Modeling Human-Robot Interaction Based on Generic Interaction Patterns

Julia Peltason and Britta Wrede
Applied Informatics, Faculty of Technology
Bielefeld University, 33615 Bielefeld, Germany
Email: jpeltaso, bwrede@techfak.uni-bielefeld.de

Abstract

While current techniques for human-robot interaction modeling are typically limited to restrictive command-control style, traditional dialog modeling approaches are not directly applicable to robotics due to the lack of real-world integration. We present an approach that combines insights from dialog modeling with software-engineering demands that arise in robotics system research to provide a generalizable framework that can easily be applied to new scenarios. This goal is achieved by defining interaction patterns that combine abstract task states (such as task accepted or failed) with robot dialog acts (such as assertion or apology). An evaluation of the usability for robotic experts and novices showed that both groups were able to program 3 out of 5 dialog patterns in one hour while showing a steep learning curve. We argue that the proposed approach allows for less restricted and more informative human-robot interactions.

Introduction

The idea of making robots a ubiquitous technology in order to help people with their every day tasks in home or office environments has imposed new challenges on interaction modeling in robotics. Moreover, if robots are to succeed in novel tasks, they must be able to learn from humans by interacting with them. Thus, interaction capabilities are a fundamental requirement for a robot not only for operating it but also for teaching it.

Yet, today's robotic systems often do not have a dedicated dialog system but employ simple keyword-spotting and command-matching techniques (e.g. (Parlitz et al. 2007)). Other systems rely on finite-state based dialog managers (e.g. (Bauer, Wollherr, and Buss 2009)) that script the interaction flow as finite-state automaton where the states are associated with certain system actions and transitions are taken based on the user's input. However, this approach does not scale well to less predictable interaction such as the ideal of human-robot interaction where two autonomous agents collaborate and contribute in a mixed-initiative manner. Besides, the finite-state based approach couples dialog and task management which makes maintenance and reuse difficult. Therefore, interaction typically has to be reimplemented from scratch in different scenarios.

On the other hand, concepts for both flexible and reusable dialog management frameworks have been established within the spoken dialog community over the recent years. Research has been focusing on traditional information seeking domains where the system first collects the required parameters, then presents the desired information to the user (e.g. an accommodation or travel information). Such approaches are not directly transferable to the robotics domain which presents new challenges for dialog system engineering as it requires mixed-initiative, multi-modal, asynchronous, multi-tasking and open-ended interaction in dynamic environments (Lemon et al. 2002).

However, several dialog modeling approaches have been adopted for robotics. Based on the information state approach (Larsson and Traum 2000), collaboration with autonomous devices (such as unmanned helicopters) involving concurrent tasks has been modeled within the WITAS project (Lemon et al. 2002). The Collagen collaboration and conversation model (Rich and Sidner 1998) has been applied within a visitor guidance demo (Sidner et al. 2005), and the Ravenclaw dialog framework (Bohus and Rudnicky 2009) has been used in a multi-robot space exploration scenario (Dias et al. 2006). These approaches have in common that the dialog manager maintains a declarative description of the goal decomposition and thus takes the role of the decision making instance which may make integration into different robotic architectures difficult.

We have introduced a technique for dialog modeling on robots that achieves a clear separation of dialog and task planning by relying on a fine-grained *task state protocol* as interface between the dialog manager and system components (Peltason and Wrede 2010). Furthermore, we have introduced the concept of *interaction patterns* which describe generic dialog strategies that can be tailored with an application-specific configuration. During interaction, the dialog manager combines the patterns in a flexible way, allowing for non-restrictive mixed-initiative interaction.

The interaction patterns have been extracted from a number of different human-robot interaction scenarios as shown in figure 1, such as the Home-Tour scenario where a mobile robot is supposed to acquire information about its environment (Booij et al. 2008), (Peltason et al. 2009), an object manipulation scenario where a humanoid robot learns how to label and grasp objects (Lütkebohle et al. 2009), and a receptionist scenario where a robot offers visitor guidance (Beuter et al. 2008). Given the diversity of these scenarios, we argue that the resulting patterns provide a good coverage of typical situations in human-robot interaction.

In this paper, after recapitulating the main concepts of the dialog manager introduced in (Peltason and Wrede 2010), we report on novel results as to the question of whether the interaction patterns constitute an understandable application programming interface to the dialog manager. We have conducted an experiment where programmers unfamiliar with the dialog manager were able to build a simple interaction scenario consisting of begin and end of interaction and at least one task involving interfacing to other components within one hour. We consider it especially promising that also programmers new to robotics were able to perform tasks that required integration of dialog with the robot system. This result suggests that the interaction patterns together with the task state protocol are appropriate concepts to encapsulate the complexity both of dialog modeling and component integration.

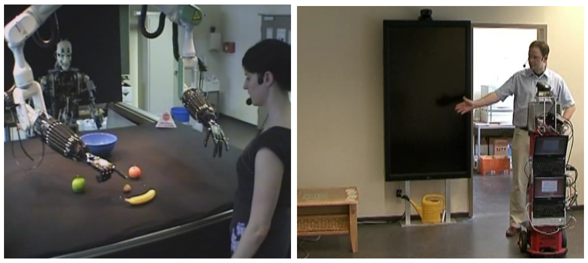

Figure 1: The Curious Robot and the Home-Tour scenario: Two human-robot interaction scenarios that provided the basis for the identification of typical interaction patterns.

	Utterance	Interaction pattern
R:	What is that?	Correctable information request
H:	This is a lemon.	Correctable information request
R:	A melon. OK.	Correctable information request
H:	No, I said 'lemon'!	Correctable information request
R:	A lemon. OK.	Correctable information request
R:	How should I grasp the lemon?	Simple information request
H:	That way.	Simple information request
R:	Alright, with the . power grasp.	Simple information request
R:	I am going to grasp it	Self-initiated action
R:	I start grasping now.	Self-initiated action
H:	What objects do you know?	Human information request
R:	Lemon.	Human information request
H:	How do you grasp the lemon?	Human information request
R:	With the power grasp.	Human information request
R:	OK, I finished grasping.	Self-initiated action

Table 1: A typical interaction with the Curious Robot where it first asks for information about an unknown object, then attempts to grasp it. During action execution, the human asks monitoring questions in order to check the robot's learning success.

A dialog manager for human-robot interaction

In this section, we describe the main concepts of the dialog manager introduced in (Peltason and Wrede 2010) that uses an abstract task protocol for interfacing with the domain level and provides a collection of generic interaction patterns as building blocks for human-robot interaction.

The task state protocol

In distributed robotic systems, various components contribute to the overall systems, for instance components for dialog management including in- and output components as well as components for perceptual analysis, motor control and components generating nonverbal feedback. They need to work together and coordinate.

To realize this, we use the concept of *tasks* that can be performed by components. Tasks are described by an execution state and a task specification containing the information required for execution. A protocol specifies task states relevant for coordination and possible transitions between them as shown in figure 2. Task updates, i.e. updates of the task state and possibly the task specification, cause event notifications which are delivered to the participating components whereupon they take an appropriate action such as creating dialog acts in case of the dialog manager.

In contrast to dialog management approaches that maintain hierarchical task representations such as Collagen's *recipes* (Rich and Sidner 1998), the *activity models* in the WITAS system (Lemon et al. 2002) or Ravenclaw's task specification (Bohus and Rudnicky 2009), our approach leaves task planning to one or more dedicated components such as e.g. an action proposal component in the Curious Robot scenario (Lütkebohle et al. 2009) or an environment representation in the Home-Tour scenario (Peltason et al. 2009). That is, the *global* discourse planning is delegated to external processes while the dialog system is responsible for the *local* discourse planning as specified with the interaction patterns described in the next section. With this separation, bottom-up decision making is facilitated. Furthermore, the dialog manager integrates easier into different types of distributed robotic architectures. The remainder of this section describes how interaction modeling benefits from the task state protocol.

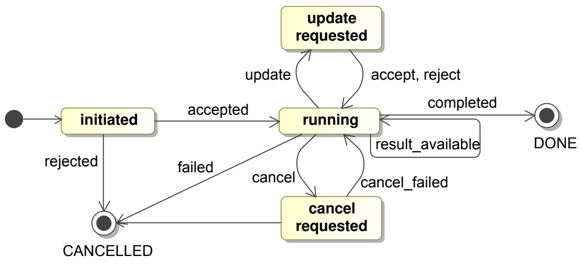

Figure 2: The task life-cycle. A task gets initiated, is running, may be canceled or updated, may deliver intermediate results and finally is completed. Alternatively, it may be rejected by the handling component or execution may fail.

Integration of action execution into interaction A robot performing actions such as manipulation of objects needs to

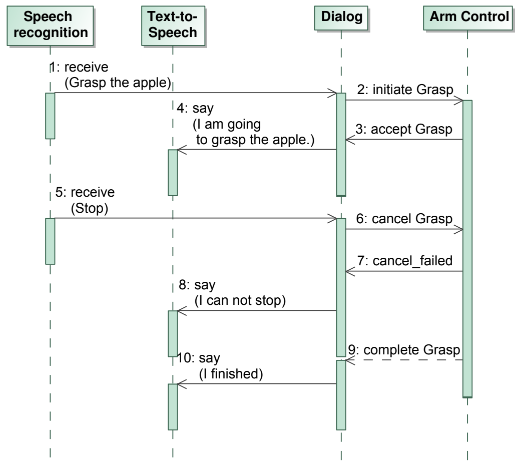

Figure 3: UML sequence diagram illustrating the component communication for a grasp request made by the human. Later, the human attempts to cancel the action which does not succeed.

be able to verbalize its intended actions and give feedback about its internal state, but it also needs to react to comments or corrections uttered by the human instructor. Having the robot e.g. performing a grasp action requested by the human requires communication between the dialog system and the arm control as shown in figure 3. As the dialog manager receives the grasp command, it *initiates* a grasp task which is accepted by the arm control. The dialog is notified about the task state update and acknowledges task execution. As the human commands canceling, the dialog sets the task state *cancel*. The arm control however fails to cancel the task and sets the task state *cancel_failed* which the dialog reacts on by apologizing. Finally, the task is *completed*, and the dialog acknowledges successful task execution. Generally, using this fine-grained protocol for component integration enables the dialog system to provide feedback for the user indicating the internal system state. On the other hand, it supports the modification of the task specification during execution and thus gives the robot the ability to react to comments, corrections and commands on-the-fly.

Mixed-initiative interaction The dialog manager executes dialog tasks for other components, e.g. greeting the human if one is detected by the vision, informing the human about anything or conversely requesting information from the human. While *human initiative* is realized whenever input from the speech understanding component is received, *robot initiative* occurs when a system component requests a dialog task to be executed. Situation permitting, the dialog manager will accept the dialog task, go into interaction with the human, and finally complete the dialog task.

Learning within interaction The task state protocol supports learning within interaction by establishing mechanisms for information transfer from the dialog system to the robotic subsystem. Once information is available from the human, the dialog manager augments the task specification with the new information and sets the task state *result_available*. Since this transition may be taken multiple times, given information can be corrected. Moreover, mixed-initiative enables *active learning*, where the learner provokes a situation providing new information instead of waiting until such situation eventually presents itself. Perceptively, integration of action execution and interaction paves the way for interactive action learning.

Interaction patterns

In interaction, dialog acts are not unrelated individual events, but form coherent segments that are determined by semantics, pragmatics and even more subtle factors such as social conventions or roles.

Accordingly, analyzing our previous human-robot interaction scenarios, we have observed recurring conversational patterns. For instance, the robot's question for an object label proceeds alike the question for the interaction partner's name or the current room: it first asks for the desired information and once the human answered the question, it typically will affirm it in order to give the human the possibility to correct it if necessary. Regularities occur also on domain level. This fact has already been accounted for by basing communication between the dialog manager and the robotic subsystem upon the generic task state protocol.

Influenced by the idea of design patterns in software engineering that offer reusable solutions to commonly occurring problems, we captured the detected commonalities as *interaction pattern*. Interaction patterns describe dialog and domain processing on a high level and have a twofold function. For the dialog manager, interaction patterns constrain the input interpretation by providing the required dialog context and determine the appropriate reaction on the input. For the scenario developer, interaction patterns serve as configurable building blocks of interaction, encapsulating both the subtleties of dialog management and the complexity of component integration.

Interaction patterns can be formalized as transducer augmented with internal state actions, 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;
- a set of 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 $T : S \times \Sigma^* \longrightarrow S \times A^* \times \Lambda^*$.

By admitting task events as input and internal actions that perform task initiation and update, the dialog level is linked with the domain level.

The patterns have been implemented as statecharts (Harel 1987), an extended form of finite state machines, which provides both an executable model and an understandable graphical representation as shown in figure 4.

For instance, the *simple action request* in figure shown 5 pattern describes an action request initiated by the human that, in contrast to the *cancellable action request* used in figure 3, can not be cancelled any more. The normal course of events is that the human requests the action to be executed, the dialog manager initiates the domain task, the responsible system component accepts execution so that the dialog

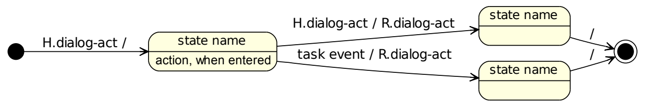

Figure 4: Interaction patterns are represented as transducer which takes as input human dialog acts and task events and produces robot dialog acts as output.

manager will assert execution. Finally, the task is completed and the robot acknowledges. In contrast, the *correctable information request* pattern is initiated by the human. Here, the normal course of event is that on receiving the respective dialog task request, the dialog manager will ask for the desired information and accept the dialog task. Once the human provided the answer, the robot will repeat it as implicit confirmation that can be corrected if necessary. Table 2 lists all patterns that have been identified so far.

Pattern configuration The patterns themselves do not determine what kind of task is to be executed or what kind of information to obtain exactly. These specifics are defined in the configuration associated with each pattern, and a concrete scenario is realized by configuring a set of patterns and registering them with the dialog manager.

In detail, it needs to be specified for the human's dialog acts what kind of (possibly multimodal) input is interpreted as a given dialog act which is done by formulating conditions over the input. For the robot's dialog acts, their surface form needs to be specified. Up to now, speech output and pointing gestures are implemented as output modalities and can be combined. Alternatively, a more sophisticated language generation component could be added in future.

Moreover, 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 for task specification updates. This is how for the robot's information request the answer is transferred from the human to the responsible system component.

Interleaving patterns during interaction During interaction, the registered patterns are employed in a flexible way by admitting patterns to be interrupted by other patterns and possibly resumed later which leads to pattern interleaving as shown in table 1. The default policy for pattern interleaving is to permit simpler patterns to be nested within temporally extended patterns. For instance, it seems reasonable to permit monitoring questions uttered by the human to be embedded in the robot's slow-going grasp execution as shown in table 1. In this way, we equip the robot with multitasking capabilities.

Pattern interleaving is realized by organizing active patterns on a stack. Whenever an input is received, the dialog manager attempts to interpret it in the context provided by the topmost pattern. If it fails, the lower and inactive patterns are tried.

Initiated by user	Initiated by robot
Cancellable action request	Self-initiated cancellable action
Simple action request	Self-initiated simple action
Information request	Correctable information request
Interaction opening	Simple information request
Interaction closing	Clarification
Interaction restart	
System reset	

Table 2: Identified interaction patterns.

Do Interaction Patterns ease dialog modeling?

As outlined above, an evaluation of the overall approach needs to focus on both aspects, the quality of the implemented dialog and the ease of programming new scenarios or new dialog features. In this paper we focus on the software engineering aspect. We have conducted a usability test where programmers were asked to build a human-robot interaction scenario using the proposed dialog manager. Based on performance measurement (Nielsen 1994), we evaluated the usability of the system for scenario developers. Beside, the participants were asked to continuously verbalize their thoughts while using the system which enabled us to identify potential deficiencies and misconceptions and lead to a number of API improvements including more precise method naming, clearer syntax for the configuration language and additional convenience methods.

Experimental Setup Participants were classified either as *robotic expert* or *robotic novice*, 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 taken 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 in implementing interaction opening and end re-

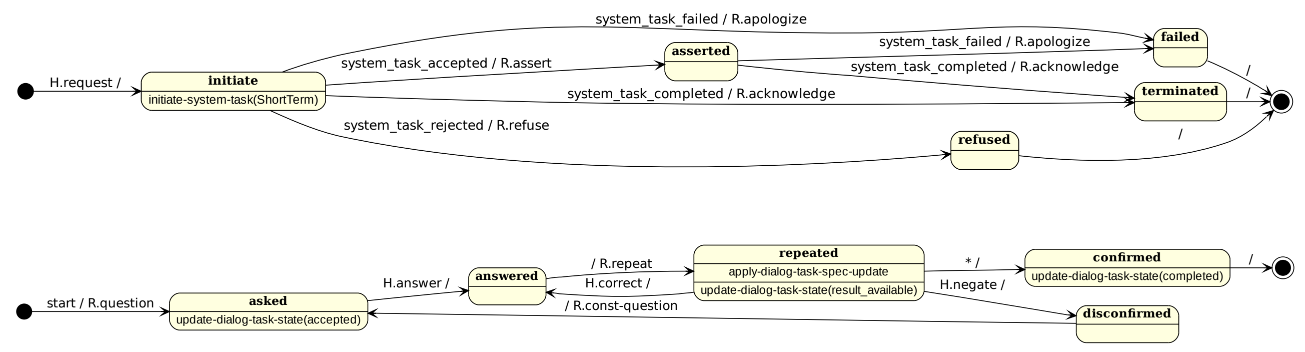
H.request /
initiate
initiate-system-task(ShortTerm)
system_task_failed / R.apologize
system_task_accepted / R.assert
asserted
system_task_failed / R.apologize
failed
system_task_completed / R.acknowledge
system_task_completed / R.acknowledge
terminated
system_task_rejected / R.refuse
refused
/
/
/
start / R.question
asked
update-dialog-task-state(accepted)
H.answer /
answered
/ R.repeat
H.correct /
/ R.const-question
repeated
apply-dialog-task-spec-update
update-dialog-task-state(result_available)
* /
H.negate /
confirmed
update-dialog-task-state(completed)
/
disconfirmed

Figure 5: Two example interaction patterns: *Simple action request*, describing a simple (i.e. non-cancellable) action request initiated by the human and *Correctable information request*, describing a information request with implicit confirmation initiated by the robot where information can be corrected later if necessary.

spectively. Task 3 was to realize a navigation command that might be rejected or fail and be cancelled by the human at any time. The appropriate pattern for this task is the *cancellable action request* which is similar to the *simple action request* shown in figure 5, except that it has a number of additional states and transitions for action cancelling. Task 4 required integration of a power management component that generated notifications whenever the battery level falls below a critical value. The complexity of this task lies in creating a context variable, allocating it with the current charge level and using it to parametrize the robot's warning. Task 5 consisted in having the robot ask persons for their name using the *correctable information request* pattern shown in figure 5. This required augmenting the task specification with the person name in order to realize the information transfer to the responsible system component. Table 3 shows an overview of the given tasks.

Results and Observations 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. Task 3 that exhibited considerable higher degree of difficulty than task 1 and 2 could be solved by seven out of eight participants. Remarkably, all of the robotic novices were able to solve it, even though it required using the *cancellable action request* pattern which 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 5, task 2 could be solved considerably faster than task 1, with 26.75 minutes at 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 effective, taking 9 minutes at average for a simple pattern like *interaction end* and 20.28 minutes for a more complex pattern like *cancellable action request*.

In general, novices 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. We therefore do not expect this result to generalize. It can probably be explained with the robotic expert group exhibiting superior programming skills in general.

Apart from performance measurement, by asking the participants to think aloud, we gained insights into on the scenario 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 similar to the *simple action request* pattern shown in figure 5 but has in addition one state and three transitions for action cancelling. While most of the robotic novices were at first overwhelmed by the complexity, some of the robotic experts became almost enthusiastic. A possible interpretation for that might be that the experts are already aware of the high integration complexity which becomes well describable by the pattern visualization. However, in the end, the novices were able to manage the task even more successful than the experts (table 4), though slightly slower (table 5).

The *cancellable action request* pattern gave us the opportunity for another valuable observation concerning the pattern visualization, having system events as input and robot dialog acts as resulting output. It could be observed that the robotic experts oriented by the system event names, e.g. 'accepted' while the robotic novices oriented by more the dialog act names, e.g. 'R.assert'. We conclude that using this combined notation supports both the robotic system engineer and the interaction designer perspective.

		overall subjects	domain experts	domain novices
Task 1	100%	8	4	4
Task 2	100%	8	4	4
Task 3	75%	8	4	4
	100%	7	3	4
Task 4	50%	4	3	1
	75%	2	1	1
	100%	1	1	0
Task 5	100%	0	0	0

Table 4: Number of subjects that succeeded to solve the respective task up to the given percentage.

Task	Description	Number of dialog acts	Challenge	Interaction pattern
1	Greeting	2		Interaction opening
2	Parting	2		Interaction closing
3	Navigation instruction	11	Task communication	Cancellable action request
4	Low battery warning	6	Task communication, Variable definition, Parametrized output	Notification
5	Acquire person name	1	Task communication, Variable definition, Task specification update	Correctable information request

Table 3: Overview of the tasks given in the usability test.

	all subjects who solved the task 100% (n)	domain experts (n)	domain novices (n)
Task 1	26.75 (8)	25.75 (4)	27.75 (4)
Task 2	9 (8)	8.75 (4)	9.25 (4)
Task 3	20.28 (7)	18.66 (3)	21.5 (3)
Task 4	12 (1)	12 (1)	na (0)
Task 5	na (0)	na (0)	na (0)

Table 5: Average time (in minutes) needed in order to completely solve a task (i.e. 100%).

Conclusion

In this paper, we have presented an approach to human-robot interaction modeling that employs generic interaction patterns to encapsulate the subtleties of dialog management and the complexity of robotic domain integration. We argue that the combined representation of task and dialog structure eases scenario implementation and at the same time enables richer interaction accounting for the real world. Moreover, by combining interaction patterns in a flexible way, less restricted interactions will become possible.

The evaluation focused on the question if this approach enables developers to implement new interaction scenarios in short time. The quantitative results from the usability study showed that both, domain experts and novices were able to complete 3 out of 5 dialog programming tasks of increasing complexity within one hour. Although domain experts were slightly faster than novices, both groups showed a steep learning curve in the second task indicating that the concepts are easy to learn. Qualitative observations support this interpretation: while domain experts tended to rely on the concepts related to the system task protocol, that is the internal processing of the robot, domain novices focused on the dialog acts, that is the surface structure of the dialog. Future studies will investigate how different aspects of the dialog framework, such as interleaving interaction patterns, affect the quality of the dialog design.

References

Bauer, A.; Wollherr, D.; and Buss, M. 2009. Information retrieval system for human-robot communication asking for directions. In *International Conference on Robotics and Automation*.

Beuter, N.; Spexard, T.; Lütkebohle, I.; Peltason, J.; and Kummert, F. 2008. Where is this? - gesture based multimodal interaction with an anthropomorphic robot. In *International Conference on Humanoid Robots*.

Bohus, D., and Rudnicky, A. I. 2009. The ravenclaw dialog management framework: Architecture and systems. *Computer Speech & Language* 23(3):332–361.

Booij, O.; Kröse, B.; Peltason, J.; Spexard, T.; and Hanheide, M. 2008. Moving from augmented to interactive mapping. In *Robotics: Science and Systems Conference*.

Dias, M. B.; Harris, T. K.; Browning, B.; Jones; Argall, B.; Veloso, M.; Stentz, A.; and Rudnicky, A. 2006. Dynamically formed human-robot teams performing coordinated tasks. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone*.

Harel, D. 1987. Statecharts: A visual formalism for complex systems. *Science of Computer Programming* 8:231–274.

Larsson, S., and Traum, D. 2000. Information state and dialogue management in the trindi dialogue move engine toolkit. *Natural Language Engineering* 6:323–340.

Lemon, O.; Gruenstein, A.; Battle, A.; and Peters, S. 2002. Multitasking and collaborative activities in dialogue systems. In *3rd SIGdial meeting on Discourse and Dialogue*, 113–124. Association for Computational Linguistics.

Lütkebohle, I.; Peltason, J.; Schillingmann, L.; Elbrechter, C.; Wrede, B.; Wachsmuth, S.; and Haschke, R. 2009. The curious robot - structuring interactive robot learning. In *International Conference on Robotics and Automation*.

Nielsen, J. 1994. *Usability Engineering*. San Francisco, California: Morgan Kaufmann Publishers.

Parlitz, C.; Baum, W.; Reiser, U.; and Hägele, M. 2007. Intuitive human-machine-interaction and implementation on a household robot companion. In *12th International Conference on Human-Computer Interaction*.

Peltason, J., and Wrede, B. 2010. Pamini: A framework for assembling mixed-initiative human-robot interaction from generic interaction patterns. In *11th SIGdial Meeting on Discourse and Dialogue*.

Peltason, J.; Siepmann, F. H.; Spexard, T. P.; Wrede, B.; Hanheide, M.; and Topp, E. A. 2009. Mixed-initiative in human augmented mapping. In *International Conference on Robotics and Automation*.

Rich, C., and Sidner, C. L. 1998. Collagen: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction* 8:315–350.

Sidner, C. L.; Lee, C.; Kidd, C. D.; Lesh, N.; and Rich, C. 2005. Explorations in engagement for humans and robots. *Artif. Intell.* 166(1-2):140–164.