



Modeling Engagement in a Multi-Party Human-Robot Dialog

Masterarbeit

im Fach Intelligente Systeme
an der Technischen Fakultät
Universität Bielefeld

von: David Klotz
dklotz@techfak.uni-bielefeld.de
Artur-Ladebeck-Str. 89
33617 Bielefeld
GERMANY

Betreuer: Dipl.-Inf. Julia Peltason
Dr.-Ing. Britta Wrede

November 30, 2010

Contents

Abstract	iii
1. Introduction	1
1.1. Motivation and Goals	2
1.2. Overview	2
2. Related Work	5
2.1. Levels of Understanding	5
2.1.1. Communication as a Collaborative Effort	5
2.1.2. Usage in Human-Machine Dialog Systems	6
2.2. Engagement in Communication	9
2.2.1. Definitions of Engagement	10
2.2.2. Situated Multiparty Engagement	10
2.3. The PaMini Dialog System	14
2.3.1. Interaction Patterns	15
2.3.2. The Task State Protocol	16
2.3.3. Interaction Management in PaMini	18
3. Scenario and Environment	19
3.1. The Humanoid Robot Nao	19
3.2. The HUMAVIPS Project	20
3.3. A Scenario for Multi-Party Interactions	21
3.3.1. An Example Dialog	22
4. The Engagement Subsystem	25
4.1. Interaction Management	25
4.2. Engagement Decisions	26
4.2.1. Engagement State, Actions and Intentions	28
4.2.2. The Engagement Control Policy	30
4.2.3. Engagement Tasks	31
4.2.4. The Engagment Process in Action	32
4.3. Scenario Configuration	32
4.3.1. Connecting Nao and PaMini	33

4.3.2. Speech Input Grammar	34
4.3.3. Interaction Patterns	34
4.3.4. Scenario-Specific Engagement Control Policy	36
4.3.5. Interfaces with the Real World	38
4.4. Analyzing Interactions	41
4.4.1. Data Collection	41
4.4.2. Results	43
5. Conclusion and Outlook	45
A. Speech Recognition Grammar	47
B. Recorded Interaction with the Engagement Subsystem	49
Bibliography	53
List of Figures	57

Abstract

Giving a robot the ability to interact naturally with humans using natural language dialog is an important goal of current research in human-robot interaction. Since most existing dialog systems target telephone-based, single-user dialogs, they are not sufficient to solve this challenge.

When a robot is situated in an environment containing multiple possible interaction partners, it has to make decisions about when to engage specific users and how to detect and react appropriately to actions of the users that might signal the intention to interact.

This thesis tries to address this by extending an existing system for the management of human-robot dialogs with new capabilities. This includes functions for managing multiple interactions, each with one or more distinct participants, and for making engagement decisions based on the actions and intentions of the present users and the current internal state of the dialog system.

1. Introduction

When we take the everyday interactions between people as a guide, a dialog based mostly on spoken natural language arguably seems to be the most common and effective form of communication. Yet, in the field of human-machine interaction, other modes of interaction still prevail in many situations. Modern dialog systems are catching up, but interacting with them is still missing the flexibility and fluidity that makes human face-to-face dialog so efficient. From the perspective of a scientist or an engineer trying to create better interactive systems, this makes the dialog between humans and machines an interesting topic for more research, because there is still much room for improvements.

When the machine in question is a robot situated in the physical world, a whole new set of challenges arise, because dialogs with a robotic system “must be asynchronous, mixed-initiative, open-ended, and involve a dynamic environment” [Lem+01]. The dialog is *asynchronous*, because a robot will always have to attend to different tasks (those given to it by the user or “background” tasks like obstacle avoidance), while still being able to respond to or address the user.

When the interaction should extend beyond simple commands given by the user, it will always be *mixed-initiative*, because the robot may e.g. need to ask the user to perform a task that the robot is unable to do. Mixed-initiative interaction is also needed as a basis for learning in interaction with a user. E.g. in a scenario like the one described in [Lüt+09], the robot can guide its own learning process by asking for missing pieces of information, like how to grasp an unknown object. Another example would be the home-tour scenario described in [Pel+09], where the robot optimizes its model about the environment by explicitly asking the human interaction partner for information meant to support or dismiss uncertain hypotheses.

In many of the future scenarios typically envisioned in research on human-robot interaction, e.g. a robotic receptionist at the front desk of a public or corporate building (see e.g. [BH09b]) or a robot assisting in the home (e.g. the robot companion Biron also described in [Pel+09]), the interaction is *open-ended*. The robot is supposed to be running continuously and the dialog needs to take this into account, for example by remembering what it was told in previous conversations.

Contrasting it with a typical telephone-based dialog system, a robot is also situated in a *dynamic environment*. On the one hand this presents new challenges to the dialog and the robotic system as a whole, but on the other hand this environment (and the fact that the robot is physically situated in it) can also provide “rich, relevant, streaming context

for the interaction” [BH09b]

Another challenge for the dialog component of a robotic system is that it may have to deal with more than one interacting person, sometimes even at the same time. The next section discusses some of the implications of that problem and how it was the most important motivation in defining the goal set out for this thesis.

1.1. Motivation and Goals

One of the traditional areas where automated dialog systems have been used has always been telephone-based dialog, e.g. for ordering tickets. In that domain, keeping track of which users the system is talking to is not an issue. There is always just one user, the interaction is one-to-one by design. While the telephone conversation is active, the user on the other end is assumed to be engaged with the system. The interaction is opened by placing the call and closed by simply hanging up the phone (cf. [BH09b, section 3.1]).

Comparing this with scenarios like those described in the last section, several important differences spring to mind. If a robot is to be used e.g. as an assistant in the household, in a lot of situations there will be multiple persons around, all of them potential interaction partners for the system. This leads to another set of requirements for systems that should be able to communicate with people in such an “open world” setting, because “participants in a dialog must coordinate their actions to establish and maintain an open communication channel.” [BH09b]

This process is commonly called *engagement* (cf. e.g. [Sid+04; BH09a; KH09] for differing but related definitions of the term in this context) and its integration into an existing system for dialog management has been the focus of the work done for this thesis.

Initially I started out by looking at models of turn-taking behavior and how to integrate them in human-robot dialogs, but quite early it became clear that a basic model of engagement seems to be a prerequisite for such higher level dialog enhancements. To put it simply: You first need to know who is there, who wants to interact with you and who you are already talking to before you can start thinking about when exactly it is appropriate to talk to them.

Therefore the goal of this thesis is to find a way to enable a robot to interact with humans in situations containing multiple persons or groups by trying to integrate a model of engagement (and the sensory functions it needs to make its decisions) into a dialog system.

1.2. Overview

The next chapter will talk about related work in different areas, like the modelling of the connection between participants in a dialog on different levels, the concept of engagement

and how these models are realized in different dialog systems for human-machine interaction. It will also describe the existing dialog system that was used in the implementation of this thesis.

The third chapter contains a description of the scenario serving as a testbed for the system developed in this thesis and the organizational and technical environment in which the work for this thesis was carried out. This provides an overview of the requirements the system is trying to fulfill.

The fourth chapter describes the main work done for this thesis, the implementation of the concept of multi-party engagement in a dialog system for human-robot interaction. It gives a general description of how the process of interaction management had to be changed to allow for different interactions with multiple participants, how the engagement decisions are made in particular and what kinds of sensory processes contribute to these decisions. This chapter will also give a short experimental evaluation of the realized system by analyzing recorded interactions with the system.

The final chapter presents a short conclusion and tries to provide some ideas for enhancements to the system developed in this thesis and for future work that may be needed in this area.

2. Related Work

This chapter gives an introduction to the main ideas and concepts having influenced the work carried out for this thesis.

It starts with a section on different levels used in the literature to characterize how the different participants in a dialog are connected. The section looks at the cognitive science foundations of this perspective as well as at the use of these levels for different purposes in several computer-based dialog systems.

The following section takes a look at the related problem of measuring or characterizing the engagement of dialog participants with each other or more specifically with an artificial dialog system, realized on a robot or as a virtual agent.

An existing dialog framework provides the practical foundations for the concepts realized in this thesis. This framework is introduced in the last section of this chapter.

2.1. Levels of Understanding

In the literature there have been several efforts to model the connections between the different participants in a dialog, from a purely linguistic or cognitive science perspective as well as from the perspective of building a synthetic dialog system. Most of them subdivide these connections into a hierarchy of states, levels or layers, characterized mainly by the progression from lower level information concerning the communication channel to higher levels related to the semantic content of the dialog. This section tries to summarize these important developments providing one of the core foundations of the ideas realized in this thesis.

2.1.1. Communication as a Collaborative Effort

Clark and Schaefer [CS87; CS89] provide a theory of human communication as a collaborative effort that is fundamental to most of the other efforts mentioned in the rest of this section.

Their main argument is that a dialog or discourse is not made up of utterances made by each of the participants in sequence, alternating according to some rules about when it is their turn to speak, but of “collective acts performed by the participants working together” [CS89]. These acts, called *contributions* by Clark and Schaefer, commonly consist of two parts, a *presentation* and an *acceptance*. The first part consists of the specification

of the content of a contribution (i.e. what the speaker wants to add to the conversation) by one participant and the second part is used “to establish the mutual belief that everyone else has understood that content well enough” [CS87, pp. 19,21–22]. They argue that this is intended to ensure the content becomes part of the mutual beliefs of all the participants, a process that is called *grounding* (see [CS87, p. 20]).

Clark and Schaefer propose a hierarchy of states a participant has to go through to accept a contribution:

- In the first state, the accepting participant *B* failed to notice that the presenting participant *A* had uttered anything.
- In the second state, *B* did notice that *A* had said something, but did not correctly hear what was said.
- In the third state, *B* correctly heard what *A* had said, but failed to understand the content of the message.
- In the fourth and final state, *B* correctly heard and understood everything *A* had said [see CS87, pp. 22–23].

2.1.2. Usage in Human-Machine Dialog Systems

Several authors have used at least some of the ideas presented by Clark and Schaefer in realizing dialog systems for the communication between humans and machines. They all used the proposed states in the process of grounding information (or similar concepts inspired by them) in similar ways, but to somewhat different purposes.

Feedback for Discourse Repairs at Different Levels

Brennan and Hulteen [BH95] extend Clark and Schaefer’s model to make it usable in the context of a dialog system for human-computer interaction. For this system, they make explicit use of these different states of understanding from the original model describing different types of feedback such a system could give at the respective levels. They split the original final state into two distinct states, used to distinguish the problems arising at the levels of syntactical parsing and semantic interpretation, adding three additional states on top of them that can be useful in a system to which users can delegate tasks.

They argue that problems in a dialog can arise in two different cases. The first occurs when a participant fails to reach one of the states necessary e.g. for accepting a contribution as grounded. That participant then has to give some kind of feedback as negative evidence notifying the participant who gave the original presentation that a problem has occurred. The second case happens when a participant thinks to have reached such a state, but the positive evidence given leads the original speaker to conclude the opposite. They also argue strongly that this is one of the reasons a dialog system should not only

give negative evidence when an error has been detected, but also positive evidence when everything seems to be going well [see BH95, p. 2].

They go on to provide examples for positive and negative feedback the system could give at the different levels to facilitate discourse repairs and describe a dynamically changing grounding criterion that determines at what levels the system will give this kind of feedback, depending on the dialog history, the physical environment and the nature of the current task. By this they want to ensure possibly interruptive feedback being given only when it is really necessary. This in turn is meant to reduce the required communicative effort from both parties involved.

Levels of Analysis in the Quartet Architecture

Paek and Horvitz [PH00] describe an architecture for dialog systems (called Quartet) that treats the process of grounding as action under uncertainty. It models some of the uncertainties involved using Bayesian networks and it uses the expected utility and similar probabilistic measures to make decisions based on that information.

The Quartet architecture analyzes the communication at four different levels, strongly inspired by the states from [CS87]. The first level, the *channel level*, looks at behaviors a speaker can execute to try to open a communication channel with a listener and at the fact that the listener has to be attending to the speaker to perceive that behavior. The next level is the *signal level* that tries to distinguish behaviors meant as communicative signals from others. Following these two lower levels are the *intention level*, modeling the core semantic content of the signals and the *conversation level* that looks at global goals of conversation, like carrying out a joint activity with all the participants.

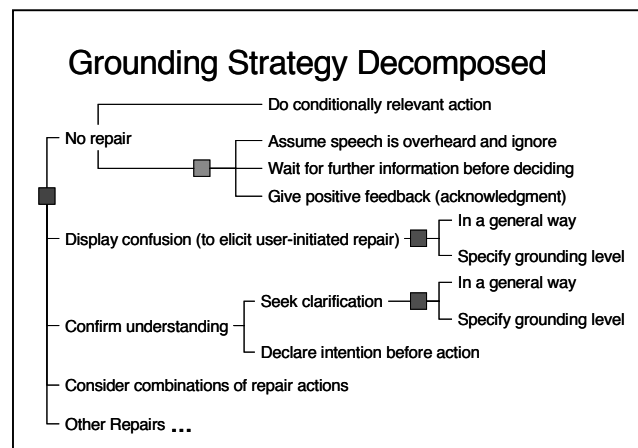


Figure 2.1.: A partial breakdown of possible repair and non-repair grounding strategies in Quartet. [taken from PH00, p. 4]

Information from all of these levels is used to make decisions about the actions the

system should take in a dialog, for example whether the system should try to initiate a repair or not. One interesting case happens when the system detects a signal at the signal level (e.g. a result from a speech recognition subsystem) without detecting an open communication channel at the lowest level. The system will then assume that the signal was not directed at itself and ignore it at the higher levels. Figure 2.1 shows the author's classification of the different grounding strategies used in the Quartet architecture.

Multi-Party Conversations with Virtual Agents

Traum and Rickel [TR02] present an information state approach as a model for a dialog system capable of holding conversations with more than two participants. Although the concept of a multi-party conversation in general could also extend to other possible configurations, the examples they present seem to be limited to situations where one human interaction partner is talking to multiple virtual agents in a simulated military training exercise.

Their system is structured in several dialog management layers and knows a set of actions that can change the information defining the state of each layer. The lower levels of the model are again focused on the establishment of communication channels between the participants. There is a contact layer keeping track of which agents can be reached, depending on the medium of communication (since the simulation includes messages sent via radio) and a layer that manages the attention state of all involved participants in regard to each other. There are also higher level layers for information regarding the current conversations (which is taken to encompass tasks like grounding information and turn-taking), social commitments and negotiations between the participants.

Another emphasis of their model is the management of multiple conversations, each with several different participants and possibly overlapping in time. Such a conversation can be opened or closed explicitly (with direct verbal or non-verbal cues) or implicitly (e.g. just by the direction or lack of attention). An example they provide of non-verbal cues for opening a conversation is the process of *distance* and *close salutations* that is made up of several steps like first gazing for a prolonged time and then approaching the other participant directly. The state of a participant in a conversation can also be changed, e.g. from passively overhearing it to actively participating in it.

Most of the lower level efforts could be summarized by the conclusion that “agents must reason about who they are talking to, who is listening, and whether they are being addressed or not” [TR02, section 2], leading to the concept of engagement in communication that is the focus of next section.

2.2. Engagement in Communication

If we take a closer look at the lower levels of the models presented in the previous section and related efforts, some similarities emerge to form something like a “least common denominator” for the handling of the communication channels in a dialog. This always includes ways to open and close these channels, and several authors emphasize the fact that this may happen using verbal cues (like greetings) or non-verbal behavior (often connected to the way mutual attention is signalled between the participants). In some models, which are rooted in a mixed-initiative mode of interaction, the system may also actively try to open such a communication channel by itself.

As an example, Bohus and Horvitz [BH09b] define several levels of core competencies for holding dialogs in the “open world” (see figure 2.2), which they take as a dynamic environment in which the dialog system is situated. In this open-world setting, people may enter or leave the conversation at any time and the dialog system has to cope with the various verbal and non-verbal signals meant to control the opening, closing and management of the required communication channels. They situate the process responsible for this at the lowest level of their hierarchy and call its area of responsibility *engagement*.

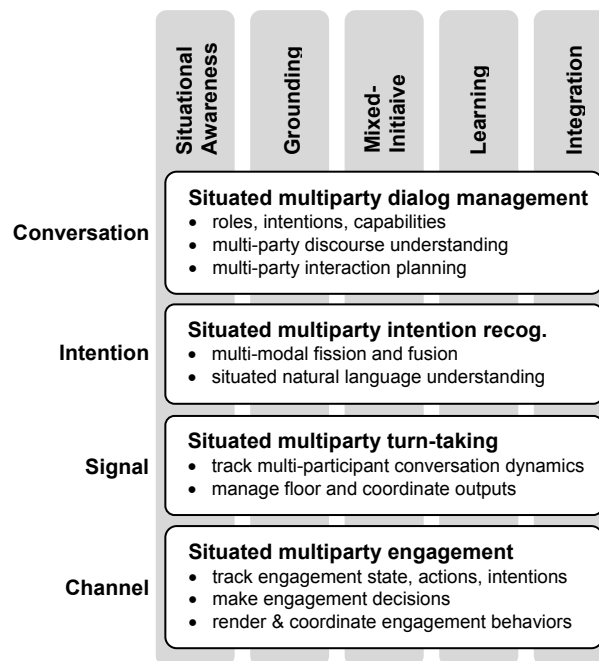


Figure 2.2.: Core competencies for open-world dialog [taken from BH09b, p. 4]

The next sections will first compare several definitions of engagement in communication and then take a closer look at how Bohus and Horvitz define their process of *situated multiparty engagement*, a definition having influenced the work carried out for this thesis

to a great degree.

2.2.1. Definitions of Engagement

In the context of modelling dialogs, engagement is defined in somewhat varying ways. Some authors think of it as a process involving rules on how communication channels are managed. One example of this is the definition from Sidner et al. [Sid+04], who see engagement as “the process by which two (or more) participants establish, maintain and end their perceived connection”. According to their definition, this process not only includes the initial contact and beginning of a collaboration, but also functions to evaluate whether to stay involved or when a conversation should be ended.

This can be contrasted with the view of Peters et al. [Pet+05], who see engagement mainly as a motivating force or a measure of how well a system manages to captivate the users interest. For them, this measure could be defined as “the value that a participant in an interaction attributes to the goal of being together with the other participant(s) and of continuing the interaction”. They also assert that engagement is strongly linked to the interest a participant has in the topic being discussed and see this as a cause for attention the user might give to the system.

Kok and Heylen [KH09] look at engagement in the context of turn-taking in a dialog. They define knowing the rules of engagement as the “knowledge of when it is appropriate or desired to say something”. To implement these rules in a dialog system on a robot or a computer, they insist that the system should be given at least the skills to recognize when it is being addressed and to detect when the speaker wants to hand over the turn to the system.

Bohus and Horvitz [BH09b; BH09a] adopt Sidner et al.’s notion of seeing engagement as a process. They extend it with capabilities they deemed necessary for dealing with multiparty interactions in an open-world environment. Since they, like Traum and Rickel, also allow for multiple open interaction with multiple participants each, they additionally characterize engagement as “the process subsuming the joint, coordinated activities by which participants initiate, maintain, join, abandon, suspend, resume, or terminate an interaction”. This implies that on top of the basic management of communication channels, they also see higher-level questions like who is taking part in the same interaction or when the system should actively engage a person as part of the engagement process. The next section will take a closer look at how they think this process should be structured.

2.2.2. Situated Multiparty Engagement

The model of *situated multiparty engagement* described by Bohus and Horvitz [BH09a; BH09b] can be characterized as consisting of three main functional parts. The first is concerned with sensing the environment of the system to gather knowledge about the potential interaction partners that can be detected in the scene. It uses this information

to decide if one of them is already interacting with the system (their *engagement state*), if somebody wants to start or leave an interaction (their *engagement actions*) and to try to correctly guess their *engagement intention* (i.e. if they might want to interact with the system without yet having actively engaged it). The second part uses this sensory information and combines it with higher level information (like the current state or goals of the dialog) to make decisions about when the system should try to engage or disengage with one of the users. These decisions are then realized into behaviors (like gestures, the making and breaking of eye contact or verbal greetings) by the third part of the engagement system.

The next sections will explain the different parts of this engagement model in more detail.

Engagement State, Actions and Intentions

The concept of an *interaction* takes a central place in Bohus and Horvitz’s model of engagement [BH09a]. They define an interaction as “a basic unit of sustained, interactive problem-solving” by the system and one or more other participants, which the authors designate as *agents*. It is important to note that the agents do not engage with the system in general, but engage (or become engaged in) a specific interaction. The configuration of which users are engaged in an interaction also is dynamic, since they can join or leave an interaction at any moment. The system itself can also only actively take part in one current interaction, but there can be several suspended interactions beside it. This can be used e.g. for modelling side-interactions subsuming, but afterwards returning to, the currently active interaction. This notion of an interaction can be compared quite closely to the information managed at the “conversation” layer of the system from Traum and Rickel [TR02] already partly described in section 2.1.2.

The engagement state, actions and intentions are modelled as a system of variables and probabilities that the system tries to keep updated for all the agents in all interactions. Some of them are inferred using sensory input taken directly from the scene surrounding the system, while some of them are updated only in reaction to actions taken by the agents or the system.

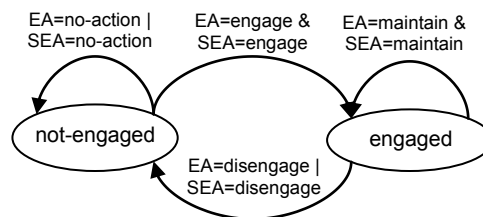


Figure 2.3.: Engagement state transition diagram. EA is the agent’s engagement action; SEA is the system’s action. [taken from BH09a, figure 2]

The *engagement state* of each agent in each interaction is modelled as a discrete variable, $ES_a^i(t)$ (where a denotes the agent, i the interaction and t the current timestep), which can take only two distinct values, *engaged* or *not-engaged*. The current state can be changed by the agent’s actions described next or by actions taken by the system itself. Figure 2.3 shows a visualization of the state space and the actions possible for one agent in one interaction. When agents try to engage the system, they don’t change automatically to the engaged state, since the system first has to respond positively to their request by also engaging them in return. Disengaging, in contrast, can be a unilateral action that either the agent or the system can take by themselves. If one side of an interaction wants to leave, there’s usually not much the other side can do to stop it.

The actions an agent can perform in regard to the engagement system are modelled with the variable $EA_a^i(t)$, which can take one of the four values *engage*, *no-action*, *disengage* and *maintain*. Which one of these actions an agent can really take depends on their current engagement state. While the agent is in the *engaged* state, the only possible actions are to *disengage* from the current interaction or to *maintain* their current engagement with the system. When the agent is not currently engaged (i.e. in the *not-engaged* state), it is possible to *engage* the system or to take *no-action*. Of course, the restrictions mentioned in the last paragraph (and visible in figure 2.3) apply to all tries to change an agents state by taking one of these actions. Which action an agent tries to take is estimated from a conditional probabilistic model of the form $P(EA_a^i(t)|ES_a^i(t), EA_a^i(t-1), SEA_a^i(t-1), \Psi(t))$. This model takes into account the current engagement state of the agent, the last engagement action which both the agent and the system have taken and some “additional sensory evidence”, $\Psi(t)$. This additional evidence $\Psi(t)$ is used to take detection results into account, e.g. for explicit engagement cues like salutations (e.g. “Hello!” or “Good bye!”) or for directly approaching and attending to the system.

Apart from the engagement state and the possible engagement actions, the model also introduces a third variable, the *engagement intention*, $EI_a^i(t)$. Like the engagement state, this variable can have the two values *engaged* or *not-engaged* for each agent. This intention is modelled separately from the engagement actions since someone can have the aim of engaging the system without yet taking any actual actions, e.g. when standing in a line waiting to interact with a receptionist. In general, this variable is also used as an estimate if an agent would respond positively if the system should try to engage them. This variable is again modelled as a conditional probabilistic model, this time of the form $P(EI_a^i(t)|ES_a^i(t), EA_a^i(t-1), SEA_a^i(t-1), EI_a^i(t-1), \Psi(t))$. This means that the current intention of an agent is conditioned on the current engagement state of the agent, the previous actions by both the agent and the system, the agent’s previous intention and again a variable for “additional evidence”, $\Psi(t)$. In this case, this evidence could include more implicit cues for engagement, like sustained attention to the system.

Engagement Decisions

Based on these models and other information, the *engagement control policy* is able to make high-level decisions about which users to engage or disengage in specific interactions. The range of actions the system can take consists of the same four actions that a user can take as an engagement action (described in the last section). Together with the user’s action they control the engagement state as shown in figure 2.3. This decision is termed the *system engagement action* (or *SEA*) in this context. The actual surface realization of the SEA is controlled by lower-level engagement behaviors, which are described in more detail in the following section.

Bohus and Horvitz [BH09a] define the system engagement action as a function of the form $\pi_{\text{SEA}}(\{\mathbf{E}_a^i\}_{a,i}, \{\mathbf{H}_a\}_a, \mathbf{\Gamma})$. This means that the decision is based on the engagement state, action and intention \mathbf{E} of all agents in all interactions, on high-level information \mathbf{H} about all agents (such as an agent’s long-term goals) and on other global context $\mathbf{\Gamma}$ (like the history or the current state of the interaction).

This definition is kept intentionally broad and abstract by the authors, because the real rules used for the SEA can be quite specific to an application or a scenario. The receptionist scenario in [BH09b] includes the example of queue management, where the virtual receptionist system suspends the interaction with the current user to engage a user waiting in line to tell them they will be “attended to shortly” or even handles their request while the current user fills out a form. Another example comes from the trivia game scenario of [BH09a], where the system engages a bystander and asks him to help the current player when he is not doing well in the game.

Engagement Behaviors

The higher-level engagement decisions (i.e. system engagement actions like *engage* or *disengage*) have to be rendered into real actions by the virtual or robotic embodied agent representing the dialog system. Bohus and Horvitz [BH09a] describe a lower-level *behavioral control policy* for this in the form of $\pi_{\text{SEB}}(\text{SEA}, \{\mathbf{E}_a^i\}_{a,i}, \Psi)$. This function takes the higher-level decision, the current engagement information for all agents and another additional evidence variable. In this case, the additional evidence is comprised of other information that was extracted from the scene and could be important for the action realization. This function is then used to sequence a series of actions meant to realize the higher-level engagement goal.

For example, the *engage* system action is realized in [BH09a] as a series of three sub-behaviors. The first of them tries to establish the attention of the user that should be engaged (for example by looking in his direction and waiting whether his focus of attention is also on the system and maybe calling out to him). The second optionally greets the user with a message that can be specified as a parameter to the engage-action. Finally, the last sub-behavior monitors if engaging the user really succeeded. It is important to

note that any of these sub-behaviors can fail (e.g. if the user doesn't react or just walks away) which would also be signalled to the higher-level engagement control system.

In contrast to the higher-level engagement control policy, these lower-level engagement behaviors often can be re-used across different settings and scenarios. The authors also have the idea that some of these behaviors in the future might be learned, e.g. from a corpus of human-human interactions.

2.3. The PaMini Dialog System

Bringing together dialog modelling and interactive robotics never is an easy task. On the one hand, there is the software-engineering challenge of integrating the dialog management with the diverse and often asynchronous subsystems of a typical robotic system (like gestures, other movements or navigation). The dialog may need to delegate tasks to them, some of which may take a long time to complete, some could fail or may be in need of more specific instructions. Some of those sub-systems may even want to delegate tasks to the dialog management system themselves, for instance when a component wants to inform the user about an event or the system's state (e.g. "My battery is nearly empty, i need to recharge!") or needs some information from a human interaction partner (e.g. "Is this the living-room?").

On the other hand the dialog model itself needs to be flexible, to support the diverse and often long-running interactions a robotic system could perform in such a scenario. It also needs to make use of being situated on a robot in the real world. It needs to know some things about the environment (including the robotic system itself) and it needs to make use of this information when interacting with a human partner.

Based on the experiences made with previous efforts to build dialog systems for robots (see e.g. [LW07]) and the requirements learned from implementing several human-robot interaction scenarios (see e.g. [Pel+09] or [Lüt+09]), Peltason and Wrede [PW10] have developed an approach to modeling the dialog with an interactive robot. A framework based on this approach has the goal of making it easier to model dialog situations and to integrate that dialog with the rest of the robotic system. This framework, called PaMini (which is short form "Pattern-based Mixed-Initiative Interactions") forms the basis for implementing the ideas presented in this thesis.

As an approach to satisfying these diverse needs, the PaMini framework combines a collection of generic interaction patterns with an abstracted interface to submit and receive tasks to and from different components in a robotic architecture. The interaction patterns have the goal of encapsulating typical interactions into configurable "building blocks for human-robot interaction" [PW10]. They are described in more detail in the following section. The task state protocol, which is detailed in section 2.3.2, provides an abstraction of the interactions between the dialog manager and other components using a general set of states and transitions for delegated tasks.

There are flexible interfaces for plugging in different modules for input and output into the dialog framework. Since the main dialog engine is agnostic to the specific type of in- or output used, this makes it easy to realize multimodal interactions, using e.g. gestures instead of (or in addition to) the speech modality. Existing modules include one for receiving input from the HMM-based speech recognition engine ESMERALDA (see [Fin99]) and one for sending speech output to the speech synthesizer MARY (see [DFK10]). While the former was also used for the speech input in the scenario realized for this thesis, the latter was replaced by the the built-in speech synthesis engine of the robotic platform used, described in section 3.1.

2.3.1. Interaction Patterns

Analyzing the interactions in scenarios of human-robot interaction like [Lüt+09], [Pel+09] or [Beu+08] helped to identify several “recurring conversational patterns” [PW10] of exchanges between a human user and the dialog system and between the dialog system and other components of the robot’s software that are found repeatedly in such scenarios. For the PaMini dialog framework, these patterns were extracted and explicitly modelled to allow reusing them in different new scenarios. This idea (and their name) were inspired in part by the concept of design patterns in software-engineering, providing generic solutions to typical problems of the field.

The interaction patterns model the interaction between the robot and the human interaction partner as a set of states the dialog can assume and a set of valid transitions between those states. These transitions can be activated either by input from the user or by events concerning the execution of delegated tasks by different software components, which is discussed in more detail in the next section. When a transition becomes active, it can then in turn produce a dialog act of the robot as an output. Each pattern is modelled as a state chart (see [Har87]), a generalization of a finite state machine that could be seen as a transducer (consuming input from the user and the task events and producing robot dialog acts as output) with added actions that are executed at each state.

Figure 2.4 shows two interaction patterns as an example. The first one, called *HumanSimpleActionRequest*, allows a human interacting with the robot to ask it to perform some kind of action. This request is defined as a dialog act (called *H.request* in the example) by the human interaction partner, which starts the interaction with that specific pattern. In the state called *initiate* in the first example, the dialog manager then gives the task to another component that is responsible for actually making the robot perform the action that was requested. Afterwards, the dialog just reacts to updates concerning the execution of this task by giving feedback to the user in the form of several dialog acts of the robot (e.g. *R.acknowledge* or *R.apologize*). The second example pattern, *RobotSuggestion*, is different only in so far as it is not started by a dialog act from the human, but by some other controlling component on the robot wanting to make a suggestion and to

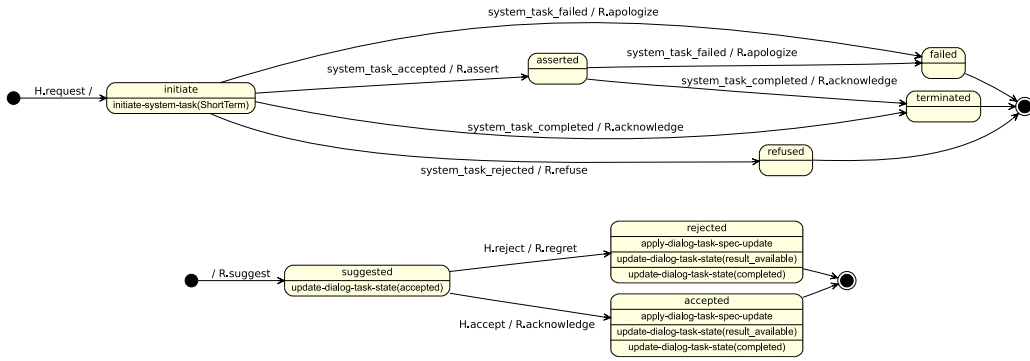


Figure 2.4.: Two interaction patterns: HumanSimpleActionRequest (which allows a human interaction partner to request an action from the robot) and RobotSuggestion (which allows the robot to suggest something that the human may either accept or reject).

get feedback from the user about it.

As can be seen in the examples, the interaction patterns are specified at a very abstract level. This is done intentionally, to make them reusable for a variety of different situations. The person designing a specific dialog can take a subset of all the existing interaction patterns and configure them for the real conversational situations in which they will be used. This involves configuring conditions for the different dialog acts the human can make, the surface form (i.e. the speech or gesture output) for the robot’s dialog acts and specifications for the tasks being delegated to other components. As an example, the dialog act *H.request* from the first example in figure 2.4 could be specified as some form of the command “Robot, please follow me!” and the dialog act *R.acknowledge* could be configured to make the robot say something like “OK, I am following you, now.”

It should also be noted that the dialog manager allows for a flexible way of combining multiple active patterns. One pattern can be interrupted by another pattern, which gets pushed on top of the first pattern in the manner of a stack. When the dialog receives input, it first tries to match it against the top-most pattern of that stack. If that fails, the other patterns in the stack will be tried afterwards. To stop this process of pattern interleaving from getting out of control, the dialog manager contains a set of rules (that can be adapted for individual situations) which restrict which patterns can be interrupted and interleaved with other patterns. To provide an example, the interleaving rules could allow a pattern for answering a simple question to be interleaved with one controlling a long running task (like grasping an object in [Lüt+09]).

2.3.2. The Task State Protocol

To achieve the generalized delegation of tasks between different components forming parts of the concept of interaction patterns described in the previous section, it was necessary

to define an abstract interface between components that is independent of the precise nature of the task. To this end, PaMini communicates with other components using the *task state protocol*, which is based on the XCF middleware framework [Wre+04].

This protocol defines several states a delegated task could assume, which can be seen in figure 2.5. When the requesting component (often denoted the *task submitter*) submits a task for execution, it first becomes *initiated*. The receiving component, also known as the *task handler*, can then decide to *accept* or to *reject* the task. When accepted, the execution of the task can be *running* for a while and may even produce *intermediate results*. The submitter can try to *update* the specification of the task or to *cancel* its execution, but the handler has the right to reject these requests. Finally, the task may be successfully *completed* or could also *fail* for some reason. Typically not all components make use of all of these states, but in this generalized form a very diverse set of tasks can be covered.

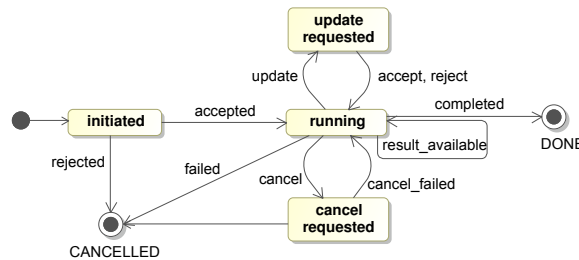


Figure 2.5.: The task life-cycle [taken from PW10, figure 2].

The task request that the submitter submits is always composed of two different parts. The first part describes the state of the task execution that was just described. This part is always the same for all components and tasks. The second part contains the actual description or specification of the task itself, which in most cases is very specific to the individual components. This distinction is what enables the dialog (and other components) to work with the tasks and react to changes in their state while being agnostic of their specifics. When the submitter or the handler update the state or the specification of a task, event notifications get sent to the respective counterpart, enabling them to react to those changes.

The use of the task state protocol for the dialog manager is twofold. First it is used to allow the patterns to delegate tasks to other components of the robotic system and to react to state changes of these tasks in a uniform way. An example of that can be seen in the first pattern from figure 2.4, where the different task states (*failed*, *accepted* etc.) are used to define the progress through the different states of the interaction pattern. Secondly, other components may also delegate tasks involving user interaction to the dialog system and can also react, e.g. when the dialog rejects to fulfil their request.

2.3.3. Interaction Management in PaMini

The PaMini dialog framework contains some simple ways to control the opening and closing of the interaction process. There is a policy that is used to decide if and when the interaction should be opened and there are different actions that can be executed from the states of an interaction pattern to reset or to close the interaction. One typical configuration is to have the policy set to allow input for one specific interaction pattern (e.g. one called *HumanInteractionOpening*) to open the interaction and to have a second pattern (e.g. *HumanInteractionClosing*) that is used to close the interaction.

This is a far simpler model than e.g. the detailed management of interactions in the models from section 2.2, but for most single-user interactions it fulfills its purpose well.

3. Scenario and Environment

This chapter tries to give a description of the technical and organizational environment that shaped the development of this thesis. It takes a look at the robotic platform used, a research project providing some thematic context and how they influenced the scope of this work.

In the last section it will also introduce the scenario the system developed for this thesis tries to realize. This scenario forms the guideline for turning the abstract ideas about engagement in communication into a tangible system.

3.1. The Humanoid Robot Nao

One of the goals of this thesis was to implement the ideas presented here in a system that is situated in the real world and can interact with humans in a natural way. Using a robotic platform supports this goal by enabling direct interaction in the physical world that can borrow much from already established models of interaction between humans. By moving towards systems that look more and more human-like however, one also runs the risk of raising the expectations towards the systems capabilities to unrealistic levels. This also has to be kept in mind when building or choosing a system for human-robot interaction.

The humanoid robot Nao (see e.g. [Ald10]), built by Aldebaran Robotics and shown in figure 3.1, provides an interesting platform for research in this area. Its head is equipped with four microphones usable for auditory perception tasks like speech recognition or sound source localization, stereo speakers for producing speech and other sounds and a visual system with two VGA cameras usable for detecting faces and other visual cues. Nao has a fully articulated humanoid body with 25 degrees of freedom, but is only about 58 cm in height, a fact that has to be considered when designing scenarios relying on face-to-face interaction between the robot and a human partner. Contained in the robot's head is an embedded computer with a 500 Mhz AMD Geode CPU and 256 MiB of RAM running the Linux operation system. This allows for autonomous operation of the robot, while the provided wireless and wired network connections also allow for embedding it in a larger distributed system.

The provided software environment for programming the robot already comes with a collection of features that enable fast prototyping of interactive scenarios, e.g. a functional omni-directional walking algorithm, a module for performing text-to-speech tasks



Figure 3.1.: The robot Nao from Aldebaran Robotics.

and simple face detection capabilities. The robot's exterior design tries to strike a balance between being human-like enough for interaction purposes but still being clearly recognized as a machine. Nao has also been chosen as the final target platform for the HUMAVIPS project, described in the next section, which confirmed the choice for realizing this thesis using the same robot.

3.2. The HUMAVIPS Project

HUMAVIPS (short for Humanoids with Auditory and Visual Abilities in Populated Spaces) is a project in the European Union's seventh framework programme that aims to provide humanoid robots with the necessary skills to interact successfully within groups of people. The robot should be able to navigate such an environment, to recognize and identify persons and to communicate with them in a natural way (see [HUM09]). This thesis was written in the larger context of research done for this project.

Reaching this goal involves research into perceptual mechanism, e.g. fusing auditory and visual data to generate hypotheses about the environment, but also into how these sensory abilities could be used to enhance the behavior the robot shows when engaging people in a dialog. Interactions in the scenario envisioned by the project also necessarily involve multiple persons, which leads to research into better multi-party dialog abilities.

The work done for this thesis tries to enhance the dialog system, allowing it to interact with groups of people, making it easier to integrate perceptual cues and to experiment with their significance for the dialog.

3.3. A Scenario for Multi-Party Interactions

The scenario that was realized had to include some specific types of interactions to make it interesting for studying the realization of engagement rules in a multi-party setting. As a basic requirement, it has to include interactions that typically involve multiple persons. It should also present the possibility to showcase the dynamic management of multiple separate, but possibly overlapping interactions and should allow people to join or leave an interaction at various points in time.

To demonstrate as much of this as possible, a simple scenario was chosen, consisting mainly of the social interaction of having the robot introduce multiple persons to each other. The robot asks individual people for their name and actively tries to draw silent bystanders into the interaction. If it succeeds in this, it first tries to find out if the participants already know each other. If this is not the case, it proceeds by introducing them. To add the challenge of overlapping interactions, interrupting an ongoing conversation by asking if the robot has already seen a specific person today is also possible. See figure 3.2 for an example interaction in this scenario.



Figure 3.2.: Two persons interaction with Nao in the introduction scenario.

In all of these situations, the robot is seated on top of a table, enabling it to engage in face-to-face interactions with human users regardless of its own height (see section 3.1). On the other hand this restricts the robot's movement, stripping the interactions of the possibilities moving around in the room would offer. For the extent of this thesis this limitation is deliberate, as it also simplifies the sensoric processing, but extending it to scenarios with a truly mobile robot is definitely planned for the future.

Besides the standard interaction patterns, e.g. for posing or answering questions, this

requires the dialog system to have the ability to detect and discern multiple people and to reason about their actions and intentions in so far as they are relevant to their engagement in the dialog. It also requires the explicit management of multiple interactions, each with their own state and different engaged users.

3.3.1. An Example Dialog

Table 3.1 shows the basic flow of events in an example dialog that follows the scenario described above. It shows at least two separate interactions with different participants. One of them happens when the robot Nao tries to introduce two human interaction partners, P1 and P2. The other interaction is between the robot and P3, a third user that just walks by and engages the robot in a side interaction by asking it a question. Asking the second user to join the first interaction could also be considered a separate interaction. But after he decides to join in, he becomes part of the longer-running original interaction.

The diagrams in the last column of the table are inspired by the notation from [BH09a]. They show the current state of the interactions between the users and the dialog system at that point in time. The circle symbol ● represents the system itself, the dark square ■ an active interaction and the lighter square ◻ an interaction that is currently suspended. The numbers represent the individual users.

Looking at the individual steps of the interaction, one can identify several actions that show the general direction for the intended engagement strategy. In step 1 and 4, the human users show the intention to interact with the robot. In this example, the robot only takes the second case as an opportunity to actively engage the second user. The dialog also includes examples of direct verbal cues for engagement or disengagement, e.g. the greeting in step 2 or the words of parting in step 16. Step 5 is an example of suspending one interaction to open a side-interaction with P2, who will later (in step 7 or 8) join the now reactivated first interaction. Suspending an interaction and opening a new one happens again in step 12, but this time the robot does not politely excuse to the users engaged in the first interaction, because the question of P3 has to be answered quickly. Disengaging can also happen implicitly by just leaving, as is shown in step 15.


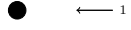


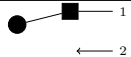
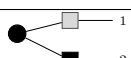
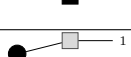



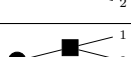
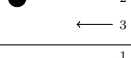
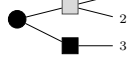
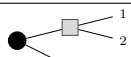
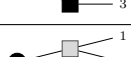
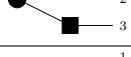
<i>Actions</i>	<i>Dialog state</i>
1. P1 enters, positions himself in front of Nao and looks at it.	
2. P1: Hello, I'm Sebastian.	
3. Nao: Hello Sebastian, I'm Nao. Nice to meet you!	
4. P2 enters the room, looks at Nao.	
5. Nao (to P1): Excuse me for a second!	
6. Nao (to P2): Hello! I'm currently talking to Sebastian, do you know each other?	
7. P2: No, I don't know him, yet.	
8. Nao: Then let's introduce you two, what is your name?	
9. P2: I'm Johannes.	
10. Nao: Johannes, this is Sebastian. Sebastian, this is Johannes!	
11. P3 enters, goes straight to Nao, asks:	
12. P3: Did you see Julia today?	
13. Nao (to P3): No, I did not see her today.	
14. P3: Ok, thanks, I'll have to look for her myself.	
15. P3 leaves.	
16. P1: We'll have to leave now. We have a meeting in 5 minutes!	
17. Nao: Bye, Sebastian! Bye, Johannes!	

Table 3.1.: Example dialog.

4. The Engagement Subsystem

The scenario described in the previous chapter was realized by extending the dialog system described in section 2.3 with a subsystem implementing a model of engagement. This model roughly follows the ideas from Bohus and Horvitz [BH09a], adapting them to fit the needs and the environment presented earlier.

This chapter first describes the architecture of the newly developed parts of the system, how they work together to track the user's engagement state and actions, to make engagement decisions and how those decisions are finally rendered into actual behavior. In the process it also describes the interfaces that allow the system to be extended, e.g. to incorporate sensory information or to change the realized behavior. Following this is a description of how this system was used to realize the desired example scenario.

4.1. Interaction Management

In the existing dialog system, all dialog state was kept inside the same interaction context. The interaction could be opened by specific actions taken by the user and could be closed or reset (see section 2.3.3 for a closer look at this), but this mechanism is not really flexible enough, especially if dynamic multi-party interactions are the goal. I extended this using the concept of an explicit unit of problem-solving or working together (see section 2.2.2 for the motivation for this approach), called an *interaction*.

Such a notion of an interaction helps partition the continuous and possibly indefinite space of encounters between a situated robotic system and its users into manageable units. Perhaps most importantly, several interactions can be layered to distinguish between multiple overlapping interactive tasks. While the robot can only actively participate in one interaction at a time, several others can be kept track of in a suspended state and may later become active again. The interactions can be opened or closed and opening a new interaction might suspend another one currently being active. Users can be engaged in or disengaged from specific interactions. Going back to the concept of the interaction patterns (described in section 2.3.1), an interaction also provides a context for patterns to be grouped together and executed separately from patterns in other interactions.

Technically, the interactions are integrated into the dialog system as a stack, the top-most interaction on the stack representing the currently active one and all interactions below it being suspended. Figure 4.1 shows an example snapshot of the current dialog state. It shows an active interaction and a suspended one below it. The active interaction

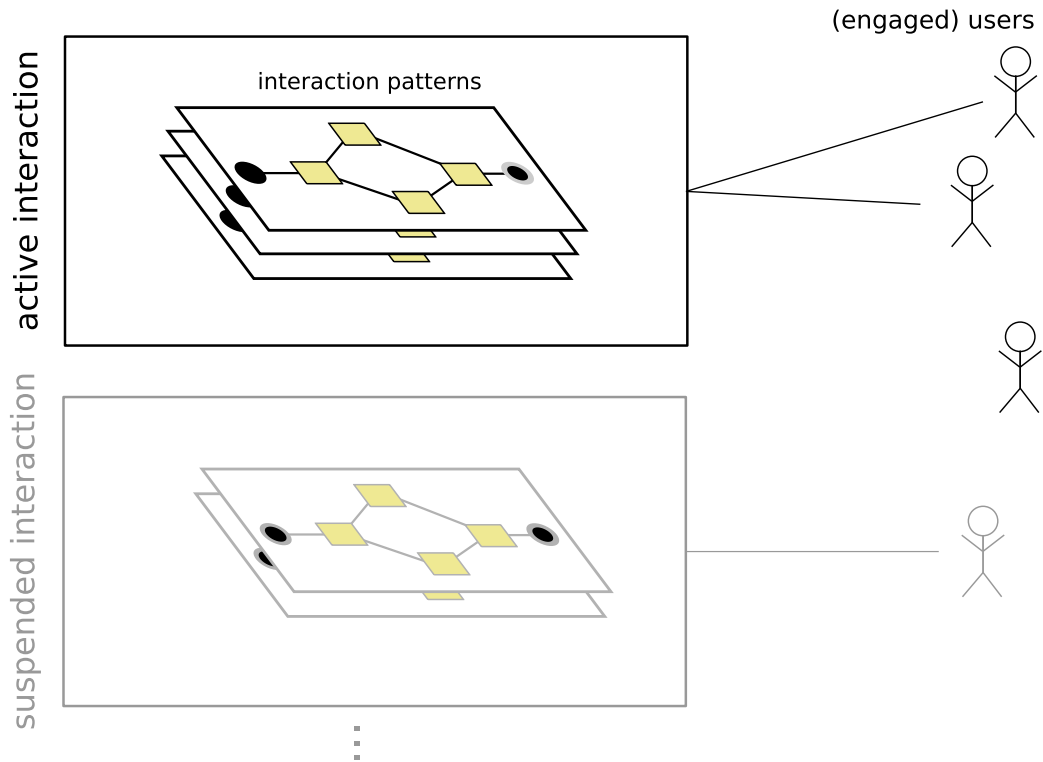


Figure 4.1.: The dialog state showing active & suspended interactions, engaged & not-engaged users and interaction patterns inside the interactions.

has two engaged users, the suspended interaction only one. There is also one user who is not engaged in any of the open interactions. Each interaction also contains its own set of active interaction patterns waiting for further input.

4.2. Engagement Decisions

The engagement subsystem comes into play in two different basic situations. The first happens when the dialog system receives input from a human user. Such an input could come from a speech recognizer, but input sources for other modalities are also possible, e.g. for reacting to gestures or other forms of recognized actions. In this case, the engagement system acts as a filter for the input, determining what should be done in reaction to the input and the current engagement or general dialog state. Actions taken in response can include opening or closing interactions, engaging users in specific interactions (or disengaging them) and processing the user's input using the interaction patterns associated with an interaction.

For the second case, the system regularly checks the current engagement and dialog state even when no input is received from the users. This is done to enable the system

to also actively make engagement decisions (e.g. trying to engage a user in a specific situation) by itself, allowing the system to take the initiative in an interaction. It also allows it to react quickly to other changes in the engagement situation that might not be seen as an explicit action by the user that could be represented as an input. The actions the system can take in these situations include basically all those that can also be done in response to user input, apart from processing an input.

Since the exact rules for these decisions are often quite specific to an interactive scenario, the policy deciding what has to be done in which situation is not fixed in the dialog system itself but can be extended or replaced easily. Later sections of this chapter include a concrete example for the rules used in such a policy. To provide the basis for making these decisions, the system includes mechanisms for deciding which engagement action is represented by an input from the user and for determining if a user currently has the intention to interact with the system.

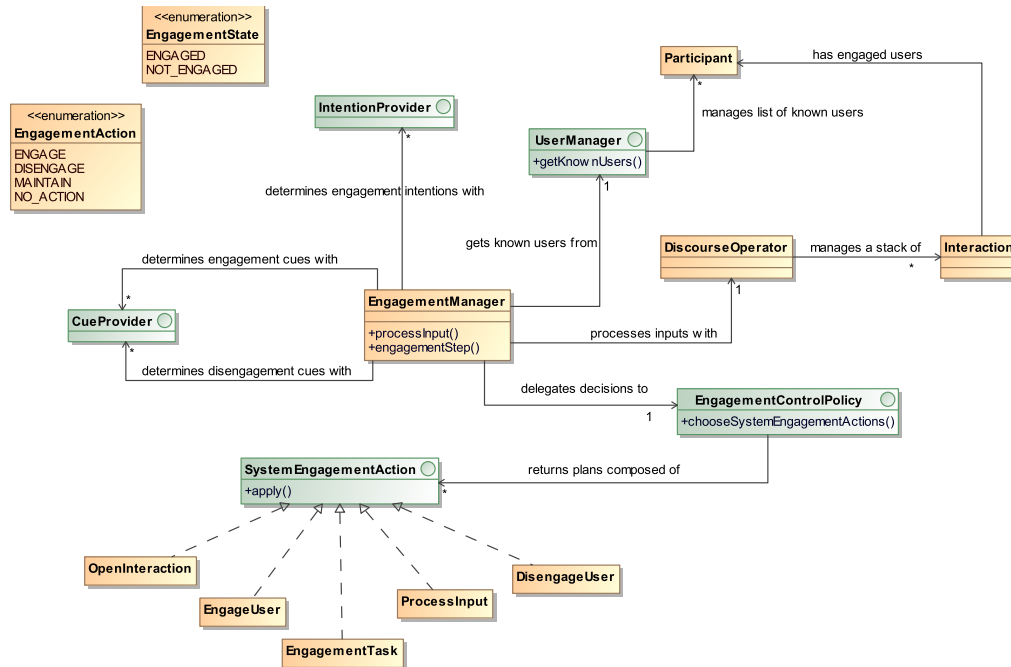


Figure 4.2.: An architectural overview of the engagement subsystem.

Figure 4.2 shows an architectural overview over the classes and interfaces used in the relevant parts of the dialog system. The *EngagementManager* provides the central hub for everything happening in the engagement subsystem. It receives all the input directed at the dialog system through its various *processInput* methods and also contains a separate thread performing the regular checks described above through the use of the *engagementStep* method. In both cases, it first collects all the necessary information

(for example from the information providers described in the following section) and then delegates the decision to the *EngagementControlPolicy*, described in section 4.2.2. This policy returns a “plan” of actions the system should take, which are then applied by the manager. Tasks like the processing of input or the opening and closing of interactions are in turn delegated to the *DiscourseOperator*, which manages the stack of interactions described in the previous section. Each of those interactions is represented by an instance of the class *Interaction*.

4.2.1. Engagement State, Actions and Intentions

It is clear that any dialog system intended for verbal interaction needs a working system for receiving speech recognition input, but a system intended to react to subtle changes in the engagement behavior of its users has more requirements on its sensory components. Such a system needs to know which interaction partners are present (and possibly where they are in relative to the robot and each other), it needs to know the significance of their actions regarding the engagement process and it needs to recognize if they want to interact with the system (and when they want to stop interacting with it). The real world (or rather the robot’s perception of it) gives us many clues for this, from the user’s focus of attention to gestures that could be recognized or information that can be gathered from the user’s position and movement relative to the robot. This section presents an approach trying to systematically integrate information about cues and hints like this into the dialog management.

The engagement subsystem includes a generic interface for integrating hypotheses about the known users, called the *UserManager* in figure 4.2. This helps by providing a list of known users and assigning them a distinct identifier to distinguish them from each other, but it could later be extended to keep track of more information about the individual users. Implementation of this interface will mostly integrate information from different sources to form person hypotheses that are stable in time and overcome the drawbacks of individual sensor modalities.

The engagement state of a user and the engagement actions a user may take are modelled using the enum classes *EngagementState* and *EngagementAction*, respectively. The engagement state (which is managed separately for each interaction) can only be *engaged* or *not-engaged* and the engagement actions are chosen from the set of *engage*, *disengage*, *maintain* and *no-action*. Since the guesses for the user’s engagement intention can have the same values as the engagement state, the same enum type is used in this case. All of these types and their values correspond closely to the variables *ES*, *EA* and *EI* as they are used by Bohus and Horvitz (described in more detail in section 2.2.2).

When receiving an input from a human user, the system will try to determine which action (if any) in regard to the engagement state this was meant to represent. To do this, the *EngagementManager* includes two different sets of *CueProviders* (see figure 4.2),

one for determining if the input received was a cue for taking an *engage* action and one for detecting cues for disengagement. These information providers are an interface for plugging in different mechanisms for detecting engagement cues. As a first step, the system determines the current engagement state of the user from which the input originated, i.e. by checking if they are engaged in an existing interaction. Depending on this state, the system then checks either the engagement cue providers or the providers responsible for disengagement cues to determine their confidence that this input was meant as such a cue. This value is in turn used for calculating confidence values for the different types of engagement actions. Examples of these cue providers could include keyword spotters or other kinds of providers that look at the content of speech input and providers that relate actions such as waving or nodding the head to their meanings in a dialog.

Another important information for the engagement decision process is the intention that a user has to engage. As was already discussed in section 2.2.2, this variable determines users that might want to interact with the system without necessarily having taken any explicit action in that regard, yet. It could e.g. be used to detect users that would probably respond positively, should the system try to engage them. To gather information about this, the *EngagementManager* includes a set of *IntentionProviders*, an interface meant to facilitate incorporating different sources of information about the user’s intention. These providers also return a confidence value for a specific user. These values are then used in combination to determine the general confidence that a user might have the intention to interact with the system. Many examples of providers in this category will be based on some measure of attention the user directs to the system, e.g. by trying to determine if and for how long the robot has been the user’s visual focus of attention.

In the case of both the cue providers and the intention providers we have a set of multiple providers, each returning a confidence value in the range of $[0, 1]$. These values are combined into general confidences for the different possible values of their respective variables (the engagement action and engagement intention described above) in the same way. First the values of all the different providers of the same category are summed according to the formula:

$$\sum_{i=1}^N w_i p_i(x)$$

In this formula N is the number of providers, w_i is a weight assigned to the i -th provider when registering it and $p_i(x)$ denotes the value the i -th provider returned for the input or the user x . In the case of the cue providers, this value is then used as the confidence for the input being an *engage* or *disengage* action, in the case of the intention provider it is used as the confidence that the intention should have the value *engaged*. The respective other values, for which no provider is used (*no-action* or *maintain* for the actions and *not-engaged* for the intention) are initialized to a fixed threshold value. Finally, only the value with the highest confidence is then passed on to the next step in the process, the

engagement control policy described in the next section.

4.2.2. The Engagement Control Policy

As has already been mentioned, the engagement control policy (a class implementing the *EngagementControlPolicy* interface shown in figure 4.2) is involved in determining the system's course of actions in two different situations. When an input is received and the steps described in the previous section have been carried out, the *EngagementManager* asks the current engagement control policy for actions that should be performed by the dialog system. To make this decision, the policy gets access to the input itself, the state of the current interaction (including information on which interaction pattern this input would match), to the engagement action this input represents and the user's current engagement intention (the latter two estimated by the process described before). It can then use as much of the information as it requires to assemble a list of actions that will form the system's reaction to the input.

The engagement manager will also perform a special engagement step at regular intervals. In this step, it will first determine the intention of all known users in the same way it determines the intention of the single user that produced an input. It will then collect the current state of all active and suspended interactions, including the engaged users and the interaction patterns currently being active in each interaction. Both of these sets of information are then passed on to the engagement control policy to determine if the system should take any actions by itself in this situation. A typical way to implement the decision making process in both cases will consist of checking a set of basic if-then rules handcrafted for special situations, e.g. of the form "IF [some conditions on the current state and/or the input are met] THEN [return these actions]". More complex variations are of course also possible, e.g. it would be interesting to try to build a policy that performs according to patterns learned from interactions, either between human users or users and the system.

Since the engagement control policy defines actions to be taken in specific interactional situations, it has to be tailored to some degree to the scenario of interactions that should be realized. To enable this, the realization of the policy can be changed in the engagement manager, even at the runtime of the system, if that is desired in a particular situation or scenario. It is also possible to create a policy implementing some general situations and extending that at a later point to provide support for more specific situations. The engagement subsystem currently contains a very basic default policy that is designed to mimic the behavior of the dialog system without any active engagement components.

The actions returned by the engagement control policy in both of the general use cases described above are objects implementing the interface *SystemEngagementAction*. The policy will always return a list of these action objects, which will be executed or applied by the engagement manager in the list's order. When one action is applied, it receives the

previous action’s result and the *DiscourseOperator* allowing it to manipulate the current interactions or to process inputs. It then has to perform the action it was designed to do and return a character string as a result. This result is meant as a light-weight way to pass results from one action to the next, but is also used by the engagement manager to determine if the application of an action has succeeded or not. When an action returns a result indicating failure, the engagement manager will stop executing the rest of the list of returned actions. The engagement subsystem comes with a set of default system engagement actions for opening interactions, engaging or disengaging users in interactions, for processing user inputs and for carrying out abstract “engagement tasks”, which are described in the next section. A specific policy can also return specialized actions characteristic for tasks only relevant in that scenario.

This use of the policy can be compared to the way the engagement control policy determines the system engagement action *SEA* in Bohus and Horvitz [BH09a], although they define this action at a much higher level of abstraction and only later determine the particular actions meant to realize this decision.

4.2.3. Engagement Tasks

As part of the actions described in the previous section, the engagement control policy can return units of action called *engagement tasks*. These tasks are specified at a very abstract level and contain only a name denoting the action to be done (e.g. “EngageUser”) and a list of users affected by this action. They are submitted using the task state protocol described in section 2.3.2 and can be handled by any component that is configured to accept tasks of this type. That component could be an interaction pattern configured to be triggered by a specific engagement task, but it could also be a component completely separate from the dialog system. A result of carrying out the action (or simply an indication of success or failure) will be returned to the system engagement action through an updated version of the task specification, regardless of the details of action execution. The specification and the current state of such an engagement task can be observed by other components apart from the submitting system engagement action and the component handling the task, a fact that could be used to realize more subtle and not completely goal-directed engagement behaviors, e.g. by controlling the eye gaze behavior of the robot.

This process makes it possible to isolate the plans of action generated by the engagement control policy a bit more from the details of the actual behavior used to carry them out, bringing back some of the abstraction and separation of concerns provided by the distinction between the system engagement action *SEA* and the system engagement behaviors *SEB* as they were proposed by Bohus and Horvitz.

4.2.4. The Engagement Process in Action

This sections shows an example of the whole engagement process being applied in a specific situation. Figure 4.3 shows the basic sequence of actions starting with an input that was received from a human interaction partner. In the first steps, the engagement manager asks the discourse operator for the current interaction and the pattern that would be matched by this input, if it was to be processed. Afterwards, the values for the engagement action and the engagement intention are calculated by asking all the registered cue and intention providers, as was described previously. This collected information is then passed on to the engagement control policy which returns a list of actions to apply. The engagement manager then iterates over all the returned actions, checking the return value in each case to see if it should continue with the next action.

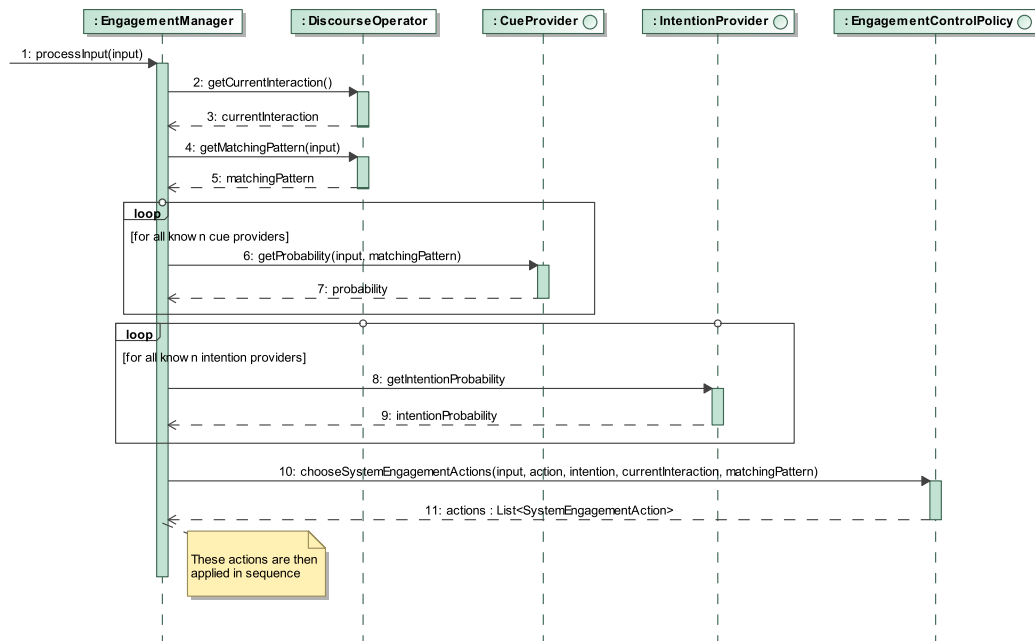


Figure 4.3.: An example of the engagement process in action.

4.3. Scenario Configuration

The following sections describe what was technically done to realize the scenario described in section 3.3. This scenario and its realization is deliberately kept simple and doesn't make use of all the features discussed up until now, but it serves as a good basic proof of concept that shows that interesting forms of interaction can be modelled with the approach taken here.

The next section introduces some of the components that were created to provide a

bridge between the dialog system and the robotic control software of the Nao platform (see section 3.1). The sections after that talks about the specific interaction patterns that were used in this scenario, the rules crafted to model the engagement control policy and what kind of information about the users was used in realizing the desired interactions.

4.3.1. Connecting Nao and PaMini

To connect the PaMini system for dialog management (see section 2.3) with the robotic platform provided by Nao, some additional software had to be developed. Figure 4.4 shows a schematic overview of the different components and the communication paths between them. The left side of the figure contains modules running on the robot and the right side shows all components running on a connected computer. This division was deemed necessary, because of the limited computing power and other constraints of the embedded computer system inside the robot.

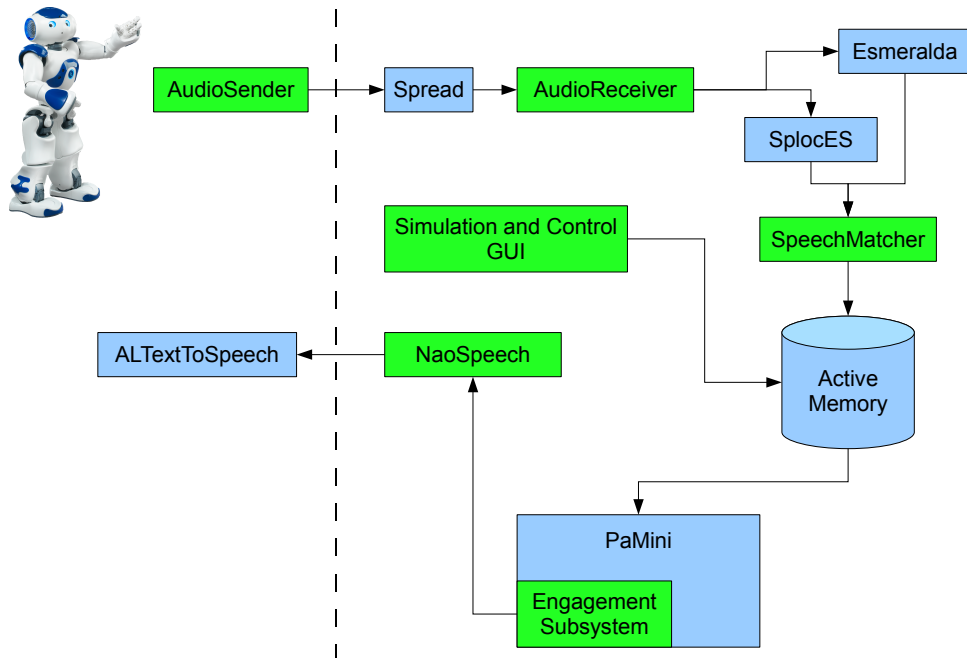


Figure 4.4.: Components of the scenario architecture. Green boxes designate the components developed for this thesis.

Perhaps the most important connecting piece of software has been the audio transport from the robot to other computers over a local area network. This was implemented as a module running on Nao’s embedded computer (labeled *AudioSender* in the figure) that accessed the audio stream from all four microphones in the robot’s head and directly

sent this stream unprocessed over the network to all components interested in receiving it. The efficient distribution over the network was realized on top of the Spread toolkit [Spr09] that provides methods for reliable distribution of data using group communication semantics. On the receiving side this audio-stream was split up (by the module named *AudioReceiver*) again into individual channels, the signal from the front microphone being fed into the speech recognizer Esmaralda (see [Fin99]) and the left and right microphone signals being directed at the audio source localization component described in section 4.3.5.

Speech production was realized using the on-board speech synthesis engine provided as part of Nao's software platform. To connect this to the dialog system, an adapter was developed (labeled *NaoSpeech* in figure 4.4) that allowed this module to be accessed over the XCF middleware-framework (see e.g. [Wre+04]). Since this interface closely followed the speech synthesis solutions that had been previously used, almost no changes on the side of the dialog system were necessary. The inputs and most other sensory information (including the information about the currently known users) reaches the dialog through the *ActiveMemory* system, which also a part of XCF.

A component that has also been developed, but has not yet been used in the system, allows the dialog system to access Nao's internal systems using the task state protocol. This could for instance be used to allow the dialog system to get information about the current state of the robot (e.g. to allow it to answer queries about this state) and in a later version maybe also as a base for controlling the robot's movements.

4.3.2. Speech Input Grammar

The speech recognizer Esmaralda was configured with a constrained grammar specific to the utterances expected in the introduction scenario. This includes among others typical greetings (e.g. "Hello, Nao!"), people introducing themselves (e.g. "I am Sebastian."), people answering questions about their relation to others (e.g. "Yes, i know her!") and people wanting to leave the interaction (e.g. by saying "Good bye!"). The appendix A includes a complete description of the grammar that was used.

Apart from improving the quality of the results of the speech recognizer (because it has to choose among fewer alternatives), structuring the utterances into this grammar also allows the dialog system to leverage the parsing work the speech recognizer has already done. One example of this is the fact that the non-terminal symbols for the different categories of utterances described above can be used in the descriptions of the different human dialog acts used by the interaction patterns.

4.3.3. Interaction Patterns

A collection of interaction patterns (see section 2.3.1) is used to specify the possible interactions between the human users and the robot in different situations. This section

provides the details of which patterns are used in which situations and how they are configured to be triggered by the right inputs or engagement actions and to produce the right dialog acts by the robot.

For opening the interaction, a pattern of the type *HumanInteractionOpening* is used. It is configured to allow a human interaction partner to greet the robot by saying a sentence of the greeting-category defined in the previous section, to which the robot will reply with an appropriate answer (e.g. “Hi, nice to meet you!”). It is important to note that the actual opening of an interaction is not controlled by the application of this pattern. Instead, this is specified through the engagement rules defined by the policy described in the following section.

To enable the robot to ask the user for his or her name, a pattern of the type *RobotSimpleInformationRequest* is used. This pattern is configured to first ask for the user’s name, wait for a reply specified by the grammatical category for introductions and acknowledge the name that was learned. As is the case for all dialog task patterns (i.e. those whose name starts with *Robot...*) it is not triggered by input received from the user, but instead by a task submitted to the dialog system. In this case, the specific task is an engagement task (with the task name “UserNameQuery”) produced by the engagement rules, but it could as well also be triggered by a completely separate component only concerned with learning the user’s names. The pattern will store the name learned in an updated version of the original task specification.

The scenario configuration also contains a pattern that is used when the robot is actively trying to engage a second user in an interaction. This pattern is also of the *RobotInformationRequest* type and is triggered by an engagement task with a name of “EngageUser”. In which exact situations this task is generated is also described in the next section. The pattern will first ask the second user if he or she already knows the first user. The answer given by the user will again be stored in the updated task specification so the result can be used by the engagement action that initially triggered this pattern.

When the robot detects that it knows the names of at least two users, it needs to introduce them to each other. This is done by a pattern of the *RobotNotification* type, which simply makes the robot produce a dialog act introducing the users (i.e. by saying “Johannes, this is Sebastian. Sebastian, this is Johannes!”). This is again triggered by the engagement rules through an engagement task, this time with the name “Introduce”, but the comment about a separate name learning component also applies here.

Asking the robot if it has already seen a specific user is possible through a pattern of the type *HumanInformationRequest*. This pattern is configured to be triggered by a human user asking if the robot has seen a user today to which the robot will give an appropriate answer (e.g. “No, i have not seen Julia today.”). The information used to decide about the robot’s answer is at the moment just the history of user names that have already been learned, but this could be expanded in the future to a more detailed temporal history of past encounters.

Finally, it is also possible to detach oneself from the interaction. If a user wants to leave and says e.g. “Good bye!”, this will trigger a pattern of the type *HumanInteractionClosing* that will make the robot reply by also saying good bye. In turn, this will also close the interaction.

4.3.4. Scenario-Specific Engagement Control Policy

The engagement control policy for the introduction scenario is implemented using a set of seven different rules. Four of these rules (labeled A1 through A4) deal with the processing of input received from users and three (labeled B1 to B3) are used in situations without any explicit input. This section contains a detailed description of these rules and the effects they are meant to achieve.

Rules for Input Processing

- A1 The first rule allows any user to engage the system when there are currently no active interactions. For this the system simply checks if the user input was classified as an *engage* action and that no current interaction exists. If this condition is met, the policy will return a list of two system engagement actions, the first of which will open a new interaction with the user who produced the input and the second action will simply process this input.
- A2 To allow anyone to disengage from the current interaction, there is a rule that tests if the input received represents a *disengage* action and that the user is indeed engaged in the current interaction. If both of that is the case, it returns a list of two engagement actions. One of them will disengage the user from the interaction and the other one will also process his or her input, since it could still match a pattern and produce a response by the robot (as is the case for the pattern used for saying “Good bye”).
- A3 The third rule for input processing is responsible simply for processing inputs of already engaged users. To achieve this, it makes sure that the input represents a *maintain* action (i.e. not a *disengage* action) and that the user is currently engaged in the interaction. In that case the policy will just return the one action necessary for processing the input.
- A4 The last rule is used to allow anyone to open a new interaction under special circumstances, which is necessary to allow the interruptions described in section 3.3. To realize this, the policy checks if the input was really meant as an *engage* action and if it would match an interaction pattern being part of a special set of priority patterns that can be specified when the policy is created. If this situation, the policy will again return the actions necessary for opening a new interaction with the user

and processing the input. In the introduction scenario, only the pattern that allows the user to ask the robot if it has already seen a specific user is classified as being such a priority pattern.

Rules for Situations Without Input

- B1 The first rule in this category is used to actively try to engage another user in an already ongoing interaction. To achieve this, a combination of different conditions has to be met. First of all, this rule only becomes active if there is already an open interaction and there are some known users. It then checks if any of these users is currently not engaged in the interaction, but has the intention to engage the system. If it finds such a user, the policy returns a list of actions meant to engage him or her. The first action will open a new, parallel interaction with that user (thereby suspending the currently active one for a while). The second action will submit the “EngageUser” engagement task, which triggers the respective pattern described in one of the previous sections. If this task returns a result indicating success, the third action will close the extra interaction and engage the user in the first interaction. As was already mentioned, the second action could also trigger some other process meant to engage the user. That this is accomplished by an interaction pattern configured for this is only one possibility.
- B2 The next rule is used to find out the name of a user. The policy contains a collection of known users names. This rule goes through all engaged users looking for one whose name is not known. If it finds such a user, it returns a list of two actions. The first will submit an engagement task named “UserNameQuery” and wait for its result. If this succeeds, the second action will use the result from the previous action and store it as the user’s name. This last action is the first instance of a system engagement action that is specific to this scenario and not part of the engagement subsystem’s standard set of actions.
- B3 The last rule of this type is used to have the robot introduce two users as soon as it knows both of their names. The rule first looks for an open interaction with at least two participants and then checks if it already knows the names of those participants. If that is the case and they haven’t already been introduced, the policy returns a list of two actions. The first action submits an engagement task named “Introduce” (that is in this case again handled by an interaction pattern configured for it) and the second action marks both participants as having been introduced so this rule will not trigger again and again for the same two users. This action updating the introduction state is the second system engagement action developed specifically for this scenario.

4.3.5. Interfaces with the Real World

This section describes the actual interfaces that were used in connecting the dialog system to the outside world. This includes getting hypotheses about the known persons and about their actions and intentions. As a basic proof-of-concept scenario, not much effort has been invested in developing or finding accurate mechanism or algorithms for providing sensoric input. This only serves to show that even with this limited data, interesting interactions can be realized and provides a basis for future work more focused on the robot's perception.

Person Hypotheses

The implementation of the *UserManager* interface that is used in this scenario gathers all information about the known users from an external component via the ActiveMemory system provided by XCF [Wre+04]. This external component could of course base these hypotheses on actual sensoric data, but for now this process is only simulated, since this sensoric procession has not been the central focus of this thesis. Figure 4.5 shows a screenshot of the graphical user interface of the simulation component, allowing an experimenter to directly manipulate the information about which users are known to the dialog system and about their state. The blue circle in the screenshot is a representation of the robot Nao and the red and green circles are hypotheses about known persons in different states.

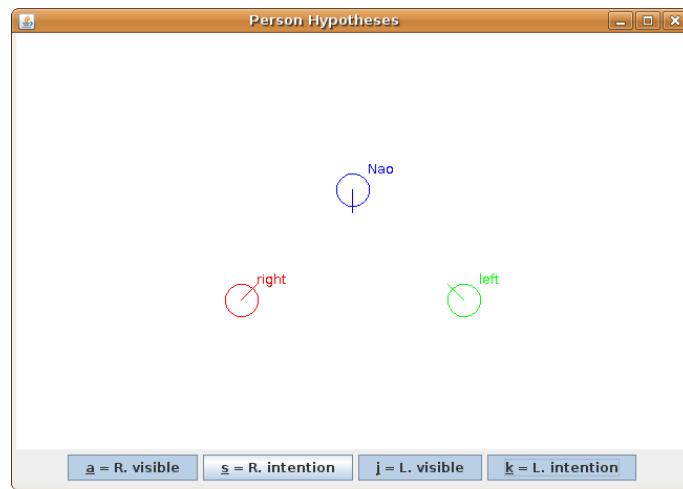


Figure 4.5.: The graphical simulation user interface.

Using Speech Input as Direct Cues

For gathering cues about engagement and disengagement actions by the users, three simple information providers based on the recognized speech that was received are used in

this scenario. The first two simply scan the received input for specific keywords (e.g. “hello” or “bye”) and take any input matching those keywords to be an explicit cue for engagement or disengagement. They are thus examples of a simple keyword-spotting procedure. The third provider, used only for engagement cues, is based on the interaction pattern configuration and the same list of priority patterns also used in realizing one rule in the engagement control policy. If an input matches a pattern from this list, that input is also believed to be an explicit cue for engagement.

A special problem arises from the fact that some decisions of the engagement process are based on the information which user generated a particular input. The speech recognition system itself does not provide any information in regard to speaker identification and while it could certainly be modified and trained to detect differences in the voices of each speaker, this is beyond the scope of this thesis. To still be able to match the input to the person hypotheses, a system was developed based on the existing sound source localization component named SplocES that has already been used in previous human robot interaction setups (see e.g. [Lan+03]).

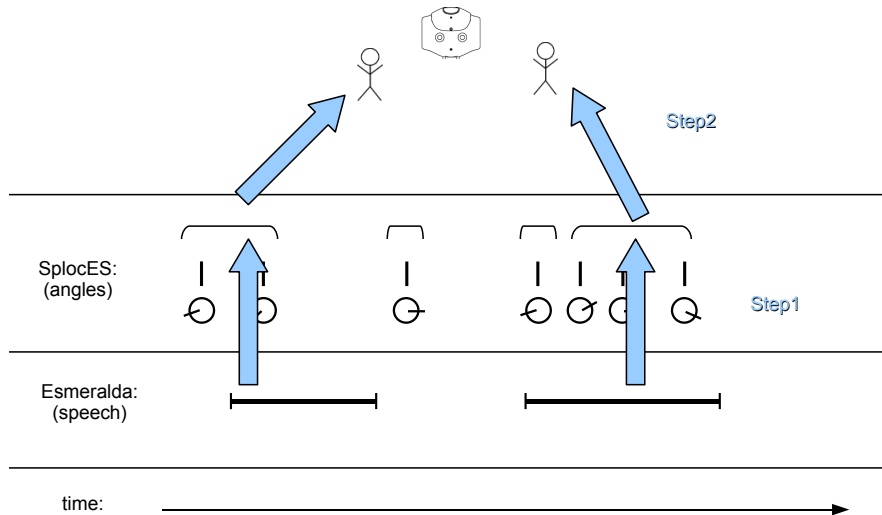


Figure 4.6.: Matching speech recognition results to individual speakers.

This component provides the directions of sound sources every time it detects sounds in speech-like frequency ranges in the input signal. Figure 4.6 shows an illustration of the current approach of mapping these detections to the speech recognition results. As an initial step to facilitate this matching, the individual detected angles are combined into “windows” of speech coming from roughly the same direction at points in time not too far apart. The first real matching step then involves finding the window that has the greatest overlap in time with the timespan of a speech recognition result. The second step of matching then takes a mean angle of this window and tries to find the person

hypothesis being closest to this direction. The results of this initial approach are still of a mixed quality and there needs to be more work done to accurately match each received input to the right person.

Detecting Engagement Intentions

In the current version of the scenario, the engagement intentions are completely simulated using the user interface described previously. They are set by the experimenter in that interface and transmitted to the dialog system through the memory based user manager, which also serves as an intention provider. The screenshot in figure 4.5 shows the person hypotheses colorized by the value of their engagement intention.

One idea for building a real measure of the engagement intention would be using a face detector (e.g. the one also described in [Lan+03] or a similar approach) that provides information about the angle of the detected face in relation to the robot. This could be used to measure when and for how long the user directly looked in the direction of the robot, i.e. how long the robot has been the user's visual focus of attention. This measure could then be interpreted as an indication of the user's engagement intention, potentially in combination with other measurements.

4.4. Analyzing Interactions

This chapter describes in some detail a test run of the developed system in the scenario described in general in chapter 3 and in its technical details in the previous sections of this chapter. This was done to collect information on the operation of the system in interaction and to identify and analyze areas that performed well and those that still need optimization.

4.4.1. Data Collection

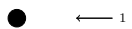

Most of what is shown in the following sections is based on logfiles produced during the system's normal operation. The information that is recorded includes the results of the speech recognizer, the results of matching this input to the known users, changes in the state of the dialog (including the open interactions and their engaged users and active patterns) and the state of the currently known users (including their engagement intention). The engagement control policy also recorded which of its rules triggered at which point in the interaction.

The interaction log as it is shown here is a condensed version and shows only the major events relevant to the interaction. When a rule is triggered inside the engagement control policy, only the short designation given in section 4.3.4 is listed. The details of the triggered rule can be found in that section. The times shown for the individual events are in the form *minutes:seconds,milliseconds*. The appendix B contains an unabridged version of the recorded events.

The current state of the dialog and the known users uses the same type of small diagram described in section 3.3 and is only repeated at those points in time where this state has changed.

Simulated Interaction

This test run was done in a completely simulated environment, i.e. no real speech recognition or other sensoric processes were used. Instead, the speech input was provided through the simulation environment described in section 4.3.5, which also “matched” the input perfectly to the users designated by the experimenter. This was done to verify the basic operation of the whole system and also to have a baseline to compare the results achieved in more complex situations to.

<i>Time</i>	<i>Action</i>	<i>State</i>
4:13,412	Speech input received: 'hello nao'	
4:13,521	Rule A1 triggered by speech input 'hello nao'.	
4:14,126	Speech output: 'Hi, nice to meet you!'	


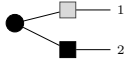
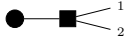
<i>Time</i>	<i>Action</i>	<i>State</i>
...	...	
4:14,970	Rule B2 triggered.	
4:15,774	Speech output: 'It seems like i do not know your name. Who are you?'	
4:30,001	Speech input received: 'i am david'	
4:30,032	Rule A3 triggered by speech input 'i am david'.	
4:30,525	Speech output: 'OK, David, i'll try to remember your name.'	
...	...	
4:51,564	The dialog currently knows about 2 users. (User 0 has intention=1.0, User 1 has intention=1.0)	
4:52,113	Rule B1 triggered.	
4:53,292	Speech output: 'Hello, do you already know each other?'	
5:13,880	Speech input received: 'no we do not'	
5:13,887	Rule A3 triggered by speech input 'no we do not'.	
...	...	
5:14,025	Rule B2 triggered.	
5:15,342	Speech output: 'It seems like i do not know your name. Who are you?'	
5:35,930	Speech input received: 'my name is sebastian'	
5:35,938	Rule A3 triggered by speech input 'my name is sebastian'.	
5:36,468	Speech output: 'OK, Sebastian, i'll try to remember your name.'	
5:37,002	Rule B3 triggered.	
5:37,974	Speech output: 'Then lets introduce you two. David this is Sebastian, Sebastian this is David.'	
...	...	

Table 4.1.: Abridged interaction test run.

Recording Real Interactions

While recording interactions with real speech input and more complex sensoric processing was originally planned, problems with the speech matching system described in section 4.3.5 and other constraints have prevented their inclusion in this thesis. Since these sensoric processes weren't part of the core focus of this thesis, optimizing them has been postponed until after the completion of this thesis. Nevertheless, using this system in more and more real interactional situations is still the goal for the future, since only real interactions can provide a benchmark for how well this approach models the details of engagement in multi-party interactions.

4.4.2. Results

The interaction in the simulated example shows that the core of the engagement subsystem works as expected and that no serious problems seem to hinder the fundamental way the engagement control policy interacts with the rest of the dialog system to achieve the management of the engagement state. Comparing this to the example dialog that was defined as a goal in section 3.3, we can see that the achieved dialog and the changes in the engagement state of the participants differ only slightly from the initial concept. The shift of initiative when the user is telling the robot his or her name stems from the fact that the opening of the interaction and the question for the name were split up to be modeled using two different interaction patterns. While this is more of a technical reason, it also serves as a nice example that giving the robot the initiative in some situations can lead to very natural dialogs.

5. Conclusion and Outlook

The work done for this thesis provides a promising approach for modeling multi-party interactions and integrating different sources of information about engagement cues and signals. While benchmarks in more realistic settings are still needed, the general architecture can be used and extended to model a variety of multi-party interactions. This chapter takes a critical look at what was achieved and what could be done in the future to enhance or expand the work done for this thesis.

One direction that could provide further insights into how a robot should behave when interacting with a group of people is having the robot move around in this group while trying to achieve its goals of interaction. This will provide new questions, e.g. how a robot should position itself when trying to address different parts of a group and how to select the correct interaction partner from a group in different situations. While the scenario used for this thesis did not include a moving robot (see section 3.3 for a discussion of this), no general assumptions in the modeling decisions made should preclude this approach from being extended in that direction. The integration of systems dedicated to generating hypotheses about interaction partners for a mobile robot (e.g. similar approaches to the work done in [Han+08]) will become an important issue when moving in that direction.

In general a greater focus should be placed in the future on integrating more approaches extracting relevant information about the environment (including possible interaction partners) from the robot's sensors. This will not only serve to provide more data on which engagement decisions could be based, but will also become helpful when looking at higher levels of dialog enhancements, e.g. a better model of multi-party turn-taking.

Integrating this work with the new approach to memory systems developed by Wienke [Wie10] for the HUMAVIPS project is also a direct goal for future development. The integrated history mechanism provided by this memory system could e.g. be used in trying to make decisions not only based on the current state or user intentions but also based on specific temporal patterns in this information. Its layered approach could also provide for better mechanism of sensor fusion and the generation of hypotheses, e.g. about the users and their actions and intentions.

The relatively static nature of the handcrafted rules currently used to implement the system engagement policy described in section 4.3.4 may likely also become a limiting factor in modeling more complex interactions at some point in the future. Finding a more flexible and possibly probabilistic modeling for the selection of engagement actions could prove a viable direction and could ultimately even lead to more data-driven approaches,

including trying to learn the correct behavior from observing interactions between humans and between human and robotic systems.

From an architectural viewpoint, a better separation of concerns between the engagement subsystem and the rest of the dialog system is desirable to some degree, e.g. by externalizing the engagement handling to another external component. Technically however, this endeavor is complicated by the relatively tight integration of some parts of the engagement subsystem within the general dialog management, so some critical thought about the interfaces between these components would be needed for this.

Regarding the differentiation of different layers or levels of understanding presented in section 2.1, this work has been focused almost completely on the lower levels of these models. Taking another look at the higher levels, e.g. the general management of dialog goals and the interaction patterns themselves, and thinking about how their current realizations could benefit from the improvements now present in the lower layers could also lead to further enhancements in this approach to dialog modeling.

The work done for this thesis succeeds in integrating the concept of engagement into an existing, usable system for modeling human-robot interactions, expanding its capabilities to model complex interactional situations. For this, it borrows perhaps most heavily from the work of Bohus and Horvitz, adapting their concepts to the needs and requirements of a new environment. In general I hope that the approach taken in this thesis will provide a fruitful foundation for future work on multi-party human-robot interactions and I look forward to applying it to situations beyond the proof-of-concept scenario realized for this thesis. I am also especially looking forward to the challenges and insights the modeling of different situations and the integration of various perceptual approaches in the HUMAVIPS project will provide when this system is used in trying to further that project's goals of fluid robot-to-group interaction.

A. Speech Recognition Grammar

This appendix lists the grammar the speech recognizer was configured to use in the scenario described in the chapters 3 and 4. This listing is derived from the original configuration of the speech recognizer by just slightly changing it to a more readable EBNF-like form. Words in quotation marks are terminal symbols (recognized words, in this case) and words without quotations denote non-terminal categories. The vertical bar represents alternatives in the grammar and the comma denotes concatenation of multiple symbols.

```
S = Greeting | Introduction | PersonQuery | Answer | GoodBye ;
```

```
Greeting = RobotName, "hello" | "hello", RobotName  
          | "hi", RobotName | "hello" ;
```

```
Introduction = RobotName, "i", "am", HumanName  
              | "i", "am", HumanName  
              | RobotName, "my", "name", "is", HumanName  
              | "my", "name", "is", HumanName ;
```

```
Answer = RobotName, PositiveAnswer | PositiveAnswer  
         | RobotName, NegativeAnswer | NegativeAnswer ;
```

```
GoodBye = "good", "bye", RobotName | "good", "bye" ;
```

```
PositiveAnswer = "yes", "i", "do" | "yes", "we", "do"  
                | "yes", "i", "know", "him" | "yes", "i", "know", "her" ;
```

```
NegativeAnswer = "no", "i", "don't" | "no", "we", "don't"  
                | "no", "we", "do", "not"  
                | "no", "i", "don't", "know", "her"  
                | "no", "i", "don't", "know", "him" ;
```

```
PersonQuery = RobotName, PersonQuestion | PersonQuestion ;
```

```
PersonQuestion = "have", "you", "seen", HumanName, "today"  
                | "did", "you", "see", HumanName, "today" ;
```

```
HumanName = "David" | "Johannes" | "Julia"  
           | "Sebastian" | "Britta" ;
```

```
RobotName = "nao" | "robot" | "piper" ;
```

B. Recorded Interaction with the Engagement Subsystem

This appendix provides the complete and detailed log of the recorded interaction described in section 4.4. Table B.1 does not include the graphical depiction of the current dialog state (i.e. the open interactions and the engaged users), but instead the changes in this state are part of the log itself. The current state is repeatedly included in the table everytime it has changed in the interaction. The rules that are triggered inside the engagement control policy use the numbering introduced in section 4.3.4. The times shown are in the form *minutes:seconds,milliseconds*.

<i>Time</i>	<i>Action</i>
3:27,325	The dialog currently has 0 open interactions.
3:55,701	The dialog currently knows about 1 users. (User 0 has intention=0.0)
4:01,557	The dialog currently knows about 1 users. (User 0 has intention=1.0)
4:13,412	Speech input received: 'hello nao'
4:13,413	Speech input matched to user 0: 'hello nao'
4:13,521	Rule A1 triggered by speech input 'hello nao'.
4:13,553	The dialog currently has 1 open interactions.
4:13,553	Interaction 1 is active and has 0 engaged users and 0 active patterns.
4:13,593	The dialog currently has 1 open interactions.
4:13,593	Interaction 1 is active and has 1 engaged users and 0 active patterns.
4:13,594	User 0 is engaged in interaction 1
4:14,126	Speech output: 'Hi, nice to meet you!'
4:14,147	The dialog currently has 1 open interactions.
4:14,147	Interaction 1 is active and has 1 engaged users and 0 active patterns.
4:14,147	User 0 is engaged in interaction 1
4:14,970	Rule B2 triggered.
4:15,774	Speech output: 'It seems like i do not know your name. Who are you?'
4:15,792	The dialog currently has 1 open interactions.
4:15,792	Interaction 1 is active and has 1 engaged users and 1 active patterns.
4:15,792	User 0 is engaged in interaction 1

<i>Time</i>	<i>Action</i>
4:15,792	Pattern 'NameQuery' of type 'RobotSimpleInformationRequest' is active in interaction 1
4:30,001	Speech input received: 'i am david'
4:30,002	Speech input matched to user 0: 'i am david'
4:30,032	Rule A3 triggered by speech input 'i am david'.
4:30,525	Speech output: 'OK, David, i'll try to remember your name.'
4:30,551	The dialog currently has 1 open interactions.
4:30,552	Interaction 1 is active and has 1 engaged users and 0 active patterns.
4:30,552	User 0 is engaged in interaction 1
4:42,664	The dialog currently knows about 2 users. (User 0 has intention=1.0, User 1 has intention=0.0)
4:51,564	The dialog currently knows about 2 users. (User 0 has intention=1.0, User 1 has intention=1.0)
4:52,113	Rule B1 triggered.
4:52,138	The dialog currently has 2 open interactions.
4:52,139	Interaction 2 is active and has 0 engaged users and 0 active patterns.
4:52,139	Interaction 1 is suspended and has 1 engaged users and 0 active patterns.
4:52,139	User 0 is engaged in interaction 1
4:52,157	The dialog currently has 2 open interactions.
4:52,157	Interaction 2 is active and has 1 engaged users and 0 active patterns.
4:52,157	User 1 is engaged in interaction 2
4:52,157	Interaction 1 is suspended and has 1 engaged users and 0 active patterns.
4:52,158	User 0 is engaged in interaction 1
4:53,292	Speech output: 'Hello, do you already know each other?'
4:53,319	The dialog currently has 2 open interactions.
4:53,320	Interaction 2 is active and has 1 engaged users and 1 active patterns.
4:53,320	User 1 is engaged in interaction 2
4:53,320	Pattern 'EngageUser' of type 'RobotSimpleInformationRequest' is active in interaction 2
4:53,320	Interaction 1 is suspended and has 1 engaged users and 0 active patterns.
4:53,320	User 0 is engaged in interaction 1
5:13,880	Speech input received: 'no we do not'
5:13,881	Speech input matched to user 1: 'no we do not'
5:13,887	Rule A3 triggered by speech input 'no we do not'.
5:14,009	The dialog currently has 1 open interactions.
5:14,009	Interaction 1 is active and has 1 engaged users and 0 active patterns.
5:14,009	User 0 is engaged in interaction 1

<i>Time</i>	<i>Action</i>
5:14,025	Rule B2 triggered.
5:14,041	The dialog currently has 1 open interactions.
5:14,041	Interaction 1 is active and has 2 engaged users and 0 active patterns.
5:14,041	User 0 is engaged in interaction 1
5:14,042	User 1 is engaged in interaction 1
5:14,661	Speech output: 'OK'
5:14,693	The dialog currently has 1 open interactions.
5:14,693	Interaction 1 is active and has 2 engaged users and 0 active patterns.
5:14,693	User 0 is engaged in interaction 1
5:14,693	User 1 is engaged in interaction 1
5:15,342	Speech output: 'It seems like i do not know your name. Who are you?'
5:15,422	The dialog currently has 1 open interactions.
5:15,422	Interaction 1 is active and has 2 engaged users and 1 active patterns.
5:15,422	User 0 is engaged in interaction 1
5:15,422	User 1 is engaged in interaction 1
5:15,422	Pattern 'NameQuery' of type 'RobotSimpleInformationRequest' is active in interaction 1
5:35,930	Speech input received: 'my name is sebastian'
5:35,931	Speech input matched to user 1: 'my name is sebastian'
5:35,938	Rule A3 triggered by speech input 'my name is sebastian'.
5:36,468	Speech output: 'OK, Sebastian, i'll try to remember your name.'
5:36,489	The dialog currently has 1 open interactions.
5:36,489	Interaction 1 is active and has 2 engaged users and 0 active patterns.
5:36,489	User 0 is engaged in interaction 1
5:36,489	User 1 is engaged in interaction 1
5:37,002	Rule B3 triggered.
5:37,974	Speech output: 'Then lets introduce you two. David this is Sebastian, Sebastian this is David.'
5:38,013	The dialog currently has 1 open interactions.
5:38,013	Interaction 1 is active and has 2 engaged users and 0 active patterns.
5:38,013	User 0 is engaged in interaction 1
5:38,014	User 1 is engaged in interaction 1
5:56,875	The dialog currently knows about 2 users. (User 0 has intention=1.0, User 1 has intention=0.0)
6:12,422	Speech input received: 'good bye'
6:12,423	Speech input matched to user 0: 'good bye'
6:12,508	Rule A2 triggered by speech input 'good bye'.

<i>Time</i>	<i>Action</i>
6:12,543	The dialog currently has 1 open interactions.
6:12,543	Interaction 1 is active and has 1 engaged users and 0 active patterns.
6:12,543	User 1 is engaged in interaction 1
6:12,654	The dialog currently has 1 open interactions.
6:12,654	Interaction 1 is active and has 0 engaged users and 0 active patterns.
6:12,682	The dialog currently has 0 open interactions.
6:13,195	Speech output: 'Good bye, have a nice day!'
6:13,233	The dialog currently has 0 open interactions.
6:20,386	The dialog currently knows about 2 users. (User 0 has intention=0.0, User 1 has intention=0.0)
6:24,836	The dialog currently knows about 1 users. (User 0 has intention=0.0)
6:27,939	The dialog currently knows about 0 users.

Table B.1.: Simulated interaction test run.

Bibliography

- [Ald10] Aldebaran Robotics. *Nao Academics DataSheet*. 2010. URL: <http://www.aldebaran-robotics.com/en/node/1166> (visited on 10/15/2010).
- [Beu+08] Niklas Beuter et al. “Where is this? - gesture based multimodal interaction with an anthropomorphic robot”. In: *Humanoids 2008. International Conference on Humanoid Robots*. Daejeon, ROK: IEEE-RAS, 2008, pp. 585–591.
- [BH09a] Dan Bohus and Eric Horvitz. “Models for Multiparty Engagement in Open-World Dialog”. In: *Proceedings of the SIGDIAL 2009 Conference*. London, UK: Association for Computational Linguistics, Sept. 2009, pp. 225–234.
- [BH09b] Dan Bohus and Eric Horvitz. “Open-World Dialog: Challenges, Directions, and Prototype”. In: *Proceedings of IJCAI2009 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*. Pasadena, 2009.
- [BH95] Susan E. Brennan and Eric A. Hulteen. “Interaction and feedback in a spoken language system: a theoretical framework”. In: *Knowledge-Based Systems* 8.2-3 (1995), pp. 143–151.
- [CS87] Herbert H. Clark and Edward F. Schaefer. “Collaborating on contributions to conversations”. In: *Language and cognitive processes* 2.1 (1987), pp. 19–41.
- [CS89] Herbert H. Clark and Edward F. Schaefer. “Contributing to discourse”. In: *Cognitive Science* 13.2 (1989), pp. 259–294.
- [DFK10] DFKI–German Research Center for Artificial Intelligence. *The MARY Text-to-Speech System*. 2010. URL: <http://mary.dfki.de/> (visited on 10/15/2010).
- [Fin99] G. A. Fink. “Developing HMM-based Recognizers with ESMERALDA”. In: *Lecture Notes in Artificial Intelligence*. Ed. by Václav Matousek et al. Vol. 1692. Berlin and Heidelberg: Springer, 1999, pp. 229–234.
- [Han+08] Marc Hanheide et al. “Who am I talking with? A face memory for social robots”. In: *2008 IEEE International Conference on Robotics and Automation*. Pasadena: IEEE, May 2008, pp. 3660–3665.
- [Har87] D. Harel. “Statecharts: A visual formalism for complex systems”. In: *Science of computer programming* 8.3 (1987), pp. 231–274.
- [HUM09] HUMAVIPS Project. *Grant Agreement Annex 1 - Description of Work*. Sept. 2009.

- [KH09] Iwan de Kok and Dirk Heylen. “Multimodal End-of-Turn Prediction in Multi-Party Meetings”. In: *ICMI-MLMI '09 Proceedings*. Cambridge, MA, USA: ACM Press, 2009, pp. 91–98.
- [Lan+03] S. Lang et al. “Providing the Basis for Human-Robot-Interaction: A Multi-Modal Attention System for a Mobile Robot”. In: *Proc. Int. Conf. on Multimodal Interfaces*. ACM. Vancouver, Canada: ACM, Nov. 2003, pp. 28–35.
- [Lem+01] Oliver Lemon et al. “The WITAS multi-modal dialogue system I”. In: *Seventh European Conference on Speech Communication and Technology*. Aalborg, Denmark, 2001, pp. 4–7.
- [Lüt+09] Ingo Lütkebohle et al. “The Curious Robot - Structuring Interactive Robot Learning”. In: *International Conference on Robotics and Automation*. IEEE. Kobe, Japan: IEEE, May 2009.
- [LW07] Shuyin Li and Britta Wrede. “Why and how to model multi-modal interaction for a mobile robot companion”. In: *Proc AAAI Spring Symposium on Interaction Challenges for Intelligent Assistants*. Stanford, 2007.
- [Pel+09] Julia Peltason et al. “Mixed-initiative in human augmented mapping”. In: *2009 IEEE International Conference on Robotics and Automation*. Kobe, Japan: IEEE, May 2009, pp. 2146–2153.
- [Pet+05] Christopher Peters et al. “A Model of Attention and Interest Using Gaze Behavior”. In: *Intelligent Virtual Agents*. Ed. by Themis Panayiotopoulos et al. Vol. 3661. Lecture Notes in Computer Science. Berlin and Heidelberg: Springer, 2005, pp. 229–240.
- [PH00] Tim Paek and Eric Horvitz. “Conversation as Action Under Uncertainty”. In: *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 455–464.
- [PW10] Julia Peltason and Britta Wrede. “Modeling human-robot interaction based on generic interaction patterns”. In: *AAAI Fall Symposium: Dialog with Robots*. Arlington, VA, USA, 2010.
- [Sid+04] Candace L. Sidner et al. “Where to look: a study of human-robot engagement”. In: *Proceedings of the 9th International Conference on Intelligent User Interfaces*. Funchal, Madeira, Portugal: ACM Press, 2004, pp. 78–84.
- [Spr09] Spread Concepts LLC. *The Spread Toolkit*. 2009. URL: <http://www.spread.org/> (visited on 10/14/2010).
- [TR02] David Traum and Jeff Rickel. “Embodied agents for multi-party dialogue in immersive virtual worlds”. In: *Proceedings of the first international joint conference on Autonomous agents and multiagent systems part 2*. Bologna, Italy: ACM, 2002, pp. 27–33.

- [Wie10] Johannes Wienke. “A Spatiotemporal Working Memory for Humanoid Robots”. Master’s Thesis. Bielefeld University, 2010.
- [Wre+04] Sebastian Wrede et al. “An XML Based Framework for Cognitive Vision Architectures”. In: *Proc. Int. Conf. on Pattern Recognition*. Cambridge, UK, 2004, pp. 757–760.

List of Figures

2.1. Grounding strategies in Quartet.	7
2.2. Competencies for open-world dialog.	9
2.3. Engagement state transitions.	11
2.4. Examples of interaction patterns.	16
2.5. The life-cycle of tasks.	17
3.1. The robot Nao from Aldebaran Robotics.	20
3.2. Two persons interacting with Nao.	21
4.1. The dialog state.	26
4.2. Architecture of the engagement system.	27
4.3. The engagement process.	32
4.4. Components of the scenario architecture.	33
4.5. The graphical simulation user interface.	38
4.6. Matching speech and speakers.	39

Erklärung

Hiermit versichere ich, die vorliegende Masterarbeit selbstständig angefertigt und keine weiteren als die angegebenen Hilfsmittel und Quellen verwendet zu haben.

Bielefeld, im November 2010

David Klotz