D5.2: Baseline System for In-Car Showcase

Tilman Becker, Nate Blaylock, Ciprian Gerstenberger,
Michael Pitz, Peter Poller, Jan Schehl

Distribution: Public

TALK
Talk and Look: Tools for Ambient Linguistic Knowledge
IST-507802 Deliverable 5.2

September 5, 2005

Project funded by the European Community
under the Sixth Framework Programme for
Research and Technological Development

The deliverable identification sheet is to be found on the reverse of this page.

Project ref. no.	IST-507802
Project acronym	TALK
Project full title	Talk and Look: Tools for Ambient Linguistic Knowledge
Instrument	STREP
Thematic Priority	Information Society Technologies
Start date / duration	01 January 2004 / 36 Months

Security	Public
Contractual date of delivery	M18 = June 2005
Actual date of delivery	September 5, 2005
Deliverable number	5.2
Deliverable title	D5.2: Baseline System for In-Car Showcase
Type	Report
Status & version	Final 1.0
Number of pages	25 (excluding front matter)
Contributing WP	5
WP/Task responsible	BMW
Other contributors	USAAR, DFKI, BOSCH
Author(s)	Tilman Becker, Nate Blaylock, Ciprian Gerstenberger, Michael Pitz, Peter Poller, Jan Schehl
EC Project Officer	Evangelia Markidou (Anne Bajart)
Keywords	Dialogue Systems, In-Car Dialogue, Dialogue Modeling, Natural Language Generation, Information-State Update, Multimodal Interfaces

The partners in TALK are:

Saarland University	USAAR
University of Edinburgh HCRC	UEDIN
University of Gothenburg	UGOT
University of Cambridge	UCAM
University of Seville	USE
Deutches Forschungszentrum fur Künstliche Intelligenz	DFKI
Linguamatics	LING
BMW Forschung und Technik GmbH	BMW
Robert Bosch GmbH	BOSCH

For copies of reports, updates on project activities and other TALK-related information, contact:

The TALK Project Co-ordinator
Prof. Manfred Pinkal
Computerlinguistik
Fachrichtung 4.7 Allgemeine Linguistik
Postfach 15 11 50
66041 Saarbrücken, Germany
pinkal@coli.uni-sb.de
Phone +49 (681) 302-4343 - Fax +49 (681) 302-4351

Copies of reports and other material can also be accessed via the project's administration homepage, `http://www.talk-project.org`

Contents

Summary

As a “Prototype”, Deliverable D5.2 consists of the baseline system, i.e., the software as set up at the developers (DFKI and USAAR) as well as at the industrial partners BMW and Bosch. The latter will also run the baseline system for the evaluation experiments.

The software is attached to this document in Appendix A on a CD-ROM, including hardware requirements and instructions for installation and execution.

Finally, this documents attempts to make the actual deliverable itself more useful and accessible by adding a description of purpose, scope, and some technical details of the baseline systems and its components.

Chapter 1

Introduction

This document describes the in-car baseline system which has been jointly developed by DFKI and USAAR (in collaboration with BMW and Bosch). The system focuses mostly on the research issues these partners are involved in (work packages 2 and, primarily, 3). It is important to note that in addition to this system, other baseline systems are being built by UEDIN and UCAM, see deliverable 4.2, and various GOT systems, see status report on task 1.6. However, this deliverable covers only the DFKI/USAAR system.

This baseline system serves several purposes within the project. First, it will provide a testbed for select multimodal experiments conducted in work package 6. These experiments will target specific multimodal research issues and then feed development of the final in-car showcase system (deliverable D5.3) which will be evolved from the baseline system. In addition, the baseline system and its evaluation (in deliverable D6.3) will provide a measure for evaluating the improvements in various research categories for the final showcase.

In this sense, the baseline system has been constructed partly using baseline versions of new technology being developed in the project, and where appropriate, existing (i.e., pre-TALK) technology, e.g., for speech recognition and synthesis.

The remainder of this document is structured as follows: Chapter 2 discusses the domain and use-cases chosen for the system. Then Chapter 3 presents the overall architecture of the baseline system and describes the individual components. Chapter 2.2 shows some example exchanges with the system and finally, Chapter 4 discusses future plans for the system, looking towards the final in-car showcase (deliverable 5.3).

The first version of the baseline system software was delivered on June 30th, 2005. Since then, a number of updates are following to address issues that are necessary for the evaluation of the baseline system to be conducted in October and November of 2005. Some of these functionalities are already included in this document.

Chapter 2

System Domain and Use-Cases

In collaboration with our industrial partners (BMW and Bosch), we selected the MP3 player domain. In the baseline system, a user, i.e., the driver, can interact multi-modally with an in-car MP3 player, which supports the usual functionalities, including, e.g., song playback and playlist creation and modification. Full specifications of the functionalities are given in section 2.1.

In addition, we use the *Lane Change Task* (LCT) software[1] which simulates the primary task of driving on an additional PC. The LCT task is under standardization as (ISO TC 22/SC 13/WG 8). This allows us to create an environment where the user is both mentally occupied with driving, but also simulates attentional distraction with respect to the multimodal input and output of the system. Figure 2.1 shows the setup with two screens with LCT running on the left screen.[2] The final version of this system will be integrated by BMW into a car.

There are two possible types of MP3 players which we considered for the baseline system. The first is the "typical" domain where a user has a (small) private music collection which is being used. The second, more recent, type is access to a large, online song download engine (such as Apple's iTunes music store[3]) which give a user access to a very large (typically more than 1 million tracks, depending on the country) music database, in addition to his own private collection.

Based on input from our automotive industrial partners (BMW and Bosch), we have opted for the first, more typical domain for the in-car baseline system. We do, however, plan to cover the second domain in our in-home version of the MP3 system.

To simulate such a large music database (and to get data for our smaller, personal user databases), we have previously, e.g., for the Wizard of Oz experiments, used FreeDB[4], a large, freely-available database of album information (it does not include the actual copyrighted songs, just information about them, including title, tracks, artist, length, etc.) This allows us to support searches on a large music database. Music playback is simulated by the system, as the actual tracks typically are not available (for copyright reasons).

The baseline system will support dialogue in German.

[1] LCT was supplied by BMW, but was originally developed at Mercedes, see [Mat03].

[2] The picture is taken from our WoZ experiments, thus the ErgoCommander is missing in this setup.

[3] http://www.apple.com/itunes/store

[4] http://www.freedb.org

Figure 2.1: LCT (lane change task), the primary task.

2.1 Supported System Functionality

The current version of the baseline system supports functionality as outlined in the following. Note that all of these features can be controlled either by speech only (monomodal), and where it was possible and reasonable, also by speech together with the ErgoCommander controller (multimodal). Due to our focus on multimodal/speech-driven communication, the monomodal usage of the ErgoCommander is restricted to the *command and control* functionality and also browsing of displayed sets/lists.

System output is multimodal in almost all situations, including a spoken message, information on the display such as text, lists or tables, and the player controls. Additional system output controlling the interaction is also multimodal, i.e., acoustic and visual feedback about microphone (and thus system) status.

1. **Command and control:** Typical functions of a media player like
   - play/stop/pause
   - select next/previous track
   - skip to track x of loaded album/playlist
   - turn on/off shuffle mode
2. **Play playable objects:**
   - Play a particular song/album/playlist
     i.e., User: ‘Spiele Yesterday von den Beatles.’ (*Play Yesterday by The Beatles.*)

- Play random song/album/playlist
  i.e., U: 'Spiele einen Rap Song von Mozart.' (*Play a rap song by Mozart.*)

3. **Identify a set/list by constraints:**
   - Identify an album set
     i.e, U: 'Zeige mir alle Pop Alben von den Beatles.' (*Show all pop albums of The Beatles.*)
   - Identify a song set
     i.e., U: 'Zeige mir alle Lieder die du hast.' (*Show me all songs that you have.*)
   - Identify song list of an album
     i.e., U: 'Welche Songs sind auf dem Album One von den Beatles ?' (*Which songs are on the album One by The Beatles?*)
   - identify song list of an playlist
     i.e., U: 'Welche Songs sind auf der Playliste Pop-Mix ?' (*Which songs are on the playlist Pop-Mix?*)
4. **Identify a feature of a song/album/playlist:** queries supported for each type:
   - song: artist, album, genre, name
   - album: songlist, artist, genre, name
   - album: songlist, name

   i.e., U: 'Von welchem Künstler ist das Lied Yesterday?' (*By which artist is the song Yesterday?*)
5. **List presentation/table browsing:** Small lists are usually presented by speech or graphics, larger lists can be presented by speech and the graphical user interface (see section 3.3).

   Browsing of displayed tables is possible via speech but also by the ErgoCommander controller.
6. **Reference resolution:** A user can refer to objects in several ways:
   (a) Direct reference to an object (i.e., to an artist or genre).
   (b) Through combining constraints on objects (i.e, specifying title and artist of a song).
   (c) Reference to a member of a salient list (e.g., 'the first', 'the second')
   (d) Reference to a currently playing object (e.g., 'What is currently playing?').
   (e) Multimodal reference by clicking on something on the display plus spoken deictic reference.

2.2 Example Exchanges

The following exchanges in German with English glosses are only intended to give an idea of the type of dialog. Note that they are far from exhaustive and the exact phrasings might not always correspond to the current version of the baseline system.

User:	Zeige alle Alben.
	Show all albums.
System:	Ich zeige die ersten sieben Alben.
	I'm showing the first seven albums.
User:	Bitte zeige mir Pop Alben.
	Plase, show me pop albums.
System:	Es gibt sechszehn Alben. Ich zeige die ersten sieben Alben.
	There are sixteen albums. I am showing the first seven albums.
User:	Weiterblättern.
	Next page.
System:	Ich zeige die nächsten sieben Alben.
	I am showing the next seven albums.
User:	Und dann brauche ich vielleicht Beatles-Alben.
	And then I might need Beatles albums.
System:	Leider habe ich Sie nicht verstanden.
	Unfortunately, I did not understand you.
User:	Von wem ist das Lied Rock Me Amadeus?
	By whom is the song Rock Me Amadeus?
multimodal:	Von wem ist dieses Lied (click on Rock Me Amadeus)?
	By whom is this song (click on Rock Me Amadeus)?
System:	Der Song Rock Me Amadeus ist von Falco.
	The song Rock Me Amadeus is by Falco.
User:	Erstelle eine neue Playliste.
	Create a new playlist.
System:	Playlist 7 wurde erstellt.
	Playlist 7 has been created.
User:	Füge Yesterday zur Playliste hinzu.
	Add Yesterday to the playlist.

Chapter 3

Architecture and Components

The architecture of the baseline system is shown in Figure 3.1. Input modalities are speech and GUI input. Speech input is recognized by the Nuance speech recognizer and analyzed using Nuance's natural language understanding system. GUI input (e.g., in the form of a selection of a table row) comes from the ErgoCommander in-car device (described below) and forwarded to the Table Presenter. The table presenter converts the ErgoCommander event (click) as a domain object (e.g., an album) and sends this to the Interpretation Manager.

The Interpretation Manager performs interpretation in context and multimodal fusion. It computes a set of communicative intentions (based on a theory of agent-based dialogue [BA05]) which describe the user's turn (also see deliverables D2.1: Integration of Ontologies and the ISU Approach and D3.1: Extended Information State Modeling).

This set of communicative intentions is then sent to the dialogue manager, which is a baseline instantiation of the collaborative problem-solving theory of dialogue described in [BA05]. Using update rules, the dialogue manager integrates user communicative intentions into the information state (IS), and then uses the current IS to drive its behavior: including making queries to the FreeDB database as well as controlling the MP3 Player (both of which are encapsulated by the MP3 Shield, whose task it is to wrap these two components with a simple API).

The Dialogue Manager also formulates its own communicative intentions (again using the model in [BA05]) and sends these to the Turn Planner. The Turn Planner then performs multimodal fission, deciding which content to realize graphically (both tabular and written language) and which to speak. Linguistic realizations (i.e., spoken utterances and written sentences) are computed by the Linguistic Planner, currently using template based-generation. The Turn Planner itself packages graphical output. The entire output request is then sent to the Output Manager, which times and coordinates graphical and spoken output, which are realized using the Table Presenter and Mary, respectively.

We now briefly describe each of the components separately.

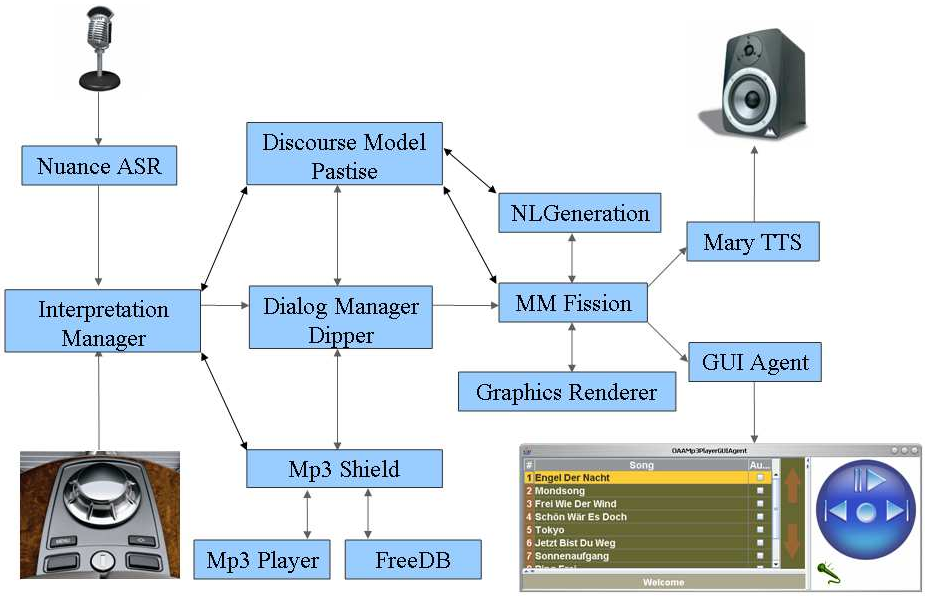

Figure 3.1: The Baseline System Architecture

3.1 Open Agent Architecture

Not shown in the figure is the middleware that ties it all together. The project as a whole has chosen to use the Open Agent Architecture (OAA) [MCM99][1] as the communication middleware between the components. The use of a common middleware allows components from other sites (including those developed by other TALK partners) to be easily integrated, regardless of platform and operating system dependencies.

OAA is flexible in that it supports the major operating systems (Windows, Mac OS, Solaris, and Linux) as well as many popular programming languages (Java, C/C++, Prolog, Lisp, Visual Basic, Delphi, Perl, and WebL).

OAA provides a set of libraries for creating agents, i.e., autonomous software components. These agents then communicate with one another through a central *facilitator* which moderates all communication. Agents can be run either on the same computer, or on different computers (with different operating systems); communication is done through sockets.

Finally, OAA is quickly becoming a kind of standard platform for dialogue system research, and has been used in many different systems ([TBC+02, LGP02, ADH+03] *inter alia*). Because of this, many core dialogue components are already OAA–enabled including some we have chosen to use in our system (Nuance and DIPPER).

[1] Freely available at http://www.openagent.com

3.2 Nuance Speech Recognition

We are using Nuance's speech recognizer version 8.5 for German. This recognizer allows for a very flexible control of both the recognition vocabulary as well as the grammar rules that syntactically specify well-formed phrases or sentences within this vocabulary.

In this section we first describe the Nuance grammar of the baseline system and then the necessary enhancements to Nuance's standard pronunciation dictionary that need to be specified to cover exceptions, in particular the foreign words (artists and titles) in the MP3 song data base.

3.2.1 Grammar

The syntactic specifications of well-formed sentences are defined by a context-free grammar that is complemented with Nuance-specific extensions. The underlying grammar specification formalism itself is a variant of a context-free grammar that additionally provides special syntactic operators that allow to specify valid constituent orderings and alternatives very compactly in the form of conjunctions, disjunctions, optional elements and (positive or Klenee) closures within a single grammar rule.

In addition to that, grammar rules are extended by so called 'slot-filling commands' of the form *<Slot Value>* that may be used to associate each grammar rule with descriptions about elementary semantic features for the respective natural language phrase or sentence that are interpreted as a kind of a pre-interpretation step.

For the MP3 baseline system, we made use of all of the above features. Consequently, the underlying speech recognition grammar is currently a first domain-specific grammar that covers typical phrases and sentences of the application domain and the use cases as described above in section 2.1. The current grammar consists of syntactic rules for about 50 different German phrase types and sentence types.

3.2.2 Pronunciation Dictionary

The recognition vocabulary is phonologically based on Nuance's standard pronunciation algorithm that needs to be extended by means of an appropriate additional pronunciation dictionary containing phonetic descriptions (in the SAMPA format) of all pronunciation exceptions, i.e., special German full forms plus proper names (e.g., artists or song titles) some of them coming from foreign languages. The recognizer's exceptions lexicon currently contains about 500 words plus 200 proper names. In the following we describe the most important phenomena and how they were integrated into our current recognizer lexicon.

The MP3 player domain is a particularly difficult domain for speech recognition. This is due, in part, to the following phenomena:

1. **Proper names**: a music database typically includes many proper names not included in a standard pronunciation dictionary, including artist/band names, song titles, etc.

2. **Creative spellings of regular words**: it is not uncommon for regular words to be spelled differently in song/album titles and artist names. Examples include the band "'N Sync" (in sync) and German song title "Romeo und Juliaaah" (Romeo und Julia).

3. **Foreign language words**: as is the case in many places today, much music listened to by Germans is in the English language. It is imperative, then that the German speech recognizer be able

to recognize foreign language words from the music database. Our music database for the baseline system contains a mix of German- and English-performing artists, and the database includes English, German and (a few) French lexical items.

We dealt with the first phenomena (proper names) by simply adding names to the pronunciation dictionary by hand. The second phenomena (creative spelling) was dealt with by changing creative spellings to their typical version by hand. We do not comment on these uninteresting processes further here.

The third phenomena—that of foreign language words—deserves further comment. Foreign words are typically not found in the standard pronunciation lexicon, and therefore need to be added. There is an addition level of complexity, however, because they need to be added as the speakers (Germans, in our case) will pronounce them.

To get an idea of these pronunciations, we annotated foreign words in a selected section of our SAMMIE-1 Wizard of Oz corpus [KKBG+05] with their phonetic transcription. As a phoneme set, we extended the German SAMPA scheme to include English and French phonemes which occurred in the corpus.

German speakers tend to have a knowledge of English (and French) pronunciation, and many actually used English and French phonemes in their pronunciation of foreign words. Note that this is somewhat different than just modeling accented speech, since these English and French words are embedded in a German utterance. Thus even good speakers of English (like our wizards) did not usually produce pure English pronunciations of English words. Usually at least some German phonemes were substituted (like /s/ for /th/) or some German phonological rules were applied (like devoicing at the end of syllables). Even the same speaker often pronounced the same word in different ways at different times.

For the baseline system, for each foreign word, we created (by hand) several possible pronunciations (encoded in our extended German SAMPA), based on intuition and what was seen in the annotated corpus.

However, one additional step was needed in order to use these within the Nuance system. Nuance (as well as all speech recognizers we know) only supports a limited set of phonemes in the pronunciation dictionary, based on native speech. Thus, there are a number of foreign phonemes (such as /th/) which are not represented within Nuance. To do this mapping, we created a program which would automatically generate a Nuance pronunciation dictionary given a set of extended SAMPA lexical entries and a set of phoneme mapping rules, where rules mapped one extended SAMPA phoneme to a list of Nuance phonemes. These rules were then written by hand.

To automate this process, we hope to use an English pronunciation dictionary in the future to automatically extract the English pronunciations of unknown words, and then use rules to generate a set of possible pronunciation by German speakers.

3.3 The multi-functional System Output GUI

The screen output of the system is realized by a graphical user interface (GUI) having realistic in-car size, as specified by BMW. This GUI realizes both multiple input and output functionalities within one frame all at once. The following figure 3.2 shows a screen shot of this output GUI.

The single functionalities of the application are as follows:

Output:

- table presentations of song lists or album lists

Figure 3.2: Screen shot of the MP3PlayerGUIAgent

- microphone feedback (open vs. closed)
- MP3 Player control GUI
- Song Panel

Input:

- title or album selection
- MP3 Player Control

3.3.1 ErgoCommander Interface

Graphics-based input by the user is realized solely by the control element used in the BMW i-Drive concept, the so-called ErgoCommander. It is a multi-function controller interface that allows for triggering seven different events. The controller may be turned in two directions (left or right), it may be shifted up, down, left or right, and it may be pushed. See figure 3.1 (lower left corner) for an image of the ErgoCommander contoller.

3.4 Interpretation Manager

In this section, we describe the interpretation manager in the baseline system. We first describe the technical basis that interpretation management relies on, namely the production rule system PATE. We then describe in general how interpretation is realized within the baseline system.

3.4.1 PATE

As well as discourse modeling, turn planning and linguistic planning, interpretation management is based on the production rule system PATE. Originally developed for the integration of multimodal input [Pfl04], during the implementation of the baseline system, we found that this engine is also adequate for modeling other dialog system components as it provides an efficient and elegant way of realizing complex processing rules.

To this end, PATE employs Typed Feature Structures (TFS) as the basic internal data representation and XML for encoding all incoming and outgoing data as well as knowledge bases (production rules, type definitions)[Kem04]. Furthermore PATE provides tools to map application-specific ontological knowledge, as frequently used in multimodal dialogue systems (like SmartKom [Wah03] and COMIC [dOB03]), to its internal TFS-based data representation. In processing TFS, PATE provides two operations that both integrate data and also are suitable for condition-matching in production rule systems, namely a slightly extended version of the general *unification*, but also the discourse-oriented operation *overlay* developed at DFKI [AB01].

3.4.2 Interpretation

Interpretation within the baseline system consists of two different layers. The first layer is represented by a context-independent semantic pre-interpretation of a user utterance by the Nuance Grammar as described in 3.2. The second layer, implemented through the PATE-based interpretation module, is responsible for modality fusion w.r.t. to the possible input modalities (speech and ErgoCommander) as well as context-sensitive interpretation. Here, the latter part relies on collaborative interaction between interpretation manager and the discourse model module *Pastis* which will be described in the next section.

3.5 Discourse Model

In this section we describe the discourse processing entities within the baseline system. First we describe the information state as it is defined within the dialogue management module (see 3.6). The we give an introduction to the discourse processing module *Pastis* which includes discourse specific information but also functionality for storing and querying certain information.

3.5.1 Information state of the Dialogue Management

The information state used for representing dialogue management specific information is shown in Figure 3.3. It is divided into four principal segments:

Input: External modules can communicate with the dialogue manager by adding to the input queues. In the *interpretation* queue, the Interpretation Manager writes the interpretation of the user's last utterance. In the case of successful generation, this will be a set of instantiated Grounding Acts (see [BA05]). In the case of unsuccessful generation, this will be a notification of the type of failure (e.g., ambiguous-definite-reference) as well as pertinent details which can be extracted from the utterance.

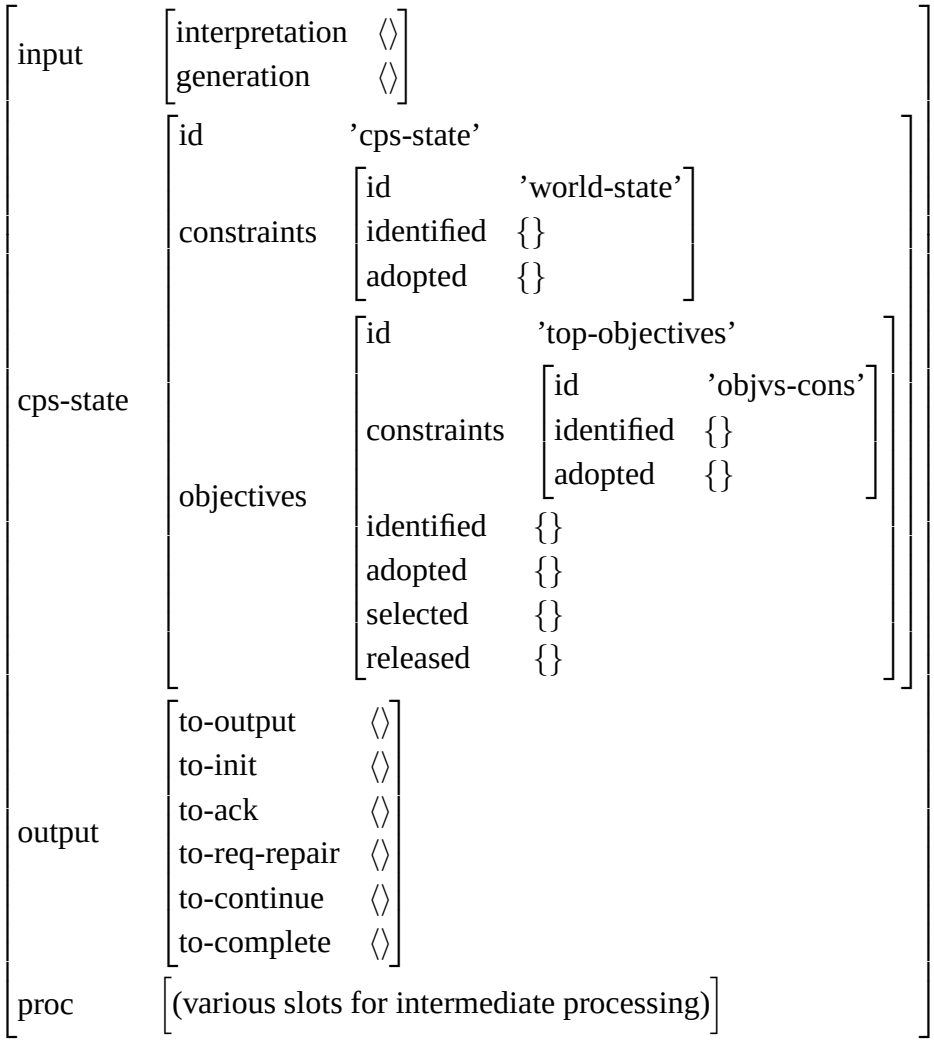

Figure 3.3: Information State within Dialogue Management Module

Likewise, when the generation subsystem has successfully generated the set of Grounding Acts requested by the dialogue manager, it writes a message to the *generation* queue, which allows the dialogue manager to update the information state appropriately.

CPS-state: This tracks the state of problem solving using a portion of the CPS state described in [BA05] and deliverable 2.1. Note that only a few slots from the original model are not represented, in particular the focus stack. We plan to include focus handling in future versions of the system.

Output: There are various queues to which acts that the system wants to output to the user are written. The *to-output* queue contains the final list of grounding acts which the dialogue manager wishes to generate, and they are sent from there to the Turn Planner.

The other queues are for convenience in processing, allowing a bare CPS act or Interaction Act to be written to one, and then those will automatically be translated into the appropriate grounding act

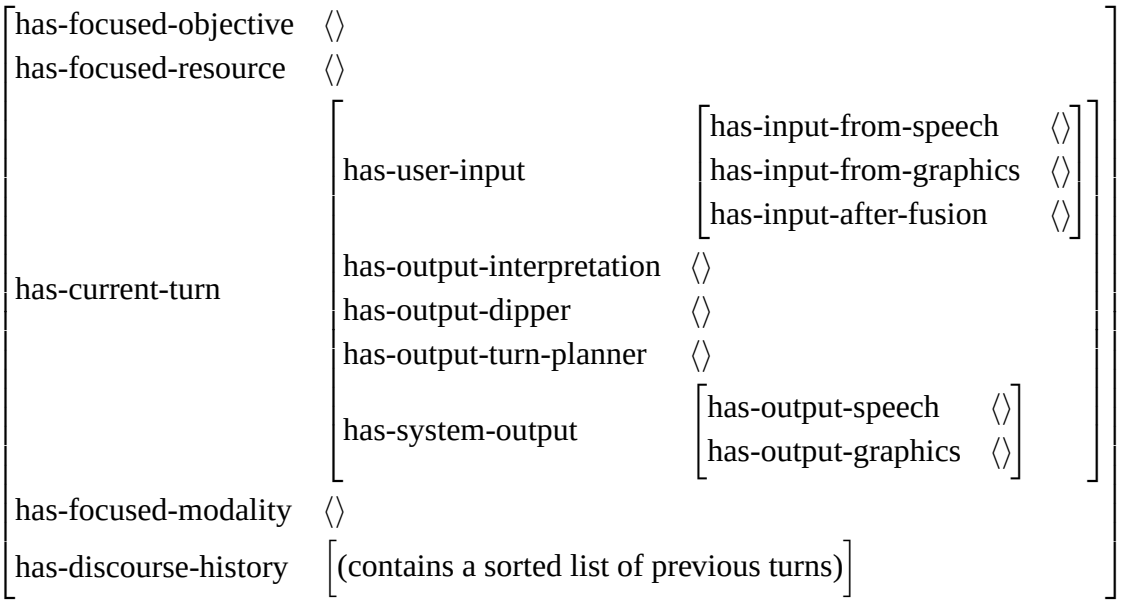

Figure 3.4: Context Representation with Pastis

and put on the *to-output* queue.

Proc: There are also a number of temporary slots which are used only during the processing of an utterance. For space reasons we do not discuss them here.

3.5.2 Discourse Representation with Pastis

As the information state represented in 3.5.1 is modeled for dialogue management specific needs, the PATE-based module Pastis is designed to provide and store modality- and discourse-specific information. In combination both together define the *Extended Information State* of our system. The context is represented as shown in Figure 3.4. External modules can communicate with Pastis in order to store their output structures or request information concerning the current or previous turn(s). Such a request can range from simply querying the user-input from the previous turn to the resolution of cross-modal references or context-enrichment of a given input-structure.

3.6 Dialogue Manager

In this section, we describe the dialogue manager in the baseline system. We first describe the software engine we use for dialogue management. We then describe the information state used in the system, and finally, the update rules for dialogue management.

3.6.1 Dialogue Management Engine

In order to implement the dialogue manager, we rewrote a large portion of the DIPPER dialogue manager engine [BKLO03] to support typed feature structures, using DFKI's JTFS Java library [KDP+04]. We also extended the update rule language to support a range of things, including feature structure manipulation, variables and working memory, as well as code blocks and simple if-branching.

We also expanded the typing system to support several data types, including booleans, integers, strings, lists, feature structures, and OAA's ICL structures. The update rule language also supports lambda extractions in a manner similar to LISP.

These extensions were necessary, first and foremost to support operations on typed feature structures, since these are the main feature of our collaborative problem-solving dialogue model. Other extensions support the writing of more robust rules by reducing rule duplication (e.g., if statements and variables).

3.6.2 Update Rules

The dialogue manager engine works on a simple production rule principle. It loops through the set of update rules and applies the first whose application conditions are currently true.

When the dialogue manager is started (or reset), a set of initialization rules are triggered, which set up the proper state. When no more rules are applicable, the rule application engine blocks until some external event (e.g., a user utterance) causes a change in the information state.

As with other information state update-based systems, dialogue management in the CPS model can be grouped into three separate processes:

Integrating Utterance Information Here the system integrates grounding acts—by both user and system—as they are executed. This is a fairly circumscribed process, and is mostly specified by the CPS model itself. This includes such rules as treating an interaction act as executed when an *ack* has been executed, or moving an object to the *adopted* slot of a context when an *adopt* CPS act is successfully executed. This process is described in detail in [BA05] as well as deliverable 3.1 "Extended Information State".

Agent-based Control Once executed grounding acts have been integrated into the dialogue model, the system must decide what to do and what to output next. The baseline system contains a number of simple agent-based rules to control this system behavior. Examples include such rules as: "if an objective has been selected and the system knows how to execute it, then execute it" and "if the user issues a *c-identify* without content (e.g., by asking a question), and the system has information to add to the act, then *continue* the *c-identify* by adding the appropriate information." We plan to refine and expand these behavioral rules for the final showcase.

Package and Output Communicative Intentions During the first two phases, communicative intentions (i.e., instantiated grounding acts) are generated, which the system wants to execute. This last phase packages these and sends them to the generation subsystem for realization. When realization is successful, the information state is updated using the rules from the first phase.

3.7 Back-end Components

This section describes the domain-application part of our system. The core component is the MP3 Shield as an *intelligent* interface to the FreeDB song data base and the jlGUI MP3 Player.

3.7.1 MP3 Shield

This module provides an interface to the MP3 player's functionality, provides on-the-fly conversion of list-like results from the FreeDB song database to ontological objects when making queries to FreeDB. Furthermore, it provides functionality to calculate the best possible distinct feature that can be asked when a user-request results in a large result-set that needs to be narrowed down. The MP3 Shield combines and simplifies all system functionality into a single, simple API that is used by the dialog manager.

3.7.2 MP3 Player

As mentioned above, we will have to simulate most music playback done by the system. There are several reasons for this. First of all, and most practically, we do not have (legal) access to such a large database of albums and songs. We may, however, depending on legal requirements, be able to have some small set of songs that we support, but this would likely only be available in non-public settings (e.g., experiments, evaluations, or private demonstrations), as public presentations (such as public demonstrations) are required in most countries to pay royalties for songs played.

Nonetheless, we still want to support limited song playback in the baseline system. For this, we use the jlGui player[2] which has already been wrapped for OAA in Work Package 1.

3.7.3 FreeDB Music Database

We are using the FreeDB music database[3] which is a large database of album information (but not actual music). As the database was very noisy (it can be updated by anyone with the right software), we cleaned it up considerably by using an automatic clustering routine to throw out similar entries as well as other heuristics to detect incorrect entries.

We have also (in conjunction with our sub-contractor CLT Sprachtechnologie) created a custom database program which allows SQL searches of the database both through OAA as well as a GUI tool (which has been used in our Wizard of Oz experiments). More details on the database and tool can be found in Deliverable D5.1.

3.8 Multimodal Fission

Multimodal fission in our baseline system is split into two modules—the PATE-based *Turn Planner* and the *Output Manager*.

[2]Freely available at http://www.javazoom.net/jlgui/jlgui.html

[3]Freely available at http://www.freedb.org

3.8.1 Turn Planner

The turn planner takes a set of CPS-specific conversational acts generated by the dialogue manager and maps them to modality-specific communicative acts. Therefore information on how information should be distributed over the available modalities (speech or graphics) can be obtained by

- the discourse module Pastis (cf. Section 3.5.2) which, based on the current discourse context, provides a mechanism for making a hypothesis on which modality the user is currently focused.
- a set of production rules that determine which kind of information should be presented through which of the possible output modalities.

Beside the derivation of the best possible distribution of information, the turn planner is also responsible to transform domain-specific information that will be presented by speech to a representation we call *Reduced Knowledge Representation* (RKR) that can be interpreted by the Linguistic Planner. Such a RKR structure differs from its source structure through its more general, abstract surface structure, which can be seen as an intermediate form between a pure ontological representation of a domain object and the logical form representation that the linguistic planner needs for deep generation with OpenCCG. Furthermore, an RKR structure specifies exactly that information that has to be presented to the user. For an example consider structure 3.5 which defines an ontology-based *play(song)*-objective and the resultant RKR structure 3.6 which would lead to an system utterance like *'Ich spiele Yesterday von the Beatles' (I will play Yesterday by The Beatles.)*.

As already mentioned at the beginning of chapter 3, the turn planner is also responsible for packaging the information that will be presented by the display. This means that the information that is going to be presented will be transformed into a slightly different format that the table presenter can interpret.

3.8.2 Output Manager

When the Turn Planner has finished processing, it sends a sorted bundle of output messages, including both speech and graphic messages, to the Output Manager. The output manager then has the task to send the appropriate messages further to the graphics renderer or the generation manager and ensure with some internal time/id management operations that the rendered output will be presented in the right order when sending it to output presentation modules (Mary TTS and Table Presenter).

3.9 Linguistic Planner

The Generation Manager takes the input from the Turn Planner—a Reduced Knowledge Representation structure (RKR structure)—and passes it to the Linguistic Planner. Its role is to turn the input into a natural language utterance.

The Linguistic Planner is also based on the production rule system PATE (cf. Section 3.4.1). Because of the parallel development of the sentence realizing modules, we first extend the template-based generation module and then the deep generation (i.e. the OpenCCG grammar). This results in two different planning rule sets: for shallow and for deep generation, respectively.

For each kind of speech act, there is one or more sentence planning rule. Based on the speech act type, the template-based rules extract the relevant information from the input structure—like objective, type of

entities (actor, patient, etc.), name of entities (songs, albums, etc)—and transform it into a flat structure, close to surface form. Using stylesheet transformations, these structures are eventually turned into natural language. For the time being, we use simple planning rules, turning each speech act into a sentence. For the sake of variation, these structure also have a randomly generated number which triggers the final surface form. For instance, a speech act of type **ACKNOWLEDGE** may be expressed by *OK!*, *Alles klar!* or *Jawohl!*. Depending on the random number, the stylesheet transformation will output one of these variants.

The resulting output is then sent to the Generation Manager, which, depending on whether the output has to be realized in spoken or written form—an information which is obtained from the Output Manager—will send the utterance either as plain text or wrapped in the Mary-XML format.

3.10 Mary Speech Synthesis

For synthesizing spoken output, we are using the Text-to-Speech system M.a.r.y. [4] [ST01]. A nice property of this system is the ability to add language markup to the input. The system supports output in both English and German, which allows us to embed English titles and artist names within a German sentence. The baseline system will use such markup for speech output, but we are exploring the possibility for later versions to convert the English titles to a German phone set and output them using that (in conjunction with our work on the ASR pronunciation dictionary described above).

[4] http://mary.dfki.de

```
<content>
  <Play-song>
    <id>id329746750</id>
    <has-song>
      <Song>
        <id>id287041266</id>
        <has-name>
          <Name>
            <id>jtfs-1312</id>
            <has-string-value>Yesterday</has-string-value>
          </Name>
        </has-name>
        <has-artist>
          <Artist>
            <id>jtfs-1313</id>
            <has-name>
              <Name>
                <has-string-value>the Beatles</has-string-value>
              </Name>
            </has-name>
          </Artist>
        </has-artist>
        <has-genre>
          <has-string-value>Pop</has-string-value>
        </has-genre>
        <has-id-from-freedb>
          <Freedb-id>
            <has-value-int>260</has-value-int>
          </Freedb-id>
        </has-id-from-freedb>
        <belongs-to-album>
          <Album>
            <id>jtfs-1319</id>
            <has-artist>
              <Artist><id>jtfs-1313</id></Artist>
            </has-artist>
            <has-name>
              <Name>
                <id>jtfs-1336</id>
                <has-string-value>One</has-string-value>
              </Name>
            </has-name>
            <has-year-of-appearance>
              <has-int-value>1973</has-int-value>
            </has-year-of-appearance>
            <has-song-list/>
          </Album>
        </belongs-to-album>
      </Song>
    </has-song>
  </Play-song>
</content>
```

Figure 3.5: Example instance of objective play(song)

```
<content>
  <Play-song>
    <id>id329746750</id>
    <has-relation>
      <Relation>
        <argument><System/></argument>
        <relation-name>Actor</relation-name>
      </Relation>
    </has-relation>
    <has-relation>
      <Relation>
        <argument>
          <Song>
            <id>id287041266</id>
            <has-relation>
              <Relation>
                <argument>
                  <Name>
                    <id>jtfs-1312</id>
                    <has-string-value>Yesterday</has-string-value>
                  </Name>
                </argument>
                <relation-name>Specifier</relation-name>
              </Relation>
            </has-relation>
            <has-relation>
              <Relation>
                <argument>
                  <Artist>
                    <id>jtfs-1331</id>
                    <has-relation>
                      <Relation>
                        <argument>
                          <Name>
                            <has-string-value>The Beatles</has-string-value>
                          </Name>
                        </argument>
                        <relation-name>Specifier</relation-name>
                      </Relation>
                    </has-relation>
                  </Artist>
                </argument>
                <relation-name>Specifier</relation-name>
              </Relation>
            </has-relation>
          </Song>
        </argument>
        <relation-name>Patient</relation-name>
      </Relation>
    </has-relation>
  </Play-song>
</content>
```

Figure 3.6: RKR representation of objective play(song)

Chapter 4

Summary and Future Directions

This deliverable describes the DFKI/USAAR baseline system for multi-modal, in-car dialogues with an MP3 application. We have integrated and mostly developed ourselves all the necessary components for such a system: Access to the modalites (i.e., speech recognition and synthesis and a graphical user interface with the ErgoCommander controller), multimodal understanding and generation components (i.e., fusion, fission, language and graphics realization, and a discourse module), a dialog managment component and a set of modules for the acutal MP3 application (i.e., the FreeDB database, the jlGUI MP3 player and the encapsulating MP3Shield module).

We have provided a first implementation of collaborative problem-solving based dialog managment, for which we had to extend the Dipper module with a typed feature structure component. Multimodal fusion allows multimodal references (*Play this song.* plus a button click) and crossmodal referencces (*Play song number 5 on the list.*), for which we had to add the discourse module Pastis. Multilingual utterances are a particular challenge, for speech recognition as well as speech synthesis. For speech recognition in this baseline system, we were able to address this challenge with a lexicon of exception, but for larger databases, this remains an open problem. For speech synthesis, we made use of the multilingual capabilities of the Mary speech synthesis component.

The following evaluation of the baseline system at partner BOSCH will begin with working out the detailed test scenario, with system setup, task descriptions, questionnaires etc. This will be prepared in collaboration with DFKI and USAAR. The results of the evaluation will serve a number of puroposes, including the influence of driving as the primary task on all aspects of the dialogues, identify problems for dialog managment (e.g., suitable collaborative moves and clarification tasks), help taylor output planning choices (e.g., selecting information, media allocation and redundancy), and serve as a comparison for the final system version.

Further development of the baseline system will depend on the results of the evaluation to be performed in the second half of 2005 (project months 19-24). Besides improvement of the various modules, we are planning to extend the system in various general ways:

- Adding other domains, e.g., interaction with a calendar application
- Demonstrating the language-independence of most modules by adding an English version of the baseline system.
- Integrating research results, in particular from work packages 2 and 3.

Module specific improvements that are already identified include

- As for NLG, the template-based generation will be gradually replaced by deep generation. This goes along with the development of the German OpenCCG grammar.
- Another important issue will be the analysis of the SAMMIE corpora with respect to sentence planning. This will result in refining our linguistic planning rules.
- Further analysis of the SAMMIE corpus and also the evaluation data will also drive better, e.g. context-dependent, multimodal fusion.
- The same goes for coverage of the ASR component, especially grammar coverage.
- Extensions of dialog management include further clarification dialogs.

Bibliography

[AB01] Jan Alexandersson and Tilman Becker. Overlay as the basic operation for discourse processing in a multimodal dialogue system. In *Proceedings of the 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Seattle, Washington, August 2001.

[ADH+03] G. Aist, J. Dowding, B. A. Hockey, M. Rayner, J. Hieronymus, D. Bohus, B. Boven, N. Blaylock, E. Campana, S. Early, G. Gorrell, and S. Phan. Talking through procedures: An intelligent Space Station procedure assistant. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*, Budapest, Hungary, April 12–17 2003. Demo session.

[BA05] Nate Blaylock and James Allen. A collaborative problem-solving model of dialogue. In *Proceedings of the SIGdial Workshop on Discourse and Dialog*, Lisbon, September 2–3 2005. To appear.

[BKLO03] Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka. DIPPER: Description and formalisation of an information-state update dialogue system architecture. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialog*, Sapporo, Japan, July 4–5 2003.

[dOB03] Els den Os and Lou Boves. Towards ambient intelligence: Multimodal computers that understand our intentions. In *eChallenges e-2003*, 2003.

[KDP+04] Hans-Urlich Krieger, Witold Drożdżyński, Jakub Piskorski, U. Schäfer, and Feiyu Xu. A bag of useful techniques for unification-based finite-state transducers. In *Proceedings of Konvens 2004*, Vienna, Austria, September 2004.

[Kem04] Benjamin Kembe. Pate - a production rule system based on activation and typed feature structure elements. Master's thesis, Saarland University, 2004.

[KKBG+05] Ivana Kruijff-Korbayov, Nate Blaylock, Ciprian Gerstenberger, Verena Rieser, Tilman Becker, Michael Kaißer, Peter Poller, and Jan Schehl. An experiment setup for collecting data for adaptive output planning in a multimodal dialogue system. In *Proceedings of the 10th European Workshop on Natural Language Generation*, Aberdeen, Scotland, August 8–10 2005. To appear.

[LGP02] Oliver Lemon, Alexander Gruenstein, and Stanley Peters. Collaborative activities and multitasking in dialogue systems: Towards natural language dialogue with robots. *Traitement Automatique des Langues (TAL)*, 43(2):131–154, 2002.

[Mat03] Stefan Mattes. The lane-change-task as a tool for driver distraction evaluation. In H. Strasser, K. Kluth, H. Rausch, and H. Bubb, editors, *Quality of Work and Products in Enterprises of the Future / Arbeit und Produkt in Unternehmen der Zukunft*, pages 57–60. Ergonomia, Stuttgart, Germany, 2003.

[MCM99] David L. Martin, Adam J. Cheyer, and Douglas B. Moran. The open agent architecture: A framework for building distributed software systems. *Applied Artificial Intelligence*, 13(1–2):91–128, January–March 1999.

[Pfl04] Norbert Pfleger. Context based multimodal fusion. In *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*, pages 265–272, New York, NY, USA, 2004. ACM Press.

[ST01] Marc Schröder and Jürgen Trouvain. The German text-to-speech synthesis system MARY: a tool for research, development and teaching. In *4th ISCA Workshop on Speech Synthesis*, Blair Atholl, Scotland, 2001.

[TBC+02] Christian Theobalt, Johan Bos, Tim Chapman, Arturo Espinosa-Romero, Mark Fraser, Gillian Hayes, Ewan Klein, Tetsushi Oka, and Richard Reeve. Talking to Godot: Dialogue with a mobile robot. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002)*, pages 1338–1343, 2002.

[Wah03] Wolfgang Wahlster. Smartkom: Symmetric multimodality in an adaptive and reusable dialogue shell. In *Krahl, R., Gnther, D. (eds): Proceedings of the Human Computer Interaction Status Conference 2003, June 2003, Berlin: DLR*, pages 47–62, 2003.

Appendix A

Baseline System Software on CD-ROM

A CD-ROM with the software of the baseline system should be found here.