COSMA

Cooperative Schedule Management Agent

BMBF-Förderkennzeichen ITW 9402

Abschlußbericht

Dr. Stephan Busemann
Thierry Declerck, MA
Dipl.-Inform. Abdel Kader Diagne
Dr. Elizabeth A. Hinkelman
Prof. Dr. Hans Uszkoreit

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Forschungsbereich Sprachtechnologie
Stuhlsatzenhausweg 3, 66123 Saarbrücken

15. Juni 1997

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Aufgabenstellung

Kalendermanagement ist ein Problem, dem täglich viele Personen und Organisationen gegenüberstehen und das typischerweise durch Kommunikation in natürlicher Sprache gelöst wird. Deshalb ist es sowohl ein natürlicher als auch ein nützlicher Test für ein natürlichsprachliches System. Interessante KI-Aspekte sind zeitliches [Allen, 1983; Koomen, 1989; Lamport, 1977] und räumliches Schließen sowie Konzepte eines Multiagentenszenarios wie Verhandlung und Zusatimmung [Cohen und Levesque, 1990b]. Kalendermanagement hat viele Vorteile als Testfall für die Evaluation von KI-Techniken: es ist konkret, leicht zu definieren, leicht erweiterbar, und sein Erfolg kann auch von naiven Computerbenutzern leicht bewertet werden, ohne daß sie auf Spezialisten zurückgreifen müssen.

Verarbeitung natürlicher Sprache beim Kalendermanagement läßt sich leicht mit Anwendungen vergleichen, die nicht aus der KI kommen: man betrachte z.B. einen effizienten Scheduling-Algorithmus, der auf einer zentralisierten Kalenderdatenbank läuft und der billige Computerleistung nutzt um optimale Zeitpläne zu berechnen. Obwohl diese zentralisierte Vorgehensweise kosteneffektiv sein kann, mag sie sich dennoch in der Praxis für einen durchschnittlichen Büroangestellten als völlig unbrauchbar erweisen. Zentralisierte Ansätze erlauben z.B. keine kreativen menschlichen Eingriffe zur Konfliktlösung. Außerdem werden sie durch verwaltungstechnische Grenzen, Datenschutz sowie die üblichen Zuverlässigkeitsüberlegungen und die katastrophale Natur zentraler Fehlfunktionen in ihrer Nützlichkeit beschränkt. In diesem Projekt standen daher dezentrale Lösungen im Mittelpunkt, die den Vorteil besitzen, den Individuen größtmögliche Kontrolle über die eigenen Zeitpläne zu geben. Außerdem wollen wir auch solchen Personen die Teilnahme am Planungsprozeß ermöglichen, deren Kalender nicht on-line zugänglich sind. Daher werden manchmal mehrere Kommunikationsrunden nötig sein. Wir glauben trotzdem, daß dies zu Systemen führen wird, die von größerem praktischen Nutzen sind und den Benutzer weniger einschränken.

Abbildung 1 zeigt eine hypothetische COSMA-Installation. Einige Instanzen von COSMA, die aus Schnittstelle, Sprachverarbeitungskomponente und Kalendersoftware bestehen, arbeiten autonom. Einige der Benutzer sind anwesend und nutzen die graphische Benutzerschnittstelle. Andere Benutzer, die kein eigenes COSMA-Programm haben, können immer noch über elektronische Post erreicht werden.

Für die Terminplanung und das Kalendermanagement sehen wir einen kooperierenden Agenten vor, einen *cooperative schedule management agent*, kurz COSMA. COSMA übernimmt Aufgaben einer Sekretariatskraft, indem es mit Dialogpartnern verhandelt und den individuellen Terminkalender des Benutzers führt. Mit der Einwilligung des Benutzers setzt COSMA sich mit anderen Menschen und COSMA-Instanzen in Verbindung, um Terminanfragen und damit verbundene Absprachen zu erledigen.

Das folgende Szenario wurde als Grundlage der Arbeiten in COSMA definiert. Das COSMA-System besteht aus

- einem natürlichsprachlichen Dialogsystem
- einem Terminplanungssystem (Kalenderverwaltung) als Backend
- einer Schnittstelle zu Standardsystemen für elektronische Post
- einer graphischen Benutzerschnittstelle

In einem elektronischen Netz sind zwei COSMA-Instanzen verbunden mit einem menschlichen Benutzer (siehe Abbildung 1). Die Kommunikation zwischen den drei Teilnehmern erfolgt über elektronische Post. COSMA macht anfangs keine Annahmen darüber, ob ein Adressat ein menschlicher oder ein maschineller Partner ist. Daher wird die Möglichkeit vorgesehen, sowohl natürlichsprachlichen Text als auch interne Repräsentationen des Kalenderverwaltungssystems zu übermitteln, so daß der jeweilige Partner sich die für ihn geeignete Fassung aussuchen kann.

Zunächst wird eine Basisfunktionalität entwickelt, von der aus das Leistungsspektrum des Systems stufenweise erweitert wird:

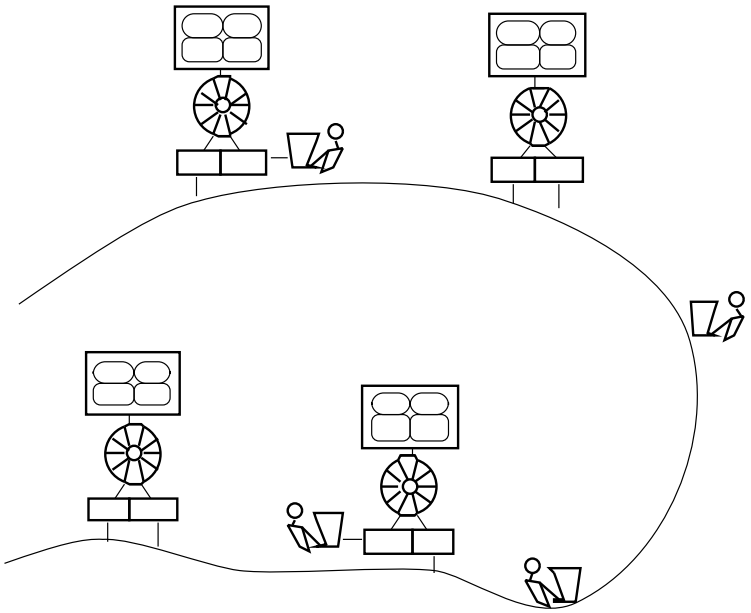

Abbildung 1: Eine dezentrale COSMA-Installation

- Die linguistischen Fähigkeiten werden zunehmen, mit denen die Basisfunktionalität erreicht wird.
- Die Fähigkeit des Systems, auf unbekannte oder fehlerhafte Eingaben adäquat zu reagieren, wird zunehmen.

Die in diesem Szenario führbaren Dialoge bauen auf einer in der ersten Projektphase zu erarbeitenden *Basisfunktionalität* auf, die in sechs elementaren Handlungen mit Terminen besteht.

Vorschlag: Einleitender Sprechakt des initiierenden Dialogpartners. Beispiel: *Ich möchte Sie am Freitag, dem 1. Juni nachmittags treffen.*

Akzeptieren: Positive Reaktion auf einen Vorschlag. Beispiel: *Ich bin mit dem Termin einverstanden.*

Zurückweisen: Negative Reaktion auf einen Vorschlag ohne Gegenvorschlag. Beispiel: *Ich bin an diesem Tag verhindert.*

Verändern: Negative Reaktion auf einen Vorschlag mit Gegenvorschlag. Beispiel: *Ich schlage anstelle von Freitag den Donnerstag, den 31. Mai vor.*

Verfeinern: Positive Reaktion auf einen Vorschlag mit Präzisierung der Termindaten. Beispiel: *Ich komme um 15 Uhr.*

Bestätigen: Beendigung des Dialogs durch den initiierenden Partner durch Zusammenfassung. Beispiel: *Alle Teilnehmer haben den folgenden Termin zugesagt: ...*

Die Schnittstelle zum Email-System erlaubt die Suche nach solchen Nachrichten im elektronischen Briefkasten des Bemutzers, die mit Terminvereinbarungen zu tun haben. Diese werden vom COSMA-System verarbeitet.

Die Integration menschlicher Dialogpartner in die Dialoge erfordert die Fähigkeit von COSMA, unterschiedliche sprachliche Formulierungen für dieselbe Basisfunktion zu verstehen. Die folgenden wesentlichen Erweiterungen werden integriert:

Anaphern: Personal- und Possessivpronomina, Rückbezug auf vorher genannte Zeitpunkte (z.B. *dann habe ich genügend Zeit, später wäre mir lieber*).

Partikeln: Kooperative sprachliche Äußerungen enthalten stets Partikeln wie *leider*, *natürlich*, *gerne*, mit denen die Einstellung des Sprechers kommuniziert wird.

Satzzahl: Die Verarbeitung von mehr als einem Satz pro Äußerung ist unabdingbar. Methoden des "shallow parsing" aus PARADICE werden es erlauben, auch aus längeren Texten (etwa mit komplexen Begründungen, warum der vorgeschlagene Termin nicht paßt) die relevante Information zu extrahieren.

Ellipsen: Einer Empfehlung des wissenschaftlichen Beirats des DFKI folgend wird COSMA eingeschränkte Klassen von elliptischen (d.h. verkürzten) Äußerungen verarbeiten können. Z.B. soll COSMA auf die Frage, ob man sich in der laufenden Woche treffen könne, folgende Antwort verstehen: *Lieber in der nächsten.*

Behandlung von Mehrdeutigkeiten: Das häufige Auftreten von Präpositionalphrasen macht erweiterte Mechanismen zu deren Zuordnung notwendig. Z.B. ist zu entscheiden, ob die Aufforderung *Streiche den Termin am 2. Mai* am 2. Mai zu erledigen ist (für irgend einen Termin) oder sofort (für den Termin am 2. Mai).

COSMA wird flexibel auf unvorhergesehene Ereignisse reagieren. Beim Auftreten unbekannter Wörter oder bei fehlerhaftem Gebrauch bekannter Wörter versucht COSMA zunächst, das Problem intern zu lösen. Die Systemarchitektur sieht Fehlermeldungsprotokolle vor, die es einzelnen Moduln (etwa der Flexionsanalyse oder dem Lexikon) erlauben, das entstandene Problem der Verarbeitungskontrolle bekannt zu machen. Sollte sich herausstellen, daß das entdeckte Problem offenbar ohne Einfluß auf die terminrelevanten Teile der Nachricht ist, kann COSMA das Problem vorerst ignorieren. Andernfalls kann COSMA aufgrund solcher interner Fehlermeldungen Klärungsdialoge einleiten, in denen der Benutzer oder alternativ der Absender des problematischen Texts um Hilfe gebeten werden.

2 Voraussetzungen zur Durchführung des Vorhabens

Bereits im Rahmen des Vorghängerprojekts DISCO wurde an Prototypsystemen in der Domäne der Terminvereinbarungen gearbeitet [Uszkoreit *et al.*, 1994]. Die Ergebnisse stellen eine ausgezeichnete Ausgangsbasis für das Projekt COSMA dar.

Bei der Lösung der Probleme, die sich bei der Verbindung von natürlichsprachlichen Komponenten und der Anwendungssoftware ergeben, wurden zwei unterschiedliche Ansätze gewählt. Beim ersten Ansatz wurden sämtliche Fähigkeiten des autonomen Agenten von der Anwendungssoftware bereitgestellt. Vorteil dieses Ansatzes ist, daß der Konstrukteur der Anwendung diese Funktionalität vollständig unter Kontrolle hat. Beim zweiten Ansatz beruhte die Funktionalität des autonomen Agenten auf den Repräsentationen der natürlichsprachlichen Komponenten. Dadurch wird die Qualität der natürlichsprachlichen Ein- und Ausgaben prinzipiell sehr verbessert. In COSMA soll versucht werden, die Vorteile beider Ansätze zu nutzen, indem wir eine Anwendungsschnittstelle einführen, die der Notwendigkeit Rechnung trägt, die Dialogkomponente mit einem Modell der Anwendungsaufgabe zu versehen. Die folgende, genauere Betrachtung wird diesen Ansatz verdeutlichen.

2.1 Der erste COSMA-Prototyp

Der erste Systemprototyp [Neumann, 1993] wurde im September 1992 vorgeführt. Die Benutzereingabe erfolgte durch elektronische Post und durch eine graphische Benutzerschnittstelle. Jede COSMA-Instanz fungierte als persönlicher Sekretär, der die folgenden Dienste leistete:

- Speicherung und Organisation eines persönlichen Terminkalenders,

- Graphische Darstellung und Veränderung der Terminangaben,
- Natürlichsprachliches Verstehen und Generieren zur Kommunikation mit anderen Teilnehmern über elektronische Post.

Das System bestand erstens aus dem Prototyp eines Terminplaners (entwickelt im DFKI-Projekt AKA-Mod, BMFT FKZ ITW 9104), der die Kalenderdatenbank führte, zeitliches Schließen erlaubte und die Kommunikation mit anderen Teilnehmern anstieß; zweitens einer graphischen Benutzerschnittstelle, die den Kalenderstatus überwachte und die Maus- und Menü-Eingaben für die Absprache neuer Termine ermöglichte; drittens aus einer Version des in DISCO entwickelten natürlichsprachlichen Systems, das, angereichert durch ein Paket anwendungsspezifischer Module, eine eingeschränkte Dialogfähigkeit zur Verfügung stellte.

Der folgende kurze Ablauf ist ein Beispiel für die vom diesem Prototypen tatsächlich ausgeführten Dialoge. In diesen Beispielen kommuniziert Benutzer A über die graphische Schnittstelle mit einer COSMA-Instanz. Benutzer B hat kein COSMA und kommuniziert über elektronische Post. COSMA C hat Anweisung, autonom zu arbeiten und eigenverantwortlich Terminabsprachen für seinen Benutzer zu treffen. Zu beachten ist, daß jede COSMA-Instanz Mitteilungen sowohl in natürlichsprachlicher Form – für den menschlichen Leser – und in seiner internen Repräsentationssprache – für die effziente Verarbeitung durch andere COSMA-Instanzen – verschickt.

- Über die graphische Schnittstelle gibt A die vollständigen Angaben für einen Termin ein: [*Art:* Besprechung; *Teilnehmer:* Tick, Trick und Track; *Zeit:* Dienstag um 10:00 Uhr; *Ort:* DFKI; *Thema:* COSMA-Demo]
- Das COSMA-System von A schickt den Terminwunsch an B und C
- B liest den Text und sendet an COSMA A: "Ich bin mit dem Termin einverstanden."
- COSMA C liest die interne Repräsentation und schickt sein Einverständnis an COSMA A
- COSMA A interpretiert den Text von B und die interne Repräsentation von C und schickt eine Terminbestätigung an B und C.

Im ersten Beispiel bittet Benutzer A um einen Termin, indem er ein Formular der graphischen Schnittstelle ausfüllt. Die elektronische Nachricht, in der die anderen Teilnehmer um Teilnahme gebeten werden, wird automatisch vom COSMA des Benutzers A generiert und an B und C verschickt. B, der die hereinkommende natürlichsprachliche Nachricht über elektronische Post selbst liest, antwortet in normalem Deutsch. Währenddessen stellt COSMA von C fest, daß der vorgeschlagene Termin in der Tat frei ist, akzeptiert den Vorschlag im Auftrag von C und zeigt ihn als möglichen Termin auf der graphischen Schnittstelle an. Die graphische Schnittstelle zu COSMA A wird jetzt auf den neuesten Stand gebracht und zeigt einen bestätigten Termin; die anderen Teilnehmer erhalten Bestätigungsmeldungen. Falls diese Teilnehmer auch über ein COSMA verfügen, werden auch ihre Displays aktualisiert.

Die Schnittstelle zwischen natürlichsprachlichem System und Terminplaner beruhte auf zwei Konzepten: Schemata für Kommunikationshandlungen und Übersetzungsprozessen mithilfe von Compilertechnologie. Das Konzept der Schemata für Kommunikationshandlungen führte zu einem erfolgreich funktionierenden Prototyp. Die Robustheit bei der Abbildung zwischen domänenabhängigen Dialoghandlungen und anwendungsabhängigen Kommunikationsschemata könnte jedoch nur durch den Einsatz von Inferenzfähigkeit erreicht werden.

2.2 Der zweite COSMA-Prototyp

Um die Robustheit der Abbildung zwischen Dialog und Anwendung zu verbessern, schloß der im September 1993 fertiggestellte Kalendermanagement-Prototyp in der Sprachverarbeitung das Agentenkonzept ein. Dies bedeutet, daß die generelle Planung, Planerkennung, Planausführung und Inferenzfähigkeit zumindest in rudimentärer Form bereitgestellt wurden. Diese Fähigkeiten

wurden sodann für die Verarbeitung deklarativer Dialogrepräsentationen verwendet. Sie wurden auch bei der Verarbeitung deklarativer Repräsentationen in der Kalenderdomäne und der Modellierung des Kalenderanwendungsprogramms benutzt. Hierfür stand sowohl dialog- als auch anwendungsbezogenes Wissen zur Verfügung, z.B. Taxonomien, Handlungsdefinitionen und Inferenzregeln.

Der Planer/Planerkenner und die Ontologie, auf die er zurückgreift, realisierten das Agentenkonzept. Die Anwendungsmodellierung gehörte zum Dialogsystem; die modellierten Anwendungen existierten außerhalb davon. Die genaue Grenze zwischen Domänenmodell und Anwendungsprogramm wurde flexibel gehalten. So konnte z.B. ein "Adressbuch"-Modul vollständig innerhalb des Anwendungsmodells angeordnet werden. Die Kalenderdatenbank lag als externe Datei vor. Hierbei ist von grundlegender Bedeutung, daß jeder dieser Bestandteile innerhalb der Anwendungsschnittstelle modelliert war und über Operationen verfügte, die, genau wie Sprechakte, für den Planer jederzeit verfügbar waren.

Das System, das im September 1993 vorgestellt wurde, soll hinsichtlich der Flexibilität und der Anwendungsfunktionalität stark verbessert werden. Im besonderen wird die Struktur der möglichen Dialoge weniger rigide sein, da sie reaktiv nach jedem Schritt geplant und nicht durch den Programmierer von vornherein festgelegt wird.

3 Planung und Ablauf des Vorhabens

3.1 Planung des Vorhabens

Das Forschungsvorhaben COSMA soll im Forschungsbereich *Computerlinguistik* durchgeführt werden und am 1. April 1994 mit einer Laufzeit von drei Jahren beginnen. Für das Projekt waren drei wissenschaftliche Mitarbeiter, darunter ein leitender Wissenschaftler, vorgesehen.

Inhaltlich soll das Projekt eng mit dem zeitgleich durchgeführten BMBF-Projekt PARADICE zusammenarbeiten: der Nachweis der Zuverlässigkeit und Funktionalität eines sprachverarbeitenden Systems wie der Dialog-Kernmaschine, die in PARADICE entwickelt wird, liegt in seiner Anwendung. COSMA wird von PARADICE wesentliche Teile der linguistischen Kernmaschine übernehmen.

Abbildung 3.1 zeigt die Arbeitspakete und die entsprechenden Balkendiagramme. Die Ergebnisse des Projekts sollen in zwei Meilensteinen demonstriert werden. Der erste Meilenstein liegt im Frühjahr 1995, der zweite im Herbst 1996.

3.2 Meilenstein I – 30. März 1995

Auf der Frühjahrssitzung 1995 des Wissenschaftlichen Beirats und des Gesellschafterbeirats wird ein natürlichsprachliches System für Terminvereinbarung demonstriert, das die Basisfunktionalität aufweist, erste Mechanismen zur Systemrobustheit enthält und Methoden zur interaktiven Fehlerkorrektur anwendet. Dies wird im folgenden präzisiert.

Das System kann gemäß seiner Parameterisierung den erkannten Fehler interaktiv beheben (Klärungsdialog) oder den Benutzer von dem entstandenen Problem in Kenntnis setzen, ohne selbst einen Lösungsversuch zu unternehmen.

Der Meilenstein umfaßt Mechanismen für die Korrektur von zwei Typen von Problemen:

Analysefehlschlag durch das System: Die in PARADICE entwickelten Methoden zur Fehlerlokalisierung werden in COSMA eingesetzt. Die Verwendung unbekannter Wörter kann somit zu Rückfragen führen. Folgendes Dialogbeispiel wird vorgeführt werden (C ist ein COSMA, B dessen Benutzer, A ein anderer menschlicher Teilnehmer):

C an A: Ich möchte Sie am Montag, dem 24. 6. 1994 treffen.

A: an C: Bedauerlicherweise kann ich nicht kommen.

C *kennt "bedauerlicherweise" nicht, kann aber die Benutzeräußerung korrekt als Absage interpretieren.*

COSMA: Balkenplan

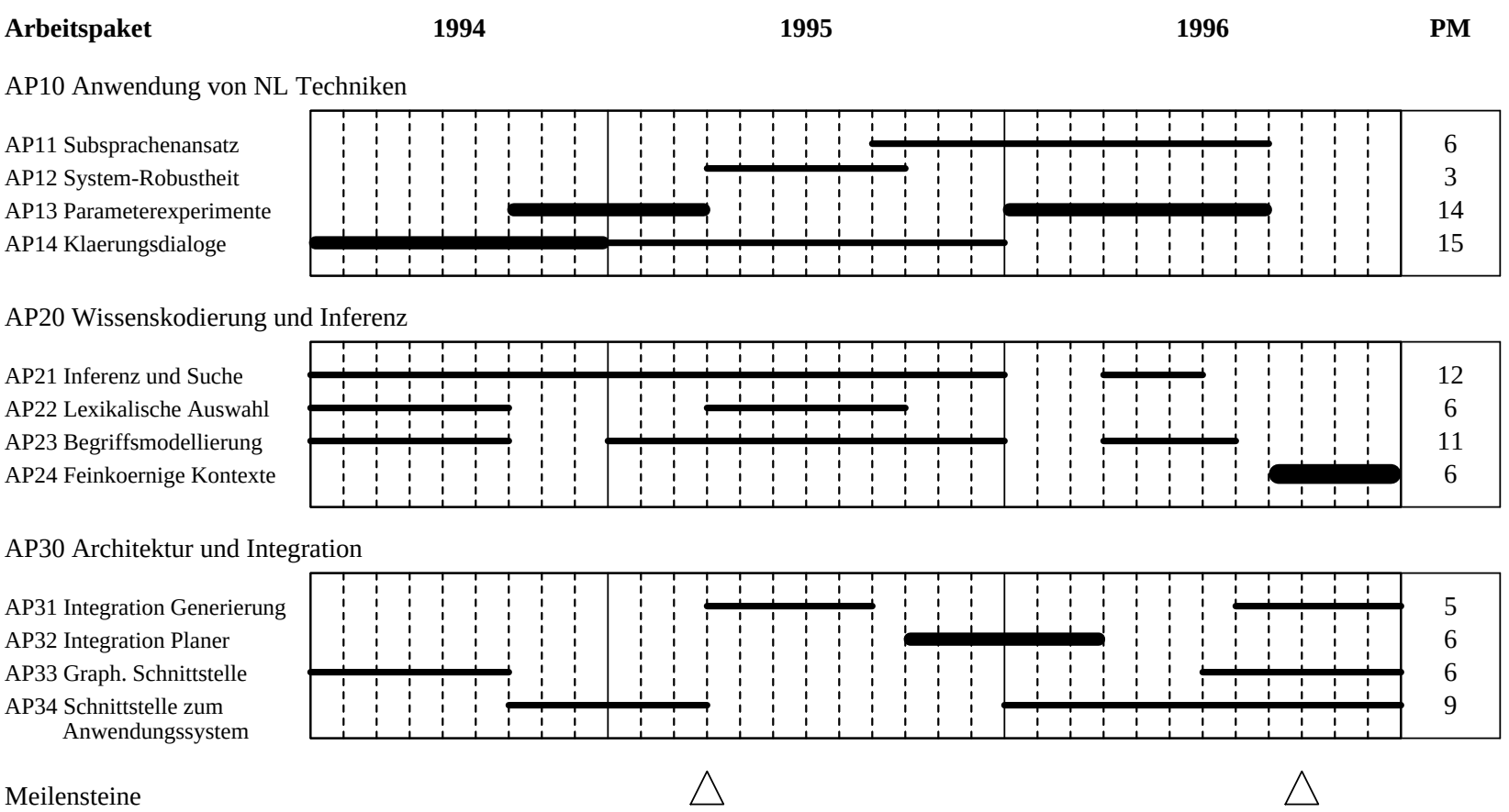

Abbildung 2: Balkendiagramm für COSMA

C an B: Was heißt 'bedauerlicherweise'?

B an C: 'Bedauerlicherweise' ist dasselbe wie 'leider'.

C *findet "leider" im Lexikon und macht entsprechendes Update.*

Falsche Terminbeschreibungen durch den menschlichen Benutzer: Bei der Einbindung menschlicher Dialogpartner ist mit inkonsistenten Terminbeschreibungen zu rechnen. Für den exemplarischen Fall einer Inkonsistenz zwischen Wochentag und Monatstag wird gezeigt, wie das System versucht, durch einen Klärungsdialog Konsistenz herzustellen. Die darauf folgende Korrektur durch den Benutzer setzt den vorherigen Dialog fort. Zum Beispiel:

C an A: Am 16. Juni will ich Dich treffen.

A an C: Ich kann an dem Freitag kommen.

C an A: Der 16.6. ist ein Donnerstag.

A an C: Ja, ich bin mit Donnerstag, 16.6. einverstanden.

3.3 Meilenstein II – 30. September 1996

Auf der Herbstsitzung 1996 des Wissenschaftlichen Beirats und des Gesellschafterbeirats werden die wesentlichen Projektergebnisse von COSMA demonstriert. Der Meilenstein umfaßt ein natürlichsprachliches System für Terminvereinbarungen, das robust ist und aktiv an der Klärung von Verstehensproblemen teilnimmt. Außerdem ist das System in der Lage, Unterbrechungen oder Revisionen von Äußerungen adäquat zu verarbeiten. Diese Leistungen werden im folgenden präzisiert.

- Die Mechanismen zur interaktiven Fehlerkorrektur umfassen die gesamte Klasse der vorkommenden inkonsistenten Terminbeschreibungen gemäß Abschnitt 7.7. Die im Projekt PARADICE erarbeiteten Mechanismen zur Systemrobustheit sind in COSMA integriert.
- Die linguistische Abdeckung der verarbeitbaren Äußerungen umfaßt anaphorische Ausdrücke, Partikeln wie *leider, natürlich, gerne*, Mehrsatzäußerungen, Ellipsen (*Lieber am 10.*) sowie die Behandlung bestimmter Mehrdeutigkeiten bei der Anbindung von Präpositionalphrasen – z.B. ist zu entscheiden, obb die Aufforderung *Streiche den Termin am 2. Mai* am 2. Mai zu erledigen ist (für irgendeinen Termin) oder sofort (für den Termin am 2. Mai.
- Der Benutzer kann mit seinem COSMA-System wahlweise über elektronische Post oder über eine graphische Benutzerschnittstelle interagieren.
- Erweiterte Funktionalität bei Terminvereinbarungen umfaßt Benutzerpräferenzen.

Diese Funktionalität wird auf zwei Arten demonstriert. Zum einen werden die beschriebenen Leistungen in Bezug auf Abdeckung, Systemrobustheit und interaktive Klärung von Problemen bzw. Fehlern anhand typischer Dialogsequenzen im Rahmen des beschriebenen Szenarios vorgeführt. Z.B. werden Beispieldialoge folgender Art geführt werden können, die das Finden von Kompromissen veranschaulichen. Die Teilnehmer sind Frau Dr. Mayer, Herr Schmidt und das COSMA von Herrn Dr. Weber (CW).

CW an Mayer, Schmidt: Dr. Weber möchte Sie am übernächsten Dienstag um 14 Uhr in seinem Hauptbüro in Saabrücken treffen, um die Projektvorlage TIMAC vorzubereiten. Das Treffen wird drei Stunden dauern.

Schmidt an CW: Könnten wir den Termin bitte um zwei Stunden verschieben? Mein Flug kommt erst um 13:03 in Frankfurt an.

Mayer an CW: Ich kann leider nur bis 16 Uhr bleiben, denn am gleichen Abend habe ich einen Geschäftstermin in Frankfurt. Ich würde ein Treffen um 13 Uhr bevorzugen.

CW an Schmidt: Leider kann Frau Dr. Mayer zu der Zeit nicht da sein. Können Sie den Flug umbuchen?

Schmidt an CW: Ich könnte frühestens um 15 h da sein.

CW an Mayer: Herr Schmidt kann erst um 15 Uhr da sein. Bitte versuchen Sie, das Treffen um diese Zeit zu ermöglichen.

Mayer an CW: Ich habe meinen Termin in Frankfurt verschieben können. Ich werde um 15 Uhr kommen.

CW an Mayer, Schmidt: Das Treffen findet um 15 Uhr in Saarbrücken statt.

Zum andern werden die Ergebnisse des zweiten Parameterexperiments veranschaulicht, indem ein erweitertes Spektrum an Möglichkeiten vorgestellt wird, das COSMA-System während der Arbeit zu *unterbrechen*.

So kann der Benutzer seinen Terminwunsch, den er über die graphische Schnittstelle mitgeteilt hat, revidieren, und das System wird mit der revidierten Fassung weiterarbeiten. Es handelt sich um eine lokale Unterbrechung, falls COSMA noch keine Nachricht aufgrund der ersten Spezifikation verschickt hat. Andernfalls verschickt COSMA eine zweite Nachricht, die als Korrektur der ersten ausgewiesen ist.

Die Unterbrechbarkeit des Systems kann im Unterschied zum Abbruch und darauffolgenden Neubeginn einer Verarbeitung dazu führen, daß nicht alle Angaben erneut verarbeitet werden müssen. Ändert der Benutzer lediglich das Datum vom 14. auf den 15. des Monats, bleiben Sprechakt (Vorschlag), Teilnehmer, Uhrzeit, Ort und Thema des Treffens konstant.

3.4 Ablauf des Vorhabens

3.4.1 Personalia

Das Projekt wurde plangemäß im Zeitraum vom 1. April 1994 bis zum 31. Dezember 1996 erfolgreich durchgeführt und durch den wissenschaftlichen Beirat des DFKI im März 1997 abschließend mit einem ausgezeichneten Ergebnis begutachtet.

Ab 1. April 1994 nahmen die wissenschaftlichen Mitarbeiter Abdel Kader Diagne (Architektur und Kontrollfluß) und Dr. Elizabeth Hinkelman (Dialogverarbeitung) sowie als leitender Wissenschaftler Dr. Stephan Busemann (Generierung) ihre Arbeit auf. Weitere Mitarbeit erfolgte durch Günter Neumann (Architektur) und Stephen P. Spackman (Schnittstelle zum Agentensystem) infolge der engen Kooperation mit dem Schwesterprojekt PARADICE. Ab Juli 1995 wurde Iris Merget auf Teilzeitbasis (10 Std./Woche) in COSMA eingestellt (Marktstudie und Systemoberfläche). Am 25. Februar 1996 verließ Dr. Elizabeth Hinkelman das DFKI. Luca Dini (Scuola Normale Superiore, Pisa, Italy) leistete als freier Mitarbeiter im Jahr 1996 wesentliche Beiträge durch die Komponente zur semantischen Interpretation und zur Fehlschlagsbehandlung. Zum 1. Juni 1996 wurde Thierry Declerck eingestellt (Universität Stuttgart). Seine Hauptaufgabe war Grammatikentwicklung für das Informationsextraktionssytem smes und die Evaluation der linguistischen Abdeckung.

Als wissenschaftliche Hilfskräfte leisteten Judith Klein (Korpus-Annotation und Evaluation), Ralf Nieser (Impementation der Systemoberfläche), Michael Wein (Implementation im Bereich Generierung) und Sven Schmeier (Erweiterung des Agentensystems PASHA-II) wichtige Beiträge zum Projekterfolg.

3.4.2 Inhaltliche Entwicklungen

In der Anfangsphase des Projekts wurde zunächst eine tiefe Analyse eingesetzt, um die Eingabe zu parsen. Durch die übernahme der Kernmaschinerie aus DISCO stand eine HPSG-Grammatik des Deutschen und ein Parser zur Verfügung. Aber im Laufe der Grammatikentwicklung wurde immer deutlicher, daß dieser Ansatz für die Anwendungsdomäne nicht optimal geeignet ist. Eine tiefe Analyse liefert nur dann Ergebnisse, wenn die gesamte Eingabe erfolgreich geparst wurde. Dies ist problematisch in zweierlei Hinsicht:

- Der Parser scheitert an unvollständigen oder fehlerhaften Eingaben.
- Es wird sehr viel Zeit gebraucht, um Information zu parsen, die eigentlich nicht relevant ist.

Da die natürlichsprachliche Eingabe via E-Mail geschieht, ist es nicht ungewöhnlich - ganz im Gegenteil -, unvollständige oder fehlerhafte Texte als Eingabe vorzufinden. Das Gesamtsystem würde daran scheitern, obwohl in vielen Fällen die relevante Information aus dem Text extrahiert werden kann. Das heißt konkret, daß der Parser für die Terminvereinbarungsdomäne einen hohen Grad an Robustheit aufweisen soll.

Schließlich stellt die Ausgaberepräsentation der verwendeten HPSG-Grammatik starke Anforderungen an Anwendungssysteme. Es stellte sich heraus, daß die eingesetzten Agentensysteme nicht ohne hohen Zusatzaufwand in der Lage waren, die linguistisch motivierten Semantikkonstruktionen zu interpretieren.

Zudem soll ein Dialogsystem effizient sein, um eine erfolgreiche Kommunikation zu ermöglichen. Die Analyse der für die Terminvereinbarung irrelevanten Textteile verlangsamt unnötigerweise den Kommunikationsprozeß.

Eingehende natürlichsprachliche Nachrichten müssen zunächst vom Gesamtsystem analysiert werden, wobei COSMA nur an den relevanten Teilen der Eingabe interessiert ist. Diese relevanten Fragmente bestehen aus temporalen Ausdrücken (PPs und NPs), aber auch aus typischen Verben und Substantiva, die bei Terminvereinbarungen eine Rolle spielen. Auch gewisse Partikeln müssen berücksichtigt werden, wenn sie für die Bestimmung von Dialogklassen von Bedeutung sind.

Daher hat sich das Projekt COSMA für eine andere Strategie entschieden: Für die natürlichsprachliche Analyse wird das Shallow-Parsing-System smes eingesetzt, das im Rahmen des Projekts PARADICE entwickelt wurde [Neumann *et al.*, 1997]. Das System bietet beides, Robustheit und die Möglichkeit der Extraktion relevanter Textfragmente. Für die semantische Analyse und eine geeignete Schnittstelle zu Agentensystemen wurde das System IMAS entwickelt. Beide Komponenten zusammen genommen stellen einen Ansatz zur Informations-Extraktion dar. Solche Verfahren werden in zunehmendem Maße für sprachtechnologische Anwendungen eingesetzt (vgl. Abschnitt 9).

4 Wissenschaftlich-Technischer Stand

Das breite Interesse an Kalendermanagement beruht zum Teil auf der breiten Akzeptanz von Softwareprodukten zu *personal organizing* und Zeitmanagement und zu einem anderen Teil auch auf der relativen Leichtigkeit, mit der eine minimale Funktionalität recht schnell implementiert werden kann. Bemerkenswert sind hier die implementierten Forschungssysteme von Mattern [Mattern und Sturm, 1989], Mitchell [Dent *et al.*, 1992] und Steiner/Haugeneder [Lux *et al.*, 1992; Lux, 1992].

Matterns System zur Terminvereinbarung ist eine verteilte Softwareanwendung, die nicht mit KI-Techniken arbeitet. Der Schwerpunkt liegt bei der Feinabstimmung der Benutzerschnittstelle, wobei besonderer Wert auf eine einfache Repräsentation wiederkehrender Zeitbeschränkungen, persönlicher Prioritäten und skalar dargestellter Präferenzen gelegt wird. Das System von Mitchell wiederum ist eine experimentelle Grundlage für ein lernendes System. Es findet Verwendung beim Vergleich von Methoden aus den Bereichen neuronale Netze und statistischer Faktorenanalyse. Im krassen Gegensatz zum System von Mattern wird hier die Benutzerschnittstelle auf die Ausgabe einer einzigen, neuen Emacs-Textverarbeitungsfunktion reduziert. Mithilfe dieser Funktion werden Einträge im on-line Terminkalender des Benutzers vorgenommen, wobei das System gelernte intelligente Standardvoreinstellungen für jedes Feld vorgibt.

Das System von Steiner und Haugeneder wurde im DFKI-Projekt KIK-TEAMWARE gemeinsam mit der Siemens AG entwickelt. Da der erste COSMA-Planer (entworfen vom DFKI-Projekt AKA-MOD) seine Wurzeln in dieser Arbeit hat, ist das System in seiner Funktionsweise COSMA sehr ähnlich. Das System plant und ändert ebenfalls Termine für mehrere Teilnehmer, verfügt aber über keinerlei natürlichsprachliche Fähigkeiten. Die Kernhypothese dieser Arbeit ist jedoch, daß kooperatives Verhalten zwischen Agenten hier unabhängig vom Kommunikationsmedium ist.

Die Kommunikation zwischen den Agenten ist nur für die Maschine lesbar und die Planungskomponente wird von den Eigenschaften des Kommunikationskanals vollständig isoliert. Obwohl der Begriff *Sprechakt* gelegentlich für die angewendeten abstrakten Protokollelemente verwendet wird, setzt dieser Ansatz eine starke Modularität voraus, die den direkten Einbau von natürlichsprachlichen Einrichtungen für die Dialogverwaltung verhindern würde. In seinem ersten Prototyp eines Kalendersystems hat das Projekt DISCO diese Architektur erprobt.

Der Einsatz einer Agenten-Architektur für das Kalendermanagement konnte bereits in den letzten Phasen des DISCO-Projekts (BMFT-FKZ ITW 9002 0) untersucht werden. Bei der Lösung der Probleme, die sich bei der Verbindung von natürlichsprachlichen Komponenten und der Anwendungssoftware ergeben, wurden zwei unterschiedliche Ansätze gewählt. Beim ersten Ansatz wurden sämtliche Fähigkeiten des autonomen Agenten von der Anwendungssoftware bereitgestellt. Vorteil dieses Ansatzes ist, daß der Konstrukteur der Anwendung diese Funktionalität vollständig unter Kontrolle hat. Beim zweiten Ansatz beruhte die Funktionalität des autonomen Agenten auf den Repräsentationen der natürlichsprachlichen Komponenten. Dadurch wird die Qualität der natürlichsprachlichen Ein- und Ausgaben prinzipiell sehr verbessert. In COSMA soll versucht werden, die Vorteile beider Ansätze zu nutzen.

Ein dem ersten Ansatz entsprechendes Agentensystem wurde im Rahmen des BMBF-Projekts CoMMA-MAPS als Diplomarbeit entwickelt und auf einer internationalen Fachkonferenz publiziert [Schmeier und Schupeta, 1996]. In dieser Arbeit wurde das Agentensystem PASHA-II zur verteilten Terminvereinbarung mit mehreren Teilnehmern entwickelt. Die konkreten Ziele waren:

- Entwurf eines Terminvereinbarungszenarios als Multiagentensystem.
- Implementierung der Agenten unter Zugrundelegung des InteRRaP-Agentenmodells, welches im Projekt CoMMA-MAPS unter Leitung von Prof. Siekmann entwickelt wurde.
- Testen der neuen Programmiersprache Oz [Smolka, 1995], entwickelt am DFKI unter Leitung von Prof. Smolka.
- Entwicklung von Kommunikationsschnittstellen zwischen den Agenten selbst und zwischen Agenten und Menschen.
- Konzeptualisierung von gebräuchlichen Interaktionsstrategien, die zwischen Menschen im Rahmen dieses Problems bestehen.
- Entwicklung einer formalen Sprache, um diese Strategien deklarativ zu spezifizieren (Konditionalplanung).

Eine detailliertere Beschreibung der Ausgangssituation zu PASHA-II und der daran anschließenden Erweiterungen in COSMA zu PASHA-III wird in Abschnitt 7.6 gegeben.

5 Verwendete Fachliteratur

Die benutzte Fachliteratur entstammt verschiedenen Gebieten der Informatik, der Computerlinguistik und der KI. Für die Systemarchitektur waren objektorientierte Paradigmen von zentraler Bedeutung [Booch, 1994] sowie Workflow-Organisation [Malone und Crowston, 1991].

Die Sprachverarbeitungsaspekte gliedern sich auf in grammatische Analyse, Semantikverarbeitung und Sprechakterkennung sowie Generierung. Hinsichtlich der grammatischen Analyse wurde durch den Einsatz des linguistischen Kernsystems [Uszkoreit *et al.*, 1994] auf eigene Vorarbeiten zurückgegriffen (u.a. *TDL* [Krieger und Schäfer, 1994]). Die Dialogakterkennung basierte auf der Theorie von Hinkelman und Traum zum sogenannten *Grounding* von Sprechakten [Traum und Hinkelman, 1992]. Der spätere Einsatz des Systems smes beruhte auf Erkenntnissen die Nützlichkeit von Informationsextraktion betreffend [Grishman und Sundheim, 1996; Appelt *et al.*, 1993]. Die Evaluation von smes in COSMA beruhte auf diagnostischen Werkzeugen, an deren Entwicklung das DFKI maßgeblich beteiligt war: DITO [Nerbonne *et al.*, 1991] und

tsdb [Lehmann *et al.*, 1996]. Die Entwicklung der Generierungskomponente wurde durch klassische Verfahren bei Produktionensystemen beeinflußt (für einen Überblick siehe [Davis und King, 1977]).

Die Agentensysteme wurden auf der Grundlage des am DFKI entwickelten Architekturmodells InteRRap [Müller und Pischel, 1993] entwickelt. Für ihre Implementation in DFKI Oz wurde auf [Smolka, 1995] zurückgegriffen.

Darüber hinaus waren Publikationen in den folgenden wissenschaftlichen Konferenzserien von besonderer Bedeutung für das Projekt:

- Conference of the Assocication for Computational Linguistics (ACL)
- Conference on Applied Natural Language Processing (ANLP)
- Conference on Computational Linguistics (COLING)
- Conference on Practical Applications of Multi-Agent Systems (PAMAS)
- European Conference for Artificial Intelligence (ECAI)
- International Joint Conference for Artificial Intelligence (IJCAI)
- International Natural Language Generation Workshop (IWNLG)
- International Pragmatics Conference

6 Zusammenarbeit mit anderen Stellen

Zusammenarbeit innerhalb der DFKI GmbH erfolgte insbesondere mit dem Projekt PARADICE. Während PARADICE auf der Basis der im Vorgängerprojekt DISCO erarbeiteten Grundlagen Kerntechnologie erarbeiten sollte, war es die Aufgabe von COSMA, diese Technologien im Rahmen eines realistischen Anwendungsszenarios zu erproben. In der ersten Projektphase übernahm COSMA von PARADICE eine Weiterentwicklung der HPSG-basierten Kernmaschine. In der zweiten Projektphase wurde diese durch das Informationsextraktionssystem smes ersetzt, das ebenfalls im Rahmen von PARADICE entwickelt wurde.

Agentensysteme wurden aus den DFKI-Projekten CoMMA-Plat (Siemens AG) und CoMMA-MAPS (BMBF ITW 9500) übernommen.

Umgekehrt wurde die in COSMA entwickelte Software anderen Projekten zur Verfügung gestellt. Das Rahmensystem zur Systemarchitektur wird im BMBF-Projekt PARADIME (BMBF FKZ ITW-9704) eingesetzt. Das Generierungssystem TG/2 findet Verwendung im EU-Projekt TEMSIS.

In Kooperation mit M. Streit (Siemens AG) wurden Vorbereitungen für eine Kopplung von COSMA mit dem multimodalen Kalendermanager TALKY getroffen, der von M. Streit am DFKI entwickelt wird. Ziel ist eine auf gesprochener Sprache beruhende, verbesserte Eingabeschnittstelle für Terminplanungsaufträge.

7 Erzielte Ergebnisse

7.1 Systemarchitektur und Kontrolle

7.1.1 Überblick

Die Ausstattung von maschinellen Agenten mit natürlichsprachlichen Dialogfähigkeiten, insbesondere im Bereich der verteilten Terminverarbeitung über E-Mail, setzt hohe Anforderungen an ein Sprachverarbeitungssystem (NL-System) und die zugrundeliegende Systemarchitektur:

- Dialoginteraktionen in natürlicher Sprache zwischen Software-Agenten und Menschen sollen gewährleistet werden, ohne dabei die Integrität des Agenten zu verletzen. Dieser sollte das NL-System lediglich als Server benutzen (NL-Server); d.h. dessen Dienste nur bei Bedarf anfordern und nicht mit Einzelheiten der linguistischen Kontrollstrategien oder der Modularisierung des NL-Systems in Berührung kommen.
- Das System soll die Zuverlässigkeit der Dialogschritte gegenüber dem Agenten garantieren; z.B. die Fähigkeit, mögliche fehlerhafte oder inkonsistente Eingaben mittels Klärungsdialoge mit dem menschlichen Dialogpartner zu korrigieren.
- Eine robuste und flexible Schnittstelle zwischen Agent und NL-Server soll vorhanden sein.
- Bestehende Komponenten sollen leicht in das heterogene System integrierbar sein.
- Der NL-Server soll in der Lage sein, alternative Ergebnisse (zu aktuellen Äußerungen) auf Anfrage von Agenten zu berechnen (Backtracking).
- Das System soll nicht abhängig von spezifischen Dialogstrategien und Dialogssteuerungsmodellen sein.

In COSMA wurde das Architekturmodell COCONUTS (Cooperating Object-Oriented Control Units) entworfen und implementiert, um die oben genannten Forderungen an ein NL-Server zu genügen. COCONUTS ist eine Architektur für natürlichsprachliche Server, dessen wesentlichen Merkmale sich wie folgt beschreiben lassen:

- **Manager-basiert**: Komponenten, Koordinations- und Kontrollaktivitätenen im System werden spezifischen Kontrolleinheiten zugewiesen. Diese werden Manager genannt. Sie kooperieren miteinander auf Client-Server-Basis, um ein vorliegendes Problem (Anfrage eines Agenten) zu lösen.
- **Objekt-orientiert und generisch**: Manager sind als interagierende Objekte konzipiert. Die Architektur selbst ist als objekt-orientiertes Framework realisiert, bestehend aus einem anwendungsneutralen Kernel, auf dessen Grundlage domänenspezifische Erweiterungen und Spezialisierungen aufgebaut werden können.
- **Client-Server-Architektur**: Der Server stellt eine generische Server-Schnittstelle zur Verfügung, die aus einer deklarativen Spezifikationssprache und Kommunikationsprotokollen besteht, über die Agenten (Clients) mit dem Server flexibel kommunizieren können.
- **Virtuelle System Architektur**: Verschiedene Systeminstanzen können definiert werden und gleichzeitig aktiv sein. So können mehrere Anfragen von verschiedenen Clients können parallel bzw. asynchron von Server verarbeitet werden.
- **Flexibles Integrationsmodell**: Jede Komponente wird von einem Manager eingekapselt (encapsulation), der Implemetierungsdetails der Komponente versteckt und nur die relevante Schnittstelle nach außen präsentiert. Damit lassen sich Komponente als Server in das System integrieren. Diese können in verschiedenen Programmiersprachen implementiert sein und auf verschiedenen Machinen laufen.

In COSMA wurden zwei Systemvarianten entwickelt, die hinsichtlich der dialog-basierten Fehlerbehandlung (Failure Handling), des Schnittstellenkonzepts und der linguistischen Kernkomponenten (z.B. Kategorie, Anzahl und Funktionalität) teilweise verschiedene Ansätze realisierten. Das Gesamtarchitekturbild des ersten Ansatzes wird durch Abbildung 3 veranschaulicht. Abbildung 4 zeigt die aktuelle Systemarchitektur, die hinsichtlich Funktionalität, Flexibilität und Wiederverwendung eine Erweiterung und Verbesserung des ersten Ansatzes darstellt. Beide Systemvarianten basieren jedoch auf demselben manager-basierten und objekt-orientierten Architekturkonzept.

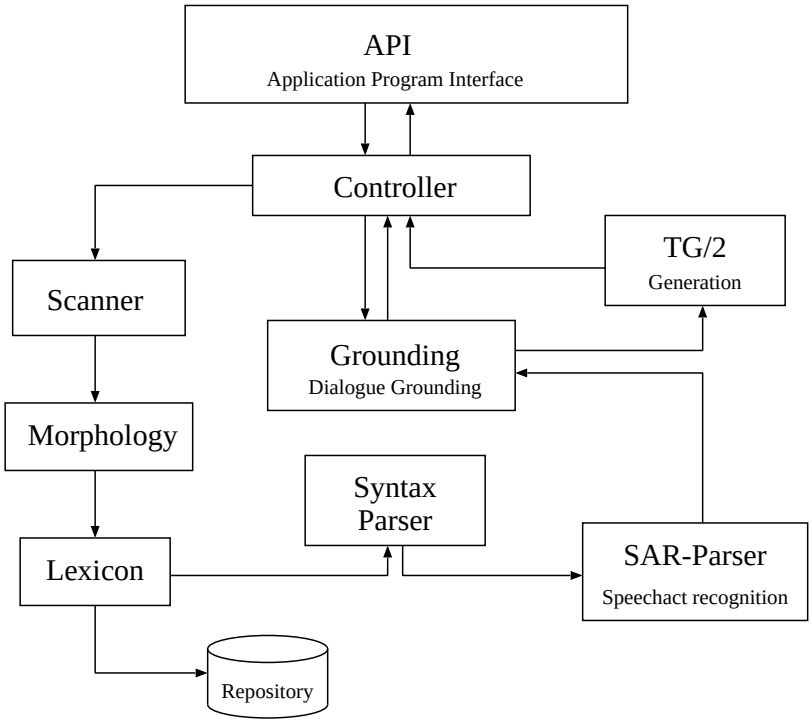

Abbildung 3: COSMA-Systemarchitektur auf der Grundlage des HPSG-basierten Kernsystems aus PARADICE und mit Dialog-Grounding-Komponente.

7.1.2 Eine manager-basierte Kontrolleinheit

Es wurde ein manager-basiertes Konzept eingeführt, das darauf basiert, alle Problemlösungsaktivit¨aten zu identifizieren und sie spezialisierten Kontrolleinheiten zuzuweisen. So eine Kontrolleinheit wird Manager genannt und läßt sich wie folgt charakterisieren:

- Ein Manager ist ein Objekt; d.h. er ist ein Instanz einer bestimmten Klasse, hat wohldefinierte Eigenschaften und ein wohldefiniertes Verhalten.
- Je nach Aufgabe und Bedarf können zusätzliche Manager definiert bzw. bestehende angepaßt werden.
- Manager sind als workflow-orientiertes Team organisiert und kooperieren untereinander nach einem Client-Server-Prinzip, wobei es von Anfang an klar festgelegt ist, welche Dienste ein Manager leistet bzw. welche Verantwortung er trägt (z.B. Datenmanagement, Report-Management, Parsing, semantische Analyse, etc.) und wie seine Schnittstellen definiert sind; d.h. wie seine Dienste in Anspruch genommen werden können.

Im folgenden werden die wichtigsten Manager-Klassen von COCONUTS für das COSMA-System in ihren wesentlichen Aspekten beschrieben.

Workflow-Manager: Das Problemlösungsverfahren in COSMA ist dialog-basiert und prozeßorientiert. Daher wurde ein spezieller Manager als Kontrollkomponente definiert, die alle Problemlösungsaktivitäten im system koordiniert. Die dadurch entstehenden Problemlösungsabläufe werden hier als *Workflow* bezeichnet. Dies schließt sowohl die dabei zu verarbeitenden Daten als auch dafür eingesetzten Resourcen ein. So koordiniert der Workflow-Manager Interdependenzen zwischen Problemlösungsaktivitäten (z.B. Parsing, semantische Analyse, Generierung) und regelt den entsprechenden Daten- und Informationsfluß gemässt einer wohldefinierten Prozedur. Dazu

gehört auch die Zuweisung von Aufgaben (Parsing, Generierung) an entsprechende Komponenten, die Gewährleistung der Zuverlässigkeit der erzeugten Ergebnisse; d.h. auch nötigensfalls die Plannung und Durchführung von Fehlerbehandlungsprozeduren. Das Verhalten und die Funktion des Workflow-Managers lassen sich durch folgende Arbeitsschritte veranschaulichen:

- *Identifizierung und Formulierung des Gesamtzieles des Workflow (workflow goal)* gemäß der zugrundeligenden Anfrage: Hier wird aus einer vorliegenden Anfrage, die in einer deklarativen Spezifikationssprache formuliert ist, das Prozeß-Ziel (Analyse oder Generierung) ermittelt und als interne Aufgabe formuliert. Ein Workflow in COSMA hat hauptsächlich zwei Gesamtziele (goals): zum einen die in einer E-Mailenthaltenen Information dem Agenten verfügbar zu machen; d.h. die Erzeugung eines semantischen Ausdrucks (IL-Ausdruck) aus einem vorliegenden Text; und zum anderen eine vom Agenten erzeugte Information für den menschlichen Dialogpartner bereitzustellen; d.h. die Generierung von natürlichsprachlichen Sätzen aus einem semantischen Ausdruck.

- *Dekomposition des Workflow-Zieles in untergeordneten Ziele (Subgoals)*: Beispielsweise ist ein smes-Template (getypte Merkmalstruktur als Ergebnis der Informationsextraktionskomponente smes) ein Subgoal bei der Analyse einer E-Mail, dessen Ziel die Erzeugung eines entsprechenden IL-Ausdrucks (Ergebnis der Semantische Analyse) ist.

- *Ermittlung und Allokation von benötigten Resourcen* für die Durchführung des Workflows: Hier werden den jeweiligen Komponenten bzw. assozierten Managern Teilaufgaben zugeordnet. Bevor ein Dialog verarbeitet wird, wird zuerst sichergestellt, daß alle benötigten Resourcen verfügbar sind. Dies wird durch das Konzept der virtuellen Systemarchitektur (VSA) gewährleitet (Abschnitt 7.1.3).

- *Erstellung eines Ausführungsplans* als Sequenz von (Teil)aufgaben, die zugleich das Workflow darstellen; z.B., die Aufgabensequenz *Text* $\rightarrow$ *Extrahierte_Information* $\rightarrow$ *Semantik-Ausdruck*, (wobei *Text* als E-Mail verfügbar ist, und *Extrahierte_Information* und *Semantik-Ausdruck* zu berechnen sind) induziert den Ausführungsplan *SMES* $\rightarrow$ *IMAS*, wobei *SMES* und *IMAS* die Manager für die jeweiligen Komponenten darstellen. Zu diesem Arbeitsschritt gehört auch das Verfügbarmachen der benötigten Daten im richtigen Format und gemäß der Inputspezifikation der aktuellen Komponente, um einen erfolgreichen Prozeßablauf zu ermöglichen.

- *Durchführung des Ausführungsplans*: Dies entspricht dem Teil des Problemlösungsprozesses, der darin besteht, Komponenten mit den richtigen Daten in der richtigen Reihenfolge zu aktivieren, und gegebenfalls Fehlerbehandlungroutinen zu starten (z.B. Klärungsdialoge mit dem menschlichen Dialogteilnehmer).

Data-Manager: Der Data-Manager stellt datenbezogene Services zur Verfügung: Datenkonvertierung, -weiterleitung und konvertierung. Auf einer Anfrage des Workflow-Manager hin, leitet der Data-Manager Daten von einer Komponenten zur anderen und führt bei Bedarf (gemäß vorliegender Input- und Output-Spezifikationen von Komponenten) eine Formatkonvertierung durch.

Report-Manager: Der Report-Manager stellt Services zur parametrisierten Report-Spezifikation, -Generierung, -Formatierung und -Printing zur Verfügung.

Condition Manager: Der Condition-Manager stellt dagegen Services zur Spezifikation und Signalisierung von Fehlerzuständen (Conditions bzw. Exceptions) zur Verfügung. Conditions sind als Objekte realisiert, die relevante Informationen für Ihre Behandlung erhalten (Detektor, Art des Fehlers bzw. Fehlercode, etc.). Der Condition-Manager stellt darüberhinaus spezielle Strategien und Verfahren zur Fehlerbehandlung in Form von *Condition-Handlern* zur Verfügung. Diese sind Objekte, die Routinen für die Behandlung spezieller Klassen von Conditions definieren (z.B. Fehler, die möglicherweise auf unbekannter Wörter im Text zurückzuführen sind). In Abschnitt 7.7 werden die realisierten Ansätze zur Fehlerbehandlung beschrieben.

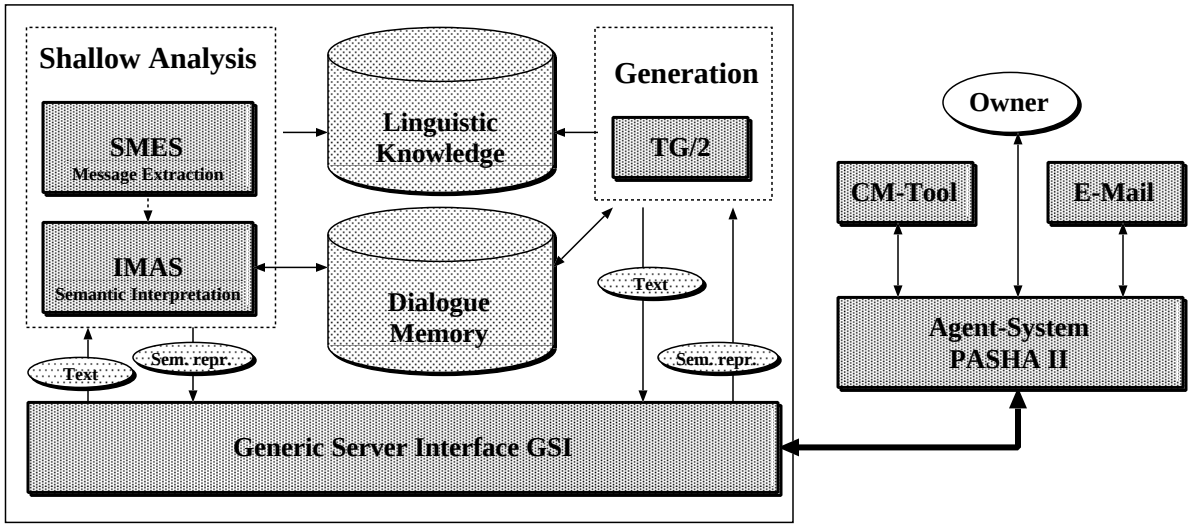

Abbildung 4: COSMA-Systemarchitektur mit smes als Kernsystem.

Interface-Manager: Diese Manager verwalten die Schnittstellen zum Server. Das COSMA-System hat GSI-Manager und GUI-Manager. Der GSI-Manager stellt eine generische Client/Server-Kommunikationsschnittstelle GSI (generic server interface) zur Verfügung (vgl. Abschnitt 7.1.4). Der GUI-Manager definiert Mechanismen und Methoden für eine effiziente Kommunikation zwischen Server und graphische Benutzeroberfläche. Die GUI-Komponente (Graphical UserInterface) ist Gegenstand von Abschnitt 7.8.

Komponentenmanager CCM (Computational Component Manager): Für jede Komponente im System wird ein Manager definiert, der diese einkapselt und sie als Server erscheinen läßt. So wird ausschließlich über den Manager auf die Komponente zugegriffen. Dieser stellt eine wohldefinierte Schnittstelle zur Verfügung, die die für das jeweilige System relevante Funktionalität bereitstellt. Mehrere Manager können mit einer und derselben Komponente assoziert werden (*Resource Sharing*), wobei sich die Manager sich gegenseitig von der Benutzung der Komponente temporär ausschliessen können, um Inkonsistenzen zu vermeiden. Dadurch kann der Server über virtuelle Systeme mit einem minimalen Einsatz von Resourcen und einem hohen Grad der Wiederverwendung von Komponenten und Mechanismen mehrere Dialoge mit verschiedenen Agenten parallel bzw. asynchron führen. Mit den CCM wurde ein Integrationsmodell entwickelt, mit dem Komponenten in das System leicht integrierbar sind, auch wenn diese in verschiedenen Programmiersprachen implementiert wurden und auf physikalisch verteilt sind.

7.1.3 Die virtuelle Systemarchitektur VSA

Die grundlegende Idee des virtuellen Systemarchitekturmodells besteht darin, ein System nicht mehr als Zusammensetzung von Komponenten im herkömmlichen Sinne zu verstehen, sondern eher als ein Team von Manager mit klaren Verantwortungen. So eine Sammlung von Managern stellt dann ein *Operationskontext* dar, denn wir hier *virtuelles System* nennen. So lässt sich ein heterogenes System mit mehreren Komponenten als ein Operationskontext darstellen, in dem bestehende Komponenten von ihren assoziierten Managern vertreten werden.

In COSMA wurde das virtuelle Architekturmodell VSA als objekt-orientiertes Modell definiert und implementiert. Ein virtuelles System ist demnach ein komplexes, aggregiertes Objekt, das sich in erster Linie aus Managern zusammensetzt. Es enthält Mechanismen und Strategien zur Koordinierung und Kontrolle der Interaktionen zwischen der darin enthaltenen Objekten. Prozeß-

und Kontrolldaten sind in einem virtuellen System eingekapselt und die Dienste sind ausschließlich über eine wohldefinierte Schnittstelle verfügbar. Für ein virtuelles System gelten folgende Eigenschaften:

- Die Anzahl der virtuellen Systeme im realen System ist konzeptuell unbegrenzt.
- Jedes Subsystem oder Gruppe von Komponenten kann als virtuelles System mit eigenem Architekturschema definiert werden.
- Jedes virtuelle System ist ein Objekt einer bestimmten Klasse. Damit existieren vererbungsrelationen zwischen virtuellen Systemen, so daß sie auch Resourcen untereinander teilen bzw. wiederverwenden können.
- Manager sind keine integralen Bestandteile von virtuellen Systemen. Sie werden lediglich so zu denen assoziiert, daß im Bedarfsfall aktiviert werden. Somit kann ein Manager von mehr als einem virtuellen System benutzt werden, wobei sein Verhalten von System zu System variieren kann. Im aktuellen System darf zu einem gegebenenem Zeitpunkt kein Manager in mehr als einem virtuellen System aktiv sein.

Der Einsatz des VSA-Modells steigert die Flexibilität des Systems und erlaubt eine höhere Wiederverwendung von Komponenten und Verarbeitungsmechanismen. So lassen sich in COSMA mehrere Anfragen von verschiedenen Agenten parallel bzw. asynchron verarbeiten (parallele Dialogverarbeitung), dadurch daß jeder Dialog in einem separaten virtuellen System verabeitet wird.

7.1.4 Die Client/Server-Schnittstelle

Um mit unterschiedlichen Anwendungssystemen interagieren zu können, muß ein natürlichsprachliches System sehr verschiedenartige Anforderungen erfüllen. Eine exakt definierte Schnittstelle, die für sehr unterschiedliche Aufgaben benutzt werden kann, ist eine Voraussetzung für die Integration eines natürlichsprachlichen Systems in nicht antizipierte Anwendungen. An eine solche Schnittstelle sind folgende Anforderungen zu stellen:

- Die Anwendung sollte das natürlichsprachliche System als Server benutzen können, anstatt eng auf Eigenschaften seiner Software oder Hardware abgestimmt zu sein.
- Die Auswirkungen von Änderungen an dem Server auf die Anwendung sollten umbedingt minimiert werden.
- Die Form der Schnittstelle sollte so einfach wie möglich gehalten werden.
- Daten sollten in einem neutralen Format kodiert werden und nicht auf eine bestimmte Anwendung oder eine bestimmte Repräsentationsebene zugeschnitten sein.
- Das Kommunikationsmodul zwischen Server und Client sollte robust und zuverlässig sein.

Diese Anforderungen wurden in COSMA auf der Grundlage der folgenden, neuen Idee umgesetzt:

- Die Schnittstelle ist deklarativ. Es werden keine Kommandos zwischen Client und Server austauscht, sondern (partielle) Beschreibungen der jeweils vorhandenen sowie der gewünschten Information.
- Als grundlegender Kommunikationskanal zwischen Client und Server dienen TCP/IP-Verbindungen.
- Die Schnittstellensprache basiert auf Merkmalsstrukturen. Sie ist flexibel, theorieneutral und erlaubt die Verwendung von *lazy references*. Der *lazy reference*-Mechanismus ermöglicht eine Optimierung der Kommunikation bzw. Reduzierung der zu übertragenden Datenmenge zwischen kommunizierenden Systemen: (i) Daten, die wiederholt zwischen Client und Server ausgetauscht werden, werden beim ersten Abschicken mit einer Referenz versehen, sodaß

für weitere Kommunikationsschritte lediglich die Referenz geschickt wird; (ii) Referenzen können für bestimmte Informationen eingesetzt werden, sodaß die Information selbst, erst bei Bedarf berechnet oder abgeschickt wird.

Die Schnittstellensprache legt das Format der auszutauschenden Messages fest. Diese können von beiden Seiten (Client und Server) aus formuliert und abgeschickt werden. Clients werden gewöhnlich Aufträge zur Verarbeitung erteilen ("Analysiere diesen Satz!", "Generiere einen Text aus dieser Struktur!"), während der Server in der Regel nach fehlender Information fragen wird. Die Messages werden vom Absender als Merkmalstruktur formuliert und entsprechend vom Empfänger interpretiert.

In COSMA wurden zwei Schnittstellenmodelle entwickelt, die in der der Syntax und Semantik der eingesetzten Features und in dem Grad der Integration von Kommunikationsprotokoll und -sprache konzeptuelle Unterschiede aufweisen.

Der API-Ansatz. Das Konzept der API (Application Program Interface) basiert darauf, den Clients die Möglichkeit zu geben, den Server über eine Programmierschnittstelle zu konfigurieren, um dessen Funktionalität zu erhöhen; z.B durch die Angabe der einzusetzenden Grammatik, des Lexikons, etc. [Spackman, 1995].

Ein Schnittstellenmodul *FedEx* wurde zu diesem Zweck entwickelt. Die zugrundeliegende Kommunikationssprache basiert auf der Fegramed-Darstellung von Merkmalstrukturen [Kiefer und Fettig, 1995] und erweitert die Fegramed-Sprache um *lazy references*. Diese werden auf der einer Seite der Schnittstelle spezifiziert und verweisen auf Werte, die auf der anderen Seite der Schnittstelle berechnet werden (müssen).

In *FedEx* wurden alle Konfigurations- und Anfrageparameter in Form von Merkmalen festgelegt. Jeden Auftrag von Client (etwa die Analyse eines Textes) versieht der Client mit einer *lazy reference*, um auf das Objekt verweisen zu können, das vom Server berechnet werden soll. So kann das Ergebnis zu einer späteren Zeitpunkt abgefragt werden.

Konfigurations- und Kommunikationsparameter sowie Protokolle wurden in der Schnittstellensprache spezifiziert und integriert und bilden damit einen festen Bestandteil der Schnittstelle. Dazu gehört auch die interne zentrale Datenstruktur in Abbildung 16.

Klärungsdialoge bzw. Anfragen vom Server an den Client lassen über eine separate Service-Schnittstelle realisieren, die ein Sub-Modul von *FedEx* mit speziellen Kommunikationsparametern und -protokollen darstellt.

Generic Server Interface (GSI). Die Grundidee von GSI besteht darin, eine generische Serverschnittstelle zu realisieren, die verschiedenen Agenten mit verschiedenen Anforderungen und Anwendungsbereichen die zu erlauben, schnell und flexibel die Dienste eines natürlichsprachlichen Servers in Anspruch zu nehmen. Ferner sollte die server-seitige parallele bzw. asynchrone Verarbeitung von multiplen Anfragen (oder Dialogen) von verschiedenen Agenten unterstützt werden. Weitere Ziele waren,

- die verteilte Verarbeitung durch eine physikalische Verteilung von Systemkomponenten zu unterstützen und
- eine leichte Anpaßbarkeit bzw. Einsatzmöglichkeit des Systems für verschiedene Anwendungsklassen zu gewährleisten.

Das GSI-Modell wurde durch den Versuch motiviert, einige Schwächen vom API-Ansatz zu korrigieren:

- Kommunikationsprotokoll und -parameter sind Teil der Schnittstellensprache. Dadurch wird die Anpaßbarkeit an geänderten Forderungen erschwert.
- Die Schnittstellensprache ist sehr restriktiv, da sie über die Festlegung der Semantik der Anfragen und der Spezifikation einer zentralen Datenstruktur die interne Verarbeitung und das Verhalten bzw. die Funktionalität des Servers vorschreibt.

- Der *lazy reference*-Mechanismus erfordert viel Aufwand (insbesondere auf der Server-Seite) für die Vermeidung von Konflikten, da die Referenz nicht von der Komponente generiert wird, die die zugehörige Information zur Verfügung stellt.

Bei dem GSI-Ansatz unterscheiden wir zwischen Kommunikationsprotokoll und Kommunikationssprache. Damit besteht die Serverschnittstelle aus zwei Modulen: die Schnittstellensprache CCL (Client/Server Communication Language) und das Kommunikationsmodul CCI (Client/Server Communication Interface).

Das CCL-Modul. Ähnlich wie die API-Schnittstellensprache FedEx ist CCL eine auf Merkmalsstrukturen basierte deklarative Spezifikationsprache. Sie enthält Mechanismen zur Verwaltung von Variablen und Referenzen. Dazu gehören auch *lazy references* und anonyme Variablen. Im Gegensatz zu FedEx werden *lazy references* von der Komponente erzeugt, die auch die jeweilige Information zur Verfügung stellt. Die Gegenkomponenten kann aber durch Benutzung von anonymen Variablen *lazy references* anfordern.

CCL besteht aus einem objekt-basierten Kernel, der einen Syntaxparser, Basisobjekte und Interfaces, sowie Mechanismen zur allgemeinen Variablen- und Dialogverwaltung enthält. Ausgehend von diesem Kernel werden durch eine objekt-orientierte Vorgehensweise anwendungs- bzw. domänenspezifische Erweiterungen definiert. So wurde CCL für COSMA um eine Interpretationskomponente erweitert, die Anfragen vorstrukturiert, relevante Informationen an den Workflow-Manager weiterleitet und erzeugte Anworten für den Client strukturiert. Diese werden dann über das Kommunikationsmodul CCI an den Client weiterleitet. In CCL definiert und operiert auf mehrere dialogbezogene Objekte:

- Das Client-Objekt, das dialog-relevante Informationen über den Agenten enthält: Benutzerinformationen (z.B. Name, E-Mail-Adresse, etc.) aktuelle Dialoge, Präferenzen, Präsentationsmodus für die Information, etc.

- Das Dialog-Objekt, das u.a. folgende Informationen enthält: Referenz zum initiierenden Agenten, Informationen über Dialogteilnehmer, aktuelle sowie verarbeitete Äußerungen/Anfragen und Dialogstatus.

- Das Utterance-Objekt enthält Informationen über eine gegebene Äußerung bzw. Anfrage: Referenz zum zugehörigen Dialog, Spezifikation, Status der Verarbeitung, erzielte Ergebnisse, etc.

CCL stellt lediglich die Kommunikationssprache zur Informations- und Datenaustausch zwischen Client und Server und enthält keine Informationen über Kommunikations- und Datentransferprotokolle. Diese sind CCI-Modul (Client/Server Communication Interface) implementiert.

CCI ist ein TCP/IP-basiertes Kommunikationsmodul, der eine reihe von Interprozeßkommunikationsdienste in Form von *Schnittstellenobjekten (interface objects)* bereitstellt. Schnittstellenobjekte enthalten kommunikationsrelevante Parameter und Methoden zum Aufbau und Management einer Client/Server-Verbindung. Das CCI-Modul stellt eine Basisklasse von Objekten zur Verfügung, die sich um spezifische Protokolle erweitert läßt.

Wesentliche Attribute eines Basisobjektes sind: Namen und (Internet-) Adressen der kommunizierenden Systemen, eine (Internet-) Adresse (inkl. Portnummer) des Hostsystems, Input- und Output-Buffer für die Pufferung von eingehenden und ausgehenden Daten. Hinzu kommen Methoden zum Aufbau, Aufrecherhalten und Abbau von Kommunikationskanälen, sowie für das Senden und Empfangen von Nachrichten.

Ein Schnittstellenobjekt wird dadurch eingesetzt, daß es an eine Komponente bzw. eines Prozeßes gebunden wird. Diese Komponente kann durch die so entstandene Kommunikationsschnittstelle mit anderen kommunikationsfähigen Komponenten im Netz interagieren. Die Anbindung eines Schnittstellenobjektes an eine Komponenten geschieht durch die Verwendung eines speziellen Objekts, das von der Komponente selbst kontrolliert wird. So ein Objekt heißt *cci owner* und dient dazu, die für die Kommunikation relevante Dienste des CCI-Objektes in Anspruch zu nehmen.

Für den COSMA-Server wurden zwei Schnittstellenobjektklassen definiert und implementiert, die die Basisklasse um TCP/IP-basierte Protokolle erweitert: ICE-CCI und *streamsock-cci*.

Die Interprozeßkommunikation der ICE-CCI Klasse basiert auf der Kommunikationssoftware ICE. ICE ist eine Architektursoftware für Interprozeßkommunikation, die im Rahmen vom Projekt Verbmobil entwickelt wurde [Amtrup, 1995]. Sie eignet sich insbesondere für verteilte Verarbeitung und weniger für das Server-Modell von COSMA.

Aus diesem Grund wurde speziell für das in COSMA realisierte Client/Server-Konzept wurden die *streamsock-cci* Klasse mit entsprechender Kommunikationssoftware implementiert. Bei *streamsock-cci* Objekten unterscheiden wir Server- und Client-Objekte. Server-Objekte haben eine wohlbekannte Internet-Adresse und enthalten einen Dämonenprozeß, der im Hintergrund läuft und Verbindungen zu Clients verwaltet. Damit kann jedes Modul, das an so einem Objekt gebundenen ist, als Server (etwa im Sinne eines ftp- oder http-Servers) fungieren. Client-Objekte werden ausschliesslich bei Bedarf aktiv. Das STREAMSOCK-Protokoll sieht folgendes typisches Vorgehen vor:

1. Der Server unterhält einen Dämonenprozeß, der über eine bekannte Adresse (inkl. TCP-Portnummer) ansprechbar ist,
2. der Client beantragt den Aufbau einer Sitzung beim Server,
3. der Server akzeptiert die Anfrage und baut die Verbindung auf,
4. der Client formuliert mittels der der Sprache CCL Anfragen, die er an de Server schickt. Es finden Interaktionen (Dialoge) zwischen Client und Server. Informationen werden in CCL repräsentiert.
5. der Client beendet die Sitzung.

Das CCI-Modul ist als allgemeines Tool konzipiert und kann als Kommunikationsschnittstelle für diverse Komponenten und Systeme dienen. So wird CCI im Projekt PARADIME eingesetzt, um das Informationsextraktionssystem smes als Server im Internet bereitzustellen.

7.2 Informationsextraktion mit SMES

Die Informationsextraktion in smes [Neumann *et al.*, 1997; Neumann, 1997] kombiniert ein Kernsystem zum Shallow-Parsing und den Einsatz spezialisierter endlicher Automaten. Die Trennung zwischen Kernsystem und einem benutzerdefinierten System von Automaten erlaubt eine modulare Beschreibung der Grammatik für sprachliche Subdomänen. Der Kern des Systems besteht aus:

- einem Tokenizer zur Erkennung der Eingabe, wobei eine Menge regulärer Ausdrücke zur Identifikation der Fragmentmuster (z.B. Wörter, Zeitausdrücke usw.) eingesetzt wird,
- einer Komponente zur schnellen lexikalischen und morphologischen Verarbeitung (basierend auf dem Morphix-Tool, das in [Finkler und Neumann, 1988] näher beschrieben ist),
- einem Shallow-Parsing-Modul, das auf einer Menge endlicher Automaten basiert,
- einer flexiblen Ausgabe-Komponente, die die extrahierten und analysierten Ergebnisse kombiniert und in unterschiedlichen Ausgabeformaten repräsentieren kann.

Die Extraktions-Muster, die für die Erkennung und Extraktion von Wortgruppen verwendet werden, sind als endliche Automaten realisiert und werden mit Lisp Funktionen kompiliert. Ein endlicher Automat besteht aus einem eindeutigen Namen, einem Erkennungsteil, einer Ausgabebeschreibung und einer Menge von Kompilationsparametern.

Auf der Basis der von der morphologischen Analyse der Textfragmente bereitgestellten linguistischen Information führt das System eine Konstituenten-Analyse durch. Die extrahierten

$$\begin{bmatrix} \texttt{TYPE:} & \langle \texttt{date\_expr}, \texttt{pp} \rangle \\ \texttt{SEM:} & \begin{bmatrix} \texttt{DATE-PREP:} & \texttt{am} \\ \texttt{WEEKDAY:} & 1 \end{bmatrix} \end{bmatrix}$$

Abbildung 5: smes-Ausgabe für die temporale PP *"am Montag."*

$$\begin{bmatrix} \texttt{TYPE:} & \begin{bmatrix} \texttt{FIRST:} & \begin{bmatrix} \texttt{MODAL:} & \texttt{koenn} \\ \texttt{NEG:} & \texttt{t} \\ \texttt{QUES:} & \texttt{no} \end{bmatrix} \\ \texttt{REST:} & \langle \texttt{komm} \rangle \end{bmatrix} \end{bmatrix}$$

Abbildung 6: Behandlung von Negation, Modus und Modalität in smes am Beispiel von *"Ich kann am Montag nicht kommen."*

Fragmente werden anschließend von einem lexikalisch gesteuerten bidirektionalen Shallow-Parser durchgeführt. Ausgehend von dem Verb wird mithilfe des ihm assoziierten Verbautomaten links und rechts vom Verb nach relevanten Konstituenten gesucht. Komplemente und Adjunkte, darunter alle temporalen Ausdrücke, werden mit dem Verb zusammengestellt. Mit der Definition der Verbautomaten im COSMA werden zusätzlich zu der verbalen Subkategorisierung Information über Negation, Modus und Modalität bereitgestellt. Diese Information wird für die nachfolgende semantische Interpretation der Fragmente gebraucht. Das Ausgabe-Format für temporale Information und Verbrahmen wird auszugsweise in den Abbildungen 5 und 6 gezeigt.

Damit die Informationsextraktion in der Terminvereinbarungsdomäne erfolgreich ist, muß das Parsing-System sowohl Zeitausdrücke einschließlich ihrer Semantik als auch grundlegende syntaktische Konstruktionstypen einschließlich der Verbkomplementation abdecken. Hierfür wurden eine Reihe von Erweiterungen an smes vorgenommen.

7.2.1 Erweiterungen der Funktionalität von smes

Infolge der modularen Architektur von smes waren Erweiterungen einfach zu integrieren.

COSMA muß in der Lage sein, unvollständige Text-Eingaben zu bearbeiten. Nicht selten werden Textfragmente extrahiert, in denen das Verb fehlt (elliptische Konstruktionen oder auch Lücken infolge von Nachlässigkeit bei der Verfassung von E-Mails). smes war nicht in der Lage, solche Fragmente zu verarbeiten, da der typischerweise verbale lexikalische "Anker" für die bidirektionale Suche fehlte. Das Kernsystem von smes wurde entsprechend modifiziert, so daß auch aus verblosen Fragmenten die für COSMA relevante Information ausgegeben wird. smes leistet keine Rekonstruktion von Ellipsen, sondern informiert nur die nachfolgende semantische Interpretation darüber, daß eine solche Konstruktion vorhanden ist.

Im Zuge dieser Änderung im Kernsystem von smes war es ebenfalls möglich, allein auftretende Modalverben (z.B. in "Ich kann aber am Montag nicht!") als Vollverben zu behandeln. Im gleichen Rahmen wurde eine Behandlung abtrennbarer Verbpräfixe ("Machen wir für Montag nachmittag einen Termin aus.") definiert.

Die COSMA-Anwendung zeigt damit positive Auswirkungen auf das Design des Kernsystems von smes. Wie sich bei der Anpassung von smes auf eine industrielle Anwendungsdomäne gezeigt hat, konnten die erfolgten Ergänzungen zur Verbesserung der Performanz von smes in dieser Domäne beigetragen.

7.2.2 Die Rolle von smes im Gesamtssytem

Ursprünglich wurde smes als Informationsextraktionsmaschine verwendet und durch die Anforderungen der COSMA-Anwendung auch mit linguistischen Verarbeitungsprozessen angereichert. smes wird aber nicht nur für die flache Analyse der natürlichsprachlichen Eingabe eingesetzt, sondern

muß auch dafür sorgen, daß die nachfolgende semantische Interpretation einen direkten Zugang zur relevanten Information haben kann. So wurde in der Ausgabe der Verbautomaten die Information über Negation, Modus und Modalität in kompakter Form dargestellt (siehe Abbildung 6). Außerdem werden relevante Partikeln ("lieber", "leider", "gerne", usw.) und prädikative Konstruktionen (z.B. in "Montag ist gut!") aus den Texten extrahiert und ausgegeben, um die semantische Interpretation und die Klassifikation der Dialog-Primitive zu erleichtern. Das System der endlichen Automaten wurde so angepaßt, daß Fragmente mit multiplen temporalen Ausdrücken korrekt verarbeitet werden. Damit werden auch Gegenvorschläge ("Statt Dienstag wäre mir Donnerstag lieber.") korrekt erkannt, oder zumindest in einer Form ausgegeben, die der semantischen Interpretation die Erkennung dieser Sprechakte ermöglicht.

7.3 Korpusbasierte Grammatikentwicklung

Zur Unterstützung einer empirisch fundierten linguistischen Verarbeitung im Rahmen der Terminvereinbarungsdomäne basiert die Grammatikentwicklung auf den Ergebnissen der Auswertung eines E-Mail-Korpus [Klein *et al.*, 1997]. Dadurch konnte die Entwicklung der syntaktischen Analysekomponente ebenso wie die effiziente Verarbeitung typischer syntaktisch-semantischer Konstruktionstypen aus der Terminplanungsdomäne optimal unterstützt werden.

7.3.1 Auswahl und Annotierung des Korpus

Um die linguistischen Phänomene, die in der Terminvereinbarungsdomäne auftreten, zu identifizieren und damit die relevante Subsprache zu spezifizieren, wurde ein E-Mail-Korpus ausgewertet. Als Basis wurden 160 reale E-Mail-Nachrichten aus einer erstellten Sammlung von mehreren Hundert E-Mails mit Termininhalten[1] manuell ausgewählt. Obwohl keine ausgearbeiteten Selektionskriterien vorlagen, wurde bei der Auswahl versucht, eine möglichst große Vielfalt typischer linguistischer Strukturen, die in der Anwendungsdomäne vorkommen anhand der selektierten Testdaten abzudecken.

Die Ergebnisse zur Klassifikation und Annotierung von Testmaterial für natürlichsprachliche Systeme aus den Projekten DiTo und TSNLP konnten gewinnbringend eingesetzt werden: Aufbauend auf den TSNLP-Richtlinien zur Annotierung von Sprachdaten (vgl. [Estival *et al.*, 1996] wurde ein Annotationsschema entworfen, das eine linguistische Annotierung der E-Mail-Nachrichten erlaubt, die nicht nur speziell der nachfolgenden Verarbeitung nutzt, sondern eine allgemeine linguistische Charakterisierung der Texte widerspiegelt.

Das Annotationsschema umfaßt neben syntaktisch-semantischen Informationen einzelner Konstituenten, d.h. der Verben und ihrer zugehörigen Komplemente und Adjunkte, auch strukturelle und funktionale Informationen zu den syntaktischen Konstruktionstypen. Zur Annotierung dieser Information wurden die in TSNLP erarbeiteten Kategorie- und Funktionslisten eingesetzt.[2] Da der Schwerpunkt von TSNLP auf der syntaktischen Beschreibung lag, während für COSMA auch semantische Informationen von großer Bedeutung sind, wurden die ursprünglichen Listen um Informationen erweitert, die semantische Merkmale der Terminvereinbarungsdomäne abbilden. Hierzu gehört zum einen die detaillierte Klassifikation der Präpositional- und Nominalphrasen, um die für die Termindomäne wichtigen Temporalausdrücke adäquat zu beschreiben wie Abbildung 7 zeigt.

Da das semantische Modul von COSMA Informationen zur Klassifizierung von Kommunikationsprimitiven benötigt, sind die E-Mail-Nachrichten außerdem mit entsprechenden Informationen zu Negation, Modus und Modalität annotiert.

Damit die für die Anwendungsdomäne erforderliche linguistische Abdeckung bestimmt werden konnte, sind die annotierten E-Mail-Texte in einem relationalen Datenbanksystem aufbereitet, so daß die vorkommenden linguistischen Konstruktionstypen gezielt herausgefiltert werden können. Außerdem erlaubt der Einsatz des relationalen Datenbankschemas die einfache Modifikation und

[1] Neben Terminvereinbarungsdialogen umfaßt die Sammlung auch Terminankündigungen, die für die Anwendung weniger relevant sind.

[2] Die Listen sind in [Estival *et al.*, 1995] angegeben.

Annotierung	Beispiel
Präpositionalphrasen Wie wäre es ...	
PP_temp	*diese Woche*?
PP_temp-date	*am 4.11*?
PP_temp-day	*am Montag*?
PP_temp-dur	*von 8-12*?
PP_temp-time	*um 10*?
Nominalphrasen Ich komme ...	
NP_temp-date	am Montag, *den 4.11.*
NP_temp-day	*Montag*, 14.00.
NP_temp-time	am Montag, *14.00.*

Abbildung 7: Semantische Annotierung prototypischer PPs and NPs (die zu den angegebenen Annotationen gehörenden Sprachteile sind *kursiv* gedruckt).

Ergänzung des Korpus um neue E-Mail-Texte und weitere Annotationen. Zwei (indirekt) miteinander verbundene Datenbanksysteme wurden eingesetzt, um das Material zu organisieren:

- **DiTo**, eine einfache Volltextdatenbank, in der die vollständigen E-Mail-Texte abgerufen werden können.
- **TSDB**, eine elaborierte Faktendatenbank zum Speichern der annotierten E-Mail-Fragmente, die das Abrufen spezieller linguistischer Konstruktionen erlaubt.

Mit Hilfe von Datenbankabfragen wurde das E-Mail-Korpus ausgewertet, so daß zum Beispiel alle Fragmente zusammengestellt werden konnten, die temporale PPs enthalten (Abbildung 8).

```
100: also so gegen 1, oder 1.30 p.m. ?
105: Wie waers naechste Woche Dienstag, um 9 Uhr?
106: geht es morgen um 9 statt 10?
107: ich wuerde gerne unser Treffen von Donnerstag auf naechsten Montag (9 Uhr?)
     verschieben
109: Ich komme mit den Resultaten am Donnerstag vorbei, ok?
111: um 15.30 spaetestens wieder da zu sein.
112: Am Montag morgen bin ich pktl. um 11 Uhr bei Dir
114: Ist Dir das recht, wenn ich am Nachmittag mal reinschaue?
116: komme nach dem Essen vorbei. (ein gutes Stueck vor 14 Uhr)
118: waere Dir ein Treffen am Dienstag, den 9. Mai, um 10.15 recht?
```

Abbildung 8: Datenbankauszug von E-Mail-Fragmenten mit temporaleb PPs.

Aus der Gesamtanalyse ergab sich eine Prioritätenliste (vgl. Abbildung 9), an der sich die geplante Entwicklung des Shallow-Parsers smes orientierte.

Ein spezielles Kommunikationsphänomen, das für die Terminvereinbarung wesentlich ist, im Korpus aber nur in sechs E-Mail-Texten nachgewiesen werden konnte, sind Gegenvorschläge. Um diesen Bereich anhand linguistischer Konstruktionstypen zu exemplifizieren, wurden Korpusdaten induziert, d.h. Versuchspersonen wurden gebeten, auf Terminvorschläge natürlichsprachliche Antworten mit einer Ablehnung des angegebenen Termins und einem Gegenvorschlag zurück zu senden. Diese 85 E-Mail-Fragmente wurden ebenfalls annotiert und in einer Datenbank abgelegt.

7.3.2 Semi-automatische Grammatikentwicklung

Die konkrete Realisierung der Automaten basiert auf der linguistischen Annotierung der E-Mail-Fragmente, die im Korpus vorkommen. Diese Annotation ermöglicht eine semi-automatische Be-

Phänomen	Anzahl	Phänomen	Anzahl
Temporale PPs	127	Temporaladverbien	33
Temporale NPs	105	Prädikation (Kopula)	29
Transitive Verben	70	Temporale verblose Ausdrücke	28
Lokative PPs	47	Thematische PPs	22
Intransitive Verben	43	Auxiliare als Hauptverben	14
Modalverben	40	Relativsätze	14
Partikelverben	35	Modalverben als Hauptverben	7
Hilfsverben	34	Ditransitive Verben	3

Abbildung 9: Vorkommen syntaktischer Phänomene im E-Mail-Korpus.

schreibung der Automaten, und zwar sowohl für die Erweiterung des Lexikons als auch für die Verfeinerung der bestehenden Automaten. Im ersten Fall wird neu auftretendes lexikalisches Material (Verben) automatisch ins COSMA-Lexikon eingetragen und mit dem entsprechenden Verbautomaten assoziiert (die Verbautomaten repräsentieren Klassen von Verben dar). Im zweiten Fall, wenn sich zeigt, daß bestehende Verbautomaten nicht alle mögliche Komplementationsstrukturen oder (temporale) Adjunktionen abdecken, wird diese Information ebenfalls halb-automatisch ins Automatensystem hinzugefügt.

Auf der anderen Seite kann smes die Annotierung von Sprachmaterial unterstützen, indem die Analyse-Ergebnisse zur Annotierung neuer E-Mail-Nachrichten verwendet werden. Dazu müssen die Automaten nur präzise definiert sind. Die Automaten zur Verarbeitung temporaler NP's und PP's sind so weit entwickelt , daß dies möglich ist. Zur Annotierung von Korpusdaten sind nur noch Ausgabeprozedur und das Ausgabeformat von smes entsprechend zu modifizieren.

Auch auf der Grundlage dieses Experiments wurden Judith Klein und Thierry Declerck eingeladen, eine Arbeitssitzung bei dem ACL-Workshop "Tagging Text with Lexical Sematics" (4-5 April 1997, Washington D.C.) zu leiten.

7.3.3 Evaluierungsszenario für smes

Ziel der Evaluierung von smes war es, ein Profil der erreichten linguistischen Abdeckung zu erhalten und die Schwachstellen der syntaktischen Analyse aufzudecken. Da sich die Evaluierung von smes auf die Bewertung des Shallow-Parsers als einer separaten Komponente des Gesamtsystems COSMA bezieht, wurde ein diagnostischer Ansatz im Rahmen der zyklischen Evaluierung unter Anwendung der Glass-Box Methode gewählt. Ausgehend von der Erkenntnis, daß Referenzmaterial wie Testsuites und Testkorpora anerkannte Werkzeuge zur Unterstützung der Grammatikentwicklung sind und sich als realistischer Maßstab zur Bewertung der entwickelten Analysemethoden eignen, mu sste zunächst eine geeignete Testmenge ausgewählt werden.

Da die E-Mail-Texte der Korpusdatenbank nicht direkt zur Systementwicklung eingesetzt worden waren, sondern dazu dienten, die Subsprache der Terminvereinbarungsdialoge und damit die Anforderungen an die syntaktische Verarbeitung allgemein zu definieren, konnten Auszüge der Sammlung als Testmaterial herangezogen werden. Mit Hilfe eines Zufallsgenerators wurden 3 x 15 E-Mail-Nachrichten als erstes Testset ausgewählt. Zusätzlich wurde diesselbe Anzahl neuer, nicht im Korpus annotierter E-Mails zufällig zusammengestellt. Aufgrund des Einsatzes eines Zufallsgenerators konnte eine E-Mail mehrmals vorkommen.

In der Testphase wurden die E-Mail-Text einzeln in smes eingespielt, wobei die Texte in einigen Fällen editiert werden, um bestimmte (für die Verarbeitung störende aber inhaltlich irrelevante) Zeichen zu entfernen. Dann wurden die Analyse-Ergebnisse im Ausgabe-Format der fegramed-Merkmalsstrukturen manuell überprüft. Da die Evaluierung im Zusammenhang mit der Einbindung von smes in COSMA gesehen werden muß also anwendungsspezifisch ist und sich eher an Performanzvorgaben als an der rein linguistischen Kompetenz orientiert, waren folgende Kriterien zur Qualität der Ergebnisse relevant:

Evaluierung	gut	mittel	schlecht
Erster Testlauf			
Datenbank-Set	51,1 %	37,8 %	11,1 %
Neue E-Mails	28,9 %	20,0 %	51,1 %
Zweiter Testlauf			
Datenbank-Set	73,2 %	13,3 %	11,1 %
Neue E-Mails	62,1 %	11,1 %	26,6 %

Abbildung 10: Ergebnisse der diagnostischen Evaluierung von smes.

- Verbtemplate: korrekte Anbindung eines Automaten an einen lexikalischen Verbeintrag und korrekte Verarbeitung der abhängigen syntaktischen Struktur
- Modalität: korrekte Verarbeitung von Modalverben und Ausgabe der besonderen Satzmodalität
- Temporale Ausdrücke: korrekte Erkennung und Zuordnung temporaler Informationseinheiten zu den entsprechenden syntaktischen Domänen

Je nach Erfüllung der Kriterien wurden folgende Qualitätsprädikate unterschieden/footnoteDabei ist anzumerken, daß es sich hier eher um eine unscharfe Klassifizierung handelt, die nicht so einfach meßbar ist.

- **Gut**: Die Analyse ist korrekt oder es liegen minimale Fehler vor, die aber das wichtigste Kriterium, die temporale Information, nicht betreffen und nicht zu Fehlinterpretationen der semantischen Analyse führen.
- **Mittel**: Es liegen mittelschwere Fehler vor, d.h. die temporale Information ist so ausgegeben, daß einen korrekte Weiterverarbeitung im Rahmen der semantischen Analyse möglich ist. Obwohl die syntaktische Analyse nicht vollständig oder nicht ganz korrekt ist, führt das Ergebnis nicht zu semantischen Fehlinterpretationen.
- **Schlecht**: Die syntaktische Analyse zeigt gravierende Fehler und Lücken, so daß eine korrekte Interpretation der Aussage nicht möglich ist.

Die Ergebnisse des ersten Durchlaufs der diagnostischen Evaluierung wurden als Feedback für die Weiterentwicklung der Automaten genutzt. Als Hauptpunkte, die zu einer schnellen Leistungssteigerung führen sollten, wurden (i) Verarbeitung der Prädikation (Kopula-Konstruktionen), (ii) Ergänzung und Modifikation bestimmter Verbtemplates und (iii) Erkennung bestimmter lexikalischer Terminausdrücke identifiziert. Nach der Verbesserungsphase wurde ein zweiter Testzyklus durchgeführt, an dessen Ergebnissen eine deutliche Leistungssteigerung für die anvisierten Problemfelder abgelesen werden konnte. Das Ergebnis beider Testläufe ist in Abbildung 10 zusammengefaßt.

7.4 Semantische Interpretation

Speziell in dem aktuellen Anwendungsbereich von Cosma (Terminvereinbarung) ist es wichtig, Sprechakte und temporale Ausdrücke aus analysierten E-Mail-Nachrichten effizient zu extrahieren und diese Information für weitere Verarbeitung durch ein Agentensystem adäquat repräsentieren zu können. Insbesondere muß dabei von bestimmten sprachlichen Eigenschaften abstrahiert werden, und es müssen gewisse implizit enthaltene Informationen rekonstruiert werden. Für diese Zwecke ist das Imas-System (Information extraction Module for Appointment Scheduling) entwickelt worden. Die Schnittstellensprache von Imas IL (Cosma Interface Language) besteht aus einer merkmalbasierten Repräsentationssprache (inkl. Parser und Printer) und eine Extraktions-

$$\begin{bmatrix} \texttt{COOP:} & \texttt{accept} \\ \texttt{RANGE:} & \begin{bmatrix} \texttt{LEFT-BOUND:} & \begin{bmatrix} \texttt{WEEKDAY:} & 1 \end{bmatrix} \end{bmatrix} \\ \texttt{APPT:} & \begin{bmatrix} \texttt{LEFT-BOUND:} & \begin{bmatrix} \texttt{HOUR:} & 14 \end{bmatrix} \\ \texttt{RIGHT-BOUND:} & \begin{bmatrix} \texttt{HOUR:} & 16 \end{bmatrix} \end{bmatrix} \\ \texttt{LOCATION:} & \texttt{DFKI} \end{bmatrix}$$

Abbildung 11: Initiale IL-Repräsentation für *Ich komme am nächsten Montag von 14 bis 16 Uhr ans DFKI.*.

$$\begin{bmatrix} \texttt{COOP:} & \texttt{accept} \\ \texttt{RANGE:} & \begin{bmatrix} \texttt{LEFT-BOUND:} & \begin{bmatrix} \texttt{YEAR:} & 1996 \\ \texttt{MONTH:} & 3 \\ \texttt{DAY:} & 18 \\ \texttt{WEEKDAY:} & 1 \\ \texttt{HOUR:} & 8 \\ \texttt{MINUTES:} & 0 \end{bmatrix} \\ \texttt{RIGHT-BOUND:} & \begin{bmatrix} \texttt{YEAR:} & 1996 \\ \texttt{MONTH:} & 3 \\ \texttt{DAY:} & 18 \\ \texttt{WEEKDAY:} & 1 \\ \texttt{HOUR:} & 18 \\ \texttt{MINUTES:} & 0 \end{bmatrix} \end{bmatrix} \\ \texttt{APPT:} & \begin{bmatrix} \texttt{LEFT-BOUND:} & \begin{bmatrix} \texttt{HOUR:} & 14 \\ \texttt{MIN:} & 0 \end{bmatrix} \\ \texttt{RIGHT-BOUND:} & \begin{bmatrix} \texttt{HOUR:} & 16 \\ \texttt{MIN:} & 0 \end{bmatrix} \end{bmatrix} \\ \texttt{DURATION:} & \begin{bmatrix} \texttt{HOUR:} & 2 \\ \texttt{MIN:} & 0 \end{bmatrix} \\ \texttt{LOCATION:} & \texttt{DFKI} \end{bmatrix}$$

Abbildung 12: IL-Repräsentation für *Ich komme am nächsten Montag von 14 bis 16 Uhr ans DFKI.*

und Inferenzmaschine. Die IL-Sprache erlaubt die Darstellung von temporalen Informationen, wobei unterschieden wird zwischen Intervallen, innerhalb derer ein Termin noch festzulegen ist (`RANGE`) und Intervallen, die den Termin beschreiben (`APPT`). Somit fordert die Semantik von IL, daß das Intervall `APPT` in dem Intervall `RANGE` enthalten ist.

Ferner werden in IMAS aufgrund linguistischer Eigenschaften des Eingabesatzes sowie des aktuellen Dialogzustandes die möglichen Kooperationsprimitiva berechnet (`arrange, reject, accept` etc.). Falls mehrere Interpretationen möglich sind, wird die Heuristik verfolgt, die informationsreichste Lesart zuerst anzubieten. Alternativen werden in der Reihenfolge abnehmender Information mithilfe von Backtracking zur Verfügung gestellt. Somit kann das Client-System auf der Grundlage seines Wissens die optimale Lesart bestimmen.

Wie Abbildung 12 verdeutlicht, erfordern die Extraktion und präzise Repräsentation von komplexen temporalen Ausdrücken mächtige Inferenzmechanismen. Die IMAS-Inferenzmaschine wendet Regeln und Strategien für die Extraktion von temporalen Informationen an; Anaphernresolution und Behandlung von Unterspezifikationen wird durch Default-Regeln behandelt. Durch Anwendung der Inferenzregeln kann implizite Information explizit repräsentiert werden. Die initiale Repraßentation für das Beispiel aus Abbildung 12 enthält nur explizite Information (vgl. Abbildung 11). Der implizite Sprechzeitpunkt dient als Referenz zur Berechnung eines absoluten `RANGE` Intervalls für *nächsten Montag*. Ferner werden, in Übereinstimmung mit dem Client-System, die

```
(defproduction time "TI5"
              (:PRECOND (:CAT TIon
                         :TEST ((time-fromto-interval-p (self))))
               :ACTIONS (:TEMPLATE (:RULE TPd (fromto-part (self)))
                                   "von "
                                   (:RULE TPintd (from-part (self)))
                                   "bis "
                                   (:RULE TPintd (to-part (self))))))
```

Abbildung 13: Produktionsregel in TGL, z.B. für "9. 8. 1995 von 14 Uhr 30 bis 18 Uhr".

im Kalender angenommenen Verfügbarkeitsszeiten (8 bis 18 Uhr) als früheste Beginn- und späteste Endzeiten eingesetzt. Dies ist besonders dann wichtig, wenn die Dialogäußerung keine weitere Einschränkungen hinsichtlich der Uhrzeit aufweist.

Für die Spezifikation von Extraktions- und Inferenzregeln wurde das System $\mathcal{CIRO}$ entwickelt. Es besteht aus einer deklarativen Sprache und einem Compiler, der die Regeln in effiziente Lisp-Funktionen übersetzt. Das System ist speziell im Kontext von COSMA entwickelt worden und basiert auf dem in der Grammatik verwendeten Merkmals-Formalismus, wodurch die Extraktion der Information wesentlich erleichtert wird. Nichtsdestotrotz kann $\mathcal{CIRO}$ darüber hinaus bei anderen Informationsextraktionsaufgaben eingesetzt werden, in denen Aktionen abhängig vom Aufbau der eingegeben Merkmalstrukturen durchgeführt werden sollen. Somit ist IMAS modular an unterschiedliche Eingabe-Repräsentationen anpaßbar. Es sind Adapter für die Ausgabe der PARADICE-Kernmaschine und für die Ausgabe des smes-Systems implementiert.

7.5 Generierung

Unter Verwendung früher entwickelter Software wurde das produktionsregel-basierte Generierungssystem TG/2 entworfen und implementiert, das als Eingabe getypte Merkmalstrukturen erhält und als Ausgabe deutsche Texte erzeugt [Busemann, 1996]. Die Verarbeitung erfolgt in zwei Schritten. Zuerst wird die Eingabestruktur in ein internes Format übersetzt, wodurch eine weitgehende Unabhängigkeit der eigentlichen Verarbeitung von externen Eingabeformaten erzielt wurde. Die Eingabe steuert dann im zweiten Schritt die Auswahl von Produktionsregeln, die wiederum die linguistische Realisierung der Eingabe sicherstellen.

Durch eine strikte Trennung von Verarbeitung und linguistischem Wissen (Templates; ein Beispiel zeigt Abbildung 13) können sehr schnell kleine, domänenspezifische Grammatiken erstellt werden. Für COSMA wurde besonderer Wert auf Terminbeschreibungen (Zeitpunkte und -intervalle) gelegt. Zur Regelbeschreibung wurde die Repräsentationssprache TGL definiert. Ein LR-Parser überprüft erstellte Regeln auf syntaktische Korrektheit.

Ein einheitliches Regelformat erlaubt die Spezifikation von unterschiedlich fein ausgearbeiteten linguistischen Objekten. Im einfachsten Fall benutzt man vorgefertigte Texte, die in wohldefinierten Situationen zum Einsatz kommen. Etwas komplexer gestaltet sich die Ebene der Templates, in denen solche Texte mit Daten kombiniert werden können. Den höchsten Grad an Flexibilität erreicht man in TG/2 durch kontextfreie Regeln.

Der allgemeine und vom sprachlichen Wissen getrennte Interpreter basiert auf einem dreistufigen Verarbeitungszyklus, wie er aus Produktionensystemen bekannt ist: Anwendbare Regeln finden ("matchen"), entscheiden, welche Regel als nächste anzuwenden ist ("Konfliktlösen"), Regel anwenden ("feuern"). Dieses Verfahren wurde um eine flexible Rücksetzstrategie erweitert, die es erlaubt, unterschiedliche Alternativen mithilfe extern definierbarer Konfliktlöse-Strategien gezielt anzusteuern. Gleichzeitig gelingt die effiziente Wiederverwendung von bisher erzeugten Teilergebnissen, was den Aufwand zur Berechnung einer zweiten und weiterer Lösungen stark reduziert.

Die Parametrisierung des Generierungsprozesses durch externe Kriterien (etwa formelle vs. informelle Anrede, Reihenfolge der präsentierten Information) ist durch die Auswahl bestimmter

Regeln in Konfliktmengen definiert und wirkt sich als Beeinflussung der Reihenfolge aus, in der alle möglichen Lösungen generiert werden. Die am meisten bevorzugte Lösung, d.h. diejenige, die den meisten Kriterien genügt, wird zuerst erzeugt.

In COSMA kann somit das Client-System (oder andere Moduln des Servers) Präferenzen für zu generierende Außerungen spezifizieren. Die Arbeiten an TG/2 wurden zum Teil im Rahmen einer Diplomarbeit durchgeführt [Wein, 1996].

7.6 Das Agentensystem PASHA

Zunächst wird die Grundarchitektur und Funktionalität der PASHA-IIAgenten beschrieben (für Details wird auf [Schmeier und Schupeta, 1996] verwiesen). Davon ausgehend werden die durch die Anbindung an den NL-Server entstandenen neuen Anforderungen an die Agenten dargelegt sowie deren funktionale und konzeptionelle Veränderungen.

7.6.1 Ausgangspunkte

Das Problem der Terminvereinbarung. Terminvereinbarung ist in der Multiagentechnologie ein häufig verrwendetes Szenario, um verschiedene Arbeitsweisen von Agenten zu erforschen. Verschiedene Autoren befaßten sich mit einer Klassifizierung der verschiedenen Ansätze zur Lösung dieses Problems. Durfee und Sen [Sen und Durfee, 1992] schlagen vor:

1. Protokolle auf der Grundlage von Verträgen (*contratcing*)
2. Verteilte Suchformalismen [Yokoo und Durfee, 1991],
3. organisatorische und soziale Regeln [Malone, 1987; Bond und Gasser, 1988; Goldman und Rosenschein, 1993],
4. Multi-Agenten-Planen [Durfee *et al.*, 1989],
5. entscheidungs- und spieltheoretisch fundierte Verhandlungsstrategien [Rosenschein *et al.*, 1986; Zlotkin und Rosenschein, 1990],
6. Linguistisch-pragmatische Ansätze [Cohen und Levesque, 1990a; Cohen und Perrault, 1979].

Der Ansatz, der in PASHA-II verfolgt wird, ist eine Kombination der Ansätze 4 und 5. Die Terminvereinbarung selbst geschieht durch Verhandlungen, die zwischen den Agenten geführt werden. Als Grundlage dienen Verhandlungsstrategien, die in Zusammenarbeit mit dem psychologischen Institut der Universität des Saarlandes aus Interviews mit Sekretärinnen aus den verschiedensten Bereichen der Gesellschaft ausgearbeitet wurden. Diese Strategien sichern eine zielgerichtetete Verhandlungsführung und spiegeln das den Agenten auferlegte Prinzip der Rationalität wider. Die Verhandlungen selbst bestehen stets aus einem Verhandlungsführer, dem *Initiator*, und einem oder mehreren Verhandlungsteilnehmern, den *Partizipienten*. Der initiator schlägt einen konkreten Termin vor und reagiert je nach gewählter Strategie auf die Antworten der Partizipienten. Die somit entstehende Lösung ist mehr oder weniger ein Kompromiß, der sowohl die Präferenzen des Initiators als auch die Präferenzen der Partizipienten mit in Betracht zieht.

Agentenarchitektur. Die Grundkonzeption des Agenten bedient sich einer Abwandlung des InteRRaP Agentenmodells [Müller und Pischel, 1993] (Abbildung 14). Sie besteht aus den folgenden 4 Schichten:

1. In der *Schnittstelle zur Umgebung* befinden sich die Module für Perzeption, physikalischen Aktionen und Kommunikation. Diese Ebene verbindet den Agenten mit der Welt und ist stark von der jeweiligen Domäne abhängig. In PASHA-II besteht diese Ebene aus einem Kommunikationsmodul zwischen den Agenten und einer grafischen Oberfläche, das auf dem Port-Socket-Konzept des Internet basiert, und der E-Mail-Verbindung zum Benutzer.

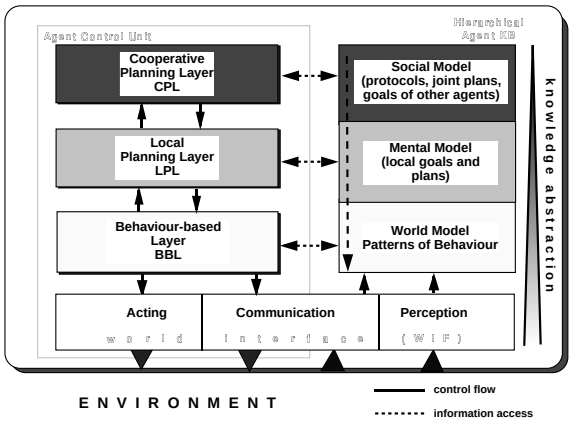

Abbildung 14: Die Agentenarchitektur InterRRaP

2. Die *verhaltensbasierte Schicht* ist zuständig für reaktives Verhalten und stellt prozedurales Wissen zur Verfügung, um Routineaufgaben ohne explizites Reasoning erledigen zu können. Diese Fähigkeiten erstehen aus sogenannten Verhaltensmustern (*Patterns of Behaviour*), die auf Nachrichten oder Ereignisse in der Umgebung des Agenten *reagieren* oder von höheren Schichten zur Ausführung veranlaßt werden.

3. Die *lokale Planungsebene* verwendert Mechanismen zum Schlußfolgern, um lokale Ziele zu erreichen und geht dabei *zielgerichtet* vor. Die generierten Pläne sind hierarchisch aufgabaute Skelettpläne, d.h. sie bestehen aus weiter verzweigbaren Unterplänen oder einzeln durch die verhaltensbasierte Schicht ausführbaren Aktionen. In PASHA-II wird diese Ebene von einem Constraint-Problemlösungs-Algorithmus ausgefüllt, der konkret für die Aufgabe der Terminfindung im lokalen Terminkalender entwickelt wurde. Die von der Kooperationsebene erzeugten Teilpläne werden hier instanziiert und an die verhaltensbasierter Ebene weitergeleitet.

4. Die *Kooperationsebene* ermöglicht die Kooperation mit anderen Agenten sowie die Erzeugung und Synchronisation von gemeinsamen Plänen, d.h. diese Ebene übernimmt alle Planungen, an denen mehrere Agenten beteiligt sind. Die Synchronisation von Verhandlungen, Plangenerierung und Ausführung geschieht durch Protokolle, welche die Reihenfolge von Nachrichten und Reaktionen auf diese festlegen. In PASHA-II wird diese Funktionalität durch einen Konditionalplaner realisiert. Dieser Konditionalplaner benutzt Programme, die in einer speziellen Programmiersprache SvenLog sind, und expandiert diese mit Hilfe eines Interpreters zu hierarchischen Skelettplänen mit Verzweigungen.

Die zweite wesentliche Komponente der Architektur ist die *Wissensbasis*, die wie der Agent hierarisch aufgebaut ist. Die einzelnen Schichten entsprechen in ihrem Abstraktionsgrad des Wissens den Anforderungen, die die korrespondierenden Layers des Agenten stellen. In der untersten Ebene der Wissensbasis findet sich das Weltmodell, darüber das mentale Wissen, das lokale Pläne und Ziele umfaßt und in der obersten Schicht schließlich das "soziale" Modell des Agenten, also das Wissen über die Möglichkeit der Kooperation, und wie diese zu bewerkstelligen ist.
Der hierarchische Aufbau beinhaltet auch, das das Wissen einer Schicht nur der korrespondierenden und höheren Ebenen des Agenten zugänglich ist.

Funktionalität. Die Funktionalität der Agenten umfaßt im wesentlichen folgende Punkte:

Interaktion mit dem Benutzer: Die Interaktionen mit dem Benutzer werden über eine grafische Oberfläche bewerkstelligt (Abbildung 15). Hier erstrecken sich die Möglichkeiten des Benutzers vom Ausmachen, Absagen und Verschieben von Terminen, über das Einstellen

von persönlichen Präferenzen innerhalb des Agenten, bis hin zu lokalen Informationsdiensten wie z.B. Übersichten über wöchentliche Stundenpläne oder Zuweisung von Prioritäten an verschiedene Termine. Darüber hinaus ist der Agent an das E-Mail-System des Benutzers angeschlossen mit Hilfe dessen er Informationen über den Erfolg von Terminverhandlungen an ihn weitergibt.

Interaktion mit dem persönlichen Terminkalender des Benutzers: Der Agent verfügt über eine seperate Schnittstelle zum persönlichen Terminkalender des Benutzers. Als Beispielsystem wurde der Calendar Manager CM von Sun Microsystems gewählt. Die Agenten sind aber offen gegenüber anderen Kalendertools. Bei Terminverhandlungen bezieht der Agent diesen Terminkalender mit in die lokale Planung ein. Das heißt, der Agent berücksichtigt einerseits die Termine, die der Benutzer lokal festlegt, also direkt in seinen Kalender schreibt, andererseits werden Termine nach erfolgreicher Verhandlung in diesen eingetragen.

Interaktion mit anderen Agenten: Die Interaktionen mit anderen Agenten finden über direkte Internet Port-Socket Verbindungen statt. Der Agent befolgt dabei die Strategie, die vom Benutzer vorgegeben ist. Verschiedene Strategien stehen dabei zur Verfügung:

Optimistische Strategie: Sofern der Initiator auf eine Einladung nur positive Antworten erhält, wird das Treffen für den vorgeschlagenen Zeitpunkt verabredet, andernfalls storniert.

Iterierende Strategie: Falls der Initiator auf eine Einladung eine Ansage erhält, so verschickt er eine neuen Vorschlag mit einem anderem Zeitpunkt. Dieser Schritt wird solange wiederholt, bis entweder eine Einigung erzielt wurde, oder die Deadline nicht eingehalten werden kann, so daß der Termin storniert werden muss.

Strategie der wechselnden Initiatoren: Sofern ein Partizipient auf eine Einladung mit einer Absage antwortet, wird ihm vom Initiator die gesamte Verhandlung übergeben. Er ist dann für den weiteren Verhandlungsprozess verantwortlich. Dieser Schritt wird ebenso solange wiederholt, bis entweder eine Einigung erzielt wird, oder die Deadline nicht eingehalten werden kann.

7.6.2 Die Erweiterung des Systems zu PASHA-III

Verschiedene technische und konzeptionelle Änderungen waren nötig um das PASHA-II-System an den NL-Server anzupassen. Konzeptionelle Änderungen sind in der Tatsache begründet, daß Kommunikation zwischen maschinellen Agenten stets eindeutig, voraussehbar und deterministisch ist, da die Agenten ein festes Modell voneinander haben. Sie können voraussehen, wie Teilnehmer von Verhandlungen auf bestimmte Situationen reagieren. So war es beispielsweise möglich, einfache, iterierende Verhandlungsstrategien zu entwerfen, ohne Gefahr zu laufen, ein Teilnehmer könnte unvorhersehbar auf einen Vorschlag reagieren. Die Beteiligung menschlicher Partner erfordert die Aufgabe dieser Annahmen. Einerseits können triviale Probleme wie inkonsistente Zeitangaben im Sprechakt des Menschen auftauchen, andererseits können aber auch gänzlich unvorhersehbare Sprechakte in Dialogen auftreten. Ferner wird die Dialogstruktur komplexer, da ein menschlicher Partizipient normalerweise nicht mit einfacher Zustimmung oder Ablehnung reagiert, sondern einen Gegegenvorschlag unterbreitet.

Technische Änderungen ergeben sich zwangsläufig daraus, daß die Kommunikation mit Menschen auf anderem Wege als über eine Internet-Port-Socket-Verbindung erfolgen muß, daß die Kommunikationssprache eine natürliche und keine formale Sprache ist, und – daraus folgend – daß zwischen natürlicher und formaler Sprache übersetzt werden muß. Im folgenden werden diese Änderungen näher beschrieben.

7.6.3 Technische Veränderungen

In PASHA-II wurden Dialoge mit einer agentspezifischen formalen Sprache über eine Internet-Verbindungen geführt. In PASHA-III werden natürlichsprachliche Dialoge über E-Mail geführt,

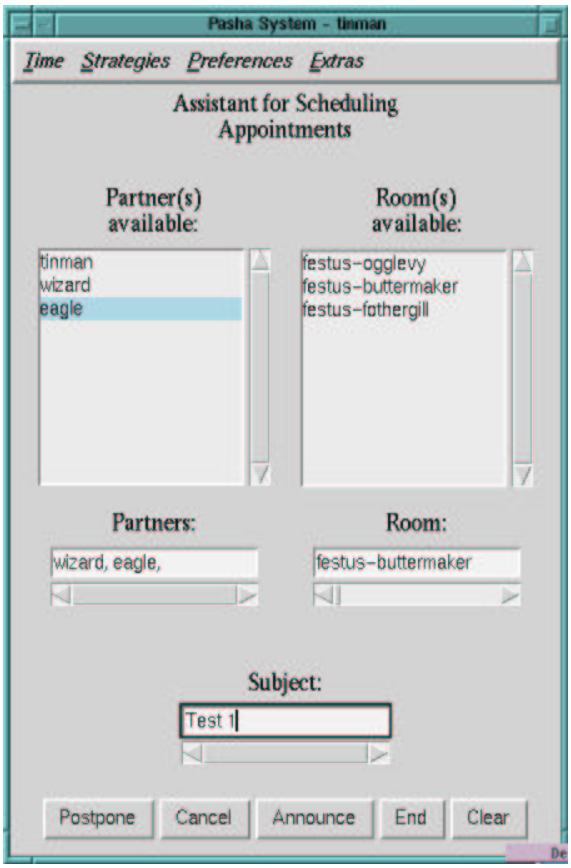

Abbildung 15: Bildschirmabzug der Benutzerschnittstelle von PASHA-II.

sofern es sich beim Absender/Empfänger um einen menschlichen Teilnehmer handelt. Diese neuen Aspekte erforderen die Veränderungen des world interfaces der Agenten; einerseits muß die Art der Verbindung zwischen den Teilnehmern verändert werden, andererseits muß eine Schnittstelle zum NL-Server geschaffen werden.

Schnittstelle zum NL-Server. Die Verbindung zum NL-Server wird mit Hilfe einer Port-Socket-Schnittstelle bewerkstelligt. Der NL-Server besitzt einen Port, an dem sich die Agenten bei Bedarf als Clients anmelden können. Der Server ist daraufhin bereit, den Client seine Dienste zur Verfügung zu stellen. Die Kommunikation zwischen Server und Client unterscheidet sich nach der Aufgabe, die der Agent vom Server gelöst habe möchte:

Analyse: Bekommt der Agent eine Nachricht von einem menschlichen Teilnehmer, so muß dessen natürliche Sprache erst in eine formale, dem Agenten zugängliche Sprache umgewandelt werden. Diese Sprache (IL, siehe Abschnitt 7.4) deckt sich jedoch im allgemeinen nicht mit der formalen Sprache, in der die Agenten untereinander kommunizieren. Somit wurde der PASHA-III Agent mit einem Parser ausgestattet. Innerhalb dieses Parsers werden IL-Ausdrücke nicht nur in die Repräsentationssprache des Agenten übersetzt, sondern die Sprechakte werden mit Hilfe einer Dialoghistorie auch nach semantischen Aspekten geprüft und gegebenenfalls andere Übersetzungsmöglichkeiten vom Server gefordert.

Generierung: Im Falle einer Nachrichtenversendung an menschliche Partner ist es erforderlich, die agenteninterne Repräsentation des Sprechaktes in natürliche Sprache zu übersetzen (siehe Abschnitt 7.5). Dafür werden Templates mit den erforderlichen Daten instantiiert und an

den Server gesendet. Diese Templates decken alle möglichen Sprechakte, die vom Agenten ausgehen können, ab. Der Agent erhält einen deutschen Text, den er dann an den Teilnehmer schicken kann. Nachrichten zwischen Agenten werden natürlich nicht in NL übersetzt.

Verbindung zu anderen Teilnehmern. In PASHA-III kommunizieren Teilnehmer über E-Mail. Nachrichten, die für Agenten bestimmt sind, enthalten dabei das Schlüsselwort "appointment". Diese Nachrichten werden in ein bestimmtes File abgespeichert, das der Agent in bestimmten Zeitabständen nebenläufig überprüft. Sobald eine Nachricht eingetroffen ist, wird der Absender und Inhalt extrahiert. Im umgekehrten Fall werden E-Mails generiert und an die Teilnehmer geschickt.

7.6.4 Anpassung an menschliche Strategien der Terminverhandlung

Während man bei der Modellierung von reinen Agentenszenarien jeden Sprechakt genau klassifizieren und somit Dialoge beschränken kann, treten in gemischten Mensch-Agent-Szenarien unvorhersehbare Situationen auf. Diese kann man einerseits durch eine geeignete Fehlerbehandlung einschränken (vgl. Abschnitt 7.7.2), andererseits kann man jedoch den menschlichen Teilnehmer nicht zu sehr in seiner Expressivität innerhalb eines Dialoges beschneiden, da dies zu unnatürlichen Handlungen zwingt. So muß es im Szenario der Terminvereinbarung für Teilnehmer möglich sein, auf eine Anfrage mit einem Gegenvorschlag zu antworten oder auch eine Anfrage mit unscharfen Zeitangaben zu stellen. Um diesen Forderungen Rechnung zu tragen wurde, eine neue Strategie für die Behandlung von Gegenvorschlägen entwickelt:

1. Stelle eine Anfrage an alle Teilnehmer
2. Wenn alle Teilnehmer zustimmen: lege den Termin fest,
3. Andernfalls antwortet ein Teilnehmer mit einem Gegenvorschlag.
   - wenn der Termin paßt: starte eine Unterverhandlung mit den übrigen Teilnehmern über diesen Termin,
   - andernfalls: gehe zu 1. (oder storniere, falls die Deadline überschritten ist).

Anfragen mit ungenauen Zeitangaben werden hingegen mit einer Darlegung der für ein Treffen möglichen Zeiträume beantwortet, so daß der Initiator sich ein Bild davon machen kann, ob der Termin überhaupt möglich ist.

Das Implementieren der neuen Strategie erfolgte in SvenLog, wobei zusätzliche Verhaltensmuster für Gegenvorschläge und Anbieten von Terminen geschaffen wurden. Durch die Einbeziehung von Menschen in die Verhandlungen selbst wurde es unumgänglich, Möglichkeiten zur nebenläufigen Abarbeitung von verschiedenen Verhandlungen zu schaffen. Dies wurde erleichtert durch die Nebenläufigkeit der Programmiersprache Oz [Smolka, 1995].

7.7 Behandlung von Analyse-Fehlschlägen

Im folgenden werden Mechanismen beschrieben, die es dem System erlauben, bei einem Fehlschlag der Analyse einer natürlichsprachlichen Äußerung angemessen zu reagieren. Viele Arten von Fehlschlägen werden innerhalb des Servers entdeckt und behandelt (Abschnitt 7.7.1), andere jedoch können nur im Agentensystem angemessen bearbeitet werden (Abschnitt 7.7.2).

7.7.1 Behandlung von Analyse-Fehlschlägen im Server

Die Erweiterung der Kompetenz von Software-Agenten bei der verteilten Terminplannung mit über E-Mail kommunizierenden menschlichen Dialogteilnehmern um natürlichsprachliche Dialogfähigkeiten stellt eine zusätzliche potentielle Fehlerquelle dar, die gerade in der Natur dieser Erweiterung liegt (Ambiguitäten, unbekannte Wörter, Tippfehler, Unterspezifikationen, u.ä.).

Aufgabe eines natürlichsprachlichen Servers in dieser Anwendung ist daher, neben den normalen Dialoginteraktionen in natürlicher Sprache auch die Zuverlässigkeit von Dialogschritten gegenüber den Client-Systemen zu garantieren, d.h. mögliche fehlerhafte oder inkonsistente Eingaben mittels Klärungsdialoge mit dem menschlichen Dialogpartner zu korrigieren.

In COSMA wurde eine dialogbasierte Strategie zur Behandlung von Analyse-Fehlschlägen realisiert, die auf dem *detection-signaling-handling*-Prinzip beruht. Nach diesem Prinzip werden Fehlschläge bzw. Fehlerzustände von den Komponenten selbst entdeckt und an eine designierte Fehlerbehandlungskomponente signalisiert. Aufgrund der vorliegenden Informationen initiiert diese einen Prozeß zur Behandlung des Fehlschlags, der einen Klärungsdialog anstößt. Der Klärungsdialog wird in der Regel durch die Generierung von Rückfragen an den menschlichen Teilnehmer initiiert und endet erst, wenn genug Information verfügbar ist, um den Fehlschlag zu reparieren und den unterbrochenen Dialog fortzusetzen.

Für die beiden Systemvarianten von COSMA (Abbildungen 3 und 4) wurden entsprechend verschiedene Varianten der Behandlung von Analyse-Fehlschlägen realisiert. Der Hauptunterschied zwischen beiden Implementationen liegt in dem Modell der Dialogsteuerung, das auch maßgebend für die Steuerung und Durchführung von Klärungsdialogen ist.

Fehlerbehandlung durch Grounding: In der ersten Systemvariante (Abbildung 3) wurde die Dialogsteuerung von der Grounding-Komponente [Hinkelman und Spackman, 1994] effektiv übernommen. Das Modell des Grounding-Prozesses beruht auf endlichen Automaten, deren Übergänge als Dialog-Kontroll-Aktionen definiert sind (z.B. Klärungsanforderung, Klärung, Bestätigung, Beendigung). Diese dienen als Grundlage für klassische Sprechakte (mitteilen, anfordern, fragen).

Die Eingabe für die Grounding-Komponente ist somit eine zentrale Datenstruktur, die alle relevanten Informationen über den gesamten Prozeßverlauf bei der Bearbeitung einer Äusserung (Analyse oder Generierung) im Dialog enthält. Zu diesen Informationen gehören auch Statusberichte sowie Ein- und Ausgaben der aktiven Verarbeitungskomponenten (Scanner, Morphologie, Parsing, Generierung, Sprechakterkennung). Abbildung 16 zeigt diese globale Struktur in Form einer getypten Merkmalstruktur [Krieger und Schäfer, 1994].

Im Falle eines Fehlerberichtes einer Komponente erzeugt Grounding eine Klärungsanforderung. Diese resultiert in einem Klärungsdialog, der asynchron zu dem aktuellen Dialog verläuft.

Diese Art von Fehlerbehandlung lässt sich leicht in einer durch Grounding gestützten Dialogsteuerung integrieren, hat aber gegenüber einer koordinationsbasierten Strategie (s.u.) schwerwiegende Nachteile:

- Schwierige Realisierung der asynchronen Dialogverarbeitung, da Daten des Klärungsdialogs in die zentrale Datenstruktur zu integrieren sind.
- Stark komponentenabhängig, daher wenig allgemein.
- Reduzierte Flexibilität und erhöhter Overhead aufgrund der Integration der relevanten Informationen in eine zentrale Datenstruktur unter einheitlichem Format (getypte Merkmalstruktur).

Koordinationsbasierte Fehlerbehandlung: Diese Strategie setzt eine Fehlerklassifizierung voraus. Zuerst werden für relevante Fehlerzustände Klassen definiert, die alle notwendigen Attribute für die Beschreibung und spätere Behandlung dieser Fehlerzustände enthalten. Beispiele für solche Klassen sind inkonsistente Datumsangaben, vage bzw. falsche Terminangaben, fehlende Information, Antwort auf falsche E-Mail oder unbekannte Wörter.

Jede Fehlermeldung wird durch ein Objekt dargestellt, das eine eindeutige Referenz hat und alle relevanten Informationen für eine spätere Fehlerbehandlung enthält; z.B. enthält eine fehlermeldung für unbekannte Wörter den Zustand der morphologischen Komponente als Detektor, die Liste und Position der unbekannten Wörter im Satz und der Satz selbst.

Jede Klasse von Fehlerzuständen stellt auch entsprechende Methoden für die Behandlung ihrer Objekte zur Verfügung. Dazu zählen Methoden zur Spezifikation und Signalisierung bzw. Bekanntgabe von Fehlerzuständen, sowie Methoden, die zu deren Behandlung beitragen; z.B. Methoden

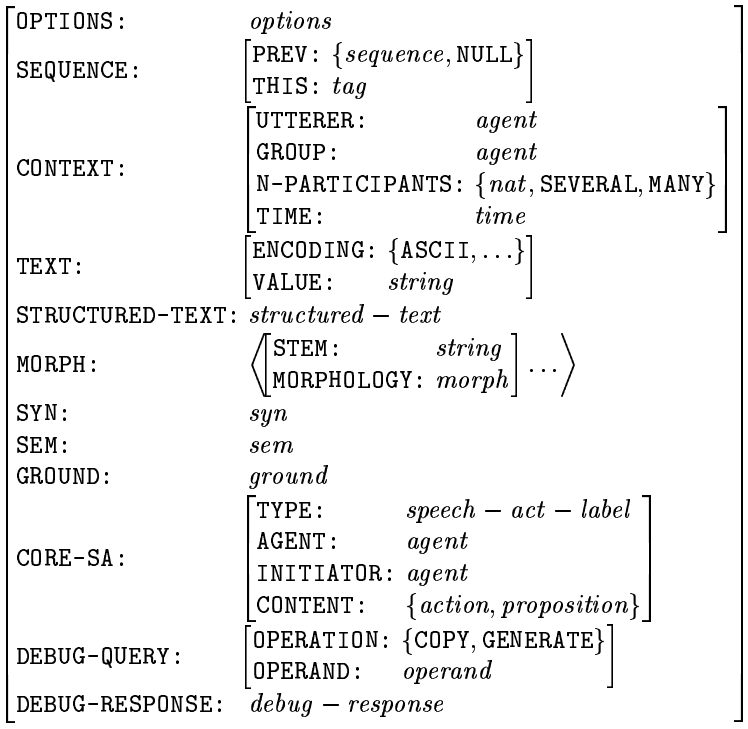

Abbildung 16: Zentrale Datenstruktur zur Bearbeitung von Anfragen eines Anwendungssystems.

zur Extraktion von relevanten Informationen aus den Fehlerzuständen und die Konstruktion von entsprechenden Eingaben für die Generierung von Klärungsanforderungen.

Jeder Fehlerklasse wurde ein eindeutiger Fehlercode zugeordnet. Darüberhinaus wurde ein spezieller Code für den fehlerfreien Zustand definiert (hier auch als Fehlercode bezeichnet). Jede Komponente schickt dem Workflow-Manager zusätzlich zu den berechneten Daten einen Fehlercode. Im Falle eines Fehlschlags stellt die Komponente die entsprechende Fehlermeldung zur Verfügung, die dann für die Fehlerbehandlung eingesetzt wird.

Die hier beschriebene Fehlerbehandlungsstrategie garantiert einen hohen Grad an Flexibilität und Wiederverwendbarkeit. Neue Fehlerklassen lassen sich leicht in die bestehende Klassenhierarchie integrieren, und entsprechende Methoden lassen sich aufgrund der Objekt-Orientierung leicht anpassen und erweitern.

7.7.2 Behandlung von Analyse-Fehlschlägen im Agentensystem

Während Agenten bei der Kommunikation untereinander stets mit korrekten und vollständigen Informationen arbeiten, ist menschliche Kommunikation möglicherweise unvollständig oder inkorrekt. Unvollständige Information wird im Server mit Hilfe des Dialoggedächtnisses weitestgehend vervollständigt. Inkorrekte Informationen können durch Konsistenzprüfungen entdeckt werden. Die gewählte Lösung verbindet die Möglichkeit der Behandlung im Server mit der Behandlung im Agentensystem in der Weise, daß globale Zeitinkonsistenzen (Wochentag paßt nicht zum Datum) vom Server, dialogspezifische Zeitinkonsistenzen (es wurde über eine andere Zeit verhandelt, als die, auf die sich der meschliche Teilnehmer bezieht) vom Agentensystem behandelt werden. Um dialogspezifische Zeitinkonsistenzen zu erkennen, ist es nötig, eine Dialoghistorie zu führen, in der die aktuellen sowie vorherige Verhandlungsdaten abgelegt sind. Diese Dialoghistorie wird von allen teilnehmenden Agentensystemen geführt. Sobald nun eine Zeitinkonsistenz auftritt, wird dem Teilnehmer eine Nachricht mit der Bitte um Berichtigung geschickt.

Wenn ein Sprechakt nicht in die momentane Dialogsituation paßt, versucht das System zunächst

eine interne Lösung zu finden, indem eine erneute Analyse des Sprechaktes vom Server gefordert wird. Falls keine Lösung gefunden werden kann, versendet der Agent eine Nachricht an den Teilnehmer mit dem Hinweis, daß die Nachricht nicht in den aktuellen Dialog paßt. Um einen solchen Fehler zu erkennen, ist es nötig, die Dialoghistorie mit einer Repräsentation aller bisheriger Dialogschritte zu erweitern.

Ein weiterer Fehler tritt bei Nichtbeantwortung von Anfragen auf. Das kann in einer fehlerhaften Datenübertragung begründet liegen, in der Unwilligkeit des menschlichen Teilnehmers, eine Anfrage zu beantworten, etc. In einem solchen Fall versendet der Agent nach Ablauf eines bestimmten Zeitraums Erinnerungsnachrichten.

Es kann vorkommen, daß der menschliche Teilnehmer binnen kürzester Zeit zwei Nachrichten mit einer Einladung für ein- und dasselbe Treffen erhält. Dies passiert, wenn eine Unterverhandlung mit einem anderen Teilnehmer geführt wird, die den Initiator dazu veranlaßt, einen neuen Zeitvorschlag zu generieren. Falls nun der menschliche Teilnehmer auf die veraltete Nachricht antwortet, so verschickt der Initiator eine Bitte auf Beachtung der neuen Nachricht. Erst nachdem eine Antwort auf die aktuelle Nachricht eingetroffen ist, kann der Dialog fortgesetzt werden.

7.8 Graphische Oberfläche

7.8.1 Die Demonstrationsumgebung

Der COSMA NL Server wird in drei verschiedenen Ebenen dargestellt. Dadurch gelingt eine externe Sicht auf den Server und die Kommunikation mit den Clients sowie eine interne Sicht auf die Funktionsweise der Servers. Die oberste Ebene ist in Abbildung 17 dargestellt. Sie zeigt das gesamte

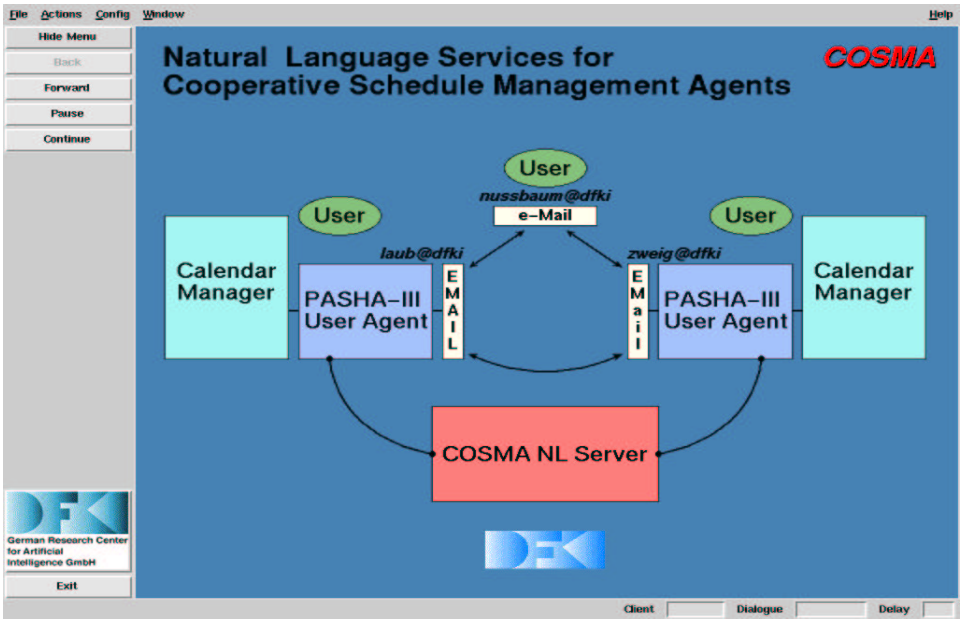

Abbildung 17: Die Demonstrationsumgebung von COSMA

Szenario für zwei Clients und einen natürlichsprachlichen Benutzer. Man kann auf dieser Ebene die Kommunikation zwischen den beteiligten Dialogpartnern verdeutlichen. Die externe Sichtweise auf das System zeigt Abbildung 18). Sie zeigt erst nur den Server in der unteren Hälfte des Bildschirms. Melden sich Clients an, so werden auch diese angezeigt. Befindet man sich auf dieser Ebene, so kann man auch die Kalendermanager (Unix `dtcm`) der Clients und ein E-Mail-Frontend (Unix `exmh`) sehen. Die Fenster dieser Prozesse sind außerhalb der Demonstrationsumgebung und müssen von Hand in der oberen Bildschirmhälfte angeordnet werden.[3] Während einer Termin-

[3]Die Fenster für die Agentensysteme sind nicht gezeigt.

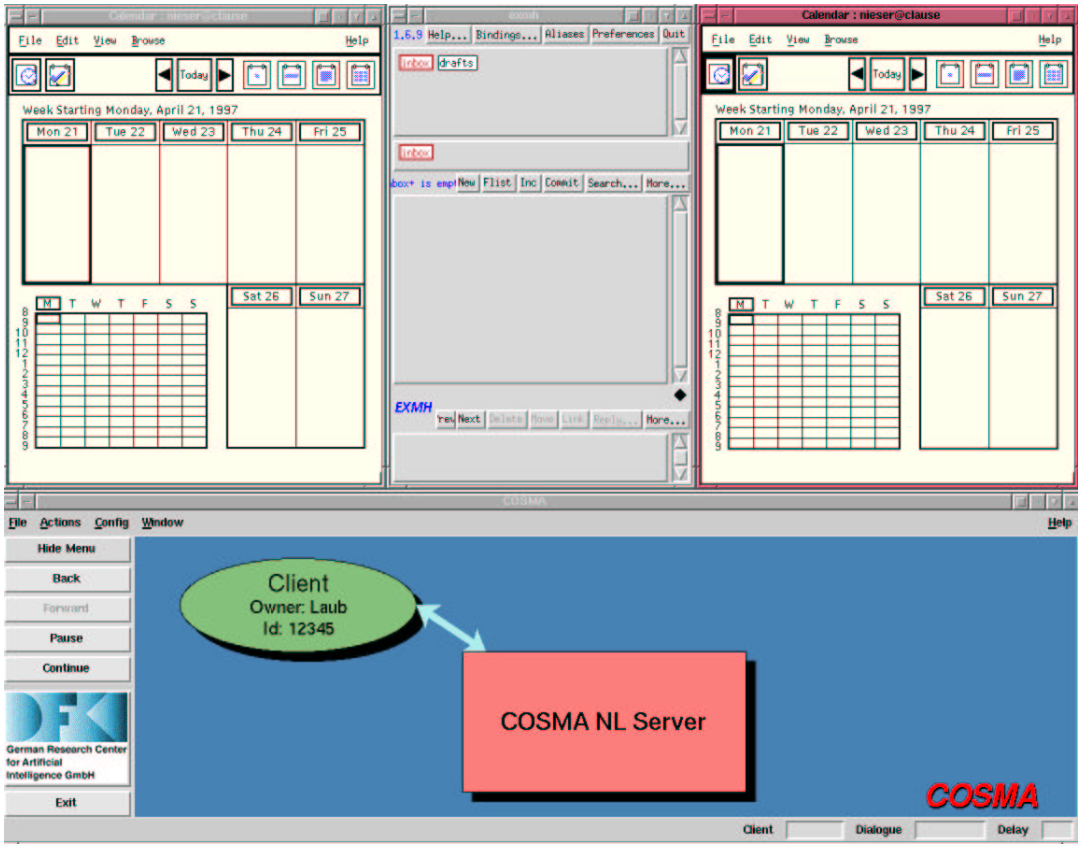

Abbildung 18: Die Funktionalitätsebene mit den beiden Kalendermanagern und dem E-Mail-Frontend. Ein Client hat sich am Server angemeldet.

absprache werden vorläufige und endgültige Termine in den Kalendern eingetragen. Die unterste Ebene ist die COSMA-Systemebene (Abbildung 19). In dieser internen Sichtweise sind die einzelnen Komponenten des Servers dargestellt. Man kann bei einem Durchlauf sehen, wie diese aktiviert werden. Die Aktivierung einer Komponente ist durch Verfärbung der entsprechenden Rechtecke und Pfeile dargestellt. Auf dieser Ebene gibt es auch die Möglichkeit, die Ein- und Ausgaben der einzelnen Komponenten durch Maus-Klicks an den Pfeilen zu visualisieren.[4]

7.8.2 Allgemeine Funktionsweise

Die Demonstrationsumgebung wird durch den Server gestartet. Zunächst wird durch Unix `xv` eine Graphik geladen, die den Schriftzug "COSMA" und den Projektnamen als Blickfang für System-Demonstrationen zeigt. Um zur Demonstrationsumgebung zu gelangen, verläßt man `xv` durch Eingabe von `q`. Die Umgebung unterteilt sich in vier Teile: das Display, die Leiste mit den Schaltflächen, die Statuszeile und das Menü.

Das Display. Im Display können zwei Arten von Ebenen dargestellt werden. Zum einen sind dies einfache Folien, die nicht vom Server angesprochen werden und zum anderen Diagramme, deren einzelne Objekte verschiedene, vom Server gesteuerte Zustände annehmen können. Diese Diagramme können sensitiv sein, d.h. einzelnen Objekten ist eine Aktion zugeordnet. Ob ein Objekt sensitiv ist, erkennt man daran, daß der Mauszeiger seine Form über dem Objekt ändert. Je nachdem, welche Aktion dem Objekt zugeordnet ist, kann man mit der linken Maustaste dem

[4]Diese Möglichkeit erwies sich für Tests und Fehlerbehandlung als nützlich.

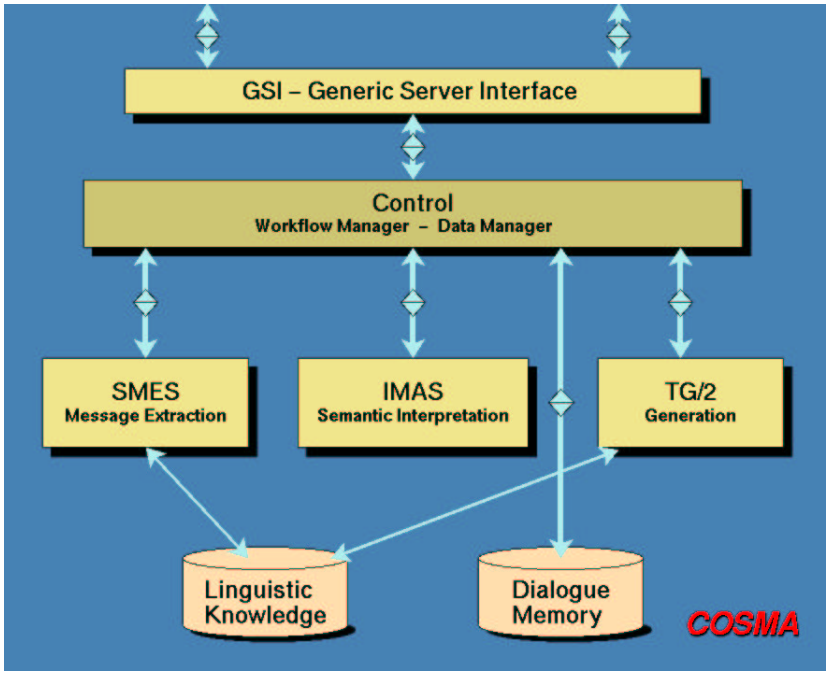

Abbildung 19: Die COSMA-Systemebene

Server den Auftrag geben, Daten anzuzeigen oder zu einer anderen Ebene gelangen. Sind mit einem Objekt mehrere Aktionen assoziiert, so erscheint ein Pop-Up-Menü mit den gegebenen Möglichkeiten. Handelt es sich wieder um ein Diagramm, so kann man das ausgewählte Objekt mit der rechten Maustaste als eine Miniatur des assoziierten Diagramms darstellen. Diese Miniatur ist nicht sensitiv, aber alle Zustandsänderungen der Objekte werden angezeigt. Durch einen erneuten Klick mit der rechten Maustaste kann man das ursprüngliche Objekt wiederherstellen.

Die Leiste mit den Schaltflächen. Die oberste Schaltfläche ist selbsterklärend, er zeigt oder versteckt die Menüleiste. Während man mit der Demonstrationsumgebung arbeitet, wandert man durch verschiedene Ebenen. Der zurückgelegte Weg wird auf einem Stapel gespeichert. Die "Back"-Schaltfläche gibt die Möglichkeit, wieder in höhere Ebenen zu gelangen, dabei wird die letzte Ebene von dem Stapel genommen. Hat keines der Objekte einen Verweis auf eine darunter liegende Ebene, aber die aktuelle Ebene einen Nachfolger, so wird die "Forward"-Schaltfläche, über die man dorthin gelangen kann, aktiviert. Das Programm legt keinen Forward-Stapel an, wie es manche WWW-Browser machen. Die "Pause"- und die "Continue"-Schaltfläche geben dem Server die Aufforderung, bei nächster Gelegenheit zu halten (i.a. nach einem vollständigen Durchlauf eines Client-Auftrags). Während einer Pause kann man sich Zwischenergebnisse des soeben beendeten Auftrags anzeigen lassen. Die "Exit"-Schaltfläche schickt dem Server die Aufforderung, den Prozess für die Demonstrationsumgebung zu beenden. Klickt man das DFKI-Logo an, so erscheint der COSMA-Blickfang.

Die Statuszeile. Die Felder "Dialogue" und "Client" zeigen an, womit der Server zur Zeit beschäftigt ist. Im "Delay"-Feld kann man eine Wartezeit angeben, die der Server zwischen jeder Aktivierung eines Objekts einlegt. Während der Server arbeitet, zählt er im "Delay"-Feld die gesamte Wartezeit hoch. Dies dient dazu, die Verarbeitungsgeschwindigkeit herabzusenken, um den Kontrollfluß visualisieren zu können.

Das Menü. Unter "Window/Geometry" kann man ein Dialogfeld erreichen, mit dem man die maximale Größe des Fensters festlegen kann. Die Angabe erfolgt in Pixeln. Läßt man ein Feld leer,

so wird der maximal mögliche Wert genommen. Die "OHP"-Schaltfläche übernimmt die Werte, die in einer zentralen Konfigurationsdatei für die Benutzung eines Panels in Verbindung mit Overhead-Projektoren eingestellt wurden. Im Eintrag "Config" kann man die Farben der Diagramme einstellen und die Schatten der Objekte ein- und ausschalten. Keine der vorgenommenen Menü-Einstellungen wird gespeichert.

8 Voraussichtlicher Nutzen und Verwertbarkeit der Ergebnisse

Die Projektergebnisse umfassen wissenschaftliche Resultate und Software-Prototypen. Für das Forschungsgebiet der Multi-Agenten-Systeme zeigen sie neuartige Methoden der Agentenkommunikation mithilfe natürlicher Sprache auf, die die Einbeziehung menschlicher Agenten in die Kommunikation zweischen maschinellen Agenten erlaubt. Erstmalig wurden hierfür vollständig implementierte Systeme entwickelt, die dies für den Bereich der Terminabsprachen beispielhaft umsetzen.

Für das Forschungsgebiet der sprachverarbeitenden KI stellen die Ergebnisse einen signifikanten Fortschritt in der Kombination unterschiedlicher Technologien für anwendungsorientierte natürlichsprachliche Dialogsysteme dar.

Die Entwicklung einer generischen, objektorientierten Architektur, die an unterschiedliche Anforderungen leicht anpaßbar ist, bedeutet eine erhebliche Vereinfachung beim Design und der Implementation komplexer Softwaresysteme. Sie wird u.a. im BMBF-Projekt PARADIME (FKZ ITW 9704) zum Einsatz kommen.

Neben dem wissenschaftlichen und technischen Nutzen ergeben sich durch die Software-Prototypen aus COSMA konkrete praktische Anwendungsmöglichkeiten. Präsentationen der Systeme auf der Computermesse CeBIT (1993 und 1997) und Gespräche mit wichtigen Herstellern und Vertreibern von Kalendermangement-Systemen haben ergeben, daß für das COSMA-System als Ganzes in der Zukunft Weiterentwicklungsmöglichkeiten zu einem entsprechenden Produkt bestehen. Dies wurde durch eine Untersuchung des deutschsprachigen Marktes von Kalenderverwaltungs-Systemen durch das COSMA-Projekt untermauert [Busemann und Merget, 1995].

Das modulare Design der Softwarekomponenten in COSMA ermöglicht zudem Anwendugsperspektiven für Teile des Systems. Die oberflächennahe Generierungskomponente TG/2 [Busemann, 1996; Wein, 1996] wird in dem EU-Projekt TEMSIS zur Generierung von Luftgüteberichten wiederverwendet (Telemantics Applications, Environment Sector C9, Nr. 2945).

9 Fortschritte bei anderen Stellen

COSMA hat auf die wichtigen Entwicklungen im bereich Sprachtechnologie unmittelbar reagiert. Durch die neuartige Kombination modernster Technologien hat das Projekt selbst Maßstäbe gesetzt.

Im Verlauf des Projekts wurde durch die Entwicklung oberflächennaher natürlichsprachlicher Analysekomponenten einerseits und durch die zunehmend corpusbasierte Ausrichtung von Verarbeitungsstrategien andererseits eine Trendwende im Bereich Sprachtechnologie offensichtlich. Der Anspruch des möglichst vollständigen maschinellen Verstehens natürlichsprachlichen Diskurses wurde zugunsten der Extraktion relevanter Information aus Diskursen zurückgestellt. Dies ergab eine sprunghaft angestiegene Erfolgsquote beim Textverstehen; gleichzeitig mußte dabei inkauf genommen werden, daß Texte nur unvollständig analysierbar waren.

Das Projekt COSMA sah in der neuen Methodologie eine bessere Chance zur Erreichung seiner Ziele und ersetzte die aus Vorgängerprojekten übernommene HPSG-basierte Kernmaschine durch das Shallow-Parsing-System smes in Verbindung mit der Eigenentwicklung IMAS.

Auch im Gebiet der Generierung wurden oberflächennahe Verfahren aufgewertet. Templatebasierte Generierungsansätze waren früher durch mangelnde Modellierbarkeit der sprachlichen Variablilität und der kommunikativen Situationen in Verruf geraten. Während der Laufzeit von

COSMA wurden eine Reihe von Generierungssystemen auf den Markt gebracht (in den USA, in Canada und in Frankreich), die alle durch eine speziell auf die jeweilige Anwendungssituation ausgerichtete Verarbeitungsstrategie und durch eine eher rudimentäre linguistische Komponente charakterisiert sind.

Mit dem in COSMA entwickelten Generator TG/2 steht eine flexible Komponente zur Verfügung, die von ihrer Konzeption her mit auf dem Markt befindlichen Systemen vergleichbar ist.

10 Erfolgte oder geplante Veröffentlichung der Ergebnisse

Die Projektergebnisse wurden fortlaufend dokumentiert und auf einschlägigen Konferenzen sowie in Technischen Berichten publiziert. Die folgende Übersicht stellt die Projektveröffentlichungen zusammen.

Literatur

[Busemann *et al.*, 1994] Stephan Busemann, Stephan Oepen, Elizabeth Hinkelman, Günter Neumann und Hans Uszkoreit. COSMA–Multi-Participant NL Interaction for Appointment Scheduling. Technical Report RR-94-34, DFKI, Saarbrücken, 1994.

[Busemann *et al.*, 1997] Stephan Busemann, Thierry Declerck, Abdel Kader Diagne, Luca Dini, Judith Klein und Sven Schmeier. Natural Language Dialogue Service for Appointment Scheduling Agents. In *Proc. 5th Conference on Applied Natural Language Processing*, Washington, DC., Seite 25–32. 1997.

[Busemann und Merget, 1995] Stephan Busemann und Iris Merget. Eine Untersuchung kommerzieller Terminverwaltungs-Software im Hinblick auf die Kopplung mit natürlichsprachlichen Systemen. Technical Document D-95-11, DFKI, Saarbrücken, September 1995.

[Busemann, 1995] Stephan Busemann. Towards Classification of Generation Subtasks. In Wolfgang Hoeppner und Helmut Horacek (Hg.), *Principles of Natural Language Generation. Papers from a Dagstuhl-Seminar*, Seite 25–32. University of Duisburg, Report SI-12, February 1995.

[Busemann, 1996] Stephan Busemann. Best-first Surface Realization. In Donia Scott (Hg.), *Eighth International Natural Language Generation Workshop. Proceedings*, Herstmonceux, Univ. of Brighton, England. 1996. Also available as Research Report RR-96-05, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, Germany.

[Busemann, 1997] Stephan Busemann. Putting Semantic-Head-Driven Generation to the Limits: Experiments with multi-purpose semantic representations. In *DFKI Workshop on Natural Language Generation*. Saarbrücken, 1997. Technical Document D-97-xx.

[Declerck und Klein, 1997] Thierry Declerck und Judith Klein. Semantic Tagging and NLP Applications. In *Proceedings of the Workshop "Tagging Text with Lexical Semantics: Why, What, and How?"*, Washington D.C. April 1997.

[Declerck und Maas, 1997] Thierry Declerck und Heinz-Dieter Maas. The integration of a part-of-speech tagger into the ALEP platform. In IAI (Hg.), *Proc. 3rd ALEP User Group Workshop*, Saarbrücken. March 1997.

[Declerck, 1996a] Thierry Declerck. An Interface between Text Structure and Linguistic Description. In *Proc. 7th Computational Linguistics in the Netherlands Meeting*, Eindhoven. November 1996.

[Declerck, 1996b] Thierry Declerck. Dealing with cross-sentential anaphora resolution in ALEP. In *Proceedings of COLING-96*, Kopenhagen, Seite Vol. 1: 280–285. 1996.

[Declerck, 1996c] Thierry Declerck. Lean formalisms, linguistic theory, and applications. Grammar development in ALEP. In *Proceedings of COLING-96*, Kopenhagen, Seite Vol. 1: 286–291. 1996.

[Hinkelman und Spackman, 1994] Elizabeth A. Hinkelman und Stephen P. Spackman. Communicating with Multiple Agents. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94*, Kyoto, Japan. 1994.

[Hinkelman, 1996] Elizabeth A. Hinkelman. Conversation Grounding in Paradice. In *5th International Pragmatics Conference*, Mexico City, Juni 1996. 1996.

[Klein *et al.*, 1997] Judith Klein, Stephan Busemann und Thierry Declerck. Diagnostic Evaluation of Shallow Parsing Through an Annotated Reference Corpus. In *Proc. Workshop on Evaluation in Speech and Language Technology (SALT)*, University of Sheffield, UK. June 1997.

[Schmeier und Schupeta, 1996] Sven Schmeier und Achim Schupeta. PASHA II – A Personal Assistant for Scheduling Appointments. In *Proc. 1st Conference on Practical Application of Multi Agent Systems*, London. April 1996.

[Spackman, 1995] Stephen P. Spackman. The Natural Language Server Application Program Interface. DFKI, 1995.

Literatur

[Allen, 1983] James Allen. Maintaining Knowledge About Temporal Intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[Amtrup, 1995] Jan Willers Amtrup. ICE—INTARC Communication Environment. Users Guide and Reference Manual. Version 1.4. Technical Document 14, Verbmobil, Dezember 1995.

[Appelt *et al.*, 1993] Doug Appelt, Jerry Hobbs, John Bear, David Israel und Mabri Tyson. A finite state processor for information extraction from real-world text. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chamb'ery. 1993.

[Bond und Gasser, 1988] A. Bond und L. Gasser. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann, Los Angeles, CA, 1988.

[Booch, 1994] Grady Booch. *Object-Oriented Analysis and Design with Applications.* Benjamin/Cummings, Menlo Park, 1994.

[Busemann und Merget, 1995] Stephan Busemann und Iris Merget. Eine Untersuchung kommerzieller Terminverwaltungs-Software im Hinblick auf die Kopplung mit natürlichsprachlichen Systemen. Technical Document D-95-11, DFKI, Saarbrücken, September 1995.

[Busemann, 1996] Stephan Busemann. Best-first Surface Realization. In Donia Scott (Hg.), *Eighth International Natural Language Generation Workshop. Proceedings*, Herstmonceux, Univ. of Brighton, England. 1996. Also available as Research Report RR-96-05, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, Germany.

[Cohen und Levesque, 1990a] P. R. Cohen und Levesque. Persistence, Intention, and Commitment? In Cohen, Morgan und Pollack (Hg.), *Intentions in Communication*, Kapitel 3, Seite 33–69. 1990.

[Cohen und Levesque, 1990b] Phillip R. Cohen und Hector J. Levesque. Persistence, Intention, and Commitment. In P. R. Cohen, J. Morgan und M. E. Pollack (Hg.), *Intentions in Communication.* MIT Press, Cambridge MA, 1990.

[Cohen und Perrault, 1979] P. R. Cohen und C. R. Perrault. Elements of a Plan-Based Theory of Speech Acts. *Cognitive Sciences*, 3(3), 1979.

[Davis und King, 1977] Randall Davis und Jonathan King. An Overview of Production Systems. In E. W. Elcock und D. Michie (Hg.), *Machine Intelligence 8*, Seite 300–332. Ellis Horwood, Chichester, 1977.

[Dent *et al.*, 1992] Lisa Dent, Jesus Boticario, John McDermott, Tom Mitchell und David Zabowski. A Personal Learning Apprentice. In *Proceedings of the National Conference on Artificial Intelligence*, 1992.

[Durfee *et al.*, 1989] E.H. Durfee, V.R. Lesser und D.D. Corkill. Trends in Cooperative Distributed Problem Solving. *IEEE Transaction on Knowledge and Data Engineering*, 1(1), 1989.

[Estival *et al.*, 1995] Dominique Estival, Sabine Lehmann, Kirstin Falkedal, Hervé Compagnion, Lorna Balkan, Frederik Fouvry, Doug Arnold, Eva Dauphin, Sylvie Régnier-Prost, Véronika Lux, Judith Klein, Judith Baur, Klaus Netter und Stephan Oepen. The Construction of Test Material. Report to LRE 62-089, University of Essex, 1995.

[Estival *et al.*, 1996] Dominique Estival, Sabine Lehmann, Kirstin Falkedal, Hervé Compagnion, Lorna Balkan, Frederik Fouvry, Doug Arnold, Eva Dauphin, Sylvie Régnier-Prost, Véronika Lux, Judith Klein, Judith Baur, Klaus Netter und Stephan Oepen. TSLNP – Test Suites for Natural Language Processing. In *Proceedings of the 16th International Conference on Computational Linguistics, COLING-96*, Copenhagen, Seite 711–716. 1996.

[Finkler und Neumann, 1988] Wolfgang Finkler und Günter Neumann. MORPHIX: A Fast Realization of a Classification–Based Approach to Morphology. In H. Trost (Hg.), *Proceedings der 4. Österreichischen Artificial–Intelligence Tagung, Wiener Workshop Wissensbasierte Sprachverarbeitung*, Berlin, Seite 11–19. Springer, August 1988.

[Goldman und Rosenschein, 1993] C. V. Goldman und J. S. Rosenschein. Emergent Coordination through the Use of Cooperative State-Changing Rules. In *Proceedings of the Twelfth International Workshop on Distributed Artificial Intelligence (IWDAI-93)*, Hidden Valley, PA, Seite 171–186. Mai 1993.

[Grishman und Sundheim, 1996] Ralf Grishman und Beth Sundheim. Message Understanding Conference-6: A Brief History. In *Proceedings of the 16th International Conference on Computational Linguistics, COLING-96*, Copenhagen, Seite 466–471. 1996.

[Hinkelman und Spackman, 1994] Elizabeth A. Hinkelman und Stephen P. Spackman. Communicating with Multiple Agents. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94*, Kyoto, Japan. 1994.

[Kiefer und Fettig, 1995] Bernd Kiefer und Thomas Fettig. FEGRAMED—An Interactive Graphics Editor for Feature Structures. Report RR-95-06, DFKI, Saarbrücken, 1995.

[Koomen, 1989] Johannes A.G.M. Koomen. *Reasoning About Recurrence*. Doktorarbeit, University of Rochester, 1989.

[Krieger und Schäfer, 1994] Hans-Ulrich Krieger und Ulrich Schäfer. *TDL*—A Type Description Language for Constraint-Based Grammars. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94*, Kyoto, Japan. 1994.

[Lamport, 1977] Leslie Lamport. The Synchronization of Independent Processes. *Acta Informatica*, 7:15–34, 1977.

[Lehmann *et al.*, 1996] Sabine Lehmann, Stephan Oepen, Sylvie Regnier-Prost, Klaus Netter, Veronika Lux, Judith Klein, Kirsten Falkedal, Frederik Fouvry, Dominique Estival, Eva Dauphin, Hervé Compagnion, Judith Baur, Lorna Balkan und Doug Arnold. TSNLP — Test Suites for Natural Language Processing. In *Proceedings of COLING 1996*, Kopenhagen, Seite 711 – 716. 1996.

[Lux *et al.*, 1992] Andreas Lux, Frank Bomarius und Donald Steiner. A Model for Supporting Human Computer Cooperation. In *AAAI Workshop on Cooperation among Heterogeneous Intelligent Systems*, 1992.

[Lux, 1992] Andreas Lux. A Multi-Agent Approach Towards Group Scheduling. Research Report RR-92-41, DFKI, Kaiserslautern, Germany, 1992.

[Malone und Crowston, 1991] Thomas W. Malone und Kevin Crowston. Toward an Interdisciplinary Theory of Coordination. Technical Report CCS TR 120, Center for Coordination Science, Sloan School of Management, MIT, MIT, Cambridge, MA, 1991.

[Malone, 1987] T. W. Malone. Modelling Coordination in Organizations and Markets. *Management Science*, 33:1317–1332, 1987.

[Mattern und Sturm, 1989] F. Mattern und P. Sturm. An Automated Distributed Calendar and Appointment System. Technical Report SFB124-24/89, University of Kaiserslautern, Department of Computer Science, Kaiserslautern, Germany, 1989.

[Müller und Pischel, 1993] Jürgen Müller und Markus Pischel. The Agent Architecture InteRRaP: Concept and Application. Research Report RR-93-26, DFKI, Saarbrücken, Germany, 1993.

[Nerbonne *et al.*, 1991] John Nerbonne, Klaus Netter, Abdel Kader Diagne, Ludwig Dickmann und Judith Klein. A Diagnostic Tool for German Syntax. In J. G. Neal und S. M. Walter (Hg.), *Natural Language Processing Systems Evaluation Workshop: Final Technical Report RL-TR-91-362*, Seite 79–96. Rome Laboratory, Griffiss Air Force Base, NY, 1991. Also available as Research Report RR-91-18, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, Germany.

[Neumann *et al.*, 1997] Günter Neumann, Rolf Backofen, Judith Baur, Markus Becker und Christian Braun. An Information Extraction Core System for Real World German Text Processing. In *Proc. 5th Conference on Applied Natural Language Processing*, Washington, DC. 1997.

[Neumann, 1993] Günter Neumann. Design Principles of the DISCO System. In *Proceedings of the Twente Workshop on Language Technology (TWLT5)*, University of Twente. 1993.

[Neumann, 1997] Günter Neumann. Methoden zur intelligenten Informationsextraktion im Internet. In *Proceedings of the 20th European Congress Fair of Technical Communications*, Hamburg. 1997. (forthcoming).

[Rosenschein *et al.*, 1986] J. S. Rosenschein, M. Ginsberg und M. R. Genesereth. Cooperation Without Communication. In *AAAI86*, AAAI86Addr. 1986.

[Schmeier und Schupeta, 1996] Sven Schmeier und Achim Schupeta. PASHA II – A Personal Assistant for Scheduling Appointments. In *Proc. 1st Conference on Practical Application of Multi Agent Systems*, London. April 1996.

[Sen und Durfee, 1992] S. Sen und E.H. Durfee. A Formal Analysis of Communication and Commitment in Distributed Meeting Scheduling. In *11th Int. Workshop on DAI*, Homestead, Glenn Arbor, Seite 333 – 344. Feb. 1992.

[Smolka, 1995] Gert Smolka. The Oz Programming Model. Research Report RR-95-10, DFKI, Saarbrücken, Germany, 1995.

[Spackman, 1995] Stephen P. Spackman. The Natural Language Server Application Program Interface. DFKI, 1995.

[Traum und Hinkelman, 1992] David Traum und Elizabeth A. Hinkelman. Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 8(3):575 – 599, 1992. Also available as Research Report RR-93-32, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, Germany.

[Uszkoreit *et al.*, 1994] Hans Uszkoreit, Rolf Backofen, Stephan Busemann, Abdel Kader Diagne, Elizabeth A. Hinkelman, Walter Kasper, Bernd Kiefer, Hans-Ulrich Krieger, Klaus Netter, Günter Neumann, Stephan Oepen und Stephen P. Spackman. DISCO–An HPSG-based NLP System and its Application for Appointment Scheduling. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94*, Kyoto, Japan. 1994.

[Wein, 1996] Michael Wein. Eine parametrisierbare Generierungskomponente mit generischem Backtracking. Diplomarbeit, Department for Computer Science, University of the Saarland, 1996.

[Yokoo und Durfee, 1991] M. Yokoo und E. Durfee. Distributed Constraint Optimization as a Formal Model of Partially Adversarial Cooperation. In *Proc. of the American Workshop on DAI*, 1991.

[Zlotkin und Rosenschein, 1990] G. Zlotkin und J. S. Rosenschein. Negotiation and Conflict Resolution in Non-cooperative Domains. In *Proc. of the National Conference on Artificial Intelligence*, Boston, Massachusetts, Seite 100–105. The American Association for Artificial Intelligence, 1990.