicht mindestens bis zu dem Sprechfortschritt mit der aktuellen Äußerung übereinstimmen.
   (e) Aus den verbleibenden Formulierungen wird die Äußerung mit dem frühesten Unterschied zur aktuellen Äußerung herausgesucht.
   (f) Aus dem linken Puffer des inkrementellen Sprachsynthesemoduls werden alle noch nicht geäußerten IUs zurückgezogen und aus der ausgewählten Formulierung alle Worte entfernt, die durch die aktuelle Äußerung bereits abgedeckt sind, weil sie schon gesagt wurden.
   (g) Die verbleibenden Worte der neuen Formulierung werden in Wort-IUs verpackt und im linken Puffer des Synthesemoduls abgelegt.
   (h) Die damit abgearbeitete PVM wird aus dem neuen Informationsaggregat entfernt.
3. Alle (übrigen) PVMs des neuen Informationsaggregates werden der Reihenfolge nach als Eingabe für einen Formulierungsprozess verwendet.
4. Aus den jeweils ausgegebenen alternativen Formulierungen wird eine ausgewählt.
5. Diese wird in Wort-IUs zerlegt.
6. Es werden, Feedback-Zuhörer an diesen IUs installiert, die über eine Veränderung des Abarbeitungsstatus informiert werden (um beispielsweise bei Eintreffen eines weiteren Informationsaggregates in der Lage zu sein, zu bestimmen, ob momentan noch ein anderes bearbeitet wird).
7. Die Wort-IUs werden in den linken Puffer des inkrementellen Synthesemoduls gelegt.

Es werden also stets *alle* Möglichkeiten, einen Sachverhalt mit Hilfe der Templates auszudrücken ausformuliert und anschließend die passendste Formulierung weiter verwendet. Welche Formulierung als passend erachtet wird ist dabei Situationsabhängig. Soll nur eine einfache Äußerung getätigt werden, bei der nicht an eine im Gange befindliche adaptiert werden muss, so werden in der aktuellen Implementierung des Prototypen alle Formulierungen als gleichwertig

angesehen und es wird daher zugunsten der Variation zufällig eine von ihnen ausgewählt.

Anders verhält sich das System im Falle einer dynamischen Anpassung bzw. der Reparatur einer Aussage. In jedem Falle werden komplette Sätze formuliert, allerdings decken die bei einer dynamischen Anpassung generierten Formulierungen auch die bereits begonnene Aussage ab. Wird also beispielsweise (wie in Abbildung 5 dargestellt) gerade der Satz „Das ist die Hauptstraße.“ ausgesprochen, der Sprechfortschritt ist bei dem Wort „die“ und es soll nun aber die Information präsentiert werden, dass sich der Erkunder auf dem Lärchenweg befindet, so werden zunächst unter anderem die Sätze „Das ist die ... äääh ... der Lärchenweg“, „Das ist die Hauptstraße und jetzt bist du auf dem Lärchenweg“ und „Hier ist die Hauptstraße und das ist der Lärchenweg“ formuliert[2].

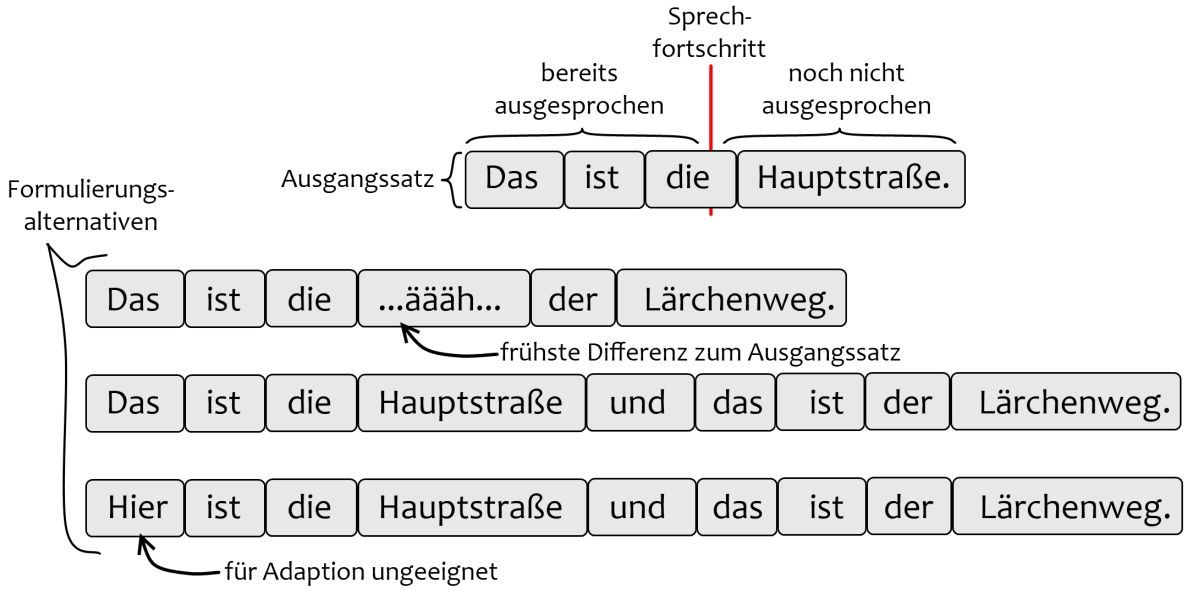


Abbildung 5: Eine Äußerung und ihre Adaptionsmöglichkeiten

Nun werden alle Formulierungen darauf überprüft, ob angesichts des Sprechfortschrittes überhaupt noch an sie adaptiert werden kann. Der dritte Satz würde in diesem Schritt ausgefiltert, da er anders beginnt als der zuvor angefangene. Aus den verbleibenden Formulierungen wird diejenige ausgewählt, die *am frühsten nach dem Sprechfortschritt eine Differenz aufweist*, weil angenommen wird, dass auf diese Weise immer so frühzeitig wie möglich *neue* Informationen präsentiert werden können. In dem genannten Beispiel wäre dies die erste Formulierung, da sie direkt nach dem Wort „die“ anders ist als der ursprüngliche Satz. Gibt es mehrere Formulierungen, die an der gleichen Stelle eine Differenz aufweisen, so wird zufällig eine von ihnen ausgewählt. Als nächstes müssen nur noch alle noch nicht ausgesprochenen Worte der ursprünglichen Äußerung aus dem linken Puffer der inkrementellen Synthese durch eine *revoke*-Operation entfernt und mit Hilfe einer *add*-Operation durch diejenigen Worte der neuen Formulierung ersetzt werden, die nach dem Sprechfortschritt stehen.

Im folgenden Kapitel 6 wird der Anpassungsmechanismus in einem Beispieldurchlauf untersucht. Dabei wird deutlich, wie mit Hilfe dieses Verfahrens sowohl Abbrüche als auch Reparaturen von Aussagen umgesetzt werden können.

[2]Es werden normalerweise wesentlich mehr Alternativen generiert, dies ist nur eine Auswahl zur Veranschaulichung.

6 Beispieldurchlauf & Diskussion

In diesem Kapitel wird dargestellt, wie der Prototyp der in Kapitel 5 vorgestellten Architektur auf die in Kapitel 2 verwendeten Beispiele reagiert. Es wird gezeigt, dass das inkrementelle Verarbeitungsmodell im Vergleich zur vorher verwendeten, nicht-inkrementellen Verarbeitung eine Sprachausgabe produziert, die wesentlich näher an den in den Beispielen antizipierten Äußerungen liegt. Es werden dabei nur die ersten beiden Beispiele betrachtet, weil das dritte Beispiel ein Phänomen beschreibt, das bei VAVETaM nicht auftritt.

Der Prototyp, der den nachfolgenden Output produziert hat verwendete dabei die gleiche Art von simuliertem Input wie bereits der Prototyp des nicht-inkrementellen Systems (vgl. Lohmann, Eichhorn und Baumann 2012). Auch der neue Prototyp kann so konfiguriert werden, dass er eine nicht-inkrementelle Verarbeitung vornimmt, was einen direkten Vergleich der Ausgaben bei inkrementeller und nicht-inkrementeller Verarbeitung ermöglicht. Hierfür wurden die beiden ersten Beispiele aus Kapitel 2 in einem Ablaufplan modelliert und als Input verwendet.

Außerdem musste bei dem Prototypen auf eine tatsächliche Anbindung an InproTK verzichtet werden, da hierbei Performanceprobleme auftraten, die zum Zeitpunkt der Erstellung des Fallbeispiels noch nicht gelöst werden konnten. Stattdessen wurde ein Dummy-Modul verwendet, das die gleiche Schnittstelle wie InproTK anbietet und das zeitliche Verhalten der Synthese annähert, aber keine Sprache ausgibt. Es ist in der Lage, durch Feedbackmechanismen über seinen simulierten Sprechfortschritt Auskunft zu geben und produziert dabei eine detaillierte Log-Ausgabe, die Grundlage für diese Analyse ist[3]. Aus diesen Logdaten ist ersichtlich, wie viel Zeit die Aussprache eines Inkrements benötigt und zu welchen Zeitpunkten die Feedbackmechanismen aktiviert werden.

In der Modellierung beider Beispiele gibt es jeweils nur zwei MEPs, die aufeinander folgen. Dies ist einerseits hinreichend, um adaptives Verhalten beobachten zu können und andererseits minimal, um die Beispiele möglichst einfach und nachvollziehbar zu halten. Die wichtigen Ereignisse in den insgesamt vier simulierten Abläufen sind in Abbildung 6 dargestellt. Als Nullpunkt der Zeitachse wurde jeweils der Zeitpunkt der Simulation der ersten MEP gewählt, da der Prototyp stark variierende Initialisierungszeiten aufweist.

In Abbildung 6a ist der zeitliche Ablauf vom Beispieldurchlauf für das erste Beispiel in Kapitel 2 dargestellt. Die in Abbildung 1 illustrierte Bewegung zu diesem Beispiel resultiert in die Erkennung zweier MEPs, nämlich `TrackMEP(pt3)` und kurz darauf folgend `LandmarkMEP(plm1)`, also die Erkennung der Erkundung der Berliner Straße und der Hauptschule. Die Abbildung zeigt, dass in diesem Versuchsdurchlauf das Aussprechen von „dies ist die Hauptschule“ bei nicht-inkrementeller Verarbeitung etwa 3200ms nach dem Anfang der Simulation begonnen wird. Bei der Verwendung inkrementeller Verarbeitung hingegen wird die gleiche Aussage bereits ca. 2500ms nach dem Start begonnen aus zu sprechen. Es konnte also durch die Anwendung eines Abbruchs trotz der zusätzlichen Füllworte „ääh“ und „also“ eine Einsparung von etwa 700ms bei der Präsentation der jeweils aktuellsten Information erzielt werden. Die Äußerung ist eine Reaktion auf die zweite, 1850ms nach dem Startzeitpunkt simulierte MEP. Noch viel wichtiger ist daher die Tatsache, dass die Reaktionszeit des Systems, also die Zeit vom Eintreffen der neuen MEP bis zum Zeitpunkt, zu dem die Sprachausgabe auf diese eingeht, ohne inkrementelle Verarbeitung rund 1400ms beträgt, während mit inkrementeller Verarbeitung nur gut 200ms vergehen. Gemeint ist damit im Falle nicht-inkrementeller Verarbeitung der Beginn der Äußerung „Dies ist die Hauptschule“, im Falle der inkrementellen Verarbeitung der Beginn des Zögerlautes „äääh“. Das Verstreichen dieser 200ms ist darüber hinaus größtenteils der Tatsache geschuldet,

[3] Ein Hinweis zu den ausführlichen Log-Mitschnitten findet sich im Anhang

dass kurz vor Beendigung der Berechnung der kombinierten Formulierungsalternativen das Wort „sind“ von der Synthese angefangen wird auszusprechen und dies erst beendet werden muss, da als Größe der Inkremente immer ganze Wörter gewählt wurden. Das System wäre also theoretisch bereits noch früher in der Lage gewesen, auf den neuen Input zu reagieren. Die `add`-Operation, die das „äääh“ in den linken Puffer der Synthese einfügt erfolgt bereits 16ms nach dem Eintreffen der zweiten MEP. Dass der Prototyp im inkrementellen Verarbeitungsmodus eine Reaktionszeit von etwa 250ms für die erste MEP benötigt scheint durch nicht abgeschlossene Initialisierungsprozesse bedingt zu sein und konnte bei anderen Durchläufen der selben Simulation nur inkonsistent reproduziert werden.

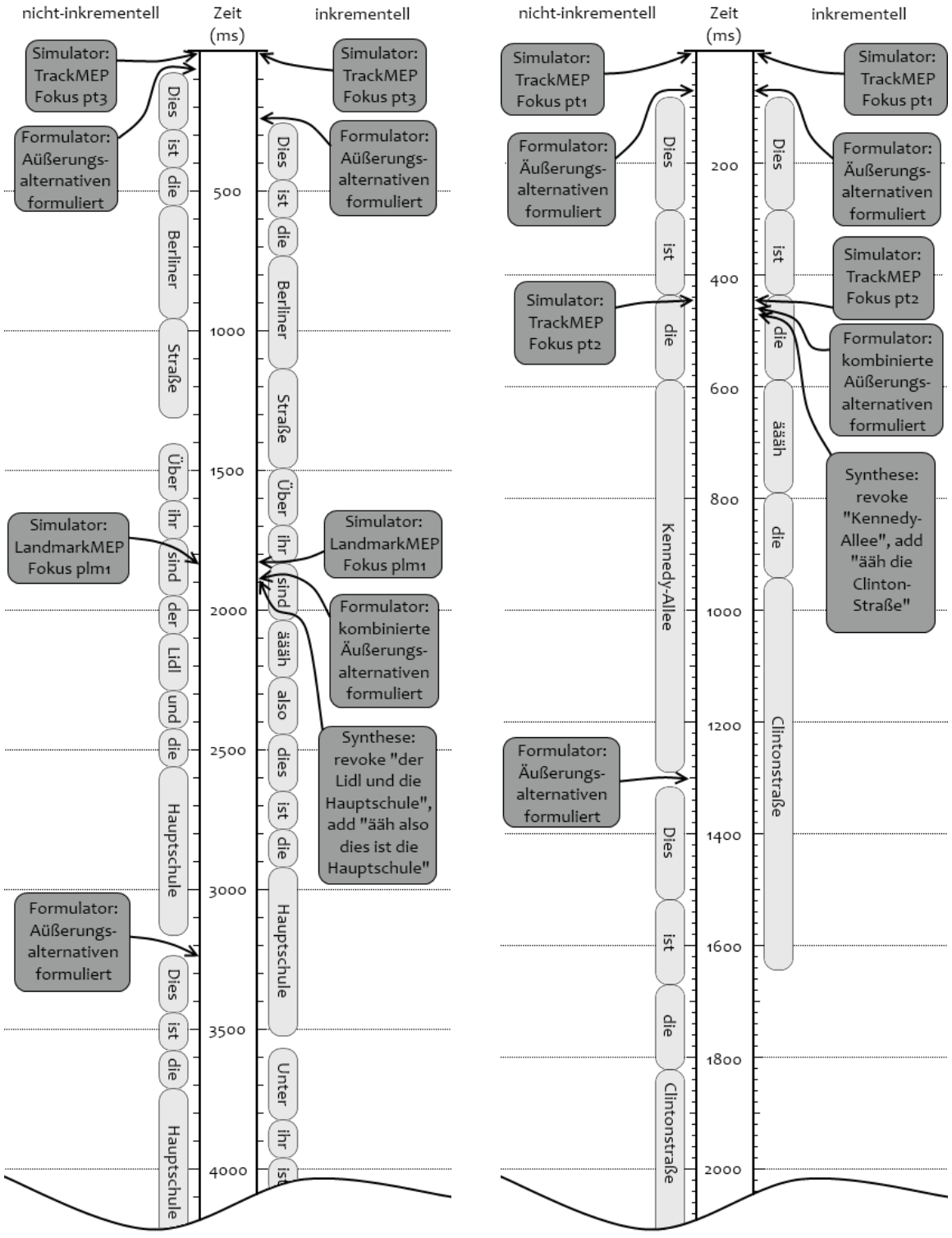
Abbildung 6b zeigt den zeitlichen Abfolge vom Beispieldurchlauf für das zweite Beispiel. Die wiederum in Abbildung 1 dargestellte Explorationsbewegung zu diesem Beispiel lässt VAVETaM zwei MEPs erkennen: `TrackMEP(pt1)` und `TrackMEP(pt2)`, die in einem zeitlichen Abstand von weniger als 500ms aufeinander folgen. Sie beschreiben die Erkennung der Erkundung der Kennedy-Allee und der Clintonstraße, worauf VAVETaM entsprechend reagiert. Betrachtet man die Reaktionsgeschwindigkeit für die zweite MEP, so wird klar, dass in diesem Durchlauf eine ähnlich beträchtliche Verbesserung wie im vorigen erzielt werden konnte. Das Aussprechen von „die Clintonstraße“ beginnt bei nicht-inkrementeller Verarbeitung rund 1200ms, bei nicht inkrementeller Verarbeitung bereits etwa 300ms nach dem Eintreffen der zweiten MEP. Die Reaktion auf die MEP beginnt allerdings bereits vorher, nämlich schon bei dem Zögerungslaut „‘ääähh“, der die Reparatur einleitet. Die Dauer vom Eintreffen der zweiten MEP bis zu dieser Reaktion beträgt ca. 100ms. Die Dauer bis zur `add`-Operation, die die neue formulierten Worte in den linken Puffer der Synthese übergibt beträgt erneut 16ms. Nach dieser Zeit (zuzüglich der Zeit für eigene Berechnungen vor der Sprachausgabe) wäre die Synthese also theoretisch in der Lage, auf den neuen Input zu reagieren, sofern sie nicht mit der Ausgabe einer anderen Äußerung beschäftigt ist.

Neben der Verbesserung im zeitlichen Verhalten von VAVETaM bei beiden Beispielen wurde außerdem durch die Verwendung von Reparaturen an den sprachlichen Äußerungen ein stetigerer Redefluss erzeugt, der menschlicher wirkt und daher in sich bereits eine Verbesserung des Interfaces darstellt. Auf der anderen Seite belieben Informationen bei der dynamischen Anpassung von Äußerungen (zunächst) unerwähnt, wie etwa die Information, dass sich im ersten Beispiel über der Berliner Straße neben der Hauptschule auch der Lidl befindet. Dies ist jedoch letztendlich der gewollte Effekt, da die Identifikation der Hauptschule mit dem Eintreffen der neuen MEP als wichtiger erachtet wird. Außerdem (dies ist aus der Grafik nicht ersichtlich) könnte bereits die nächste Äußerung dieses Problem wieder beheben, sei sie beispielsweise „Unter ihr ist die Berliner Straße und rechts neben ihr der Lidl“.

7 Zusammenfassung & Ausblick

In dieser Arbeit wurde gezeigt, dass natürlichsprachliche Interfaces durch die Verwendung von inkrementeller Verarbeitung verbessert werden können, da sich menschliches Verhalten auf diese Weise besser oder überhaupt erst modellieren lässt. Eine zentrale Rolle spielen dabei Reparaturen von sprachlichen Äußerungen, die anhand von drei Beispielen veranschaulicht wurden. Dies erfordert eine Aufhebung der klassischen Unterteilung der Sprachproduktion in Generierung und Synthese, weil diese beiden Teilsysteme für die dynamische Anpassung von produzierter Sprache enge Feedbackmechanismen unterhalten müssen.

Nachdem Grundlagen und Forschungsergebnisse zur Produktion natürlicher Sprache präsentiert wurden, folgte eine detaillierte Beschreibung der beiden Systeme VAVETaM und InproTK. Die Architektur von VAVETaM wurde dann erweitert, um beide Systeme auf eine Weise mit-



(a) Beispiel 1

(b) Beispiel 2

Abbildung 6: Zeitstrahlen der beiden Beispiele

einander interagieren zu lassen, die Reparaturen von sprachlichen Äußerungen ermöglicht. Eine prototypische Implementation dieser Architektur wurde anschließend in einem Beispieldurchlauf bewertet und die Resultate diskutiert.

Es hat sich gezeigt, dass sich erst die jüngste Forschung der Thematik der Verbindung von inkrementeller Generierung und inkrementeller Synthese angenommen hat, wohingegen vorherige Forschung zwar Inkrementalität in die Produktion natürlicher Sprache integriert hat, dies aber oft mit einem Fokus auf die Generierung getan hat. Dies liegt sicherlich daran, dass zunächst nur einzelne Aspekte erforscht werden können, bevor ein ganzheitlicher Ansatz die Erkenntnisse zusammenführen kann. Auch die in Kapitel 5 vorgestellte Architektur kann mitnichten Vollständigkeit für sich beanspruchen, sondern beleuchtet vielmehr bewusst den Aspekt der *Verbindung* von Generierung und Synthese in einem inkrementell arbeitenden System.

7.1 Probleme und zukünftige Arbeit

Der Prototyp musste aus technischen Gründen an zwei entscheidenden Stellen auf simulierte Komponenten zurückgreifen, nämlich bei der Erhebung des Inputs[4] und bei der inkrementellen Sprachsynthese. Bei der Simulation des Inputs konnte auf sehr präzise transkribierte Explorationsverläufe zurückgegriffen werden, die bei einem Wizard-Of-Oz Experiment während der Vorstudien zum ursprünglichen Entwurf von VAVETaM entstanden. Die inkrementelle Sprachsynthese wurde durch ein Modul Simuliert, das das antizipierte Verhalten von InproTK nachahmt und dabei keine tatsächliche Sprachausgabe, sondern nur Log-Mitschnitte der Verarbeitungsabläufe produziert. Die Dauer für die simulierte Aussprache von Inkrementen musste dabei angenähert werden.

Obwohl diese beiden Dummy-Module zwar hinreichend geeignet waren um das adaptive Verhalten der Architektur zu untersuchen, sollte ihre Ersetzung durch tatsächlich funktionierende Module bei weiterer Arbeit an dem Projekt oberste Priorität haben. Nicht nur weil erst bei einem voll funktionsfähigen System die Untersuchungsergebnisse akkurat sind, sondern vielmehr weil sich so überhaupt erst die Möglichkeit eröffnet, Nutzerstudien und verlässliche statistische Untersuchungen an dem System zu vollführen. Selbst weitere Wizard-Of-Oz Experimente sind zunächst nicht sinnvoll, weil dabei der Wizard für die Eingabe von korrekt spezifizierten MEPs zuständig wäre, was eine nicht zu bewältigende Aufgabe während einer Exploration in Echtzeit ist. Die einzige denkbare weitergehende Untersuchung am bestehenden System ist die Bewertung von Systemausgaben aus inkrementeller und nicht-inkrementeller Verarbeitung durch Versuchsteilnehmer.

Des Weiteren könnte bei einer Weiterentwicklung Abstand von dem bestehenden Template-Ansatz genommen werden, wenn beispielsweise eine Formulierung mit Hilfe einer Baumadjunktionsgrammatik implementiert würde. Diese könnte weiterhin die Ersetzungsmechanismen zur Umsetzung von Reparaturen an sprachlichen Äußerungen verwenden und gleichzeitig eine größere Vielfalt und Flexibilität von möglichen Äußerungen erlauben. Es wäre außerdem Sinnvoll, die optimale Größe von Inkrementen systematisch zu untersuchen. Je kleiner diese sind, desto schneller kann ein System reagieren, zu kleine Inkremente auf der anderen Seite würden Abbrüche und Reparaturen an Stellen hervorbringen, die nicht natürlich wirken. Auch das Verfahren zur Bewertung der am besten geeigneten Formulierung für eine Situation könnte verfeinert werden. Es wäre beispielsweise erstrebenswert, bei besonders schnellen Benutzerinterkationen möglichst kurze Formulierungen zu bevorzugen, während langsamere Explorationshandlungen eher ausführlichere Formulierungen hervorbrächten.

[4] Dies war auch schon beim Prototypen des Ausgangssystems VAVETaM der Fall.

7.2 Verallgemeinerbarkeit der Ergebnisse

In diesem letzten Abschnitt wird diskutiert, inwiefern sich die Erkenntnisse aus den Kapiteln 5 und 6 auch in anderen Zusammenhängen, als den hier dargestellten verwenden lassen.

Die in Kapitel 6 dargestellten Abläufe sind Beispiele dafür, wozu VAVETaM auf einer genereller Ebene in der Lage ist. Es verfügt durch die inkrementelle Architektur über einen Mechanismus, der die dynamische Anpassung der im Gang befindlichen Äußerung ermöglicht, immer vorausgesetzt, dass die Formulierungskomponente mindestens eine Äußerung generiert, die mindestens bis zum Sprechfortschritt deckungsgleich mit der aktuellen Äußerung ist. Es ist also ein rein technisches Problem, die Formulierungskomponente mit allen erdenklichen möglichen Äußerungen anzureichern, damit auch in anderen Situationen eine dynamische Anpassung stattfinden kann.

Blickt man über die Domäne von VAVETaM hinaus, so finden sich weitere denkbare Einsatzbereiche für den Anpassungsmechanismus, denn seine relative Unabhängigkeit von der eigentlichen Formulierungskomponente macht ihn flexibel einsetzbar. Denkbar wären andere sprachgestützte Navigationsaufgaben, die in einer schnelllebigen Umgebung angesiedelt sind. Betrachtet man aktuelle Navigationssysteme von Autos, so ist dort kein großer Bedarf für derart flexible Anpassung von sprachlichen Äußerungen. Individuellere und interaktivere Arten der persönlichen Navigation wären jedoch möglich, wenn beispielsweise ein Fußgänger unterwegs mit einem tragbaren System, das die Umgebung durch eine geeignete Kamera wahrnehmen kann, stark situative verbale Richtungsangaben erhält. Ein solches Navigationssystem könnte eine Äußerung wie etwa „Da, wo jetzt der Radfahrer mit der gelben Jacke langfährt musst du links einbiegen.“ produzieren, eine Äußerung, die nur durch die sehr akkurate Synchronisation der Sprachproduktion mit äußeren, unbeeinflussbaren Ereignissen möglich ist.

Noch genereller ausgedrückt kann der Anpassungsmechanismus tendenziell überall dort angewendet werden, *wo ein automatisch generierter, möglichst natürlicher Sprachfluss auf durch das generierende System unbeeinflussbare aber beobachtbare Ereignisse eingehen soll.* Ein hervorragendes Beispiel für einen solchen Einsatzbereich ist das in Abschnitt 3.3 vorgestellte System VITRA von Herzog u. a. (1996), das Handlungen von Fußballspielern auf dem Feld automatisch kommentiert.

VAVETaM hat also auf der einen Seite, wie im vorigen Abschnitt aufgezeigt, noch großes Entwicklungspotenzial und auf der anderen Seite schon in seinem jetzigen Zustand Erkenntnisse hervorgebracht, die weit über die eigene Domäne hinaus Geltung haben.

Anhang

A Pseudocode der Formulierungskomponente

```
void process(MapExploratoryProcedure mep) {
    if (mepProcessingOngoing()) { // eine MEP wird gerade noch verarbeitet
        List<PreverbalMessage> currentPVMs = getCurrentPVMs();
        if (currentPVMs.size() == 1){
            //die einzige aktuell wiedergegebene PVM wird versucht,
            //mit der ersten Neuen zu kombinieren
            PreverbalMessage currentPVM = currentPVMs.getFirst();
            PreverbalMessage newPVM = mep.getFirstPreverbalMessage();
            List<Formulation> combinedFormulations =
                formulate(currentPVM, newPVM);
            for (Formulation combinedFormulation: combinedFormulations) {
                if (!canAdapt(getCurrentFormulation(), combinedFormulation)) {
                    combinedFormulations.remove(combinedFormulation);
                }
            }
            if (combinedFormulations.size() > 0) {
                Formulation formulation = getBestFit(combinedFormulations);
                doAdapt(currentFormulation, formulation);
                // die erste PVM wurde damit abgearbeitet und wird entfernt
                mep.removeFirstPreverbalMessage();
            }
        }
    }
    for (IPreverbalMessage pvm: mep.getPreverbalMessages()) {
        List<Formulation> formulations = formulate(pvm);
        if (!formulations.isEmpty()){
            // beste verfügbare Formulierung
            Formulation formulation = formulations.getFirst();
            installUpdateHooks(formulation.getIUs());
            synthesis.hypothesisChange(formulation.getIUs(),
                formulation.getEditMessages());
        }
    }
}
```

Die Hauptmethode der Formulierungskomponente

B Anleitung zur Inbetriebnahme des Prototypen

Eine lauffähige Version des Prototypen aus Kapitel 6 liegt dieser Arbeit als gepackte Datei bei. Im ZIP-Archiv befindet sich eine Datei namens ‚README' mit weiteren Instruktionen zur Inbetriebnahme des Prototypen.

C Logs der Beispieldurchläufe

Die ausführlichen Logs der Beispieldurchläufe liegen dieser Arbeit als Textdateien bei. Für beide Beispiele existiert sowohl für den Durchlauf mit als auch für den Durchlauf ohne inkrementelle Verarbeitung eine eigene Datei.

Literatur

Baumann, Timo und David Schlangen (Sep. 2012). “Evaluating Prosodic Processing for Incremental Speech Synthesis”. In: *Proceedings of Interspeech*. ISCA. Portland, USA (siehe S. 2, 9).

Baumann, Timo, Maike Paetzel, Philipp Schlesinger und Wolfgang Menzel (März 2013). “Using Affordances to Shape the Interaction in a Hybrid Spoken Dialog System”. In: *Proceedings of ESSV*. Bielefeld, Germany (siehe S. 13).

Beutnagel, Mark, Alistair Conkie, Juergen Schroeter, Yannis Stylianou und Ann Syrdal (1999). “The AT&T next-gen TTS system”. In: *Joint meeting of ASA, EAA, and DAGA*. Citeseer, S. 18–24 (siehe S. 2).

Buschmeier, Hendrik, Timo Baumann, Benjamin Dorsch, Stefan Kopp und David Schlangen (2012). “Combining Incremental Language Generation and Incremental Speech Synthesis for Adaptive Information Presentation”. In: *Proceedings of SigDial*. Seoul, Korea, S. 295–303 (siehe S. 8).

Clark, Herbert H (2002). “Speaking in Time”. In: *Speech Communication* 36.1, S. 5–13 (siehe S. 6).

Dethlefs, Nina, Helen Hastie, Verena Rieser und Oliver Lemon (2012). “Optimising incremental generation for spoken dialogue systems: reducing the need for fillers”. In: *Proceedings of the Seventh International Natural Language Generation Conference*. Association for Computational Linguistics, S. 49–58 (siehe S. 9).

Dohsaka, Kohji und Akira Shimazu (1996). “A computational model of incremental utterance production in task-oriented dialogues”. In: *Proceedings of the 16th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, S. 304–309 (siehe S. 7).

Guhe, Markus, Christopher Habel und Heike Tappe (2000). “Incremental event conceptualization and natural language generation in monitoring environments”. In: *Proceedings of the first international conference on Natural language generation-Volume 14*. Association for Computational Linguistics, S. 85–92 (siehe S. 8).

Habel, Christopher (1982). “Referential nets with attributes”. In: *Proceedings of the 9th conference on Computational linguistics-Volume 1*. Academia Praha, S. 101–106 (siehe S. 10).

Herzog, Gerd, Anselm Blocher, Klaus-Peter Gapp, Eva Stopp und Wolfgang Wahlster (1996). “VITRA: Verbalisierung visueller Information”. In: *Informatik Forschung und Entwicklung* 11.1, S. 12–19 (siehe S. 7, 8, 22).

Joshi, Aravind K, Leon S Levy und Masako Takahashi (1975). “Tree adjunct grammars”. In: *Journal of computer and system sciences* 10.1, S. 136–163 (siehe S. 7).

Joshi, Aravind K und Yves Schabes (1991). *Tree-adjoining grammars and lexicalized grammars*. Techn. Ber. University of Pennsylvania (siehe S. 7).

Jurafsky, D. und James H. Martin (2009). *Speech and Language Processing*. 2nd Ed. Prentice Hall (siehe S. 1).

Kennington, Casey, Spyridon Kousidis und David Schlangen (2013). “Interpreting situated dialogue utterances: an update model that uses speech, gaze, and gesture information”. In: *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (siehe S. 13).

Kilger, Anne und Wolfgang Finkler (1995). *Incremental generation for real-time applications*. eng. Techn. Ber. Postfach 151141, 66041 Saarbrücken: Universitäts- und Landesbibliothek. URL: http://scidok.sulb.uni-saarland.de/volltexte/2011/3752 (siehe S. 7, 8).

Levelt, Willem JM (1983). “Monitoring and self-repair in speech”. In: *Cognition* 14.1, S. 41–104 (siehe S. 7).

— (1993). *Speaking: From intention to articulation*. MIT press (siehe S. 2).

Liberman, Mark Y und Kenneth W Church (1992). “Text analysis and word pronunciation in text-to-speech synthesis”. In: *Advances in speech signal processing*, S. 791–831 (siehe S. 2).

Lohmann, Kris, Ole Eichhorn und Timo Baumann (Juni 2012). “Generating Situated Assisting Utterances to Facilitate Tactile-Map Understanding: A Prototype System”. In: *Proceedings of the Third Workshop on Speech and Language Processing for Assistive Technologies*. Montréal, Canada: Association for Computational Linguistics, S. 56–65. URL: http://www.aclweb.org/anthology/W12-2908 (siehe S. 11, 12, 14, 18).

Lohmann, Kris, Matthias Kerzel und Christopher Habel (2012). “Verbally Assisted Virtual-Environment Tactile Maps: A Prototype System”. In: *Proceedings SKALID*, S. 25 (siehe S. 1, 9, 14).

McTear, Michael F. (2004). *Spoken Dialogue Technology–Toward the Conversational User Interface*. Springer (siehe S. 1).

Otsuka, Tsugumi, Kazunori Komatani, Satoshi Sato und Mikio Nakano (2013). “Generating More Specific Questions for Acquiring Attributes of Unknown Concepts from Users”. In: *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (siehe S. 13).

Reiter, Ehud und Robert Dale (2000). *Building Natural Language Generation Systems*. Cambridge, U.K.: Cambridge University Press (siehe S. 5–7, 14).

Schlangen, David und Gabriel Skantze (2009). “A general, abstract model of incremental dialogue processing”. In: *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, S. 710–718 (siehe S. 2, 13).

Schulz von Thun, Friedemann (2006). *Miteinander reden 1-3*. Rowohlt Taschenbuch (siehe S. 5).

Shannon, Claude Elwood und Warren Weaver (1949). *The Mathematical Theory of Communication*. Illini books Bd. 1. University of Illinois Press. ISBN: 9780252725487 (siehe S. 4).

Skantze, Gabriel und Anna Hjalmarsson (2013). “Towards incremental speech generation in conversational systems”. In: *Computer Speech & Language* 27.1, S. 243–262 (siehe S. 8).

Stone, Matthew, Christine Doran, Bonnie Webber, Tonia Bleam und Martha Palmer (2003). “Microplanning with communicative intentions: The SPUD system”. In: *Computational Intelligence* 19.4, S. 311–381 (siehe S. 8).

Taylor, Paul (2009). *Text-to-speech synthesis*. Bd. 15. Citeseer (siehe S. 2).

Watzlawick, Paul, Janet H Beavin und Don D Jackson (1969). *Menschliche Kommunikation: Formen, Störungen, Paradoxien*. 1. Aufl. H. Huber (siehe S. 5).

Ich versichere, dass ich die Bachelorarbeit im Studiengang Mensch-Computer-Interaktion selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg, den 28. Oktober 2013

_____

Ole Eichhorn

Hiermit erkläre ich mich einverstanden mit der Einstellung dieser Arbeit in die Informatik-Bibliothek der Universität Hamburg.

Hamburg, den 28. Oktober 2013

_____

Ole Eichhorn