Modeling Human-Robot-Interaction Based on Generic Interaction Patterns

Julia Peltason

Julia Peltason
Angewandte Informatik
Technische Fakultät
Universität Bielefeld
jpeltaso@techfak.uni-bielefeld.de

Abdruck der genehmigten Dissertation zur Erlangung
des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
Der Technischen Fakultät der Universität Bielefeld
am 13.03.2012 vorgelegt von Julia Peltason
am 12.06.2013 verteidigt und genehmigt

Gutachter
Prof. Dr. Britta Wrede
Prof. Dr. David Schlangen

Prüfungsausschuss
Prof. Dr. Britta Wrede
Prof. Dr. David Schlangen
Prof. Dr. Philipp Cimiano
Dr. Robert Haschke

Gedruckt auf alterungsbeständigem Papier nach ISO 9706

Acknowledgement

This thesis would not have been possible without the help and support of many people. First I would like to thank my supervisor Britta Wrede. In spite of her tight schedule, she always found time for me whenever I needed advice. I also thank my thesis committee for their time in reviewing and evaluating my thesis. I am grateful to Gerhard Sagerer and Franz Kummert for giving me the opportunity to graduate at the Applied Informatics group and to participate in various interesting projects.

I would like to thank everyone I have collaborated with. Thanks to my colleagues from the Home-Tour team (in particular to Marc Hanheide) and the Curious Robot team (in particular to Ingo Lütkebohle and Robert Haschke) for fruitful collaboration and lots of good ideas (sometimes more than we could put into practice). Thanks for collaboration also to my (former) student workers Anja Phillipsen, Annika Rothert, Birte Carlmeyer, Maikel Linke and Marian Pohling. I enjoyed working with you! I would also like to thank the users of my software, especially Frederic Siepmann, for providing feedback and making helpful suggestions for improvement. And thanks a lot to Nina Riether for helping with the statistics.

Finally, I want to thank my colleagues and friends from the Applied Informatics group: Agnes Swadzba, Angelika Dierker, Holger Dierker, Ingmar Berger, Lars Schillingmann, Manja Lohse, and all the others who made my time there such an enjoyable time. Special thanks is due to Ingo Lütkebohle who played such an important role for me, not just for this thesis, but also in my real life.

Contents

Part I

A Pattern-Based Approach to Human-Robot-Interaction

1 Introduction

Interaction provides many benefits for the control of robot systems, not only in domains in which it is the primary function of the robot to interact socially with people, but also in domains in which the robot requires peer-to-peer interaction skills for solving a specific task in collaboration with the human [Fon03]. Speech is the most natural access to the robot's capabilities, and – in contrast to traditional teleoperation – Human-Robot-Interaction is two-way [FTB01]: The user can specify tasks, on the execution of which the robot can provide feedback, both participants can convey information and ask questions – and all that while being busy with another task [CO95].

Nevertheless, dialog management is slightly neglected in robotics. Even the Human-Robot-Interaction (HRI) community either focuses on single aspects of interaction, such as the role of gaze or the robot's external appearance, or uses very simple approaches for dialog management that exhibit a number of shortcomings.
As an example, at the ACM/IEEE HRI 2011 conference, which is the one of the most recognized and most selective conferences in the field, only four out of 13 contributions presented systems that were autonomous. Of those four, only one system involved an interaction with the robot at all [SWWB11], but it did not make use of a dedicated dialog system. At the AAAI Dialog with Robots 2010 Symposium, which was explicitly dedicated to dialog modeling on robots, 15 out of 21 interactive robots were autonomous. From those, only seven relied on approaches that can be characterized as generalizable [COB10, IST$^+$10, NIN$^+$10, PW10a, RS10, Ros10, SIKN10], while the others relied on very specific or scripted solutions.
At the same time, existing approaches for dialog management in traditional domains, such as information retrieval, do not fully take the special nature of Human-Robot-Interaction into account. This position (which will be further detailed throughout this thesis) may seem surprising, given the fact that approaches exist that consider phenomena like turn-taking [RE09], incrementality [SS09], grounding [Tra94] or problem solving [FA98].

Compared to modeling those subtleties, modeling interaction through which a robot can be taught or instructed appears almost trivial. But let us consider a simple fetch-and-carry task in which a mobile robot companion is instructed to fetch an object from another room, e.g. from the kitchen. Besides the skills directly related to the task, such as face recognition, object recognition and navigation skills, this scenario requires interaction skills that go beyond mere understanding of the human's instruction. For example, if the object the robot is supposed to find is not in its usual place, or if an obstacle is blocking the robot's way, it needs to request clarification or assistance from the human. While the robot is on its way in the kitchen, the human might ask for tomorrow's weather forecast, so the robot needs to perform two tasks at a time. The human might even change his or her mind and instruct the robot not to bring coffee, but rather a glass of water, so the robot needs to revise its task while it is being executed.

Further, in order to execute the given task, the robot needs to have a model of the spatial layout of its environment and of the relevant objects within, and a model of where the objects are typically located. As these models depend on the individual environment, they can not be preprogrammed but need to be learnt – preferably through natural interaction. This sets further requirements for the robot's interaction capabilities, including requesting and revising information in order to obtain models that are correct, consistent, and as complete as possible.

But a robot that co-exists with humans in their daily environment needs to be not only useful, but to a certain extent also enjoyable, or at least non-intrusive, i.e. they need to be able to interact with people in a socially acceptable manner [Dau07]. This means that a robot needs to possess also some basic social skills and an awareness of the social situation. For example, in order not to disturb humans when they are interacting with each other, it might make use of models of engagement, multi-party dialog or proxemics, all of which are closely related to dialog management in general.

Altogether, this example illustrates well that it is not the interaction as such which makes the fetch-and-carry task quite challenging, its complexity lies rather in combining the diverse functionalities, and to integrate them with the robot's interaction capabilities. This is where the main difference between HRI and traditional domains lies, and the reason why existing approaches to dialog modeling are not always well suited for modeling HRI.

This difference can be viewed from two perspectives. From an interaction perspective, it is the difference between *situated* and *non-situated* interaction. The situatedness of interaction entails that the robot system is operating in changing and unpredictable real-world environments, which involves issues such as autonomy, multimodality, multitasking or learning. These characteristics of HRI and their technical implications for a dialog system will be discussed later in this thesis.

From a system's perspective, the difference lies in being an *integrated* rather than a *stand-alone*[1] dialog system. This means that, on the one hand, the dialog system has to coordinate with other components of the system, e.g. for action or perception, and that it needs to provide concepts how coordination can be done in a systematic manner. On the other hand, this implies a number of non-functional requirements, most importantly usability and easy reconfigurability, in order to enable new interaction sce-

1 Of course, traditional dialog systems are not completely stand-alone either, but have some sort of application back-end as well. The back-end is, however, typically less autonomous than in robotics, and communication is often based on a master-slave relationship.

narios to be implemented not only by specialists in dialog modeling, but also by roboticists.

Consequently, the general goal of the present work is to develop an approach to dialog modeling that considers the specific requirements of HRI, that integrates well into complex robotic architectures, and that at the same time is easy to use and easy to understand, also for non-experts.

The suggested approach was not born out of theory, but developed in an iterative process. Its underlying concepts and the resulting dialog framework evolved from experiences with previously used dialog managers (and their shortcomings), and from experiences gained through the implementation of several HRI scenarios. But the implemented scenarios served not only as a means to gain insights into the domain, but – at a later stage – also as a test bed and proof-of-concept for the developed approach. Scenario development followed an iterative approach as well: several aspects (ranging from user behavior over dialog strategies to speech understanding performance) were evaluated in different versions of the scenarios and the findings were incorporated in the following iterations. Thus, the development of the approach follows a complex implementation-evaluation cycle, which includes both dialog modeling and scenario implementation, both of which in turn mutually influence each other.
The approach aims to consider different perspectives. In particular, the proposed dialog model considers both the dialog designer's and the roboticist's perspective, leading to an integrated view of dialog and task management. During the development of the approach, it was sometimes helpful to look at dialog modeling from a software engineering perspective, which is reflected in the use of principles such as identifying patterns, generalizing from concrete examples, or encapsulating what varies. Finally, the evaluations address both the developer's view, i.e. the usability of the framework, and the end user's view, i.e. the usability of the resulting systems.
Altogether, the contributions of the present work concern three areas: dialog modeling, development of HRI scenarios, and the design process that combines the two. They are listed in detail in section 9.

The present thesis consists of two parts. The first part (chapters 2 to 4) is dedicated to the proposed approach itself, while the second part (chapters 5 to 8) addresses the implemented scenarios. In detail, it is organized as follows:

Part 1: A Pattern-Based Approach to Human-Robot-Interaction

Chapter 2 discusses the foundational work on dialog modeling, both in general and in robotics, and on evaluation of interactive systems. Based on this discussion, the adopted approaches for dialog modeling and evaluation are outlined.

Chapter 3 motivates the benefits of a well-defined component interface and introduces the

Task State Protocol, which establishes the external interface of the presented dialog framework. Its benefits for implementing HRI scenarios are illustrated based on several use cases.

Chapter 3.4.4 presents the concept of Interaction Patterns, which serve both as internal dialog model and as developer's application programming interface (API). Details about the dialog management process and aspects such as global discourse management, grounding and multimodality, are provided.

Chapter 4 presents the developer-centered evaluation of the approach. Framework efficacy is addressed by discussing four case studies in which the approach is compared with existing, well-established approaches. Framework usability is investigated through a usability test.

Part 2: Applications of the Proposed Approach

Chapter 5 outlines the iterative development process of the novel approach to dialog modeling and gives an overview of all scenarios that were implemented within the context of this work, either with the suggested PaMini (Pattern-based Mixed Initiative) framework or with its predecessors.

Chapter 6 describes two preliminary scenarios that the author contributed to, namely the Home-Tour scenario and different iterations of the Curious Robot scenario. These scenarios helped to establish a comprehensive understanding of the domain, and to identify several overarching principles that influenced the development of the proposed approach. Further, this chapter describes several evaluations that address different aspects of the scenarios.

Chapter 7 gives a detailed account of the Curious Flobi scenario, which served as the main test bed for the proposed approach, and of its development process. It is the successor version of the Curious Robot scenario described in the previous chapter, and it incorporates various results obtained through the evaluation of the preliminary scenarios. This chapter also describes a large-scale user study based on the PARADISE approach that was conducted with the system.

Chapter 8 gives a brief description of all further scenarios that were developed using the PaMini framework, not by the author herself but by different developer teams. Their diversity is illustrated by analyzing the Interaction Patterns they make use of.

Chapter 9 summarizes the present work and discusses its contributions.

2 Foundational Work in Dialog Modeling and Evaluation

This chapter presents an overview of foundational work on dialog management, both in spoken dialog systems and in HRI. Research in these disciplines has long developed separately: while researchers in HRI have been focusing on nonverbal aspects of communication, the spoken dialog systems community has been focusing on non-situated systems, mostly in information-seeking domains. Approaches to dialog modeling in general will be addressed in section 2.1, while the situation in HRI will be discussed separately in section 2.2, considering also the special requirements of HRI. An issue that is an essential part of the development process of interactive systems is evaluation. Section 2.3 discusses the state-of-the art of evaluation techniques, both for spoken dialog systems and for interactive robot systems. Based on the previous discussions, the adopted approach to dialog modeling and evaluation will be set into context of the research landscape in section 2.4.

2.1 Approaches to Dialog Modeling

Concepts and methods for dialog modeling have been maturing for more than 30 years, both in research and in industry. Accordingly, a wide range of approaches exists. Section 2.1.1 will present common categorizations of approaches and suggest a fundamental distinction between descriptive and mental-state approaches. To analyze the specific strengths and weaknesses of the approaches, some representatives of each will be discussed in detail in sections 2.1.2 and 2.1.3. We will return to these examples systems at different locations of this thesis in order to discuss more specific aspects, such as the general architecture or their application to robotics.

2.1.1 Categorization of Approaches

A common categorization of approaches to dialog modeling has been made by McTear [McT04]. It distinguishes between finite state-based, frame-based and agent-based approaches. In **finite state-based** systems, the dialog structure is modeled as a graph whose nodes represent system actions, such as prompting or requesting information or performing some actions (e.g. a database lookup). The transitions specify all possible dialogs. For example, in a travel booking system, a node might represent a dialog state in which the system asks for the destination and the user gives the city name, as sketched in figure 2.1.

This would trigger a transition to the next dialog state, in which the system asks for confirmation of the destination. The major advantages of this approach are its simplicity and its intuitiveness. Also, with this approach, speech recognition can be made very robust by switching the speech recognition grammar depending on the current dialog state and the user input it permits. However, the development process is less robust, at least for more complex interactions, because a dialog graph may become unmanageable as it grows. Also, the frame-based approach is suitable for interactions that are primarily guided by the system. In principle, it would be possible to realize a mixed-initiative interaction style with it, but considering all possible user actions at any state would bloat the graph more and more. Altogether, the main disadvantage of the approach is that it is inflexible in that it does not allow for deviations from the dialog paths that are specified in the graph explicitly. On the other hand, this is what enables automated testing of all possible dialogs, which is why the finite state-based approach is widely used in commercial systems. In general, the finite state-based approach is suited for well-structured domains that rely on system initiative, such as form-filling tasks in which the system gathers information from the user in a fixed order.

Similar as finite state-based systems, **frame-based** systems are applied in classical information-seeking domains. However, they allow for a greater degree of flexibility by maintaining an explicit representation of the required information in form of a frame, together with a control algorithm that determines the next system action, based on content of the frame and the user's input. If the user gives more information than the system asked for, the control algorithm will fill the frame with additional information and proceed with the appropriate state. Thus, the transitions between the dialog states are not hard-coded, but determined flexibly by the control algorithm. With this procedure, the frame-based approach can overcome some of the limitations finite state-based systems have: they enable overanswering and allow information to be gathered in any order. On the other hand, they require a less restricted (and thus more error-prone) speech recognition, and can not be tested easily in an automated manner. Both are issues in particular for commercial systems.

Both the finite state-based and the frame-based approach are not suitable for more complex domains in which interaction is modeled as an interplay between two independent agents that collaborate to solve a task or to negotiate information. In such **agent-based** systems, both agents may introduce new topics or suggest actions. Often, the agents' beliefs, desires and intentions are modeled explicitly, and agents are capable of reasoning about them. This is what drives the dialog. Also, generic mechanisms for information grounding or error detection and recovery are implemented. In general, the dialog flow is not pre-structured in these systems, but planned dynamically based on the present context. Utterances tend to be more complex, and it is more important to capture their details. Accordingly, speech recognition must be almost unrestricted, while sophisticated natural language understanding is required. Most research in spoken dialog system has been dedicated to the agent-based approach, but it has not been transferred to the commercial field.

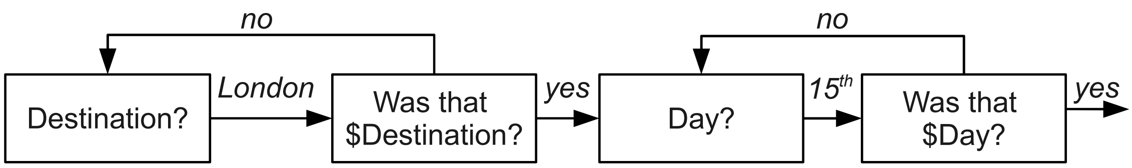


Figure 2.1: A simple dialog graph for a finite state-based travel system (after [McT04]).

A similar, but finer categorization is made by [AFS01]. As a subgroup of the frame-based approaches, he additionally mentions systems that can deal with and switch between **sets of contexts**, each represented based on frames. Also, he distinguishes between **plan-based** and **agent-based** models. In the plan-based approach, system and user collaboratively construct a plan, while the agent-based approach may involve a dynamically changing world that the agent needs to account for.

However, both categorizations are not quite consistent and mix the distinguishing criteria. While, in McTear's categorization, the finite state-based and the frame-based model are characterized by their implementation details, the agent-based model is characterized by the system's capabilities and the requirements of the domain. This applies, in a similar way, to Allen's categorization, too. Also, the categories are not general enough, as the frame-based and set of context-based models applies per definition only to form-filling domains.
Throughout this thesis, a more general categorization will be made. We will distinguish between **descriptive approaches** on the one hand and **mental-state approaches** on the other hand: descriptive approaches include all types of models that specify the dialog flow explicitly (at least partially), while in mental-state approaches the dialog flow is created dynamically, emerging from a model of the interaction goals or of the interaction partners' mental state, or based on deduction and logics. This categorization reflects the situation in linguistic dialog modeling, where two main traditions can be identified: approaches that model the internal attitudes of the interaction partners and their underlying cognitive state and, in contrast, approaches that describe the public and conventional aspects of how interaction typically proceeds [FE07].

2.1.2 Descriptive Approaches

A typical representative of the descriptive approach is the CSLU toolkit [SNC$^+$96, SCd$^+$98]. Through a graphical editor, a finite-state based dialog graph can be created that is based on dialog units that cover functions like answering the phone, speaking a prompt, recognizing speech input and the like. The graphical dialog specification is automatically translated into a script which can be executed by a special-purpose programming shell. Besides the graphical editor, the toolkit provides an environment for integrating speech signal analysis, speech synthesis, facial animation of a virtual character, and for training of speech recognition.

To date, the VoiceXML dialog mark-up language [FDH+04] is widely used, particularly in the commercial field. As VoiceXML has been designed for telephone-based applications, in the automotive field proprietary solutions are common, most of them similar to VoiceXML (e.g. [HWS+03]). VoiceXML has been recognized as a W3C standard since 2004 and is supplemented by other standards for specification of speech recognition grammars (SRGS) or speech synthesis output (SSML). Dialog scripts are interpreted by a browser which communicates through a transport protocol such as HTTP with an application server where the required knowledge sources (e.g. grammars or databases) reside. The basic element of a dialog script are forms and menus. A form contains a number of fields to be filled by the user, each associated with a prompt specifying the system output and a grammar restricting the user input. A menu presents a number of options to the user and thus provides a method for specifying the next dialog state, i.e. the next form to execute. Additionally, frame-based dialog modeling can be realized with the built-in form filling algorithm (FIA). This way of dialog modeling requires a specification of the forms only, as the dialog flow is determined by the FIA. The FIA executes the form associated with the next unfilled field. If the user overanswers, the respective field is filled, and the associated form can be skipped.
The main drawback of VoiceXML is, however, that it lacks support for abstraction. For example, the system prompts need to be specified on word-level. This makes it difficult to model dialog at a more abstract level, or to define reusable patterns of subdialogs, and is probably the reason why VoiceXML is rarely used in spoken dialog systems research. In the commercial sector, dialog scripts are often generated automatically from graphical specifications of the dialog state machine. The form-filling approach has not yet become established, due to harder testability.

However, descriptive approaches are not only common in the industry, and they are not limited to finite state-based dialog modeling, but also include more complex approaches, such as the RavenClaw dialog manager which is being developed at Carnegie Mellon University [BR09]. A large number of speech applications have been implemented with it, spanning from a bus information system [RLB+05], to calendar applications [SMHS07], to a support application for aircraft maintenance [BR05]. The RavenClaw dialog manager is embedded into the Olympus framework for conversational interfaces. In addition to RavenClaw, Olympus consists of components for speech recognition, semantic parsing, language generation and speech synthesis. The components communicate via a centralized message-passing infrastructure [BRH+07] (see section 3.2).
At the core of RavenClaw's dialog model is the dialog task specification, which encapsulates the domain-specific aspects of the control logic and forms a hierarchical plan for the interaction, executed by the domain-independent dialog engine. The dialog task specification is represented by a tree of *dialog agents*, each handling a subtask of the interaction, such as greeting the user or presenting the result of a database lookup. There are two types of dialog agents: *dialog agencies* that represent tasks that are further decomposed and

fundamental dialog agents that are terminal nodes in the tree, implementing atomic actions. The fundamental dialog agents further fall into four categories. An *Inform* agent produces an output, a *Request* agent requests information from the user, an *Expect* agent expects information from the user without explicitly requesting it, and an *Execute* agent performs back-end calls, such as database access. During interaction, the dialog engine traverses the tree in a depth-first manner, unless otherwise specified by pre- and postconditions or by error handling and repair activities. Agents from the task tree are put on top of a dialog stack in order to be executed and are eliminated when completed.

The RavenClaw framework will be discussed from a more hands-on perspective in section 4, where a re-implementation of the Curious Robot scenario – one of the development scenarios of the presented approach – with RavenClaw will be investigated in the scope of a comparative case-study of different dialog system approaches. A possible task tree for the Curious Robot is shown in figure 4.1.

RavenClaw has originally been categorized as an essentially plan-based dialog framework by its authors [BR09] using Mc Tear's categories, mainly because of the separation of domain specific and domain independent dialog knowledge. However, based on our above categorization, it clearly falls into the category of descriptive approaches, as it specifies the dialog flow beforehand in a descriptive way, even though deviations are admitted under specific conditions.

2.1.3 Mental-state Approaches

A well-known representative of the mental-state approaches is the TRIPS [FA98, AFS01] system (and its predecessor TRAINS [FAM96]). The TRAINS/TRIPS systems were developed at University of Rochester within a long-term project that lasted for almost a decade. The goals of this project were to "undertake a serious study of human-human spoken dialogue, [to] build a series of robust research prototypes untrained users could interact with, [and to] use the prototypes as a platform for in-depth research in natural language understanding, mixed-initiative planning, and reasoning about time, actions and events" [Tri00].

The TRIPS architecture features distributed dialog management and decision making. It does not include a central dialog manager as such, but relies on loosely coupled agents that exchange information by passing messages. Figure 2.2 shows the architecture in detail. Its main components are the interpretation manger, which is responsible for interpreting user utterances and the underlying intentions, the behavioral agent, which drives the system behavior with respect to problem solving, and the generation manager, which coordinates planning and generation of speech and multimodal output. The three subsystems work asynchronously.

In the TRIPS system, conversation is driven by a model of collaborative problem solving, together with a rich representation of the discourse context. The discourse context maintains information needed to coordinate the system's conversational behavior, i.e. information to generate and interpret anaphora and ellipses, turn-taking information and

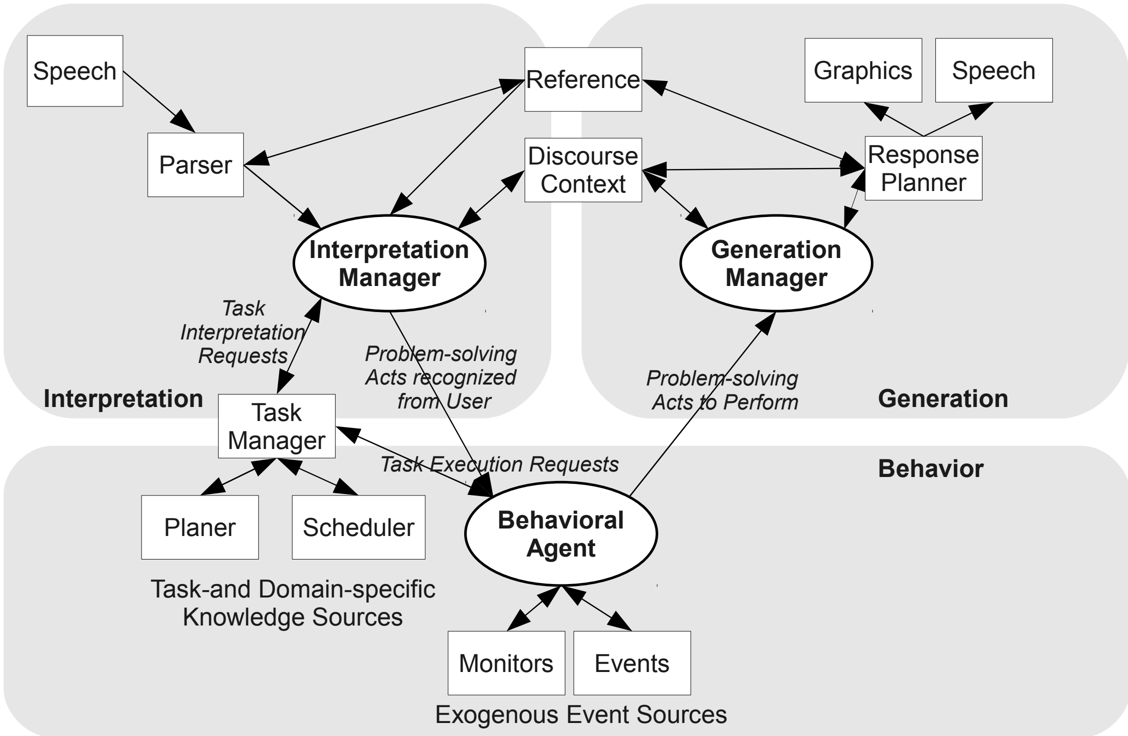


Figure 2.2: The TRIPS architecture (after [AFS01]).

information about outstanding discourse obligations. It assists the interpretation manager and the generation manager in interpreting the user's utterances and in generating speech and multimodal responses, respectively. The problem solving model is maintained by the task manager and describes the domain in terms of objectives, solutions, resources, situations and a number of different actions that both user and system can contribute to the collaborative task. On the one hand, it assists the interpretation manager in recognition and interpretation of user actions. On the other hand, it executes the problem solving steps selected by the behavioral agent.

Another approach that relies on a task model, together with a representation of the collaborative discourse state, is the collaboration manager Collagen (for *Coll*aborative a*gen*t) [RS98]. However, Collagen is not an actual dialog system, but intended as a plug-in for intelligent user interfaces. It takes a rather observational role, relying on the *collaborative interface paradigm*, which is illustrated in figure 2.3. In this paradigm, a

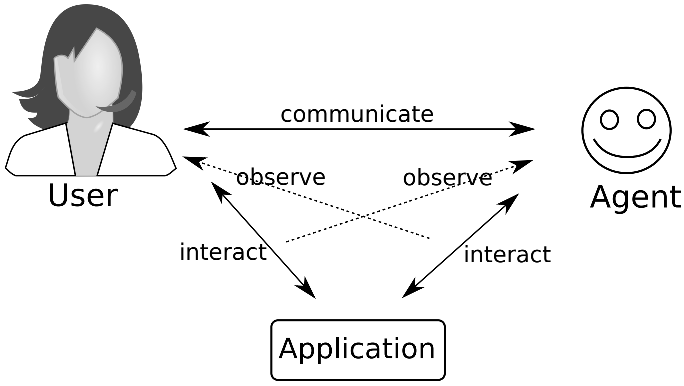


Figure 2.3: Collagen's collaborative interface agent paradigm (after [RS98]).

software agent[1] assists the user in operating an application, both communicating with each other as well as with the application. They are informed about each others' actions either by a reporting communication ("I have done x") or by direct observation.
Collagen maintains a task model for the specific application domain which defines the typical domain goals and procedures for achieving them. The task model is a collection of goal composition rules, called *recipes*. Collagen tracks the user's progress with respect to the task model and automatically generates system utterances and choices for user utterances, based on the current discourse state. One component of the discourse state is the *focus stack*, representing its attentional aspects. The focus stack contains hierarchical discourse segments, each contributing to a specific *shared plan*. A shared plan corresponds to the intentional aspects of the discourse and is represented as a(possibly still incomplete) *plan tree*, specifying the actions to be performed, and by whom.
A large number of desktop applications have been developed based on Collagen, including an assistant for air travel planning [RS98], for email [GSBR99], and for graphical user interface development [RSL01], a video recorder [RSL01] and a programmable thermostat [DKF01]. We will return to the Collagen approach in section 4 where the re-implementation of the Curious Robot scenario with different approaches is described.

An approach that focuses on the aspect of information exchange is the information state approach to dialog modeling [TL03]. The key idea is that the dialog is driven by the relevant aspects of information (the *information state*), and how they are updated by applying *update rules*, following a certain *update strategy*. The update rules regulate the update of the information state, given the current information state and the performed or observed dialog moves. The update strategy, then, decides which rules to apply if several

1 Notice that Collagen is not identical to the assisting software agent, which is treated as a "black box". Collagen is rather the mediator of the communication between the agent and the user (cf. figure 8 in [RS98]).

ones are applicable.
The Information State approach is formulated very generic, and the term *information state* is intentionally kept very abstract. One may choose to model the external aspects of the dialog, such as variables to assign, or rather the internal state of the agents, such as goals, intentions, beliefs and obligations, in order to realize a plan-based dialog management. Also the other components of the approach may be implemented very differently. For example, the update strategy can be as simple as to pick the first rule that applies, or it can be based on game or utility theory. Thus, it is not a dialog system in the narrower sense, but rather a toolkit to implement dialog systems and different theories of dialog modeling. Several implementations of the information state approach have been presented, including TrindiKit [TL03] and Dipper [BKLO03], and a number of dialog systems were constructed based on it (e.g. GoDiS [LLC+00], Siridus [KKERK03], Godot [TBC+02a] or WITAS [LBGP01]), some of them also on robots (see section 2.2.2). Section 4 will describe the re-implementation of the Curious Robot based on the information state approach.

2.2 Dialog Modeling for Human-Robot-Interaction

Human-robot interaction differs in many aspects from traditional speech interfaces as known from the information seeking or travel booking domain. These differences affect not only the interaction design, but also have implications for the dialog management approach itself. Section 2.2.1 discusses the characteristics of HRI compared to non-situated domains. The considerations based on this domain analysis strongly influenced the proposed approach and will be revisited at different locations of the thesis. Section 2.2.2 presents the state-of-the-art in dialog modeling on robots, which still reflects the historical separation of the dialog systems community and the HRI community: On the one hand, the traditional approaches to dialog modeling do not fully account for the characteristics of HRI, and on the other hand, roboticists lack expertise in dialog management and discourse modeling.

2.2.1 Characteristics of Human-Robot-Interaction

Robots are situated in their environment, which they can perceive and manipulate through their actions, and they are supposed to communicate about what they see and do. Also, the environment is dynamic, and the situation may change rapidly. A robot needs to be able to react to situation changes quickly, and this may involve (perhaps urgent) interaction activities. Interactions in such dynamic environments are not predictable beforehand, and they have no clearly defined beginning or end. i.e. they are potentially open-ended [LGBP02]. Also, interactions with multi-purpose service robots may span several domains, and it must be possible to switch between domains.
Joint activities are coordinated within interaction, for which the human interaction partner gives instructions and sets the goal. The activities may be temporally extended, and there may be multiple such interactions at a time. Their progress needs to be monitored over

time, and communicated to the human. Also, the human may wish to revise on-going activities and re-specify their goal, or to cancel them. Problems need to be solved jointly within interaction. In other situations, the robot might be supposed to act autonomously, which requires that it is able to take actions on its own initiative, and to comment on these.
Because of the mixed-initiative interaction style and, more importantly, because of its physical embodiment, the robot is, to a limited extent, perceived as an equal interaction partner. In addition, there is often no clearly defined role allocation such as expert and asker. Instead, both interaction partners assist each other, exchange information and learn from each other. In particular the latter – learning within interaction – is a feature specifically relevant to robotics. With the perception of the robot as a partner, social aspects of the interaction become more important. Humans might expect the robot to respect social conventions and main forms of politeness, or they might attribute to it a certain personality. Another implication of the robot's embodiment is that the interaction will involve not only verbal communication, but also non-verbal cues.

To sum up, in robotics we have to deal with **situated interactions** in **dynamic environments**, that are inherently **unstructured** and **open ended**. Within interaction, **joint activities** need to be coordinated that may be **temporally extended** and **executed concurrently**. Interaction follows a **mixed-initiative style** that enables **learning within interaction** and **social and multimodal behavior**.

2.2.2 State-Of-the-Art

Research in spoken dialog systems and research in HRI have long developed separately. While researchers in HRI have either been focusing on nonverbal aspects of communication, the spoken dialog systems community has been focusing on non-situated systems, mostly in information-seeking domains. As a consequence, the traditional approaches to dialog modeling often do not fully account for the above discussed characteristics of HRI (discussed further below and in section 4), while roboticists lack expertise in dialog management and discourse modeling. Only recently has the spoken dialog systems community discovered robotics as a new and challenging domain, and the communities have begun to converge. However, the state-of-that-art in HRI still reflects the historical separation.

Many robot systems, especially in the field of service robotics, possess **no dedicated dialog manager**. Interaction rather follows command-control style, based on simple key-word spotting and command matching techniques, or by pressing buttons on a touch screen. In such systems, the user gives instructions that trigger action execution, but execution proceeds black box-like and is not well integrated with the speech interface. Once execution has begun, the user can not intervene any more, nor can the robot request clarification or assistance from the human. Examples for this interaction style include the mobile service robots Perses [BWK+03] and Care-O-bot [PBRH07] and the museum guide RHINO [BCF+98].

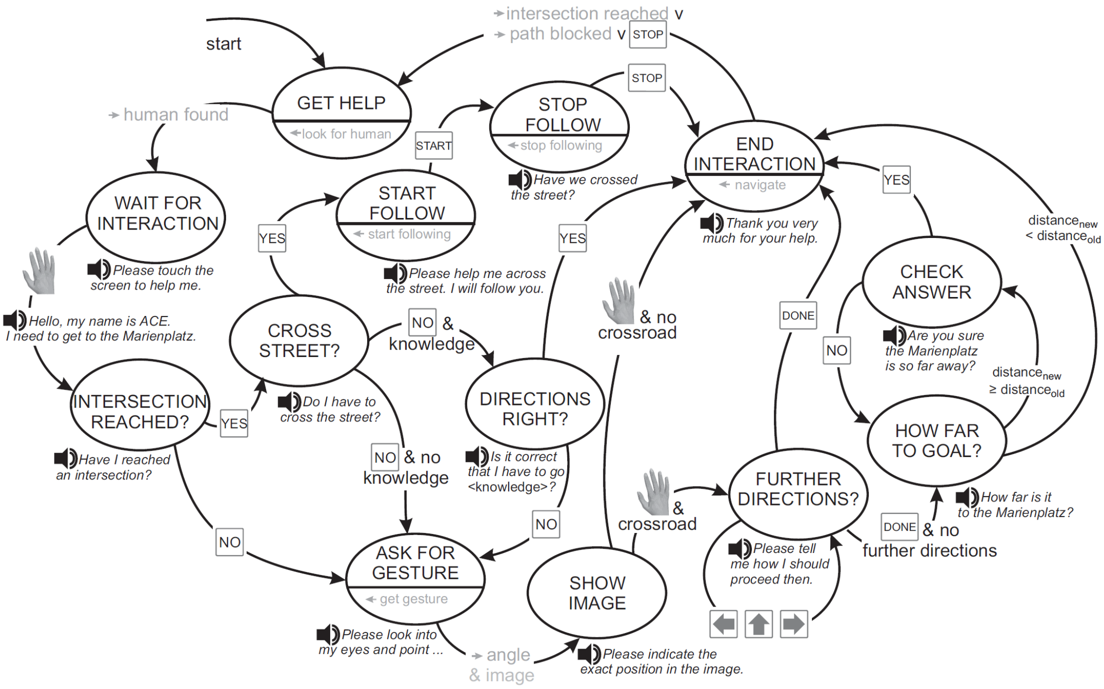


Figure 2.4: A finite state machine for HRI where dialog and task control are tightly coupled (from [BWB09]).

Most robot systems apply a **finite state-based approach** for dialog modeling as described in section 2.1.1. In contrast to the command-control approaches, where dialog and task control is not integrated well, there is even the risk that dialog and task control are too closely interwoven. As an example, figure 2.4 shows the finite state machine that models interaction for the Autonomous City Explorer [BWB09], where the the system's navigation capabilities are well integrated into the interaction by admitting system signals (marked gray) to trigger transitions. On the other hand, because of the tight coupling, the depicted model represents a very specific solution that can not be transferred to a new scenario easily. It will also be difficult to realized an unstructured interaction with mixed-initiative interaction style with it, as typically required in HRI. Also, the dialog graph is a monolithic control structure that will be difficult to expand and to maintain, particularly as the robot functionalities get more complex. Other systems that employ finite state-based dialog models are the Jijo-2 mobile robot for office services [MAF+99], the hospital assistant Hygeiorobot [Spi01] and the service robot HERMES [BG02].

Several robot systems adopt **plan-based** approaches to dialog modeling. In contrast to plan-based dialog models in traditional dialog systems, interaction planning is not performed by a dedicated dialog planner, but by a general action planning component. This is why plan-based approaches to HRI can often be found in domains where the robot explores its environment in order to perform tasks in it. For example, Dora The Explorer [HHS+10] aims to derive spatial and categorical knowledge of its environment, driven by ontology-based reasoning. Krujiff and colleagues have developed a robot that learns about visual objects by further inquire about what it does not understand, building up a multi-agent belief model based on abductive reasoning [KJL10]. Dzifcak and colleagues present a system for navigation in office environments that generates goal representation in temporal and dynamic logic from natural language instructions and produces an action sequence to achieve it [DSBS09].
Similar to the finite state-based approach, in plan-based systems the dialog behavior is coupled closely with the system behavior. Since dialog emerges from reasoning about the domain, this might result in non-intuitive and unpredictable dialog behavior that is difficult to tune [PH05]. Also, interaction in plan-based systems is restricted to the task at hand. Social utterances, small-talk or pleasantries, which may make up an integral part of HRI in some scenarios, can not be modeled well with this approach because such types of utterances can not be translated easily into some kind of logic or into a goal representation.

Besides the general approaches discussed above, several dialog models that had originally been developed for non-situated domains have been applied to HRI. One of them is the collaboration manager **Collagen**, which has been described in section 2.1.3. It has been applied to implement the interaction with the penguin robot Mel that engages human visitors in a conversation in order to instruct them in a simple manipulation task [SLK+05]. The focus of this scenario lies in modeling engagement gestures such as head movements and gaze, but it is not very sophisticated from a dialog modeling perspective. Basically, the robot lists the necessary steps one after another, and the human acknowledges their execution. The main reason for this is that Collagen is not intended as a dialog manager, but rather as an observing collaboration manager. Hence, it is not well suitable for dialog modeling as such, whether in classical domains or in HRI. The case-study described in section 4 confirms this observation.

Also **RavenClaw** (cf. section 2.1.2) has been applied to a HRI scenario. In this scenario, a multi-robot team performs a search task, guided by the instructions of a human operator. The robots are not aware of each other, i.e. each uses its own instance of RavenClaw for communication with the user. Instructions can either be addressed to a specific robot directly, or be broadcasted so that the robots will bid on the task. Bids are not managed through interaction, but through a central operator software. As a domain-independent dialog framework, RavenClaw can well be applied to HRI. However, the case-study described in section 4 suggests that the back-end integration does not meet all requirements of robotics, in particular with respect to asynchronous coordination of back-end calls. This might become an issue in HRI scenarios in which the robot acts less autonomous, but requires more guidance and assistance during action execution.

The **Information State** approach (cf. section 2.1.3) has been used in robotics several times. One of its application is the WITAS dialog system for multi-modal conversations with an unmanned helicopter that has on-board planning and vision capabilities [LGBP02, Lem03]. WITAS makes use of an Activity Model that describes the system's planned activities, current activities, and their execution state, and of a dialog move tree that represents the current state of the conversation. Utterances are represented as logical forms that may refer to activities in the Activity Model. Together with the current state of the dialog move tree, they cause an update of the Information State (which in turn may affect both the Activity Model and the dialog move tree). Other examples include the mobile robots Godot [TBC+02b] and the one presented by Burke and colleagues [BHL02], and the virtual agent Hassan [GDR+08].
Unlike the plan-based approaches to HRI, these systems posses a dedicated dialog planner, based on the Information State approach. It is often combined with a general action planning component, e.g. in form of a planner agent that communicates with the dialog manager via the architectural framework. Thus, domain and dialog planning are kept separate. However, a potential drawback in more open domains and dynamic environments is the increasing amount of update rules, whose order may even effect the system behavior. This becomes evident already in the toy example described in section 4.

Altogether, it can be stated that for dialog modeling on robots, either rather simple approaches are common, or approaches that have originally been developed for other types of domains. Both are not capable fully accounting for the special characteristics of HRI as detailed in section 2.2.1. In particular, the descriptive approaches are not capable to realize a flexible dialog flow that enables the robot to react quickly to its dynamic environment. Mental-state approaches, in contrast, are often too information-oriented (i.e. the Information State approach), or do not account for the social aspects of HRI (i.e. the plan-based approaches). Another issue is the integration of dialog management and task management. To put it simple: it is either too close, or not close enough. Surprisingly, approaches that are specifically designed to model dialog with robots do not exist so far.

2.3 Evaluation of Interactive Systems

The spoken dialog systems community has long been focusing on evaluating interaction in non-situated systems, such as telephone- and PC-based information access. Within the HRI community, on the other hand, evaluation of interaction often has been neglected. Instead, evaluation focuses on isolated aspects of the robot system, and often even only on effects of physical appearance of the robot. Sections 2.3.1 and 2.3.2 describe the methods and techniques that are common in each of the fields.

2.3.1 Evaluating Dialog Systems

Initially, a mainly system-centered view on evaluation predominated in the SDS community. One of the first approaches to evaluating the performance of a dialog system was the concept of **reference answers** [HDM+90], where the system's performance is assessed through the proportion of system answers that match pre-defined reference answers. This approach can not account for different expedient dialog strategies, and it is mainly applicable within the information retrieval domain (which has been the focus of research in SDS at that time).

Subsequently, various more refined **objective measures** for system performance have been proposed, such as inappropriate utterance ratio, correction ratio, concept accuracy, transaction success and number of turns (e.g. [HDM+90], [HP93], [SWP92], [DG95]). With these metrics, that can be obtained automatically from interaction logs, interactions become quantifiable and comparable with each other. Also different dialog strategies can be compared based on the metrics. Consider a comparison of two railway information systems [DG95], where system A uses an explicit confirmation strategy, whereas system B uses an implicit confirmation strategy, as illustrated in table 2.1. Danieli and Gerbino found that the explicit confirmation strategy had a higher transaction success rate and produced less inappropriate utterances and repairs, but generated dialogs that were twice as long as with the implicit confirmation strategy.

User:	I want to go from Torino to Milano.
System A:	Do you want to go from Trento to Milano? Yes or No?
User:	No.
User:	I want to go from Torino to Milano.
System B:	At which time do you want to leave from Merano to Milano?
User:	No, I want to leave from Torino in the evening.

Table 2.1: Explicit and implicit confirmation strategy for a railway information system (after [DG95]).

However, one limitation of the objective metrics approach is that the user's subjective perception of the system is not included in the evaluation, i.e. they describe the behavior of the system during an interaction, but do not necessarily reflect the perceived performance. In the above example, Daniele and Gerbino concluded that the explicit confirmation was more robust, but could not tell whether system A's higher transaction success or system B's efficiency was preferred by the users. In general, making such interpreting statements requires to identify the factors that affect user satisfaction. Moreover, once the relevant factors for user satisfaction have been found, the results can be expected to generalize (to a certain extent) to other (comparable) domains and tasks.

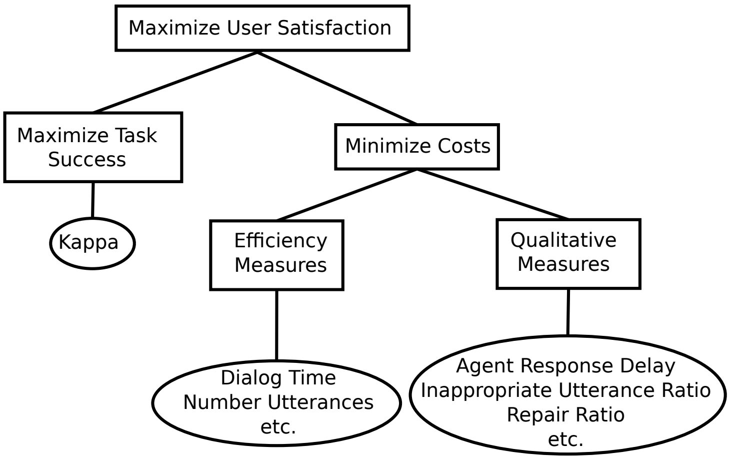


Figure 2.5: PARADISE's structure of objectives for spoken dialog performance (after [WLKA98])

Against this background, Walker and colleagues introduced a more user-centered view on evaluation. They proposed a general methodology for evaluating dialog systems called PARADISE (PARAdigm for DIalogue System Evaluation) that **combines objective and subjective measures** [WLKA98]. Its main idea is to estimate subjective judgments of user satisfaction as a linear combination of several objective measures. For this purpose, several assumptions are made. First of all, PARADISE assumes that system performance (the target variable) is correlated with, and can be approximated by, user satisfaction. User satisfaction is a subjective variable that can be assessed e.g. through a questionnaire. The model further posits that there are two types of factors that are potentially relevant for user satisfaction: task success and dialog costs, with the latter being comprised of dialog efficiency and dialog quality. Using linear regression, the relative contribution of the success and cost factors to user satisfaction are quantified. The described structure is shown in figure 2.5.

For each of the categories, appropriate objective measures are proposed. Dialog efficiency can be assessed for instance through the overall number of utterances and the time required to achieve a given goal. As measures for dialog quality, the average system response delay, inappropriate utterance and repair ratio are suggested. To calculate the task success, the dialog task is represented as an attribute-value matrix (AVM). The AVM consists of the information that must be exchanged between the system and the user during the dialog, represented as a set of ordered pairs of attributes and their possible values. An individual interaction is represented as an instantiation of the AVM, containing the information that has actually been exchanged. Based on comparisons between the AVM for the task and for the individual interactions, a confusion matrix is generated. Whenever information has been transmitted correctly, i.e. the corresponding value in the interaction AVM matches the task AVM, the appropriate diagonal entry of the confusion matrix is incremented by 1.

The off diagonal entries represent misunderstandings that remained uncorrected in the dialog, whereas the corrected misunderstandings are not incorporated in the confusion matrix (but their effect is reflected in the dialog costs). Task success is operationalized using the Kappa coefficient

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

which is calculated from the confusion matrix. P(A) is the proportion of times that the AVMs for the individual interactions agree with the task AVMs, and P(E) is the proportion of times that they are expected to agree by chance. Originally, the Kappa coefficient has been introduced for measuring inter-annotator reliability [Car96], but it can also be used to express how well the transmitted information matches the intended information.

To sum up, the PARADISE evaluation framework enhances previous evaluation methodology by identifying the objective factors that contribute to the subjective user satisfaction. It is thus able to offer an explanatory model for user satisfaction. As the results obtained are expected to generalize across domains, they can serve as predictive model for future applications. Also, by using a task representation that abstracts from a specific dialog strategy, the approach decouples *what* the system needs to achieve from *how* the system carries out the task via dialogue, which supports comparisons among dialog strategies.

Several shortcomings of PARADISE have been reported. Robinson and colleagues criticized the fundamental idea of the PARADISE approach: that the system performance should only be determined by the users of the system (i.e. through the user satisfaction), without including any objective rating [RRT10].
Möller and colleagues found that the generalizability of the results was significantly reduced when extrapolating from one system to another: the variance in the data that could be predicted with the model decreased from around 50-70% in the training data to 30-50% in the unseen data [MES08]. They conclude that, while the predictive power of the model is limited, it still does a good job in judging individual users and interactions, and can be useful for system optimization.
Other points of criticisms concern the proposed measures. Both the objective measures and the subjective user satisfaction rating are kept so general "as to be of questionable usefulness" [RRT10]. For example, Kamm and colleagues identified as relevant factors task success and dialog quality [KWL99], which is neither surprising nor helpful.
Further, it has often been reported that the speech recognizer performance is the major predictor of a dialog system's performance, and it has even been shown that with increasing recognition performance the significance of the other predictors can change [WBK99]. Hajdinjak conclude that the influence of automatic speech recognition is so large as to hinder the other factors from showing significance [HM06].
Hone and Graham criticized in particular the questionnaire that was proposed to assess

user satisfaction [WKL00], as being based "neither on theory nor on well-conducted empirical research" [HG00]. Also, the items in the questionnaire are all summed up although they refer to different aspects of user satisfaction. Hone and Graham argue that it would be more appropriate to sum up only subcategories that measure the same aspect.
Finally, another issue is that the proposed task representation using attribute-value pairs is appropriate only for information retrieval tasks. This means that for other, less information-dominated domains such as robotics, different measures for task success have to be found.

In parallel, the user-centered view has become even more important, leading to approaches that are purely based on **subjective measures**, and with it to the development of valid and reliable questionnaires. Several generic subjective usability questionnaires have been proposed, and have been applied to speech applications also. The best known are the Questionnaire for User Interaction Satisfaction (QUIS) [Shn86] and the Software Usability Measurement Inventory (SUMI) [Kir96]. QUIS consists of five question categories, referring to the overall reaction to the software, reactions to the display, to terminology and system information, to learnability and to the system's capabilities. QUIS was initially published without empirical validation, which was however approved later [CDN88]. In contrast, SUMI was validated in an iterative process, with the resulting main categories likeability, efficiency, helpfulness, control and learnability.
Despite the development of generic questionnaires, ad-hoc techniques predominated in evaluation of speech systems until then (inlcuding the questionnaires used in PARADISE-style evaluations). To address these shortcomings, Hone and Graham developed the Subjective Assessment of Speech System Interfaces (SASSI) questionnaire [HG00]. The aim of SASSI is to provide a tool that is widely applicable to all types of speech recognition interfaces. SASSI was validated in an iterative process, from which result the six main categories system response, accuracy, likeability, cognitive demand, annoyance, habitability and speed.

Despite the above mentioned criticism, the PARADISE approach has established as state-of-the-art in dialog system evaluation, and has been employed extensively over the past twenty years. Most of the points of criticism can be overcome by revising the measurements used, and by combining the PARADISE approach with reliable and validated questionnaires.

2.3.2 Evaluating Interactive Robot Systems

Robotics is a highly diverse field, ranging from autonomous rescue robots to general-purpose service robots or social toy robots. Accordingly, no standard methodology for evaluation has matured, such as the PARADISE approach for evaluating dialog systems. Much of the evaluation is done on **component or algorithmic level** of the (not necessarily interactive) robot system. Often, evaluation is done by performing a given task in the particular domain, measuring performance robustness and efficiency of the algorithm. For example, in robot navigation and SLAM, possible tasks include traversing a door-

way [EAC00], navigating through an office environment [MEBF+10] or re-localizing after being kidnapped to a random place [TFBD00]. Typical benchmarks for manipulation tasks include grasping a set of everyday objects [RHSR07] or opening a jar [SEHR10]. In interactive object learning, recognition rates and required time per object are measured [KWK07], [RDO11].
The task is performed either in the real world, or in simulation (where parameters such as the amount of sensor noise can easily be controlled and varied), or both. The specific measures used depend on the purpose of the component or algorithm that is subject to evaluation, and on the representations used.

In the area of social robots, evaluation has been focusing on **single aspects of the system**, too. A large amount of research effort has been dedicated to the external appearance of robots, addressing the question of how appearance influences the user's expectations of the system, and the interpretation of the system's actions. For example, Hegel and colleagues used neuroimaging data to investigate how a robot's randomized actions in a game setup are perceived by its human opponent and how their interpretation is influenced by the appearance of the robot [HKK+08], using neuroimaging data. Also, the relation between predictability and anthropomorphic inferences and acceptance of the robot was investigated [EKB11], as well as the attributations made to a robot that is percieved as uncanny [HMP08]. Lohse and colleagues have shown pictures of real robots to participants and asked what kind of task they would expect the robot to perform [LHS+07], showing that the system's appearance and abilities strongly influence people's expectations.
Most of the above studies did not involve real interaction with the system, but were based on showing pictures or silent video clips. However, while the appearance of a system undoubtly shapes the user's expectations, the author believes that expectations are likely to be accommodated quickly during interaction, depending on the actual system behavior. Hence, the significance of the above studies for scenarios that involve an actual task is limited.

Also evaluations that include real interactions with a system mainly concentrate on single aspects of the system. Many of them concern the behavior of mobile robots in everyday environments, like the preferred approaching behavior when the human is unaware of the robot's presence [SKG+09]. Spatial preferences during interaction were investigated too [WDW+06], leading to a "robotic etiquette" for robot companions [WDWK07]. Perrin and colleagues have compared six different ways to convey navigational information provided by a robot to a human: visual, auditory, and tactile feedback modalities, and combination of these [PCR+08]. Using both objective measures (accuracy and rapidity of the user's reaction) and subjective measures (gathered through a questionnaire), they assessed suitability of the information given by the robot.

Another aspect that has been addressed by numerous studies is nonverbal communication and behavior. One of the most important communicative resources in human-human communication is gaze: Seeing where the interaction partner looks at provides valuable information about what is being talked about, and facilitates to establish joint attention, mutual understanding and to show engagement in interaction. Accordingly, many studies have addressed the role of gaze in human-robot interaction. For example, Staudte and Crocker investigated the influence of both robot speech and gaze on the human gaze, and on the human's understanding of a robot utterance [SC09]. Similarly, Sidner and colleagues investigated the impact of where a robot looks during conversation with regard to objects of interest referred to within the conversation [SLK+05]. Mutlu and colleagues modeled human-like gaze behavior in a story-telling setup and tested the influence of gaze frequency on the subject's task performance and on their perception of the robot [MHF06]. Apart from gaze behavior, other nonverbal signals concerning a robot's mimics have been investigated, including synchronized lip movements [Bre03] or the expression of basic emotions such as happiness, fear, anger, disgust and sadness and surprise [HSW+06], [KFS10].

Methods applied for evaluation can be classified into three groups: subjective measures, objective measures and physiological measurements. Table 2.2 shows for the above mentioned studies the applied evaluation methods. The most frequently used evaluation methods is collecting **subjective measures**, almost always in form of questionnaires measuring the users' attitudes. While this technique is quick to conduct, its conceptual pitfalls and limitations are often underestimated [BCK08]: The development of a validated questionnaire involves a considerable amount of work, but is indispensable in order to achieve valid and comparable results. Also, users may assess their experience differently afterwards, possibly tending to socially accepted answers.
A perhaps less influencable method is to gather **objective measures**. This can be done by observing the user behavior, measuring parameters such as the time spent for the interaction with the robot or the user's response time. However, not each variable in a study is reflected in observable behavior. Moreover, assessing the user's behavior based on video recordings manually is a tedious task. An alternative way to collect objective measures is by logging system data (e.g. task success measures), which enables automated data acquisition and analysis.
A third method is to rely on **physiological measurements**. They are taken not after, but during interaction, and they provide a direct indication of the user's state as they can hardly be influenced deliberately. Unfortunately, the results are often not clearly interpretable. For instance, skin conductivity and heart rate provide an indication of the user's arousal, but it is impossible to say whether the arousal is caused by anger or by joy [BCK08].
Several measures can complement each other. Two recommendations have been given as best practice for evaluation of interactive robot systems: First, studies should be conducted with a large number of participants to gain a higher probability of obtaining statistically significant results, and, second, several methods of evaluation should be combined to achieve reliable results [BM10]. However, this view neglects that valuable observations

can be made also with a smaller number of participants through qualitative analysis.

Aspect	***Example study***	***Evaluation method***	***Interaction type***
Appearance	[HKK+08]	Physiological measurements	WOz
	[EKB11]	Questionnaire	Video clip
	[HMP08]	Questionnaire	Video clip
	[LHS+07]	Questionnaire	Video clip
Spatial behavior and navigation	[SKG+09]	Questionnaire	Real interaction
	[WDW+06]	Questionnaire	WOz
Gaze behavior and mimics	[SC09]	Eye-tracking	Video clip
	[SLK+05]	Questionnaire	Real interaction
	[MHF06]	Questionnaire	Real interaction
	[Bre03]	Questionnaire	Video clip
	[KFS10]	Questionnaire	Video clip

Table 2.2: A sample of aspects that have been investigated in interactive robots.

In the last decade, an effort to identify **standardized metrics** for human-robot interaction has been observed. A common evaluation standard would make findings comparable, and the notion of "system quality" would become measurable. The main challenge is the diverse range of human-robot applications, which is the reason why metrics from other fields (e.g. dialog system evaluation) can not directly be transferred to robotics.

An early suggestion has been made by Dautenhahn and Werry [DW02]. Inspired by an existing technique in psychology, they determined so-called micro-behaviors that the user displays during interaction with the robot, such as gaze, pointing or touch. These micro-behaviors are coded manually on a second-by-second basis and can serve as a basis for statistical analysis. However, they do not capture the actual task success of a scenario, but rather the social aspects of the interaction.

Olsen and Goodrich have proposed six metrics for human-robot-interaction: *task effectiveness*, *neglect tolerance* (i.e. how the robot's task effectiveness declines over time when the robot is neglected by the user), the robot's *attention demand* (i.e. the fraction of total task time that the user must attend to the robot) and its complement, the *free time*, *fan out* (i.e. the number of robots that a user can effectively operate at once) and *interaction effort*. The proposed metrics are kept rather generic, and no concrete suggestion is made how they could be operationalized.

More concrete measures were proposed by Steinfeld and colleagues [SFK+06]. They proposed five categories of metrics for task-oriented mobile robots: *navigation*, *perception*, *management*, *manipulation* and *social*, and discuss example metrics for each categoriy. Further, they propose that the quality of human-robot interaction can be analyzed in

terms of three aspects: *operator performance*, *robot performance* and *system performance*, measuring how well the human and the robot perform as a team. For example, a measure proposed for operator performance is the cognitive workload, indicated by physiological data. A measure for robot performance could be the robot's autonomy, indicated by above mentioned neglect tolerance. As a metric for system performance, they suggested for instance the appropriate use of mixed-initiative, measured through the percentage of requests for assistance made by robot and user, respectively, and the number of interruptions of the operator rated as non-critical. While several possible metrics are proposed, no instruction is suggested how they could be offset against one another.
In general, most research on evaluation metrics has not concentrated on developing generic metrics, but either with specific tasks in mind, such as imitation tasks [AND06], motion planning [CC01] and logistics [EKL$^+$11], or within specific settings, such as multi-robot [SWK08] or teleoperated systems [Gat08]. It can be said that this is still an open, and challenging, research issue.

Similar as in evaluation of dialog systems, **standardized questionnaires** have been developed. Nomura and colleagues have developed the Negative Attitudes Towards Robots Scale (NARS) [NKSK04], [NKS06], which they used to explain participants' assessment of robot behavior styles. Moreover, it was used to measure changes in attitudes towards robots over time in a long-term interaction setting [NKSK08]. Its internal consistency and validity could be confirmed in Japanese studies [NKSK04]. Bartneck and colleagues have developed the *Godspeed* questionnaire that measures the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots, whose reliability and validity were indicated by several empirical studies [BCK08].
Both questionnaires gauge users' general attitude towards robots, or to a certain type of robot, using questions such as "I would feel very nervous just standing in front of a robot" or "I feel that in the future society will be dominated by robots." Hence, they are not designed to assess a real interaction with the robot, which reflects the above mentioned deficiency in evaluation of robot systems: that a substantial part of the research work is focusing on purely external aspects, not considering a real experience with the robot.

In contrast, benchmarks are a common means to measure success and quality of human-robot interaction in an **integrated approach**. For the purpose of benchmarking human-robot interaction, the RoboCup@Home league was founded in 2007 [LSR06]. In this competition, robots perform tasks such as finding objects, following people, or detecting faces in realistic everyday environments. The measurement is judged by a jury on the basis of the competition rules. However, while such competitions are an appropriate means for comparing the performance of different robot systems, they do not provide insights into the underlying causes.
An integrated approach, analysing both system and interaction level in an integrated manner has been proposed by Lohse and colleagues [LHS$^+$07]. Their systemic interaction analysis (SInA) approach draws from both multimodal interaction analysis, but considers

not only the users' behavior, but also what happens *within* the system. After a qualitative analysis of the video material, cases are identified in which the interaction deviates from the prototypical interaction, and their underlying system design problems. This enables the development of possible solutions, such as redesigning system components, or affecting the users' behavior e.g. by giving more feedback.
With the PARADISE approach, which established itself as state-of-the-art in dialog system evaluation, interaction and system level can be related in an even more systematic manner, by identifying the relative contributions of the different system parameters to the users' perception of the system performance (cf section 2.3.1). For example, Foster and colleagues have applied PARADISE in a construction task, where the user builds wooden construction toys according to the robot's instructions [FGK09]. Several success measures for dialog quality, dialog efficiency, and task success are used. Dethlefs and colleagues apply PARADISE to a wayfinding tasks, where the user requests a route description. As a metrics for success they measured how well the user was able to find the target location based on the robot's instruction. However, in both scenarios, the user has relatively little interaction choices. For instance, in the construction task, all the user can do is either acknowledge the construction progress, or ask the robot to repeat the current step.

To sum up, no standard methodology for evaluation has matured in robotics so far. Evaluation often focuses on single components, or on single aspects of the systems. Accordingly, no standard set of metrics has been identified, which is due to the diversity of applications. Standardized questionnaires typically address only external aspects of the robot, or the user's general attitude towards robots. Thus, they are not suitable for evaluating real interaction scenarios. In contrast, the PARADISE approach would be a viable approach for this purpose. Surprisingly, it is being used rarely in the robotics community. The reason for that might be, first, that the integrated view to HRI is often neglected, and, second, the historical separation of the communities, which is reflected also by the methodologies used.

2.4 The Adopted Approach

Based on the previous discussions, the following section puts the adopted approach into context of existing directions of research in dialog modeling and evaluation.

2.4.1 Dialog Modeling

In section 2.1.1, we proposed a general distinction between descriptive and mental-state approaches to dialog modeling. Both have their benefits and drawbacks, as discussed throughout the sections 2.1.2, 2.1.3 and 2.2.1. The main advantage of the descriptive approaches is their simplicity and their robustness (at least for smaller systems). One of their drawbacks is, however, that the dialog flow needs to be specified in advance. This holds not only for finite state-based approaches, which allow no deviations at all, but also for more sophistic descriptive approaches such as RavenClaw, whose dialog task specification still requires a pre-structured dialog flow, but allows to specify conditions that trigger deviations. While this is well suited for well-structured and predictable domains, it is not feasible in robotics, where the dialog flow typically is open and depends on changes in the dynamic environment.

The mental-state approaches, on the other hand, feature more flexibility with respect to the dialog flow. Because their behavior is very complex, it requires intimate knowledge of the dialog manager's functioning to build an application. This stands in contrast to the descriptive approaches, where tools to support the development process (e.g. graphical editors) long exist. Also, because of their complexity, system behavior may become difficult to predict and result in unintuitive dialogs that are hard to tweak. Further, testability and robustness is an issue.

The information-oriented approaches to dialog modeling (i.e. both the frame-based and the Information State approach) might not be appropriate for robot scenarios that focus not so much on exchanging and negotiating information, but rather on learning, performing or demonstrating actions. For this kind of scenarios an action state approach would be more suitable.

The present work suggests a **hybrid approach**: Local dialog management is modeled in a descriptive manner (in form of Interaction Patterns), whereas global dialog management, i.e. the specification of the dialog flow, is not fixed beforehand but determined flexibly, based on the state of system and environment. This achieves the flexibility of mental-state approaches, while preserving the simplicity and robustness of a purely descriptive approach. Moreover, the Interaction Pattern approach is based on an abstraction of dialog, which enables reusability across scenarios – a feature that most descriptive approaches do not implement. The integration of dialog and task modeling – an issue that is often neglected – will be based on an abstract Task State Protocol.

2.4.2 Evaluation

The evaluations and analyses conducted within the scope of this thesis aimed to consider as many aspects as possible. Therefore, not one single study was conducted, but several kinds of studies and evaluation methods were combined. They can be divided into two groups: **developer-centered evaluations**, i.e. the evaluation of the dialog framework, and the **end-user-centered evaluations**, i.e. the evaluation of the HRI scenarios in which the dialog framework was deployed. Table 2.3 lists the different aspects that were evaluated.

One aspect of the **developer-centered evaluation** is the **usability of the framework**. This evaluation provides information about the ease with which developers can work with the framework, by observing them performing a given task. It also gives information about

Field	*Aspect*	*Study type*	*Method*	*Chapter*
Developer-centered	Usability	Usability test	Descriptive statistics, observation	4.1
	Efficacy	Comparative case study	Observation	4
End-user-centered	Does robot initiative facilitate learning?	Explorative test run	Video analysis	6.1.3
	Does robot initiative facilitate interaction?	Explorative video study	Descriptive statistics	6.2.3
	Speech understanding performance	Explorative user study	Descriptive statistics	6.3.2
	User strategies in object teaching	WOz study	Qualitative analysis	7.1.2
	Speech recognition performance	Explorative test run	Descriptive statistics	7.1.3
	Factors relevant to user satisfaction	User study	PARADISE approach	7.4, 7.4.2
	Influence of robot initiative	User study	Descriptive statistics	7.4, 7.4.2
	Recurring phenomena	User study	Qualitative analysis	7.4, 7.4.2

Table 2.3: Aspects of dialog modeling evaluated in this thesis.

possible drawbacks in the API. Another aspect is **framework efficacy**, i.e. the question how well suited the framework is for its intended purpose. This question is investigated by analyzing a comparative case study, wherein a given target scenario was reimplemented by using different state-of-the-art dialog frameworks.

The **end-user centered evaluation** address different levels: (i) initially, new dialog strategies were explored, (ii) concrete system parameters were optimized during an iterative design process, and (iii) a user-study was conducted that promises generalizable results. On the dialog strategy level, the focus was placed on exploring mixed-initiative interaction strategies. Two facets of mixed initiative were investigated: can it **facilitate learning**, and can it **facilitate interaction**? The first aspect was evaluated by analyzing an explorative test run, the latter by analyzing a video study.
During the iterative design process of our main scenario, the Curious Flobi, several parameters were optimized. First, the **speech understanding performance** of the predecessor scenario was measured by means of a user study. This analysis helped to derive

strategies how to reduce speech understanding problems in future iterations. Second, a Wizard-of-Oz (WOz) study on object teaching was analysed in order to gain an impression of how users demonstrate objects to a robot and of the **interaction strategies** they use. This analysis formed the basis for the design of the speech recognition grammar and the dialog system configuration. To exclude that the resulting speech recognition grammar is too complex, which would reduce recognition accuracy, a pre-test of **speech recognition performance** was conducted.

With the result, a large-scale user study became possible. The study followed an integrated approach, focusing not on single aspects of the system, but rather on the overall system. One objective of the study was to identify the **factors that contribute to user satisfaction**. This was evaluated by means of a PARADISE-style regression analysis. Further, the mixed-initiative dialog strategy was evaluated in this scenario, too. As a three-level between-subjects factor, the **degree of robot initiative** was varied. Moreover, a qualitative analysis revealed **interaction phenomena**, whose causes were then traced back to the responsible system components in order to identify possible solutions.

3 Specifying the External Interface: The Task State Protocol

This chapter takes an external view on the suggested approach and focuses on the external interface of the dialog framework and its role within the robot architecture. Section 3.1 states the importance of a well-defined component interface and identifies its requirements, which partly result from the general requirements of HRI as discussed in section 2.2.1. Architectures for dialog systems will be discussed in section 3.2, with particular focus on their integration with the domain back-end. Section 3.3 presents an approach for a component interface that relies on the concept of tasks and their states. Section 3.4 illustrates its benefit for HRI and describes how it supports crucial features of HRI, such as mixed task initiative, tight integration of action execution and interaction, multitasking and interactive learning.

3.1 Motivation

A dialog system for human-robot interaction is not a stand-alone component, but embedded in the robot control architecture that consists of several diverse components and may be highly complex. The components of the architecture have to communicate with each other to cooperate. In this process, the dialog system plays a key role: it can be seen as a mediator between the user and the robot's functionalities. For instance, if the human gives an instruction, the dialog system will ensure that it is translated into the corresponding system command so that it can be handled by the responsible component. Conversely, the dialog system will keep the human informed about the current system state.

Out of the characteristics of human-robot-interaction that were discussed in section 2.2.1, several requirements for the communication between the dialog system and the robot subsystem arise. First, because of the situatedness of human-robot interaction, on-going activities and visible objects will be the topic of conversation. Thus, to achieve real-world integration, the communication needs to support close collaboration between the dialog system and the respective robot components to integrate their results into the interaction. Second, human-robot interaction requires the coordination of joint activities, and there may be several concurrent activities. Also, they may be longer running and need to be monitored over time. On the other hand, due to the dynamics of the environment, situation changes may happen at any time, and require human intervention, or at least notification. This means that the coordination of the activities needs to be asynchronous, i.e. the dialog system does not wait until a specific activity is finished, but continues computation and still remains responsive to status updates of ongoing activities. Third, human-robot interaction involves a mixed-initiative interaction style. For the communication between components, this means that the communication flow needs to be bidirectional. Interaction episodes will be initiated not only by the dialog system, upon a human's initiative, but also by other system components, if human intervention or clarification is required. Accordingly, in order to equip a robot with the ability to take task initiative autonomously, system activities will not only be triggered by the dialog system, on the human's instruction, but may be triggered by other system components as well. As a consequence, the responsibility for the interaction flow will not lie exclusively with the dialog system any more (cf. 3.8.2 for further discussion).

The way a component interacts with other components, i.e. its external interface, greatly affects its degree of integration with the rest of the system, and the effort integration takes. A design principle to achieve good interoperability is a contractually defined component interface that specifies the structure and semantics of the information exchanged between components, as well as an agreement how to use it [BS09]. This chapter will focus more on the information that is exchanged, while the following chapter will detail how the information is used, i.e. how the dialog manager internally takes decisions based on it.

This chapter will present the Task State protocol: an approach for a well-defined dialog system interface that i) meets the above described requirements for communication between the dialog system and the robotic subsystem and ii) allows the dialog system to treat all components in a uniform manner, instead of relying on individual back-end interfaces to the different components.

The Task State protocol, or Task State communication pattern, has been developed and investigated within several projects, and several colleagues have contributed to it, namely the Home-Tour team (consisting of Marc Hanheide, Frederic Siepmann, Torsten Spexard and myself) and the Curious Robot team (consisting of Ingo Lütkebohle and Robert Haschke and myself). A very first draft was used in the Home-Tour scenario (see chapter 6.1), for the communication between the then used Sunshine Dialog system (see section 5.1) and a room classification component. Later, within the Curious Robot scenario (see chapter 6.2), more use cases were identified, and the communication pattern was further refined. Lütkebohle investigated its role as a general coordination pattern and provided a toolkit that supports request, monitoring and revision of tasks [LPP+11]. On the way towards concepts for a reusable dialog framework, this motivated me to rely on the Task State protocol as the exclusive interface to the dialog framework.

3.2 Foundational Work

In traditional architectures for dialog systems, components are structured in a pipelined manner. A speech recognizer recognizes the words the user said, a natural language understanding component assigns a meaning to these words, the dialog manager then determines how utterances fit into the dialog context so far and decides the next system action (e.g. clarifying, confirming, asking for further information, or making a lookup in an information source like a timetable or a flight schedule), and a language generation component chooses the words and phrases used in the response, which is then verbalized by the text-to-speech synthesis [McT04]. Pipeline architectures can typically be found in information access domains, for example in the Ariadne system [Den02], in the LIMSI system [LRGB99], or in the MALIN dialog system, which is shown in figure 3.1. In such kinds of architectures, the dialog manager tends to be a monolithic component that controls everything in the system [OM00]. This makes it hard to build modular systems that comply with the requirements of robotics, such as real-world integration, concurrent activities and mixed initiative. Also for building more advanced conversational systems that approach human-like behavior and account for phenomena such as back-channeling, turn-taking, grounding, interruptions and incremental processing, this type of architecture is not flexible enough [All01].

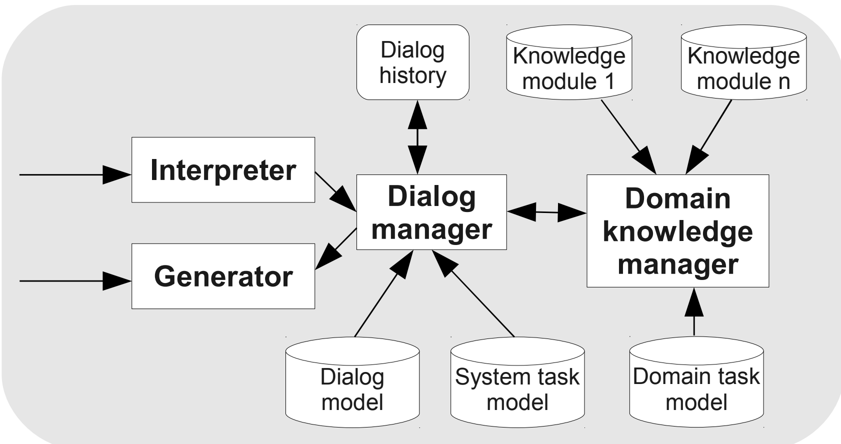


Figure 3.1: The MALIN dialog system architecture: a typical pipeline architecture (after [EJ00]).

Accordingly, research in the spoken dialog system community has been shifting towards more modular, distributed architectures, where dialog management is divided into smaller components. Nowadays, most architectures follow a client-server model with a central component that regulates the interaction between the other components. The logical information flow still mostly follows a strictly sequential pipeline, though. Frameworks and infrastructures for distributed dialog system architectures have been developed, which entails the question of inter-component communication and component interfaces. In the dialog community, this issue has been addressed in two different ways: components communicate either through user-defined interfaces, or through standardized interfaces based on agent communication languages.

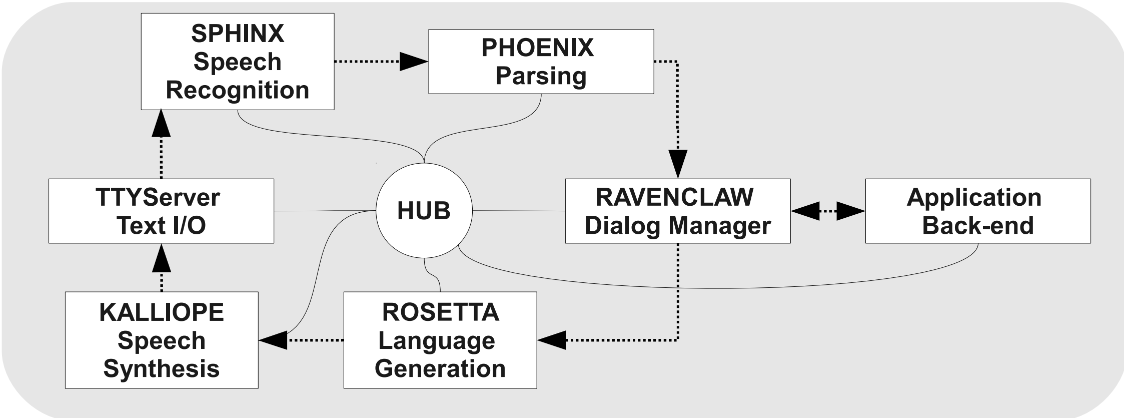


Figure 3.2: The RavenClaw dialog system architecture (after [BR09]), which relies on the Galaxy-II infrastructure. The lines denote the communication connections, whereas the dashed arrows denote the logical information flow. Some components (such as the Date-Time server) have been skipped for the sake of clarity.

The most prominent example of the first approach is the Galaxy-II Communicator architecture, a distributed hub-and-spoke infrastructure for constructing dialog systems [SHL$^+$98]. The RavenClaw framework, which was introduced in section 2.1.2, is based on the Galaxy-II infrastructure. Figure 3.2 shows the architecture of the RavenClaw dialog framework. In this client-server architecture, components are connected via a central hub component and communicate through frame-based messages. The hub component handles all communication between the components, based on sequential rules for routing messages, specifying basically the operation to perform and the resulting input and output variables. Figure 3.3 shows a result frame from the Curious Robot case-study described in section 4, where the Curious Robot scenario was re-implemented based different dialog frameworks, including

```
UserQuery:
    {type: gripquery
     label: apple
     grip: power_grip
     taskstate: completed
    }
```

Figure 3.3: A Ravenclaw result frame from the Curious Robot case study, in which a grip database answers a grip query from the dialog system ("How do you grasp the apple?").

RavenClaw, which relies on Galaxy-II.
From robotics point of view, the Galaxy-II architecture has two drawbacks. First, the semantics of the frames is not specified, i.e. it needs to be defined by the respective component developer. This means more freedom, but also more effort for the developers. Particularly in robotics, where the application back-end consists not of a single database, but of several cooperating components that are implemented by different developers, a common standard would facilitate their integration with the dialog system. Second, the Galaxy-II framework does provide both blocking and non-blocking function calls, but it lacks provisions for monitoring the execution of non-blocking operations. This makes it hard to implement concurrent activities with it. These drawbacks are not specifics of the Galaxy-II framework, but are widely found in similar architectures for dialog systems, e.g. in the CSLU toolkit [SCd+98], the WIT dialog framework [NMY+00], and the iHUB integration framework [RS05].
The above problems were further investigated in the Curious Robot case study. Additionally, several other drawbacks were identified in this case study. For example, as the logical information flow follows a pipeline (cf. figure 3.2) where the back-end actions are triggered by the dialog system, it is hard to realize system task initiative and to integrate real-world events. Also, RavenClaw does not support reusability of interaction episodes. These drawbacks however are rather related to the RavenClaw dialog manager than to the Galaxy-II architecture itself.

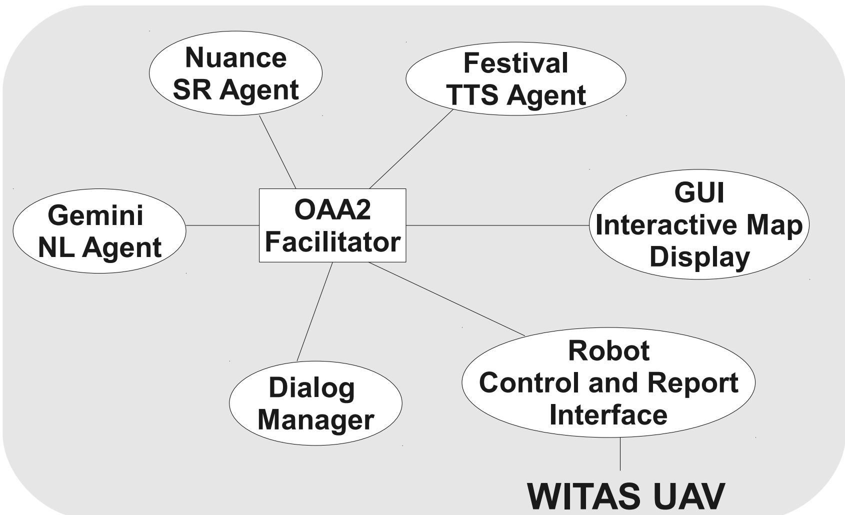


Figure 3.4: The WITAS architecture (after [LBGP01]).

An example of the second approach, where communication between components is based on agent communication languages, is the Open Agent Architecture [MCM99]. The OAA has been designed for distributed agent systems in general, but provides several agents specifically for the purpose of building dialog systems, e.g. for speech recognition,

natural language parsing and generation, text-to-speech synthesis and database queries. Several dialog frameworks rely on it, for instance TrindiKit [TL03], Dipper [BKLO03] and WITAS [LBGP01], whose architecture is shown in figure 3.4.
The OAA is a framework for integrating software agents in a distributed environment. In the context of the OAA, an agent is defined as a software process that provides services or, in the OAA terminology, *capabilities* to other agents, registers these with the framework and is able to "speak" the Interagent Communication Language (ICL). The ICL is a logic-based language similar to Prolog, used to perform queries, execute actions, or exchange information. ICL messages have the form $solvable(Goal, Parameters, Permissions)$. The solvable, which corresponds to an agent capability, specifies what action should be performed. The facilitator forwards it to the agent which is able to solve it. The responsible agent will then try to solve the given goal by unifying the variables contained in the goal expression. For example, in the case study described in section 4, the goal $listObjects(Result)$ was used to request a list of all visible objects from a vision agent. In its response, the vision agent binds the list of object names to the variable $Result$, e.g. $listObjects([apple, [banana, lemon]])$. Besides the goal itself, a message may contain additional control parameters ($Parameters$ and $Permissions$) that specify how the task should be routed, or which agent should perform it.
As for asynchronous processing, the OAA provides more flexibility than the Galaxy-II architecture. All agents are independent processes that run asynchronously and may be written in different programming languages. The OAA can handle asynchronous delivery and triggers, which enables the dialog system to keep track of multiple parallel activities and to react to real-world events. However, similar as with the Galaxy-II architecture, the OAA does not provide a clearly defined component interface, but leaves specification of component communication to the individual developers.

The influential TRIPS architecture, which was introduced in section 2.1.3, uses an alternative approach for communication between different agents. It makes use of the agent communication language KQML, which is based on ideas from speech act theory [FFMM94]. KQML defines a fixed set of *performatives*, that correspond to the illocutionary force of a speech act in natural language. For example, *ask-all* is a directive performative meaning that "A wants to know all B's responses that would make the content true of B", and *tell*

```
tell  :language      Prolog
      :ontology      objects
      :in-reply-to   q1
      :sender        Vision
      :receiver      Dialog
      :content       ''listObjects(([apple, [banana, lemon]]))'')
```

Figure 3.5: Example KQML message from a vision component to the dialog component, providing a list of the currently visible objects.

is an assertive performative meaning that “A states to B that A believes the content to be true” [LF94]. Figure 3.5 shows the above mentioned example message from a vision agent that knows about present objects in the KQML message format.
Like other agent communication languages, KQML adopts a cognitive-state approach to communication, which assumes that a conversation emerges from the agents’ beliefs and intentions. Accordingly, while syntax and semantics of single performatives are detailed in technical specifications, no explicit agreements have been formulated that regulate the sequences in which messages may occur. The correct interpretation of a message, and a meaningful reaction to it, is left to the individual agents. However, as the semantics of the performatives is specified in natural language, the semantics may be interpreted slightly different by different developers. Also, the semantics of a message may depend on the context. For example, an *inform* performative should sometimes be interpreted as a *suggestion* but in another context, as a *command* [EH99]. Thus, interoperability of agents can not be guaranteed.
This shortcoming has long been recognized in the intelligent agent community. As a consequence, *conversation policies* have been suggested that constrain the messages that are sent by specifying shared conventions about what messages may follow each other[1], either in form of graphs (e.g. [LF94, Par96, Kön09]), or in form of constraints to the messages (e.g. [MB95, EH99, HHB99]). There is no consensus on the appropriate form of conversational policies, let alone their semantics [HHB99], and they have not been included in the specifications of the agent communication languages. Those dialog management approaches that base communication on agent communication languages, such as TRIPS, have not adopted the concept of conversation policies, but rely on more specific approaches for the interpretation of agent messages, e.g. based on Allen’s problem solving model [ABF02].

To sum up, in robotics the issue of inter-component communication – and in particular the communication with the back-end – is crucial for the integration of a dialog system with the robotic subsystem, as detailed in section 3.1. In the spoken dialog community, on the other hand, this issue has often been neglected. Asynchronous coordination is provided by some, but not all, architectural frameworks for dialog systems. Dialog systems rarely provide well-defined component interfaces, but often rely on component-specific solutions, e.g. based on user-defined result frames. Alternatively, communication between components in distributed dialog system architectures is often based on agent communication languages, which do specify a message format but lack a definition of valid message sequences. Thus, from robotics point of view, existing dialog systems lack concepts for clearly defined interfaces to the domain level that allows for a detailed information exchange between the dialog system and the robotic subsystem.

1 The concept of conversation policies has influenced my idea of Interaction Patterns (cf. Chapter 3.4.4) that determine which dialog acts may follow each other.

3.3 The Task State Protocol

Robots that are to assist humans in their real-world environments must be equipped with a variety of capabilities, ranging from perceiving the environment, detecting and recognizing humans, navigation in dynamic environments, to localizing and manipulating objects. These activities are initiated, and can be supported by, interaction with the human, where human and robot exchange information about their common environment and collaborate on joint tasks in a mixed-initiative interaction that goes beyond command-control style. From the human's point of view, the dialog system provides a gateway to the robot's capabilities. It takes the human's commands and forwards it to the components that carry it out, keeps the human informed about the robot's current state, and reports about execution success while – in more complex settings – the human can at any time stop or revise the robot's actions. On the other hand, it enables the robot to verify the given command, or to ask for missing information. As argued before, from a software engineering point of view, this requires a close collaboration between the dialog system and the components that carry out the robot's activities. To address this issue, a general communication model is used: the Task State Protocol. It provides a fine-grained coordination mechanism for robot activities and, at the same time, serves as a well-defined component interface to the dialog system.

The basic concept is that components execute and request *tasks*, acting as *servers* and *clients* respectively. A task consists of its *specification* (which is relevant for execution) and the *state* (which is relevant for coordination) it is in. While the task specification depends on the type of the task, the set of possible states are the same for all tasks. The task life cycle can be described by the finite-state machine depicted in figure 3.6: Normally, a task gets *initiated*, *accepted* and finally is *completed*. Alternatively, it might *fail*, or be *rejected* right away. During execution, it might already deliver *intermediate results*, or be *updated*

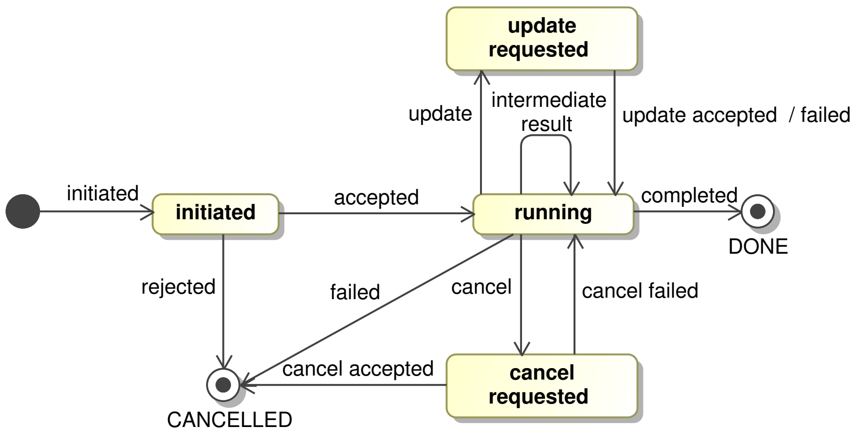


Figure 3.6: The task state machine that describes the general life cycle of tasks.

or *canceled* (which again might fail). Table 3.1 lists the semantics of those task state updates. The updates cause event notifications that are delivered to the participating components, together with the current task specification. However, not every task must support all state changes.
The concept of task-based coordination is not new: It has been identified as a common design pattern for coordination in robotics, and variations of it are applied in several robot architectures [LPP+11]. However, its application to, and benefit for, human-robot interaction has not been investigated in the robotics community yet.
Technically, tasks are represented as XML document that contain a STATUS element, indicating their current state (cf. figure 3.7). Events are delivered asynchronously through an event bus provided by the middleware. The Task State Protocol has so far been implemented based both on the robotic middlewares XCF [FW07] and RSB [WW11] (but can in principle realized with any middleware that supports asynchronous communication). Lütkebohle has developed a toolkit that supports task management, both for server's and for client's side [LPP+11]. It offers functionality for requesting tasks and updating their states (and possibly their specification as well) via a task object that encapsulates the detailed XML handling. Most importantly, the toolkit makes sure that the update operations comply to the state machine shown in figure 3.6. It also takes on error management, e.g. by detecting and recovering concurrent updates or delayed delivery.

The Task State Protocol in the form as described here evolved within the Home-Tour scenario, where a mobile robot assistant has to become acquainted with human's living environment by interacting with a human during a guided tour (see Chapter 6.1). In this context, a very first version of the Task State Protocol has been used for modeling the communication between the dialog system and a room representation component.

Task state	***Update operation***	***Semantics***
initial	initiated	The client initiates the task.
initiated	accepted	The server begins execution.
	rejected	The task will not be executed.
running	intermediate result	The server has updated the task specification.
	completed	The server has completed the task.
	failed	The task could not be completed.
	update	The client has updated the task specification.
	cancel	The client requests termination of execution.
update requested	update accepted	The server conforms to the updated task specification.
	update failed	The server continues execution with the previous task specification.
cancel requested	cancel accepted	The server stops execution.
	cancel failed	The execution will be continued.

Table 3.1: The semantics of the task state updates.

```
<GRASP>
  <STATUS value="initiated"/>
   <Region varianceFirstMajorAxis="434" varianceSecondMajorAxis="433" pixelCount="4384">
      <coord ref="image" kind="relative" x="158" y="67" width="0.4" height="0.8"/>
      <Object detectorLabel="apple">
          <Grip type="TwoFingerSpecial"/>
      </Object>
   </Region>
</GRASP>
```

Figure 3.7: Example task specification for a grasp operation.

Representations were based on laser-range data. The robot has to turn by 360°to acquire these. As this normally takes some time, it was desired that the robot acknowledges the execution at the beginning of the learning process to provide feedback about its internal action state ("I will have a look at it"). For this purpose, a basic version of the task life cycle was used, comprising the events *initiated*, *accepted* (marking the execution begin) or *rejected* (when the hardware was otherwise busy), and *completed*.

The concept was further refined within the Curious Robot scenario, an interactive object learning and manipulation scenario where a humanoid robot learns labels of objects and how to grasp them, assisted by a human tutor (see Chapter 6.2). In particular, the robot's grasping operation provided an interesting use case for coordinating more complex actions: The human can intervene at any time, to stop or to correct an on-going grasping operation. Moreover, during an on-going grasping action, the human can bring up new topics, e.g. asking the system about information it has, which requires to coordinate multiple tasks in parallel. To meet these demands, the Task State Protocol was iteratively extended. The final version shown in figure 3.6 has proven to be detailed enough to support the desired functionality and, at the same time, general enough to be applicable to a variety of use cases in very diverse scenarios. Its role as a general coordination mechanism has been investigated [LPP$^+$11, Lüt11]. Accordingly, in the Curious Robot system not only the communication between the dialog system and the back-end, but the interplay of all components relies on the concept of task states.

Based on the experiences with the Home-Tour and the Curious Robot scenario, which were both implemented with the then used Sunshine dialog system, concepts for the reusable and customizable PaMini (<u>Pa</u>ttern-based <u>Mi</u>xed I<u>ni</u>tiative) dialog framework were developed. As a consequence thereof, the role of the Task State Protocol as a generic interface became more important. From the dialog system point of view, it establishes a uniform interface to the back-end – and in particular to action execution – which allows the dialog system to treat all tasks in a uniform manner. From the back-end point of view, the Task State Protocol represents a well-defined component interface to the dialog system. It provides architectural guidelines for component developers, thus facilitating integration. The architecture it entails is schematically shown in figure 3.8: The dialog manager exchanges information with the robot back-end – consisting of several task servers and clients – based on the Task State Protocol vial an asynchronous event bus. Note that the back-end components may (and often do) act as server and client at the same time. Also, the back-end will typically include additional components that do not participate in the task communication with the dialog system. They are not shown in the figure.

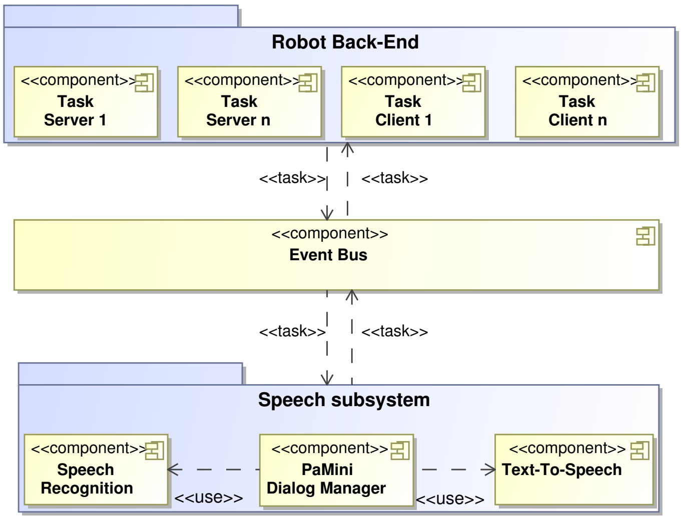


Figure 3.8: Schematic architecture. The PaMini dialog manager exchanges information with the robot back-end based on the Task State Protocol via an asynchronous event bus.

3.4 Advanced HRI Based on the Task State Protocol

In the following, several interaction examples from the Curious Robot scenario are described which illustrate the interplay of the dialog system and the robot system. Each example emphasizes a particular aspect crucial to human-robot-interaction: Mixed task initiative, the integration of action execution and interaction, multitasking and interactive learning. Examples are presented as UML sequence diagrams which show how components operate with another and the sequence of messages they exchange. Please note that technically the events are not sent directly to the individual components as the diagrams might suggest, but delivered by an event bus to the components that are registered to the respective event type. However, for the purpose of clarity, the event bus is not shown in the diagrams. Note also that technically the communication with the speech recognition and text-to-speech component is modeled as tasks as well, which for the purpose of clarity is not reflected in the diagrams.

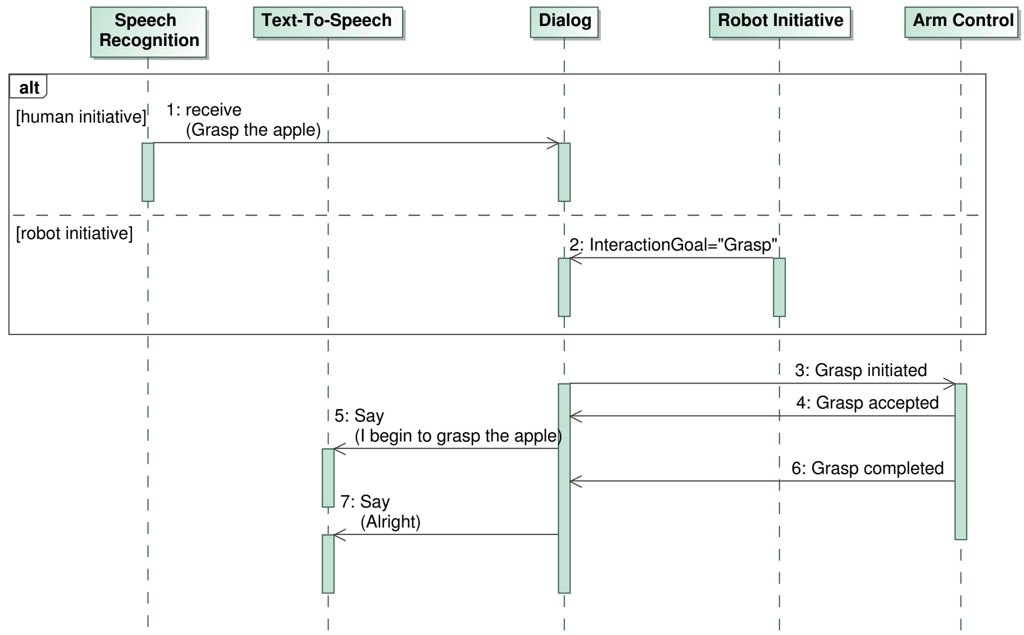


Figure 3.9: Mixed task initiative: Grasping can be initiated either on the human's or on the robot's initiative.

3.4.1 Use Case 1: Realizing Mixed Task Initiative

The first example illustrates a grasping action that can be proposed either by the human or by the robot. The diagram shown in figure 3.9 distinguishes between these two alternatives. In the first case, the grasp command is given by the human, whereas in the second case, the robot is grasping on its own initiative. Having received the command, either by the speech recognition in case of human initiative (1) or by an initiative planning component (2), the dialog initiates the appropriate task (3). The arm control accepts (4) and completes (6) it, which is verbalized by the dialog system (5 and 7).

The example serves to illustrate two aspects. First, the motor control server is agnostic about how the task came about, i.e. whether it was initiated on the human's or on the robot's initiative. The further process flow is the same. Second, the decision about the robot's actions (in this case, its grasping action) are not taken by the dialog system, as it is the case with dialog systems for traditional domains (cf. the discussion in chapter 4). Instead, the proposals for actions come from the back-end. Thus, robot task initiative can be realized as a reaction to real-world events (e.g. that the apple was newly detected in the scene), rather than following a prestructured dialog flow.

The diagram slightly simplifies component interaction. Unlike as depicted, the current interaction goal *Grasp* was modeled not as a simple event message, but rather as a task request which is either accepted or rejected by the dialog system, depending on whether the current dialog situation allows for interjection from the robot or not.

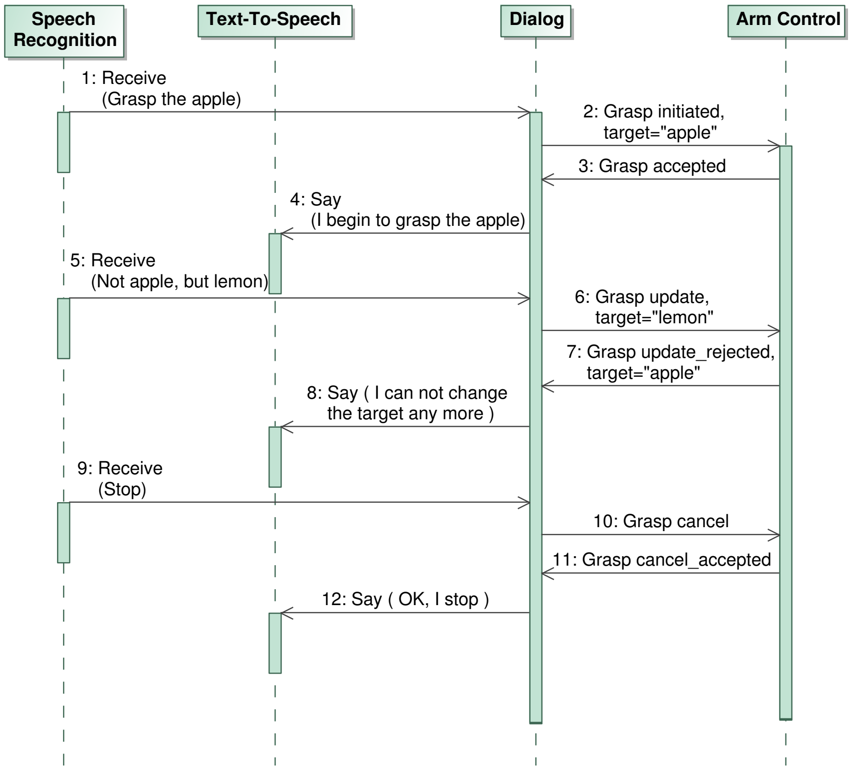


Figure 3.10: Integration of action execution and interaction: Grasping with user corrections.

3.4.2 Use Case 2: Integrating Action Execution and Interaction

The second example addresses the interaction between the dialog system and the motor control. Figure 3.10 describes an interaction sequence in which the human gives the order to grasp the apple (1), whereupon the dialog initiates the appropriate task (2). The arm control accepts (3) and begins execution, which is announced by the dialog system (4). During grasping, the human attempts to correct the target of grasping (5). The dialog system forwards the requested correction to the arm control by modifying the task specification and setting the task state to *update* (6). However, the arm control is not capable of updating the operation and rejects the update (7), causing a notification generated by the dialog system (8). The human then gives the order to stop execution (9). Again, the dialog system forwards the request to the arm control via the *cancel* state (10). The arm control accepts canceling (11), and the dialog system generates the appropriate verbalization (12).

This example emphasizes the tight integration of interaction and action execution. With the *update* and *cancel* events, the Task State Protocol provides mechanisms to repeatedly modify ongoing tasks. Conversely, the resulting state updates (such as *cancel accepted*)

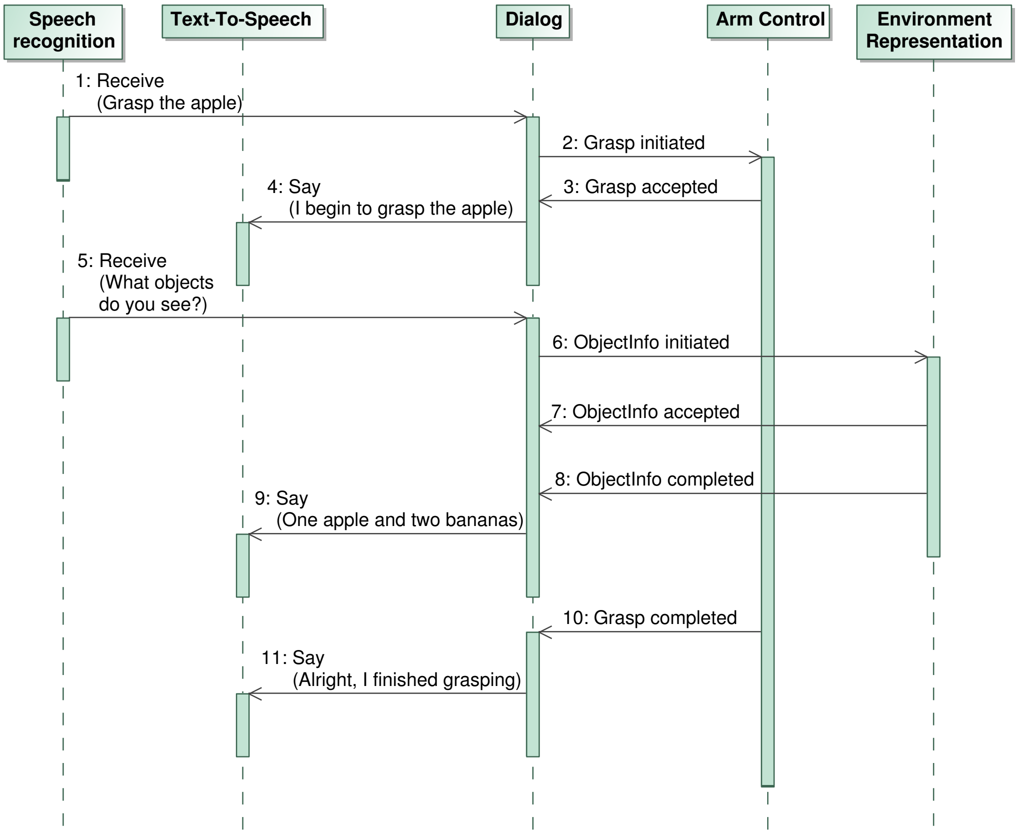


Figure 3.11: Multitasking: User requests information during grasping.

cause event notifications that enable the dialog system to generate feedback on the internal system state. This includes also the case where state updates can not be realized, which is indicated by event notifications such as *update rejected*.

3.4.3 Use Case 3: Multitasking

In the third example, two tasks are executed in parallel. During an ongoing grasping action (1-4), the human requests information about the objects in the scene (5). This causes a task for the environment representation component to be initiated (5). The environment representation accepts the task (6), adds the requested information to the task specification and completes it (7). The information is verbalized by the dialog system (8). Next, the grasp task is completed and acknowledged as well (9, 10).
The example demonstrates how asynchronous event delivery enables multitasking. Having initiated the task for grasping, the dialog system remains responsive to new commands. New tasks can be initiated, even though completion of running tasks is still pending. From an interaction point of view, this enables to manage multiple open topics at a time within

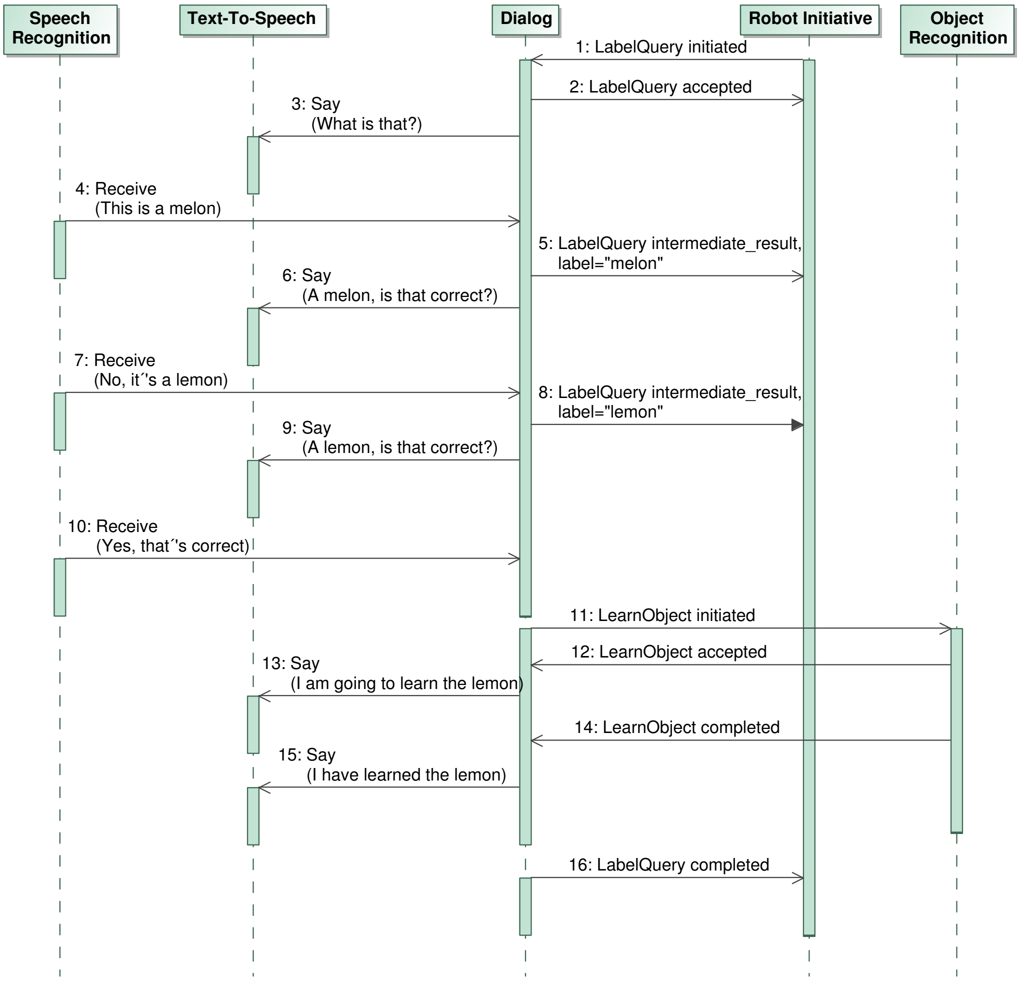


Figure 3.12: Interactive learning. Label learning with user corrections.

a conversation, and to switch between the topics.

3.4.4 Use case 4: Enabling Interactive Learning

The final example shows how the Task State Protocol can be applied to realized interactive learning. More specifically, the robot asks the human for an object label that is unknown to it. First, an initiative planning component requests a LabelQuery task from the dialog system (1). The dialog system accepts the task (2) and, accordingly, verbalizes the question about the object label (3). As the human answers the question (4), the dialog system already publishes the received information (5) before asking the human for final confirmation (6), which gives the human the opportunity to correct the misunderstood label (7). Again, the dialog system already publishes the label (8) before asking for confirmation (9). This time, the human confirms the label (10), whereupon the dialog system initiate a LearnObject task for the object recognizer (11). The LearnObject task is accepted (12) and completed (14). Both state updates are verbalized by the dialog system (13, 15). On notification that the LearnObject task has been completed, the dialog system completes also the LabelQuery task.

Based on this example, several aspects can be discussed. First, the Task State Protocol allows the dialog system to gather information from the human and to transfer it to the responsible system components. With the *corrected* state, the information can be revised and corrected repeatedly. Moreover, with the *intermediate result* transition the information may be submitted incrementally, or preliminary information may be already published although it is still under negotiation. Second, as in the previous example, two interleaving tasks (LabelQuery and LearnObject) are executed. However, unlike before, the tasks are logically related: the dialog system can not complete the LabelQuery task until the LearnObject task has been completed. Third, note that in the present example, the dialog system acts both as task server (for the LabelQuery task) and as task client (for the LearnObject task) at the same time, which assigns it a coordinating role within the overall system.

In the Curious Robot system, there was no component that actually did react to the unconfirmed label. However, one could easily think of cases in which preliminary publication of unconfirmed information would be useful. For example, slow-going actions such as manipulation or navigation tasks could already be prepared even though the precise goal specification is still preliminary.

The previous chapter has taken an external view on the proposed approach to dialog modeling. In contrast, this chapter will provide insights into its internal functioning. Section 3.5 establishes the link between the internal dialog model and the process of dialog design, and it introduces the twofold function of the proposed Interaction Patterns, which serve both as internal dialog model and as application programming interface (API) for dialog designers. Section 3.6 reviews foundational work that has influenced the concept of Interaction Patterns, both from the field of descriptive dialog modeling and from the field of dialog system API design. Section 3.7 defines the Interaction Patterns and their features. Their function as internal dialog model is discussed in section 3.8, and their function as dialog system API is discussed in section 3.9. Section 3.10 gives an overview of the existing Interaction Patterns and their development over time.

3.5 Motivation

The previous chapter described how the dialog system communicates with other components and the role it takes within the overall architecture. However, it left open a number of questions: How is the appropriate task operation determined from a user instruction? How are incoming task events verbalized? How are modifications of a task specified? More general, how does the dialog manager fulfill its two main responsibilities, which are (i) to determine how an utterance fits into the dialog context and (ii) to decide the system's next action [McT04]? These issues largely depend on the underlying dialog model. As stated in section 2.4.1, the presented work favors a descriptive approach for *local* dialog modeling.

An aspect that is strongly influenced by the chosen approach for dialog modeling is the process of dialog design: Which steps are required to build a spoken dialog application, and what amount of effort and expertise does this take? One reason why a descriptive dialog model was favored is that it provides better understandability of the possible dialogs than mental-state approaches, and thus has the potential to facilitate the design process, even for non-experts. In this context, Bohus and Rudnicky point out several desirable characteristics for dialog frameworks and their API [BR09][1]:

- **Task independence**: The framework should establish a clear separation between the domain-specific aspects of dialog logic and domain-independent dialog control principles, so that dialog designers need to specify exclusively the domain-specific aspects.
- **Flexibility**: The framework should support a wide range of application domains and interaction styles.
- **Modularity**: The single functions (e.g. dialog planning, in- and output processing, error handling) should be encapsulated in subcomponents with well-defined interfaces.
- **Reusability**: Solutions should be reusable across applications[2].
- **Scalability**: The framework should support the development of practical real-world spoken language applications. Since these are typically developed iteratively, this means that the framework should support robust expansion of applications.

In summary, these desirable features may serve as design principles for the API of a dialog framework. They have, amongst other considerations, guided the adopted approach. We will return to these issues and check their realization in section 3.8.2.

The present chapter will introduce the concept of generic Interaction Patterns: a formalization of recurring conversational structures and system actions at an abstract level. Their purpose is twofold: From dialog management perspective they (i) provide context for input interpretation and specify the system's response. From dialog designer's perspective they (ii) provide configurable building blocks of interaction that can be combined in a flexible manner. The link to the domain level is established through the previously described Task State Protocol.

Like the Task State Protocol, the concept of Interaction Patterns has been developed

1 In addition, they mention *open source* and *transparency* with respect to logging.

2 However, the RavenClaw approach (which is the one proposed by Bohus and Rudnicky) does not support reusability very efficiently itself. See section 4 for discussion.

based on the experience with the Home-Tour scenario and the Curious Robot scenario (cf. chapters 6.1 and 6.2), both of which were implemented with the then used Sunshine dialog system (cf. section 5.1). They provided plenty of use cases for identifying recurring patterns in human-robot interaction, both on conversational and on task level. The observed recurring structures were then generalized and implemented as Interaction Patterns in the PaMini framework. Since then, many further scenarios have been built based on PaMini's Interaction Patterns, most notably the Curious Flobi scenario, which served as the main testbed and proof-of-concept for the approach (cf. chapter 7).

3.6 Foundational Work

This section reviews foundational work that has influenced the concept of Interaction Patterns, both from the field of descriptive dialog modeling and from the field of dialog system API design. As abstraction of dialog is a prerequisite for reusability, section 3.6.1 discusses formalisms that describe the typical course of conversation at a high level. Section 3.6.2 identifies techniques to achieve an easy-to-use API based on a number of example systems that explicitly address rapid-prototyping of dialog applications.

3.6.1 The Internal View: Descriptive Dialog Models

Linguistic approaches to dialog modeling can be divided into plan-based, or cognitive-state approaches, and descriptive approaches[FE07]. While the approaches of the first type model interaction partners' beliefs, desires and intentions, the approaches of the latter type focus on the interaction partners' resulting behavior. Within the tradition of descriptive

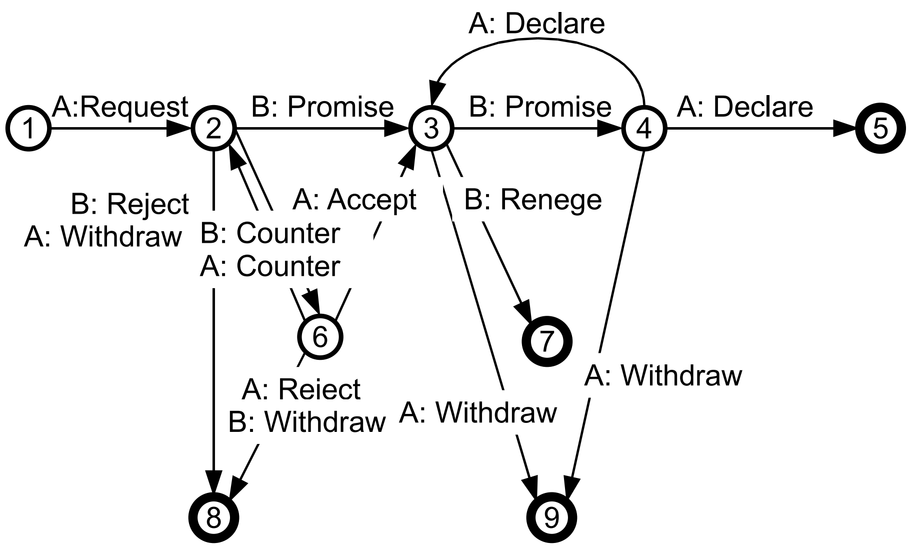


Figure 3.13: Winograd's conversation for action (after [Win86]).

dialog modeling, there is in particular the notion of **dialog games** [LM77, Man88, Hul00]. The underlying assumption is that conversation is carried out towards a goal, and a dialog game, then, describes conventions about typical conversations and how goals can be achieved, specified in form of rules. For example, in their early work, Levin and Moore have identified dialog games for helping, action-seeking, information-seeking, information-probing, instructing and griping, based on a wide variety of dialogs including transcripts of a lunar mission, radio talk shows and teaching interactions [LM77].
A related concept within the field of conversational analysis is the concept of **adjacency pairs** [SS73, Lev83]. Adjacency pairs consist of two turns that are functionally related such that the first turn restricts the type of the second turn, i.e. the first turn determines its preferred, or expected, follow-up. For example, the resonse to a greeting is a greeting, or an apology requires an acceptance or a rejection. Between the two turns of an adjacency pair, other pairs may occur as insertion sequences to form larger segments. This concept has inspired the design of dialog strategies in practical spoken dialog systems. Admissible adjacency pairs and possible insertion sequences serve as a dialog model and are specified in form of dialog grammars (e.g. [Bri02, KGK+09]), as attachment of a dialog move in the WITAS system [LGBP02] (cf. section 2.2.2), or learned from annotated corpora (e.g. [BPH+09]).
In the field of multiagent systems, the above mentioned cognitive-state approaches are predominant, but a number of researchers advocate for more explicit, descriptive approaches of modeling conversations in form of **conversation policies** [LF94, MB95, Par96, EH99, HHB99, Kön09]) (cf. also the discussion in the context of inter-component communication in section 3.2). The basic idea dates back to Winograd's work in the field of computer supported collaborative work where the commitments that individuals take within their daily activities and methods for coordinating them are modeled [Win86]. Figure 3.13 shows a simple "conversation for action", in which A makes a request to B. B can either accept, decline or make a counter-offer with alternative conditions, and so on. The multiagent community has adopted this concept for modeling conversations between software agents rather than between human individuals. The software agents communicate through agent communication languages (e.g. KQML [FFMM94] or FIPA [ON98]) which are typically designed in the style of human language and include terms like *asking* or *apologizing*. Conversations policies determine admissible sequences of agent communication messages, often in the form of finite-state automata, similar as the one shown in figure 3.13. In contrast to the above linguistic formalisms, conversation policies are intended to generate conversations in technical systems rather than to analyze human-human conversations. For this reason, and because they provide a comprehensible graphical representation, they strongly influenced my concept of Interaction Patterns.

Common to the described approaches is that they take a functional view on language, which is inspired by **speech act theory** [Aus62, Sea69]. Speech act theory has brought up the view of utterances as actions: Utterances – or speech acts – can change the mental and interactional state of interaction partners, in a similar way like physical actions change the state of the world. The described approaches focus on the illocutionary level of speech acts and the separation between the illocutionary force (which specifies the type of action, e.g. *rejecting a request*) and the propositional content (which specifies the details of the actions). In spoken dialog systems, this distinction can be used to separate the interactional function of an utterance from its specific wording and thus to make an abstraction over utterances. In fact, most dialog systems make abstractions over utterances, which is however often referred to as *dialog acts*. In the following, this term will be adopted when it is not the speech act theory in the strict sense, but rather this separation in a more technical sense which is referred to.

3.6.2 Dialog System APIs

Virtually all approaches to dialog systems claim to be easy configurable in some way or another. One of the first approaches that explicitly addressed the issue of API usability and rapid development of dialog applications was the CSLU toolkit [SNC+96, SCd+98] which provides a graphical editor and a toolkit environment for creating finite-state based dialog systems (cf. section 2.1.2).
Another **toolkit approach** represents the WIT toolkit for building spoken dialog systems [NMY+00]. Also the WIT toolkit pursues a whole systems approach and provides an environment for integrating speech recognition, language understanding and generation, and speech output. For each of the components, a domain-specific knowledge source needs to be defined. Based on a user-defined semantic frame specification of the domain, an integrated parsing and discourse processing method plans the output using a unification grammar [NMH+99]. The WIT toolkit relies on a more sophisticated dialog model than the CSLU toolkit, but will presumably requires more expertise from the application developer.

In a similar way, most approaches achieve easy reconfigurability by separation of domain-specific and domain-independent knowledge. Some approaches emphasize the definition of **task models**, while others focus of the identification on **generic dialog strategies**. Examples for the first category are Collagen with its Recipes (cf. section 2.1.3) and WITAS with its Activity Model (cf. section 2.2.2), while RavenClaw (cf. section2.1.2) focuses on the identification of domain-independent dialog strategies for error handling and grounding.

Another approach that focuses on describing domain-independent dialog strategies has been proposed in the context of the ARIADNE dialog system [Den02]. It relies on the slot-filling approach and uses an explicit dialog state (similar as the information state approach described in section 2.1.3). To develop a spoken dialog application with it, it is required to specify a number of domain specific knowledge sources, most notably an ontology and a set of service descriptions that specifies for each back-end application what kind of information is necessary to invoke that service. During interaction, the dialog state keeps track of the goals that are compatible with the information gathered so far. To control the dialog, the system relies on generic dialog processing algorithms, which are also called Interaction Patterns. These are procedures that entail sequences of utterances. Four types of Interaction Patterns are incorporated into the system: The Question pattern requests information from the user, the Undo pattern removes information, the Correction corrects an information, and the State pattern handles help requests. Denecke's Interaction Patterns are specified in a declarative way, and their execution is based on constraint logics. Depending on the dialog state and the compatible goals, the system instantiates the appropriate Interaction Patterns. Similar as the Interaction Patterns proposed in the present work, the shape of the patterns – i.e. the specific sequence of utterances they include – varies and is determined as the dialog develops. Another commonality between the different concepts of Interaction Patterns is that they both model not only sequences of utterances, but also system operations. However, while Denecke's Interaction Patterns operate purely at the information level by updating the dialog information state, the Interaction Patterns proposed here operate additionally at the domain level by updating back-end tasks through the Task State Protocol. Also, Denecke's Interaction Patterns do not serve as an API specification for dialog designers (which is one of the basic functions they take up in the present approach), but can rather be seen as built-in system capabilities that are triggered automatically as appropriate.

A similar approach has been proposed by Bui and colleagues [BRM04], also in the domain of slot-filling applications. In their approach, the domain is modeled as a set of relational database tables. The dialog model consists of a set of interconnected Generic Dialog Nodes (GDN), each of which refers a column in the database. The GDN are configured by the application developer with a grammar to interpret the user input and the prompts the system will say. Based on this configuration, each GDN performs a simple interaction with the purpose to obtain a value for the associated attribute from the user. The local dialog flow management is handled by a single GDN. Each GDN can handle five situations: OK, Repeat, Help Request, No Input and No Match. More general strategies determine the global dialog flow management, e.g. how to deal with inconsistencies. The proposed approach is embedded into a process model for developing spoken dialog applications, which includes conducting WOz studies, as well as internal and external field studies.

Gandhe and colleagues have introduced an approach to rapidly developing dialog capabilities for virtual characters based on the Information State approach [GDR+08]. As the ones described above, this approach operates on a domain specification describing the objects and characters of the domain, as well as their attributes and possible values, or their goals. This authoring process is supported by a graphical user interface. From the domain description, the dialog acts that may occur during interaction are generated automatically. For example, for a specification of an object with certain attributes and possible values, an associated *assert* dialog act is generated. During interaction, the dialog manager updates the information state according to the occurring dialog acts, and generates the content of the response. The agent's conversational obligations – i.e., the sequences of dialog acts – and the rules according to which the information state is

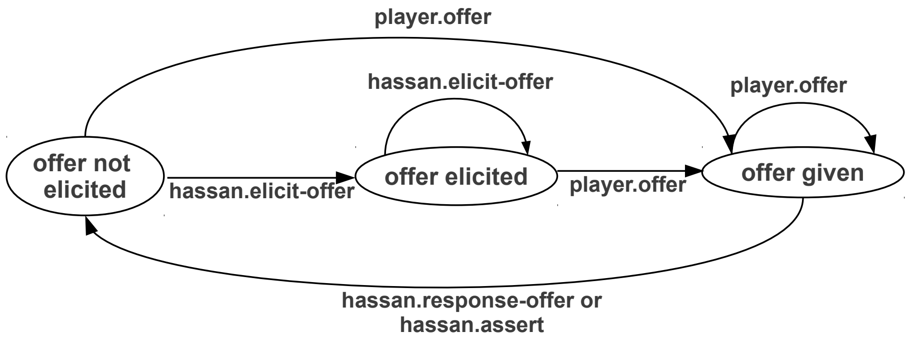


Figure 3.14: Finite state machine modeling the agent Hassan's conversational obligations associated with an offer subdialog (after [GDR+08]). Not shown are the conditions and updates to the information state.

updated[1] are implemented as finite state machines. Figure 3.14 shows the finite state machine for an offer subdialog. Thus, the final state machines model the local discourse coherence, while the global coherence is determined by the system's information state. In this respect, they are similar to the Interaction Patterns proposed in the present work. Also, both concepts are modeled as a kind of finite state machine. Gandhe's obligation descriptions, however, model exclusively the dialog act sequences, but not the associated system actions (such as updates of the information state).

3.7 Generic Interaction Patterns

As outlined above, the proposed Interaction Patterns describe recurring subdialogs, such as making a proposal, or requesting information. They mainly draw on ideas from conversation policies in multiagent systems and from design patterns in software engineering. From the first, they take up the idea of representing admissible sequences of dialog acts as a finite-state machine. From the latter, they take up the idea of providing reusable solutions for recurring problems to developers. The result is an executable graphical model that considers both the conversation level and the system level and has a twofold function: it serves both as internal dialog model and as API for developers. Interaction Patterns can be defined as follows:

Interaction Pattern An Interaction Pattern is a sequence of human dialog acts, robot di-

1 In the original implementation of the Information State approach, the update rules are specified as Prolog rules [TL03].

alog acts[1] and system actions. It can be specified as an 8-tuple $< H, R, T, S, s_0, F, A, \Sigma, \Lambda, T >$, consisting of

- a set of human dialog acts H and a set of robot dialog acts R, e.g. *H.request* or *R.assert*;
- a set of incoming task events T, e.g. *accepted* or *failed*;
- a set of states S representing the interaction state, e.g. *await_confirmation*;
- an initial state $s_0 \subset S$;
- a set of final states $F \subset S$
- a set of system actions A the dialog manager performs, e.g. initiating or updating a task or reset interaction;
- an input alphabet $\Sigma \subset (H \cup T)$;
- an output alphabet $\Lambda \subset R$;
- a transition function $T : S \times \Sigma^* \longrightarrow S \times A^* \times \Lambda^*$.

In the following, Interaction Patterns will not be characterized by explicitly specifying the tuple $< H, R, T, S, s_0, F, A, \Sigma, \Lambda, T >$, but implicitly through their graphical representation, as illustrated in figure 3.15. The graphical representation shows that the structure of Interaction Patterns has been inspired by finite-state transducers with in- and output [Mea55]: User input and incoming task events are modeled as input, while robot dialog acts are modeled as output of a transition. Contrary to traditional finite-state transducers, Interaction Patterns additionally perform actions within the states.

With the above definition, Interaction Patterns are specified at a generic level, based on dialog acts. Thus, to describe a specific interaction, additional information needs to be specified:

Interaction Pattern configuration An Interaction Pattern configuration is associated with an Interaction Pattern and specifies the surface form of the dialog acts as well as possibly required parameters for the system actions.

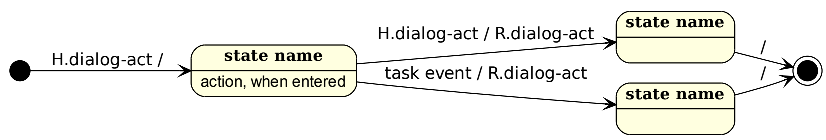


Figure 3.15: Schematic graphical representation of an Interaction Pattern.

1 This does not mean that the application of the approach is exclusively restricted to robotics. It can be (and has been) used to model interactions with all kinds of technical systems, for example virtual agents or non-embodied systems. Similarly, the interaction partner denoted with *human* could also be a technical system itself.

Given the above definitions of an Interaction Pattern and its configuration, it makes sense to distinguish between the generic pattern definition and its specific manifestation:

Interaction Pattern instance An Interaction Pattern instance[1] is an Interaction Pattern together with its Interaction Pattern configuration.

Finally, a specific dialog with specific instances of Interaction Patterns will require a variable context where dynamic information is stored:

Variable context Associated with each Interaction Pattern instance is a variable context that is defined through a set of variables and their values.

The intended function of Interaction Patterns as an instrument for dialog management that, on the one hand, restricts the context to facilitate interpretation, but at the same time allows for greatest possible flexibility, implies the following desirable properties.

Of course, a dialog may require several Interaction Patterns of the same type in different variations, for example different types of instructions or suggestions. Thus, it is indispensable that multiple instances can be created from an Interaction Pattern:

1. Duplicability For a specific Interaction Pattern, multiple Interaction Pattern Instances with differing Interaction Pattern configurations may exist.

Another desirable feature is that the user should be enabled to cancel an Interaction Pattern instance:

2. Cancelability Interaction Patterns may be cancelled externally.

A distinction must be made between cancellation and failure of an Interaction Pattern: while failure cases, such as rejection of a system task, are considered in the structure of Interaction Patterns and thus can be considered as normal termination, a cancellation is triggered externally and constitutes an abnormal termination. A cancellation of an Interaction Pattern may entail the cancellation of further Interaction Patterns, for example if the user request a system restart.

Further, to enable topic shifts and multitasking, it is necessary to allow several Interaction Patterns to be active at the same time.

3. Interleavability Interaction Patterns may be interleaved.

1 The term *Interaction Pattern instance* will hereafter also be referred to as *configured Interaction Pattern*, or simply as *Interaction Pattern* if the distinction is not important or clear from the context.

Note that *interleaving* Interaction Patterns provides even more flexibility than *nested* Interaction Patterns. As an example, consider Interaction Pattern P1 with states P1S1, P1S2 and P1S3 and Interaction Pattern P2 with states P2S1, P2S2 and P2S3. A state sequence admitted by interleaving, but not by nesting, would then be P1S1, P2S1, P1S2, P2S2, P1S3, P2S3. Admitting such interleaving Interaction Patterns entails a tree structure of discourse, where each branch can be expanded at any time[1].
Together with duplicability follows in particular that interleaving of two Interaction Pattern instances of the same type is possible. In contrast to cancelability, which enables the user to cancel Interaction Patterns *explicitly*, interleaving enables the user to cancel *implicitly*, e.g. by ignoring a question and going on with another topic.

To ensure that Interaction Patterns can be combined without side effects, and that new patterns can be added safely, another desirable feature of Interaction Patterns is that they should be connectible in series in a stateless manner:

4. Self-Containedness Interaction Patterns are completely self-contained.

This means in particular that Interaction Patterns do not require a specific dialog context before they can be triggered: all necessary variables and parameters are either obtained during execution of the Interaction Pattern, or specified by the Interaction Pattern Configuration beforehand.

Finally, to ensure that the information obtained by an Interaction Pattern can be made permanent, the variable context must be accessible to all Interaction Patterns:

5. Common context All Interaction Patterns operate at a common variable context, and each Interaction Pattern can modify the variable context.

Together with interleavability, this implies in particular that inner Interaction Patterns can alter the variable context of the outer Interaction Patterns, and that the update persists even after the inner pattern has terminated, which allows e.g. for inserted corrections (though corrections are mostly incorporated directly in the respective Interaction Pattern).

Figure 3.16 shows as an example of the Human Simple Action Request. It describes an action request initiated by the human that, in contrast to the Human Cancellable Action Request, cannot be canceled once execution has begun. The normal course of events is that the human requests the action to be executed, the dialog manager initiates the system task, the respective system component accepts execution whereupon the dialog manager asserts execution. Finally, the task is completed, which is again acknowledged by the dialog manager. Additionally, there are transitions for the cases in which the system task

1 However, to handle cases where this flexibility is not appropriate, the PaMini framework offers possibilities to restrict interleaving.

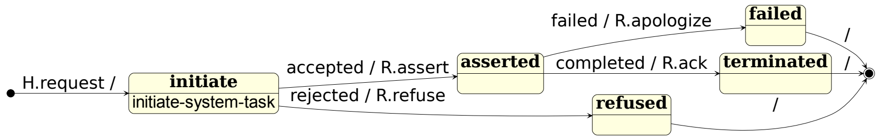


Figure 3.16: Human Simple Action Request.

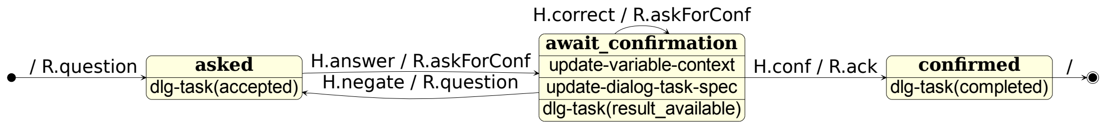


Figure 3.17: Robot Information Request With Explicit Confirmation.

is rejected or fails.
In contrast, the Robot Information Request with Explicit Confirmation is initiated by the robot. Technically, this means that a dialog task is requested from the dialog manager. Dialog situation permitting, the dialog manager will ask for the desired information and accept the dialog task. Once the human has provided the answer, the robot will ask for confirmation. Even if the information has not been confirmed yet, the dialog manager already updates the variable context as well as the dialog task and its specification to indicate that a first result is available. If the human confirms, the dialog manager acknowledges and completes the dialog task. Alternatively, the human may negate that the information to be confirmed is correct, or correct it directly.
It is noticeable that the Human Simple Action Request, which is initiated by the human, is mainly driven by the incoming task events, while the Robot Information Request with Explicit Confirmation, which is initiated by the robot, is driven by the human's utterances, which the dialog manager then translates into outgoing task operations. Also, both patterns do not make use of the full Task State Protocol. For example, the Human Cancellable Action Request does not allow cancellation of the on-going task, and it does not allow to update the task specification during execution.

Table 3.2 lists all identified Interaction Patterns. They can be divided into six topics: Patterns that deal with actions, patterns that deal with information negotiation, patterns that refer to present objects, patterns that have an interactional functions, general-purpose patterns and clarification patterns. For each topic a range of patterns is available, differing e.g. in initiative (e.g. an action may be initiated either by the human or by the robot itself), in the communication structure (e.g. information may be confirmed implicitly or explicitly), or in the task communication (e.g. obtained information may be acknowledged by the respective system component or not). Among these variations, developers can choose among the pattern variations which fit best with their needs. A graphical representation and short description of all patterns can be found in Appendix B.

Technically, Interaction Patterns are implemented as statecharts [Har87] using the the Apache Commons SCXML engine [Apa07]. The PaMini framework embeds them in a dialog management process that triggers patterns, manages and interleaves them, and makes meta-decisions, e.g. about closing and opening the interaction. These, rather internal, aspects are detailed in section 3.8. Additionally, they are wrapped in a Java API that allows developers to configure patterns and to assemble them into an interaction scenarios. These external aspects will be discussed in section 3.9

Interaction Pattern	***Abbreviation***
Human Simple Action Request	H Action Req
Human Simple Action Request with Explicit Confirmation	H Action Req Conf
Human Cancellable Action Request	H Cancel-Action Req
Human Cancellable Action Request with Explicit Confirmation	H Cancel-Action Req Conf
Robot Self Initiated Simple Action	R Action
Robot Self Initiated Cancellable Action	R Cancel-Action
Robot Self Initiated Cancellable Action with Explicit Confirmation	R Cancel-Action Conf
Human Information Request	H Inf Req
Robot Simple Information Request	R Inf Req
Robot Correctable Information Request	R Cor-Inf Req
Robot Information Request with Explicit Confirmation	R Inf Req Conf
Robot Information Request with Explicit Confirmation and Task Acknowledgment	R Inf Req Conf Task Ack
Robot Rejectable Information Request with Explicit Confirmation	R Rej-Inf Req Conf
Robot Rejectable Information Request with Explicit Confirmation and Task Acknowledgement	R Rej-Inf Req Conf Task Ack
Human Object Demonstration	H Obj Demo
Human Object Demonstration with Explicit Confirmation	H Obj Demo Conf
Human Object Test	H Obj Test
Human Interaction Opening	H Int Open
Human Interaction Closing	H Int Close
Human Interaction Reset	H Int Reset
Human System Reset	H Sys Reset
Robot Interaction Opening	R Int Open
Human Simple Statement	H Statement
Robot Notification	R Notification
Robot Simple Statement	R Statement
Robot Suggestion	R Suggestion
Robot Ask Repeat	R Ask Repeat
Robot Suggest Interaction Reset	R Suggest Reset

Table 3.2: List of all Interaction Patterns and their abbreviations. Blocks are: (i) Action patterns, (ii) information patterns, (iii) object patterns (iv) interactional patterns, (v) general patterns and (vi) clarification patterns.

3.8 Interaction Patterns as an Internal Dialog Model

This section describes the internal view of the Interaction Patterns, i.e. their role as an internal dialog model. Section 3.8.1 first details how dialog management is based on Interaction Patterns. Section 3.8.2 discusses several implications of the approach with respect to discourse planning, grounding, clarification and multimodality.

3.8.1 The Dialog Management Process

A rough schematic of the **processing flow** within the PaMini dialog manager is shown in figure 3.18. Input is received either from the user, e.g. from a speech recognition or speech understanding component, or task events from the robotic subsystem. In case of user input, the dialog manager has to select the appropriate Interaction Pattern first. This includes mapping the user input to a dialog act. The dialog act or task event, respectively, is then fired as input event on the Interaction Pattern's statechart. As a result, task operations may be performed, and a robot dialog act is generated, which is configured with its concrete manifestation according to the Interaction Pattern's configuration. The configured dialog act is then performed by an output source, e.g. a text-to-speech component.

Figures 3.19, 3.20 and 3.21 show this process in more detail. Figure 3.19 illustrates the dialog manager's activities when receiving **user input**. First, it is checked whether the interaction is open already. If so, it is checked whether the current pattern to check is interleavable with the one on top of the stack (if it is not the topmost one itself). If this is the case, it is further checked whether the current Interaction Pattern matches the given user input. The match criterion is whether the pattern is in a state from which a transition can be triggered through a dialog act as which the given user input can be interpreted according to the conditions specified by the pattern configuration. According to this criterion, all patterns are checked in sequence, until a matching one is found. The default search strategy is testing the active patterns (i.e. the patterns that have been initiated and

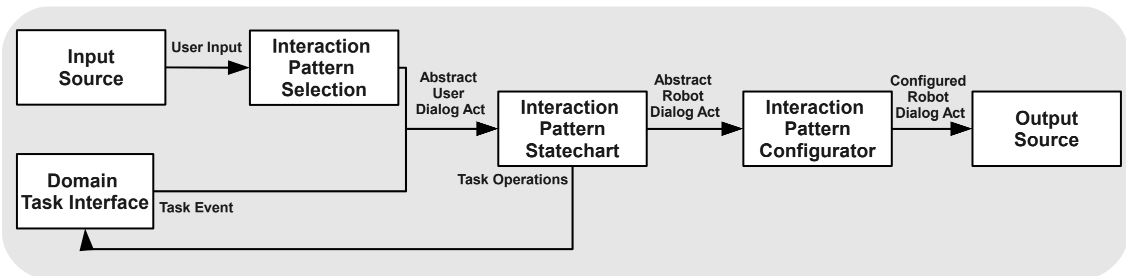


Figure 3.18: Schematic architecture of the PaMini dialog manager.

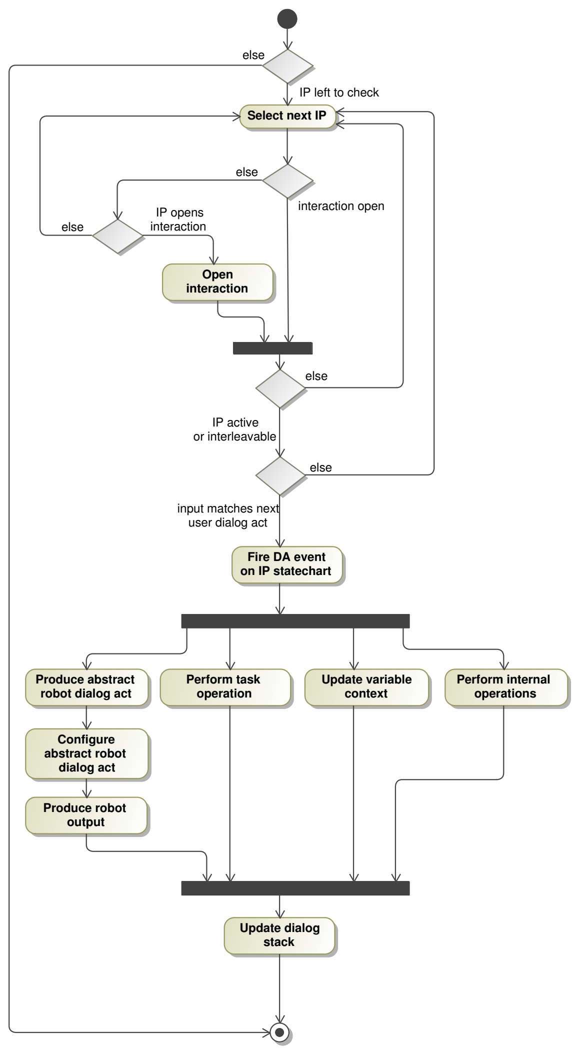


Figure 3.19: UML activity diagram illustrating how user input is processed.

are not terminated yet) on the dialog stack[1] first, beginning with the topmost one, then testing the inactive patterns. If the pattern matches, the dialog act represented by the user input is fired as an event onto the Interaction Patterns's statechart. This will cause four types of activities: (i) the production of a robot dialog act, which is represented in abstract form first, then configured and performed, (ii) the execution of task operations, i.e. the update of the task state and possibly the task specification as well, (iii) an update of the

1 Since the default search strategy considers not only the topmost, but also the patterns below, it is not an actual stack, but rather a list of patterns. However, the strategy can be configured easily such that the characteristics of a stack are established.

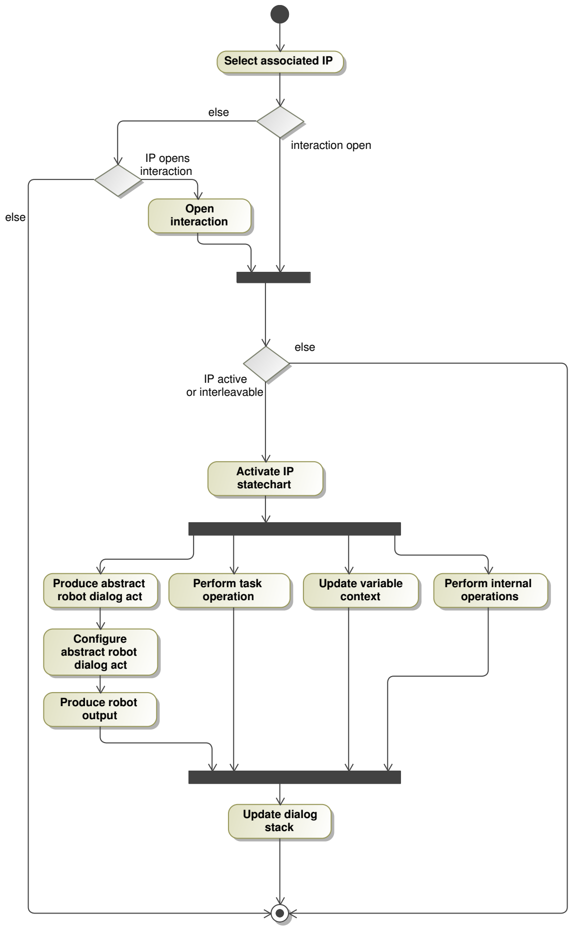


Figure 3.20: UML activity diagram illustrating how task requests are processed.

variable context and (iv) internal operations, such as opening or resetting the interaction. While (i) will be executed for each user input, the activities (ii-iv) are optional. Also, (i) is specified as an output of the Interaction Pattern, while (ii-iv) are specified as state action, which means that (i) will be executed asynchronously while (ii-iv) are executed in blocking mode. Finally, the dialog stack is updated, which includes putting the current pattern on top of the stack or removing it if it is finished. In case that the interaction had not yet been open on receiving the user input, the current Interaction Pattern is only processed further if it is capable to open the interaction. By default, this applies only to the Human Interaction Opening or Robot Interaction Opening. This restriction is intended to prevent the robot from reacting to noise in its environment or to utterances that are not directed at it.

Figure 3.20 shows the processing of **dialog tasks** that are requested from the dialog system by other system components. This occurs whenever an Interaction Pattern should

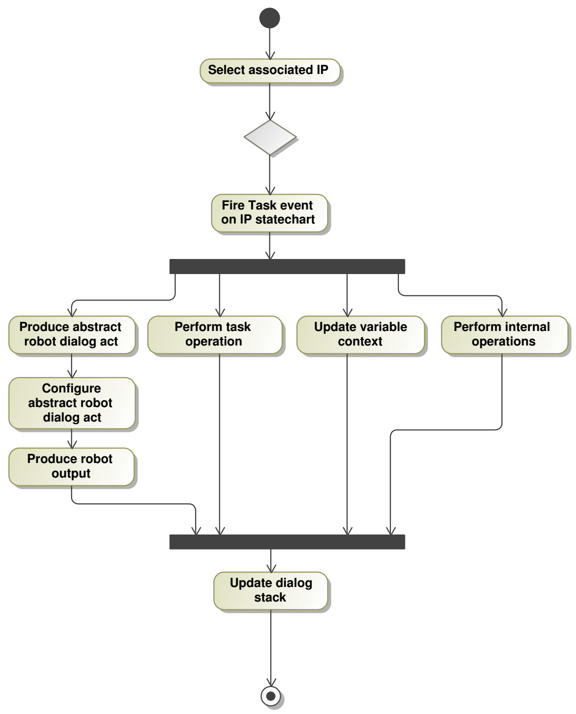


Figure 3.21: UML activity diagram illustrating how task events are processed.

be triggered that is initiated by the robot. Processing of task requests is very similar to processing of user input, except that the pattern associated with the task request is known from the start. Thus, the search for the matching pattern can be omitted.

Processing is even simpler for incoming **task events**, shown in figure 3.21. The task events pertain to tasks that were initiated by the dialog manager. Thus, as with processing of task requests, the pattern they pertain to is known from the start. Additionally, the interleaving check is omitted: even if the latest pattern does not allow interleaving with the pattern the events refer to, the notifications about on-going tasks are still processed. This makes sure that important system information is passed to the user in any case.

3.8.2 Global Discourse Planning, Grounding and Other Aspects

Looking back at figure 3.18 which shows the processing flow within the PaMini dialog manger, the absence of a large-scale interaction model beyond the structures determined by the Interaction Patterns is noticeable. In fact, the dialog manager does not employ a model for **global discourse planning**, i.e. a model of how Interaction Patterns should be combined during interaction, such as WITAS' activity model or Collagen's recipes. Instead, the Interaction Patterns are triggered either through user input (e.g. the Human Simple Information Request shown in figure 3.16), or through a task request (e.g. the

Robot Information Request with Explicit Confirmation shown in figure 3.17). Thus, while local discourse planning is determined through the structure of Interaction Patterns, global discourse planning is not handled within the dialog system.
This decision results from the specific requirements of the robotic domain and has been a very conscious one. As discussed in section 2.2.1 and 3.1, the dialog flow can typically not be pre-structured beforehand in robotics. It rather evolves dynamically during interaction, as a reaction to changes in dynamic environment whose timing and order cannot be predicted. The dialog system does not have sufficient information about the situation the robot is in, and is therefore not qualified to take decisions about the system behavior. From a software engineering point of view, this implements the principle of separation of concerns: The dialog system's responsibility is to provide a library of Interaction Patterns, while their invocation is decided by an external handler. The external handler could for instance be a central decision-making component (e.g. the component delivering the current *interaction goal* in the Curious Robot scenario (cf. section 6.2)). Alternatively, the events determining the dialog flow could come directly from the respective system components (e.g. the environment representation component in the Home-Tour scenario (cf. section 6.1)), which implements a more reactive manner of control. Thus, the proposed approach to dialog management can be integrated with different architecture and control styles (provided that they support the Task State Protocol).
This concept stands in contrast to dialog approaches for traditional, non-situated domains, where the dialog system generally controls the functioning of the rest of the system. In fact, the case study described in section 4 shows that – beside the lacking support for asynchronous communication – one of the main difficulties in transferring approaches from traditional dialog domains to robotics is that they require a pre-structured interaction model (in case of RavenClaw) or task model (in case of Collagen).

Many dialog systems employ explicit models of **grounding**. The RavenClaw framework, for example, supports different built-in grounding policies for the concepts that are gathered during interaction (e.g. city_name, date etc.). The grounding policies are decoupled from the actual dialog flow specification and include mainly implicit and explicit confirmation [BR09]. A more sophisticated model of grounding has been proposed by Traum [Tra94]. His model describes how mutual understanding is established in a conversation by performing sequences of *grounding acts* within *discourse units*. A discourse unit consists of an initial presentation, and of as many additional utterances a required to ground the presentation. Grounding acts such as Repair, Acknowledge or Cancel describe the level of functionality of an utterance that affects grounding. Figure 3.22 shows a transition model that specifies admissible sequences of grounding acts. For example, propositional content may first be presented by the initiator (Initiate(I)), who may request acknowledgement (ReqAck(I)) for it. If the responder acknowledges (Ack(R), the discourse unit is grounded. In addition to the transition network model shown here, he proposed a simplified, but more efficient, finite-state model of grounding, which was implemented within the scope of the TrindiKit dialog toolkit [LT00].

The approach of Interaction Patterns does not model the grounding process explicitly. Rather, grounding is incorporated implicitly within the Interaction Patterns by providing variations with different confirmation strategies (implicit, explicit, none). In particular the Interaction Patterns related to information negotiation resemble Traum's transition models of grounding. Both models consider confirmation, correction and rejection of information. However, they differ in how they model utterances. While the Interaction Pattern approach simply maps user input as a whole to dialog acts, Traum's grounding model differentiates the levels of functionality the utterance may have with respect to turn-taking, grounding etc. Also, Traum's model may serve as a cognitive model of an agent's mental state during a conversation, while the Interaction Patterns are intended as a purely technical model.
From usability point of view, a generic grounding policy may be beneficial because it ensures a consistent system behavior, both at different locations in the dialog and also across scenarios [BR09]. This is a feature that the Interaction Patterns also exhibit.

Clarification of non-understood utterances is closely related to information grounding. In contrast to misunderstandings, which can be resolved through corrections within the respective patterns, in case of non-understandings, the speech recognition result is not interpretable at all. For this purpose, there are clarification patterns that ask the user to repeat the utterance, or suggest to start abandoning the current situation and to start over again. Unlike the other patterns, the clarification patterns are built in, and they are interleavable with all other patterns. Additionally, for each robot dialog act a rephrasing may be defined in the Pattern Configuration, which is applied if the human's reply to it is non-interpretable. This generates dialog in the following style.

R What is that?
U [non-understandable]
F Pardon, what did you say that was?
U A banana.

Another crucial aspect in embodied and situated interaction is **multimodality**. Since the presented approach operates at the dialog act level (as most dialog managers do), the origin and modality of the in- and outputs is not specified. This allows PaMini to manage multimodal interactions as well. Technically, additional in- and output sources can be implemented and plugged into the framework. The scenarios that have been implemented with PaMini so far (cf. chapter 8) make use of multimodal output primarily. For example, in the Curious Robot scenario, the robot points at the object while asking for its label. In the PlaSta scenario, the robot accentuates its utterances with bodily gestures. Non-verbal input information has so far been used more often in form of system events that initiate certain Interaction Patterns. In the multi-party quiz game, for example, a Robot Interaction Opening is triggered if a human approaches the robot who seems willing to interact with it. However, PaMini does not offer provisions for fusing or synchronizing multiple modalities. This is expected to be done externally.

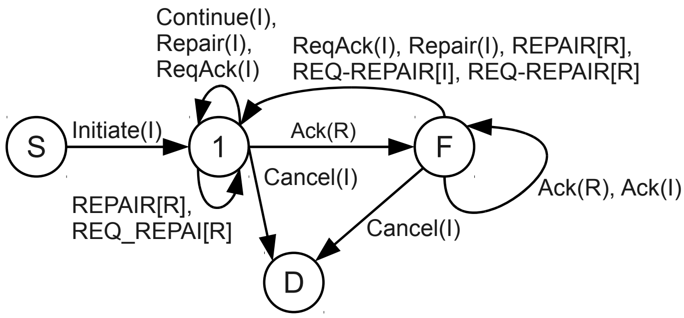


Figure 3.22: Traum's recursive transition network for discourse units (after [Tra94]). Recursive transitions to other networks are represented in capital letters. The actor is given in brackets (I for initiator, R for Responder). The states S and F denote the start state and final state, respectively, and the state D denotes a state in which the discourse unit has been abandoned.

In the motivation of this chapter, **desirable characteristics** for dialog APIs have been mentioned. Having provided the necessary details on the proposed approach, we can now check whether they have been realized:

- **Task independence**: With the Task State Protocol as gateway to communication between the dialog system and the domain subsystem, the dialog and the domain level are kept separated, but by linking task events with dialog acts in the Interaction Patterns, a fine-grained integration of dialog and domain is achieved.
- **Flexibility**: The approach is not tailored to a specific domain or interaction type, but relies on general principles. Its flexibility is demonstrated by the variety of applications that were implemented based on it, as detailed in chapter 8.
- **Modularity**: The PaMini API encapsulates, on the one hand, the details of dialog management by providing default behavior that will be appropriate in most cases. On the other hand, the default behavior (in particular the strategies for pattern interleaving, pattern selection, interaction opening and clarification) is made configurable or overridable to take into account special cases.
- **Reusability**: As the Interaction Patterns have emerged from identifying recurring structures over different domains, they, by definition, enable reuse across scenarios. This is demonstrated by the wide range of different scenarios they are used in, as described in chapter 8, and by an analysis that investigate for each pattern its usage in the different scenarios, shown in table 5.1.
- **Scalability**: The self-containedness of Interaction Patterns allows for an iterative development process in which scenarios are gradually extended without breaking existing functionality. The iterative development process of the Curious Flobi scenario demonstrates that, with this approach, even the development of complex HRI applications become manageable.

3.9 Interaction Patterns as Building Blocks for Interaction

As pointed out above, Interaction Patterns serve not only as internal dialog model for the dialog manager, but also as configurable building blocks of interaction for dialog designers. The patterns themselves determine the general course of events, but do not specify what the robot says or what kind of task is to be executed exactly. These specifics are defined in the **configuration** that is associated with each pattern. Thus, to realize a concrete scenario, a set of Interaction Patterns needs to be chosen and configured. In detail, it needs to be specified what kind of (possibly multimodal) user input is interpreted as a specific dialog act. For the robot's dialog acts, their surface form needs to be specified, which also may be multimodal.

Technically, the configuration of the dialog acts is written in a domain specific XML configuration language. Figure 3.23 shows an excerpt from a pattern configuration associated with the Robot Information Request with Explicit Confirmation shown in figure 3.17. It specifies that the robot dialog act *R.question* in state *initial* is expressed by asking "What is that?", accompanied by a pointing gesture and mimics. Additionally, a rephrasing for this dialog act is given that will be used if the user's reply is not interpretable. Further, it is specified that the XML representation of the user's answer in state *asked* will be interpreted as dialog act *H.answer* if it matches the given XPath expression.

Besides the dialog acts, also the task communication needs to be configured. This includes the task specification itself as well as possible task specification updates. In addition, the definition of context variables is customizable by the developer. Context variables can be used for parameterizing the robot's dialog acts, and to use them for task specification updates. The task communication and the variable context is not configured within the XML configuration file, but through PaMini's Java API. Thus, the different configurational aspects are kept separated. The basic dialog act configuration is specified in a clearly laid out manner using the XML configuration language, while the more complex task and variable configuration (which are not always necessary anyway) are configured in Java, which provides more flexibility to the developers. Altogether, the following steps are required to set up an Interaction Pattern through the Java API (not including the implementation of the back-end):

1. Writing the dialog act configuration in XML.
2. Creating an Interaction Pattern object in Java, which requires as its arguments
   - the dialog act configuration file,
   - a task specification provider, if the pattern initiates a system task,
   - a task updater, if the pattern modifies the task specification of a task,
   - a triggering XPath expression, if the pattern represents a dialog task,
   - a variable provider, if the pattern makes use of variables.
3. Registering the Interaction Pattern with the dialog manager.

4. Customizing further parameters of the interaction in Java, including
   - the interleaving strategy,
   - the pattern selection strategy,
   - the clarification strategy,
   - the interaction opening strategy.
5. Starting the dialog thread.

Step 2) covers the configuration of a single Interaction Pattern, including beside the dialog act configuration the task and variable communication. While the dialog act configuration is required for each pattern, the task configuration is only required for those Interaction Patterns that involve tasks. In contrast, the variable communication is completely optional. Also, step 4) is optionally and covers the customization of interaction in general. For all configuration parameters that are specified through the Java API, default classes are provided by the dialog framework. In addition to these, dialog designers may implement their own classed as required.

```
<robotDialogAct state="initial" type="R.question">
   <output>
      <verbalization text="What is that?"/>
      <point coordinatesXpath="//Region"/>
      <mimic name="Curious" />
   </output>
   <rephrasing>
      <verbalization text="What did you say was that?"/>
      <point coordinatesXpath="//Region"/>
      <mimic name="Smile" />
   </rephrasing>
</robotDialogAct>

<humanDialogAct state="asked"
   xpath="/utterance/semanticInfo[category='description']" type="H.answer"/>
```

Figure 3.23: Excerpt from an Interaction Pattern configuration.

3.10 The Pattern Library and its Development

Figure 3.24 shows the number of Interaction Patterns at a given time, and its increase over time. Often, the implementation of new patterns was related to the development of new scenarios and their requirements. To point out this relation, red dots mark the scenarios that were crucial for the extension of the pattern library[1]. In the beginning, two

1 In addition to the marked scenarios, more scenarios have been implemented, but not all of them required the implementation of new Interaction Patterns. See figure 5.2 for a complete overview of all implemented scenarios.

patterns were implemented as a proof of concept. Soon after, the re-implementation of the (simulated) Curious Robot and Home Tour scenario caused a quick boost of growth up to ten patterns. With the first release, the development of the Receptionist Vince scenario began. Together with the beginning of development of the RoboCup and the Curious Flobi scenario, this entailed a gradual growth up to 28 patterns within nine month. From 06/11 to the time of writing (02/12), the number of patterns has remained stable. However, the need of an additional action pattern has arisen from two scenarios independently. Both the Memory Game Scenario (cf. section 8 and an upcoming new scenario – a sports companion for spinning – requires a pattern in which the robot gives an instruction to the human, the completion of which is controlled by a vision component. A new Interaction Pattern for this use case will be added soon.

Figure 3.25 shows in detail the creation date of each Interaction Pattern, in relation to the main development scenarios. It is noticeable that for each pattern group, a few representatives were available very early, such as the Human Cancellable Action Request (belonging to the action patterns), the Robot Correctable Information Request (belonging to the information patterns), the Human Interaction Opening (belonging to interactional patterns), or the Human Simple Statement (belonging to the group of general patterns). Variations of these patterns were added later. For example, the RoboCup required the implementation of new action patterns where the robot's action was to be confirmed explicitly, in order to cope with the background noise. An exception to this are the object-related patterns, which were only introduced in the course of the development process of the RoboCup and the Curious Flobi scenario. For further discussion of different usages of the patterns within the different scenarios, see also section 5 and particularly figure 5.2.

Given the increasing number of Interaction Patterns, and the fact that many patterns differ only slightly from each other, the question arises whether the pattern library has come to a point at which it contains unnecessary redundancy. Closer examination of the structure of the existing Interaction Patterns reveals that, again, recurring elements can be found, for example dialog act sequences that deal with explicit confirmation of actions and information, or with correction of information. Thus, an obvious approach to reduce redundancy would be to provide not complete Interaction Patterns as a whole, but parts of patterns that the dialog designers can assemble into individual Interaction Patterns on their own. However, this approach has several drawbacks. It would be difficult to preserve the self-containedness of Interaction Patterns as well as the conformance to the Task State Protocol. Also, redundancy would probably only be shifted from the framework to the individual scenarios, and the reusability potential would not fully be exploited. Last, it would reduce usability and limit accessibility, especially for unexperienced developers, because designing new Interaction Patterns – even from given components – requires expertise both in dialog design and in system engineering. Hence, an alternative approach would be to condense similar Interaction Patterns into a common core and making

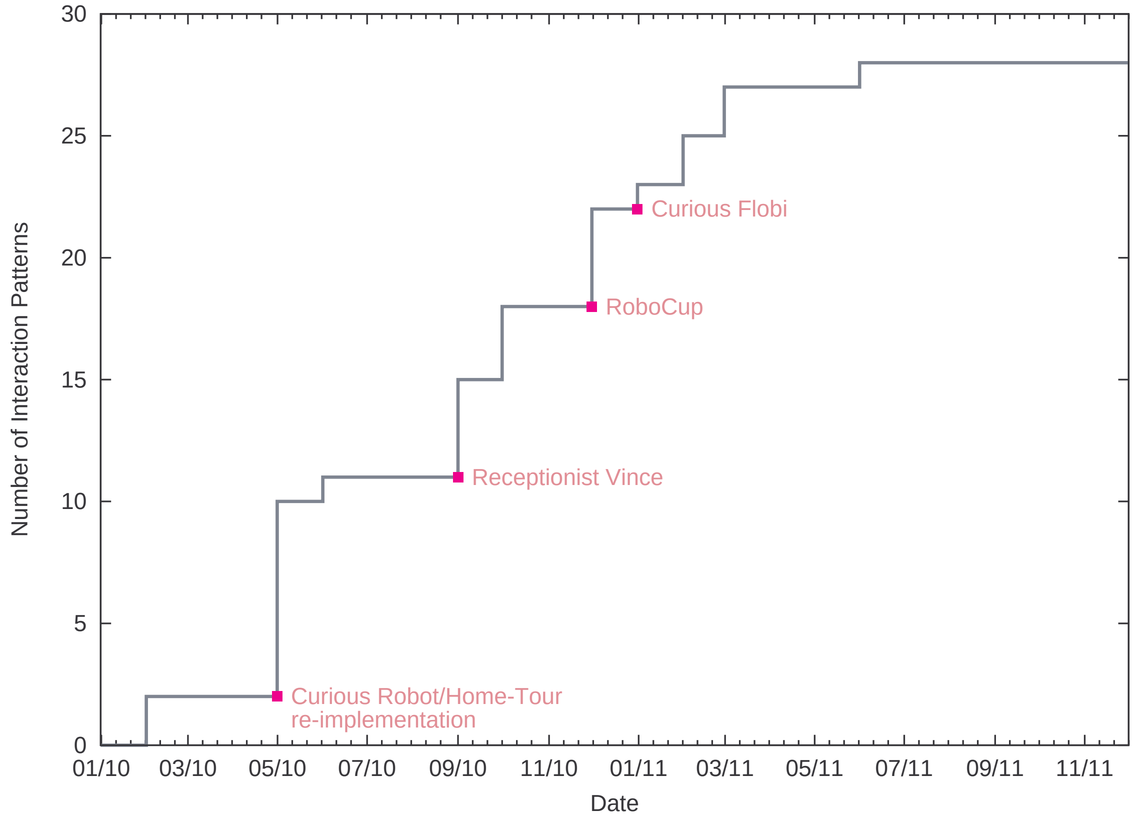


Figure 3.24: The number of Interaction Patterns over time. The points mark the beginning of development of scenarios that entailed the implementation of new Interaction Patterns.

the differences (i.e. the confirmation or correction strategy) configurable through the framework's API. A drawback of this approach would be the increased configuration effort. Altogether, the question how redundancy over Interaction Patterns can be reduced while preserving framework usability and the favorable properties of Interaction Patterns deserves further research.

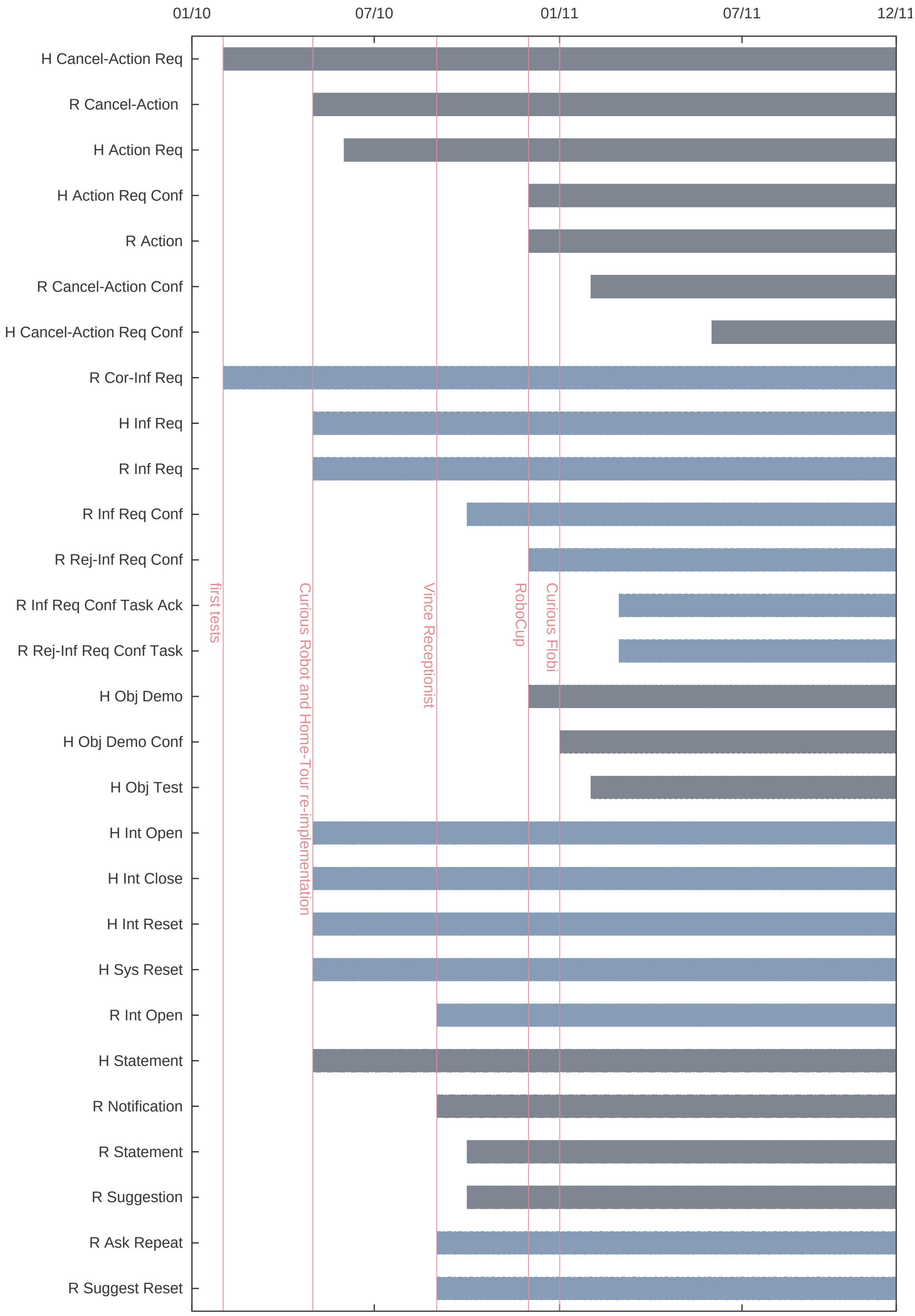


Figure 3.25: Creation date of each Interaction Pattern. Blocks are: (i) Action patterns, (ii) information patterns, (iii) object patterns (iv) interactional patterns, (v) general patterns and (vi) clarification patterns. Within the blocks, patterns are sorted by creation date. The lines mark the begin of development of scenarios that entailed the implementation of new Interaction Patterns.

4 Developer-Centered Evaluation of the Proposed Approach

As outlined in section 2.4.2, an evaluation of the overall approach needs to include not only the quality of the implemented dialog, but also evaluation of the framework itself. While several aspects of interaction quality are discussed throughout the second part of this thesis, this chapter describes two studies that focus on evaluation of the framework. Section 4 describes four case studies that investigate the efficacy of the proposed PaMini framework, i.e. the question how well the framework serves its purpose. Framework usability, i.e. the ease of programming new scenarios, is evaluated in section 4.
This section describes four case-studies in which a typical robotic scenario is re-implemented with different dialog frameworks, namely the RavenClaw framework, the Dipper implementation of the Information State Approach, the collaboration manager Collagen, and the proposed PaMini framework. Besides demonstrating the efficacy of the PaMini framework for implementing a typical robotics scenario, the aim of this comparison is twofold. On the one hand, it is meant to give an overview of state-of-the-art dialog modeling techniques, and to illustrate the differences between these. On the other hand, it attempts to illustrate challenges specific to robotics, and why approaches from traditional domains often struggle to meet them (being well aware that the investigated approaches originally had not been intended for robotics).

A simplified version of the Curious Robot object learning and manipulation scenario (which will be described in detail in chapter 6.2) was chosen as target scenario for our case studies. Since it includes problems of perception and learning as well as action oriented communication, it can be considered as a typical robotic application and is thus suitable for the case studies. In detail, the target scenario for the case studies is determined as follows. Interaction is carried out following a mixed-initiative style by letting the robot ask for unknown objects and by allowing the user to initiate a teaching or a query episode at any time. Thus, whenever the robot detects an unknown object, it asks for its label (for the case studies, references through non-verbal gestures were not considered). Once the label is given by the user, the robot asks how to grasp the object, which the human is expected to answer by naming the grip type. Having acquired both label and grip, it autonomously grasps the object, while reporting both start and completion or failure of the action. Grasping may also be rejected by the back-end right away, or the user may cancel the ongoing grasping action. Additionally, the user can at any time ask the robot to enumerate the objects learnt so far or how to grasp a specific object. The interaction is opened and closed by the user, by greeting and saying goodbye respectively.

Although the target scenario is kept simple, it presents a number of typical challenges that dialog systems in robotics have to face, as discussed in section 2.2.1. First of all, the robot must *react dynamically* to its environment. Timing and order of the robot's questions cannot be fixed beforehand since they depend on the robot's perception of the world. The action to select next therefore is supposed to come from a back-end component, in form of an *interaction goal* which may be either *label*, *grip* or *grasp*. Second, the user's test questions require the dialog system to cope with *focus shifts* and, as they may be asked during the robot's grasping action, even with *multitasking abilities*. Finally, going on with the interaction during grasping while still enabling feedback about the on-going action and the possibility to cancel it requires some kind of *asynchronous coordination* between dialog system and back-end.

As the focus of the case studies lies on dialog modeling, the goal was not to achieve a fully fledged implementation running on a robotic platform. Therefore, speech recognition and speech synthesis were replaced by text in- and output and all perception and motor activities have been simulated. Also, subtle yet important aspects of the interaction were ignored, such as nonverbal cues, social behavior or the engagement process that typically precedes the interaction.

4.0.1 Case Study 1: Ravenclaw

The first case study investigates the RavenClaw dialog manager, which has already been introduced as an example for a descriptive dialog model in section 2.1.2. A well-maintained documentation including step-by-step tutorial helps getting started with the framework. To realize a speech application with Ravenclaw, the developer has to create a context-free grammar for the semantic parser, output templates for the natural language generation and of course the dialog task specification, consisting of a tree of dialog agents, each capable to handle a subtask of the interaction. There are different agent types: *Agencies* represent the non-terminal nodes, i.e. the tasks that are further decomposed, *Inform* agents produces system output, *Request* agents request information from the user, *Expect* agents expect information from the user without explicitly requesting it, and *Execute* agents perform back-end calls, such as a database query. A domain independent dialog engine executes the tree in a depth-first manner using a stack. Through pre- and postconditions associated with the agents a deviation from the processing flow can be achieved.

Figure 4.1 shows a possible dialog task specification for our test scenario. Its main part is the *PerformTask* agency, which is divided into two agencies handling human and robot initiative respectively. The *SystemInitiative* agency is reset after completion and executed repeatedly unless the user initiative agency is triggered or the user ends the interaction. It consists of an *Execute* agent fetching the current interaction goal from the back-end, and

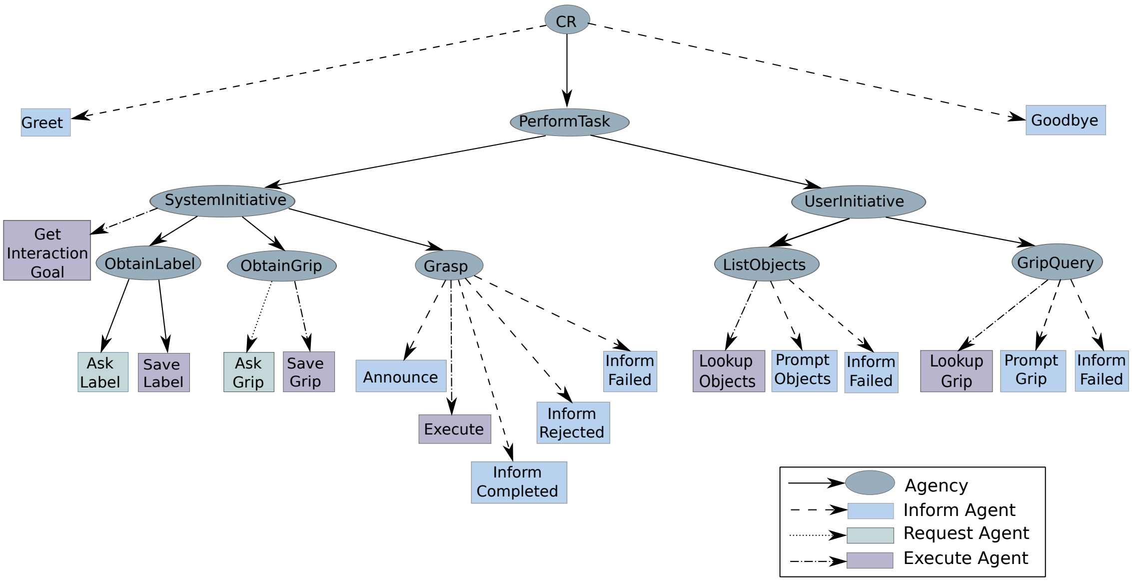


Figure 4.1: Ravenclaw's task tree for the Curious Robot scenario.

the agencies *ObtainLabel*, *ObtainGrip* and *Grasp*. *ObtainLabel* and *ObtainGrip* request label and grip respectively, and communicate it to the back-end where it gets stored. *Grasp* first announces grasping, then executes it and finally reports success, rejection or failure. The three agencies are not executed in succession, but alternatively, on conditions such as the current interaction goal (not shown in the figure). The *UserInitiative* agency can be activated by the user's test questions at any time. This is achieved by adding a trigger directive to its subagents, making *ListObjects* and *GripQuery* respectively the currently focused agent, i.e. the topmost agent on the stack. Table 4.1 illustrates a typical dialog example, including two focus shifts.

Technically, all dialog agents are defined as C++ macros that communicate with the back-end by exchanging user-defined frame data structures. Figure 4.2 illustrates the portion of the dialog task specification that defines the *Grasp* agency and its subagents *Announce*, *Execute*, *InformCompleted*, *InformRejected* and *InformFailed*. *Grasp* is only executed if the interaction goal has the value *label* (line 2), and it succeeds if the grasp action has either been rejected, completed, or has failed (lines 12-14). *Announce* prompts the speech output specified for this situation (line 17). *Execute* then calls the associated back-end function (line 21-22) and stores its result (line 23). Depending on the result (lines 26, 30, 34), the appropriate output is generated (lines 27, 31, 35).

Most requirements of the target scenarios could be realized with Ravenclaw. While the installation of the Olympus framework required considerable effort, including adaptions in

	Utterance	Comment
U1	Hello.	
R1	Hi, hello.	
R2	What is that?	Back-end returns interaction goal *label*
U2	This is an apple.	
R3	How should I grasp the apple?	Back-end returns interaction goal *grip*
U3	What objects do you know already?	Focus shift: *ListObject* focused agent
R4	I know 'apple'.	
R5	How should I grasp the apple?	*ObtainGrip* focused agent again
U4	How do you grasp a banana?	Focus shift: *GripQuery* focused agent
R6	Sorry, I don't know.	
R7	How should I grasp the apple?	*ObtainGrip* focused agent again
U5	With the power grasp.	
R8	I am going to grasp the apple now.	Back-end returns interaction goal *grasp*
		Back-end blocks during grasping
R9	Sorry, I failed to grasp the apple.	
U6	Goodbye.	
R10	Goodbye.	

Table 4.1: Example dialog for the Curious Robot implementation with Ravenclaw.

the source code, writing the dialog agents could be done without difficulties. When it comes to a real-world robotic scenario, a shortcoming might however be that the dialog task tree largely pre-defines the interaction flow. As suggested in our target scenario, a robot needs to react not only to the user's utterance, but also to many kinds of events that occur in its environment. With Ravenclaw, this can be achieved by controlling the navigation through the task tree with pre- and postconditions. However, for highly unstructured scenarios with many possible paths through the task tree, the dialog structure may thus become unclear, up to unstructured spaghetti code at the worst. Already our toy scenario contains a number of "jumps" in the control flow in order to react to the current interaction goal, the user's focus shifts and the back-end results.

Further, difficulties regarding the asynchronous coordination of back-end calls were encountered. While Ravenclaw does support asynchronous back-end calls, it does not provide mechanisms that support further communication between dialog and back-end about a running back-end action. In the target scenario, grasping was therefore implemented using a blocking back-end call, which enables the robot to report success or failure when it is done. With the blocking back-end call however, the interaction cannot be maintained during action execution, and also the possibility to cancel the action could not be realized.

Another issue is reusability. Even for our basic test scenario, the dialog task specification shown in figure 4.1 contains several agents that have a similar structure, e.g. *ObtainLabel* and *ObtainGrip*, or *ListObjects* and *GripQuery*, and one can easily think of another agency with the same structure as the *Grasp* agency, e.g. a following or navigation task. With

```
DEFINE_AGENCY( CGrasp,
   PRECONDITION((int)C("result.interactiongoal") == 2
)
DEFINE_SUBAGENTS(
   SUBAGENT(Announce, CAnnounce, "")
   SUBAGENT(Execute, CExecute, "")
   SUBAGENT(InformCompleted, CInformCompleted, "")
   SUBAGENT(InformRejected, CInformRejected, "")
   SUBAGENT(InformFailed, CInformFailed, "")
)
SUCCEEDS_WHEN(
   (SUCCEEDED(InformCompleted) ||
    SUCCEEDED(InformRejected) ||
    SUCCEEDED(InformFailed)))

DEFINE_INFORM_AGENT( CAnnounce,
   PROMPT( "inform grasping <result")
)
DEFINE_EXECUTE_AGENT( CExecute,
   EXECUTE(
      C("query_type") = NQ_GRASP;
      pTrafficManager-> Call(this, "backend.query <query_type >new_result");
      C("result") = C("new_result");)
)
DEFINE_INFORM_AGENT( CInformCompleted,
   PRECONDITION((int)C("result.taskstate") == RC_COMPLETED)
   PROMPT( "inform grasping_completed <result")
)
DEFINE_INFORM_AGENT( CInformRejected,
   PRECONDITION((int)C("result.taskstate") == RC_REJECTED)
   PROMPT( "inform grasping_rejected <result")
)
DEFINE_INFORM_AGENT( CInformFailed,
PRECONDITION((int)C("result.taskstate") == RC_FAILED)
   PROMPT( "inform grasping_failed <result")
)
```

Figure 4.2: Ravenclaw's dialog task specification for the *Grasp* agency and its subagents.

the *Inform*, *Expect* and *Execute* agents as the only unit of pre-modeled conversational capabilities, Ravenclaw does not account for such recurring structures, which are not specific to robotics but will occur in any domain.

A new version of the Olympus dialog architecture (in which Ravenclaw is embedded) is described briefly in [RE07]. This new version (which is not the one that has been used for this case study) features a multi-layer architecture for event-driven dialog management. It was originally designed to address the issue of reacting to conversational events in real-time so as to enable flexible turn-taking and to react to barge-ins. With the proposed architecture, also non-conversational events (e.g. perceptual events) can be handled. It therefore seems probable that some of the above difficulties could be resolved with it. In particular, with an event-based architecture, the dialog manager could react directly to a change of the current interaction goal. Also, it could react to update events of a robot action (such as *grasping begins*), while keeping the interaction going. However, it lacks an overarching structure for temporally extended actions (such as the *tasks* in the PaMini framework), and it lacks a generic mechanism for handling such events (such as the *Task State Protocol* in PaMini). This means that the event processing, i.e. keeping track of the events associated with the dialog moves, is still left to the developers.

Apart from the above difficulties, Ravenclaw has proven to support certain aspects of the target scenario very efficiently. For one, speech understanding integrates naturally into dialog modeling and output generation. The concepts of the semantic speech understanding grammar designed by the scenario developer are available within the dialog specification and within the output generation component. Dialog variables do not need not be specified explicitly.
Further, Ravenclaw uses a generic grounding model that provides several strategies for concept grounding, such as implicit and explicit confirmation strategies, and non-understanding recovery strategies, such as repeating the original prompt, or asking the user to repeat or rephrase [BR08]. The grounding policies are specified in a configuration file, which is the reason why the dialog task specification in figure 4.1 does not contain agents for confirming and correcting label and grip.
Finally, the fact that Ravenclaw does not provide pre-modeled conversational structures can also be viewed as a benefit: the scenario developer does not have to stick to the structures provided, but has full control over the dialog flow.

4.0.2 Case Study 2: Collagen/Disco

The second approach that was investigated is the collaboration manager Collagen (for *Coll*aborative a*gen*t) [RS98]. The Collagen approach has already been described as an example for mental-state-based dialog modeling in section 2.1.3. Even though it is rather a plug-in for intelligent user interfaces than a dialog system in the narrower sense (and thus does not have provisions for speech in- and output), it was included in our case studies because it addresses some aspects that are very relevant for robotics, such as agents communicating about a task and coordinating their actions in order to work towards a shared goal, while accounting for physical actions as well. At the core of Collagen is a task model for the specific application domain. It defines the domain goals and specifies how to achieve them based on goal composition rules, called *recipes*. The task model is used to track the user's task progress and to generate appropriate system utterances automatically. The case study was however not conducted with the Collagen framework itself, but with its open-source re-implementation Disco [HR10].

In order to implement the target scenario, only the collection of recipes for goal decomposition needs to be specified. They are shown in figure 4.3. Originally, recipes are defined using the XML-based task model description standard CE Task 1.0 [CET08]. However, a tree representation is used here for the sake of better readability. The upper part of the figure shows the top-level goals *Greeting*, *ObjectQuery* (i.e. the user asks to enumerate

	Configured utterance	Generated utterance	Comment
U1	Hello.	Let's achieve *Greeting*.	User selects goal *Greeting*
R1	Hello.	Ok.	Robot executes *SayHello*
U2	Let's explore the objects on the table.	Let's achieve *RobotInitiative*.	User selects goal *RobotInitiative*
			Back-end returns interaction goal *label*
R2	What is that?	Please execute *TellLabel*.	Robot asks user to perform *TellLabel*
U3	An apple.	An apple.	User asserts that *TellLabel* done
R3	Ok.	Ok.	Robot executes *SaveLabel*
U4	Let's explore the objects on the table.	Let's achieve *RobotInitiative*.	User selects goal *RobotInitiative*
			Back-end returns interaction goal *grip*
R4	How should I grasp it?	Please execute *TellGrip*.	Robot asks user to perform *TellGrip*
U5	What objects do you know already?	What objects do you know already?	Focus shift: User selects goal *ObjectQuery*
R5	Ok.	Ok.	Robot executes *ListObjects*
R6	How should I grasp it?	Please execute *TellGrip*.	Back to *TellGrip*
U6	With the power grasp.	With the power grasp.	User asserts that *TellGrip* done
R7	Ok.	Ok.	Robot executes *SaveGrip*
U7	Let's explore the objects on the table.	Let's achieve *RobotInitiative*.	User selects goal *RobotInitiative*
			Back-end returns interaction goal *grasp*
			Robot executes *Grasp*
R8	Ok.	Ok.	*Grasp* failed
U8	Goodbye.	Let's achieve *Goodbye*.	User selects goal *Goodbye*
R9	Goodbye.	Ok.	Robot executes *SayGoodbye*

Table 4.2: Example dialog for the Curious Robot implementation with Collagen/Disco.

the objects learnt), *GripQuery* (i.e. the user queries the appropriate grip for a specific object) and *Goodbye*, each of which can be achieved by a robot's action. For instance, the goal *Greeting* can be achieved by the robot's *SayHello* action. It may seem somewhat surprising that the mutual greeting can be achieved by the robot's *SayHello* action alone, but the user's greeting has already been carried out with the user selecting the top-level goal *Greeting*, as illustrated at beginning of table 4.2 (utterances U1, R1). The top-level goal *RobotInitiative*, shown in the lower part of figure 4.3, covers the goals and actions concerning the robot's initiative. It is divided into the subgoals *ObtainLabel*, *ObtainGrip* and *Grasp*, each with an applicability condition over the current interaction goal. The subgoal *ObtainLabel* can be achieved with the user executing *TellLabel* and the robot executing *SaveLabel*; likewise with *ObtainGrip*. Again, it might seem surprising that the *ObtainLabel* subgoal does not imply a robot action such as *AskLabel*, but, similar as with the greeting, the robot's label query is expressed as a suggestion to the user to execute

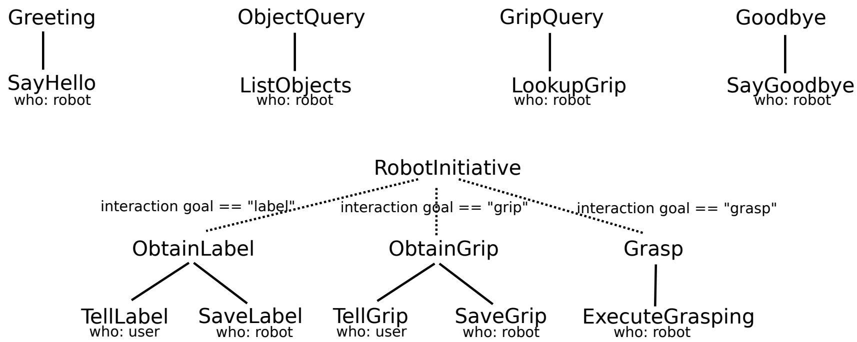


Figure 4.3: Collagen's recipes for the Robot Initiative goal.

TellLabel (cf. table 4.2, utterances R2 and U3).

Listing 4.4 shows how the recipe for *RobotInitiative* is coded in the XML task specification language. It is decomposed into its three subtasks (lines 3, 12 and 21), which again are decomposed further (lines 5-6, 14-15 and 22). Lines 4, 13 and 22 encode the applicability conditions for the respective subtask. The value of the built-in variable *external* indicates whether it is the user or the system who is supposed to execute the subtask. For example, *TellLabel* is assigned to the user (line 7), while *SaveLabel* is assigned to the system (line 8). Further, variables can be passed from one subtask to another, for instance the label (line 9) or the grip name (line 18). A task model description may also contain JavaScript fragments that connect the model with the underlying application or device, as required for polling the current interaction goal (lines 5, 19, 33). The dialog fragment shown in table 4.2 illustrates in detail how the dialog evolves from these recipes: the user selects the top-level goals, and the robot either performs its part of the task, if possible, or suggests an appropriate action to the user.

A fundamental difference to the implementation with Ravenclaw is that only the task needs to be specified, not the dialog flow itself. The dialog is generated automatically out of a generic rule framework [RLGR02] based on the current discourse state and the recipes. Rules specify the system's next action for a particular situation. For instance, the *Execute* rule specifies that a primitive task that is assigned to the agent should be executed directly, whereas the *AskWho* rule states that for a task whose executor is not determined, the system should return an utterance of the form "Who should perform *goal*?". Collagen provides a collection of default rules, and further rules can be plugged in to implement a different collaboration style.
The generated output can be customized as to how tasks can be referred to and how their execution is confirmed. Table 4.2 contrasts the customized version of the output with the automatically generated version, e.g. "Hello" versus "Let's achieve *Greeting*" in utterance U1. Additionally, the rules do not only generate the system's next action but also the

```
<task id="RobotInitiative">

   <subtasks id="ObtainLabel">
       <applicable> getInteractionGoal() == "label" </applicable>
       <step name="TellLabel" task="TellLabel"/>
       <step name="SaveLabel" task="SaveLabel"/>
       <binding slot="$TellLabel.external" value="true"/>
       <binding slot="$SaveLabel.external" value="false"/>
       <binding slot="$SaveLabel.label" value="$TellLabel.label"/>
   </subtasks>

   <subtasks id="ObtainGrip">
       <applicable> getInteractionGoal() == "grip" </applicable>
       <step name="TellGrip" task="TellGrip"/>
       <step name="SaveGrip" task="SaveGrip"/>
       <binding slot="$TellGrip.external" value="true"/>
       <binding slot="$SaveGrip.external" value="false"/>
       <binding slot="$SaveGrip.grip" value="$askGrip.grip"/>
   </subtasks>

   <subtasks id="Grasp">
       <applicable> getInteractionGoal() == "grasp" </applicable>
       <step name="ExecuteGrasping" task="ExecuteGrasping"/>
       <binding slot="$ExecuteGrasping.external" value="false"/>
       <binding slot="$this.success" value="$ExecuteGrasping.success"/>
   </subtasks>

 </task>
```

Figure 4.4: The RobotInitiative recipe, coded in Collagen's task specification language.

agenda for the user, i.e. a list of candidate actions and utterances. The choices for the user to say, or rather to type, are presented as a menu. For example, the customized choices generated after the robot's label query (R2 in table 4.2) include rejecting the proposed action ("I'm not going to answer your question"), abandoning the top-level goal ("Let's not explore the objects on the table") and focus shifts ("What objects do you know already?", "How do you grasp a banana?").

Although our target scenario is not at all the type of scenario Collagen was intended for originally, many requirements of the target scenario can be realized with it. Its model of collaborative discourse, wherein two interaction partners collaborate on a task by proposing and performing actions, aptly supports the focus shifts that were stipulated in the specification.
In contrast, the robot's task initiative that generates its query for label and grip could not be implemented using the default agent that comes with the framework since it does not suggest top-level goals on its own. It should however be easily possible to adapt the default implementation such that it is able to propose *ObtainLabel*, *ObtainGrip* and *Grasp*

autonomously. For the case study, a work-around was applied by introducing the top-level goal *RobotInitiative*, which the user is to select explicitly ("Let's explore the objects on the table."), whereupon the robot chooses between *ObtainLabel*, *ObtainGrip* and *Grasp*, depending on the current interaction goal.
Another problem that was encountered affects the communication of back-end results, such as the success of grasping or the robot's enumeration of the objects learnt so far. Collagen does not support variable system utterances, e.g. by template-based output generation. This is the reason why the robot simply answers *Ok* when the user asks to enumerate the known objects (cf. R5 in table4.2), or why the robot does not communicate that grasping has failed (cf. R8 in table 4.2). Admittedly, Collagen does not claim to be a complete natural-language processing system, and within the collaborative interface-agent paradigm it would probably be the underlying application that is responsible for representing the application-specific results to the user.
The automatic generation of system utterances is a very powerful technique. However, while the wording of the generated utterances can be configured, the developer can not control *when* utterances are generated. This is the reason why the begin of grasping can not be announced (cf. R8 in table 4.2). Also, generating utterances automatically leads to asymmetry in the task model: while some of the user utterances are explicitly represented as subtasks (e.g. *TellLabel* and *TellGrip*), the system utterances are not present in the task model.
The most serious shortcoming pertains to error handling. The task model provides a built-in *success* variable, indicating the success of a subtask. It is used to control replanning. However, a binary value might not always provide sufficient information. Some applications might want to discriminate between a failure and a rejection of the subtask, or between different error causes. For instance, if a plan fails because the underlying application is otherwise busy, it might be reasonable to re-execute the plan later, whereas retrying might be pointless if the requested functionality is unavailable in general. This shortcoming is particularly serious as the task specification language has been approved by the Consumer Electronics Association (CEA) und the name CE Task 1.0 as a standard for task models relevant to consumer electronic devices [CET08].
Finally, just as the Ravenclaw framework, Collagen does not provide mechanisms for asynchronous coordination of task execution. Thus, neither the user's monitoring questions during grasping could be realized, nor could the grasping action be canceled.

4.0.3 Case Study 3: Dipper

The third case study explored the Information State (IS) approach to dialog modeling [TL03], which has been introduced in section 2.1.3. The IS approach relies on a models of the relevant aspects of information (the *information state*), which is updated by applying *update rules*, based on a certain *update strategy.* The Prolog-based TrindiKit is known as the original implementation of the IS approach [TL03]. Others followed, based on different programming languages. For the case study, the stripped-down re-implementation Dipper

[BKLO03] was used.

Dipper is set on top of the Open Agent Architecture (OAA), a C++ framework for integrating different software agents in a distributed system [MCM99]. OAA agents provide services that other agents may request by submitting a high-level Interagent Communication Language (ICL) expression (a *solvable*, which can be viewed as a service request) to the *facilitator* agent that knows about all agents and mediates the interaction between them. In addition to the *facilitator* and the *Dipper* agent, the implementation of the target scenario includes a *SpeechRecognitionAgent* and a *TTSAgent* for (simulated) speech in- and output, a *MotorServer* agent that simulates grasping, an *ObjectDatabase* that stores object labels and the associated grip, and an *ActionSelection* agent that selects the current interaction goal.

The upper part of listing 4.5 (lines 1-7) shows the information state for the Curious Robot scenario, which is designed such that it models the most obvious information, namely the current interaction goal (line 5), the current user utterance and its interpretation (line 2-3), and incoming events from the back-end task (line 4). Further, it contains control flags that determine whether the system is ready to receive speech input (line 6) or task events (line 7).
The lower part of listing 4.5 (lines 9-33) shows the update rules that are necessary to realize the robot's label query. Update rules are written in the Prolog-like Dipper update language, specified by the triple ⟨*name, conditions, effects*⟩, with *name* a rule identifier, *conditions* a set of tests on the current information state, and *effects* an ordered set of operations on the information state. The first rule, *getInteractionGoal*, deals with the situation when no interaction goal is set (line 10). In that case, an OAA solvable is sent that polls the interaction goal (line 11) and updates the information state with the result (line 12). The second rule, *waitForUtterance*, is applicable if the *listening* flag is set (line 15). It posts a solvable for the *SpeechRecognitionAgent* (line 16), integrates the result into the information state (lines 17-18) and resets the flag (line 19). The *processLabel* rule applies if the user has given an object label (line 22-23). It posts solvables for acknowledging and storing the label (lines 24-25) and resets the information state (lines 26-18). The last rule, *LabelQuery*, posts a solvable that will trigger the label query and set the flag for receiving speech input (lines 32-33), if the current interaction goal is *label* (line 31).

When implementing the target scenario, the idea of a central information state that determines the next steps of the interaction appears to be very intuitive. Also, the division of responsibilities between distributed agents enables a modular approach that roughly resembles the distributed event-based architecture of the original system.
However, problems with respect to the update rules and the update strategy were encountered. While TrindiKit leaves it to the developer to implement the (possibly highly complex) update strategy, Dipper provides a built-in update strategy that simply selects the first rule that matches, applies its effects to the information state, and starts over

```
infostate(record([is:record([
   utterance:atomic,
   interpretation:atomic,
   task_event:atomic,
   interaction_goal:atomic,
   listening:atomic,
   awaiting_event:atomic])])).

urule(getInteractionGoal,
   [eq(is:interaction_goal,'')],
   [solve(getInteractionGoal(X),
    [assign(is:interaction_goal,X)]),]).

urule(waitForUtterance,
   [eq(is:listening,yes)],
   [solve(recognize(X, Y),
    [assign(is:utterance, X),
     assign(is:interpretation, Y),
     assign(is:listening, no)])]).

urule(processLabel,
   [eq(is:interpretation,label),
    eq(is:interaction_goal, label)],
   [solve(store(is:utterance)),
    solve(say(is:utterance Okay)),
    assign(is:interaction_goal, ''),
    assign(is:utterance, ''),
    assign(is:interpretation, '')]).

urule(LabelQuery,
   [eq(is:interaction_goal,label)],
   [solve(say('What is that'),
    [assign(is:listening,yes)])]).
```

Figure 4.5: Dipper's information state definition and update rules for the label query.

with checking the first rule again. This means that rules are executed on a first-come, first-served principle, where the order of the rules matters, resulting in a brittle system behavior. In our case study, this is the reason why e.g. the *processLabel* rule is defined before the *LabelQuery* rule. If it was the other way round, the system would loop over the *LabelQuery* and never execute *ProcessLabel*. Of course, the problem could in principle be overcome by introducing additional control flags, but this would make the information state unnecessarily complex. As a result of this update strategy, some requirements of the target scenario could not be realized.

Focus shifts could only partly be implemented. The problem was not to define the appropriate update rules (e.g. *processListObjects*), but rather that user utterances are processed only at specific points in time, that is, only if the *listening* flag (which were adopted from the example in [BKLO03]) is set. Thus, a focus shift may be initiated only when the robot expects the user to speak, e.g. after having asked the label query. If the rule for speech recognition was applicable at any time, it might conflict with other rules. Also, asynchronous coordination, which the OAA framework actually supports well, could only partly be realized, due to the first-come, first-served update strategy that enables speech input only at certain points. Thus, the robot's feedback on the grasping action could be realized by explicitly waiting for respective task events by virtue of the

waitForTaskEvent rule, whereas the possibility to cancel an on-going grasping action could not be implemented because the *waitForUtterance* rule would have conflicted with the *waitForTaskEvent* rule.
Another issue is that the update rules handle both organizational tasks (such as polling different input sources or producing output) and dialog management tasks. A clear separation of concerns could make the dialog strategy more obvious and prevent the information state from being overloaded with control flags.
In a real-world application, testability and maintainability might become issues. As rule systems get more complex, their behavior can become very hard to predict. Already in our simplified Curious Robot implementation, which required about 15 rules, it was not easy to identify the actual dialog flow.

4.0.4 Case Study 4: PaMini

Finally, the target scenario was re-implemented with the PaMini approach. PaMini is the only one among the discussed frameworks that targets specifically human-robot interaction. To realize the target scenario, a distributed system was set up with (simulated) components that communicate via the Task State Protocol (cf. chapter 3). The system includes components for speech in- and output, an action selection component, and a motor server. This breakdown is similar to the one in the Dipper-based implementation.

When selecting the interaction patterns to use, in some cases more than one pattern provided by PaMini was appropriate. For instance, there are a few patterns modeling a robot information request, differing in the confirmation strategy (implicit or explicit). Also, the system-initiated action may be cancelable or not, and the robot may ask for permission before action execution or not. For the label query, an explicit confirmation strategy was applied, whereas the grip name was confirmed implicitly because the grip names have been proven to be recognized reliably by the speech recognizer. The robot's grasping action was modeled as non-acknowledged yet cancelable action execution. Altogether, seven Interaction Patterns were required, one each for greeting, parting, the robot's label and grip query, the user's test questions and the robot's self-initiated grasping. They are listed in table 4.3.

For instance, to realize the robot's label query, the *Robot Information Request with Explicit Confirmation* pattern was used (cf. Appendix B), in which the robot asks a question (*R.question*), which the human answers (*H.answer*), whereupon the robot explicitly asks for confirmation of the given answer (*R.askForConfirmation*). Once the human confirms (*H.confirm*), the robot acknowledges (*R.acknowledge*). Additionally, the human has the opportunity to correct the given information at certain points. As for task communication, PaMini accepts the dialog task on the robot's question, updates it with the available information as soon as the human has answered the question, and completes it when the human confirms.

Subdialog	Interaction Pattern
Greeting	Human Interaction Opening (H Int Open)
Obtain label	Robot Information Request with Explicit Confirmation (R Inf Req Conf)
Obtain grip	Robot Simple Information Request (R Inf Req)
Grasping	Robot Self-Initiated Cancelable Action (R Cancel-Action)
List objects	Human Information Request (H Inf Req)
Grip test	Human Information Request (H Inf Req)
Parting	Human Interaction Closing (H Int Closing)

Table 4.3: Interaction Patterns for the Curious Robot scenario with PaMini.

Listing 4.6 shows the associated dialog act configuration. It determines conditions for the human dialog acts and how exactly the robot dialog acts should be expressed, both being possibly multimodal. The dialog act *R.question* in state *initial*, for example, is specified as the utterance *What is that?*, combined with a pointing gesture (lines 3-6). Similarly, in order to be interpreted as *H.answer*, the XML representation of the user utterance has to match the XPath expression[1] */utterance/description* (lines 8-9).
Apart from the dialog acts, the developer has to configure the task communication (i.e. the task specification for tasks initiated by the dialog system, and possible task state updates), as well as the definition of variables (used for parameterizing the robot's dialog acts and within the task specification). While the dialog act configuration is written in the domain-specific XML configuration language, the latter two are specified by extending Java base classes. Since not each Interaction Pattern involves task communication or the use of variables (for example, simple patterns such as *Human Interaction Opening* typically don't), only the dialog act configuration is obligatory.
The definition of variables and the task communication often go hand in hand. For instance, when the human answers the robot's information request (*R.question* and *H.answer*, respectively), the object label is extracted from the user utterance and assigned to the variable *label*, which is then used to parameterize the robot's confirmation request *R.askForConfirmation* (e.g. 'Apple. Is that correct?'), and to augment the task specification with the label so as to transfer it to the responsible system component.
Further, the developer has to configure which patterns may be interleaved with each other. In order to model the focus shifts and multitasking capabilities demanded by the user's test questions, each pattern must be intereavable with the patterns for the subdialogs *List objects* and *Grip test*. Technically, pattern interleavability is configured in an additional configuration file.

1 The XPath expressions in the example are kept simple. In practice, more complex expressions are common that combine several sub-expressions using negations, disjunctions or conjunctions. With such kind of expressions, a mock speech understanding process can be achieved, as has been done in the Curious Flobi scenario presented in chapter 7.

1.41.2

	Utterance	Comment	Interaction Pattern
U1	Hello.		H Int Open
R1	Hi, hello.		H Int Open
R2	What is that?	Back-end publishes interaction goal *label*	R Inf Req Conf
U2	This is a lemon.		R Inf Req Conf
R3	Melon. Is that correct?		R Inf Req Conf
U3	No, it's a lemon!		R Inf Req Conf
R4	Lemon. Is that correct?		R Inf Req Conf
U4	Yes, this is correct.		R Inf Req Conf
R5	How should I grasp the lemon?	Back-end publishes interaction goal *grip*	R Inf Req
U5	With the power grasp.		R Inf Req
R6	Alright.		R Inf Req
R7	I am going to grasp the lemon.	Back-end publishes interaction goal *grasp*	R Cancel-Action
		Grasp task *initiated* by PaMini	R Cancel-Action
R8	I start grasping now.	Grasp task *accepted* by back-end	R Cancel-Action
U6	How do you grasp a banana?	R Cancel-Action interleaved with H Inf Req	H Inf Req
R9	Sorry, I don't know.	to realize focus shift and multi-tasking	H Inf Req
U7	Stop!	*Cancel requested* by PaMini	R Cancel-Action
R10	Ok, I stop.	Grasp task *canceled* by back-end	R Cancel-Action
U8	Goodbye.		H Int Closing
R11	Goodbye.		H Int Closing

Table 4.4: An example dialog for the Curious Robot implementation with PaMini. For each utterance, the associated Interaction Pattern is given; see table 4.3 for their abbreviations. The interaction features focus shifts, multitasking, and cancellation of an on-going grasping action.

In addition to the necessary Interaction Patterns, a background process is required that triggers the patterns initiated by the robot. For the case study, the background process was implemented as a Java process that first sends the interaction goal *label*, then *grip* and *grasp* to the dialog manager via the XML-based middleware XCF [WHBS04]. In contrast to the implementations with Ravenclaw and Collagen, PaMini does not apply a polling strategy to retrieve the interaction goal, but it is notified on the respective event.

As the Curious Robot has been one of the development scenarios for PaMini, it is not surprising that all of the stipulated requirements could be met. With the Task State Protocol, state updates of temporally extended back-end calls such as grasping are delivered by event notification, enabling PaMini to give feedback on the on-going action, as illustrated

```
<patternConfiguration name="LabelQuery">

   <robotDialogAct state="initial" type="R.question">
      <verbalization text="What is that?"/>
      <point coordinatesXpath="//Region"/>
   </robotDialogAct>

   <humanDialogAct state="asked" xpath="/utterance/description"
      type="H.answer"/>

   <robotDialogAct state="asked" type="R.askForConfirmation">
      <verbalization text="%label%. Is that correct?"/>
   </robotDialogAct>

   <robotDialogAct state="awaitConfirmation" type="R.askForConfirmation">
      <verbalization text="%label%. Is that correct?"/>
   </robotDialogAct>

   <humanDialogAct state="awaitConfirmation" xpath="/utterance/description"
      type="H.correct"/>

   <humanDialogAct state="awaitConfirmation" xpath="/utterance/negation"
      type="H.negate"/>

   <robotDialogAct state="awaitConfirmation" type="R.question">
      <verbalization text="Ok, and what's correct?"/>
   </robotDialogAct>

   <humanDialogAct state="awaitConfirmation" xpath="/utterance/confirmation"
      type="H.confirm"/>

   <robotDialogAct state="awaitConfirmation" type="R.acknowledge">
      <verbalization text="%label%. Alright. "/>
   </robotDialogAct>

</patternConfiguration>
```

Figure 4.6: PaMini's dialog act configuration for the Robot Information Request with Explicit Confirmation, which was required to realize the robot's label query.

in table 4.4 (R8, R10). Conversely, PaMini can update or cancel tasks on-line (U7-R10). By admitting interleaving Interaction Patterns, the interaction can be maintained during task execution. During the robot's grasping action, for instance, the user initiates a focus shift by asking a question (U6-R9), which is modeled by interleaving a *Robot Self-Initiated Action* pattern with a *Human Information Request*.
Perhaps the most striking difference to the other dialog frameworks affects discourse planning. While *local* discourse planning is determined by the Interaction Patterns, *global* discourse planning – i.e., how the patterns are combined – is not done within the dialog framework, but is decided by the back-end (or by what the user says, of course). This enables the dialog system to respond to the dynamic environment in a flexible and reactive

way (cf. also section 3.8.2 for further discussion).
While the other frameworks discussed provide generic strategies for grounding or collaboration, PaMini goes one step further in this respect by providing pre-modeled "building blocks of interaction", intended to encapsulate the subtleties of dialog management and domain integration. Both a usability test (see section 4.1) and experiences within numerous scenarios (see section 5) support that Interaction Patterns enable developers to rapidly implement new interaction scenarios. Furthermore, as the Interaction Patterns are kept self-contained, new features can be added without breaking existing functionality (of which one is running the risk e.g. with Dipper's update rules). This significantly eases incremental system development.

In PaMini, the definition of variables is not as straightforward as e.g. with Ravenclaw, where variables are derived directly from the semantic speech recognition grammar. PaMini, in contrast, leaves variable handling to the developer. Variables can be extracted either from user or system input using XPath expressions, and the developer has to write the necessary code for this purpose, using base classes the framework provides. On the other hand, with this approach, the dialog manager does not rely on a specific method for speech recognition. In fact, PaMini is being used with several speech recognition and understanding modules in different scenarios. The first iterations of the Curious Robot scenario, for example, employed a speech understanding approach based on a frame-slot semantics [HWS06], while in the Receptionist Vince scenario (cf. section 8.1), a different speech understanding component is used, provided by a related working group. In contrast, later iterations of the Curious Robot work directly on the speech recognizer result in order to achieve robust keyword spotting.
Also, as PaMini outsources much of the responsibility for discourse planning to the back-end, little support is provided for interactions whose structure is determined by the current dialog state (or the current *information state*) rather than by an "intelligent back-end". This might be a hassle in information negotiating scenarios, in which the back-end typically plays a more passive role. Related to that, as PaMini does not maintain an explicit representation of the information that needs to be gathered during the dialog, overanswering is not supported. This deficiency can be illustrated using the Sports Companion scenario (cf. section 8.6). In this scenario, a robot is supposed to assist the user in developing an exercise plan. The plan includes seven exercise units of different types, which the user has to distribute over the week. For each unit, week day, time slot and type has to be specified, while satisfying several constraints. For example, intensive units are not allowed to be scheduled on two consecutive days. In order to allow the user to specify more than one information item at a time (e.g. "On Monday, 7pm, I would like to have an interval unit."), each possible combination of information has to be pre-programmed, i.e. for each combination, a pattern instance needs to be configured. In contrast, using an explicit representation of the required information, together with a meta-algorithm such as the VoiceXML Form Interpretation Algorithm (FIA) as discussed in section 2.1.2, information gathering could be handled in a generic way. Overall, PaMini

can be referred to as an *action-oriented*, rather than as *information-oriented* approach, relying on tasks rather than on information structures as the fundamental concept that drives the interaction.

4.0.5 Discussion

These case studies were performed with the goal to contrast state-of-the art approaches to dialog modeling, and to identify pitfalls and potential remedies for dialog modeling on robots. Table 4.5 lists the distinctive features and summarizes the results of the case studies.

Four state-of-the art approaches were investigated. Ravenclaw and PaMini can be referred to as descriptive approaches, whereas Collagen/Disco and Dipper fall into the category of mental-state approaches. Consequently, only the latter two feature planning or plan recognition capabilities. For the descriptive approaches, a visualization of the dialog flow is feasible and would facilitate dialog design, but this feature is only supported by PaMini. The descriptive approaches keep dialog and task structure well separated. To do so, Ravenclaw employs a domain-independent dialog engine, together with a domain-specific dialog description. PaMini relies on a fine-grained Task State Protocol as interface between dialog and domain level. Collagen/Disco automatically generates the system utterances based on a task model and the current discourse state. In this respect it is similar to PaMini, which combines task states with robot dialog acts. However, PaMini operates at a more abstract level than Collagen/Disco. Also, Collagen/Disco does not allow to configure the dialog structure (except for, with limitations, the exact wording), which PaMini allows through providing a large selection of different interaction patterns. Thus, in the Collagen/Disco framework, the dialog structure emerges directly from the task structure. In general, plan-based approaches inherently tend to keep dialog and domain less separated as they often rely on the same mechanism both for task and dialog planning (though the ones investigated in the case studies do not so).

The dialog configuration is written either in programming languages (such as C++ or Java), or using a domain-specific language (often XML-based) that has been developed for the specific purpose. Here, a trade-off between flexibility and complexity has to be found. The same is true for the back-end specification, which plays a larger role in robotics as it does in traditional domains.

Also, aspects regarding system integration are of particular importance of robotics, where the dialog manager coordinates with the complex robotic system. The question of how to model the interaction with the back-end tends to be solved individually by each framework. Binary success variables, as used in the reasoning-based Collagen/Disco approach seem to be somewhat underspecified for a satisfying information behavior of the robot. The user-defined result frame allows for more freedom but also imposes much knowledge and work on the developer. From this perspective, PaMini's Task State Protocol appears to be

	Ravenclaw	*Collagen/Disco*	*Dipper*	*PaMini*
Type of approach	Descriptive	Plan-based	Plan-based	Descriptive
Plan recognition, planning	No	Yes	Yes	No
Visualization	No	No	No	Yes
Relationship between dialog and task structure	Separated	Dialog structure emerges from task structure	Separated	Separated
Domain-specific configuration	C++ Macros	XML	Prolog-like Dipper update language	Java and XML
Back-end specification	Arbitrary components	JavaScript	Arbitrary components	Arbitrary components
Discourse planning	Task tree	Recipes	Information state update rules	Locally: Interaction Patterns, globally: Back-end
Communication with Back-end	User-defined result frame	Optional binary *success* variable	Interagent Communication Language	Task State Protocol
Asynchronous Coordination	No; Latest version: Yes	No	Yes (polling-based)	Yes (event-based)
Pre-modeled conversational skills	Grounding and repair	Collaborative plan execution	No	Patterns for various situations
Grounding model	Explicit	None	None	Implicit
Focus shifts	Yes	Yes	With limitations	Yes
Multimodality	Yes	No	Yes	Yes
Multi-party interaction	Yes, n systems:1 user	No	No	Yes, 1 system:n users

Table 4.5: Distinctive features of dialog modeling approaches.

a good compromise between both, allowing the developer an easy and standardized yet flexible interaction with the back-end.

In robotics, the discourse planning is affected not only by the user's utterances but also by the perceptual context. Contrary to the other approaches discussed, PaMini outsources global discourse planning to the back-end, allowing for a less restricted dialog structure. Discourse planning can be executed either by a centralized back-end process (i.e. a planner) or in a distributed way, resulting in a more reactive architecture.

Asynchronous coordination has been identified as crucial for integrating action execution and interaction. Whether a dialog system is able to handle asynchronous action execution depends largely on the middleware used and whether it supports event notifications or not.

Most dialog frameworks provide some kind of pre-modeled conversational skills, in form of generic strategies for grounding and repair, or for collaborative plan execution, or else in form of PaMini's interaction patterns. Dipper does not provide any pre-modeled dialog strategies. It could, thus, be referred to rather as a toolkit to build dialog frameworks than as a complete dialog framework. Also, focus shifts are supported by most frameworks. Internally, the discourse is typically represented as a stack, with the focused dialog element being on top. With Dipper, which does not maintain such a built-in structure, focus shifts could be implemented only with limitations, at the price of introducing several control flags. Further, the only framework that provides a generic grounding strategy is Ravenclaw. In PaMini, grounding is incorporated implicitly in the structure of the interaction patterns (cf. also section 3.8.2 for further discussion), whereas neither Dipper nor Collagen/Disco support grounding.

In order to keep our case studies simple, we have limited the target scenario to verbal interaction. Nevertheless, nonverbal behaviors and multimodality are crucial aspects in situated dialog. Except for Collagen/Disco, which relies on text in- and output, multimodality could have been realized with all of the discussed dialog managers, as they operate at the semantic level below which the in- and output sources may be exchanged. The new version of Ravenclaw supports multimodal in- and output by providing agents for modality integration and for the production of multimodal output [RE07]. Both Dipper and PaMini rely on a distributed architecture with arbitrary sources for in- and output. PaMini, for instance, provides a collection of available output modalities such as pointing gestures or mimics (depending on the robot platform), that can be combined. However, neither Dipper nor PaMini handles the issues of input fusion and output synchronization. Moreover, human-robot interaction demands more than classical 1:1 interactions. Often, the robot will be situated in environments where multiple possible interaction partners are present, or a robot might even have to collaborate with other robots. Thus, the capability of multi-party interaction is another crucial requirement. PaMini has recently been extended to be able to manage multiple interactions (with multiple participants each), and a multi-party engagement model [BH09] has been integrated in a Multi-Party Quiz game (see section 8.3). Ravenclaw has provisions for the opposite case, in which multiple robots collaborate, forming a team [DHB+06].

4.1 A Usability Test

This section reports on a usability test in which developers unfamiliar with the PaMini framework were asked to build a human-robot interaction scenario with it. The usability test combines two usability methods proposed by Nielsen [Nie94]. First, **performance measurement** is applied to measure the efficiency of use and the learnability of the framework. Efficiency is quantified by the time users take to complete a given task, by the number of tasks that can be completed within a given time limit, and by the number of tasks that can be completed within a given time limit. Learnability is quantified by comparing the measures for similar tasks at the first time and at a later stage. Second, **user observation and thinking aloud** techniques reveal potential misconceptions of the PaMini API and suggest new features.

4.1.1 Experimental Setup

Participants were classified either as *roboticists* or *non-roboticists*, each group consisting of four individuals. Classification was based on the participants' statements about previous knowledge on robotic architectures, both in general and in-house, as well as the Task State Protocol as described above. However, all participants were unfamiliar with the dialog manager.

Having acquainted with the system by reading the documentation for 10-15 minutes, participants were asked to solve a list of tasks with one hour given as time limit. The tasks were given in abstract textual form and had to be broken down by the participants into subtasks, such as selecting the appropriate Interaction Pattern, writing the dialog act configuration using the XML configuration language, possibly writing additional variable or task configuration in Java, registering the pattern with the dialog manager and finally testing the produced code using a prepared simulation. Participants were instructed to solve the tasks autonomously. The experimenter was available for specific questions, though, and intervened if problems occurred that concerned general issues such as Java, XML or the IDE rather than the Interaction Patterns itself. For each task, the time was recoreded that the participant took for solving it completely or up to a certain proportion. A task was considered to be solved 100% if the source code was completed and tested successfully, 75% if it was untested or slightly incomplete, and 50% if it exhibited substantial incompletenesses or if the participant gave a detailed oral description of a possible solution.

In detail, five tasks with increasing complexity were given. Task 1 and 2 were designed to be fairly simple and consisted of implementing interaction opening and end respectively, using the Human Interaction Opening and Human Interaction Closing. Task 3 was to realize a navigation command that might be rejected or fail and be canceled by the human at any time. The appropriate pattern for this task is the Human Cancellable Action Request. Task 4 required integration of a power management component that generated notifications whenever the battery level falls below a critical value. The required Robot Notification as such is very simple, but the complexity of this task lies in creating a context variable, allocating it to the current charge level and using it to parametrize the robot's warning. Task 5 consisted of having the robot ask persons for their name using the Robot Correctable Information Request. This required augmenting the task specification with the person's name in order to realize the information transfer to the responsible system component. Table 4.6 shows an overview of the given tasks. The full task instruction can be found in appendix C.

Task	*Interaction Pattern*	*# DA*	*Challenge*
1) Greeting	Human Interaction Opening	2	
2) Parting	Human Interaction Closing	2	
3) Navigation instruction	Human Cancellable Action Request	11	Task communication
4) Low battery warning	Robot Notification	1	Task communication, Variable definition, Parametrized output
5) Acquire person name	Robot Correctable Information Request	6	Task communication, Variable definition, Task specification update

Table 4.6: Overview of the tasks given in the usability test, including the required number of dialog acts, the major challenge of the task, and the required Interaction Pattern.

4.1.2 Results and Observations

Performance Measurement

Within the one hour time limit, all participants were able to solve task 1 and 2, and none proceeded up to task 5 as shown in table 4.7. Task 3 which exhibited a considerably higher degree of difficulty than task 1 and 2 could be solved by seven out of eight participants. Remarkably, all of the non-roboticists were able to solve it, even though the required Human Cancellable Action Request involves complex domain integration using the Task State Protocol. This result suggests that, first, the Task State Protocol abstracts from integration details in an intuitive way and, second, that the graphical representation of the interaction describes linking domain and dialog level in an understandable way.
As shown in table 4.8, task 2 could be solved considerably faster than task 1, with 26.75 minutes on average compared to 9 minutes, though possessing the same degree of difficulty. This suggests that once participants got accustomed to the API, they use it fairly effectively, taking 9 minutes at average for a simple pattern like Human Interaction End and 20.28 minutes for a more complex pattern like Human Cancellable Action Request.
In general, non-roboticists took slightly more time for each tasks. This applies to task 1 and task 2 as well, even though these tasks do not include any interfacing to the robotic subsystem. This result cannot be explained with the roboticists group exhibiting better programming skills in general: both groups rated their general programming skill with 3.5 on a scale from 1 to 5.

User Observation and Thinking Aloud

Apart from performance measurement, the participants were asked to continuously verbalize their thoughts while using the system, which provided insights into the developers' view of

the dialog manager. For instance, it was interesting to observe the participants' reaction faced with the graphical representation of the Cancellable Action Request pattern required for task 3, which is one of the most complex Interaction Patterns, consisting of 11 state transitions. While most of the non-roboticists were at first overwhelmed by the complexity, one of the roboticists (a very experienced scenario developer) became almost enthusiastic. A possible interpretation for this might be that the robiticist are already aware of the high integration complexity of a cancellable instruction, but that they feel it becomes well manageable through the visualization and the Task State Protocol. However, in the end, the non-roboticists were able to manage the task even more successful than the roboticists (cf. table 4.7), though slightly slower (cf. table 4.8).
The Human Cancellable Action Request gave us the opportunity for another valuable observation concerning the pattern visualization, because it has both system events as input and robot dialog acts as the resulting output. It could be observed that the robotic experts oriented themselves using the system event names, e.g. *accepted*, while the robotic novices rather oriented by more the dialog act names, e.g. *R.assert*. It can be concluded that using this combined notation supports both the robotic system engineer and the interaction designer perspective.
Besides, the thinking-aloud method enabled us to identify potential deficiencies and misconceptions. It lead to a number of API improvements, including more precise method naming, clearer syntax for the configuration language and additional convenience methods.

Task	*Percentage*	*All developers*	*Roboticists*	*Non-Roboticists*
1) Greeting	100%	8	4	4
2) Parting	100%	8	4	4
3) Navigation instruction	75%	8	4	4
	100%	7	3	4
4) Low battery warning	50%	4	3	1
	75%	2	1	1
	100%	1	1	0
5) Acquire person name	100%	0	0	0

Table 4.7: Number of subjects that solved the respective task up to the given percentage

Task	*Time in minutes*	*Roboticists*	*Non-Roboticists*
1) Greeting	26.75	25.75	27.75
2) Parting	9	8.75	9.25
3) Navigation instruction	20.28	18.66	21.5
4) Low battery warning	12	12	na
5) Acquire person name	na	na	na

Table 4.8: Average time (in minutes) needed to completely solve the respective task. Only those participants were considered who solved the respective task 100%.

4.1.3 Discussion

The evaluation focused on the question of whether this approach enables developers to whom the system is unknown to implement new interaction scenarios in short time. The results from the performance measurement showed that both roboticists and non-roboticists were able to complete 3 out of 5 dialog programming tasks of increasing complexity within one hour. Although roboticists were slightly faster than novices, both groups showed a steep learning curve in the second task, indicating that the concepts are easy to learn. Results from the user observation support this interpretation: while roboticists tended to rely on the concepts related to the system task protocol, that is the internal processing of the robot, non-roboticists focused on the dialog acts, that is the surface structure of the dialog.

Part II

Applications of the Proposed Approach

5 Scenario-Based Design

5.1 Development History of the PaMini framework

The approach that underlies the PaMini dialog framework has not been developed from scratch. It rather evolved over time, based on experiences with previous approaches – and their shortcomings. Essentially, there were two predecessor systems that influenced the concepts of the suggested PaMini approach, carrying the working title Sunshine dialog and Moonlight dialog. In order to illustrate the evolution of PaMini, and to contrast the approaches, the previous approaches will be outlined in the following.

The **Sunshine dialog** system is based on a computational model of grounding that has been presented by Li [Li07]. In Li's grounding model, the grounding unit is an *Exchange*, which is formed by two dialog acts. The first one plays the role of a *Presentation*, and the second one represents the *Acceptance* of it. Between *Exchanges*, a grounding relation exists that specified how the grounding of the present *Exchange* affects its mother *Exchange*. For example, the *Support* relation indicates that the present *Exchange* facilitates the grounding of the mother *Exchange* by providing more information, and that the grounding of the mother *Exchange* should be retried, given the new *Exchange*.
Figure 5.1 illustrates how the robot's label query was modeled based on Li's grounding model. By asking for the label, the robot makes a *Presentation*, which is accepted by the human's answer so that the *Exchange* (R1, H1) can be grounded. The robot's confirmation request and the human's reply forms an *Exchange* (R2, H2) that is connected with its mother *Exchange* via the Default grounding relation, likewise the robot's request for the correct label (R3, H3). Again, the robot asks for confirmation of the new label (R4), but the human's reply can not be interpreted (H4). Thus, the robot makes a new *Presentation* by asking for clarification (R5), which stands in Support relation to its mother. The human repeats the answer (H5), which grounds the clarification request (R5). As the grounding relation to the previous *Exchange* is Support, it also assists in the grounding of the robot's original *Presentation* (R4). Finally, the robot makes a *Presentation*, stating that it will learn the label (R6), and accepts it itself as learning succeeds (R7).
Li's grounding model and the resulting Sunshine dialog system has several shortcomings, in particular regarding usability. From a practical perspective, implementing new interaction scenarios with it is extremely tedious: for each possible *Presentation* and each possible input, it needs to be specified which functionality the input has with respect to grounding. This requires the interaction designer to have intimate knowledge of the proposed grounding model. Even if this is the case, it is not a trivial task to map all

possible interactions onto the domain of the proposed model. For example, the grounding model – whose basic units are pairs of dialog acts – does not serve well in describing dialog act triples (e.g. “What’s that” – “An apple” – “Alright, an apple”). The basic problem is that, unlike other computational models of grounding, such as the one proposed by Traum [Tra94], Li’s model does not account for larger conversational structures that go beyond pairs of *Presentations* and *Acceptances*.
Another basic problem is the chosen abstraction level: the only abstraction the approach makes is in terms of admissible sequences of *Presentations* and *Acceptances* – an abstraction that is too general to be useful for dialog management. In contrast, Traum’s grounding actions, which are comparable to Li’s grounding relations, are more fine-grained and thus more informative than these, and Traum’s model additional includes an intermediate abstraction level that models speech acts, such as *inform*, *suggest* or *accept*.
Also, while the principle of grounding, i.e. the assumption that in an interaction both partners attempt to ground open *Exchange*, is interpreted as the basic mechanism that drives interaction, the grounding model does not account for *what* is grounded, i.e. it provides no means to represent the newly acquired information.

The successor system, the **Moonlight dialog** system, has been developed by the present author and can be viewed as an interim system which already incorporates some of PaMini’s concepts. Technically, it is a refactored version of the Sunshine dialog. In this system, the internal dialog management still relies on Li’s grounding model, but its external interface, i.e. the domain integration (which was also a shortcoming of the Sunshine

R1	What is that?	Pre
H1	An apple.	Acc
	Grounding relation: Default	
R2	Banana. Is that correct?	Pre
H2	No!	Acc
	Grounding relation: Default	
R3	What’s correct?	Pre
H3	Apple.	Acc
	Grounding relation: Default	
R4	Apple. Is that correct?	Pre
H4	*non-understandable*	Unqualified
	Grounding relation: Support	
R5	Pardon?	Pre
H5	Yes.	Acc
	Grounding relation: Default	
R6	OK, I am going to learn the apple.	Pre
R7	I have learnt the apple.	Acc

Figure 5.1: Grounding-based dialog modeling in the Sunshine dialog

system), was revised and addressed in a systematic manner for the first time. In particular, the Moonlight system makes use of a first version of the Task State Protocol. Thus, with the Moonlight dialog system, the new interface could already be applied and tested in a number of scenarios, while new concepts for internal dialog management matured. At the same time, a first generalization of the dialog flow was made by mapping task states onto *Presentations* or *Acceptances* and the appropriate grounding relation. In the example dialog shown in figure 5.1, for example, the robot's confirmation of learning (R7) is made when the associated task is completed. In the Moonlight system, the appropriate grounding information was automatically determined from the task state.

Moonlight's successor is the **PaMini** framework as described within this thesis. As already in the Moonlight system, its external interface is specified through the Task State Protocol, which in the meantime has been investigated more systematically and extended for several new states. As for internal dialog modeling, the grounding-based model was replaced by the concept of Interaction Patterns. Their development was essentially influenced by the observation that conversations are composed of larger recurring structures that go beyond Li's grounding structures. As already prototyped in the Moonlight system, the Task State Protocol links the conversation level and the domain level, but the mapping from task states to grounding states was replaced by a mapping from task states to robot dialog acts. Also, the PaMini framework explicitly accounts for information transfer to the robot subsystem by making use of the new task states *intermediate_result* and *update* within the Interaction Patterns that deal with information negotiation.

5.2 Overview of the Implemented Scenarios

Essential for the development of the concepts that underly the PaMini framework was not only the iterative process described above, but also a number of accompanying scenarios that provided a wide range of use cases – and pointed out the deficiencies of the existing approaches. This is in line with the general guidelines for framework development, suggesting that abstractions are developed by generalizing from concrete examples, and that developing three applications in order to gain intimate knowledge of the domain will be worthwhile[1] [RJ96].

Figure 5.2 gives an overview of the scenarios that either have contributed to the development of PaMini, or were implemented based on PaMini. As shown in the figure, two scenarios have served as examples: the Home-Tour scenario[2] and the Curious Robot

1 In fact, in addition to the Curious Robot and the Home-Tour scenario, a third example scenario has been developed in which a robot acts as visitor guide, using a real map of the environment that is shared with the user [BSL+08].

2 Different to what is suggested by figure 5.2, the beginning of the development of the Home-Tour scenario was earlier than 2007, but this was the time when the author joined the developer team.

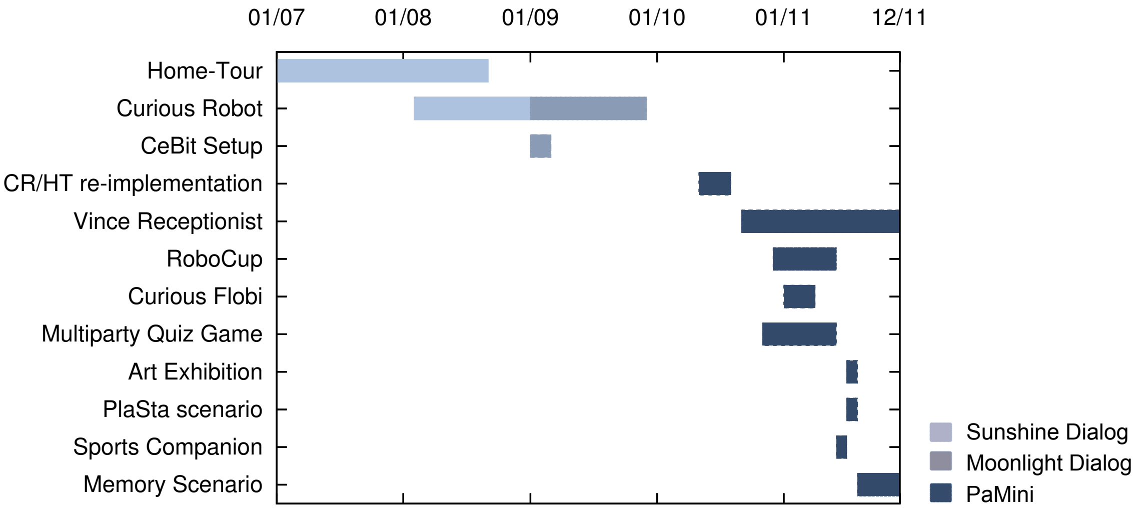


Figure 5.2: Overview of the scenarios implemented with the PaMini dialog framework and its predecessors.

scenario. Both were initially implemented with the Sunshine dialog system, but during the development process of the Curious Robot scenario, the Sunshine system was replaced by the Moonlight system. Also the CeBit setup, a stripped-down version of the Curious Robot, was implemented with the Moonlight system. These preliminary scenarios, that helped to form the proposed concepts, will be outlined in chapter 6.
As soon as the development of PaMini was finished, the Home-Tour and the Curious Robot were re-implemented based on PaMini as a proof of concept. In addition to that, a further iteration of the Curious Robot – the Curious Flobi – was realized with PaMini and evaluated extensively. It is described in chapter 7.
The scenarios mentioned so far were implemented by the present author herself. Additionally, a number of further scenarios were implemented by different developers, either by student assistants or in the scope of courses or bachelor's and master's theses. This demonstrates the understandability and usability of the framework. Moreover, several of these have been implemented in a very short time, notably the Art Exhibition scenario, the PlaSta scenario and the Sports Companion, with at least the first two being being fully integrated robot systems. This demonstrates that PaMini successfully supports rapid prototyping of interaction scenarios. The additional scenarios are briefly described in chapter 8.

Table 5.1 lists for all scenarios implemented with PaMini the Interaction Patterns used. This information describes not only the complexity, but also the characteristics of a scenario. In this respect, interesting differences can be found between the individual scenarios. For example, the RoboCup@Home scenario makes extensive use of action-related Interaction Patterns, while other scenarios, such as the Receptionist Vince, rely almost exclusively on information-oriented patterns. This suggests that both types of interactions,

action-oriented and information-oriented, can be realized with PaMini.
The pattern usage also allows to draw conclusions about the distribution of initiative in a scenario. The (information-oriented) Curious Flobi and the (action-oriented) RoboCup scenario can be characterized as mixed-initiative: actions and information requests, respectively, can be initiated both by the human and by the robot. In the Receptionist scenario, it is exclusively the human who asks for information, whereas both in the Sports Companion and in the Quiz Game scenario, the robot is the one who requests information from the human. Interaction is opened either by the human only (Curious Flobi, RoboCup, Quiz Game), or by the robot only (Art Exhibition, Sports Companion, Memory Game), or by both (Receptionist, PlaSta). Having the robot open the interaction often involves a perception component that recognizes the presence of a potential interaction partner.
Object-related patterns have so far been used only in the Curious Flobi scenario. Even though object learning is addressed in the RoboCup scenario as well, it does not feature reference resolution. Thus, object learning is realized using general action patterns in the RoboCup scenario.
From the general patterns, robot notifications are used most often. They are extensively used in the PlaSta scenario and the Sports Companion scenario, mostly to give the users feedback without expecting an answer, e.g. about their performance or about remaining time in a time-critical task. Statements (on which a reply is expected) are also common. Like notifications, they do not include task communication and are therefore used for sub-interactions that are irrelevant for the overall system. In the Curious Flobi system, for instance, they were used to model the *Exchange* of pleasantries.
All scenarios make use of the built-in clarification patterns. The default clarification strategy, albeit simply, appears to be sufficient, as it is not extended or overwritten.

Interaction Pattern	***Receptionist Vince***	***RoboCup@Home***	***Curious Flobi***	***Multiparty Quiz Game***	***Art Exhibition***	***PlaSta Scenario***	***Sports Companion***	***Memory Game***
H Cancel-Action Req		3						
R Cancel-Action		3						
H Action Req		2						
H Action Req Conf		2	1					
R Action		3						
R Cancel-Action Conf		1						
H Cancel-Action Req Con		4						
Σ Action Patterns	0	18	1	0	0	0	0	0
R Cor-Inf Req								
H Inf Req	5		1					
R Inf Req				3			2	
R Inf Req Conf		3					3	
R Rej-Inf Req Conf								
R Rej-Inf Req Conf Task								
R Inf Req Conf Task Ack			1					
Σ Information Patterns	5	3	2	3	0	0	5	0
H Obj Test								
H Obj Demo Conf			1					
H Obj Demo			1					
Σ Object Patterns	0	0	2	0	0	0	0	0
H Int Open	1	1	1	1		1		
H Int Close	1		1	1				
H Int Reset	1		1				1	
H Sys Reset	1							
R Int Open	1				1	1	1	1
Σ Interactional Patterns	5	1	3	2	1	2	2	1
H Statement	6	5	9			1		1
R Notification	9	6	3		6	23	10	7
R Statement	1						7	
R Suggestion	1			2	2	8	4	9
Σ General Patterns	17	12	12	2	8	32	21	5
R Ask Repeat	1	1	1	1	1	1	1	1
R Suggest Reset	1	1	1	1	1	1	1	1
Σ Clarification Patterns	2	2	2	2	2	2	2	2

Table 5.1: Overview on usage of the single Interaction Patterns in the different scenarios.

6 Preliminary Scenarios

This chapter describes the preliminary scenarios that have not yet been implemented based on suggested approach, but with the previously used dialog system. However, they provided plenty of use cases that assisted in gaining knowledge of the robotics domain, and of its specific challenges. Of course, the scenarios described in this chapter, and their underlying concepts, were not realized by myself alone, but in collaboration with colleagues: The Home-Tour scenario was developed in collaboration with Marc Hanheide, Frederic Siepmann, Elin Topp and Torsten Spexard, and the Curious Robot and the CeBit scenario and was developed in collaboration with Christof Elbrechter, Robert Haschke, Ingo Lütkebohle and Lars Schillingmann.
Although the scenarios and the hardware platforms they are running on are very different, three overarching themes can be identified: First, both scenarios deal with learning through interaction. Second, the scenarios have in common that they rely on a mixed-initiative dialog strategy. In particular, two facets of the robot's task initiative have been explored: how it facilitates learning, and how it facilitates the interaction as such. Third, from a technical point of view, they all rely on the Task State Protocol for communication between the dialog system and the back-end.

6.1 The Home-Tour: Jointly Building Up a Model of the Environment

The first example scenario that influenced the development of the suggested approach is the Home-Tour scenario. Two general concepts have emerged from it. First, it could be studied how robot initiative can be realized, based on information provided by a system component, and how interactive learning benefits from mixed initiative. Second, it provided a use case for system integration based on a first version of the Task State Protocol, which was used to give the user feedback during the robot's slow-going room exploration. The Home-Tour scenario is described in more detail in [PSS$^+$09].

6.1.1 Scenario Description

In the Home-Tour scenario a mobile robot assistant has to become acquainted with its working environment by interacting with a human during a guided tour. A basic requirement for the robot is being able to learn a spatial model of the environment and to integrate human and robotic representation. Previous iterations of the Home-Tour[1] based

1 The author joined the development team at a later stage of the scenario.

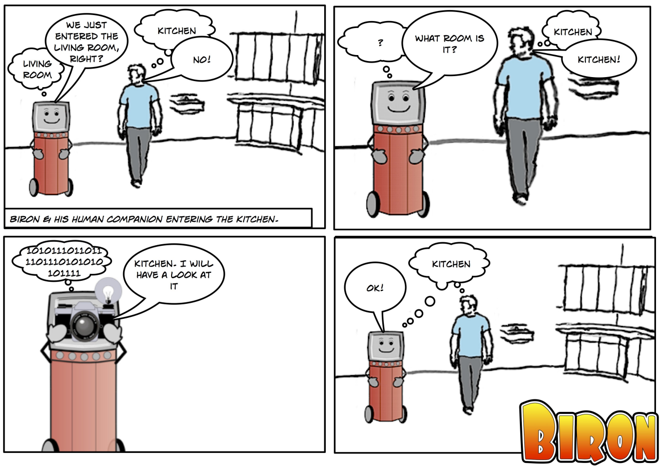


Figure 6.1: A scene from the Home-Tour scenario: The robot is taking the initiative in order to verify its (incorrect) hypothesis which leads the human to correct it. This comic was created by Florian Lier.

a lot on the human's initiative in a rather command-style fashion [LKF+04, LHW+05]. In particular, the human needed to show the robot around, explicitly demonstrating rooms and objects one after another. Thus, the human fully controls what the robot learns. As the scenario targets unexperienced users that only have very limited knowledge about the system and its internal model, the model they intuitively teach by their own initiative may be sparse and possibly erroneous.

Consequently, the main idea of the new iteration of the Home-Tour is to facilitate learning by mixed-initiative interaction. In particular, it focuses on the question of how the learning process can be improved by equipping the robot with abilities that enable *active learning*, so that the robot is capable of actively provoking a situation that provides new information instead of passively waiting until such a situation presents itself. This obviously has the potential to speed up and optimize the learning process.
Accordingly, on the one hand, the user can still take the active role and teach the robot rooms explicitly or check the learning process by asking monitoring questions. However, the more crucial aspect is that the robot is able to take initiative on its own and to advance the learning process by verifying existing information resolving uncertainty. This makes learning an interactive process, in the course of which information can continuously be refined. A typical scene from the Home-Tour in which an incorrect hypothesis is corrected is shown in figure 6.1. Table 6.1 lists example dialogs for each of the mentioned use cases.

Initiative	***Situation***	***Example dialog***
Human	Room teaching	H: This is the kitchen! R: Kitchen. I will have a look at it. R: (looking) R: OK!
Human	Test questions	H: Where are we? R: We are in the kitchen.
Robot	Verifying hypothesis	R: We just entered the hallway, right?
		a) Human agrees H: Yes. R: OK!
		b) Human corrects H: No, this is the living room. R: Living room. OK!
		c) Human disagrees *without providing correction* H: No! R: What room is it? H: The living room R: Living room. OK.
Robot	Resolving uncertainty	R: Are we still in the living room?
		a) Human agrees H: Yes. R. OK.
		b) Human corrects H: No, this is the kitchen. R: Kitchen. OK!
		c) Human disagrees *without providing correction* H: No! R: What room is it? H: The kitchen. R: Kitchen. OK.

Table 6.1: Example dialogs in the Home-Tour.

6.1.2 System Overview

The platform used in this scenario is the mobile robot BIRON (Bielefeld Robot Companion) shown in figure 6.2. BIRON is equipped with several sensors that allow perception of the current situation as a basis for interaction. In particular, it uses a pan-tilt camera for perception of the human interaction partner, stereo microphones and speakers for speech interaction, and a laser scanner that provides data for computing a representation of its environment.

The two major software components of the system are the dialog system – the grounding-based Sunshine Dialog described in section 5.1 – and a component for Human Augmented

Figure 6.2: The BIRON platform

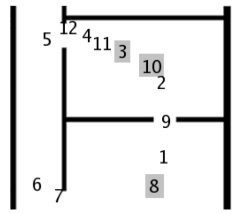


Figure 6.3: Spurious detections in the Home-Tour scenario (from [Top08]).

Mapping (HAM) which has been developed by Elin Topp [Top08] at KTH and integrated with the BIRON system in the EU project Cogniron. The HAM component maintains a representation of the robot's spatial environment that integrates a robotic map with human concepts that are communicated via the dialog system, such as a room[1]. The representations are calculated based on laser range data gathered during a 360° exploration turn. The HAM system continuously publishes hypotheses about the room the robot currently is in. The dialog system reacts to the hypotheses, resulting in the robot taking initiative. In case of human initiative, it is the dialog system that creates a room hypothesis, and the HAM system reacts to that either by creating a new concept (which involves an exploration turn), or by correcting an existing concept.

The coordination between the dialog system and the HAM component relies on a first version of the Task State Protocol, with the admissible state sequence *initiated*, *accepted/rejected*, *completed/failed*. Its initial purpose was to give the user feedback before the robot starts its slow-going exploration turn ("I will have a look at it"), triggered by the *accepted* state, because it has been observed in previous user studies that some users were irritated by the robot turning away. The usage of the task states, however, differ in two major aspects from their later usages.
First, tasks were often accepted and completed by different components. For example, when the human demonstrates a room, new room learning task is initiated by the dialog

1 Technically, the HAM system distinguishes between *regions* and *locations*. Regions can be roughly defined as rooms, whereas locations can be defined as larger objects that are contained in the regions. The described integration of dialog system and HAM applies only to region learning.

system based on the human's utterance. The task is accepted by the arbitration component now, indicating that it allows the HAM system to take control of the robot's driving motors. This triggers HAM to start the exploration by turning the robot around. At the same time, the dialog system is triggered to give verbal feedback about the processing state ("I will have a look at it."). As soon as the exploration is finished, it is the HAM system that sets the state completed. In consequence, the hardware arbitration is reset, the robot memorizes the new representation and the dialog system gives another verbal confirmation.
In contrast, the later usage of the Task State Protocol, in particular Lütkebohle's toolkit implementation [LPP+11], promoted a rather client-server based view, where a task is initiated by the client, can be observed by an arbitrary number of components which may react to state changes, but is executed by only one server. This is because knowing the components that are entitled to modify a specific task facilitates detection and recovery of race conditions. Considering this convention, the triadic interaction between HAM, dialog system and arbitration would rather be split up in several subtasks.
The second difference pertains to the task state transitions. The HAM component continuously updates the room hypothesis, which is tracked by the dialog system. If the current hypothesis changes, or becomes uncertain, the dialog system addresses the human for clarification. This requires the dialog system to maintain a history of hypotheses, and to use it as a basis for decision making. With the explicit *update* and *update accepted/failed* transitions, which were added later, the dialog system would be able to register directly on hypothesis changes, without maintaining their history.

6.1.3 Evaluation: Analyzing a Test Run

The system has not been evaluated in a real user study, but a test run has been analyzed as a proof of concept of the integration approach and the mixed-initiative interaction strategy. The evaluation focuses primarily on the performance of the HAM component. The data analysis has been carried out and published by Elin Topp [Top08], but it also allows to draw conclusions regarding the interaction strategy.

The test run was conducted in an office environment at Bielefeld University that can be compared to a part of an apartment, consisting of two adjacent rooms and a hallway. The rooms were labeled as "living room", "kitchen" and "hallway", respectively. The kitchen can be reached both from the hallway and the living room. This allows to do laps through the rooms, so that the system behavior can be investigated also when a known room is re-entered. Figure 6.3 shows the layout of the environment in which the test run was conducted.
The test run starts in the living room (the lower one in figure 6.3), then human and robot pass from the kitchen to the hallway, enter the living room again, and finally go from the living room to the kitchen a second time, where the example tour is finished. The human starts interaction by labeling the living room. Subsequently, when leaving the living room, the robot takes over initiative and requests the new room label ("We just left the living

room, right?"). The remaining interaction is driven mainly by the robot's clarification questions.
Overall the robot asked 12 times for a confirmation of a hypothesized transition. The respective locations are marked gray in figure 6.3. Five of these actually occurred in a situation where human and robot just left the room (4, 5, 7, 9, 12). Four more can be explained by human and robot being close to door passages (1, 2, 6, 11). The locations of the remaining three clarifications questions (3, 8, 10) are not plausible and due to a spurious hypothesis. In case of question 3 and 10, the robot's hypothesis has become uncertain and needs to be confirmed by the human (e.g. "Are we still in the kitchen?"), but in case of question 8, the robot has an incorrect hypothesis and assumes that they had left the living room ("We just left the living room, right?").

In general, this test run shows the systems's ability to build up a model of the environment through interaction with the human, and to continuously update and correct its internal representation. More specifically, from an interaction point of view, it demonstrates that a more correct model can be acquired when the robot attempts to fill information gaps on its own initiative. In particular, the robot's clarification requests at locations 3, 8 and 10 show that its internal model had become unstable and even incorrect and can actually be improved through the robot's capabilities to take initiative and to actively contribute to the learning process.

6.2 The Curious Robot: Exploring Salient Objects

While the previously described Home-Tour focuses on the question how mixed initiative facilitates learning, the Curious Robot scenario addresses the question how mixed initiative facilitates the interaction itself. From a technical perspective, the scenario has advanced the refinement of the Task State Protocol by providing new use cases. It can also be stated that the Task State Protocol has been applied more systematically in this scenario, and that it has been identified and investigated as a general coordination principle for the first time [Lüt11].

6.2.1 Scenario Description

The second scenario is an interactive object learning and manipulation scenario with a humanoid robot exploring objects that are interesting or salient for it, assisted by the human tutor. Experiences with the original Home-Tour – which, in contrast to the above described iteration, relied mainly on the human's initiative – show that untrained users require a significant amount of prior instruction to complete the task [LHRS09], because the robot's interaction model is not immediately obvious. Therefore, their behaviors and interaction strategies vary enormously, which makes it almost impossible for a system to cope with.

In the Curious Robot scenario, the interaction strategy consequently focuses on the question how mixed-initiative allows to structure the interaction for the users and thus make their behavior more predictable. In particular, the robot asks the user about object labels and how to grasp them. By asking about objects at its own instead of leaving it to the user to demonstrate them the robot provides guidance within interaction which in particular unexperienced users can benefit from. It also communicates what is interesting for it, which unexperienced users might not be aware of. Not least, with using robot initiative to determine the objects to learn, the error-prone visual analysis of human demonstration behavior can be bypassed. If for an object both the label and the appropriate grasping technique is known, the robot grasps it at its own initiative and puts it away. On the other hand, the user can trigger or abort a grasping action at any time. However, in contrast to the Home-Tour scenario and the later Curious Flobi scenario, users cannot demonstrate objects on their own initiative, but they can test the robot and check its knowledge by asking test questions about learnt objects and the grip technique appropriate for a specific object. An overview of the robot's interaction capabilities is given in table 6.2.

Initiative	***Situation***	***Example dialog***
Robot	Asking for label	R: What is that? ⟨pointing⟩ H: This is a banana.
	Asking for grip	R: How can I grasp the banana? H: With the power grasp.
	Grasping	R: I am going to grasp the banana. R: I start grasping now. R: ⟨grasping⟩ R: OK!
Human	Grasping instruction	H: Grasp the apple! R: OK. I start grasping now. R: ⟨grasping⟩ R: OK!
	Interrupting system	H: Stop! R: OK, I'll stop. ⟨stops grasping⟩ R: OK!
	Test questions	H: How would you grasp the apple? R: With the power grasp.
		H: What objects do you already know? R: I know the apple and the banana.
		H: What objects are present on the table? R: Two apples and one lemon.

Table 6.2: Example dialogs in the Curious Robot scenario.

6.2.2 System Overview

The platform used for the Curious Robot scenario consists of two Mitsubishi robot arms fixed to the ceiling, with a left and right Shadow robot hand attached, combined with an anthropomorphic robot torso in the background that serves as interaction partner. The

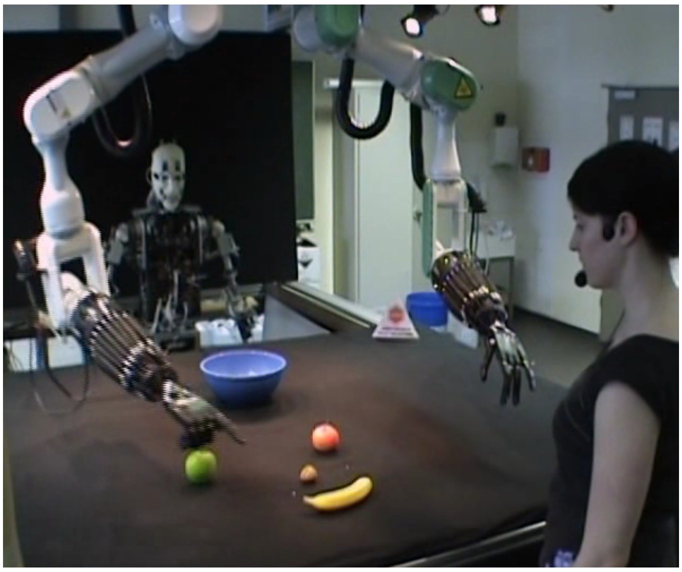


Figure 6.4: The Curious Robot setup.

setup is shown in figure 6.4. Sensors not visible in the figure are an overhead camera and a headset microphone.

The software system is composed of three subsystems for speech and dialog management, visual analysis, and motor activities. Initially, the Sunshine Dialog system was used for dialog management, but in the course of the development process, it was transparently exchanged by the Moonlight system, as already mentioned in section 5.2. The subsystem for visual analysis generates the robot initiative, e.g. asking for objects, based on visual bottom-up saliency. Based on a the saliency of a region an object is located in, and based on the context information known about the object (i.e. label and appropriate grip type), the vision subsystem proposes an interaction goal that the robot pursues: "acquire label", "acquire grip type", or "grasp". The interaction goals are initiated as a task executed by the dialog system, if the dialog situation permits this.
The motor subsystem controls grasping and performs pick-and-place operations using three basic grasp prototypes. Operations can be triggered via a task interface.
Thus, the coordination between the three subsystems relies exclusively on the Task State Protocol. In the course of the scenario development, the task life-cycle was being extended for the states *cancel*,*cancel accepted/failed*, *update*, *update accepted/failed* and *intermediate_result*, and a dedicated toolkit for requesting and monitoring tasks was developed [LHS$^+$10]. Regarding task coordination, the temporally extended grasping action provided an interesting use case. For the first time, tasks were split into subtasks: the interaction goal "grasp", proposed as a dialog task by the visual analysis, is executed by the dialog system, which initiates a grasp task for the action subsystem. Also, for the

first time, interleaving subdialogs were realized by admitting the user to ask test questions during an on-going grasping action. It also allowed to cancel an on-going action, based on the newly introduced task states.

6.2.3 Evaluation: A Video Study

The scenario was evaluated by means of a video study. Ten participants who had no prior experience with the system were asked to watch a video in which a person interacted with the system. The video was stopped at preset times, and users were asked what they would do in this situation. The questions were always asked after after the robot had acted, but before the person in the video reacted to it, to guarantee an unbiased answer. The evaluation originally aimed to explore several aspects: the expectation that is raised through the robot's appearance, how users interpret a faulty situation (where, for example, the robot points at an empty spot), how they would recover such situations, and on the effectiveness of the robot's guidance. The below description will focus on the latter, but a more detailed description of the study has been published in [LPS+09].

To investigate the effectiveness of the robot's guidance, two contrasting situations were compared, in which the robot's behavior did or did not call for a specific user reaction. The first situation was immediately after the robot's label query ("What is that?"), while the second one was after the grip query ("How should I grasp the ...?"). The first question can be answered very intuitively, by simply naming the object label, but the second question is somewhat confusing because it is not specified *which aspect of grasping* it refers to. This difference is well reflected by the results of the study, shown in table 6.3. The participant's replies to the first question were very consistent. Only three constructions were used, and they are all slight variations of each other. In contrast, the second question was much more open and accordingly yielded more variation in the user behavior. Answers referred to five fundamentally different aspects of grasping, and there were also many variations in the specific wording. Moreover, in the guided situation, participants answered quicker (5 seconds vs. 19 seconds, measured from end of question to end of answer) and required less clarification from the experimenter (1 vs. 5 participants) compared to the situation where the robot provides less guidance.

As an aside, the robot's grip query was intended to get results on how subjects intuitively describe grasping. It was observed that 7 out of 10 participants, perhaps unconsciously, complemented their verbal description of grasping with a gesture. This suggests that *demonstrating* an action is more natural than *describing* it. Consequently, this feature was added in a later iteration of the Curious Robot scenario, in which grasping may also be demonstrated non-verbally using a data glove [LPH+10].

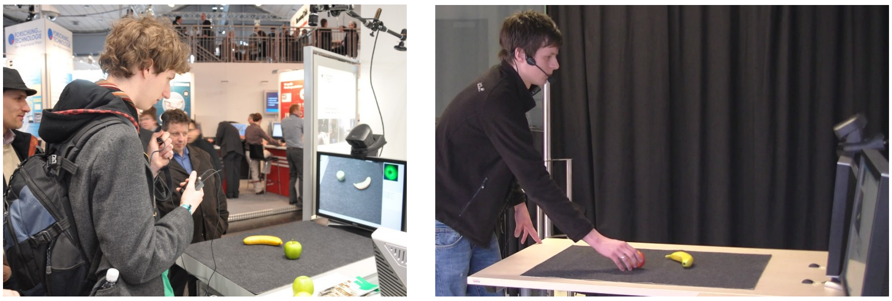
Situation	*Answer or Aspect Described*	*% of Participants*
"What is that?"	"That is a..."	70%
	"a ..."	20%
	"a yellow ..."	10%
"How should I grasp the ...?"	Effector position relative to object	30%
	Trajectory of effector	20%
	Fingers to Use	40%
	Force to Use	30%
	Grasp point on object	20%

Table 6.3: Replies after System Initiative

6.3 The CeBit Setup: A Stripped-Down Version of the Curious Robot

The present section describes the CeBit setup, a reduced version of the Curious Robot scenario. It was developed on the occasion the CeBit 2009 trade fair, where it proved its robustness under real world conditions. The system also underwent a user study which provided the opportunity for analyzing speech understanding performance, which has a major influence on the system's overall performance.

6.3.1 Scenario Description and System Overview



(a) Visitor interaction at CeBit 2009 (b) User study in the lab (from [Poh09])

Figure 6.5: The CeBit setup, a stripped-down version of the Curious Robot system.

A stripped-down version of the system has been developed to be exhibited at CeBit 2009, one of the world's largest IT fairs. The system was run there as an interactive live demo over six days, demonstrating its robustness in long-term use. The "robot' consisted of a computer, with loudspeakers and a pan-tilt camera attached to it, and an overhead camera as shown in figure 6.5. The overhead camera is used for perception of the table on which the objects to learn are placed, whereas the pan-tilt camera has purely interactional function, representing an "interaction partner".

Due to the absence of the end effectors, interaction was reduced to its object learning aspects. The interaction capabilities of the system included asking for an object label, listing the objects that it has learnt so far, and listing objects that are present at the table. All of them are initiated on the systems's initiative: the label query is triggered when there are objects left to learn, and the latter two are triggered at regular intervals once all objects are known. When asking for an object label, the referred object is shown on the screen. The pan-tilt camera provides both task-related and communicative feedback by focusing either at the interaction partner, or at the objects on the table. The control of the camera is based on the observation of the task communication and depends on what tasks are currently being executed.

Altogether, the CeBit scenario features a very constrained interaction, relying on system initiative only and providing only few capabilities. However, this enabled it to cope well with the high background noise level at the trade fair. Despite its limited capabilities, the system has been well received by the visitors. One reason for this may be the interactivity of the system. In particular, induced by the system listing all visible objects, users start testing the system by moving objects or putting them aside, or by presenting new everyday objects (which luckily had been considered when designing the speech recognition grammar).

6.3.2 Evaluation: Analyzing Speech Understanding Performance

As a part of a bachelor's thesis, an explorative user study has been conducted with the CeBit setup whose aim was to evaluate the system feedback provided by the movements of the pan-tilt camera [Poh09]. Additionally, the data was re-analyzed by the author of this thesis with regard to speech recognition performance, being known as a factor that greatly affects the overall performance of the system. For instance, it has been reported that the mean recognition score is the largest contributor to user satisfaction [WLKA98] [WKL00]. It is thus worth having a closer look at the speech recognition performance in the scenario at hand. Doing so will enable us to identify the different sources of speech recognition errors. Moreover, as the effects of speech recognition errors can be alleviated through appropriate recovery and repair policies within the dialog, we aim to derive strategies how to deal with them, as a basis for future iterations of the system.

In the study, 10 participants (5 female, 5 male) interacted with the system without prior instruction. The study setup did not define a specific goal that the participants had to achieve. Instead they were informally told to interact with the system as long as they wished to, and they were encouraged to play around with the system. For the re-analysis of the study, the video recordings of the interactions were combined with the system logs of the speech recognition and the speech understanding component. Each user utterance was labeled manually according to the categories below.

The interactions included 39.2 user utterances in average, i.e. the corpus collected contains 392 user utterances, 58 of which were false positive results from the speech recognizer. The remaining 334 utterances were labeled according to whether they could be processed (understood) correctly: A user utterance was labeled as *understood correctly*, if the utterance could be processed correctly by the dialog system. Otherwise, it was labeled as *understanding error*.
A common measure to evaluate understanding capabilities of a speech system is *concept accuracy* [BEG+96], which will be further discussed, and applied, in section 7.1.3. However, concept accuracy differs from the measure used here: While concept accuracy provides the proportion of concepts of an utterance transmitted correctly, correctness of understanding tells the proportion of utterances for which enough concepts were transmitted correctly for the dialog system to act correctly. Correctness of understanding was selected as measure here because it says something about how interaction is affected by understanding problems, considering the whole processing chain of an utterance, up to the dialog system.

Understanding errors may have various sources. In general, they typically occur due to mismatches between the expressed form of the user's intent and the system's modeling abilities [BR08]. For example, novice users might not be fully aware of the system's functionalities and its limitations and might try to perform an operation that the system cannot handle. Also, even if the system can handle the goal formulated by the user, it might be the case that the system's language model does not cover the specific formulation. We refer to this sort of user utterances as *invalid* (although this term reflects a somewhat system-centered view). Invalid utterances will result in understanding errors (unless handled otherwise, e.g. by out-of-vocabulary models [BG00]). On the other hand, even *valid* utterances may result in recognition errors if the user's pronunciation does not match the system's acoustic model, or if they are made at unexpected times.

Based on these considerations, the understanding errors were further broken down into *misunderstandings* and *non-understandings*. In a *misunderstanding*, the system operates on an interpretable (but incorrect) representation of a valid user utterance. In contrast, in a *non-understanding*, the system fails to construct an interpretable representation out of a (valid or invalid) utterance. Detection of non-understandings is, by definition, trivial, while reliable detection of misunderstandings has been identified as a key problem [LHS00]. The dialog system therefore needs to provide strategies to recover from misunderstandings, such as the interruption of actions and correction of given information.

As shown in table 6.4, 27% of the user utterances were not understood correctly. Considering that the scenario at hand is rather restricted, with many utterances just consisting of the object label, this number seems relatively high. A closer look at the non-understandings revealed that many of them were in fact unparseable as a whole, but would still have contained enough information for the dialog system to act upon. Hence, it was decided to replace the speech understanding component by more robust key-word matching techniques

Speech understanding	***Subcategory***	***Proportion***
Understood correctly		73%
Understanding error		27%
	Misunderstanding	27%
	Non-understanding	73%

Table 6.4: Proportion of correctly and incorrectly understood utterances. The latter are further broken down into misunderstandings and non-understandings.

in future iterations (as further detailed in section 7.1.1).

As we can see from table 6.4, the non-understandings represent a larger proportion of the understanding errors than the misunderstandings (73% and 27% respectively). Thus, we have taken a closer look at them and have broken down the invalid utterances among these into the following error causes:

- *Out-of-capability*: The utterance is beyond the system capabilities, or even beyond the system domain.
- *Out-of-vocabulary*: The utterance is within the system capabilities, but the specific wording is not part of the speech recognizer vocabulary.
- *Out-of-context*: The utterance is within the system vocabulary, but cannot be interpreted given the current dialog context.
- *Meta-comments*: The utterance is not directed to the system, but e.g. a question to the experimenter.

As table 6.5 shows, 25% of the non-understandings result from valid utterances. These represent the actual understanding errors, i.e. the user utterance is within the system's modeling capabilities and was made at the right time, but the system still fails to classify it. However, the larger proportion (75%) of the non-understandings is caused by invalid utterances.
Among these, the out-of-capability utterances (which are typically out-of-vocabulary as well, but have not been counted as such) represent by far the largest portion (72%). They are due to the fact that many participants attempted to teach more than one object ("here are two lemons") or to ask about a specific object ("show me the apple"). Also, half of the users attempted to reverse the roles and to demonstrate novel objects to the system. Other out-of-capability utterances were due to actual deficiencies in the dialog strategy design. For example, whenever all objects had been learned, the system enumerated the present objects. This encouraged the participants to correct the enumeration if it was incorrect which unfortunately had not been forseen.

Out-of-vocabulary utterances constitute another significant portion of the invalid utterances (16%). Almost all of them occur when participants use a label that is not within the speech recognition vocabulary. Most participants did not confine themselves to teaching the available fruits, but started to teach everyday objects such as wallets, mobile phone or keyrings. We had anticipated this and considered many object labels in the vocabulary, but could not foresee all labels used. For example, we failed to include answers such as “that’s me” or “that’s my sleeve”.
In general, both out-of-capability and out-of-vocabulary utterances occurred mainly in the second half of the interaction. This suggests not so much the failure of the structuring strategy, but rather that the participants had intentionally provoked theses errors by experimenting with the system in order to explore its limits. This interpretation is supported by the observations from the CeBit trade fair. Again, such experimentation behavior often started in the second half of the interaction, when users had already taught a few objects successfully.

Utterance	***Cause of Error***	***Proportion***
Valid utterance		25%
Invalid utterance		75%
	Out-of-capability	72% (i.e. 15% of all utterances)
	Out-of-vocabulary	16% (i.e. 3% of all utterances)
	Out-of-context	8% (i.e. 2% of all utterances)
	Meta commentary	4% (i.e. 1% of all utterances)

Table 6.5: Breakdown of non-understandings into valid and invalid utterances. The latter are further broken down into the different error causes.

To sum up, the results of this analysis show that non-understandings constitute the major part of understanding errors in the CeBit setup. Most of them were caused by so-called invalid utterances. Among these, out-of-capability and out-of-vocabulary utterances play a major role. When preparing the next iteration of the system – the Curious Flobi setup – we therefore have to pay special attention to these error sources, without neglecting the others. Possible strategies to mitigate the effects of the different understanding errors, or even to reduce their number, will be discussed in section 7.1.1.

7 Curious Flobi: Admitting More User Initiative

This chapter describes the Curious Flobi scenario – the main test bed for the proposed approach. Its design process is detailed in section 7.1. The resulting scenario and its technical realization is described in sections 7.1 and 7.2. Section 7.4 presents a PARADISE-style user study that evaluates the system.
The work described in this chapter has not been accomplished by the author alone: The system was implemented in collaboration with Ingo Lütkebohle, and the evaluation was conducted in collaboration with Ingo Lütkebohle and Nina Riether[1].

7.1 Preparatory Activities: The Design Process

This section presents the design process of the Curious Flobi scenario and the preparatory activities undertaken. Section 7.1.1 points out the consequences of the analysis of the CeBit set up described in the previous chapter. Sections 7.1.2 and 7.1.3 describe how the dialog strategy and the speech recognition grammar have been design based on a WOz study on object teaching.

7.1.1 Lessons Learned from the CeBit Setup

Section 6.3.2 has given a detailed analysis of the different error causes for speech recognition errors in the CeBit setup, especially for non-understandings. In this section, the consequences for the interaction strategy of the current iteration are discussed. Strategies are derived how to reduce their number, and how to cope with the inevitable ones.

As detailed in section 6.3.2, in contrast to non-understandings, **misunderstandings** are by definition non-detectable. However, uncorrected misunderstandings may have serious consequences, as they may result in the system performing erroneous operations, or using faulty parameters. In the CeBit setup for instance, the system might learn wrong labels. Representing more than a quarter of all recognition errors, the system must provide interaction strategies to recover, and to repair them. The Curious Flobi scenario was already implemented based on the PaMini framework, whose interaction patterns incorporate a range of, and varieties of, recovery strategies, such as confirming an action

1 In detail: I have developed the study setup and proposed the subjective measures. The objective measures were defined in collaboration with Ingo Lütkebohle, who also realized an automated calculation of the objective measures. Nina Riether calculated the regression analysis of the data.

before beginning it, canceling it, and explicit and implicit confirmation of information, giving the user the possibility to correct it. Besides applying appropriate strategies within interaction, recovery of information needs to be considered at the system level as well, i.e. if a wrong label has been learnt, the responsible component (e.g. the object recognition) must allow to overwrite it with the correct one later, or at least to merge the representations.

In order to reduce the number of non-understandings that are caused by **valid user utterances**, the speech processing was revised. Both in the Curious Robot system and in the CeBit setup, a speech understanding component was used that represented utterances as hierarchical, linked frame-slot structures, each rated based on completeness of the structures [HWS06]. Utterances for which the linking algorithm fails to construct a reliable representation are rejected. However, many of the rejected utterances, though unparseable, still provide the information required (e.g. an object label) and thus would be rejected unnecessarily. In short, the fundamental deficiency of the approach is that it does not take into account the dialog context when parsing the utterances.
As a consequence, it was decided not to employ the above speech understanding algorithm, but to directly use the speech recognition result as input for the dialog system. This becomes possible as the HMM-based speech recognizer used in all of our scenarios integrates a language model in form of an LR(1)-grammar [Fin99], providing not only the recognized chain of words, but also the corresponding grammar tree. By matching conditions over the nonterminal symbols of the grammar tree within the dialog system, robust key-word matching can be achieved, driven by the dialog expectation. As it was to turn out during the user study described in section 7.4, this approach worked surprisingly well and enabled correct dialog decisions even if an utterance contains many incorrectly classified words.

Out-of-capability utterances have been identified as the largest error source. Considering the somewhat restricted capabilities of the CeBit setup, this seems not surprising. A large proportion of these errors were due to users' attempts to demonstrate objects themselves, which strongly suggests that they would prefer to take a more active role in the interaction. As a consequence, the current iteration was extended as to allow for more user initiative. In particular, users can demonstrate objects on their own initiative now, and they can ask test questions about a specific object.
Moreover, a WOz study on object teaching (cf. section 7.1.2) served as foundation to study typical demonstration behavior. The analysis revealed that not only the task-related but also social elements are crucial in such interactions. Thus, the results of the WOz study have fundamentally influenced the design of the interaction strategy.

Another significant cause of error were **out-of-vocabulary** utterances. Out-of-vocabulary utterances decrease with grammar size, but so does in-grammar accuracy. Thus, a balance between wide grammar coverage and good in-grammar accuracy had to be found. Again, the analysis of the aforementioned WOz study served as a guide for the design of the speech recognition grammar, giving insights into the verbal strategies users apply in demonstrating objects, as well as into their social interaction behavior. Additionally, the resulting speech recognition grammar was evaluated and fine-tuned in a pre-test, enabling iterative grammar improvements (cf. section 7.1.3).

Out-of-context utterances occurred rarely in the CeBit setup, which is probably due to the rather restricted interaction capabilities. With increasing system capabilities, in particular regarding user initiative, out-of-context utterances are expected to occur more frequently if left unchecked. Fortunately, PaMini's interleaving interaction patterns facilitate implementation of a non-restrictive interaction strategy that allows for interjections and social feedback. However, unrestricted interleaving may not always be appropriate, e.g. interleaving of two object demonstration patterns may confuse the user a little.

Finally, a minor problem in the evaluation of the CeBit study was **meta commentary**, i.e. utterances that addressed the experimenter. They can easily be reduced in future studies by making sure that the experimenter is not present during the interaction. Table 7.1 summarizes the strategies proposed in this section.

Cause of Error	***Breakdown into subcategories***	***Strategy***
Misunderstanding		Dialog provides recovery strategies
Non-understanding		
	Valid utterances	Expectation-based key-word matching
	Out-of-capability	Extension of system capabilities
	Out-of-vocabulary	Pre-evaluation of speech recognition grammar
	Out-of-context	Flexible interleaving of interaction patterns
	Meta commentary	Experimenter not present

Table 7.1: Strategies to deal with the different speech recognition error sources.

7.1.2 Analysis of a WOz Study on Object Teaching

In order to study typical demonstration behavior, the author re-analysed an earlier study on object teaching conducted by her colleagues [LHL$^+$09], which originally focused on facial expressions in HRI. The 11 participants received the explicit instruction to teach the names of several objects to the mobile robot BIRON. They were advised to check that the robot had actually learned the objects. However, it was not specified how they should present and check the objects. The study was completely based on the WOz paradigm, with a wizard controlling all robot actions (i.e. view direction and utterances). To simulate typical recognition performance, misunderstandings and misclassifications were randomly induced by the operator.

Examining the interactional aspects of the study, we found that all interactions had a very similar structure, consisting of an opening part, a task-related part and a closing part. 36% of the interactions additionally feature transitional phrases that introduce the task-related part. In the opening phase, introducing each other (82%) and exchanging pleasantries (18%) are frequent. Aside from object demonstrations, the task-related phase consists of checking learned objects (45%) and transitional phrases between the objects (36%). Praising the robot for correctly learned objects turned out to be universal (100%). Sometimes the task-related part includes closing remarks (36%). Table 7.2 lists a sample of utterances that participants used during the study.

Dialog segment	***Example utterances***
Greeting	Hello. Hello, BIRON. Can you hear me? Hi, I'm new here.
Introducing	My name is X. This is X Y from Bielefeld. My name is X, as I said before.
Exchanging pleasantries	How are you? What's up?
Task description	We are going to work together today. I am going to teach you a couple of things, one after another. I would like to show you some household items. There are many things. I would ask you to recognize them.
Attracting attention	BIRON, look here. Do you recognize this?
Object demonstration	This is a bottle. This is a ball, for playing. This is a pen. You use it for writing. Pen.
Checking	What is this? What could this be? Do you known what this is called? Do you remember this one?
Praising	That's correct. Fine, thank you. Well done.
Transitional phrases	Would you like to see more items? A new word. Let's go on with the next one.
Closing task	Okay, we have practiced enough for now. I have showed you all objects. That's it.
Parting	Good bye! I have to move on now. Bye! Take care, BIRON.

Table 7.2: Example utterances from the WOz object teaching study. Blocks are: i) Opening interaction, ii) transitional phase, iii) task-related phase and iv) closing interaction.

Based on the insights gained from this study, the interaction strategy of the present scenario was designed. Not all of the observed strategies could be transferred directly. For example, the object descriptions sometimes include rich functional or anecdotal descriptions that are too complex for an automatic system to cope with, so they were simplified. In general, however, most of the observed strategies were realized. Table 7.4 in section 7.2 lists the resulting interaction capabilities of the system.

7.1.3 Design of the Speech Recognition Grammar

When developing spoken language systems, system designers find themselves confronted with the *vocabulary problem*: On the one hand, designers need to predict and consider the words people use when operating the system while on the other hand larger vocabularies lead to substantial decrease in recognition performance [BRRL98]. Hence, preparing the Curious Flobi scenario, we need to ask the question whether recognition performance will still be acceptable when adopting all of, or most of, the user utterances observed in the WOz study.

In order to obtain a worst-case estimation, in a first step all user utterances from the WOz study have directly been included in the ESMERALDA speech recognition grammar [Fin99], with a resulting grammar size of 423 words[1], and *concept accuracy* has been evaluated for a random selection of sentences from the grammar. In contrast to *word accuracy*, which considers the surface form of each single word of an utterance, concept accuracy indicates how many of the semantic concepts of the utterance have been transmitted correctly. In a whole-system setup, concept accuracy is a more expressive measure, as it allows variations in the wording of a concept as long as it is does not affect selection of a system action (e.g. “an apple” vs. “a apple” or “yes” vs. “yeah”). Concept accuracy (CA) can be calculated using the formula

$$CA = 100(1 - \frac{C_S + C_I + C_D}{C})$$

where C, C_S, C_I and C_D indicate the number of overall concepts, the number of substituted concepts, the number of inserted concepts and the number of deleted concepts. [CR01] further distinguish three categories of concepts:

- A *non-concept* slot “is a slot that contains information that, while captured in the grammar is not considered relevant for selecting a system action. Politeness expressions, such as ’please’, are an example.”’ [CR01].
- A *value-insensitive* slot “is a slot whose identity, rather than specific value, is sufficient to drive system action” [CR01]. An example would be the slot “confirmation”.
- A *value-sensitive* slot “is a slot for which both the occurrence and the value of the slot are important” [CR01], for example the slot “city_name”.

1 Note that the grammar provides additional constraints for the statistical modeling, and that the speech recognizer is still capable to classify utterances that are not generated by the grammar (provided that the words are included in the statistical model)

While incorrectly classified non-concepts do not have an effect on system performance, they were explicitly included in our evaluation in order to keep the results comparable with results reported in literature. Word accuracy and concept accuracy typically do not differ widely [CR01], [SW02], with a linear relation between them [BEG+96].

For the pre-evaluation of the speech recognition grammar, 75 sentences generated by the above grammar have been selected randomly and were spoken by 6 volunteers (3 male, 3 female). Sentences contained 160in total concepts, 2.13 on average. A comprehensive list of the selected sentences is given in appendix D. Concept accuracy has been determined for each sentence according to the above formula.
Note that often more than one word is required to form a concept, e.g. the concept “task” is specified through the words “showing objects”, or a distinction has to be made between “a ball” and “not a ball” in order to capture corrections. Also, concept accuracy could in theory take a negative value (e.g. if many additional concepts are inserted). In these cases, which occurred only a few times, concept accuracy for the respective sentence has been set to zero. As shown in table 7.3, a mean concept accuracy of 78.48% was achieved (compared with 79.9% word accuracy measured for the same recognizer [Fin99]). Values reported in literature vary considerably: While [BEG+96] and [SW02] achieve as much as 92% and 86% concept accuracy respectively, [BR08] and [Lem04] report 74% and 68.9% concept accuracy respectively. However, the test sentences were not uttered spontaneously but were read off from a prepared list. Thus, this value is only an upper bound for concept accuracy in a real setup.

Mean	78.48%
Standard deviation	6.14%
Min	70.89%
Max	87.97%

Table 7.3: Concept accuracy achieved for the test grammar obtained from the WOz study.

Nevertheless, the achieved concept accuracy seems encouraging enough to suggest that using the WOz study as a guideline for grammar design is a viable strategy. As a consequence, the grammar obtained from the WOz study was transferred to the real system, with minor adjustments. All utterances from the user study that occurred more than once were adopted directly. The grammar was then adapted from the domain of household items to our fruit-learning domain. By sticking with the fruit-learning domain (like in the Curious Robot scenario) we hoped to prevent the rich functional descriptions that we have observed in descriptions of the everyday objects (e.g. “Look here, and if you need straight lines for handcrafting or want to cut straight, then you need a ruler”). Thus, the resulting grammar could be reduced to 230 words (from 423 word for the original WOz grammar), which leads to expect an increase of concept accuracy. However, compared with the 119 words for the Home-Tour grammar and the only 37 words for the Curious

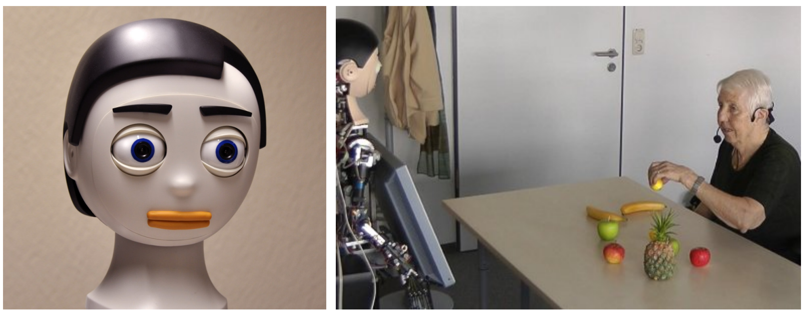
Robot grammar, it still covers a relatively large vocabulary. While no formal evaluation of word or concept accuracy has been conducted yet, accuracy appeared to be completely satisfactory during testing the system, and also during its real-world application within the user study described in section 7.4.

7.2 Scenario Description

A lot of preparatory work was done during the design process of the Curious Flobi scenario. The most important decisions were (i) to admit more user initiative than in the Curious Robot scenario, and (ii) to use robust keyword matching instead of the previously used speech understanding component. Also, the both the interaction strategy and the speech recognition grammar were optimized based on a WOz study. This section gives an overview on the resulting scenario.

Figure 7.1 shows the hardware platform used and the scenario setup. Apart from basic social talking skills (greeting, complimenting, etc.), the system has three main capabilities: i) Learning objects the user demonstrates, ii) asking for an object on its own initiative, and iii) answering requests regarding object labels and known objects. Table 7.4 lists the interaction capabilities of the system. Most of them that have been adopted from the utterances observed in the WOz study (see table 7.2).

Opening, closing and restarting the interaction was modeled using the Human Interaction Opening, Closing and Restart patterns. For modeling the different (both task-related and social) user remarks regarding feedback on system actions, complimenting and transitional phrases that precede an object teaching episode, the Human Simple Statement pattern was used. While a simple statement alone models only a small part of the interaction, they may be chained together to form more coherent interaction segments. Table 7.2 shows an example from the user study in which two simple statements are triggered in sequence,



(a) The Flobi head (b) Demonstrating a fruit

Figure 7.1: Scenario overview.

with the first one modeling a user introduction, and the second one praising the robot. The user's reply is classified as a praise, due to the word 'nice', which the robot again replies to very generically by thanking the human. Thus, thanks to the keyword spotting approach, the robot is able to react to a variety of utterances in a meaningful way, and even to utterances that had not really been foreseen.

U	My name is Anneliese.
F	Hello Anneliese. Nice to meet you.
U	Oh, nice to meet you too!
F	Thank you.

Figure 7.2: Chaining together simple statements.

Also the user's attempt to attract the robot's attention ("Flobi, look here") was modeled as a simple statement which is always being replied to with "Yes, I'm looking", without including any task communication. This pattern still does not have a pure dummy function, as the robot orients its gaze at the currently most visually salient point (or at the user, as appropriate for the task state). Thus, if the point the user tries to attract the robot's attention to already lies in the robot's field of view, it has a good chance to actually become the most salient point (e.g. as a result of the user's salient nonverbal gestures or hand movements).

If the user demonstrates an object, the referred object, or rather the referred region, has to be determined. Once the region has been identified, it can be associated with the given label. Thus, an object demonstrating episode involves two consecutive tasks: the *resolve reference* task and the *learn object* task, which can only be executed if the *resolve reference* task succeeds. To model this process, the Human Object Demonstration pattern was introduced. Since each of the two tasks may fail, the pattern has to consider the different error conditions. Another pattern that includes reference resolution has been introduced to model the user's test questions to check if the robot is able to recognize a specific object: the Human Object Test pattern. It is conceivable that, in the future, both patterns will be applied in other learning tasks, such as learning and recognizing people's faces or rooms.

Failure of learning has to be considered also in the reverse case, when it is the robot who asks for an object on its own initiative. The existing Robot Information Request with Explicit Information pattern was not sufficient for this purpose, as it transfers the information to the responsible system component, but does not acknowledge its processing. For this reason the Robot Information Request with Explicit Information and Task Acknowledgment was introduced. In contrast, in the original Curious Robot scenario, learning was not acknowledged, but assumed as a process that follows always. This was changed for two reasons. First, breaking down learning to a task with clearly defined states allows for more detailed feedback on the learning progress. Second, the restricted but robust object learning setup from the Curious Robot scenario was exchanged for one that offers more possibilities but fails occasionally, which had to be accounted for by the interaction strategy (cf. the description of the system components below).

Dialog segment	***Example interaction***	***Interaction pattern***
Greeting	User: Hello, Flobi Flobi: Hi, hello.	Human Interaction Opening
Introducing	User: My name is Anneliese. Flobi: Hello, Anneliese. Nice to meet you.	Human Simple Statement
	User: What is your name? Flobi: My name is Flobi?	Human Simple Statement
	User: Nice to meet you, Flobi! Flobi: Ditto.	Human Simple Statement
Exchanging pleasantries	User: How are you? Flobi: Fine, thank you!	Human Simple Statement
Task description	User: I am here to show you some objects. Flobi: Alright, let's start!	Human Simple Statement
	Flobi: There are many objects on the table I don't know yet.	Robot Notification (only C2 & C3)
Object query	Flobi: What is that? User: This is an apple. Flobi: Alright, an apple, is that correct? User: Yes. Flobi: Okay, I am going to learn the apple. Flobi: I have learned the apple.	Robot Rejectable Information Request with Explicit Confirmation and Task Acknowledgement (only C2 & C3)
Yielding Initiative	Flobi: You can show me something, too.	Robot Notification (only C2 & C3)
Attracting attention	User: Flobi, look here! Flobi: Yes, I'm looking.	Human Simple Statement
Object demonstration	User: This is a yellow lemon. Flobi: Alright, a yellow melon. Is that correct? User: No, it's a lemon. Flobi: Alright, a lemon. Is that correct? User: Yes, that's correct. Flobi: Okay, I am going to learn the lemon. Flobi: I have learned the lemon.	Human Object Demonstration with Explicit Confirmation
Listing learned objects	User: What have you learned so far? Flobi: I have learned a lemon and an apple.	Human Information Request
	User: Okay, next one. Flobi: Okay.	Human Simple Statement
Checking	User: Flobi, do you know this one? Flobi: I will have a look at it. Flobi: This is an apple.	Human Object Test
Praising	User: Well done, Flobi. Flobi: Thank you.	Human Simple Statement
Transitional phrases	User: Would you like to learn another object? Flobi: Yes, I want to learn them all.	Human Simple Statement
Closing task	User: Okay, that's it. Flobi: Are you leaving already? User: Yes. Flobi: What a pity! Good bye.	Human Simple Statement
Parting	User: Good bye, Flobi. Take care! Flobi: Bye, see you later.	Human Interaction Closing
Clarification	User: Restart. Flobi: Okay, let's start over again.	Human Interaction Reset
	Flobi: Pardon me?	Robot Repeat Request
	Flobi: Should we better start over again? User: Yes, let's start over again. Flobi: Okay.	Robot Suggest Restart

Table 7.4: Interaction capabilities of the system and the employed interaction patterns. Blocks are: i) Opening interaction, ii) transitional phase, iii) task-related phase and iv) closing interaction. Note that in addition to the listed utterances, many variations in the wording are possible.

7.3 System Overview

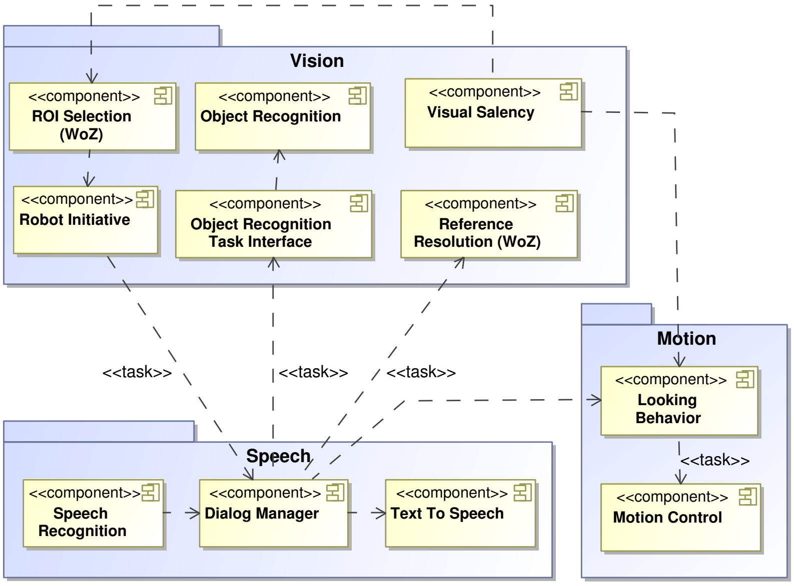


Figure 7.3: Components of the Curious Flobi system. Dashed lines indicate general event notifications, while lines labeled with *task* indicate task communication.

The scenario is implemented on the Flobi anthropomorphic robot head (shown in figure 7.1) [LHS+10]. Besides the wide-angle, high resolution stereo cameras that are built in the head, the hardware includes a flat-screen display on which the referred object is shown when the robot asks for a label. From a software perspective, the system consists of three major parts for vision, speech and motion, as depicted in figure 7.3. As with the previous scenarios, components communicate through event notifications via XCF [WHBS04], most of them relying on task-based communication.

To realize object learning and testing, vision and speech part interact in a tightly integrated manner through the above mentioned interaction patterns (Human Object Demonstration with Explicit Confirmation, Human Object Test and Robot Rejectable Information Request with Explicit Confirmation and Task Acknowledgement). Components involved are, besides the dialog manager, the reference resolution component, the object recognition task interface (a proxy of the object recognition component [HLI+10]), and the robot initiative component. The dialog manager initiates the *resolve reference* and *learn object* tasks, handled by the reference resolution and the object recognition task interface, respectively. Conversely, the *ask for object* tasks that makes the robot ask for an object label, is initiated by the robot initiative component and handled by the dialog manager.

The system is not fully autonomous. As has been observed in the WOz study described above, and has been confirmed by the user study with the present system (cf. section 7.4, in particular table 7.7), there are many different ways of referring an object, and we did not want to restrict the user's natural behavior. We rather intended to explore the different strategies through an unrestricted study, and to provide thus some data regarding the kind of tutoring behavior typically occurring in such situations. Hence, replacing the reference resolution by a WOz component appeared necessary as a first step to approach the problem of reference resolution. In addition, the region of interest (ROI) selection also uses a WOz component because we felt that the autonomous component used in the original Curious Robot scenario was not predictable enough for the purposes of a user study. However, both WOz components were fully integrated into the overall system, interacting with other components via the Task State Protocol.

Using WOz components for certain tasks does of course not mean that the system runs completely error-free. Reference resolution may still fail if the wizard is not able to recognize what object the user's description refers to, or if it is not within the robot's field of vision. In fact, this occurs not all too rarely, (see section 7.4.2). For the ROI selection, the wizard occasionally made the robot ask for an unexpected object (e.g. the edge of the table or a part of the user's body) in the user study, not following a fixed strategy, but at least once per interaction. In addition, there are errors which the autonomous system components produce, such as errors in speech recognition and object recognition. Even learning an object may fail, for instance if the user puts the object aside before learning has completed, or if the user occludes the objects during learning (cf. section 7.4.2).

7.4 Evaluation: A PARADISE-style User Study

This section presents the evaluation of the Curious Flobi system. The evaluation follows the PARADISE approach (cf. section 2.3.1). By relating subjective and objective measures through a regression analysis, the (objective) factors that contribute to different aspects of (subjective) user satisfaction can be identified. Besides the PARADISE analysis, the influence of the robot's task initiative was investigated. A qualitative analysis of the interactions completes the evaluation. Section 2.3.1 describes the setup of the study. The results of the study are discussed in section 7.4.2.

7.4.1 User Study Setup

Participants

Out of 32 participants who took part, we used 28 recordings (14 male, 14 female), with the remaining four excluded due to technical problems during their trial. Most of the participants had been recruited at a university event for the general public and thus represented a wide age range, with mean age at 33.5 years, minimum 21 and maximum 79.

On a scale from 1 (none) to 6 (lots of), the average rating for knowledge of computers was at 5.07, of speech systems at 2.52, of robot systems at 1.96 and of programming experience at 2.26. They were compensated for their participation in the experiment.

Instructions

In order to study natural demonstration behavior, participants received as little instruction as possible. They received written instructions, specifying that they were to engage in interaction with the robot Flobi, and that Flobi was supposed to learn object labels during interaction. They were also advised to check that the robot had actually learned the labels. It was, however, not specified how they were to present and check objects. They were told that they should interact with the robot as long as they wished, with 5-10 minutes recommended as a guideline. Also, they were informed that they could begin the interaction by greeting the robot, and end the interaction by saying goodbye. In addition, participants were advised not to be discouraged by speech recognition problems, and that they could repeat or rephrase their utterance in such cases. Last, an emergency phrase ("Restart") was provided. The interactions were in German. No other person was present in the room during the interaction. A translated instruction hand-out can be found in Appendix E.

Wizard control

As described in section 7.3, the system is not fully autonomous, but contains two WOz components: reference resolution and ROI selection. In the study, the experimenter first instructed the participants, then left the room and took the role of the wizard. The wizard control station was located in an adjacent room, where the robot's field of view was displayed on a computer screen. The wizard's tasks were to identify the objects that were referred to and to mark them in a graphical user interface. Moreover, in condition C2 and C2 (which will be described below) the wizard had to trigger robot initiative by marking the objects the robot should ask for in the graphical user interface. The participants were not told that the system was partially controlled by the experimenter.

Objective measures

A wide range of objective measures has been collected, most of which were derived from system logs. For each component, the relevant event notifications were logged, such as speech recognition results, text-to-speech output, dialog pattern state changes as well as object recognition and reference resolution tasks. With these log data, a detailed reconstruction of the interaction can be achieved. The data was also annotated manually based on the video material to capture inappropriate robot utterances, or the correctness of the robot's answer on a test question. In total, we used 28 of these for evaluation (see table 7.5). As proposed in the PARADISE framework [WLKA97], we divided them

into the categories dialog efficiency, dialog quality and task success. Technically, the interactional aspects of system performance (i.e. the dialog quality and dialog efficiency measures) are calculated mainly based on information related to the Interaction Patterns, whereas the Task State Protocol provides information at task level (i.e. the task success measures).
The **dialog efficiency** measures capture the rapidity of the interaction and include for example the duration of interaction, the number of user and robot utterances within a certain time unit, the mean length of user utterances, or the number of objects learning episodes within a certain time unit. The **dialog quality** measures address the smoothness of the interaction. We considered for example gaps, overlaps, repairs and label corrections. The **task success** measures concentrate on the outcome of the interaction with respect to object learning. Among others, we measured the proportion of successful reference resolution and object learning tasks, the proportion of correct robot answers to test questions, and the user's out-of-capability utterances.

Subjective measures

In addition to the objective measures described above, we collected subjective measures based on a questionnaire the participants were asked to complete after the interaction with Flobi had finished. We attempted to rely on standardized questionnaires as far as possible. In this regard, a trade-off had to be found between validated but much generic, and more informative but non-validated questions.
The questionnaire consisted of 50 items, that we aggregated into seven category measures. The first four categories, **dialog efficiency**, **task success**, **cooperativeness** and **usability**, refer to the interaction itself. They contain questions to assess the participants' impression of dialog efficiency and task success, on how cooperative they felt the robot behaved during the interaction, and on how they rated the usability of the system. The interaction-oriented items are roughly based on the evaluation of the COMIC dialog system [WFOB05], which we adapted for our specific scenario. The remaining three categories, **likeability**, **perceived intelligence** and **animacy** address the participants' impression of the robot. They were adopted from the standardized Godspeed questionnaire [1] [BCK08]. In addition, the questionnaire included five single (summarizing) questions, targeting the overall impression of **ease**, **efficiency**, **clarity**, **pleasantness** and **understandability** of the interaction. All replies to the questions had to be given using a six-point Likert-scale. The complete questionnaire can be found in Appendix F.

1 However, we skipped the categories **anthropomorphism** and **perceived safety**, as we considered them irrelevant for the scenario at hand.

Performance functions

The objective and subjective measures have been related to each other by means of a PARADISE-style evaluation. This evaluation method uses stepwise multiple linear regression to make predictions about subjective measures, like user satisfaction, based on several objective performance dimensions, like task success, dialogue quality, or dialogue efficiency (cf. chapter 2.3.1). The performance functions that result from this analysis supply answers to questions like: Which are the relevant factors that contribute to user satisfaction? Which components need to be optimized in future iterations of the system? The results are, to a certain extent, generalizable to similar systems.
The PARADISE approach originally suggests the Kappa coefficient as a measure for task success. The Kappa coefficient can be used to measure how many of the concepts were transmitted correctly during an interaction (cf. chapter 2.3.1). It is suitable for classical information seeking domains, but does not appropriately cover the complex task structure of an action-oriented robotic scenario. Hence, the Kappa coefficient was replaced by the above described objective measures for task success.
Moreover, in contrast to the original PARADISE method, user satisfaction was not assessed by a single target variable, but broken down into the different subjective measures described above, like ease or efficiency of the interaction. These rather abstract concepts were further broken down into several items that are easier assessable by the users. For example, the intuitiveness of interaction was assessed by questions like "I found the last object easier to teach than the last one", and the dialog quality was assessed by asking for the appropriateness of the robot's utterances regarding content and timings. Additionally, the above summarizing items that ask for the users' overall impression were directly used as target variables.

Between-subjects factor

Moreover, we were interested in the influence of the robot taking initative. As a three-level between-subjects factor, the degree of task initiative of the robot was varied:

- Condition C1 (User Initiative) allows for user initiative only.
- Condition C2 (Mixed Initiative) allows for both user and robot initiative.
- Condition C3 (Structured Initiative) is identical with C2, except that the robot additionally yields initiative explicitly.

More specifically, in condition C1 the only way to teach the robot objects was to demonstrate the objects one after another. In condition C2, learning is performed in mixed initiative. The robot asks for an object label on its own initiative at interaction begin. Also in later stages of the interaction the robot would ask for object labels, provided that no other interaction episode is ongoing. In condition C3, the robot yields initiative explicitly after having asked for two object labels at the start of the interaction ("You can show me something, too"). By means of an analysis of variance (ANOVA) the differences between the groups regarding subjective and objective measures were evaluated.

7.4.2 Results

Objective measures

A sample of the objective measures is shown in table 7.5. Users interacted in average 11.07 minutes with the robot (Time), teaching 9.43 objects during interaction (Obj), 0.91 per minute (Obj/min). In 79% of all learning trials, the referred object could be resolved (Ref), and 78% of these could be learned by the system successfully (Learn). Thus, 62% of the demonstration episodes could be performed successfully (Learn). 53% of the objects (of which not all had actually been taught before, though) could be recognized on the user's request (Check), 55% of these correctly ($\text{Check}_{\text{correct}}$). Thus, overall 29% of the recognition requests could be performed successfully ($\text{Check}_{\text{correct}}$).
As described previously, an object demonstration consists of two tasks: reference resolution and learning. The reference resolution fails if the referred object can not be identified by the wizard, e.g. because it is out of the robot's visual field (and thus out of the wizard's visual field, too), or because it is occluded. Learning typically fails if the object is put away or occluded during the learning process. Similarly, an object recognition request consists of reference resolution and object recognition.

Participants employ a variety of different referencing strategies. As shown in table 7.7, 27.86% of the learning episodes were initiated by the robot, 72.14% by the users. Among these, the most frequent referencing strategy was lifting the object, followed by spatially isolating it, pointing at it or touching it. Combinations of these were also observed. In 4.57% of all cases, participants did not use a visual reference strategy at all, but referred to the object verbally. 10.81% of the object references were false positives due to speech recognition misunderstandings. Figure 7.4 depicts examples for the different referencing strategies. The variety of strategies would certainly present a tough challenge to an autonomous reference resolution component. As a consequence, it might become necessary to restrict interaction at this point, e.g. by opting for one common (yet automatically well recognizable) strategy.

The most noticeable result regarding the objective measures is that considerable individual differences exist between the single interactions (which coincides also with the observations from the qualitative analysis). The differences can be found alike in the measures for dialog efficiency, for dialog quality as well as for task success. Regarding dialog efficiency, for instance, the system learned in average 9.43 objects during interaction, with a minimum of 2 and a maximum of 19 objects (Obj). Between 0.2 and 1.6 objects were taught (Obj/min) per minute. Looking at the dialog quality measures, overlaps are a significant measure for turn-taking problems and speech recognition performance in general: recognizer performance is much lower for user utterances that overlap with robot utterances than for

Abbreviation	***Measure***	***Min***	***Max***	***Mean***	***Stdev***
Time	Duration of interaction (in minutes)	5.10	21.14	11.07	4.05
UU	# User utterances	38	219	95.68	39.65
MLU	Mean length of user utterances (in seconds)	0.70	1.39	1.01	0.02
UU/min	User utterances per minute	3.16	7.90	3.55	2.20
RU/min	Robot utterances per minute	5.44	15.19	10.91	2.50
Obj	# Objects learned	2	19	9.43	3.92
Obj/min	Objects learned per minute	0.2	1.6	0.91	0.39
ObjDemo	# Object demonstrations	2	25	10.07	5.97
ObjDemo/min	Object demonstrations per minute	0.22	2.54	0.93	0.58
Obj_{corr}	# Correctly learned objects	1	17	8.79	3.85
Obj_{corr}/min	Correctly learned objects per minute	0.14	1.56	0.79	0.31
ObjTest	# Object tests	0	32	10.64	7.76
ObjTest/min	Object tests per minute	0.00	2.07	0.92	0.59
$\text{Steps}_{\text{ObjDemo}}$	Average interaction steps for object demo	4.23	6.80	5.80	0.60
$\text{Steps}_{\text{ObjReq}}$	Average interaction steps for object request	0.00	4.00	3.41	1.21
Gaps	Length of global pauses (in seconds)	0.53	3.32	1.50	0.62
Overlaps	% of time UU and RU overlapping	0%	49%	20%	14%
UU_{delay}	Average delay before UU (in seconds)	0.38	5.12	2.04	1.15
Repair	% of RU dealing with repair	4%	38%	14%	8%
RU_{inapp}	% of inappropriate RU	1%	26%	6%	6%
NPattern	# Completed interaction patterns	18	69	43.50	13.65
Interleave-Ratio	Ratio of interleaving to NPattern	7%	79%	48%	15%
Ref	Success rate of reference resolution	36%	100%	79%	15%
Learn	Success rate of object learning	45|35.6%	100|79%	78|62%	20|15%
Check	Success rate of object check	0%	100%	53%	29%
$\text{Check}_{\text{correct}}$	Correctness rate of object check	0|0%	100|53%	55|29%	38|20%
UU_{ooc}	Out-of-capability UU	0	28	7.64	6.80
UU_{ooc} %	Percentage of out-of-capability UU	0%	21%	6%	5%

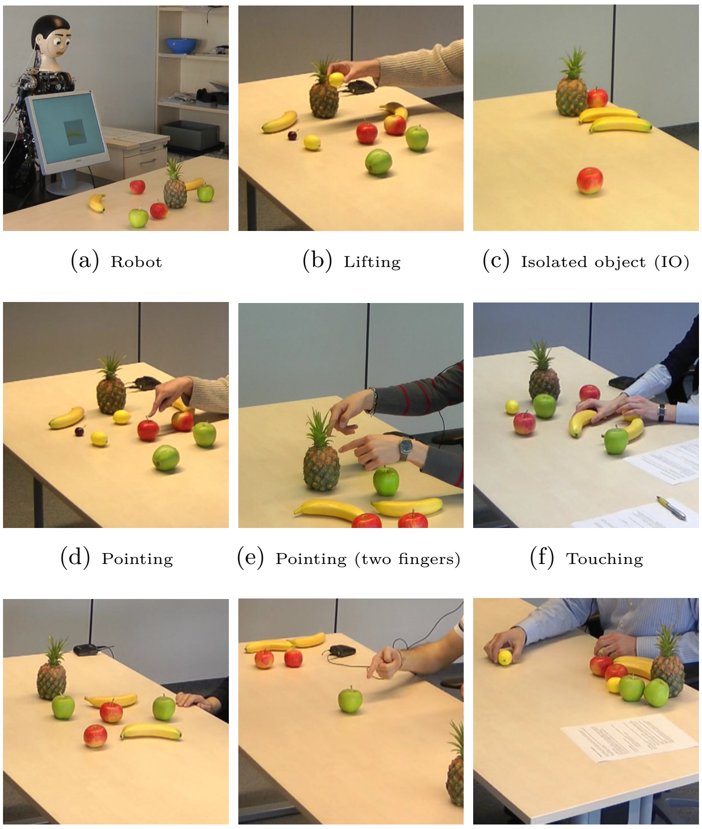
Table 7.5: Objective measures. Blocks are: i) dialog efficiency, ii) dialog quality, and iii) task success.

Category	***Percentage***	***Example***
Correct object check	17%	“no, it’s not an apple, its a lemon”
Meta object-organization	17%	“i will put this there”
More than one object	13%	“there are two lemons”
Gaze	12%	“what are you looking at?”
Cancel learning	5%	“no, stop”
Categories	5%	“all these objects are fruits”
Repeat	3%	“please repeat”
Next	3%	“continue”
Color	3%	“what color is the apple?”
Others (task-related)	20%	
Others (non task-related)	2%	

Table 7.6: Ratios for out-of-capability utterances.

Referencing strategy	***Percentage***
Robot	38%
Lifting	19%
Isolated object	14%
Pointing	13%
False positive	11%
Touching	10%
Non-visual	5%
Isolated object + pointing	1%
Isolated object + touching	1%

Table 7.7: Ratios for referencing strategies.



(a) Robot (b) Lifting (c) Isolated object (IO)

(d) Pointing (e) Pointing (two fingers) (f) Touching

(g) Non-visual (h) IO + pointing (i) IO + touching

Figure 7.4: The different object referencing strategies.

non-overlapping utterances. In some cases overlapping utterances (Overlaps) did not occur at all (0%), whereas in other cases almost half of the utterances overlapped (49%). As overlaps inevitably entail speech recognition problems, this is reflected also in the proportion of repair utterances (Repair) and inappropriate robot utterances (RU_{inapp}), ranging between 4% and 8%, and 1% and 26% respectively. The ultimate task success depends strongly on the demonstration strategy the user applies, even if the reference resolution is a WOz component. However, the strategies applied vary enormously, which is reflected in the large variance for success of reference resolution (Ref) and object learning (Learn), ranging between 36% and 100%, and 45% and 100%. Even more striking is the variance for object recognition (Check), which ranges between 0% and 100%.
A possible approach to address the individual differences between the users would be to calculate the objective measures on-line, during interaction. Doing so would enable the system to react to interaction problems right away, and to adapt its behavior accordingly, e.g. by giving more guidance on the demonstration strategy, explicitly communicating its capabilities, or increasing its own initiative.

Also the out-of-capability user utterances (UU_{ooc}) exhibit a large variance, but we were pleased to see that they were rather few in number (6% in average). In contrast, in the CeBit setup they accounted for 75% of all "invalid" user utterances, i.e. for 15% of all user utterances (cf. table 6.5). This means that the preparatory activities, consisting of the analysis of a WOz study and a pre-test of the speech recognition described in section 7.1.2 and 7.1.3, contributed to adapting the system capabilities to the users' expectations.
Nevertheless, it is interesting to have a closer look at the out-of-capability utterances. Table 7.6 lists the categories they belong to. Some reveal a deficit in the dialog strategy, as for example the attempt to correct the robot's reply to a recognition query (17%), which was one of the most frequent out-of-capability utterances. Also the attempt to cancel an ongoing learning task (5%), or to ask the robot to repeat its utterance (3%) were common error causes. In future iterations of the system, these issues will be fixed by adding the required functionality, or by adapting the interaction patterns used (e.g. making the Human Object Demonstration pattern cancellable).
Others demand abilities on the task level that the current system cannot account for, such as trying to demonstrate more than one object at a time (13%), teaching categories (5%) or querying colors (3%). In the condition C2 (where the robot asks for object labels on its own initiative, but does not yield initiative explicitly), some participants requested the robot to ask for the next object (3%).
Other common causes for out-of-capability utterances were remarks on object organization (17%) and on the robot's gaze direction (12%). The latter reveals that the robot's gaze is an important cue for the human interaction partner, and that the gaze behavior was not optimal at the time. As a consequence, a more natural gaze behavior is being developed that will be coordinated more closely with the dialog state.

Subjective measures

Table 7.8 shows the results for the subjective measures. All aspects were generally rated positively, with ratings of more than 4 (on a scale from 1 to 6) for ease of use, clarity, pleasantness, understandability, robustness, cooperativeness and likeability and more than three for efficiency (both as single-item and as aggregated measure), usability, perceived intelligence and animacy.
As with the objective measures, we can observe large individual differences between participants: for all single-item measures, values range between 1 to 6, and also most aggregated measures exhibit a large range of values.

Measure	***Min***	***Max***	***Mean***	***Stdev***
Ease of use	1	6	4.25	1.43
Efficiency	1	6	3.57	1.45
Clarity	1	6	4.29	1.12
Pleasantness	1	6	4.04	1.43
Understandability	1	6	4.50	1.43
Efficiency	1.20	5.20	3.30	0.96
Usability	1.60	5.00	3.52	0.87
Robustness	1.60	6.00	4.23	1.07
Cooperativeness	2.33	6.00	4.05	0.88
Likeability	3.00	6.00	4.80	0.93
Perceived Intelligence	2.50	6.00	3.83	0.85
Animacy	1.86	6.00	3.55	0.93

Table 7.8: Results of subjective measures. Blocks are: i) Single-item measures, ii) aggregated measures.

Performance functions

Most of the performance functions resulting from the PARADISE evaluation (shown in table 7.9) appear plausible, but some of them point out unexpected relationships between measures that help to identify deficiencies of the system. The performance functions generally exhibit high R^2 values, indicating that they do explain much of the variance in the data.
For instance, the number of objects learned per minute the interaction is a strong predictor for the **ease of use**, together with the overall number of demonstrated objects and the duration of interaction, suggesting that users who found interaction easy and were successful in teaching objects tended to have longer interactions with the system.
In contrast, **usability** as an aggregated measure from questions on control, predictability, concentration, clarity of when to speak is explained by a different function. Here, not only the number of objects learned per minute has an impact, but also how many interaction steps were required for learning, and the number of completed interaction patterns (which can be interpreted as a measure for the general interaction success, taking into account not only the task-related but also the social aspects). Somewhat surprisingly, the user

Measure	*Function*	R^2		β	*Significance*
Ease of use	-0.30 + 4.33(Obj/min) + 1.15(ObjDemo) + 0.12(Time)	.471	Obj/min	.792	p<.001
			ObjDemo	.466	p<.05
			Time	.399	p=.059
Efficiency	No significant model				
Clarity	4.63 - 6.92(Repair) + 1.39(Obj/min)	.367	Repair	-.467	p<.01
			Obj/min	.336	p<.05
Pleasantness	3.58 + 0.18(Learn)	.184	Learn	.465	p<.05
Understandability	3.78 + 0.14(Ref)	.196	Ref	.443	p<.05
Efficiency	1.42 - 5.09(Repair) + 0.61($Steps_{ObjDemo}$) - 0.06(UU_{ooc}) - 0.10($Steps_{ObjReq}$)	.441	Repair	-.403	p<.05
			$Steps_{ObjDemo}$	.394	p<.05
			UU_{ooc}	-.404	p<.05
			$Steps_{ObjReq}$	-.316	p=.081
Usability	4.03 - 0.22($Steps_{ObjDemo}$) + 2.39(Obj/min) - 0.21(UU/min) + 0.03(NPattern)	.553	$Steps_{ObjDemo}$	-.747	p<.001
			Obj/min	.736	p<.001
			UU/min	-.521	p<.01
			NPattern	.391	p<.05
Robustness	5.07 - 6.08(Repair)	.183	Repair	-.428	p<.05
Cooperativeness	1.89 + 1.79(Obj_{corr}) + 0.001(Gaps)	.446	Obj_{corr}	.698	p<.001
			Gaps	.418	p<.05
Likeability	No significant model				
Perceived Intelligence	3.02 + 0.001(Gaps)	.171	Gaps	.171	p<.05
Animacy	3.67 + 1.09(Obj/min) - 0.118(UU/min)	.281	Obj/min	.443	p<.05
			UU/min	-.332	p=0.074

Table 7.9: Performance functions. Blocks are: i) Single-item measures, ii) aggregated measures.

utterances per minute had a negative impact on usability, and also on the **perceived intelligence**. Qualitative analysis revealed that for users who tend to talk rather fast and keep talking even during the robot's utterances, there is a risk of cumulating delays in the robot responses, as the user utterances are simply queued and processed one after another. This indicates the need for a more flexible turn-taking behavior that enables the robot to suspend or abort its own utterances, or to ignore user utterances if there is a newer one present.

Asking for **efficiency** directly did not yield a significant model. However, estimating **efficiency** as an aggregated measure (from questions on time required for object learning, clarity of referred objects, and general functionality of the system) suggests as predictors the number of repair utterances, interaction steps required for object demonstrations and object requests, as well as the out-of-capability utterances. The interaction steps for object demonstrations contribute positively, which may seem surprising at first. Looking at the interaction pattern used for object demonstrations, we realized that a high number of steps indicates multiple object corrections, while a low number of steps indicates that already the reference resolution fails. Thus, an explanation for the positive contribution of interaction steps might be that failures of reference resolution have a stronger negative effect on efficiency than correcting a misunderstood label, and are even more frustrating. The factors that influence **clarity** of interaction are the number of repair utterances and the number of objects learned per minute, with the former being a major measure for interaction success and the latter for task success. The **pleasantness** is predicted (not very strongly, though) only by the number of objects learned per minute, but the measures might miss some factors that are relevant for pleasantness.

Interestingly, we note a significant impact of reference resolution failures on the **understandability** score. This can be attributed to the fact that for such failures, the robot only reports that it could not determine what the user referred to, not why. This provides very little information toward good error recovery, leaving users guessing. Qualitative analysis confirms this interpretation: several users explicitly asked the robot why it was not able to determine the object referred to. This result points out again the frustrating effect of reference resolution failures, and the importance of more informative feedback in error conditions.

The **robustness** of the system as an aggregated measure (from questions on reliability and robustness of the system) is predicted by the proportion of repair utterances, but not by measures that specifically refer to object learning and recognition. This demonstrates that a robust interaction can alleviate deficiencies at task level.

Cooperativeness (aggregated from questions on the robot's readiness to interact, interest, attentiveness and autonomy) is affected not only by correctly learned objects, but also by gaps between utterances. While this may seem less obvious at first sight, it might be further evidence for the crucial role of turn-taking: longer gaps between utterances indicate a smooth turn-taking, and thus better speech recognition performance. It is probably for the same reason that gaps contribute positively to **perceived intelligence** as well.

Between-subjects factor: impact of the robot's task initiative

As mentioned above, the degree of the robot's task initiative was varied as a three-level between-subjects factor. Significant differences between the conditions were calculated based on both an ANOVA (i.e. the overall test on differences) and a post-hoc analysis (i.e. the pairwise comparison of the three groups). Many of the differences found are trivial, e.g. that in the conditions C2 and C3, where that the robot was asking for object labels on its own initiative, the associated Interaction Pattern occurred significantly more often. The non-trivial results for the subjective measures are shown in table 7.10. It can be seen that user initiative only requires a higher level of concentration than mixed or structured initiative, and that structured initiative requires even less concentration than mixed initiative. This indicates that the robot's initiative – particularly when explicitly releasing it –actually facilitates the interaction by providing guidance, thus reducing user's uncertainty what to do next. Similar results have been found for the tension the users felt during interaction. However, unexpectedly, users felt more tense in the structured initiative interactions than in the mixed initiative interactions.

In the robot initiative conditions C2 and C3, users found the robot more thorough than in the user initiative condition C1. Perhaps the robot's initiative gave the impression that it specifically asks for those objects that it was not able to classify reliably yet, in order to re-train them again. In fact, the objects it asked for were selected randomly.

Table 7.11 shows the non-trivial differences for the objective measures, indicating that in interaction with user initiative only, user utterances tend to be significantly longer than in interactions where the robot asks, too. As the user's replies to the robot's label query were usually very short, this result is not surprising, but it points out that having the robot ask for objects might be less error-prone. However, the repair utterances were not significantly lower in the robot initiative conditions.

A result that is difficult to explain is that in the conditions C2 and C2, where the robot actively asked for objects, users checked the robot's knowledge more often ("Flobi, what was this?"). Interestingly, the overall number of objects learned did not differ between the conditions.

Another difference could be found in the ratio between the mean length of user utterances and gaps. A high value indicates indicates shorter gaps in overall terms, while a low value indicates longer gaps. As shown in the table, gaps were shortest in the user initiative condition C1, and longest in the mixed-initiative condition C2. This result is supported by the qualitative analysis described in the next section: it was observed that, in the mixed-initiative condition, users often wait for the robot's next question instead of demonstrating something on their own. Obviously, this effect could be reduced in condition C3 by the robot's invitation to the user to demonstrate objects to it.

Questionnaire item	***C1 UserIni***	***C2 MixedIni***	***C3 StructuredIni***	***ANOVA Significance***
"Interaction always required highest concentration"	4.56	4.00	3.18	$p{=}.051$
"During interaction I felt tense"	5.38	3.75	4.18	$p{=}.070$
"I found the robot (sloppy...thorough)"	3.65	4.95	4.74	$p<.05$

Table 7.10: Questionnaire items for which both the ANOVA and post-hoc tests showed significant differences between the groups.

Objective Measure	***C1 UserIni***	***C2 MixedIni***	***C3 StructuredIni***	***ANOVA Significance***
Mean length of utterance in seconds (MLU)	1.39	0.94	1.11	$p<.001$
Number of object tests (ObjTest)	4.11	17.50	11.00	$p<.05$
Ratio of user utterances and gaps (UU/gaps)	.65	.49	.56	$p<.05$

Table 7.11: Objective measures for which both the ANOVA and post-hoc tests showed significant differences between the groups.

Qualitative analysis

Replacing objects during learning: It could be observed frequently that users replace the focused object with another one before learning had been completed. As the object recognizer completes the learning task only if the object to learn is successfully classified, learning will fail if the object is not present any more. While a few users continue to do so throughout the interaction, this particularly occurred at the first demonstration episode. On the robot's notification that it was beginning to learn most users returned the object to its original place, like in the following interaction excerpt:

U	Flobi, this is a banana.
F	Alright, this is a banana, is that correct?
U	Yes, that's correct, Flobi. (putting banana away)
F	Okay, I am going to learn the banana.
U	Oh, then I will put it here again.

Also, some users proceed to the next object immediately before learning had been completed. In this case, demonstration of the next object will be interpreted as a label correction of the previous one. Thus, the object recognizer will associate the view of the first object with the label of the second one (which the user is not aware of, though):

U	Flobi, this is a banana.
F	Alright, this is a banana, is that correct?
U	Yes, and this is an apple! (replacing banana with apple)
F	Alright, an apple, is that correct?
U	Yes.
F	Okay, I am going to learn the apple.

Repairing incorrect object references: Most users were very cooperative during the interaction with the robot. They actively tried to adapt to the robot's limitations, and to compensate these. This effect can be observed well with incorrect object references, that many users try to "repair" by replacing the object:

U	This is an apple.
F	Alright, this is a hand, is that correct?
U	This is a hand! (puts up hand)
F	Okay, I am going to learn the hand.

As above, the object recognizer will associate the first object with the second object's label. Similar repair effort can be observed in case of false-positive object references, i.e. when an utterance is erroneously interpreted as an object demonstration, which also leads to learning an incorrect label. From the user's point of view, however, the incorrect reference has been repaired:

U	Okay, that was the apple, let's learn the next object.
F	Alright, this is an apple, is that correct?
U	(puts apple in focus) Exactly, this is an apple.

Social utterances as fallback: As described previously, social utterances were widely used, in particular praising the robot for correctly learned or recognized objects. In addition, social utterances provide structure for the human as they are based on conventions well-known to the users, whereas it is initially not known how to demonstrate objects to the robot. Several interactions could be observed in which the users, for whom the first demonstration attempt had failed, fall back to social utterances before trying again.

U Flobi, I am going to show you some objects today.
F Sorry, I cannot recognize which object you mean.
U Uhm...my name is Marc.
F Hello Marc, nice to meet you.
U How are you?
F Fine, thank you.

Back-channel utterances: Often, the robot was praised not only for correctly learned or correctly recognized objects, but also for intermediate steps towards this goal, which had the function of back-channel utterances. Unfortunately, the dialog configuration was not prepared for that, i.e. interleaving the object demonstration or object test pattern with the pattern used for praising was not permitted. As a consequence, the robot reacted with clarification requests to these utterances. Nevertheless, users kept employing them throughout the entire interaction.

U Flobi, this is an apple.
F Alright, this is an apple, is that correct?
U Yes, Flobi.
F Okay, I am going to learn the apple.
U Very good, Flobi.
F Pardon?
F I have learnt the apple.

Staccato phrases: Some users tend to speak particularly slow to the robot, stressing the significant words, with noticeable pauses between the individual words. While this facilitates understanding for humans, automatic speech recognition experiences problems with it. Especially, due to the pauses between words, utterances are segmented into parts that cannot be interpreted in isolation:

U Flobi
F Pardon?
U This...this one
F Pardon?
F ...is a green...
F Pardon?
U ...a green apple
F Alright, this is an apple, is that correct?

Not accepting the robot's limitations: Some users did not accept, or realize, the robot's limitations. For instance, one participant said "yes, you can" in reply to the robot's notification that it was not able to identify the referred object. Another participant used to say "Flobi, you listen to me" in reaction to speech recognition errors or the robot's request to repeat the utterance. Also, the questionnaire results contained comments that the robot had not listened well.

Semantically complex utterances: As a special case of not realizing the robot's limitations, some users form sentences that go beyond the dialog system's current capabilities. For instance, in the below interaction excerpt, the dialog system is able to respond correctly to certain key words, but is not able to capture the entire meaning:

F	Sorry, I cannot recognize which object you mean.
U	I mean this object in my hand is a banana.
F	Alright, this is a hand, is that correct?
U	Yes, but in my hand I am holding a banana.
F	Okay, I am going to learn the hand.
U	The hand is not a banana.
F	Alright, a hand, is that correct?
U	My hand lies on the table.
F	Alright, a hand, is that correct?
U	Um, no, because you should learn what objects are lying on the table!
F	Pardon?

Adhering to one single interaction strategy: Users of any system tend to adhere to strategies that once have turned out to be successful. However, a few users strictly adhered to a specific interaction strategy or to a specific wording – even if it was not successful. In the below interaction excerpt, the user confirms the robot's request for label confirmation with a specific wording ("hast du richtig verstanden"), which unfortunately had not been included in the speech recognition grammar. Even worse, the speech recognition reproducibly interprets this utterances as the opposite, i.e. as a rejection ("das ist richtig falsch dann"). Although the utterance would have been processed correctly with a little variation in the wording, the participant keeps using this specific wording for each confirmation, up to 5 times in sequence and 14 times in total:

F	Alright, a banana, is that correct?
U	Have correctly understood. [In German: Hast du richtig verstanden.]
F	What's correct?
U	Banana was correct.
F	Alright, a banana, is that correct
U	Have correctly understood. [In German: Hast du richtig verstanden.]
F	What's correct?

Alignment effects: Repeatedly it could be observed that users tend to align their wording to the robot's utterances. For instance, users asked "will you have a look at it?", as a result of the robot announcing "I will have a look at it" before answering an object recognition request. Similarly, a user replied "Sorry, I can not recognize which object you mean" to the robot's label request, which is exactly what the robot says in the reverse case. Unfortunately, the system was not prepared to all of these utterances that users adopted from the robot.

Problems with initiative shift: In condition C2 (MixedIni), where the robot takes initiative at the begin of the interaction, but does not explicitly yield initiative later, problems with shifting initiative could be observed. Several participants did not demonstrate any objects on their own initiative, but waited for the robot to ask for the next object, even though this results in fairly long gaps between the teaching episodes sometimes. For this condition, we had the guideline that the time intervals between the robot's label queries (which were triggered by the WOz ROI selection) should vary, and that some of them should be sufficiently long to allow the user to take over initiative. One participant tried to speed up interaction by explicitly asking the robot to proceed to the next object. Unfortunately, the robot interprets the request as an announcement that the user will now demonstrate the next object, with the consequence that the interaction gets stuck:

F I have learnt the apple.
U Next.
F Okay.
[gap of 1:20 min]
F What is that?

7.5 Summary

This chapter has described the Curious Flobi system, which can be seen as a proof-of-concept for the proposed approach to dialog modeling, and has in turn advanced the approach by providing new use cases on object teaching. Besides the description of the scenario and the system architecture, a detailed account of the design process has been given: Based on experiences from the CeBit scenario, it has been decided to extend the capabilities of the Curious Robot for more user initiative. The analysis of the nonunderstandings in the CeBit scenario led to the decision to exchange the speech understanding component with more robust keyword matching. A WOz study on object teaching served as a guideline to design the interaction strategy for the Curious Flobi scenario, including the configuration of the speech recognition.
A large-scale user study has been conducted, in which a wide range of both subjective and objective measures have been calculated from the interactions. The evaluation addresses (i) the relationship between the objective and subjective measures, (ii) the influence of the robot's task initiative and (iii) a qualitative analysis of the interactions.

The relationship between objective and subjective measures has been investigated by means of a PARADISE-style regression analysis. The resulting performance functions have revealed some unexpected relations, e.g. the influence of turn-taking problems (as indicated by short gaps between utterances) on how cooperative and intelligent users perceived the robot. To a certain extent, the results are generalizable, e.g. the influence of badly designed system feedback on general understandability of the system.
The investigation of the robot's task initiative has shown differences between the conditions, but some of them were not immediately understandable. Also, some presumed differences could not be shown, e.g. differences in the number of learned objects. Thus, this issue deserves to be addressed in a future study with more participants per condition.
The qualitative analysis explains and supports result from the PARADISE and the between-subjects evaluation. Moreover, it points out several deficiencies of the system that future iterations need to address, e.g. the system must cope with (or prevent) the user replacing objects during the learning.

8 Further Scenarios

This chapter describes briefly a number of additional scenarios that have been implemented with the proposed PaMini framework, but not by the author herself. A variety of platforms and scenario types are represented, ranging from robots to virtual agents, from information-oriented to rather action-oriented interactions, relying on diverse forms of system back-ends. This demonstrates that (i) the suggested approach is comprehensible to developers, and (ii) that it is not restricted to a certain interaction type, but that it is versatile enough to be applied to very different scenarios.

8.1 Receptionist Vince

The Receptionist Vince scenario is a joint project which several working groups at CITEC Bielefeld are contributing to. Main actor in the scenario is the virtual agent Vince, whose task is to provide visitor information about the CITEC. Vince can generate synchronized multimodal output, as specified based on a multi-modal mark-up language, for which a new output source had to be added to PaMini. In the receptionist scenario, Vince communicates not only with the visitor, but also with the mobile robot BIRON, whose task is to show the visitors around. Figure 8.1 shows the interaction setup. An example dialog is given in figure 8.3.

From a dialog modeling perspective, the implementation still has a couple of shortcomings. Technically, the communication between Vince and BIRON has not (yet) been modeled as an actual multi-party dialog, but BIRON's part of the interaction is generated by a few of pre-scripted outputs. Also, as can be seen from figure 8.3, Vince instructing BIRON has not been modeled using an Action Request, but rather by combining several Notifications. This implies that failure cases can not be accounted for in a systematic manner based on the Task State Protocol, but are handled based on time-out conditions.

As listed in table 5.1, this scenario makes mainly use of Information Requests (modeling the visitor's questions), Statements (for modeling small-talk and exchange of pleasantries) and Notifications (for Vince's communication with BIRON).

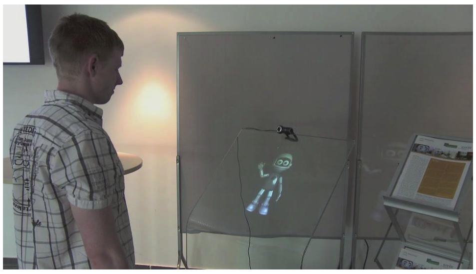


Figure 8.1: The Receptionist Vince.

Figure 8.2: ToBi at the 2011 RoboCup@Home, performing the FollowMe task.

U	Hello!	H Interaction Opening
V	Hi! I am Vince. How may I help you?	
U	Who are you?	H Information Request 1
V	My name is Vince. I'm a virtual agent and I contribute to the	
V	Dialog Demonstrator project.	H Information Request 1
U	And what can you do?	H Information Request 2
V	I can provide information about employees and working groups	
V	at the CITEC.	H Information Request 2
U	Oh, good. I am looking for Prof. Ritter.	H Information Request 3
V	You can find him in Q1-123.	H Information Request 3
V	Would you like BIRON, the mobile robot, to show you the way?	R Suggestion
U	Yes, please.	R Suggestion
V	Okay, please wait a moment.	R Suggestion
B	[is approaching]	
B	Hello Vince, my friend, what may I do for you?	
V	This is BIRON. He will take you to the lift.	R Notification
	BIRON, please take our guest to the lift.	R Notification
B	Alright, please follow me.	
U	Bye bye, Vince!	H Interaction Closing
V	Good bye.	H Interaction Closing

Figure 8.3: An example dialog with the Receptionist Vince.

8.2 The RoboCup@Home Challenge

The RoboCup@Home league involves several benchmark tests for service robots that measure their performance in realistic home environments, instructed through natural language. For the 2011 challenge, the team of Bielefeld (ToBI) has integrated the PaMini framework [WSZL11] for the first time. The example shown in figure 8.4 illustrates ToBi's interaction capabilities as demonstrated at the FollowMe task, shown in figure 8.2. Besides the demonstration of the robot's face recognition, person following and room classification

capabilities, the aim was to demonstrate the flexible interaction flow through interleaving patterns: while the robot is performing the person following task, it can still continue interaction – a feature that systems with a rather pre-structured dialog flow did not exhibit.
As is evident in table 5.1, the RoboCup scenario makes heavy use of all sorts of action-oriented Interaction Patterns, initiated both by the human and by the robot itself, often including explicit confirmation of the instruction in order to cope with the high background noise at the competition.

R	Alright, let's start the task!	R Notification
R	Hi! Please look into my eye while I tell you a bit about myself. My name is ToBI and I'm from Bielefeld.	R Self-Initiated Simple Action
R	What I'm doing right now is trying to learn your face. So please give me a few more seconds.	R Self-Initiated Simple Action
R	From now on I am able to recognize you!	R Self-Initiated Simple Action
R	Please tell me what to do!	R Notification
H	ToBi, follow me please!	H Cancellable Action Request
R	Alright, I will follow you now!	H Cancellable Action Request
H	What time is it?	H Information Request
R	It's 4pm.	H Information Request
H	ToBi, stop!	H Cancellable Action Request
R	Okay, I will stop here.	H Cancellable Action Request
H	ToBi, this is the kitchen!	H Simple Action Request with Expl. Conf.
R	Do you want me to remember the kitchen?	H Simple Action Request with Expl. Conf.
H	Yes, please!	H Simple Action Request with Expl. Conf.
R	Alright.	H Simple Action Request with Expl. Conf.

Figure 8.4: An example dialog from the 2011 RoboCup@Home challenge.

8.3 A Multi-Party Quiz Game with Nao

The Multi-Party Quiz Game has been developed within the scope of a master's project that addressed situated multi-party interaction [Klo10]. The work relies on Sidner's definition of *engagement states* and *engagement actions* [SKLL04], which has been extended by Bohus for the concept of *engagement intentions* [BH09]. Within the scope of this project, PaMini was extended so as to be able to manage multiple interactions at a time, each with one ore more participants. An engagement component makes engagement decisions based on the interaction partners' actions, their presumed engagement intention (based on cues such as distance or body posture), and the current dialog state.
A multi-party quiz game scenario with the robot Nao demonstrates the new interaction capabilities. Figure 8.5 shows the scenario setup. An interaction example is given in

figure 8.7. In this example, the robot first engages in interaction with person H1 by beginning a quiz game. As human H2 seems to have the intention of joining the interaction, the robot invites him or her to join the quiz game. Later, H1 disengages, but the interaction is continued with H2.
This scenario uses only few Interaction Patterns, mainly Information Requests for modeling the various quiz questions.

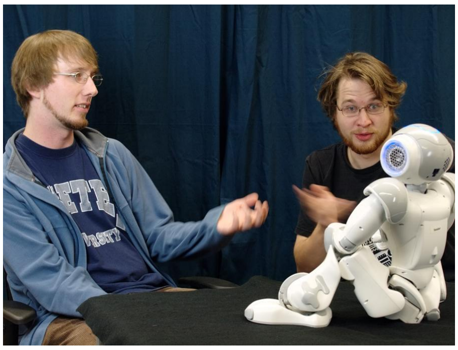


Figure 8.5: Multi-party quiz game with Nao

Figure 8.6: Nao explaining an exhibit.

H1	Hello, Robot.	H Interaction Opening
R	Hi, I am Nao!	H Interaction Opening
R	Do you want to play a quiz game with me?	R Suggestion 1
H1	Yes, I do.	R Suggestion 1
R	Fine.	R Suggestion 1
R	The question is: What is hypotension?	R Simple Information Request 1
H1	Low blood pressure.	R Simple Information Request 1
R	That was right, congratulations!	R Simple Information Request 1
R	[to the newly arrived H2]	
R	Hey, over there, do you want to join this quiz game?	R Suggestion 2
H2	Yes, sure.	R Suggestion 2
R	Fine.	R Suggestion 2
R	The question is:	
R	What is the founding year of Bielefeld University?	R Simple Information Request 2
H2	Uh, 1965?	R Simple Information Request 2
R	That was wrong, sorry!	R Simple Information Request 2
H1	I have to leave, bye!	H Interaction Closing
R	Good bye, have a nice day!	H Interaction Closing

Figure 8.7: An example dialog from the multi-party quiz game with Nao.

8.4 Art Exhibition Scenario

In the Art Exhibition scenario, Nao acts as a museum guide, offering visitors to provide background information about the exhibits. The robot's speech is accentuated by bodily gestures, as depicted in figure 8.6. Figure 8.8 shows an interaction excerpt, with the communicative gestures put in parenthesis. With this scenario, a study has been conducted that investigated whether the robot's nodding influences the user's decision on whether the robot should proceed with its explanations or not. A previous, non-interactive version of the scenario has been described in [PWSS11].
As table 5.1 illustrates, the scenario features a rather restricted interaction, relying mainly on Notifications (for the background information about the exhibits) and on Suggestions (for asking whether more information should be provided or not). On the other hand, it had to cope with real world conditions, including interactions with real visitors who were not prepared to interact with a robot, and perception under natural lighting conditions.

R	Hello.	R Interaction Opening
H	Hello, robot.	R Interaction Opening
R	My name is Nao. May I [openArms] tell you [nodding] something about the	R Suggestion
R	exhibited pictures?	R Suggestion
H	Yes, sure.	R Suggestion
R	Okay [nodding], great.	R Suggestion
R	Behind me [armsBack] there are six pictures [...]	R Notification
R	Do you want me [nodding] to explain another one?	R Suggestion
H	Yes, please.	R Suggestion
R	Fine [nodding].	R Suggestion

Figure 8.8: An example dialog from the Art Exhibition Scenario.

8.5 PlaSta Scenario

The PlaSta (Planetary Station) scenario has been developed within the scope of a research project which is supported by the German Aerospace Center (DLR). The scenario modeled an experiment which was conducted in a five day isolation study, with the aim to investigate which effects the robot's presence has on the participants' cognitive performance. In the experiment, participants had to solve a cognitive test on spatial perception [PLL^{+}95], with an example task depicted in figure 8.9. Nao acts as experimenter, guiding the participants through the individual steps, giving feedback and assistance.
As with the Art Exhibition scenario, Nao produces both speech and gestures. Also, the interaction is similarly restricted, in this case to achieve controlled conditions for the experiment. As above, the most frequently used patterns are Notifications (to introduce the task and provide information about the state of the task, time left, and performance)

and Suggestions (to offer assistance).

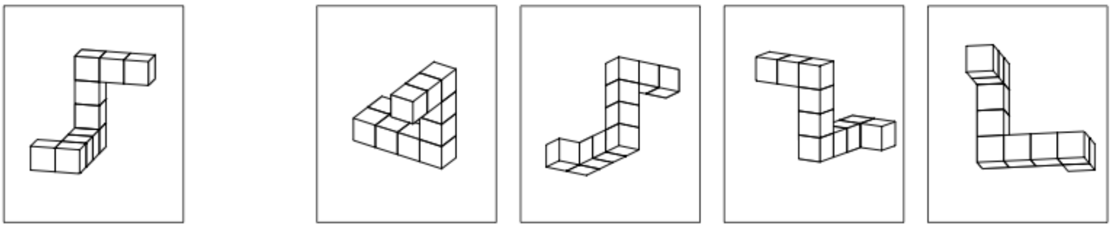


Figure 8.9: Example task from the PlaSta experiment.

R	[welcome] Hello and welcome to our study.	R Interaction Opening
H	Hello.	R Interaction Opening
R	Thanks for participating. [selfReference] My name is Nao. I will guide you through the experiment and do my best to assist you. Please touch the screen [showLeft] to start the experiment.	R Notification 1
R	Look at those [showLeft] four figures. All of them are show the same object, rotated by a certain angle. Please try to imagine the rotation of the objects by comparing the pictures. Please tell me when you are finished.	R Simple Statement
H	I'm finished.	R Simple Statement
R	Great. Here are another five [showLeft] images, showing a new	
R	[showRight] object. Two of them are equal to the one on the left side. Try to find them and mark the images by [push] touching the screen.	Robot Notification 2
H	[enters solution]	
R	You have answered the question correct right away, very [showRightUp] good. Can we [pointForwardAdmonishment] proceed?	Robot Notification 3 Robot Suggestion
H	Yes, let's go on.	Robot Suggestion
R	Fine.	Robot Suggestion

Figure 8.10: An example dialog from the PlaSta experiment.

8.6 Sports Companion

The Sports Companion scenario has been implemented withinin the scope of a student project in very short time (two weeks only). It does not run on on a real robot, though, but as a desktop application with simulated speech recognition. The main idea of the scenario was to realize an adaptive system whose degree of guidance varies according to the user's behavior. Based on several basic cues that were calculated on-line (such as number of different words per utterance, or gaps and overlaps between utterances), the system classifies the user as being *talkative* or *non-talkative*. The students implemented an example application for exercise planning. In this example application, non-talkative users receive more guidance by the system than talkative users, based on the assumption that talkative users provide more information on their own, and are in generally more comfortable while interacting with the system.

Figure 8.11 shows an example interaction with the Sports Companion application. The user type is categorized at interaction begin, based on the user's reaction to the systems attempt to engage in small-talk. The user's reply to the system's relatively open questions ("Do you work out a lot?") are not actually processed, but serve mainly as a basis for calculating the measures required for user classification. In the example, the user is classified as talkative. As a consequence, the system asks a more open question about the users preferences ("What type of session would you like to begin with?"), instead of suggesting a specific option ("Let's begin with an endurance session.").

As shown in table 5.1, the Sports Companion scenario relies, besides different types of Information Requests (for gathering information about exercise preferences), mainly on Notifications (to inform the user about constraints or number of remaining units), Statements (for small-talk) and Suggestions (for suggesting possible unit types if the user is non-talkative).

R	Hi, my name is Flobi. I would like to develop your personal exercise plan with you today.	R Interaction Opening
H	Hello, Flobi, nice to meet you. My name is Tom.	R Interaction Opening
R	How are you today?	R Simple Statement 1
H	Fine, thank you.	R Simple Statement 1
R	Do you work out a lot?	R Simple Statement 2
H	Yes, I jog every morning, and doing yoga in the afternoon.	R Simple Statement 1
R	What type of session would you like to begin with?	R Simple Information Request 1
H	With an endurance session.	R Simple Information Request 1
R	Okay.	R Simple Information Request 1
R	On what day?	R Simple Information Request 2
H	Tuesday.	R Simple Information Request 2
R	Okay, Tuesday.	R Simple Information Request 2

Figure 8.11: An example dialog between the Sports Companion and a *talkative* user.

8.7 Playing Memory with Flobi

In the Memory scenario, which is still under development, Flobi is supposed to invite people to play a memory game, as shown in figure 8.12 and 8.13. At the current, preliminary stage, a basic test on pairs has been realized to achieve a first integration. In general, the challenge of the scenario lies in the integration of interaction, perception and the control of game play.

Currently, the system makes mostly use of Notifications (for asking the user to arrange or turn over cards), and Suggestions (for finding out whether the robot has recognized the cards correctly or not). For asking the user to perform physical actions, a new Interaction Pattern has been found to be necessary, modeling the robot giving an instruction to the human, whose execution is monitored with the help of a perception component. Evidence for its necessity has also been found in a cycling scenario for which the development will start soon. In this scenario, the robot is supposed to accompany people during a cycling session, and to monitor (and possibly to correct) certain aspects of the exercise.

Figure 8.12: Playing memory with Flobi.

H	Hello, Flobi.	H Interaction Opening
R	Hi. I am ready to play with you.	H Interaction Opening
R	Do you want me to explain the rules first?	R Suggestion 1
H	Yes, please.	R Suggestion 1
R	Okay. Please turn over two cards,	
	and I will guess if it is a pair or not. Let's begin!	R Suggestion 1
H	[Turns over cards]	
R	The cards are 'dog' and 'cat' – am I right?	R Suggestion 2
H	No, I'm afraid not.	R Suggestion 2
R	Oh, I was mistaken, then.	R Suggestion 2
R	Would you like to play once more?	R Suggestion 3
H	No, thanks.	R Suggestion 3
R	Okay, see you later.	R Suggestion 3

Figure 8.13: An example dialog from the Memory scenario.

9 Summary and Contributions of the Thesis

The present thesis proposes a novel approach to dialog modeling on robots, which was implemented in the PaMini (Pattern-based Mixed Initiative) dialog framework. Its underlying concepts were developed based on extensive studies of the domain, accomplished through implementation of several preliminary scenarios, and were applied and evaluated in a number of studies and scenarios. The present work makes contributions the fields of i) dialog modeling, ii) human-robot interaction design, and iii) evaluation. In detail, the research done within the scope of this thesis can be summarized as follows.

A **Task State Protocol** is proposed that allows the dialog manager to request, monitor and revise tasks which are executed by the robotic subsystem. The protocol allows the dialog manager to treat all tasks in a uniform manner, thus establishing a well-defined component interface that facilitates system integration. It enables fine-grained yet standardized communication with the domain which, from interaction perspective, allows for more informative feedback on, and better interactive control of, ongoing actions.

The internal dialog model relies on **Interaction Patterns** that model recurring conversational structures of HRI at an abstract level. By combining task events from the Task State Protocol with dialog acts, they link dialog management and domain management. Interaction Patterns serve also as configurable building blocks of interactions, establishing the developer's API of the proposed framework. They abstract from, and encapsulate, the subtleties of dialog modeling and thus allow non-expert developers an easy access to tried and tested dialog procedures. At run-time, Interaction Patterns can be combined and interleaved in a flexible manner. Thus they provide the advantages of descriptive dialog modeling (understandability), while overcoming their restrictions (inflexibility in dialog flow).

The evaluation of the approach included a **developer-centered evaluation**. A case study was conducted in which the implementation of a typical HRI scenario with PaMini was compared to implementations with well-established dialog frameworks for non-situated domains. The case study demonstrates not only the efficacy of the framework, but also points out the differences between the investigated approaches and gives a review of state-of-the art approaches to dialog modeling. The comparison with other frameworks showed that the special focus on robotics – which has to the author's knowledge been addressed for the first time – indeed pays off, given the special nature of HRI. Moreover, framework usability was demonstrated in a usability test in which developers unfamiliar with the framework were able to develop a small interaction scenario within one hour. This shows that Interaction Patterns and the Task State Protocol are concepts that are easy to understand for developers unexperienced with dialog modeling and robotics.

Several **preliminary scenarios** were implemented, either with a different dialog manager, or with previous versions of PaMini. Most notably, the Home-Tour scenario, in which a mobile robot acquires a spatial model of its environment, and the Curious Robot scenario, an object learning and manipulation scenario with a humanoid robot. The preliminary scenarios greatly helped both to establish a detailed understanding of the domain and to identify essential use cases for HRI. Both scenarios feature a mixed-initiative dialog strategy for interactive learning. Explorative evaluations demonstrate that mixed initiative has the potential not only to facilitate the learning process (in case of the Home-Tour), but also the interaction itself (in case of the Curious Robot). A user study with a variation of the Curious Robot scenario was analyzed with respect to speech understanding problems. It revealed i) problems with the then-used speech understanding approach and ii) that users prefer a more active role, in particular at later stages of the interaction. Both issues were addressed in the next iteration of the system, the Curious Flobi scenario.

The **Curious Flobi** scenario was implemented based on the novel approach. In an iterative design process, several parameters were optimized: i) Based on the observations from the previous evaluation, the system capabilities were extended for more user initiative, ii) a WOz study on object teaching was analyzed with respect to the demonstration strategies users apply and assisted in the design of the dialog strategy, and iii) the speech recognition configuration underwent an informal test with respect to concept accuracy. The result of this process is a complex mixed-initiative object learning scenario which features not only task-related, but also social interaction. With this scenario, a large-scale user study based on the PARADISE approach became possible. A wide range of objective and subjective measures were related to each other in order to identify objective factors that are relevant for different aspects of user satisfaction. The study is one of the first attempts to apply the PARADISE approach to HRI, and by explaining up to 55% of variation in the data, it demonstrates that the chosen method is promising for conducting system-level evaluations, and that the chosen metrics were appropriate. Additionally, three different degrees of mixed initiative were compared. This evaluation has been complemented with a qualitative analysis which, on the one hand, helped to explain results from the PARADISE evaluation and, on the other hand, helped to identify deficiencies of the dialog strategy and the system in general.

A number of **further scenarios** have been implemented using the proposed approach. They were implemented not by the author herself, but by different developers, which demonstrates the usability and understandability of the approach. Some of them were implemented in a very short time, which demonstrates that the approach supports rapid-prototyping of interaction scenarios. Also, their high diversity demonstrates the versatility of the approach.

From the author's point of view, the three main contributions of the presented work are as follows.

- **Identifying issues crucial for HRI**: Based on literature review, experiences from example scenarios, and practical case-studies with different dialog frameworks, several issues have been identified that are crucial for implementing advanced HRI. Most notably, the necessity of a well-defined yet fine-grained interface between the dialog manager and the domain subsystem has been recognized. Another issue is the distribution of functionality between the dialog manager and the robotic back-end. The present work suggests that the global dialog flow is not determined by the dialog manager (as is often the case in traditional dialog modeling), but rather externally, enabling better reactivity to the robot's dynamic environment.
- **Providing constraints for developers**: Both the Task State Protocol and the Interaction Patterns restrict the developer's leeway for decisions. But as they were distilled from experiences concerning both system integration and interaction design, this does not represent a limitation, but a rather facilitation for the developers. This is supported by the ease with which a variety of scenarios was implemented by different developers.
- **Comprehensive evaluation of the proposed concept**: The proposed concepts evolved from a sample of basic use cases identified in preliminary example scenarios, and were then applied and tested in a wide range of new scenarios. The evaluation process included also the developer-centered view, an aspect often neglected in dialog modeling. The developed scenarios were integrated into a complex implementation-evaluation cycle that addressed various aspects, ranging from speech understanding performance to object teaching strategies, making use of various methods. Thus, not only the developed concepts, but also the chosen methodology – which represents a principled approach – are the contribution of the present work.

The concepts and results have been published as follows:

Journal articles

- J. Peltason, H. Rieser, S. Wachsmuth and B. Wrede. On Grounding Natural Kind Terms in Human-Robot Communication. KI - Künstliche Intelligenz, 27(2):107–118, 2013 ([PRWW13])
- J. Peltason and B. Wrede. The Curious Robot as a Case-Study for Comparing Dialog Systems. AI Magazine, 32(4):85–99, 2011. ([PW11])

Book chapters

- J. Peltason, H. Rieser, S. Wachsmuth: The hand is no banana! On communicating natural kind terms to a robot. In *Alignment in Communication: Towards a New Theory of Communication*, 2013. ([PRW13])
- J. Peltason and B. Wrede: Structuring Human-Robot-Interaction in Tutoring Scenarios. In *Towards Service Robots for Everyday Environments*, 2012. ([PWb])
- I. Lütkebohle, J. Peltason, L. Schillingmann, C. Elbrechter, S. Wachsmuth, B. Wrede, and R. Haschke: A Mixed-Initiative Approach to Interactive Robot Tutoring. In *Towards Service Robots for Everyday Environments*, 2012. ([PWa])

Conference papers

- J. Peltason, N. Riether, B. Wrede and I. Lütkebhole: Talking with Robots about Objects: A system-level evaluation in HRI. In *7th ACM/IEEE Conference on Human-Robot-Interaction (HRI)*, 2012. ([PRWL12])
- D. Klotz, J. Wienke, J. Peltason, B. Wrede, S. Wrede, V. Khalidov and J.-M. Odobez: Engagement-based Multi-party Dialog with a Humanoid Robot. In *SIGDIAL 2011 Conference, Association for Computational Linguistics*, 2011. ([KWP+11])
- J. Peltason, H. Rieser, S. Wachsmuth and B. Wrede. On Grounding Natural Kind Terms in Human-Robot Communication. KI - Künstliche Intelligenz, 27(2):107–118, 2013
- J. Peltason and B. Wrede: PaMini: A Framework for Assembling Mixed-Initiative Human-Robot Interaction from Generic Interaction Patterns. In *SIGDIAL 2010 Conference, Association for Computational Linguistics*, 2010. ([PW10b])
- J. Peltason and B. Wrede: Modeling Human-Robot Interaction Based on Generic Interaction Patterns. In *AAAI Fall Symposium: Dialog with Robots*, 2010. ([PW10a])
- I. Lütkebohle, J. Peltason, L. Schillingmann, C. Elbrechter, B. Wrede, S. Wachsmuth and R. Haschke: The Curious Robot - Structuring Interactive Robot Learning. In *IEEE International Conference on Robotics and Automation*, 2009. ([LPS+09])
- J. Peltason, F. Siepmann, T. P. Spexard, B. Wrede, M. Hanheide and E. A. Topp: Mixed-Initiative in Human Augmented Mapping. IEEE International Conference on Robotics and Automation, 2009. ([PSS+09])
- O. Booij, B. Kröse, J. Peltason, T. Spexard and M. Hanheide: Moving from Augmented to Interactive Mapping. In *Robotics: Science and Systems Conference*, 2008. ([BKP+08])

- N. Beuter, T. Spexard, I. Lütkebhole, J. Peltason, F. Kummert: Where is this? Gesture Based Multimodal Interaction With An Anthropomorphic Robot. In *IEEE-RAS International Conference on Humanoid Robots*, 2008. ([BSL+08])

Workshop papers

- J. Peltason, H. Rieser, S. Wachsmuth and B. Wrede: "The hand is not a banana". On Developing a Robot's Grounding Facilities. In *SemDial 2012 (SeineDial): The 16th Workshop on the Semantics and Pragmatics of Dialogue*, 2012. ([PRWW12])
- I. Lütkebohle, J. Peltason, B. Wrede and S. Wachsmuth: The Task-State Coordination Pattern, with Applications in Human-Robot-Interaction. In *Learning, Planning and Sharing Robot Knowledge for Human-Robot Interaction*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011. ([LPWW11])
- I. Lütkebohle, J. Peltason, R. Haschke, B. Wrede and S. Wachsmuth: The Curious Robot Learns Grasping in Multi-Modal Interaction. In *Interactive Communication for Autonomous Intelligent Robots*, 2010. ([LPH+10])
- J. Peltason: Position Paper. In *Young Reseachers' Roundtable on Spoken Dialog Systems*, 2010. ([Pel10])
- J. Peltason, I. Lütkebohle, B. Wrede and M. Hanheide: Mixed Initiative in Interactive Robotic Learning. In *Workshop on Improving Human-Robot Communication with Mixed-Initiative and Context-Awareness*, 2009. ([PLWH09])

Bibliography

[ABF02] James Allen, Nate Blaylock, and George Ferguson. A problem solving model for collaborative agents. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, AAMAS '02, pages 774–781, New York, NY, USA, 2002. ACM.

[AFS01] James Allen, George Ferguson, and Amanda Stent. An architecture for more realistic conversational systems. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, pages 1–8, New York, NY, USA, 2001. ACM.

[All01] Toward conversational human-computer interaction. *AI Magazine*, 22:27–37, October 2001.

[AND06] Aris Alissandrakis, Chrystopher L. Nehaniv, and Kerstin Dautenhahn. Action, state and effect metrics for robot imitation. In *The 15th IEEE International Symposium on Robot and Human Interactive Communication, ROMAN*, pages 232 –237, sept. 2006.

[Apa07] Apache Commons. Commons scxml, 2007. [Online: http://commons.apache.org/scxml; accessed 08-Feb-2012].

[Aus62] John L. Austin. *How to do things with words.* Harvard University Press, Cambridge, 1962.

[BCF+98] Wolfram Burgard, Armin B. Cremers, Dieter Fox, Dirk Hänel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. The Interactive Museum Tour-Guide Robot. In *Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[BCK08] Christoph Bartneck, Elizabeth Croft, and Dana Kulic. Measuring the anthropomorphism, animacy, likeability, perceived intelligence and perceived safety of robots. In *Metrics for Human-Robot Interaction Workshop in affiliation with the 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI 2008)*, 2008.

[BEG+96] Manuale Boros, Wieland Eckert, Florian Gallwitz, GÃ¼nther Goerz, Gerhard Hanrieder, and Heinrich Niemann. Towards Understanding Spontaneous

Speech: Word Accuracy vs. Concept Accuracy. In *Fourth International Conference on Spoken Language Processing ICSLP 96*. IEEE, 1996.

[BG00] Issam Bazzi and James R. Glass. Modeling out-of-vocabulary words for robust speech recognition. In *International Conference on Spoken Language Processing*, pages 401–404. ISCA, 2000.

[BG02] Rainer Bischoff and Volker Graefe. Dependable multimodal communication and interaction with robotic assistants. In *International Workshop on Robot and Human Interactive Communication*, pages 300–305. IEEE, December 2002.

[BH09] Dan Bohus and Eric Horvitz. Models for multiparty engagement in open-world dialog. In *SIGDIAL 2009 Conference*, pages 225–234. Association for Computational Linguistics, 2009.

[BHL02] Carl Burke, Lisa Harper, and Dan Loehr. A Flexible Architecture for a Multimodal Robot Control Interface. In *Intelligent Situation-Aware Media and Presentations Workshop*. AAAI, 2002.

[BKLO03] Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka. DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture. In *4th SIGdial Workshop on Discourse and Dialogue*, pages 115–124, 2003.

[BKP+08] Olaf Booij, Ben Kröse, Julia Peltason, Thorsten Spexard, and Marc Hanheide. Moving from augmented to interactive mapping. In *Proceedings of the Robotics: Science and Systems workshop Interactive Robot Learning*, pages 21–23, June 2008.

[BM10] Cindy Bethel and Robin Murphy. Review of Human Studies Methods in HRI and Recommendations. *International Journal of Social Robotics*, 2(4):347–359, December 2010.

[BPH+09] Kristy Elizabeth Boyer, Robert Phillips, Eun Young Ha, Michael D. Wallis, Mladen A. Vouk, and James C. Lester. Modeling dialogue structure with adjacency pair analysis and hidden markov models. In *Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL-Short '09, pages 49–52, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[BR05] Dan Bohus and Alexander Rudnicky. Larri: A language-based maintenance and repair assistant. In Wolfgang Minker, Dirk Bühler, Laila Dybkjær, and Nancy Ide, editors, *Spoken Multimodal Human-Computer Dialogue in Mobile Environments*, volume 28 of *Text, Speech and Language Technology*, pages 203–218. Springer Netherlands, 2005.

[BR08] Dan Bohus and Alexander I. Rudnicky. Sorry, I Didn't Catch That! In Laila Dybkjær, Wolfgang Minker, and Nancy Ide, editors, *Recent Trends in Discourse and Dialogue*, volume 39 of *Text, Speech and Language Technology*, pages 123–154. Springer Netherlands, 2008.

[BR09] Dan Bohus and Alexander I. Rudnicky. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech & Language*, 23(3):332–361, July 2009.

[Bre03] Cynthia Breazeal. Emotive qualities in lip-synchronized robot speech. *Advanced Robotics*, pages 97–113, 2003.

[BRH+07] Dan Bohus, Antoine Raux, Thomas K. Harris, Maxine Eskenazi, and Alexander I. Rudnicky. Olympus: an open-source framework fro conversational spoken language interface research. In *proceedings of HLT-NAACL 2007 workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technology*, 2007.

[Bri02] Derek Bridge. Towards conversational recommender systems: A dialogue grammar approach. In *Workshop in Mixed-Initiative Case-Based Reasoning at the Sixth European Conference in CaseBased Reasoning*, ECCBR 02, pages 9—22, 2002.

[BRM04] Trung H. Bui, Martin Rajman, and Miroslav Melichar. Rapid dialogue prototyping methodology. In P. Sojka, I. Kopecek, and K. Pala, editors, *Proceedings of the 7th International Conference on Text, Speech Dialogue (TSD)*, volume 3206/2 of *Lecture Notes in Computer Science*, pages 579–586, Berlin Heidelberg New York, September 2004. Springer Verlag.

[BRRL98] Susan E. Brennan, Pamela S. Ries, Claire Rubman, and Gregory Lee. The vocabulary problem in spoken language systems. In S. Luperfoy, editor, *Automated Spoken Dialog Systems*. MIT Press, 1998.

[BS09] Davide Brugali and Patrizia Scandurra. Component-based robotic engineering (Part I) [Tutorial. *IEEE Robotics & Automation Magazine*, 16(4):84–96, December 2009.

[BSL+08] Niklas Beuter, Thorsten Spexard, Ingo Lütkebohle, Julia Peltason, and Franz Kummert. Where is this? - Gesture Based Multimodal Interaction With An Anthropomorphic Robot. In *International Conference on Humanoid Robots*. IEEE-RAS, 2008.

[BWB09] Andrea Bauer, Dirk Wollherr, and Martin Buss. Information retrieval system for human-robot communication: asking for directions. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ICRA'09, pages 1522–1527, Piscataway, NJ, USA, 2009. IEEE Press.

[BWK+03] Hans-Joachim Böhme, Torsten Wilhelm, Jürgen Key, Carsten Schauer, Christof Schröter, Horst-Michael Groß, and Torsten Hempel. An approach to multi-modal human-machine interaction for intelligent service robots. *Robotics and Autonomous Systems*, 44(1), 2003.

[Car96] Jean Carletta. Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics*, 22:249–254, June 1996.

[CC01] Chih-Jung Chiang and Gregory Chirikjian. Modular robot motion planning using similarity metrics. *Autonomous Robots*, 10:91–106, 2001. 10.1023/A:1026552720914.

[CDN88] John P. Chin, Virginia A. Diehl, and Kent L. Norman. Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '88, pages 213–218, New York, NY, USA, 1988. ACM.

[CET08] Task Model Description (CE TASK 1.0), CEA-2018 (ANSI). 2008.

[CO95] Philip R Cohen and Sharon L Oviatt. The role of voice input for human-machine communication. *Proceedings of the National Academy of Sciences of the United States of America*, 92(22):9921–9927, 1995.

[COB10] Sonia Chernova, Jeff Orkin, and Cynthia Breazeal. Crowdsourcing HRI through online multiplayer games. In *AAAI Fall Symposium: Dialog with Robots*. AAAI, 2010.

[CR01] Ananlada Chotimongkol and Alexander I. Rudnicky. N-best speech hypotheses reordering using linear regression. In *INTERSPEECH*, pages 1829–1832, 2001.

[Dau07] Kerstin Dautenhahn. Socially intelligent robots: dimensions of human–robot interaction. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1480):679–704, 2007.

[Den02] Matthias Denecke. Rapid prototyping for spoken dialogue systems. In *19th International Conference on Computational Linguistics*, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[DG95] Morena Danieli and Elisabetta Gerbino. Metrics for evaluating dialogue strategies in a spoken language system. In *AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation*, volume 16. AAAI, 1995.

[DHB+06] Mary B Dias, Thomas K. Harris, Brett Browning, Edward G. Jones, Brenna Argall, Manuela Veloso, Anthony Stentz, and Alexander I. Rudnicky. Dynamically Formed Human-Robot Teams Performing Coordinated Tasks. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone*, 2006.

[DKF01] Elyon DeKoven, David V. Keyson, and Adinda Freudenthal. Designing collaboration in consumer products. In *CHI '01 extended abstracts on Human factors in computing systems*, CHI EA '01, pages 195–196, New York, NY, USA, 2001. ACM.

[DSBS09] Juraj Dzifcak, Matthias Scheutz, Chitta Baral, and Paul Schermerhorn. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 4163 –4168, may 2009.

[DW02] Kerstin Dautenhahn and Iain Werry. A quantitative technique for analysing robot-human interactions. In *IEEE/RSJ Intelligent Robots and Systems*, volume 2, 2002.

[EAC00] Christof Eberst, Magnus Andersson, and Henrik I. Christensen. Vision-based door-traversal for autonomous mobile robots. In *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 1, pages 620 –625, 2000.

[EH99] Renée Elio and Afsaneh Haddadi. On abstract models and conversation policies. *Issues in Agent Communication*, pages 301–314, 1999.

[EJ00] Annika F. Eriksson and Arne Jönsson. Dialogue and domain knowledge management in dialogue systems. In *Proceedings of the 1st SIGdial workshop on Discourse and dialogue - Volume 10*, SIGDIAL '00, pages 121–130, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[EKB11] Friederike Eyssel, Dieta Kuchenbrandt, and Simon Bobinger. Effects of anticipated human-robot interaction and predictability of robot behavior on perceptions of anthropomorphism. In *Proceedings of the 6th international conference on Human-robot interaction*, HRI '11, pages 61–68, New York, NY, USA, 2011. ACM.

[EKL+11] Wolfang Echelmeyer, Alice Kirchheim, Achim L. Lilienthal, Hülya Akbiyik, and Marco Bonini. Performance indicators for robotics systems in logistics applications. In *IROS Workshop on Metrics and Methodologies for Autonomous Robot Teams in Logistics (MMART-LOG)*, 2011.

[FA98] George Ferguson and James F. Allen. TRIPs: an integrated intelligent problem-solving assistant. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, pages 567–572, Menlo Park, CA, USA, 1998. AAAI.

[FAM96] George Ferguson, James F. Allen, and Bradford W. Miller. TRAINS-95: Towards a mixed-initiative planning assistant. In *Third Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 70–77, 1996.

[FDH+04] Jim Ferrans, Peter Danielsen, Andrew Hunt, Brad Porter, Bruce Lucas, Scott Mcglashan, Steph Tryphonas, Ken Rehor, Daniel C. Burnett, and Jerry Carter. Voice Extensible Markup Language (VoiceXML) Version 2.0. Technical report, W3C, March 2004.

[FE07] Raquel Fernandez and Ulle Endriss. Abstract models for dialogue protocols. *Journal of Logic, Language and Information*, 16(2):121–140, April 2007.

[FFMM94] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire. KQML as an agent communication language. In *CIKM '94: Proceedings of the third international conference on Information and knowledge management*, pages 456–463, New York, NY, USA, 1994. ACM.

[FGK09] Mary E. Foster, Manuel Giuliani, and Alois Knoll. Comparing objective and subjective measures of usability in a human-robot dialogue system. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 879–887, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[Fin99] Gernot A. Fink. Developing HMM-based recognizers with ESMERALDA. In *Proceedings of the Second International Workshop on Text, Speech and Dialogue*, TSD '99, pages 229–234, London, UK, UK, 1999. Springer-Verlag.

[Fon03] Terrence W. Fong. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42(3-4):143–166, March 2003.

[FTB01] Terrence W. Fong, Charles Thorpe, and Charles Baur. Collaboration, Dialogue, and Human-Robot Interaction. In *Proceedings of the 10th International Symposium of Robotics Research*, London, 2001. Springer Verlag.

[FW07] Jannik Fritsch and Sebastian Wrede. *An Integration Framework for Developing Interactive Robots*, volume 30 of *Springer Tracts in Advanced Robotics*, pages 291–305. Springer, Berlin, 2007.

[Gat08] Ioannis Gatsoulis. Performance metrics and human-robot interaction for teleoperated systems by. *Mechanical Engineering*, 2008.

[GDR+08] Sudeep Gandhe, David DeVault, Antonio Roque, Bilyana Martinovski, Ron Artstein, Anton Leuski, Jillian Gerten, and David R. Traum. From domain specification to virtual humans: An integrated approach to authoring tactical questioning characters. In *Proceedings of Interspeech*, Brisbane, Australia, 2008. International Speech Communication Association (ISCA).

[GSBR99] Dan Gruen, Candy Sidner, Carolyn Boettner, and Charles Rich. A collaborative assistant for email. In *CHI '99: CHI '99 extended abstracts on Human factors in computing systems*, pages 196–197, New York, NY, USA, 1999. ACM Press.

[Har87] David Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8:231–274, 1987.

[HDM+90] Lynette Hirschman, Deborah A. Dahl, Donald P. McKay, Lewis M. Norton, and Marcia C. Linebarger. Beyond class A: a proposal for automatic evaluation of discourse. In *Proceedings of the workshop on Speech and Natural Language*, HLT '90, pages 109–113, Stroudsburg, PA, USA, 1990. Association for Computational Linguistics.

[HG00] Kate S. Hone and Robert Graham. Towards a tool for the Subjective Assessment of Speech System Interfaces (SASSI). *Nat. Lang. Eng.*, 6(3-4):287–303, 2000.

[HHB99] Mark G. Heather, Heather Holmback, and Jeffrey Bradshaw. What Is a Conversation Policy? In *Workshop on Specifying and Implementing Conversation Policies, Autonomous Agents '99*, pages 118–131, 1999.

[HHS+10] Nick Hawes, Marc Hanheide, Kristoffer Sjöö, Alper Aydemir, Patric Jensfelt, Moritz Göbelbecker, Michael Brenner, Hendrik Zender, Pierre Lison, Ivana Kruijff-Korbayov, Geert-Jan Kruijff, and Michael Zillich. Dora the explorer: A motivated robot. In *AAMAS '10: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 1617–1618. International Foundation for Autonomous Agents and Multiagent Systems, May 2010.

[HKK+08] Frank Hegel, Soeren Krach, Tilo Kircher, Britta Wrede, and Gerhard Sagerer. Theory of mind (ToM) on robots: a functional neuroimaging study. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, HRI '08, pages 335–342, New York, NY, USA, 2008. ACM.

[HLI+10] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Pascal Fua, and Nassir Navab. Dominant orientation templates for real-time detection of texture-less objects. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:2257–2264, 2010.

[HM06] Melita Hajdinjak and France Mihelic. The PARADISE Evaluation Framework: Issues and Findings. *Computational Linguistics*, 32(2):263–272, June 2006.

[HMP08] Chin C. Ho, Karl F. MacDorman, and Z. A. Dwi D. Pramono. Human emotion and the uncanny valley: a GLM, MDS, and Isomap analysis of robot video ratings. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, HRI '08, pages 169–176, New York, NY, USA, 2008. ACM.

[HP93] Lynette Hirschman and Christine Pao. The cost of errors in a spoken language system. In *EUROSPEECH'93*, 1993.

[HR10] Philip Hanson and Charles Rich. A Non-Modal Approach to Integrating Dialogue and Action. In *AIIDE 2010: AI and Interactive Digital Entertainment Conference*, 2010.

[HSW+06] Frank Hegel, Thorsten Spexard, Britta Wrede, Gernot Horstmann, and Thurid Vogt. Playing a different imitation game: Interaction with an empathic android robot. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 56 –61, 2006.

[Hul00] Joris Hulstijn. Dialogue Games are Recipes for Joint Action. In *Proceedings of Gotalog '00, 4 th Workshop on the Semantics and Pragmatics of Dialogues, Gothenburg, Sweden*, pages 00–5, 2000.

[HWS+03] Stefan W. Hamerich, Yu-Fang H. Wang, Volker Schubert, Volker Schless, and Stefan Igel. XML-based dialogue descriptions in the GEMINI project. In *Berliner XML Tage*, pages 404–412, 2003.

[HWS06] Sonja Hüwel, Britta Wrede, and Gerhard Sagerer. Robust speech understanding for multi-modal human-robot communication. In *15th International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 45–50. IEEE Press, IEEE Press, 2006.

[IST+10] Naoto Iwahashi, Komei Sugiura, Ryo Taguchi, Takayuki Nagai, and Tadahiro Taniguchi. Robots that learn to communicate: A developmental approach to personally and physically situated human-robot conversations. In *AAAI Fall Symposium: Dialog with Robots*. AAAI, 2010.

[KFS10] Rachel Kirby, Jodi Forlizzi, and Reid Simmons. Affective social robots. *Robotics and Autonomous Systems*, 58:322–332, March 2010.

[KGK+09] Mare Koit, Olga Gerassimenko, Riina Kasterpalu, Andriela Raabis, and Krista Strandson. Towards computer-human interaction in natural language. *International Journal of Computer Applications in Technology*, 34:291–297, March 2009.

[Kir96] Jurek Kirakowski. *Usability Evaluation in Industry*, chapter The Software Usability Measurement Inventory: Background and Usage, pages 169–177. London: Taylor & Francis, 1996.

[KJL10] Geert-Jan M Kruijff, Miroslav Janicek, and Pierre Lison. Continual processing of situated dialogue in human-robot collaborative activities. In *RO-MAN, 2010 IEEE*, pages 594 –599, 2010.

[KKERK03] Ivana Kruijff-Korbayová, Stina Ericsson, Kepa J. Rodríguez, and Elena Karagjosova. Producing contextually appropriate intonation in an information-state based dialogue system. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 227–234, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[Klo10] David Klotz. Modeling engagement in a multi-party human-robot dialog. Masters thesis, Bielefeld University, 2010.

[Kön09] Ralf König. State-Based Modeling Method for Multiagent Conversation Protocols and Decision Activities. In Jaime G. Carbonell, Jörg Siekmann, Ryszard Kowalczyk, Jörg P. Müller, Huaglory Tianfield, and Rainer Unland, editors, *Agent Technologies, Infrastructures, Tools, and Applications for E-Services*, volume 2592 of *Lecture Notes in Computer Science*, chapter 13, pages 151–166. Springer, Berlin, Heidelberg, 2009.

[KWK07] Stephan Kirstein, Heiko Wersing, and Edgar Körner. A biologically motivated visual memory architecture for online learning of objects. *Neural Networks*, 2007.

[KWL99] Candace A. Kamm, Marilyn A. Walker, and Diane J. Litman. Evaluating Spoken Language Systems. In *American Voice InputOutput Society AVIOS*, 1999.

[KWP+11] David Klotz, Johannes Wienke, Julia Peltason, Britta Wrede, Sebastian Wrede, Vasil Khalidov, and Jean-Marc Odobez. Engagement-based multiparty dialog with a humanoid robot. In *SIGDIAL 2011 Conference*, page 341–343. Association for Computational Linguistics, Association for Computational Linguistics, 17/06/2011 2011.

[LBGP01] Oliver Lemon, Anne Bracy, Alexander Gruenstein, and Stanley Peters. The WITAS multi-modal dialogue system. 2001.

[Lem03] Oliver Lemon. Managing dialogue interaction: A multi-layered approach. In *Proceedings of the 4th SIGDial Workshop on Discourse and Dialogue*, 2003.

[Lem04] Oliver Lemon. Context-sensitive speech recognition in ISU dialogue systems: results for the grammar switching approach. In *8th Workshop on the Semantics and Pragmatics of the Dialogue, CATALOGUE'04*, 2004.

[Lev83] Stephen Levinson. *Pragmatics*. MIT Press, Cambridge, 1983.

[LF94] Yannis Labrou and Tim Finin. A semantics approach for KQML—a general purpose communication language for software agents. In *Proceedings of the third international conference on Information and knowledge management*, CIKM '94, pages 447–455, New York, NY, USA, 1994. ACM.

[LGBP02] Oliver Lemon, Alexander Gruenstein, Alexis Battle, and Stanley Peters. Multi-tasking and collaborative activities in dialogue systems. In *Proceedings of the 3rd SIGdial workshop on Discourse and dialogue*, pages 113–124, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[LHL+09] Christian Lang, Marc Hanheide, Manja Lohse, Heiko Wersing, and Gerhard Sagerer. Feedback interpretation based on facial expressions in human–robot interaction. In *International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2009.

[LHRS09] Manja Lohse, Marc Hanheide, Katharina J. Rohlfing, and Gerhard Sagerer. Systemic interaction analysis (sina) in hri. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, HRI '09, pages 93–100, New York, NY, USA, 2009. ACM.

[LHS00] Diane J. Litman, Julia B. Hirschberg, and Marc Swerts. Predicting automatic speech recognition performance using prosodic cues. In *Proceedings of NAACL-00*, pages 218–225, 2000.

[LHS+07] Manja Lohse, Frank Hegel, Agnes Swadzba, Katharina Rohlfing, Sven Wachsmuth, and Britta Wrede. What can i do for you? appearance and application of robots. In *Proceedings of "The Reign of Catz and Dogz? The role of virtual creatures in a computerised society"*, AISB'07, pages 121–126. Newcastle University, 2007.

[LHS+10] Ingo Lütkebohle, Frank Hegel, Simon Schulz, Matthias Hackel, Britta Wrede, Sven Wachsmuth, and Gerhard Sagerer. The Bielefeld Anthropomorphic Robot Head "Flobi". In *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010.

[LHW+05] Shuyin Li, Axel Haasch, Britta Wrede, Jannik Fritsch, and Gerhard Sagerer. Human-style interaction with a robot for cooperative learning of scene objects. In *Proceedings of the 7th international conference on Multimodal interfaces*, ICMI '05, pages 151–158, New York, NY, USA, 2005. ACM.

[Li07] Shuyin Li. *Multi-modal Interaction Management for a Robot Companion.* Phd, Bielefeld University, Bielefeld, 2007.

[LKF+04] Shuyin Li, Marcus Kleinehagenbrock, Jannik Fritsch, Britta Wrede, and Gerhard Sagerer. "BIRON, let me show you something": evaluating the interaction with a robot companion. In *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics*, pages 2827–2834, 2004.

[LLC+00] Staffan Larsson, Peter Ljunglöf, Robin Cooper, Elisabet Engdahl, and Stina Ericsson. Godis: an accommodating dialogue system. In *ANLP/NAACL Workshop on Conversational systems*, ANLP/NAACL-ConvSyst '00, pages 7–10, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[LM77] James A. Levin and James A. Moore. Dialogue-games: metacommunication structures for natural language interaction. *Cognitive Science*, 1(4):395–420, 1977.

[LPH+10] Ingo Lütkebohle, Julia Peltason, Robert Haschke, Britta Wrede, and Sven Wachsmuth. The curious robot learns grasping in multi-modal interaction. In *Interactive Communication for Autonomous Intelligent Robots*, Anchorage, AK, USA, 2010. video submission with abstract.

[LPP+11] Ingo Lütkebohle, Roland Philippsen, Vijay Pradeep, Eitan Marder-Eppstein, and Sven Wachsmuth. Generic middleware support for coordinating robot software components: The Task-State-Pattern. *Journal of Software Engineering for Robotics*, 1, 2011.

[LPS+09] Ingo Lütkebohle, Julia Peltason, Lars Schillingmann, Britta Wrede, Sven Wachsmuth, Christof Elbrechter, and Robert Haschke. The curious robot - structuring interactive robot learning. In *IEEE International Conference on Robotics and Automation*, pages 4156 –4162, may 2009.

[LPWW11] Ingo Lütkebohle, Julia Peltason, Britta Wrede, and Sven Wachsmuth. The Task-State Coordination Pattern, with applications in Human-Robot-Interaction. In *Learning, Planning and Sharing Robot Knowledge for Human-Robot Interaction*, number 10401 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2011. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany.

[LRGB99] Lori Lamel, Sophie Rosset, Jean-Luc Gauvain, and Samir Bennacef. The Limsi Arise system for train travel information. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 501–504, 1999.

[LSR06] Tim Laue, Kai Spiess, and Thomas Röfer. Simrobot - a general physical robot simulator and its application in robocup. In A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, editors, *RoboCup 2005: Robot Soccer World Cup IX*, volume 4020 of *Lecture Notes in Artificial Intelligence*, pages 173–183. Springer, 2006.

[LT00] Staffan Larsson and David R. Traum. Information state and dialogue management in the trindi dialogue move engine toolkit. *Nat. Lang. Eng.*, 6:323–340, September 2000.

[Lüt11] Ingo Lütkebohle. *Coordination and composition patterns in the "Curious Robot" scenario.* PhD thesis, Bielefeld, Germany, 2011.

[MAF+99] Toshihiro Matsui, Hideki Asoh, John Fry, Youichi Motomura, Futoshi Asano, Takio Kurita, Isao Hara, and Nobuyuki Otsu. Integrated natural spoken dialogue system of Jijo-2 mobile robot for office services. In *Sixteenth National Conference on Artificial intelligence and the Eleventh Innovative Applications of Artificial Intelligence conference*, pages 621–627, Menlo Park, CA, USA, 1999. AAAI.

[Man88] William C. Mann. Dialogue games: Conventions of human interaction. *Argumentation*, 2(4):511–532, November 1988.

[MB95] Mark S. Fox Mihai Barbuceanu. Cool: A language for describing coordination in multi agent systems. In *First International Conference on Multiagent Systems*, pages 17–24, 1995.

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

[McT04] Michael F. McTear. *Spoken dialogue technology - toward the conversational user interface.* Springer, 2004.

[Mea55] George H. Mealy. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34(5):1045–1079, 1955.

[MEBF+10] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The Office Marathon: Robust navigation in an indoor office environment. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 300–307, May 2010.

[MES08] Sebastian Möller, Klaus-Peter Engelbrecht, and Robert Schleicher. Predicting the quality and usability of spoken dialogue services. *Speech Communication*, 50(8-9):730–744, August 2008.

[MHF06] Bilge Mutlu, Jessica K. Hodgins, and Jodi Forlizzi. A Storytelling Robot: Modeling and Evaluation of Human-like Gaze Behavior. In *Proceedings of HUMANOIDS'06, 2006 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2006.

[Nie94] Jakob Nielsen. *Usability engineering*. Morgan Kaufmann Publishers, San Francisco, Calif., 1994.

[NIN+10] Mikio Nakano, Naoto Iwahashi, Takayuki Nagai, Taisuke Sumii, Xiang Zuo, Ryo Taguchi, Takashi Nose, Akira Mizutani, Tomoaki Nakamura, Muhammad Attamim, Hiromi Narimatsu, Kotaro Funakoshi, and Yuji Hasegawa. Grounding new words on the physical world in multi-domain human-robot dialogues. In *AAAI Fall Symposium: Dialog with Robots*. AAAI, 2010.

[NKS06] Tatsuya Nomura, Takayuki Kanda, and Tomohiro Suzuki. Experimental investigation into influence of negative attitudes toward robots on human-robot interaction. *AI & Society*, 20:138–150, February 2006.

[NKSK04] Tatsuya Nomura, Takayuki Kanda, Tomohiro Suzuki, and Kennsuke Kato. Psychology in human-robot communication: an attempt through investigation of negative attitudes and anxiety toward robots. In *13th IEEE International Workshop on Robot and Human Interactive Communication, ROMAN 2004*, pages 35–40, 2004.

[NKSK08] Tatsuya Nomura, Takayuku Kanda, Tomohiro Suzuki, and Kennsuke Kato. Prediction of human behavior in human–robot interaction using psychological scales for anxiety and negative attitudes toward robots. *Robotics, IEEE Transactions on*, 24(2):442 –451, april 2008.

[NMH+99] Mikio Nakano, Noboru Miyazaki, Jun-ichi Hirasawa, Kohji Dohsaka, and Takeshi Kawabata. Understanding unsegmented user utterances in real-time spoken dialogue systems. In *37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 200–207, Stroudsburg, PA, USA, 1999. Association for Computational Linguistics.

[NMY+00] Mikio Nakano, Noboru Miyazaki, Norihito Yasuda, Akira Sugiyama, Jun-ichi Hirasawa, Kohji Dohsaka, and Kiyoaki Aikawa. WIT: A Toolkit for Building Robust and Real-Time Spoken Dialogue. In *Proceedings of SIGdial Workshop,150-159*, pages 150–159, 2000.

[OM00] Ian M. O'Neill and Michael F. McTear. Object-oriented modelling of spoken language dialogue systems. *Natural Language Engineering*, 6(3& 4):341–362, 2000.

[ON98] P. D. O'Brien and Richard C. Nicol. FIPA - Towards a Standard for Software Agents. *BT Technology Journal*, 16:51–59, July 1998.

[Par96] Van Dyke Parunak. Visualizing Agent Conversations: Using Enhanced Dooley Graphs for Agent Design and Analysis. In *Proceedings of Second International Conference on Multi-Agent Systems (ICMAS'96)*, pages 275–282, 1996.

[PBRH07] Christopher Parlitz, Winfried Baum, Ulrich Reiser, and Martin Hägele. Intuitive Human-Machine-Interaction and Implementation on a Household Robot Companion. In *Proceedings of the 12th International Conference on Human-Computer Interaction*, 2007.

[PCR+08] Xavier Perrin, Ricardo Chavarriaga, Céline Ray, Roland Siegwart, and José del. A comparative psychophysical and EEG study of different feedback modalities for HRI. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, HRI '08, pages 41–48, New York, NY, USA, 2008. ACM.

[Pel10] Julia Peltason. Position paper. In *Young Reseachers' Roundtable on Spoken Dialog Systems*, Tokyo, Japan, 2010.

[PH05] Roberto Pieraccini and Juan Huerta. Where do we go from here? Research and commercial spoken dialog systems. In *Proceeedings of the 6th SIGDdial Worlshop on Discourse and Dialogue*, pages 1–10, September 2005.

[PLL+95] Michael Peters, Bruno Laeng, Kerry Latham, Marla Jackson, Raghad Zaiyouna, and Chris Richardson. A redrawn Vandenberg and Kuse mental rotations test: different versions and factors that affect performance. *Brain and cognition*, 28(1):39–58, June 1995.

[PLWH09] Julia Peltason, Ingo Lütkebohle, Britta Wrede, and Marc Hanheide. Mixed initiative in interactive robotic learning. In *Workshop on Improving Human-Robot Communication with Mixed-Initiative and Context-Awareness*, Toyama, Japan, 2009.

[Poh09] Marian Pohling. Verhaltensweisen zur Steuerung der Blickrichtung eines humanoiden Roboters. Technical report, Bielefeld University, 2009. Bachelor Thesis.

[PRW13] Julia Peltason, Hannes Rieser, and Sven Wachsmuth. The hand is no banana – on communicating natural kind terms to a robot. Alignment in Communication: Towards a New Theory of Communication. 2013.

[PRWL12] Julia Peltason, Nina Riether, Britta Wrede, and Ingo Lütkebohle. Talking with robots about objects: a system-level evaluation in hri. In *HRI*, pages 479–486, 2012.

[PRWW12] Julia Peltason, Hannes Rieser, Sven Wachsmuth, and Britta Wrede. The hand is not a banana. on developing a robot's grounding facilities. Proceedings of SemDial 2012 (SeineDial): The 16th Workshop on the Semantics and Pragmatics of Dialogue, pages 179–181, 2012.

[PRWW13] Julia Peltason, Hannes Rieser, Sven Wachsmuth, and Britta Wrede. On grounding natural kind terms in human-robot communication. *KI - Künstliche Intelligenz*, 27(2):107–118, 2013.

[PSS+09] Julia Peltason, Frederic H.K. Siepmann, Thorsten P. Spexard, Britta Wrede, Marc Hanheide, and Elin A. Topp. Mixed-initiative in human augmented mapping. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 2146 –2153, 2009.

[PWa] Julia Peltason and Britta Wrede. A mixed-initiative approach to interactive robot tutoring. In E. Prassler, W. Burgard, U. Handmann, R. Haschke, M. Hägele, G. Lawitzky, B. Nebel, W. Nowak, P. Pläger, U. Reiser, and M. Zöllner, editors, *Towards Service Robots for Everyday Environments*. Springer. In press.

[PWb] Julia Peltason and Britta Wrede. Structuring human-robot-interaction in tutoring scenarios. In E. Prassler, W. Burgard, U. Handmann, R. Haschke, M. Hägele, G. Lawitzky, B. Nebel, W. Nowak, P. Pläger, U. Reiser, and M. Zöllner, editors, *Towards Service Robots for Everyday Environments*. Springer. In press.

[PW10a] Julia Peltason and Britta Wrede. Modeling human-robot interaction based on generic interaction patterns. In *AAAI Fall Symposium: Dialog with Robots*. AAAI, 2010.

[PW10b] Julia Peltason and Britta Wrede. Pamini: A framework for assembling mixed-initiative human-robot interaction from generic interaction patterns. In *SIGDIAL Conference*, pages 229–232, 2010.

[PW11] Julia Peltason and Britta Wrede. The curious robot as a case-study for comparing dialog systems. *AI Magazine*, 32(4):85–99, 2011.

[PWSS11] Karola Pitsch, Sebastian Wrede, Jens-Christian Seele, and Luise Süssenbach. Attitude of german museum visitors towards an interactive art guide robot. In *Proceedings of the 6th international conference on Human-robot interaction*, HRI '11, pages 227–228, New York, NY, USA, 2011. ACM.

[RDO11] Pierre Rouanet, Fabien Danieau, and Pierre Y. Oudeyer. A robotic game to evaluate interfaces used to show and teach visual objects to a robot in real world condition. In *6th ACM/IEEE International Conference on Human-robot interaction*, HRI '11, 2011.

[RE07] Antoine Raux and Maxine Eskenazi. A multi-layer architecture for semi-synchronous event-driven dialogue management. In *IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 514–519, 2007.

[RE09] Antoine Raux and Maxine Eskenazi. A finite-state turn-taking model for spoken dialog systems. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 629–637, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[RHSR07] Frank Röthling, Robert Haschke, Jochen J. Steil, and Helge J. Ritter. Platform portable anthropomorphic grasping with the bielefeld 20-dof shadow and 9-dof tum hand. In *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, pages 2951–2956, San Diego, California, USA, Oct 2007. IEEE, IEEE.

[RJ96] Don Roberts and Ralph Johnson. Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks. In *Proceedings of the Third Conference on Pattern Languages and Programming*, volume 3, 1996.

[RLB+05] Antoine Raux, Brian Langner, Dan Bohus, Alan W. Black, and Maxine Eskenazi. Let's go public! taking a spoken dialog system to the real world. In *INTERSPEECH 2005*, 2005.

[RLGR02] Charles Rich, Neal Lesh, Andrew Garland, and Jeff Rickel. A plug-in architecture for generating collaborative agent responses. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, AAMAS '02, pages 782–789, New York, NY, USA, 2002. ACM.

[Ros10] Robert Ross. Putting things in context: Situated language understanding for human-robot dialog(ue). In *AAAI Fall Symposium: Dialog with Robots*. AAAI, 2010.

[RRT10] Susan Robinson, Antonio Roque, and David Traum. Dialogues in Context: An Objective User-Oriented Evaluation Approach for Virtual Human Dialogue. In Nicoletta C. Chair, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh conference on International Language Resources* *and Evaluation (LREC'10)*, Valletta, Malta, May 2010. European Language Resources Association (ELRA).

[RS98] Charles Rich and Candace L. Sidner. COLLAGEN: A Collaboration Manager for Software Interface Agents. *User Modeling and User-Adapted Interaction*, 8:315–350, 1998.

[RS05] Norbert Reithinger and Daniel Sonntag. An integration framework for a mobile multimodal dialogue system accessing the semantic web. In *INTERSPEECH*, pages 841–844, 2005.

[RS10] Charles Rich and Candace Sidner. Collaborative discourse, engagement and always-on relational agents. In *AAAI Fall Symposium: Dialog with Robots*, 2010.

[RSL01] Charles Rich, Candace L. Sidner, and Neal Lesh. COLLAGEN: Applying collaborative discourse theory to human-computer interaction. *AI Magazine, Special Issue on Intelligent User Interfaces*, 22(4):15–25, 2001.

[SC09] Maria Staudte and Matthew W. Crocker. Visual attention in spoken human-robot interaction. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, HRI '09, pages 77–84, New York, NY, USA, 2009. ACM.

[SCd+98] Stephen Sutton, Ronald Cole, Jacques de Villiers, Johan Schalkwyk, Pieter Vermeulen, Mike Macon, Yonghong Yan, Ed Kaiser, Rrian Rundle, Khaldoun Shobaki, Paul Hosom, Alex Kain, Johan Wouters, Domonic Massaro, and Michael Cohen. Universal speech tools: the CSLU toolkit. In *International Conference on Spoken Language Processing*, pages 3221–3224, 1998.

[Sea69] John R. Searle. *Speech acts : an essay in the philosophy of language*. Cambridge University Press, London, 1969.

[SEHR10] Jan Steffen, Christof Elbrechter, Rorbert Haschke, and Helge J. Ritter. Bio-inspired motion strategies for a bimanual manipulation task. In *International Conference on Humanoid Robots (Humanoids)*, 2010.

[SFK+06] Aaron Steinfeld, Terrence Fong, David Kaber, Michael Lewis, Jean Scholtz, Alan Schultz, and Michael Goodrich. Common metrics for human-robot interaction. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, HRI '06, pages 33–40, New York, NY, USA, 2006. ACM.

[SHL+98] Stephanie Seneff, Ed Hurley, Raymond Lau, Christine Pao, Philipp Schmid, and Victor Zue. Galaxy-ii: A reference architecture for conversational system development. In *International Conference on Spoken Language Processing*, INTERSPEECH, pages 931–934, 1998.

[Shn86] Ben Shneiderman. *Designing the user interface: strategies for effective human-computer interaction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

[SIKN10] Komei Sugiura, Naoto Iwahashi, Hisashi Kawai, and Satoshi Nakamura. Active learning for generating motion and utterances in object manipulation dialogue tasks. In *AAAI Fall Symposium: Dialog with Robots*. AAAI, 2010.

[SKG+09] Satoru Satake, Takayuki Kanda, Dylan F. Glas, Michita Imai, Hiroshi Ishiguro, and Norihiro Hagita. How to approach humans?: strategies for social robots to initiate interaction. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, HRI '09, pages 109–116, New York, NY, USA, 2009. ACM.

[SKLL04] Candace L. Sidner, Cory D. Kidd, Christopher Lee, and Neal Lesh. Where to look: a study of human-robot engagement. In *Proceedings of the 9th international conference on Intelligent user interfaces*, IUI '04, pages 78–84, New York, NY, USA, 2004. ACM.

[SLK+05] Candace L. Sidner, Christopher Lee, Cory D. Kidd, Neal Lesh, and Charles Rich. Explorations in engagement for humans and robots. *Artifical Intelligence*, 166(1-2):140–164, 2005.

[SMHS07] Svetlana Stenchikova, Basia Mucha, Sarah Hoffman, and Amanda Stent. RavenCalendar: a multimodal dialog system for managing a personal calendar. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, NAACL-Demonstrations '07, pages 15–16, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[SNC+96] Stephen Sutton, David G. Novick, Ronald Cole, Pieter Vermeulen, Jacques de Villiers, Johan Schalkwyk, and Mark Fanty. Building 10,000 spoken dialogue systems. In *International Conference on Spoken Language Processing*, volume 2, pages 709–712, 1996.

[Spi01] Dimitris Spiliotopoulos. Human-robot interaction based on spoken natural language dialogue. In *European Workshop on Service and Humanoid Robots*, 2001.

[SS73] Emanuel A. Schegloff and Harvey Sacks. Opening up closings. *Semiotica*, 8(4):289–327, 1973.

[SS09] David Schlangen and Gabriel Skantze. A general, abstract model of incremental dialogue processing. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 710–718, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[SW02] Stephanie Seneff and Chao Wang. Modelling Phonological Rules through Linguistic Hierarchies. In *Pronunciation Modeling and Lexicon Adaptation for Spoken Language Technology (PMLA)*, pages 71–76, 2002.

[SWK08] Frank E. Schneider, Dennis Wildermuth, and Andreas Kräußling. Discussion of exemplary metrics for multi-robot systems for formation navigation. *International Journal of Advanced Robotic Systems*, 2008.

[SWP92] Elizabeth Shriberg, Elizabeth Wade, and Patti Price. Human-machine problem solving using spoken language systems (SLS): factors affecting performance and user satisfaction. In *Proceedings of the workshop on Speech and Natural Language*, HLT '91, pages 49–54, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[SWWB11] Julie Shah, James Wiken, Brian Williams, and Cynthia Breazeal. Improved human-robot team performance using chaski, a human-inspired plan execution system. In *Proceedings of the 6th international conference on Human-robot interaction*, HRI '11, pages 29–36, New York, NY, USA, 2011. ACM.

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

[TBC+02b] Christian Theobalt, Johan Bos, Tim Chapman, Arturo Espinosa-Romero, Mark Fraser, Gillian Hayes, Ewan Klein, Tetsushi Oka, and Richard Reeve. Talking to Godot: dialogue with a mobile robot. volume 2, pages 1338–1343 vol.2, 2002.

[TFBD00] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence*, 128(1-2):99–141, 2000.

[TL03] David Traum and Staffan Larsson. *The Information State Approach to Dialogue Management*, pages 325–353. Kluwer Academic Publishers, 2003.

[Top08] Elin A. Topp. *Human-Robot Interaction and Mapping with a Service Robot: Human Augmented Mapping.* PhD thesis, 2008.

[Tra94] David R. Traum. *A Computational Theory of Grounding in Natural Language Conversation.* PhD thesis, Rochester, NY, USA, 1994.

[Tri00] The TRAINS project: Natural spoken dialogue and interactive planning, 2000. `http://www.cs.rochester.edu/research/cisd/projects/trains/` Accessed 20-Feb-2012.

[WBK99] Marilyn Anne Walker, Julie Boland, and Candace Kamm. The utility of elapsed time as a usability metric for spoken dialogue systems. In *ASRU99*, 1999.

[WDW$^+$06] Michael L. Walters, Kerstin Dautenhahn, Sarah N. Woods, Kheng Lee Koay, I. René J. A. te Boekhorst, and David Lee. Exploratory studies on social spaces between humans and a mechanical-looking robot. *Connection Science*, 18(4):429–439, 2006.

[WDWK07] Michael L. Walters, Kerstin Dautenhahn, Sarah N. Woods, and Kheng Lee Koay. Robotic etiquette: results from user studies involving a fetch and carry task. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, HRI '07, pages 317–324, New York, NY, USA, 2007. ACM.

[WFOB05] Michael White, Mary E. Foster, Jon Oberl, and Ash Brown. Using facial feedback to enhance turn-taking in a multimodal dialogue system. In *HCI International*, 2005.

[WHBS04] S. Wrede, M. Hanheide, C. Bauckhage, and G. Sagerer. An active memory as a model for information fusion. In *7th International Conference on Information Fusion*, pages 198–205, 2004.

[Win86] Terry Winograd. A language/action perspective on the design of cooperative work. In *Conference on Computer-Supported Cooperative Work CSCW '86*, pages 203–220, New York, NY, USA, 1986. ACM Press.

[WKL00] Marylin A. Walker, Candace Kamm, and Diane J. Litman. Towards developing general models of usability with PARADISE, 2000.

[WLKA97] Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. PARADISE: a framework for evaluating spoken dialogue agents. In *8th Conference of the European Chapter of the Association for Computational Linguistics*, 1997.

[WLKA98] Marylin A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. Evaluating spoken dialogue agents with PARADISE: Two case studies. *Computer Speech & Language*, pages 317–347, October 1998.

[WSZL11] Sven Wachsmuth, Frederic Siepmann, Leon Ziegler, and Florian Lier. Tobi - team of bielefeld: The human-robot interaction system for robocup@home 2010. Technical report, Bielefeld University, 2011.

[WW11] Johannes Wienke and Sebastian Wrede. A middleware for collaborative research in experimental robotics. In *2011 IEEE/SICE International Symposium on System Integration, SII2011*. IEEE, IEEE, 2011.

List of Figures

List of Tables

A XML Schema Definition for the Pattern Configuration Language

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:element name="patternConfiguration">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:group ref="dialogActs" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:group name="dialogActs">
        <xsd:sequence>
            <xsd:element ref="humanDialogAct" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="robotDialogAct" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:group>

    <xsd:element name="humanDialogAct">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="param" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="state" type="xsd:string" use="required"/>
            <xsd:attribute name="xpath" type="xsd:string"/>
            <xsd:attribute name="type" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="param">
        <xsd:complexType>
            <xsd:attribute name="type" type="xsd:string" fixed="String" use="required"/>
            <xsd:attribute name="name" type="xsd:string" use="required"/>
            <xsd:attribute name="xpath" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="robotDialogAct">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="output" maxOccurs="1"/>
                <xsd:element ref="rephrasing" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="state" type="xsd:string" use="required"/>
            <xsd:attribute name="type" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="output">
        <xsd:complexType>
            <xsd:all>
                <xsd:element ref="verbalization" minOccurs="0"/>
                <xsd:element ref="point" minOccurs="0"/>
                <xsd:element ref="mimic" minOccurs="0"/>
                <xsd:element ref="vince" minOccurs="0"/>
```

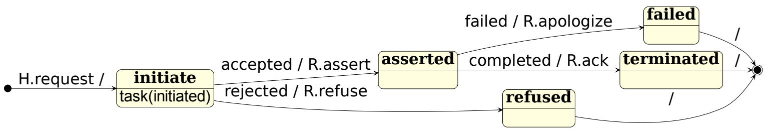
```
                <xsd:element ref="naoBehavior" minOccurs="0"/>
                <xsd:element ref="motionSpeechSync" minOccurs="0"/>
            </xsd:all>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="rephrasing">
        <xsd:complexType>
            <xsd:all>
                <xsd:element ref="verbalization" minOccurs="0"/>
                <xsd:element ref="point" minOccurs="0"/>
                <xsd:element ref="mimic" minOccurs="0"/>
                <xsd:element ref="vince" minOccurs="0"/>
                <xsd:element ref="naoBehavior" minOccurs="0"/>
                <xsd:element ref="motionSpeechSync" minOccurs="0"/>
            </xsd:all>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="verbalization">
        <xsd:complexType>
            <xsd:attribute name="text" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="point">
        <xsd:complexType>
            <xsd:attribute name="coordinatesXpath" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="mimic">
        <xsd:complexType>
            <xsd:attribute name="name" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="vince">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="act" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

   <xsd:element name="act">
      <xsd:complexType>
         <xsd:attribute name="text" type="xsd:string" use="required"/>
         <xsd:attribute name="function" type="xsd:string" use="required"/>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="naoBehavior">
      <xsd:complexType>
         <xsd:attribute name="behavior" type="xsd:string" use="required"/>
         </xsd:complexType>
   </xsd:element>

   <xsd:element name="motionSpeechSync">
      <xsd:complexType>
              <xsd:attribute name="text" type="xsd:string" use="required"/>
         </xsd:complexType>
      </xsd:element>
</xsd:schema>
```

B The Pattern Library

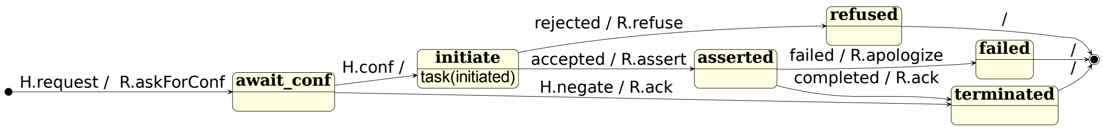
Shown below are the graphical representations of all Interaction Patterns, as provided in PaMini's developer API. They have been generated automatically from the XML statechart definition associated with each Interaction Pattern, using the open source graph visualization software Graphviz.

Action Patterns

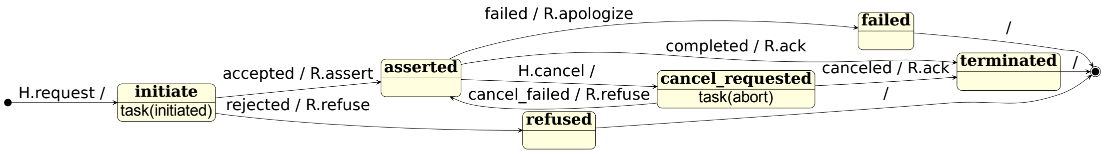
Human Simple Action Request



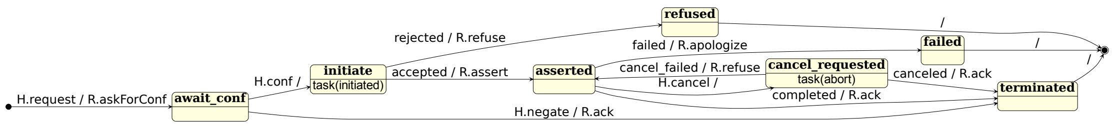
Human Simple Action Request with Explicit Confirmation



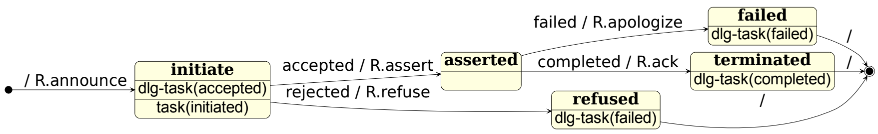
Human Cancellable Action Request



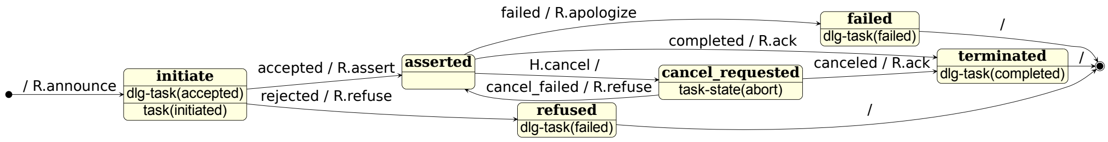
Human Cancellable Action Request with Explicit Confirmation



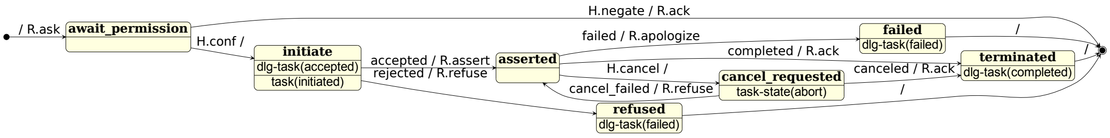
Robot Self-Initiated Simple Action



Robot Self-Initiated Cancellable Action

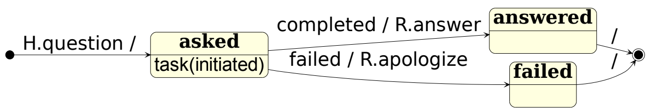


Robot Self-Initiated Cancellable Action with Explicit Confirmation

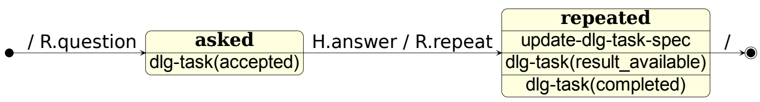


Information Patterns

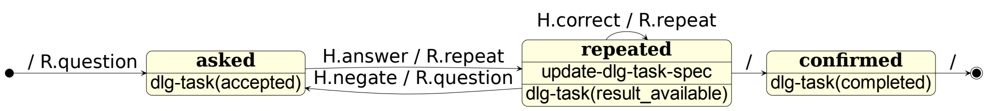
Human Information Request



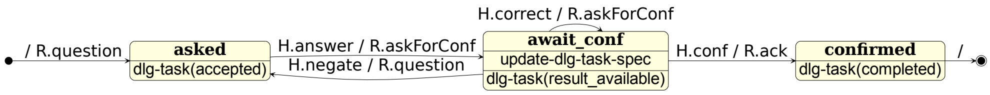
Robot Simple Information Request



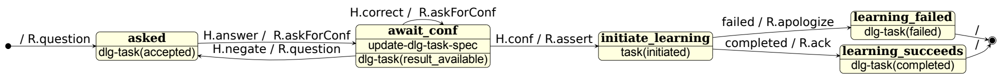
Robot Correctable Information Request



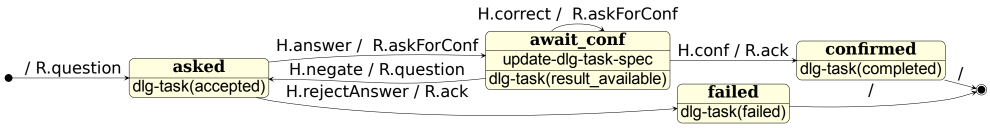
Robot Information Request with Explicit Confirmation



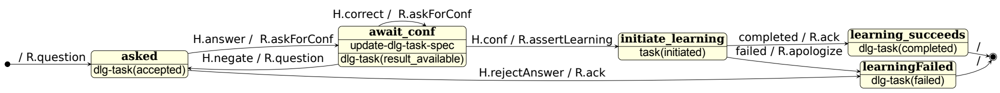
Robot Information Request with Explicit Confirmation and Task Acknowledgement



Robot Rejectable Information Request with Explicit Confirmation

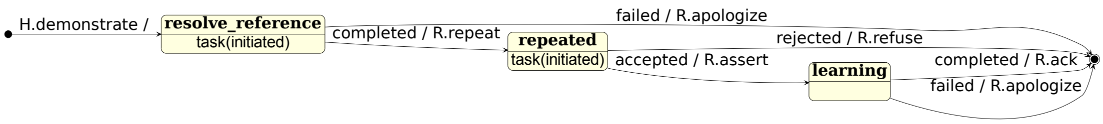


Robot Rejectable Information Request with Explicit Confirmation and Task Acknowledgement

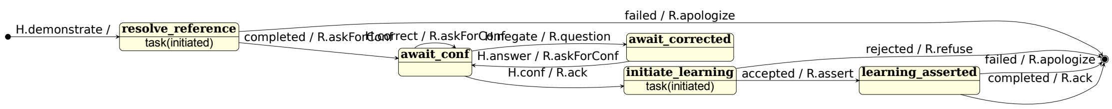


Object Patterns

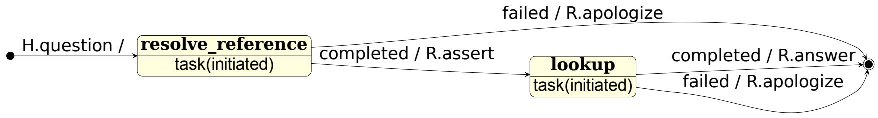
Human Object Demonstration



Human Object Demonstration with Explicit Confirmation

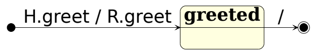


Human Object Test

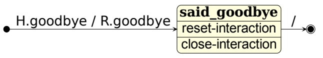


Interactional Patterns

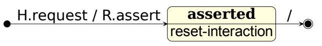
Human Interaction Opening



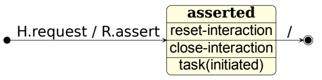
Human Interaction Closing



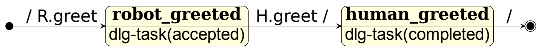
Human Interaction Reset



Human System Reset

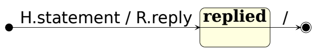


Robot Interaction Opening

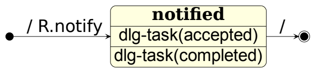


General Patterns

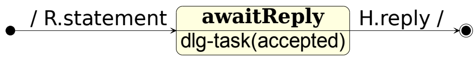
Human Simple Statement



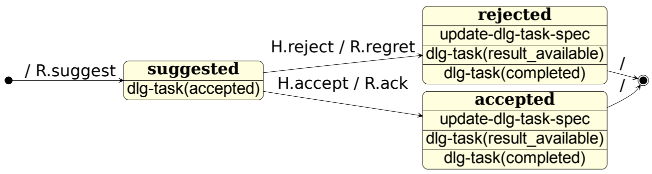
Robot Notification



Robot Simple Statement

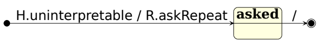


Robot Suggestion

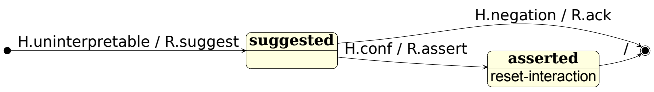


Clarification Patterns

Robot Ask Repeat



Robot Suggest Interaction Reset



C Programming Tasks for the Usability Test

Task 1: Greeting[1]

In our scenario, the human begins interaction by greeting the robot. The robot greets back.

Example interaction

H Hello, robot.

R Hello, human.

We assume a speech understanding component that, whenever a greeting occurs, provides data matching the XPath "/utterance/semanticInfo/frame[@val='Greeting']".

Task 2: Parting

In addition, the human should be capable to end the interaction.

Example interaction

H Bye, bye.

R Bye, see you later.

We assume a speech understanding component that, whenever a greeting occurs, provides data matching the XPath "/utterance/semanticInfo/frame[@val='Greeting']".

Task 3: Navigation Instruction

Our robot is a mobile robot that is able to follow the human on request. The human can cancel the follow task at any time.

Example interaction

H Follow me!

R OK, I follow you.

H Stop!

R OK, I stop.

In the class AsuXPathCollection, you find a collection of XPath expressions that are useful for checking conditions of speech understanding data.

The navigation module is registered on tasks with the following task specification: <FOLLOW/>.

1 The tasks were originally given in German. They were translated into English for the purpose of this thesis.

The Interaction Patterns you will use for this task consists of quite a lot of transitions. There is a configuration template available that may facilitate writing the configuration document.

Task 4: Low Battery Warning

If the battery level is critical, the robot needs to take initiative in order to notify the human about it, and to state the current battery level.

We assume a power management component that requests a dialog task with the following task specification: <power level='X'>.

Hint: Data provided by the power management control match the XPath expression "//power".

Example interaction

The task <power level='10'> should generate the following robot notification:

R Battery charge is low, only 10% remaining.

Task 5: Acquiring Person Name

Finally, the robot is supposed to ask humans unknown to him for their name.

Example interaction

R Hi, what's your name?

H My name is Peter.

R Nice to meet you.

For this purpose, there is a person management component. If the interaction partner is unknown to the robot, it initiates a dialog task with the following task specification: "<unknown_person name=">".

The task specification should be augmented with the human's name, e.g. "<unknown_person name='Peter'>".

The speech understanding component provides for an introduction data that match the XPath expression "frame[@val='Name_personal']".

An XPath that extracts the name from the data is "//frame[@val='Name_personal']/child::name".

D Sentences for Evaluating the Concept Accuracy

1. Hallo Biron, ich bin Carlo und ich werde dir jetzt mal ein paar Sachen zeigen.
   (Hello Biron, I am Carlo and I'm going to show you a couple of things now.)
2. Hallo Biron, ich bin Debora. Ich bring dir jetzt neue Wörter bei.
   (Hello Biron, I am Debora. I am going to teach you new words now.)
3. Hallo Biron, ich bin Anke. ich wollt' dir mal heute ein paar schicke Sachen zeigen.
   (Hello, Biron, I am Anke, as I said before. I wanted to show you some great things today.)
4. Hier ist der Heinz aus Bielefeld.
   (This is Heinz from Bielefeld.)
5. Ja, ich bin hier um dir ein paar Sachen beizubringen und ich werde dir die Sachen jetzt einfach nacheinander zeigen und dir die Namen der Objekte beibringen.
   (Yeah, I am here to teach you some things, and I am just going to show you the things, one after another, and teach you their names.)
6. Ja, freut mich Biron, wie gesagt, ich bin Arne und werde dir gleich ein paar Objekte erklären.
   (Yeah, I am pleased, as I said before, I am Arne and I am going to explain you some objects now.)
7. Hallo, mein Name ist Daniela. Wie geht es dir?
   (Hello, my name is Daniela. How are you?)
8. Wir werden heute zusammen arbeiten. (We are going to work together today.)
9. Hallo Biron, na hörst du mich? Wer da?
   (Hello Biron, well can you hear me? Someone there?)
10. Ich möchte ihnen einige Haushaltsgegenstände zeigen, der Reihe nach.
    (I would like to show you some household items, one after another.)
11. Ich werde dir jetzt ein paar Dinge vorstellen, die du, wie ich weiß, noch nicht kennst.
    (I am going to present you some things that you, as I know, don't know yet.)
12. Möchtest du wissen, was hier vor mir auf diesem Tisch liegt?
    (Would you like to know, what is on the table in front of me?)
13. Du hast hier viel auf dem Tisch.
    (You have lots of things on the table here.)
14. Biron, das ist eine Zeichenhilfe, ein Lineal.
    (Biron, this is a drawing aid, a ruler.)
15. Biron, das was du hier siehst, ist ein Buch.
    (Biron, what you see here is a book.)
16. Hier habe ich einen Schokoriegel.
    (I have a choccolate bar here.)
17. Das ist ein Datenträger, man nennt sie CD.
    (This is a data medium, it is called CD.)
18. Das ist eigentlich nicht interessant, aber wir können das mal verallgemeinen, eine Süßigkeit.
    (That's not really interesting, but we could generalize it, a sweet.)
19. Das hier ist ein Schokoriegel.
    (This here is a choccolate bar.)

20. Dieses Objekt ist ein Kugelschreiber.
(This object is a ballpoint pen)
21. Nein, keine Kugel, sondern ein Kugelschreiber.
(No, not a ball, but a ballpoint pen.)
22. Das ist ein Ball, damit kann man Spielen.
(This is a ball, you can play with it.)
23. Ich wiederhole nochmal, das ist eine CD.
(I repeat, this is a CD.)
24. Das ist keine Tasche, sondern eine Flasche.
(This is not a bag, but a bottle.)
25. Eine Flasche.
(A bottle.)
26. Möchtest du noch mehr Artikel kennenlernen?
(Would you like to get to know more items?)
27. Ok, machen wir einfach mal weiter.
(Ok, let's just go on.)
28. Das ist eine Schere.
(No, these are scissors. Scissors.)
29. Toll, hast du dir gut gemerkt.
(Great, well memorized.)
30. Nein, das ist ein Lineal.
(No, this is a ruler.)
31. Nein, das ist eine Tasse zum Trinken.
(No, this is a cup, for drinking.)
32. Schallplatte, ja. man sagt auch CD dazu.
(Record, yes. It is also called CD.)
33. Guck mal, und wenn du zum Basteln mal gerade Linien brauchst oder gerade Ausschneiden möchtest, dann brauchst du ein Lineal.
(Look here, and if you need straight lines for handcrafting or want to cut straight, then you need a ruler.)
34. Dies ist ein Lineal, ziemlich flach und gerade.
(This is a ruler, quite flat and straight.)
35. Das ist ein Stift. Das einfache Wort ist Stift und das komplizierte Wort ist Kugelschreiber. Kannst du dir das merken?
(This is a pen. The simple word is pen and the complicated word is ballpoint pen. Can you remember that?)
36. Das ist eine Flasche, Flasche.
(This is a bottle, bottle.)
37. Und was ich besonders gerne mag, das sind Bücher. Ein Buch besteht aus vielen Seiten.
(And what I particularly like are books. A book consists of many pages.)
38. Nein, ein Buch.
(No, a book.)
39. So nun hab' ich dir alles gezeigt, Biron, möchtest du noch etwas sehen? Nochmal den Ball?
(Well, I have shown you everything, Biron, you want to see more? The ball again?)
40. Du hast das wiedererkannt, das finde ich gut.
(You have recognized this, I like that.)
41. Dieses Objekt, was du hier in meiner Hand siehst, nennt sich Lineal.

(The object you see here in my hand is called ruler.)
42. Sehr schön, da haben wir doch schon eine ganze Menge gelernt.
(Very nice, we have learned quite a lot already.)
43. Das ist ein Buch. B-U-C-H, Buch.
(This is a book. B-O-O-K, book.)
44. Nein, das ist eine CD-ROM.
(No, this is a CD-ROM.)
45. Eine CD.
(A CD.)
46. Das ist ein Buch, da kann man drin lesen.
(This is a book, you can read in it.)
47. Guck noch mal rein, lieber computer, das ist ein Buch, was ich hier in der Hand hab, zum Lesen.
(Have one more look, dear computer, this is a book, what I have in the hand, for reading.)
48. Ein Lineal oder ein Zeichnstab.
(A ruler or a drawing aid.)
49. Nicht Flasche, sondern Tasse.
(Not bottle, but cup.)
50. Und das ist ein Stück Schokolade von Mars. Ein Stück Mars-Schokolade.
(And this is a piece of choccolate of Mars. A piece of Mars choccolate.)
51. Eine Flasche, wo man Wasser draus trinken kann.
(A bottle, where you can drink water out of.)
52. Nein, das ist eine Schere. Eine Schere.
(No, these are scissors. Scissors.)
53. Hatten wir den Ball schon? Hatten wir den Ball? Das ist ein Ball.
(Did we have the ball already? Did we have the ball? This is a ball.)
54. Das ist, hier, das ist ein Schokoriegel. Hier, hier unten.
(This is, here, this is a choccolate bar. Here, down here.)
55. Ich hab's dir schon gezeigt.
(I've shown it to you already.)
56. Und dieses Objekt?
(And this object?)
57. Genau, gut gemerkt.
(Right, well remembered.)
58. Das ist ein Buch. Ein schweres Buch. Was ist das?
(This is a book. A heavy book. What is that?)
59. Das ist gut. Wie heißt das Objekt?
(This is good. What is the object called?)
60. Was ist das?
(What is that?)
61. Nein, Biron, was ist das hier?
(No, Biron, what is this here?)
62. Richtig.
(Correct.)
63. Und das? Was ist das?
(And this? What is this?)
64. Ja, Biron, ich möchte mich jetzt verabschieden. Auf Wiedersehen.

(Well, Biron, I would like to say goodbye now. Goodbye!)

65. Ja, Biron, also eigentlich haben wir jetzt schon alle Objekte einmal durch, und wären damit fertig.
(Well, Biron, actually we have had all objects already, and would thus be done.)
66. Biron, ich bin jetzt mal weg. Schönen Tag noch, tschüss.
(Biron, I'm gone. Have a nice day, bye.)
67. Ich gehe jetzt nach Hause, Biron, tschüss.
(I am going home now, Biron, bye.)
68. Ok Biron, wir haben genug geübt für's Erste, und ich bedanke mich, dass wir zusammen arbeiten konnten. Auf Wiedersehen, Biron.
(Ok, Biron, we have practiced enough for now, thank you for working together. Goodbye, Biron.)
69. Bis demnächst.
(See you soon.)
70. Ja, dann möchte ich mich herzlich von dir verabschieden, und weiter, alles Gute.
(Well, I would like to say good bye warmly, and all the best in the future.)
71. Ich muss jetzt weiter, tschüss.
(I have to go now, bye.)
72. Wir sind jetzt fertig und ich würde jetzt tschüss sagen.
(We are done now and I would say goodbye now.)
73. Ciao Biron, ich hoffe du hast was gelernt.
(Ciao Biron, I hope you have learned something.)
74. Ja Biron, hat mich gefreut dich kennen zu lernen, auf Wiedersehen.
(Well Biron, I was pleased to meet you, goodbye.)

E User Instruction for the Curious Flobi User Study

General information[1]

Dear participant, welcome to our study with the robot Flobi!

In the following, you will receive relevant information about the study. If you still have questions, please don't hesitate to address the experimenter.

The robot Flobi

The robot Flobi is a prototype we use to investigate human-robot interaction. It is equipped with a microphone and cameras in its eyes, i.e. it can see you and hear you. It can understand speech input in German, and react to it. Flobi will be tested so that it can be improved in future. In this study, Flobi is supposed to learn the labels of objects.

Procedure

The study will proceed as follows:

1. First you will be asked to fill part 1 of the questionnaire.
2. Subsequently, the interaction with Flobi will follow.
3. Finally, you will fill part 2 and part 3 of the questionnaire.

Your task

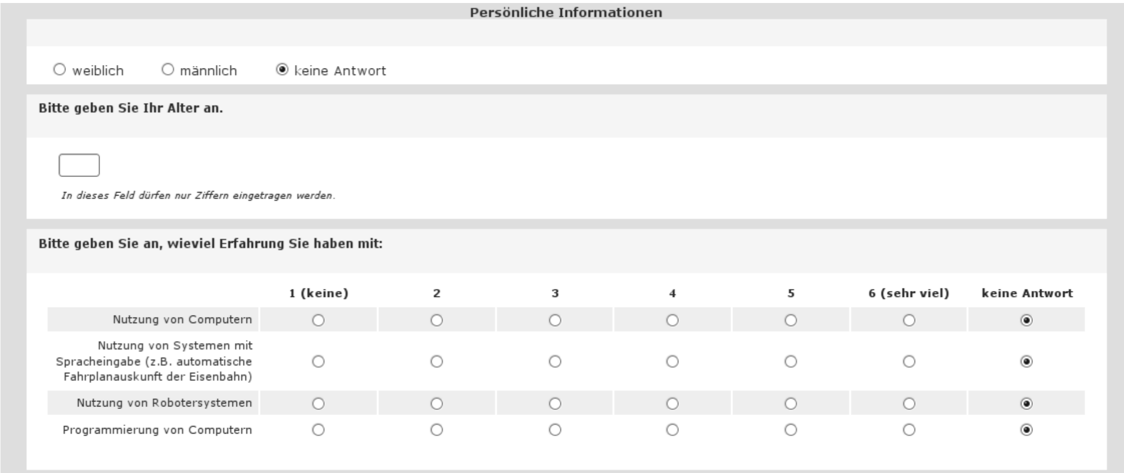
You will now have the opportunity to engage in interaction with Flobi. This will take about 5-10 minutes. Here are some hints that might be helpful:

- You can begin interaction by greeting Flobi, and end it by saying goodbye.
- Flobi is supposed to learn the labels of objects within the interaction. For this purpose, you may use the objects you will find on the table.
- Flobi recognizes objects best when they are lying on the table.
- Please make sure that Flobi actually has learned the objects.
- Speech input is often not recognized correctly. In these cases, it might help to repeat the input more clearly, or to rephrase it. In difficult situations, the instruction "restart" may be helpful.

1 The instructions were originally given in German. They were translated into English for the purpose of this thesis.

F User Questionnaire for the Curious Flobi User Study

The questionnaire has been created using LimeSurvey, an open source survey application. Participants were asked to fill the questions at a desktop computer. The personal information and part 1 of the questionnaire had to be filled out before the interaction with the robot, part 2 after it.
The questionnaire shown below is the one used for the conditions C2 (MixedIni) and C3 (StructuredIni), where the robot asks for objects on its own initiative. For condition C1 (UserIni), which enables user initiative only, the questions on the robot initiative has been omitted ("Wenn der Roboter nach einem Objeckt gefragt hat, war mir immer klar, welches gemeint war.").
The questions of part 1, targeting the user's expectation towards the robot *before* interaction, have not been considered in the evaluation yet. However, we plan to relate them to the user's impression of the robot *after* interaction in a future evaluation.


Persönliche Informationen

weiblich   männlich   keine Antwort

Bitte geben Sie Ihr Alter an.

In dieses Feld dürfen nur Ziffern eingetragen werden.

Bitte geben Sie an, wieviel Erfahrung Sie haben mit:

	1 (keine)	2	3	4	5	6 (sehr viel)	keine Antwort
Nutzung von Computern							
Nutzung von Systemen mit Spracheingabe (z.B. automatische Fahrplanauskunft der Eisenbahn)							
Nutzung von Robotersystemen							
Programmierung von Computern							

Teil 1

Bitte schätzen Sie auf der folgenden Skala Ihre Erwartungen an den Roboter ein.

	1	2	3	4	5	6		keine Antwort
unsympathisch	○	○	○	○	○	○	sympathisch	◉
unfreundlich	○	○	○	○	○	○	freundlich	◉
unhöflich	○	○	○	○	○	○	höflich	◉
unangenehm	○	○	○	○	○	○	angenehm	◉
gemein	○	○	○	○	○	○	lieb	◉
unkooperativ	○	○	○	○	○	○	kooperativ	◉
nicht hilfreich	○	○	○	○	○	○	hilfreich	◉
inkompetent	○	○	○	○	○	○	kompetent	◉
unwissend	○	○	○	○	○	○	kenntnisreich	◉
schlampig	○	○	○	○	○	○	sorgfältig	◉
dumm	○	○	○	○	○	○	intelligent	◉
irrational	○	○	○	○	○	○	rational	◉
träge	○	○	○	○	○	○	lebhaft	◉
mechanisch	○	○	○	○	○	○	organisch	◉
künstlich	○	○	○	○	○	○	natürlich	◉
untätig	○	○	○	○	○	○	interaktiv	◉
teilnahmslos	○	○	○	○	○	○	zugänglich	◉
passiv	○	○	○	○	○	○	aktiv	◉
unvorhersagbar	○	○	○	○	○	○	vorhersagbar	◉
ruhig	○	○	○	○	○	○	gesprächig	◉
desinteressiert	○	○	○	○	○	○	interessiert	◉
langweilig	○	○	○	○	○	○	abwechslungsreich	◉
unaufmerksam	○	○	○	○	○	○	aufmerksam	◉
unselbstständig	○	○	○	○	○	○	selbstständig	◉
lernunfähig	○	○	○	○	○	○	lernfähig	◉

Teil 2

Bitte geben Sie an, wie sehr Sie den folgenden Aussagen zustimmen.

	Stimme nicht zu					Stimme zu	keine Antwort
Ich habe das Gefühl, das Experiment lief wie geplant.	○	○	○	○	○	○	◉
Der Roboter funktionierte zuverlässig.	○	○	○	○	○	○	◉
Es hat lange gedauert, bis ich dem Roboter alle Objekte beigebracht hatte.	○	○	○	○	○	○	◉
Wenn der Roboter nach einem Objekt gefragt hat, war mir immer klar, welches gemeint war.	○	○	○	○	○	○	◉
Wenn ich dem Roboter ein Objekt gezeigt habe, war ihm immer klar welches ich meinte.	○	○	○	○	○	○	◉
Das Tempo der Konversation war angemessen.	○	○	○	○	○	○	◉
Meiner Ansicht nach muss der Roboter noch stark verbessert werden.	○	○	○	○	○	○	◉
Während der Interaktion mit dem Roboter war immer höchste Konzentration erforderlich.	○	○	○	○	○	○	◉
Während der Interaktion mit dem Roboter war mir immer klar, was ich tun konnte oder musste.	○	○	○	○	○	○	◉
Während der Interaktion mit dem Roboter war mir von Anfang an klar, was ich tun konnte oder musste.	○	○	○	○	○	○	◉
Ich hatte immer das Gefühl, dass ich die Situation unter Kontrolle hatte.	○	○	○	○	○	○	◉
Während der Interaktion mit dem Roboter fühlte ich mich angespannt.	○	○	○	○	○	○	◉
Wenn eine Eingabe falsch erkannt wurde, war mir klar, was ich tun musste.	○	○	○	○	○	○	◉
Das letzte Objekt war einfacher beizubringen als das erste.	○	○	○	○	○	○	◉
Der Roboter funktionierte auf eine Weise, wie ich es erwartet hätte.	○	○	○	○	○	○	◉
Während der Interaktion mit dem Roboter war mir immer klar, zu welchem Zeitpunkt ich sprechen konnte.	○	○	○	○	○	○	◉
Wenn ich dieses Experiment noch einmal durchführen müsste, würde ich mich anders verhalten.	○	○	○	○	○	○	◉

Teil 3

Die Reaktionen des Roboters waren:

	1	2	3	4	5	6		keine Antwort
verzögert	○	○	○	○	○	○	prompt	◉
inhaltlich angemessen	○	○	○	○	○	○	inhaltlich unpassend	◉
zeitlich angemessen	○	○	○	○	○	○	zeitlich unangemessen	◉

Die Interaktion mit dem Roboter

	1	2	3	4	5	6		keine Antwort
fiel mir schwer	○	○	○	○	○	○	fiel mir leicht	◉
war ineffizient	○	○	○	○	○	○	war effizient	◉
war verwirrend	○	○	○	○	○	○	war selbsterklärend	◉
war frustrierend	○	○	○	○	○	○	war angenehm	◉
war kompliziert	○	○	○	○	○	○	war leicht zu verstehen	◉

Ich habe den Roboter erlebt als

	1	2	3	4	5	6		keine Antwort
unsympathisch	○	○	○	○	○	○	sympathisch	◉
unfreundlich	○	○	○	○	○	○	freundlich	◉
unhöflich	○	○	○	○	○	○	höflich	◉
unangenehm	○	○	○	○	○	○	angenehm	◉
gemein	○	○	○	○	○	○	lieb	◉
unkooperativ	○	○	○	○	○	○	kooperativ	◉
nicht hilfreich	○	○	○	○	○	○	hilfreich	◉
inkompetent	○	○	○	○	○	○	kompetent	◉
unwissend	○	○	○	○	○	○	kenntnisreich	◉
schlampig	○	○	○	○	○	○	sorgfältig	◉
dumm	○	○	○	○	○	○	intelligent	◉
irrational	○	○	○	○	○	○	rational	◉
träge	○	○	○	○	○	○	lebhaft	◉
mechanisch	○	○	○	○	○	○	organisch	◉
künstlich	○	○	○	○	○	○	natürlich	◉
untätig	○	○	○	○	○	○	interaktiv	◉
teilnahmslos	○	○	○	○	○	○	zugänglich	◉
passiv	○	○	○	○	○	○	aktiv	◉
unvorhersagbar	○	○	○	○	○	○	vorhersagbar	◉
ruhig	○	○	○	○	○	○	gesprächig	◉
desinteressiert	○	○	○	○	○	○	interessiert	◉
langweilig	○	○	○	○	○	○	abwechslungsreich	◉
unaufmerksam	○	○	○	○	○	○	aufmerksam	◉
unselbstständig	○	○	○	○	○	○	selbstständig	◉
lernunfähig	○	○	○	○	○	○	lernfähig	◉

Was hat Sie bei der Interaktion mit dem Roboter besonders gestört?

Was hat Ihnen an der Interaktion mit dem Roboter gefallen?

Haben Sie Vorschläge, wie man das System verbessern könnte?