Separating Representation, Reasoning, and Implementation for Interaction Management: Lessons from Automated Planning

Mary Ellen Foster and Ronald P.A. Petrick

Abstract Numerous toolkits are available for developing speech-based dialogue systems. We survey a range of currently available toolkits, highlighting the different facilities provided by each. Most of these toolkits include not only a method for representing states and actions, but also a mechanism for reasoning about and selecting the actions, often combined with a technical framework designed to simplify the task of creating end-to-end systems. This near-universal tight coupling of representation, reasoning, and implementation in a single toolkit makes it difficult both to compare different approaches to dialogue system design, as well as to analyse the properties of individual techniques. We contrast this situation with the state of the art in a related research area—automated planning—where a set of common representations have been defined and are widely used to enable direct comparison of different reasoning approaches. We argue that adopting a similar separation would greatly benefit the dialogue research community.

Keywords Interaction management · Automated planning · Representation and reasoning · Systems integration

1 Introduction

A fundamental component of any dialogue system is the *interaction manager* [1], whose primary task is to carry out *action selection*: that is, based on the current state of the interaction and of the world, the interaction manager makes a high-level decision as to which spoken, non-verbal, and task-based actions should be taken next by the system as a whole. In contrast to more formal, descriptive accounts of dialogue

M.E. Foster (✉)
School of Computing Science, University of Glasgow, Glasgow, UK
e-mail: MaryEllen.Foster@glasgow.ac.uk

R.P.A. Petrick
Department of Computer Science, Heriot-Watt University, Edinburgh, UK
e-mail: R.Petrick@hw.ac.uk

K. Jokinen and G. Wilcock (eds.), *Dialogues with Social Robots*,
Lecture Notes in Electrical Engineering 427,
DOI 10.1007/978-981-10-2585-3_7

(e.g., [2]), which aim to model the full generality of language use, work on interaction management has concentrated primarily on developing end-to-end systems and on evaluating them through interaction with human users [3, 4].

A number of toolkits are available to support the construction of such end-to-end dialogue systems. Such a toolkit generally incorporates three main features. First, it provides a *representational formalism* for specifying states and actions. Second, the state/action representation is usually tightly linked to a *reasoning strategy* that is used to carry out action selection. Finally, most toolkits also include a set of *infrastructure building tools* designed to support modular system development. While these three features can clearly simplify the task of implementing an individual end-to-end system, the fact that the features are so tightly connected does complicate the task of comparing representational formalisms or reasoning strategies: in general, to carry out such a comparison, there is no alternative but to re-implement the entire system across multiple frameworks [5, 6].

In this chapter, we argue that the dialogue community could benefit from the wider use of system development techniques that break these tight connections among action selection, representation, and technical middleware. As motivation for this view, we use a related research field as as an example: automated planning. At a basic level, the core problem studied in automated planning is also one of context-dependent action selection. However, in the planning community, the focus has been on defining domains in common representation languages and on comparing different action-selection (i.e., planning) strategies within this common context, especially through a series of regularly organised planning competitions [7]. This has had important benefits for the planning community, such as allowing planning engines and domains to be directly compared and shared; also, the study of the representation languages themselves has led to a better understanding of the inherent trade-offs in choosing different representations. We believe that a similar approach could also benefit the dialogue community.

This chapter is structured as follows. We begin with a survey of available interaction management toolkits, summarising the representation, reasoning, and technical facilities provided by each. We then outline certain research directions in the automated planning community, concentrating on how common representations are used and exploited. We also present a recent example of relevant work where an off-the-shelf planner has been used to support interaction management for a socially intelligent robot bartender. Finally, we discuss the potential benefits of developing interaction management strategies that separate representation, reasoning, and integration, and outline our plans for future research in this area.

2 A Survey of Interaction Management Toolkits

In the traditional three-level architecture (Fig. 1) that is typical of a multimodal interactive system [8, 9], the interaction manager sits at the highest level and reasons about the most abstract structures, such as knowledge and action, usually

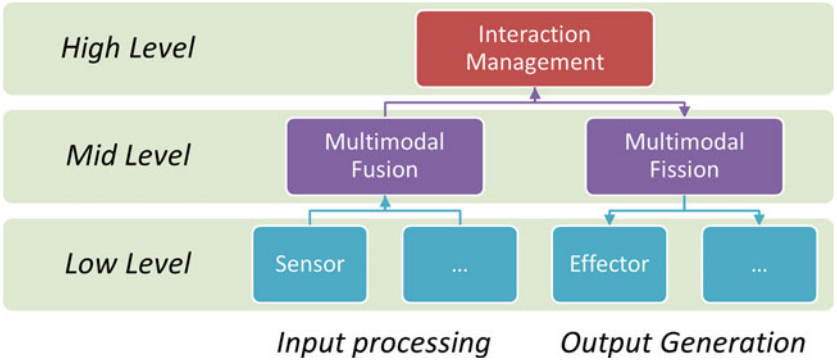

Fig. 1 Typical multimodal system architecture

represented in a logical form, and chooses high-level action specifications for the system to perform. The mid-level components deal with abstract, cross-modality representations of states and events: for input, multimodal fusion [10] combines continuous information from the low-level input sensors into a unified representation, while for output, multimodal fission [11] translates high-level communicative actions into concrete specifications for the individual output components. Finally, the components at the low level deal with modality-specific, highly detailed information: on the input side, this might include coordinates from a vision system or hypotheses from a speech recogniser, while the low-level output components would deal with instructions such as motion specifications for an embodied agent or content for a speech synthesiser.

In this section, we survey a representative set of dialogue systems toolkits, including several well-established, widely-used ones as well as a number of more recently developed toolkits. In particular, we concentrate on the representations used to support high-level interaction management, but also discuss the reasoning mechanisms and relevant details of any accompanying technical architecture.

2.1 TrindiKit/DIPPER

One of the widely used approaches to dialogue management is the Information State Update (ISU) approach, which is exemplified by TrindiKit [12] and its lighter-weight Java reimplementation DIPPER [13]. The core of this approach is the use of an *information state* which represents the state of the dialogue and which is updated by applying *update rules* following a given *update strategy*. The details of an information state are determined by the needs of a particular application. For example, the information state might include external aspects such as variables and their assign-

ments (as in a slot-filling dialogue), or it might include internal agent states such as goals and beliefs (for a more plan-based dialogue strategy). TrindiKit and DIPPER both make use of the Open Agent Architecture (OAA) [14], which provides a middleware for integrating software agents into a distributed system.

A similar ISU approach has also been taken in more recent dialogue systems, but using other infrastructure. For example, the MATCH system [15] uses a similar approach to modelling the information state, while the Flipper toolkit [16] and the dialogue manager for the EMOTE robot tutor [17] both implement ISU-style dialogue management using XML rules.

2.2 Ravenclaw

Another widely-used toolkit is Ravenclaw [18], which is based around a *dialogue task specification* representing the domain-specific aspects of the control logic. This representation forms a hierarchical plan for the interaction and is executed by a domain-independent engine at run time. The specification consists of a tree of *dialogue agents*, each of which handles a sub-task of the dialogue (e.g., greeting the user). The dialogue engine traverses the tree in a depth-first order, putting agents from the tree onto an execution stack and removing them when they are completed. The agents are defined through C++ macros that communicate by exchanging user-defined data structures through a message-passing system.

2.3 COLLAGEN/DISCO

COLLAGEN [19] is a toolkit based on the *collaborative interface paradigm*, which assumes that a software agent is collaborating with a user to operate an application programme, with both agents communicating with each other as well as interacting with the application. COLLAGEN has been used to implement a range of interface agents, including ones for travel booking and for controlling a programmable thermostat. More recently, COLLAGEN has been extended into an open-source tool called DISCO [20], which combines hierarchical task networks (HTNs) with traditional dialogue trees to permit semi-automated dialogue authoring and dialogue structure reuse. The target scenario is specified as a collection of *recipes*—rules for decomposing a goal into subgoals and for accomplishing those subgoals. In contrast to Ravenclaw, where the dialogue flow must be specified, COLLAGEN and DISCO only need a specification of the tasks; the dialogue is then generated automatically via a generic rule framework.

2.4 OpenDial

OpenDial [21] is a domain-independent toolkit for developing spoken dialogue systems. Its primary goal is to support robust dialogue management, using a hybrid framework that combines logical and statistical approaches through probabilistic rules to represent the internal models of the framework. OpenDial also includes a Java-based blackboard architecture where all modules are connected to a central information hub which represents the dialogue state, along with a plugin framework allowing new modules to be integrated.

2.5 IrisTK

IrisTK [22] is a toolkit for the rapid development of real-time systems for face-to-face multi-party interaction which accompanies the Furhat robot head [23]. IrisTK provides an XML-based scripting language for defining statecharts [24] that map input events to output events depending on the system state, along with an event-based distributed architecture that allows a system to be built by integrating modules such as speech recognition and synthesis; it also incorporates pre-built modules for such common tasks.

2.6 InproTK

InproTK [25] is designed for the development of systems which are able to support incremental processing through a network of modules which continuously read input and output the processed result as Incremental Units (IUs). IUs are passed between modules, with the connections between modules specified through a configuration file. InproTK comes with a selection of pre-built modules, and also allows developers to write their own. While the original InproTK used its own internal middleware, a more recent system update [26] adds support for three message-passing protocols (XML-RPC, the Robotics Service Bus [27], and InstantReality [28]), along with a meta-communication layer which mediates among the three protocols.

2.7 Pamini

The Pamini approach [29] was specificially designed to support human-robot interaction. It includes a task-state protocol which abstracts away from the details of processing in the robot system (e.g., perceptual analysis, motor control, or output generation); it also provides a set of generic interaction patterns (such as action

requests and clarification processes) which allow it to support rapid prototyping and combination. Pamini also includes support for mixed-initiative interaction and for online learning during an interaction. Action selection is carried out through interleaving of a set of patterns on a stack: whenever a new piece of input is detected, it is sent to all of the active patterns in turn until one is able to deal with it. Pamini makes use of the XCF middleware [30], which is written in C++ and based on the Internet Communications Engine (Ice) middleware protocol [31].

2.8 Summary

As highlighted throughout this section, and as summarised in Table 1, each of the described toolkits provides a different representation of the information needed for action selection, including declarative update rules, statecharts, interaction patterns, or the more procedural representations used by toolkits such as Ravenclaw and COLLAGEN. Each toolkit also incorporates its own reasoning mechanism to make use of the defined representation—in fact, the representation and reasoning components are often so tightly related that they cannot be fully distinguished. Finally, the majority of the toolkits described either provide or make use of a specific technical middleware framework. As a result, the task of choosing a specific toolkit generally also means adopting both its reasoning strategy and its associated technical infrastructure.

This diversity of toolkit approaches has had a clear impact on the research carried out in the dialogue systems field. In particular, while it is common to compare interaction management strategies within a single framework—for example, by comparing action-selection policies that are learnt from data against hand-coded policies

Table 1 Summary of toolkits considered

Toolkit	Representation	Reasoning	Technical
Trindikit/DIPPER	Information state	Update/selection rules	Open Agent Architecture (C++)
COLLAGEN/DISCO	Recipes	Generic rule framework	Java API
Ravenclaw	Task tree, agenda	Tree traversal	C++ macros, message passing
OpenDial	Probabilistic rules	Event-based state update	Java-based blackboard architecture
IrisTK	XML state charts	Event-based state update	Java event-based distributed architecture
InproTK	Modules, incremental units	Incremental processing	Various middleware options
Pamini	Interaction patterns	Stack of active patterns	XCF middleware (Ice/C++)

[32]—it is relatively uncommon to compare the representational ability and reasoning performance across different frameworks. One of the few studies that did carry out this sort of cross-toolkit comparison is [5], in which the same interactive system was separately implemented using Ravenclaw, DIPPER, Collagen/DISCO, and Pamini. The target domain was the "Curious Robot": a humanoid robot able to engage in an interactive object learning and manipulation scenario. Note that carrying out this comparison required the entire dialogue system to be implemented separately in each formalism; there was no possibility of transferring any representations or reasoning components across the implementations. The overall conclusion of this experiment was that, while all of the toolkits were able to support basic one-on-one spoken interactions, there was more diversity in their approach to and support for more advanced interactive tasks such as multimodal fusion/fission and multi-party interaction. In addition, the different systems were found to offer a range of methods for linking the dialogue state with the task state, which was a feature that was particularly relevant for the Curious Robot scenario. In a more recent study comparing DISCO and Ravenclaw [6], the comparison again required the entire dialogue system to be implemented end-to-end in each individual formalism; in a small-scale user study, few differences were found in the resulting implementations.

3 The Separation of Representation and Reasoning in Automated Planning

The overall problem of selecting high-level actions for an intelligent agent is not unique to dialogue systems, but is a problem addressed in a variety of research communities including automated planning. In planning, the emphasis is on applying problem-solving techniques to find an ordered sequence of actions (called a *plan*) that, when chained together, transform an initial state into a state where a set of specified goal objectives are achieved.[1] The general planning problem is usually divided into two parts: a description of the planning domain and the definition of a planning problem instance to be achieved within that domain [33]. A *planning domain* provides a definition of the symbols and actions used by the planner. Symbols are used to specify the objects, properties, states, and knowledge that make up the planning agent's operating environment, normally defined in a logic-like language. Actions are typically specified in terms of the state properties that must be true to execute that action (the action's *preconditions*) and the changes that the action makes to the state when it is executed (the action's *effects*). A *planning problem* provides a definition of the initial state the planner begins its operation in, and a description of the goals to be achieved. A central goal of planning research is to build *domain-*

[1] This differs somewhat from the task of interaction management, where the goal is (usually) to find the next system action, rather than a complete action sequence. However, we note that a system that is able to achieve the latter can also be used in the former context.

independent planning systems that are able to solve a range of planning problems in a variety of domains, rather than just a single problem in a single domain.

One important feature of research in automated planning is that the tools developed by this community usually support one of a number of common representation languages, such as PDDL [34], PPDDL [35], or RDDL [36], among others. Many of these languages have been developed or extended as part of the International Planning Competitions (IPC) [7, 37]—a series of competitions in which different planning systems compete against each other on a common set of planning problems—which have run approximately every other year since 1998 within the context of the International Conference on Automated Planning and Scheduling (ICAPS).[2] Even when some planners implement their own representation languages, which may differ (usually syntactically) from the standard planning languages, additional work is often performed to establish the relationship between such languages and the more common representations.

These activities have led to some important benefits for the planning community. First, by adopting common representations, the task of modelling a planning problem can be separated from the task of implementing an efficient engine for solving those problems. This allows different planning engines to be developed and directly compared, either quantitatively or qualitatively, on a common set of inputs (i.e., planning problems). Second, planning domains and planning engines can be shared, leading to the development of common benchmarks for future planning systems, as well as an improvement in the baseline systems that can solve problems in these domains. In particular, the IPC has contributed greatly to these activities by creating and requesting new domains, which has in turn helped spur the development of more powerful planning tools. The activities of the IPC have also resulted in a repository of planning domains which can be studied, analysed, and reused as necessary [37], with additional efforts from the community aimed at making planning tools and domains more accessible.[3] Finally, the representation languages themselves—and the planning problems they support—can be studied and compared, leading to a better understanding of the complexity of particular classes of domains and problems [38], and the tradeoffs of using one language over another. This work has close connections to related communities such as knowledge representation and reasoning (KR&R) formal logic. It has also produced some interesting research directions, such as a range of compilation approaches which seek to transform more complex planning problems into simpler forms that are solvable in an efficient manner with existing tools [39, 40].

We believe that similar approaches could be applied within the dialogue systems research community, leading to similar positive results. In the following section, we give a concrete example of this approach, where a domain-independent automated planner is used for the task of interaction management in a scenario involving a socially intelligent humanoid robot.

[2] http://www.icaps-conference.org/.

[3] http://planning.domains/.

4 Plan-Based Interaction Management in a Robot Bartender Domain

The JAMES robot bartender (Fig. 2)[4] has the goal of supporting socially appropriate multi-party interaction in a bartending scenario. Based on (uncertain) observations about the users in the scene provided by the vision and speech recognition components, the system maintains a model of the social context and the task state, and decides on the socially-appropriate responses that are required to respond to human users in that setting.

In this context, high-level action selection is performed by a domain-independent planner which manages the interactions with customers, tracks multiple drink orders, and gathers additional information as needed with follow-up questions [41]. In particular, the task of interacting with human customers is mixed with the physical task of ensuring that the correct drinks are delivered to the correct customers. Plans are generated using PKS (Planning with Knowledge and Sensing) [42, 43], a planner that works with incomplete information and sensing actions. Figure 3 shows an example of two actions from the robot domain, defined in the PKS representation. Here, `ask-drink` models an information-gathering dialogue action that asks a customer for their drink order, while `serve` is a physical robot action for serving a drink to a customer. The complete planning domain description also includes actions such

Fig. 2 The JAMES robot bartender

[4] http://jamesproject.eu/.

```
action ask-drink(?a : agent)
    preconds: K(inTrans = ?a) &
              !K(ordered(?a))
              !K(otherAttnReq) &
              !K(badASR(?a))
    effects:  add(Kf,ordered(?a)),
              add(Kv,request(?a))
```

```
action serve(?a : agent, ?d : drink)
    preconds: K(inTrans = ?a) &
              K(ordered(?a)) &
              Kv(request(?a)) &
              K(request(?a) = ?d) &
              !K(otherAttnReq) &
              !K(badASR(?a))
    effects:  add(Kf,served(?a))
```

Fig. 3 Example PKS actions in the JAMES bartender domain

Table 2 A partial list of actions in the robot bartender planning domain

Action	Description
`greet(?a)`	Greet agent `?a`
`ask-drink(?a)`	Ask agent `?a` for a drink order
`serve(?a,?d)`	Serve drink `?d` to agent `?a`
`bye(?a)`	End an interaction with agent `?a`
`wait(?a)`	Tell agent `?a` to wait
`ack-order(?a)`	Acknowledge agent `?a`'s order
`ack-wait(?a)`	Thank agent `?a` for waiting
`ack-thanks(?a)`	Acknowledge agent `?a`'s thanks
`inform-drinklist(?a,?t)`	Inform agent `?a` of the available drinks of type `?t`

Table 3 A plan for interacting with a single customer in the bartender domain

Plan steps	Description
`greet(a1),`	Greet agent `a1`
`ask-drink(a1),`	Ask agent `a1` for a drink order
`ack-order(a1),`	Acknowledge agent `a1`'s order
`serve(a1,request(a1)),`	Serve `a1` the drink that was ordered
`bye(a1).`	End the interaction with agent `a1`

as `greet(?a)` (a purely social action to greet customer `?a`), `bye(?a)` (end an interaction with `?a`), and `wait(?a)` (tell `?a` to wait, e.g., by nodding), among others. A partial list of the actions in the robot bartender domain is given in Table 2. The planner uses these actions to form plans by chaining together particular action instances to achieve the goals of a planning problem. For instance, Table 3 shows a five step plan for interacting with a single customer in the bartender domain.

An important design decision for the robot bartender was to define the state and action representations separately from the tools used to reason about them, and also from the infrastructure needed for the planner to communicate with the rest of the system (which employs the Ice object middleware [31]). In addition to supporting

the modular, distributed development of the system, this also permitted the PKS planner to be exchanged with a completely separate interaction manager based on Markov Decision Processes [32], with no other changes needed to the system. In terms of our particular planning approach, PKS's representation can be compiled into a variant of standard PDDL, allowing the bartender domain to be tested with other planning systems. Using this approach, we performed a series of offline experiments with other planners to study particular aspects of the planner's performance in the bartender domain, for instance how it scales when the number of agents or the number of subdialogues is increased [44]. The modular design of our approach also means that, if necessary, the planner could be replaced in the robot system with an alternative domain-independent planners, with few changes needed to the underlying domain representation or the high-level software infrastructure. Similarly, our planning approach could be easily integrated into other interactive systems using its existing application programming interface [45].

5 Summary and Future Work

We have examined a number of toolkits designed to support the development of speech-based dialogue systems. While the features provided by these toolkits simplify the prototyping and deployment of an individual end-to-end system, each individual toolkit also tends to be very tightly linked to the representations, reasoning techniques, and even technical infrastructure used to connect the interaction manager to the rest of the system. This makes it difficult either to compare different approaches or to analyse the properties of individual techniques without re-implementing the entire system from the ground up in multiple frameworks.

We argue that the approach taken in the automated planning community—where domains have long been defined in common representation languages, and action-selection strategies compared within this common context—could also benefit the dialogue systems community, by permitting diverse approaches to be benchmarked and compared more easily. Although some early work was carried out in this area [46–50], the approach has for the most part been largely overlooked (with the exception of approaches like [51–55]). Note that common tasks such as the Dialogue State Tracking challenge [56] do exist in the dialogue community; however, to our knowledge, there has never been a successful effort to develop standard, high-level representations for use in interaction management.

There are also opportunities for the automated planning community to benefit from closer collaboration with the dialogue systems community. For instance, problems in dialogue systems can also serve as the basis for new challenge domains for planning, showcasing planning tools and techniques, and possibly extending the standard planning representations with the features needed to model new types of problems. Beyond the opportunity for novel test domains, there are also some important general lessons that the planning community can learn from dialogue systems.

For instance, dialogue systems are inherently application driven and, as such, any adoption of planning techniques must be situated in the context of larger, more complex systems of which planning is a single component. This often requires a degree of maturity in tool development that goes beyond offline lab-tested code, with a focus on robustness and the development of standard application programming interfaces (APIs). While there have been recent attempts to build such systems within the planning community [57], more work is still needed. Finally, the issue of user evaluation is at the heart of dialogue systems work, with a focus on (non-expert) users actually using the developed tools. As a result, dialogue systems domains are often driven by the needs of the real-world application, rather than lab-based assumptions, which could help facilitate the wider adoption of planning approaches in such settings.

More generally, the work described here is situated in the context of a larger research programme aimed at revisiting the use of techniques from automated planning in the context of natural language interaction. We believe that the time is right for the natural language community—and, in particular, the dialogue systems community—to benefit from recent advances in the area of automated planning. We have already demonstrated that components from the two communities can be successfully combined in the JAMES bartender [41]; we plan to continue our work in this area by exploring the challenges and opportunities that arise from the intersection of these two research fields. In particular, we believe that the adoption of common, formally understood representation languages for states and actions that are separated from reasoning mechanisms and technical infrastructures can facilitate closer links between the two research communities.

Acknowledgements This research has been partially funded by the European Union's Seventh Framework Programme for research, technological development and demonstration under grant no. 270435 (JAMES, http://james-project.eu/) and grant no. 610917 (STAMINA, http://stamina-robot.eu/), and by the European Union's Horizon 2020 research and innovation programme under grant no. 688147 (MuMMER, http://mummer-project.eu/).

References

1. Bui, T.H.: Multimodal dialogue management—state of the art. Technical Report 06–01, University of Twente (UT), Enschede, The Netherlands (2006)
2. Asher, N., Lascarides, A.: Logics of Conversation. Cambridge University Press (2003)
3. Jokinen, K., McTear, M.: Spoken Dialogue Systems. Morgan & Claypool (2009)
4. McTear, M., Callejas, Z., Griol, D.: The Conversational Interface. Springer International Publishing (2016)
5. Peltason, J., Wrede, B.: The curious robot as a case-study for comparing dialog systems. AI Mag. **32**(4), 85–99 (2011)
6. Olaso, J.M., Milhorat, P., Himmelsbach, J., Boudy, J., Chollet, G., Schlögl, S., Torres, M.I.: A multi-lingual evaluation of the vAssist spoken dialog system: comparing Disco and RavenClaw. In: Jokinen, K. & Wilcock, G. (Eds.) Dialogues with Social Robots, Springer pp.221–238 (this volume) (2016)
7. Coles, A., Coles, A., García Olaya, A., Jiménez, S., Linares López, C., Sanner, S., Yoon, S.: A survey of the seventh international planning competition. AI Mag. **33**(1), 83–88 (2012)

8. Gat, E.: Three-layered architectures. In: AI-Based Mobile Robots: Case Studies of Successful Robot Systems. MIT Press (1998)
9. Dumas, B., Lalanne, D., Oviatt, S.: Multimodal interfaces: a survey of principles, models and frameworks. In: Human Machine Interaction, Lecture Notes in Computer Science, vol. 5440, pp. 3–26 (2009)
10. Atrey, P.K., Hossain, M.A., El Saddik, A., Kankanhalli, M.S.: Multimodal fusion for multimedia analysis: a survey. Multimedia Syst. **16**(6), 345–379 (2010)
11. Foster, M.E.: State of the art review: multimodal fission. Deliverable 6.1, COMIC project (2002)
12. Larsson, S., Traum, D.R.: Information state and dialogue management in the TRINDI dialogue move engine toolkit. Nat. Lang. Eng. **6**(3&4), 323–340 (2000)
13. Bos, J., Klein, E., Lemon, O., Oka, T.: DIPPER: Description and formalisation of an information-state update dialogue system architecture. In: Proceedings of SIGdial, pp. 115–124 (2003)
14. Martin, D.L., Cheyer, A.J., Moran, D.B.: The open agent architecture: a framework for building distributed software systems. Appl. Artif. Intell. **13**(1–2), 91–128 (1999)
15. Johnston, M., Bangalore, S., Vasireddy, G., Stent, A., Ehlen, P., Walker, M., Whittaker, S., Maloor, P.: MATCH: an architecture for multimodal dialogue systems. In: Proceedings of ACL, pp. 376–383, Philadelphia, Pennsylvania, USA (2002)
16. ter Maat, M., Heylen, D.: Flipper: an information state component for spoken dialogue systems. In: Intelligent Virtual Agents. Lecture Notes in Computer Science, vol. 6895, pp. 470–472. Springer, Berlin (2011)
17. Janarthanam, S., Hastie, H., Deshmukh, A., Aylett, R., Foster, M.E.: A reusable interaction management module: use case for empathic robotic tutoring. In: Proceedings of goDIAL, Gothenburg, Sweden (2015)
18. Bohus, D., Rudnicky, A.I.: The RavenClaw dialog management framework: architecture and systems. Comput. Speech Lang. **23**(3), 332–361 (2009)
19. Rich, C., Sidner, C.L.: COLLAGEN: a collaboration manager for software interface agents. User Model. User-Adap. Inter. **8**(3–4), 315–350 (1998)
20. Rich, C., Sidner, C.L.: Using collaborative discourse theory to partially automate dialogue tree authoring. In: Intelligent Virtual Agents, Lecture Notes in Computer Science, vol. 7502, pp. 327–340 (2012)
21. Lison, P.: A hybrid approach to dialogue management based on probabilistic rules. Comput. Speech Lang. (2015)
22. Skantze, G., Al Moubayed, S.: IrisTK: a statechart-based toolkit for multi-party face-to-face interaction. In: Proceedings of ICMI, pp. 69–76 (2012)
23. Al Moubayed, S., Beskow, J., Skantze, G., Granström, B.: Furhat: a back-projected human-like robot head for multiparty human-machine interaction. In: Cognitive Behavioural Systems, Lecture Notes in Computer Science, vol. 7403, pp. 114–130 (2012)
24. Harel, D.:Statecharts: a visual formalism for complex systems. Sci. Comput. Program. **8**(3), 231–274 (1987). ISSN 0167-6423. http://dx.doi.org/10.1016/0167-6423(87)90035-9
25. Baumann, T., Schlangen, D.: The InproTK 2012 release. In: Proceedings of the NAACL-HLT Workshop on Future directions and needs in the Spoken Dialog Community: Tools and Data, pp. 29–32 (2012). http://projects.ict.usc.edu/nld/SDCTD2012/
26. Kennington, C., Kousidis, S., Schlangen, D.: InproTKs: a toolkit for incremental situated processing. In: Proceedings of SIGdial, pp. 84–88 (2014)
27. Wienke, J., Wrede, S.: A middleware for collaborative research in experimental robotics. In: Proceedings of the 2011 IEEE/SICE International Symposium on System Integration, pp. 1183–1190 (2011)
28. Kousidis, S., Kennington, C., Schlangen, D.: Investigating speaker gaze and pointing behaviour in human-computer interaction with the mint.tools collection. In: Proceedings of SIGDIAL, pp. 319–323, Metz, France (2013)
29. Peltason, J., Wrede, B.: Pamini: a framework for assembling mixed-initiative human-robot interaction from generic interaction patterns. In: Proceedings of SIGdial, pp. 229–232 (2010)

30. Wrede, S., Hanheide, M., Bauckhage, C., Sagerer, G.: An active memory as a model for information fusion. In: Proceedings of the 7th International Conference on Information Fusion, pp. 198–205 (2004)
31. Henning, M.: A new approach to object-oriented middleware. IEEE Internet Comput. **8**(1), 66–75 (2004)
32. Keizer, S., Foster, M.E., Lemon, O., Gaschler, A., Giuliani, M.: Training and evaluation of an MDP model for social multi-user human-robot interaction. In: Proceedings of SIGdial (2013)
33. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory and Practice. Morgan Kaufmann (2004)
34. McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D.: PDDL—The Planning Domain Definition Language (Version 1.2). Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control (1998)
35. Younes, H.L.S., Littman, M.L.: PPDDL1.0: an extension to PDDL for expressing planning domains with probabilistic effects. Technical Report CMU-CS-04-162, Carnegie Mellon University (2004)
36. Sanner, S.: Relational dynamic influence diagram language (RDDL): language description. http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf (2010)
37. ICAPS: ICAPS competitions. http://www.icaps-conference.org/index.php/Main/Competitions (2015)
38. Rintanen, J.: Complexity of planning with partial observability. In: Proceedings of ICAPS, pp. 345–354 (2004)
39. Palacios, H., Geffner, H.: Compiling uncertainty away in conformant planning problems with bounded width. J. Artif. Intell. Res. **35**, 623–675 (2009)
40. Albore, A., Palacios, H., Geffner, H.: A translation-based approach to contingent planning. In: Proceedings of IJCAI, pp. 1623–1628 (2009)
41. Petrick, R.P.A., Foster, M.E.: Planning for social interaction in a robot bartender domain. In: Proceedings of ICAPS 2013 (2013)
42. Petrick, R.P.A., Bacchus, F.: A knowledge-based approach to planning with incomplete information and sensing. In: Proceedings of AIPS, pp. 212–221 (2002)
43. Petrick, R.P.A., Bacchus, F.: Extending the knowledge-based approach to planning with incomplete information and sensing. In: Proceedings of ICAPS, pp. 2–11 (2004)
44. Sharma, V.: Automated Planning for Natural Language Robot Dialogue. M.Sc. Project, University of Edinburgh, Edinburgh (2012)
45. Petrick, R.P.A., Gaschler, A.: Extending knowledge-level planning with sensing for robot task planning. In: Proceedings of PlanSIG (2014)
46. Perrault, C.R., Allen, J.F.: A plan-based analysis of indirect speech acts. Am. J. Comput. Linguist. **6**(3–4), 167–182 (1980)
47. Appelt, D.: Planning English Sentences. Cambridge University Press, Cambridge (1985)
48. Hovy, E.: Generating Natural Language Under Pragmatic Constraints. Lawrence Erlbaum Associates, Hillsdale (1988)
49. Cohen, P., Levesque, H.: Rational interaction as the basis for communication. In: Cohen, P., Morgan, J., Pollack, M. (eds.) Intentions in Communication, pp. 221–255. MIT Press, Cambridge (1990)
50. Young, R.M., Moore, J.D.: DPOCL: a principled approach to discourse planning. In: Proceedings of INLG, pp. 13–20, Kennebunkport, Maine, USA (1994)
51. Koller, A., Stone, M.: Sentence generation as planning. In: Proceedings of ACL, pp. 336–343, Prague, Czech Republic (2007)
52. Benotti, L.: Accommodation through tacit sensing. In: Proceedings of LONDIAL, pp. 75–82, London, UK (2008)
53. Brenner, M., Kruijff-Korbayová, I.: A continual multiagent planning approach to situated dialogue. In: Proceedings of LONDIAL, pp. 67–74 (2008)
54. Koller, A., Petrick, R.P.A.: Experiences with planning for natural language generation. Comput. Intell. **27**(1), 23–40 (2011)

55. Mackaness, W., Boye, J., Clark, S., Fredriksson, M., Geffner, H., Lemon, O., Minnock, M., Webber, B.: The SpaceBook project: pedestrian exploration of the city using dialogue based interaction over smartphones. In: Proceedings of the 8th Symposium on Location-Based Services, Vienna, Austria (2011)
56. Henderson, M., Thomson, B., Williams, J.D.: The second dialog state tracking challenge. In: Proceedings of SIGdial, pp. 263–272, Philadelphia, PA, USA (2014)
57. Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carrera, A., Palomeras, N., Hurtos, N., Carreras, M.: ROSPlan: planning in the robot operating system. In: Proceedings of ICAPS (2015)