Documented Prototype of Dialogue Manager

Project acronym:	ALIAS
Project name:	Adaptable Ambient Living Assistant
Strategic Objective:	ICT based solutions for Advancement of Social Interaction of Elderly People
Project number:	AAL-2009-2-049
Project Duration:	July, 1st 2010 – Juni, 30th 2013 (36months)
Co-ordinator:	Prof. Dr. Frank Wallhoff
Partners:	Technische Universität München Technische Universität Ilmenau Metralabs GmbH Cognesys GmbH EURECOM Guger Technologies Fraunhofer Gesellschaft pme Familenservice YOUSE GbR

D3.4

Version:	1.01
Date:	28. April 2011
Autor:	Andreas Franke

Dissemination status: PU

This project is co-funded by the Ambient Assisted Living (AAL) Joint programme, by the German BMBF, the French ANR, the Austrian BMVIT.

Once completed please e-mail to WP leader with a copy to
eric.bourguignon@tum.de and frank@wallhoff.de**.**

Del 3.4	Executive Summary
Following the concept described in D3.1, a prototype of the dialogue manager has been realized, based on the CES core system. It features an interface to the RobotDaemon and an input channel for natural language. Application binaries have been installed on the robots, and their functionality has been verified with a basic navigation scenario.	

Dissemination Level of this deliverable (*Source: Alias Technical Annex p20 & 22***)**	
PU	Public
Nature of this deliverable (*Source: Alias Technical Annex p20 & 22***)**	
R	Report

Due date of deliverable	30.04.2011
Actual submission date	28.04.2011
Evidence of delivery	

Authorisation			
No.	**Action**	**Company/Name**	**Date**
1	Prepared	cognesys GmbH, Andreas Franke	28.04.2011
2	Approved	IUT, Jens Kessler	04.05.2011
3	Approved	TUM-MMK, Jürgen Geiger	20.05.2011

Table of contents

1. Introduction

The ALIAS robot will have to perform a variety of functions and behaviours in order to be useful as a companion for elderly people. These behaviours need to be selected and activated depending on the communication with the user on the one hand, and the actual state of the robot on the other hand. In the ALIAS project, the high-level selection of an appropriate robot behaviour in the light of the available inputs is the task of the dialogue manager. In this sense, it controls the robot.

This makes the dialogue manager a central component of the robot system: It is a component that has many connections to other modules from which it receives inputs. And when it generates decisions about the robot's behaviour, it sends the corresponding outputs or commands to the appropriate modules that can realize the desired actions.

The general concept for the dialogue manager has been described in deliverable D3.1. Figure 1 shows the overall architecture again:

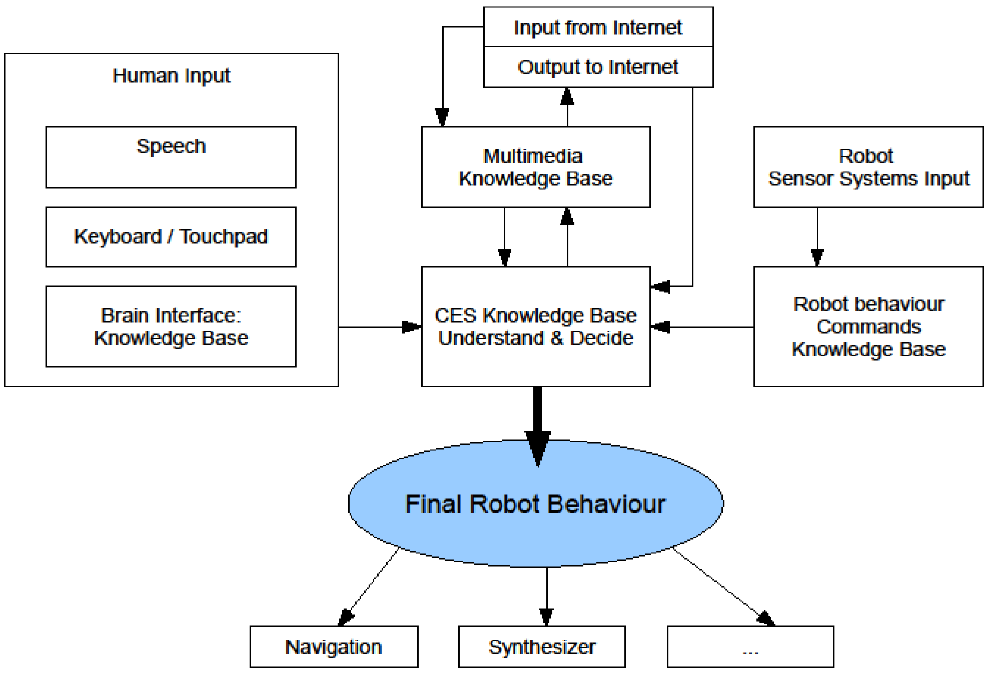
Input from Internet
Output to Internet
Human Input
Speech
Keyboard / Touchpad
Brain Interface: Knowledge Base
Multimedia Knowledge Base
Robot Sensor Systems Input
CES Knowledge Base Understand & Decide
Robot behaviour Commands Knowledge Base
Final Robot Behaviour
Navigation
Synthesizer
...

Figure 1: Overall Architecture of the Dialogue Manager

As we concluded there, the dialogue manager has to integrate the information from diverse sources, namely human input, robot input and knowledge about available services. This integration is based on the Cognesys core system as a central “translation” unit.

2. Requirements

In this section, we summarize the functional expectations with regard to the dialogue manager, and the constraints derived from the technical environment.

2.1 Input to the Dialogue Manager

Summarizing deliverable D3.1, section 3, the dialogue manager needs to process input from the following modules:

- Net-based services and other applications: The robot offers a multitude of services to the user, e.g. reminder, email, web browsing, gaming, etc. The dialogue manager must be able to activate these different services, either through free speech or using textual input on the touch screen. In most cases, the dialogue manager must also be able to terminate the services. Depending on the individual level of integration with each service, the dialogue manager must also receive globally relevant inputs and events from the net-based services.
- Robot sensors & spatial orientation system (high-level modules e.g navigation)
- Speech interface
- Touch screen
- Brain computer interface
- Person detection and identification by voice and face

Based on input from these modules, it must select and activate suitable behaviours in response, and generate appropriate speech acts.

While performing this task, the set of active behaviours must always be consistent. In particular, it must be ensured that conflicting applications and processes are not active at the same time. For example, when the battery is empty, any requests to play games need to be delayed until the robot has been recharged.

In order to ensure this consistency in performing robot control and user interaction, the dialogue manager has to keep track of the current global state of the robot.

2.2 Technical Requirements

The available computing resources and hardware environment of the robot are shown in figure 2.

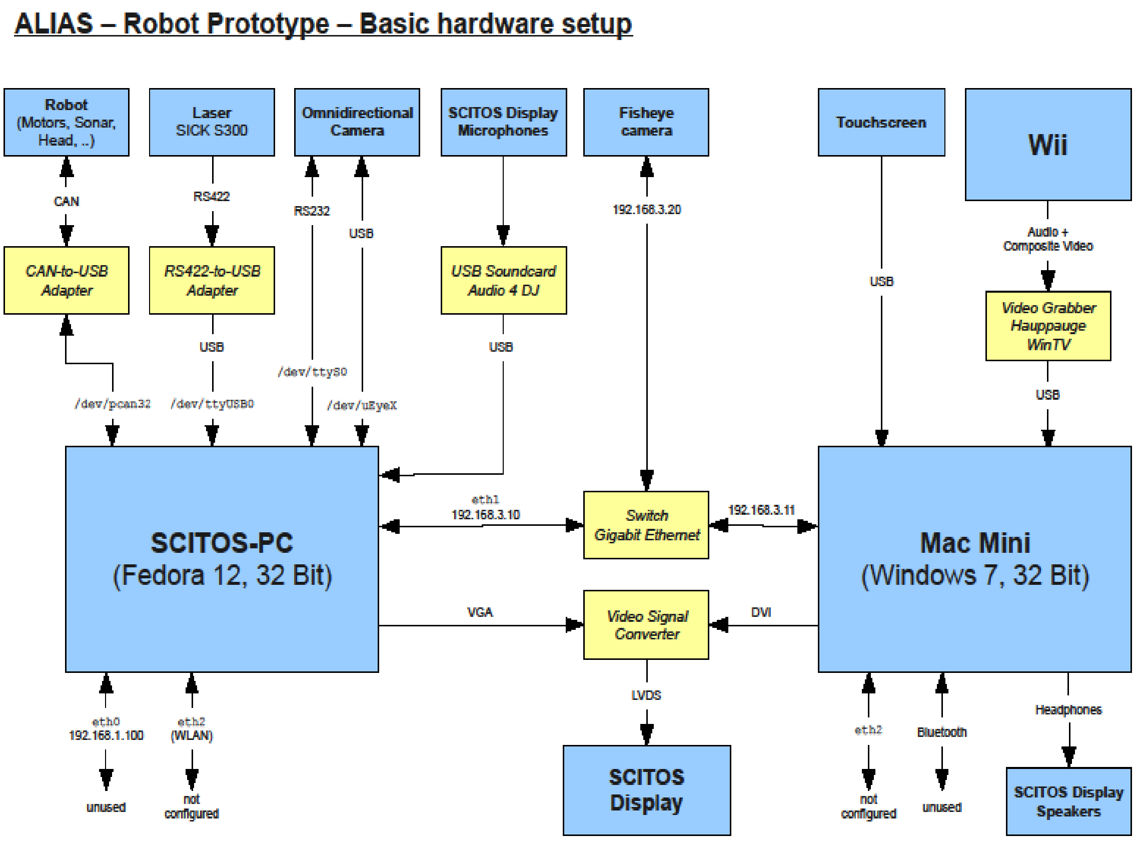

Figure 2: Basic Hardware Setup of the ALIAS Robot Prototype

While the navigation subsystem and other more hardware-oriented modules are running on the SCITOS-PC, many of the user services will be realized as web applications running in the browser, or as application binaries. In both cases, they will be running on the Mac Mini, running MS Windows. Since the dialogue manager must be able to launch and terminate these application services, it has been decided for it to run on the Mac Mini, too. Thus the dialogue manager is a Windows application, too. As a side effect, this minimizes development effort through reuse of existing infrastructure. Effective communication with relevant modules on the SCITOS-PC is ensured by the fast ethernet connection.

Since the dialogue manager shares its host machine with several other application processes, it needs to be designed to make efficient use of the computational resources available there.

3. Design & Structure

From a functional perspective, the dialogue manager can be seen as implementing a mapping from events (ranging from simple ones like “Battery empty” to more complex ones like input from the speech recognizer), to result signals (notably control signals for the robot on the one hand, and visual and/or natural language output to the user).

In order to produce a consistent behavior, the dialogue manager must respond to the incoming events and inputs from applications and the rest of the system in such a way that no conflicting signals are generated at the same time. To ensure this, event processing is effectively serialized, so that any events are processed one at a time. Currently this is realized as an event processing loop, but more sophisticated scheduling mechanisms for the case of multiple events occurring almost at the same time may be considered in future versions.

Upon receiving updates from these subsystems, the dialogue manager decides on a suitable response behavior and activates it by sending a command to the respective module.

Our design can be split in two parts: an inner part operating on a uniform conceptual representation that maps the input events to some suitable output actions, and an outer part that deals with the communication protocols and translates from and to the external data formats.

Using this architecture with its conceptual foundation and a set of independent, possibly diverse I/O adapters, enables the dialogue manager to integrate a wide range of different modules in the heterogeneous robot environment without having to commit to a common, uniform, system-wide middleware infrastructure that every component has to use.

We now describe the two parts in more detail.

3.1 The Inner Part of the Dialogue Manager

Inputs from other parts of the system can only reach the core part of the dialogue manager after they have been received and properly preprocessed by the corresponding input adapters, which are not part of the dialogue manager core. Therefore, we can enforce the principle that the core part of the dialogue manager operates on a strictly conceptual level. It processes the given information in two phases, as shown in figure 3. The first one of these is the understanding phase. Its purpose is to establish the semantics of the input. After that, there is a second phase for decision making, resulting in appropriate actions. In the following, we describe these two processing phases in more detail.

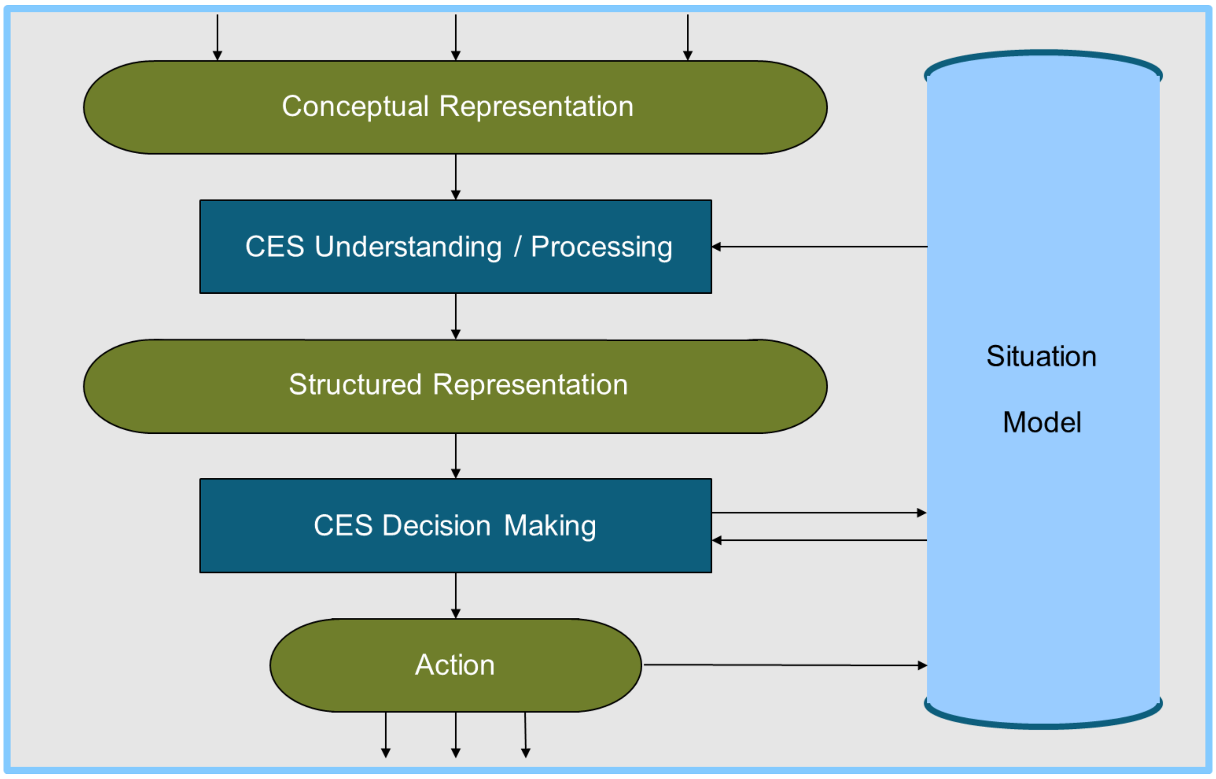
Conceptual Representation
CES Understanding / Processing
Structured Representation
CES Decision Making
Action
Situation Model

Figure 3: Schematic View of the DM Core

An input that reaches the core part of the dialogue manager can be represented as a sequence of concepts. For example, this can be a natural language utterance like “come here please”, or an event received from the robot hardware, like a “BatteryEmptyEvent”. While the meaning of the latter is straightforward, it is especially in the case of natural language input important that the semantics of the phrases received has to be reconstructed. This is achieved by establishing the appropriate links among the concepts of the phrase processed, based on the relevant parts of the world knowledge and the expert knowledge. In this way, the input is converted into a structured representation. In the case of “come here please”, the concept “here” is determined to be the destination location for “come”. The result is a structure that represents the fact that the ***robot*** is ***requested*** to ***move to*** the ***current location of the user***: request (move(Robot, Location(User))).

During the understanding phase, the relevant information is retrieved from the situation model. In our example, this may be the current location of the robot and the last known location of the user, and the fact that there is no specific application active on the robot that the utterance could be related to. This situation information also contributes essentially to e.g. efforts to disambiguate between different word senses for a given input word, or to choose from a set of alternative words (cf. Section 4.2.2).

In the second phase, the structured information is processed further in order to derive appropriate actions or commands for the robot. In preparation for the decision, the situation model may need to be consulted again. We use the situation model to keep track of the relevant ongoing activities and active processes. Thus, before a decision to engage in a new activity, e.g. to drive towards the user, is taken, any conflicting ongoing activities can be determined, e.g. in case the robot is already limited in his scope of action due to an empty battery. The check for potential side effects and conflicts relies heavily on the expert knowledge. Among other things, it provides assertions like „gaming on the Wii and other user application services are mutually exclusive“, so they cannot be active together.

In the end, the decision making phase determines a particular set of actions. When that point is reached, the situation model is updated accordingly, e.g. by adding an entry for a newly started activity.

3.2 The Outer Part of the Dialogue Manager

For communication with the rest of the ALIAS system, the dialogue manager core is embedded in a communicator shell which provides input & output adapters for interaction with the other modules. In the process, it serializes the incoming data for processing by the DM Core as described in the previous section. This design is presented in figure 4.

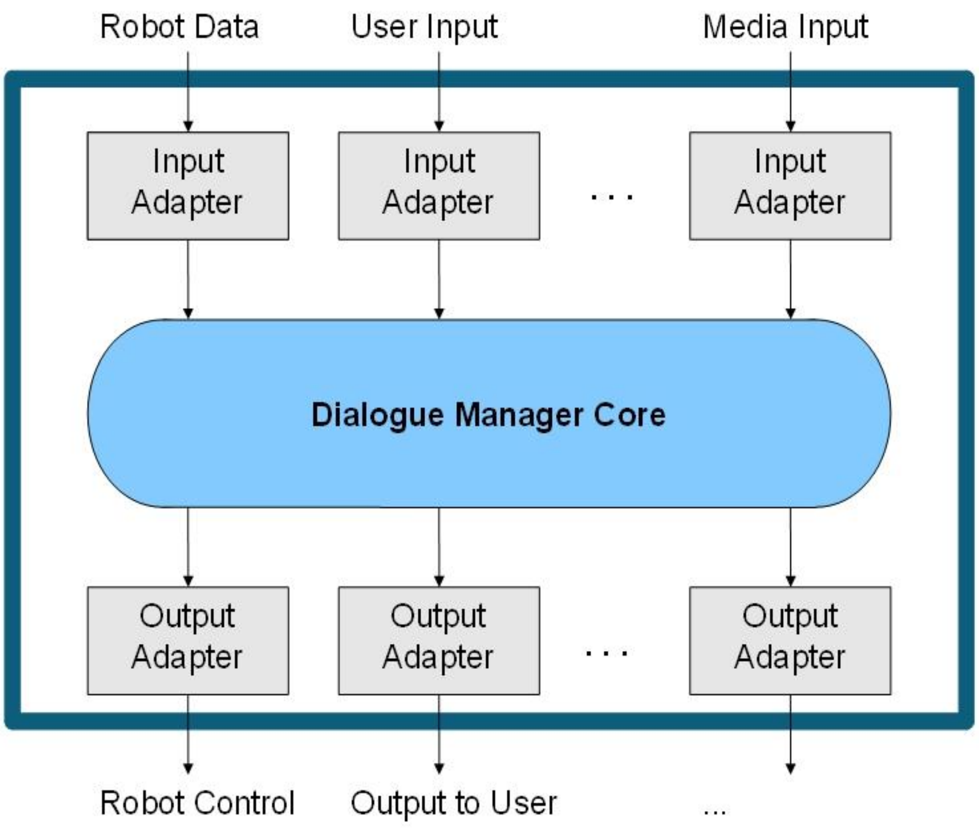

Figure 4: Schematic View of the DM Communicator

There is of course no fixed connection between input and output channels: an input arriving via a certain input adapter can generate outputs for any number of output adapters. Please see some example scenarios:

- The robot fires a „BatteryEmptyEvent“. Such an event will be preprocessed by the leftmost input adapter and then passed on to the DM Core. In response, the Core may issue at least two actions, which are then forwarded to their destinations by different output adapters: (i) a control command (sent via the leftmost output adapter) that tells the robot to drive to the docking station (if there is one), and (ii) an output to the user that explains what is happening, after being postprocessed by the second output adapter.
- The user says „come here please“. This input is received from the speech recognizer via the second input adapter and passed on to the DM Core, which may then decide to comply with this user request and issue an appropriate action. This action is then passed on to the leftmost output adapter, which translates it into a „GotoUser“ event and finally fires that event in the robot control system.
- If the user says „I would like to make a phone call to my sister“, the processing is similar to the previous case, except that the actions determined by the DM Core may in this case include activating the phone application, which is carried out by a yet another output adapter.

The implementation details of the DM Communicator are described in section 4.2.

4. Implementation

In order to provide for an efficient development cycle in the light of different change rates, the dialogue manager implementation has been split into two parts: (i) the core engine for dealing with the inputs on the conceptual level, thereby producing appropriate outputs, and (ii) an outer shell for integrating the connections with the different components of the robot, driving the serialized event loop.

In the current prototype, both of these parts have been implemented in Common Lisp, using LispWorks as the development environment. The resulting application binaries have been installed to the robots, as is further described in section 5.

4.1 The Dialogue Manager Core

The DM Core implementation is based on the Cognesys core engine, with a custom knowledge base for the ALIAS project (see deliverable D3.2). This core engine indeed makes efficient use of computational resources (as specified in section 2.2).

Internal to the dialogue manager, the DM Core provides a TCP/IP-based interface for use by the DM Communicator.

4.2 The Dialogue Manager Communicator

The DM Communicator provides input and output adapters that enable the various system components to communicate with the DM Core.

These system components are:

- the robot hardware (e.g. battery) and the navigation system
- the speech recognizer, providing natural language input from the user
- the speech synthesis system, providing natural language output to the user
- the symbolic keyboard with its buttons
- the brain computer interface
- physiological monitoring
- sensor processing modules like face identification, voice identification and person detection
- specific applications like email, gaming, etc.

The input and output to the robot hardware and navigation system is described in section 4.2.1, the speech input interface is described in section 4.2.2, and the other interfaces are described in section 4.2.3.

4.2.1 The RobotDaemon Interface

The DM Communicator has got a combined input/output adapter connection to the RobotDaemon process running on the SCITOS PC (i.e. the Linux machine). This connection is based on the SCITOS-Script-Interface protocol from MetraLabs. In this protocol, the RobotDaemon acts as a server, while the DM Communicater acts as a client. The messages are exchanged over a TCP/IP connection; the message format is based on ASCII and XML.

This protocol defines two mechanisms for information exchange, using a shared TCP/IP connection: The first mechanism allows the dialogue manager to send a REQUEST message to the RobotDaemon and receive the corresponding ANSWER message (with matching id). Independently of this, the protocol specifies an event framework as a second mechanism. In this framework, the dialogue manager can register for events it is interested in (via the REGISTER_EVENT message), for which it will later receive notifications (in the form of EVENT messages) whenever it occurs. Using the FIRE_EVENT message, the dialogue manager can trigger events, too.

Support for all of these messages has been implemented in the DM Communicator. The current prototype primarily makes use of the event framework, as described in section 5.1.

4.2.2 The Interface to the Speech Recognition Module

The integration with the speech recognition module developed by Fraunhofer IDMT is currently under way. In the first version, an utterance will be received by the DM Communicator as a sequence of words, possibly augmented with alternatives in case of recognition uncertainty.

Upon receiving an input sequence, the dialogue manager generally performs the task of correcting errors and linking up the concepts, thereby in effect understanding the meaning. As different alternatives come in, they are passed through to the core. In the process of linking up the concepts, the core then chooses a suitable alternative that is consistent with the context. Suppose “come” is followed by the alternatives coming from the speech recognition “hear” / “here”, it will be determined that “here” fits well together with “come”, as opposed to “hear” which does not.

On the technical side, priorities are to ensure seamless future extensibility, while at the same time avoiding complexity as much as possible. To that effect, both the communication protocol (HTTP) and the encoding language for the payload data (JSON or XML) will be based on existing standards, with suitable libraries readily available for all programming languages used in our project.

4.2.3 Other Interfaces

The interfaces to other system modules are currently still under development.

- The brain computer interface will be based on an upcoming standard developed by GTec.
- Communication with the person detection module will be realized via the RobotDaemon interface.
- Incoming events from applications like "email has arrived" will be handled similar to events from the RobotDaemon.
- Speech acts targeted at the user will be sent to the speech synthesis system.
- Starting of an external application will be possible in two variants:
  - Launching of an external Windows application binary.
  - Launching of a browser-based web application.

5. Usage & Testing

The dialogue manager prototype consists of two executables located on the MacMini machine in the directory **`C:\ALIAS\`** :

1. **`DialogueManagerCore.exe`**
2. **`DialogueManagerCommunicator.exe`**

When the robot is started, the DM Core is launched first, followed by the DM Communicator, which then connects to the RobotDaemon. This connection is automatically established again if the RobotDaemon needs to be restarted.

5.1 Testing

In order to be able to simulate selected chains of events without a robot, a test application called **`DummyRobotDaemon.exe`** has been built. It can be started on any machine using the MS Windows Operating System; the same can be done with the dialogue manager binaries. In this case, the IP-Address of that machine can be given to the DM Communicator at startup as a command line parameter, e.g.

```
DialogueManagerCommunicator.exe --rd-host 127.0.0.1
```

Then the RobotDaemon server will be expected on that host instead. As usual, the port number currently used for the RobotDaemon server is **`22222`**, but this can be changed, too, if need be, by providing the **`--rd-port`** option.

With this setup, the round trip from the dialogue manager to the Navigation system and back can be verified as follows:

A natural language input of "bitte komm her" (German for "come here please") is submitted from the DM Communicator Test Panel.

This input is passed to the DM Core for processing, eventually causing the DM Communicator to fire a **`GotoUser`** event in the RobotDaemon process.

The RobotDaemon activates an associated behaviour that currently uses the face identification module to compute the approximate location of the user and then calls the navigation system to drive the robot to that position in a polite manner.

Depending on the success of executing the navigation plan, eventually an appropriate event is fired, e.g. **`NavigatorGoalReachedEvent`** or **`NavigatorGoalNotReachableEvent`** .

This event is received by the DM Communicator and passed on to DM Core for processing.

The above scenario works both on the robot (with the real RobotDaemon) and in a simulator configuration (with our test server). The field test has been performed during the "Integration Meeting“ at MetraLabs in Ilmenau on April 11, using the robot from TUM-MMK.

6. Conclusion & Outlook

As we have seen above, the dialogue manager prototype has been developed and tested successfully, both on the robot and in our pure software environment. We have realized some basic scenarios to demonstrate its effectiveness, in particular with regard to the interaction with navigation and face identification services.

In the near future, the interfaces to the speech recognizer and other modules will be further refined as integration progresses. The specification process for the concrete data format for the transmission of the speech recognizer results to the dialogue manager, is currently ongoing; the same holds true for the brain computer interface.