

Users Talk to their Model Trains: Interaction with a Speech-based Multi-Agent System^{*}

Alexander Huber
University of Erlangen-Nuremberg (IMMD 8)
Haberstr. 2
D-91058 Erlangen, Germany
alexander.huber@cs.fau.de

Bernd Ludwig
University of Erlangen-Nuremberg (IMMD 5)
Haberstr. 2
D-91058 Erlangen, Germany
bernd.ludwig@cs.fau.de

ABSTRACT

This paper presents an interactive multi agent system (MAS) for controlling a model train installation. User intentions can be communicated to the system via speech and are processed by hierarchically organized autonomous agents with distributed knowledge and responsibilities in the application domain. Agents on deeper levels in the hierarchy execute, observe, and refine plans constructed by higher level agents. Conflicts between requirements of tasks and the current application state are resolved by local planning and negotiating with the user. For this purpose a multi-modal natural language dialogue system is incorporated as the user interface.

Categories and Subject Descriptors

I 2.11 [Artificial Intelligence]: Distributed Artificial Intelligence; I 2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search—*Plan execution, formation, and generation*; I 2.7 [Artificial Intelligence]: Natural Language Processing—*Discourse, Language generation, Language parsing and understanding, Speech recognition and synthesis*; H 1.2 [Models and Principles]: User/Machine Systems—*Human information processing*

1. INTRODUCTION

We describe a multi-agent system (MAS) for the control of a model train scenario including the simulation of the domain operations executed in an application situation at the same time. The system supports interactivity by allowing speech as an input modality. In this way, we implement a user-friendly strategy to handle complex systems in general. For example, the user may give commands like "Set speed of train 1 to 17!", "Switch point 5!"; even highly complex

^{*}The authors want to thank ROCO Modellspielwaren, Salzburg (Austria) for providing trains, waggons, and tracks and Lenz Elektronik GmbH, Gießen (Germany) for the digital control unit.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'02, July 15-19, 2002, Bologna, Italy.

Copyright 2002 ACM 1-58113-480-0/02/0007 ...\$5.00.

commands as "Move train 1 from station A to station B at 13:34!" are understood. Interaction is not restricted to commands like those above but even queries about the system state are possible. At any time, users are allowed to give commands for domain operations. The agents have to verify whether the intended operations are possible in general and admissible in the current situation to guarantee the failure-safe service of the system. Under the assumption of a closed world, the overall system features hierarchical planning, plan execution, and plan observation by several agents with different responsibilities and capabilities. In this way, the system knowledge is distributed and organized hierarchically corresponding to the tasks each agent has to fulfill. The hierarchy in the system architecture ensures the modularity of the system and a clear separation of micro and macro tasks of the various agents. Eventually, the information about the current system state is incomplete and may be obtained from other agents or by negotiating with the user. Agents communicate in KQML [2]. The content of messages is related to an application ontology formalized with the help of description logics [1].

2. THE MODEL TRAIN CONTROL

As explained in [4], each agent layer in the MAS represents another competence level. On top there is the Central Control Agent (*CCA*) which is the interface between the application and the dialogue system. The main task of the *CCA* is to satisfy the user's goals by computing a sequence of actions. If such a plan can be found, its execution will be delegated to other agents. Otherwise the agent explains the reasons why the goal cannot be achieved. Goals can be unsatisfiable either because the user assumptions are contradictory to the system state or because there is no sequence of actions whose effects entail the goal. In the first case, the *CCA* tries to compute plans to align system state and user assumptions. In the second case, the user must modify his goal. If possible, the system proposes modifications which are satisfiable given the current system state. In this special application the *CCA* maintains a model about all Route Agents (*RAs*) which is represented as a undirected connectivity graph. Each node is a *RA* and the edges symbolize connections between two *RAs*.

On the competence levels below the *CCA* there are exclusively application specific agents. First there are Route Agents (*RAs*). Each *RA* controls all trains and points in a section of the railroad: It computes the route a train will take and switches all points involved. A specialized *RA* is

called Station Agent which keeps a time table for trains.

Train Agents (*TR*) represent the trains in the simulation. Each *TR* executes and observes the plan computed by its supervising *RA*. If the state of the model train installation differs from the expected one the *TR* informs the responsible *RA* for exception handling (see Sect. 4 below for details).

The Simulation Agent (*SA*) simulates all operations and controls the interface to the model train installation and maintains a complete model of the railroad.

3. THE DIALOGUE MANAGER

The management of negotiating between user and *CCA* is handled by a dialogue manager described in detail in [5]. For user utterances, input from a speech recognizer is processed by a parser resulting in a representation for the meaning of the utterance. The dialogue manager has three tasks to master in order to enable negotiation with the *CCA*. On the *discourse level*, the function of user utterances (they can ask questions, give commands, state, repeat, accept, reject facts) with respect to the current state of a negotiation has to be analyzed. *Domain* and *user level* let application pragmatics come into play. Utterances must be integrated in the current negotiation by relating user plans to pending goals and system plans which are not executed completely.

4. PLAN GENERATION AND EXECUTION

As outlined above, the dialogue system analyzes user utterances in order to detect the user's intentions. Based on knowledge about how these intentions are linked to application specific goals, the *CCA* constructs a situation in which such goals should be reached after executing appropriate domain actions starting from the given situation.

4.1 Hierarchical Execution

In order to fulfill the user's intentions, the *CCA* starts the execution of the computed plan. He delegates the actions to the responsible agents down in the hierarchy: For each action to be executed, the *RA* constructs a plan for the *CCA* action and thereby refines the *CCA* plan step by dividing it into several subactions it can execute for the section it controls. In a recursive manner, the *RA* again delegates the execution of the actions it has planned to agents on a lower level in the hierarchy. In this case, the *TR* are addressed which inform the *SA* to update the state of the simulation.

4.2 Resolving Underspecification

During planning and executing domain operations, agents may be faced with incomplete knowledge. The user is free to let the preconditions and goal descriptions for domain operations underspecified as in "Move train 1". Here, at least a destination is missing which is an obligatory argument for the move operation. As the system always tries to be as cooperative as possible, its strategy for handling incomplete information is to seek clarification from the user. In this way, it is ensured that the user intentions are satisfied. So, if obligatory information (from the viewpoint of generating plans in the application domain) is missing, the system starts a clarification dialogue, while for non-obligatory information default values can be assumed, or the missing facts can be derived from the current application situation. In the above example, it can be assumed that the train should depart from its current position. If these assumptions turn

out to be ambiguous or cannot be made at all, the system starts clarification dialogues with the user.

4.3 Resolving Conflicts

SA's update operations are acknowledged by the model train hardware in the case of successful as well as failed execution. The *TR* uses this information to observe the execution of the actions in its local view. In the case of an error, the *TR* tries to construct a new plan that can fulfill the task given by a *RA*. If no plan can be found, the calling *RA* is informed. Now, the *RA* tries to find a new plan for its task or informs the *CCA*. If the *CCA* fails to find a solution for the user's intentions, it has to negotiate with the user. Except from those "bottom up" conflicts while executing plans, the case of "top down" conflicts in the planning phase exists as well: when constructing a plan, the initial situation in the application may be inconsistent with the discourse situation; i.e. the user's assumptions do not match the current state of the application, and therefore the user intentions cannot be fulfilled. Again, the *CCA* starts negotiating with the user about the alternatives to resolve the conflict. The aim of the negotiation is to find out which alternative the user prefers. "Top down" conflicts may appear on the *RA* level as well: Eventually, no plan for the refinement of the *CCA* action can be constructed in the given situation from the *RA*'s point of view. As in the "bottom up" case, now the *CCA* has to find an alternative plan making use of its more global view of the application.

5. FUTURE WORK AND CONCLUSIONS

Our approach allows for a flexible adaptation of a dialogue system to various applications. Empirical evidence for this claim stems from experience with other projects ([3]). The separation of responsibilities and capabilities among the agents in the system proved to be useful for modularity, scalability and portability.

6. ACKNOWLEDGEMENTS

We thank Prof. Görz and Prof. Niemann for their support and many helpful discussions on earlier drafts of this paper.

7. REFERENCES

- [1] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Foundations of Knowledge Representation*, pages 191–236. CSLI-Publications, 1996.
- [2] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. Bradshaw, editor, *Software Agents*. MIT Press, 1997.
- [3] A. Hildebrand and V. Sá. Embassi: Electronic multi-media and service assistance. In T. K. A. Heuer, editor, *Proceedings of the Workshop on Intelligent Interactive Assistance and Mobile Multimedia Computing*, pages 50–59, Rostock, 2000.
- [4] A. Huber and B. Ludwig. A natural language multi-agent system for controlling model trains. In *AIS 2002—Simulation and Planning in High Autonomy Systems*, pages 145–149, Lisbon, 2002.
- [5] B. Ludwig. Dialogue Understanding in Dynamic Domains. In *BI-DIALOG 2001—Proceedings of the 5th Workshop on Formal Semantics and Pragmatics of Dialogue*, pages 287–297, Bielefeld, 2001.