A MULTI-AGENT FRAMEWORK FOR A HYBRID DIALOG MANAGEMENT SYSTEM

Stefan Schwärzler, Joachim Schenk, Günther Ruske and Frank Wallhoff

Institute for Human-Machine Communication
Technische Universität München, Germany
{sts,joa,rus,waf}@mmk.ei.tum.de

ABSTRACT

The importance of dialog management systems has increased in recent years. Dialog systems are created for domain specific applications, so that a high demand for a flexible dialog system framework arises. There are two basic approaches for dialog management systems: a rule-based approach and a statistic approach. In this paper, we combine both methods and form a hybrid dialog management system in a scalable agent based framework. For deciding of the next dialog step, two independent systems are used: the JAVA Rule Engine (JESS) as expert system for rule-based solutions, and the Partially Observable Markov Decision Process (POMDP) as model-based solution for more complex dialog sequences. Using a speech recognizer and text-to-speech systems, the human can be guided through a dialog with approximately ten steps.

Index Terms— dialog management, hybrid framework

1. INTRODUCTION

For a spoken dialog system, one can distinguish between six task areas. Depending on the system, these areas are more or less developed, and often their transitions are fluent [1]. The speech recognizer recognizes spoken phonemes and returns a sequence of words according to a lexicon. The sentence analysis (parsing) assigns a meaning to this sequence and exports the ordered, relevant information in any system language [2]. The dialog management determines the dialog strategy and thus, how the system responds to the user's input. During the communication with external sources, the information is written to or read from databases. The generation of responses is virtually the opposite of the sentence analysis and translates words from the system language into a sequence of words that the user understands. An audio output, e. g. realized by a text-to-speech system, converts this sequence of words via a speech synthesis. All components are depicted in Fig. 1.

To ensure flexibility, it is important to define an accurate interface between the components. Besides the input/output modules, which e. g. embed programs [3, 4], the flexibility in the dialog management is also of importance. In general, we distinguish between two different approaches: rule-based and model-based systems. In addition to the rules, facts are defined in rule-based systems [5]. These systems are especially suitable for simple dialog structures and rapid prototyping of a dialog management system. TRINDIKIT [6] is a toolkit, which includes information states and update rules. In addition, the dialog transitions can be defined and tested. It is based on the open agent architecture standard [7], but has limitations; there is no directory facilitator and all agents have to run on the same machine.

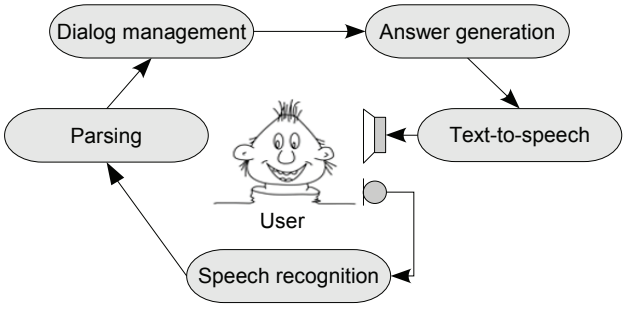

Fig. 1. Components of a speech dialog system.

Furthermore, depending on the user scenario, a dialog management system performs better, with a rule-based expert system or a model-based statistical approach. Model-based approaches use information from the semantic parser [2], the dialog history, and from a domain-specific dialog corpus. In [8] a mathematical model to describe dialog strategies, the Partially Observable Markov Decision Process (POMDP), is detailed.

This paper presents an agent-based system architecture, whose agents communicate with external programs via both a server and an interface, which is able to change the dialog strategy in a rule- or model-based manner during its runtime. The rule-based method, which is implemented with JESS [5], and the POMDP-model [8] can be processed in this framework.

The rest of this paper is organized as follows: After this introduction, which presented agent based frameworks in the dialog management domain, in the next section, the required I/O agents are listed, and the dialog management methods are described with specific examples. A descriptive application system architecture describing the modules is listed in Sec. 3. After the implementation of the architecture, the treatise closes in Sec. 4 with a summary and additional modalities to be added in future work.

2. AGENT BASED FRAMEWORK

Programs, which are capable of a certain autonomous behavior, are known as agents [9]. Their behavior is such that for every possible consequence of perception they maximize their

own success. Their activities are based on the perceptions and the existing knowledge. In our case, the perceptions are the semantic slots from the speech recognizer, which are interpreted as described in [2]. According to [10], agents have the following relevant characteristics:

autonomous to certain extend, agents may keep their own control over their own action, their thread, and their decision-making.

proactive agents not only react to events, they can even take a targeted initiative, if it is appropriate.

social agents communicate with each other; they can share their tasks and results.

With the JAVA Agent Development Framework (JADE) (see [10]), we use a toolkit that supports peer-to-peer applications. Besides the graphical user interface, JADE provides further debug tools, e. g. SNIFFER (see Fig. 2), INTROSPECTION AGENT (see Fig. 3) and is based on the OPEN AGENT ARCHITECTURE [7].

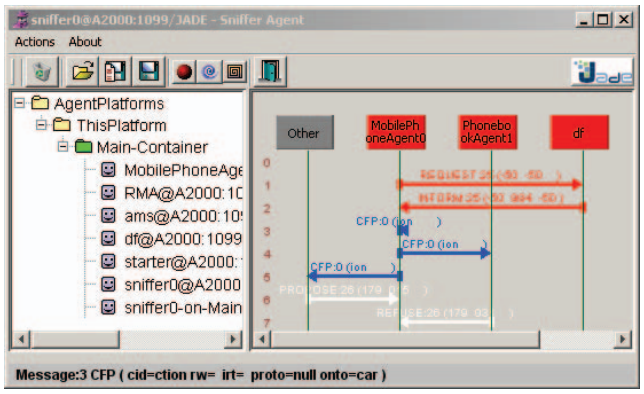

Fig. 2. The JADE Sniffer is a graphical tool to analyze the agent communications.

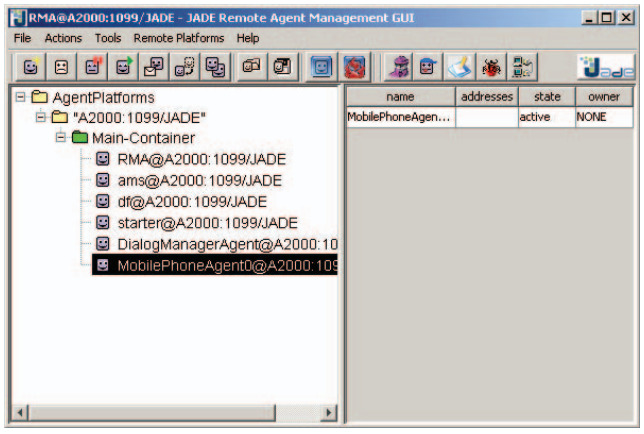

Fig. 3. The JADE Introspection agent shows the agent status and the queue of messages of the agents.

2.1. Expert Systems and Rule-Based Agents

In [5] a toolkit is used, which is based on a forward-chaining algorithm and includes a knowledge-based system whose rules are optimized using the RETE algorithm [11]. The required facts come from the semantic decoding [2] of a speech recognizer integrated into an input agent. Based on these facts, the system can draw a conclusion based on rules after a triggering event.

Algorithm 1 JESS [5] rules for air travelling information system.

```
(defrule  path-finder
(path  ?X ?Y)
(path  ?Y ?Z)
⇒ (assert  path  ?X ?Z))
(defrule  global rules
(available-ticket-to ?X
(?A ← (ordered-ticket-to  ?X)
⇒ (printout t "Searching of a flight to")
(assert (given  ?X))
(retract  ?A)
(printout  t Booking of a flight to ?X
(defrule  global rules2
(not available-ticket-to ?X)
(?A ← (ordered-ticket-to  ?X)
⇒ (printout t "Searching of a flight to")
(assert (given  ?X))
(retract  ?A)
(printout t "Searching flight to" ?X "is not available!")
```

In Alg. 1, a flight route from X to Y is calculated recursively (pathfinder). Then, a flight route to X is searched for, and if possible, it is reserved. This example of booking a flight contains rule-based processes and is particularly suitable for expert systems (with database connection) and rule-based agents.

2.2. Model-Based Agents

In [8], a model-based approach to dialog management is described. The POMDP is a Markov Decision Process (MDP), which will be extended in such a way that the states are not observable. This is done in a similar way as for the Hidden Markov Model (HMM), where the states of the Markov chain are hidden and an observation is linked to the states with certain probabilities. In contrast to the MDP, the POMDP does not know the current state with certainty. Hence, for each state its probability is inferred based on the current observation. In contrast to the HMM, these are Markov decision-making processes, where actions can be executed and change their environment accordingly. The problem of a dialog management system means that the system itself can perform certain actions, usually asking questions or producing results. POMDPs are suitable to model this behavior. The POMDP consists of a set of States $\mathcal{S}$, a set of observations $\mathcal{O}$, and a set of actions $\mathcal{A}$ (see graphical model in Fig. 4). During a training phase, these parameters are estimated and form a transition matrix $\mathbf{T} = P(s'|a_m, s)$, which gives the probability of the current state s' subject to the condition of the previous state s and the executed action a_m. During the training phase, one determines the reward as a function of the state s and the executed actions a_m in a reward matrix. The observation matrix $\mathbf{Z} = P(o'|s', a_m)$ reflects the observation conditioned on the current state and the previous action. From all three matrices generated during the training phase, $\mathbf{T}, \mathbf{R}, \mathbf{Z}$, a pol-

icy is created. Later during operation, a suitable vector **b** can be constructed from it. In addition to the policy, a geometric discount factor λ is considered, which enhances the weight of the rewards that are closer to future times. This factor is important in agent-based systems, as a possible change of topic and corrections could lead to infinitely long dialogs. For initialization an additional belief state $\mathbf{b}_0$ is defined.

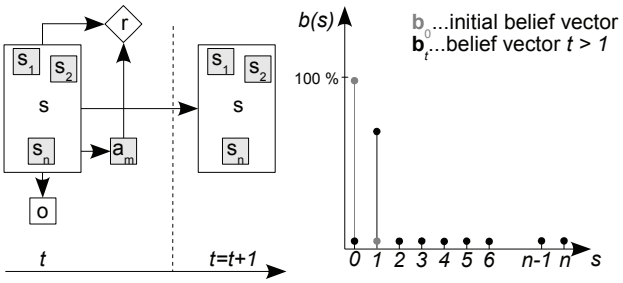

Fig. 4. Using POMDP [8] model for the management of dialog strategies.

As shown in Fig. 4, during the operation phase the appropriate action a_m is chosen according to the current state belief $\mathbf{b}_t$, and the state will be changed to the next state s'. There, an observation of semantic slots o takes place, and the belief state $\mathbf{b}_t$ is updated following Eq. 1, and where k is a normalization factor.

$$b'(s') = k\, P(o'|s', a_m) \sum_{s \in S} P(s'|a_m, s)\, b(s). \qquad (1)$$

Model-based agents are particularly suitable for more complex dialog structures with a corresponding large training set from which the parameters of the model are estimated. In this paper, we focus on the use of the POMDP model in the dialog framework. The necessary training of this model is described in [12].

2.3. Input/Output Agents

Input/output agents are used to communicate with external programs. A speech recognizer [3] or a TTS system [4] can be connected to the agent community via a Transmission Control Protocol (TCP). The speech recognizer and speech-synthesis generator provide the verbal communication with the user for the whole agent system. The grammars for the corresponding semantic decoding is adapted by other agents and sent to the speech recognizer. In addition, other modalities e. g. hand gestures recognition, head movements (shaking or nodding) can be incorporated into the agent framework via the input and output devices. Similarly, the external MATLAB scripts [13] are embedded in this agent class.

3. SYSTEM ARCHITECTURE

The system is designed such that many components can be reused. In the **generic agent**, all general agent tasks are realized, such as initialization and registration with the agent community. Therefore, all agents have to register their services in the yellow pages (see Fig. 5). Furthermore, for initialization, the JESS module loads knowledge bases, rules, and facts. The JESS rule engine can communicate via the **JESS agent** with the agent community. A subclass of the generic agent is the **server agent**, where the TCP socket connection is built. More specific socket classes are the TTS server agent and the speech recognizer server agent. The TTS (MARY, see [4]) connects via the **TTS client** to the TTS server agent. Therefore, speech processor methods are implemented in the **TTS server agent**, e. g. speaking text, input types, output types, and audio types. In the same way, the actions (which are received from the speech **recognizer client**) are implemented in the speech **recognizer server agent**, e. g. create an action in the agent community, add a grammar file, or remove a grammar file from the speech recognizer (VOCON [3]). The recognizer server is receiving the semantic slots with confidences from the speech module. The model-based **POMDP agent** receives the semantic slots and proposes actions to the **TTS agent**. The agent includes the external MATLAB based POMDP module.

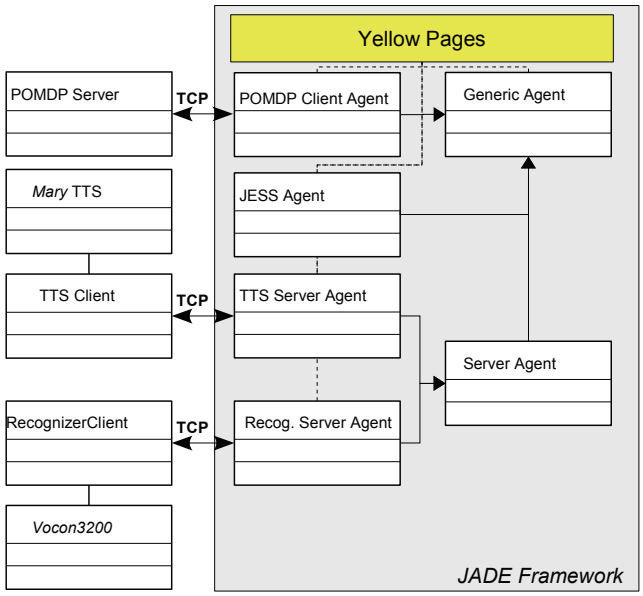

Fig. 5. Using JADE Framework [10] for a hybrid dialog management architecture.

3.1. Communication and Interaction of the Agents

The UML sequence diagram in Fig. 6 shows the principle interaction at the beginning of the agent based framework. At first, all agents have to register their services on the yellow page of the JADE framework. The yellow page acts as an information broker and knows, which agents are available or occupied. Then, the **JESS agent** initializes the JESS rule-based system with a set of rules and facts, which are loading from a file. After receiving semantic slots [2], the **agent speech** is looking for a service in the yellow page. Both reasoning agents (JESS and POMDP) offers their services to the agent speech and receive the semantic information from the speech agent. If the JESS agent cannot infer a query (see Fig. 6), the TTS agent is waiting for the POMDP reasoning. This module infers a query for the next dialog step. Otherwise, the user has to repeat the question.

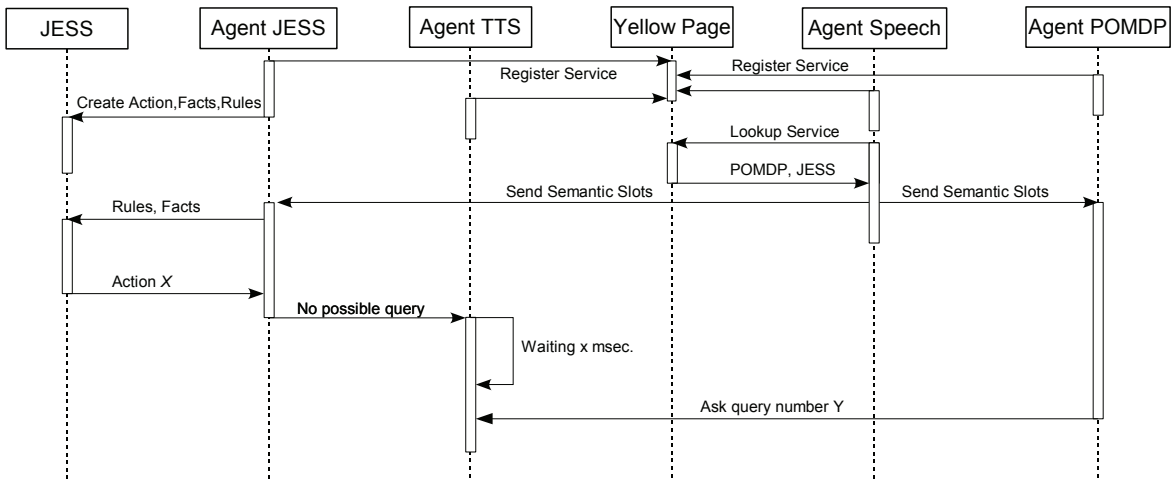

Fig. 6. UML Sequence diagram of the agent-based framework.

3.2. Hybrid Reasoning

JESS is embedded in the JADE framework and implemented in JAVA. The POMDP is realized as an external MATLAB script in the POMDP toolbox [13]. Both dialog management agents (POMDP and JESS) are able to reasoning from the semantic slots, which are received from the recognizer agents. If there is a rule for reasoning, then the JESS agent outperforms the POMDP agent. Otherwise, the system receives a confidence value from the POMDP agent. It is then necessary to repeat the question in the air travelling information system task.

4. CONCLUSION AND OUTLOOK

The current research report describes a hybrid dialog management system for flexible applications, like e. g. in information systems (air travelling) or in cognitive technical systems. Besides the flexible and scalable system architecture, several input/output modules for a spoken dialog system have been integrated. In future work, we will integrate more modalities like hand gestures, head shaking or nodding, and visual feedbacks in our system. The ATIS database [14] has been used for the training of the model-based POMDP agent. In order to evaluate the facilities and possible drawbacks of this framework, a use case with a flight-booking scenario was introduced. The system successfully terminates within approximately ten dialog steps.

5. ACKNOWLEDGMENTS

This work has been funded within the Excellence Cluster COTESYS by the German Research Foundation (DFG).

6. REFERENCES

[1] M. F. McTear, *Spoken Dialogue Technology*, Springer, London, 2004.

[2] S. Schwärzler, J. Geiger, J. Schenk, M. Al-Hames, B. Hörnler, G. Ruske, and G Rigoll, "Combining Statistical And Syntactical Systems For Spoken Language Understanding With Graphical Models," in *Proc. of the 9th International Speech Communication Association (Interspeech 2008)*, 09 2008, pp. 1590–1593.

[3] N. N., "Software development kit version 2.0 developers guide," Tech. Rep., ScanSoft Inc., 2004.

[4] N.N., *International Journal of Speech Technology*, vol. 6, chapter The German Text-to-Speech Synthesis System MARY: A Tool for Research, Development and Teaching, pp. 1381–2416, Springer Netherlands, Amsterdam, Netherlands, 2003.

[5] E. Friedman-Hill, *Jess in Action Java Rule-based Systems*, 2003, ISBN: 1930110898.

[6] S. Larsson, A. Bernman, J. Hallenborg, and D. Hjelm, "Trindikit manual," 2004.

[7] A. Cheyer and D. Martin, "The Open Agent Architecture," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 4, no. 1, pp. 143–148, March 2001, OAA.

[8] S. Young, "Using POMDPs for dialog management," in *IEEE/ACL Workshop*, Palm Beach, Aruba, 2006.

[9] S. Russel and P. Novig, *Artificial Intelligence*, Prentice Hall, New Jersey, 2003.

[10] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa, "JADE a white paper," *In search of innovation*, vol. 3, no. 3, September 2003.

[11] C. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," in *Artificial Intelligence*, 1982, vol. 19, pp. 17–37.

[12] L. P.Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," in *Artificial Intelligence*, 1998, vol. 101, pp. 99–134.

[13] T. Taha, "POMDP Toolbox v0.1," Tech. Rep., Clemson University, Clemson, USA, 2007.

[14] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington, "The ATIS Spoken Language Systems Pilot Corpus," Website, 1990, Available online at `http://acl.ldc.upenn.edu/H/H90/H90-1021.pdf`; visited on March 31th 2009.