

Extensibility and Reuse in an Agent-Based Dialogue Model

Anh Nguyen and Wayne Wobcke
School of Computer Science and Engineering
University of New South Wales
Sydney NSW 2052, Australia
{anht,wobcke}@cse.unsw.edu.au

Abstract

Most existing dialogue systems have been developed specifically for some predefined task domain(s), thus the issues of extensibility and reuse of the dialogue model are often not given much attention. In this paper, we describe our work on extending an existing agent-based dialogue model for e-mail management to further handle the calendar task domain. We discuss how our approach can achieve modularity of the dialogue model at the level of agent plans, facilitating the reuse of discourse-level plans and enabling the extension of the dialogue model to cover different application domains.

1. Introduction

Natural language dialogue systems can allow a more natural and effective way for human-computer interaction, through a conversational user interface. Dialogue modelling and control are of great importance for these systems to be able to recognize the user's requests and maintain a coherent interaction. Existing dialogue systems are mostly developed for some predefined task domain(s). The dialogue requirements are usually well specified and once the dialogue model has been developed, it is not intended to have further adaptation or extension for use in other applications. Hence the issues of reuse and extensibility have been given little consideration. However, being able to reuse a dialogue model would reduce the cost of developing a dialogue system for a new task domain. Moreover, a multi-domain dialogue system would likely require the addition or removal of domain-specific back-end components, and the extension or modification of the dialogue model for handling different domains. This suggests a need to focus on the issue of extensibility in dialogue modelling. We believe a generic dialogue model can be developed for a certain class of domains, which would enable the adaptation of the model for a new domain or its extension for handling additional tasks.

In this paper, we discuss extensibility and reuse in our agent-based dialogue management system, the Smart Personal Assistant (SPA) [8]. Many existing systems make use of a task model and a dialogue model, however these are not always clearly distinguished. In the SPA, there is a clear separation between the task model and the dialogue model due to the use of a special dialogue manager agent. The SPA requires sophisticated dialogue interaction, and so requires a complex dialogue model, but not necessarily a complex task model. For our application, it is more important that the dialogue model can be reused, so that the system can be extended without much cost, at the same time maintaining the level of dialogue sophistication. The modularity of the SPA's dialogue model is at the level of agent plans, enabling the reuse of discourse-level plans, so that adding a new task domain requires only the development of some domain-dependent plans and domain-specific knowledge.

We now briefly review existing approaches to dialogue modelling and discuss how these methods can support extensibility and reuse. The idea behind the state-based approach is to enumerate all possible dialogue states and represent them in a graph or finite state machine [3]. State-based dialogue models contain largely domain-dependent information, i.e. each state is defined by a set of variables representing different aspects (most are domain specific) of the ongoing conversation. Thus building a state-based dialogue system for a new application domain requires developing a new set of dialogue states and transitions.

Frame-based approaches consider dialogue interaction as a slot-filling problem [1]. The dialogue models consist of predefined sets of ordered rules to constrain the possible sequences of user utterances. The task model is represented as a hierarchy of conversation topics or frames with several slots for related pieces of domain-specific information. Although some of the rules or frames can be reused, extending a frame-based dialogue model for handling additional task domains is difficult because of the lack of modularity. Moreover, appropriate priorities among dialogue control rules are crucial but difficult to specify and maintain.

The ProBot dialogue scripting language of Sammut [7] facilitates the development and maintenance of dialogue control rules by grouping them into contexts (scripts), each corresponding to some dialogue topic. Although some degree of modularity can be attained, there is still the difficulty of reusing and extending a large rule-base of scripts.

Mirkovic and Cavedon [4] have proposed a method for extension of dialogue models that are based on the dialogue move and information state update approach. A dialogue move scripting language has been introduced, which allows inheritance of existing (possibly domain-independent) dialogue move scripts for developing domain-specific extensions. Each script consists of conditions for mapping the user input to a dialogue move and rules to specify the required information updates. Inheritance in this approach is similar to the use of *interface* in the Java programming language, which helps improve abstraction but not much reuse.

Collagen [6] is a Java middleware for building dialogue systems, which has been used in a number of applications. Collagen uses plans to model tasks and focuses on using plan recognition for identifying the user's requested tasks and managing the dialogue interaction. Collagen provides a common architecture for developing dialogue systems that require complex task models but constrained dialogue interaction. On the other hand, our application needs to support more flexible dialogue interaction, thus requires a more complex dialogue model.

Our application, the SPA, is a multi-agent system, which consists of a central dialogue manager agent, a user interface agent and back-end task assistant agents. The system allows the user to interact with the collection of task assistants through a unified interface that includes natural language dialogue. We consider dialogue interaction as rational action, exhibiting goal-directed behaviour. Thus we model the dialogue as a set of modular plans, each associated with a dialogue aspect [5]. The dialogue manager controls the dialogue interaction by appropriately selecting and executing its plans according to the context. In this model, discourse-level plans—corresponding to domain-independent aspects—are separated from domain-level plans—used for performing domain tasks. Hence, due to the modularity of this plan-based approach, there is a potential for the dialogue model, particularly the discourse-level plans, to be reused for other SPA-like applications. In fact, our initial implementation of the SPA [5] covered only the e-mail management domain, and our extended dialogue model [8] included an additional calendar management domain and addressed coordination of the two assistants.

In the remainder of this paper, we first describe in more detail the SPA dialogue model for e-mail management tasks. Next, we discuss our work in reusing and extending the dialogue model for the integration of the calendar assistant.

2. Dialogue Modelling in the SPA

A BDI agent architecture based on the PRS system [2], JACK Intelligent AgentsTM, has been employed for the development of the SPA's dialogue manager agent. The dialogue model is encoded in the plans of the agent. Our approach to dialogue is based on the theory of speech acts, so that in order to recognize the user's intentions, the dialogue manager agent needs to first recognize the associated speech acts being performed. We separate domain-independent conversational acts from domain-dependent task goals, thus each dialogue plan is a modular unit, handling a discourse-level goal such as recognizing the user's intention or a domain-level aspect such as contacting a back-end assistant to perform a domain task. The plans are generic though make use of domain-specific knowledge.

Figure 1 shows the overall structure of our plan-based dialogue model, in which the dialogue manager's plans are roughly arranged into four different groups according to their purpose: *semantic analysis*, *pragmatic analysis*, *task processing* and *response and clarification generation*. Each group itself contains several plans. In this figure, there is more than one task processing group to indicate that there is a task processing sub-group for interacting with each task assistant. Dialogue processing is done automatically as the result of the BDI interpreter selecting and executing plans according to the current context. The selection of plans is triggered by the agent's external or internal events.

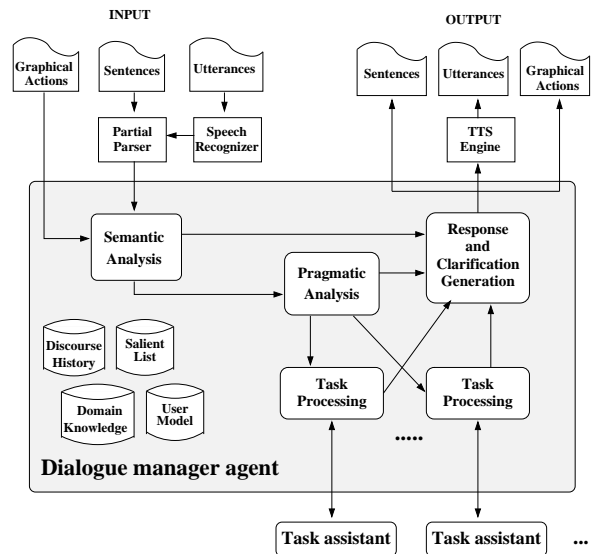


Figure 1. Agent-Based Dialogue Model

The dialogue manager agent maintains the conversational context and other domain-specific knowledge as its internal beliefs, which include the following:

- *Discourse History*: for maintaining the conversational context such as information about the current and past dialogue states.
- *Salient List*: for maintaining a list of objects which have been mentioned previously in the conversation, i.e. the objects that are in the focus of attention.
- *Domain-Specific Knowledge*: includes domain-specific vocabulary and information of the tasks that are supported. This is used in interpreting the user's requests.
- *User Model*: for maintaining information about the user such as current device, preferred modality of interaction, physical context, preferences, etc.

3. Dialogue Model Reuse and Extension

Our agent-based approach to dialogue management considers dialogue acts as rational actions, exhibiting goal-directed behaviour. Using a BDI agent architecture allows the dialogue model to be explicitly represented in dialogue plans so that both dialogue and problem solving plans can be expressed in a common framework. Thus our plan-based dialogue model achieves modularity at the level of agent plans but also allows a clear separation between discourse-level and domain-level dialogue plans. Moreover, domain-specific knowledge is defined in a uniform format for each back-end task assistant and provided to the dialogue manager when the task assistant is connected to the system. This reduces the amount of effort required to adapt or extend the dialogue model for use in different task domains. For example, in our initial implementation with only the e-mail task assistant, the dialogue model has about 20 discourse-level plans and 6 domain-level plans, in addition to some auxiliary plans for handling system authentication, etc. In extending the dialogue model for the calendar domain, we have reused the discourse-level plans, hence needed to develop only another 6 domain-level plans and the calendar domain knowledge description.

3.1. Reusable Discourse-Level Dialogue Plans

The dialogue manager's discourse-level plans are used for analysing the user's utterances, which includes: classifying the utterances to a task domain, formulating a semantic representation of the utterances, identifying the conversational acts being performed and recognizing the underlying intentions of the user. Each plan is a modular unit used for handling a specific domain-independent dialogue aspect. There are roughly 20 discourse-level plans in our implementation of the SPA. However, in other applications, the number of discourse-level plans could vary,

depending on the complexity of the required dialogue interaction. If the application domain requires more complex dialogue behaviours, our approach would facilitate the addition of discourse-level plans, for instance, to handle additional conversational acts. Note that the generic and domain-independent nature of these additional plans can be preserved, and so the reusability and extensibility of the dialogue model. The important point is that our approach achieves modularity at the level of plans and there is a separation between discourse-level and domain-level plans, thus enabling the reuse and extension of our plan-based dialogue model for other applications. We list below some of the most important discourse-level plans in our dialogue model:

Domain Classification plan: Classify the user's utterance into a task domain by automatically calculating the likelihood based on the occurrences of domain-specific keywords. These keywords are predefined in a domain-specific vocabulary, which is kept in the dialogue manager's domain knowledge.

Semantic Analysis plan: Formulate a semantic representation of the user's utterance. The manipulation process is based on the general linguistic knowledge of the dialogue manager as well as some limited domain knowledge. This domain knowledge is supplied systematically by the task assistants so that the use of this knowledge in the plan does not affect the plan's generic nature.

Act Type Determination plan: Determine type of the conversational act that the user is performing. This is done by considering syntactic and semantic features of the user's current utterance and the previous conversational acts performed by both the user and the dialogue manager. This information is maintained in the conversational context, which is kept in the discourse history.

Intention Identification plan: Identify the user's intention, i.e. the requested task. The diagram in Figure 2 illustrates the execution of this plan. The Start, End and Fail points indicate the beginning, end and failure of plan execution. Diamonds denote decision making points. Rectangles indicate calls to procedures for data manipulation. The curved rectangles denote the executions of a sub-plan. First, a set S of the possible user's conversational acts is determined by the *Act Type Determination* plan. The dialogue manager selects the most likely element in this set to be the user's current conversational act CA . Depending on the type of CA , either a request must be identified or the partially recognized request in the discourse history must be updated. An appropriate event is raised which triggers execution of another plan for handling this conversational act. This involves identifying the type of the requested task or the attributes of the domain objects mentioned in the request. If the intention identification process fails, the dialogue manager selects the next element in S , and repeats this process.

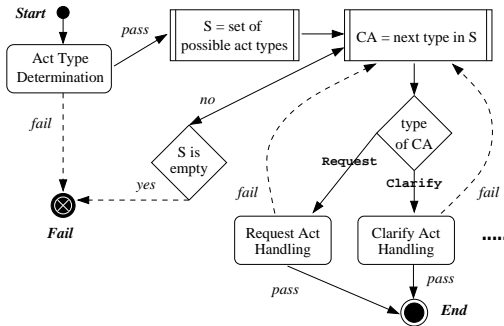


Figure 2. Intention Identification Plan

Act Handling plan set: There is one plan for each possible type of the user’s conversational acts. In executing the plan, the dialogue manager appropriately updates the dialogue’s intentional structure (discourse history) and triggers execution of other plans for determining the user’s intention based on the recognized conversational act.

Reference Resolution plan: Resolve anaphoric references and temporal adverbs. The resolution of definite noun phrases (of the domain objects) requires the dialogue manager to interact with the back-end task assistants.

Task Type Determination plan: Use keywords in the domain-specific vocabulary to recognize the type of the requested task. Because the domain of the task has been determined, the selection of the corresponding domain knowledge can be done automatically, allowing the plan to be domain-independent.

People Determination plan: Determine if any contact names or references to some persons are mentioned in the user’s utterance. If any task assistant provides an address book then from within this plan, the dialogue manager may interact with the assistant to identify a set of people that best match the mentioned name.

Clarification Generation plan: Generate clarification or confirmation requests so that they can be directed to the user for resolving ambiguities. These questions are generated using predefined templates provided in advance by the back-end task assistants and retrieved automatically from the dialogue manager’s domain knowledge.

Graphical Action Handling plan: Process the user’s actions on the device’s graphical interface, which may involve performing domain tasks such as deleting e-mail messages, going back to show the previous display or updating the salient list.

Response Generation meta-plan: Select appropriate domain-level plans for generating system responses which are tailored to the user’s device, context and preferences.

3.2. Domain-Specific Knowledge

In order to provide sophisticated dialogue interaction, the language vocabulary (e.g. proper names, objects) used in the application should be unconstrained. It is also impractical to build and inefficient to use a large dictionary for language understanding. Nevertheless, because the task domains are known (e.g. e-mail or calendar management), domain-specific dictionaries can be constructed with reasonable effort. These dictionaries provide domain-specific keywords which can be used as the boundary patterns for extracting unconstrained attribute values in the user’s utterances. For example, considering a possible phrase describing a search: “Find all new messages about school meeting”, the system does not need to understand the logical meaning of “school meeting” because the presence of other domain-specific keywords such as *find*, *messages* and *about* is sufficient to recognize the user’s search request. In this search, the phrase “school meeting” represents the topic of interest, which can be used for information retrieval without the system understanding the meaning of the phrase.

The domain-specific knowledge also includes other information about the types of tasks that are supported by each task assistant. Moreover, for each task type, there is information about the required parameters or the kinds of objects to be manipulated. Hence in processing the user’s utterance, the dialogue manager can use the domain-specific vocabularies together with a description of the object’s attributes to identify if any objects are mentioned in the utterance. Other important information is also specified in the task description, such as whether or not the task requires a confirmation. For instance, an appointment cancellation task should be confirmed by the user to avoid mistakes.

To achieve interoperability, the domain knowledge is predefined for each task assistant using a common format, and then made available to the dialogue manager when required. This enables the addition and removal of task assistants to and from the SPA. In addition, this domain-specific knowledge is used to formulate task descriptions in an appropriate format for delegation to the corresponding task assistant. The following shows a part of the domain knowledge description for the e-mail management domain, which is represented in XML format, showing the mapping from verbs to domain tasks.

```

<object type="DomainKnowledgeDesc" domain="EMAIL">
  <object type="TaskRequest">
    <field name="type">ARCHIVE</field>
    <object type="TaskObject">
      <field name="type">MAIL</field></object>
    <object type="TaskObject">
      <field name="type">FOLDER</field></object>
  </object>
  <object type="Verb">
    <field name="probability">80</field>
    <field name="word">file</field>
    <field name="synonym">archive</field>
    <field name="mappedTask">ARCHIVE</field>
  ...

```

3.3. Domain-Level Plans for E-Mail Management

Domain-level plans are those that require domain-specific knowledge to be encoded in the plan, and include the following:

Domain Object Determination plan set: Determine conditions or attribute values of any domain objects mentioned in the user's request. This may involve combining information in the user's current utterance (such as in the case of clarification or confirmation) and in the partially recognized request. For the e-mail management domain, there is an *E-mail Determination* and a *Folder Determination* plan.

Domain Task Processing plan: Processing tasks in each domain, which may require interacting with the back-end assistant agent, such as to search for new messages or archive messages to some folder. If interaction with some back-end task assistant is required, the dialogue manager delegates the task to the task agent. Otherwise, other plans are triggered for immediate response generation.

Domain Task Response Handling plan: There is one plan for processing responses from each back-end task agent. The responses may be notifications of new message arrivals or responses to the delegated tasks.

Domain Task Response Generation plan set: There is one plan set for generating system responses in each task domain so that the system responses can be adapted according to the user's device and available modalities.

3.4. Dialogue Extension for Calendar Domain

Although extending the dialogue model for handling the additional calendar task domain means that a new set of domain-level plans must be defined for the calendar task assistant, these domain-dependent plans are very similar in structure to those of the e-mail management domain. Therefore, defining domain-level plans for new applications does not require much effort. Even though the total number of the dialogue plans must increase as the number of integrated task assistants increases, the dialogue model is scalable and computational efficiency is not affected because most of the time the dialogue interaction is in a single domain, thus involves only the plans in that domain.

For handling the calendar management domain, we have extended our initial dialogue model to include new domain-level plans, including an *Appointment Determination* plan, a *Todo Determination* plan, a *Calendar Task Processing* plan, a *Calendar Task Response Handling* and a *Calendar Task Response Generation* plan set. The extended dialogue model allows the user to have continuous dialogue interaction with both e-mail and calendar task assistants as illustrated in a short example scenario as follows:

```
User Is there any new mail from John?
SPA You have two new messages from John Lloyd.
User Show me the one about slides please.
SPA Displays correct message from John Lloyd
User I need to see him 5pm tomorrow about the slides.
SPA Do you want to enter that appointment to calendar?
User Yes, enter it to the Research category please.
SPA Where are you going to meet him?
User My office.
SPA Appointment has been created.
```

4. Conclusion

Research in dialogue systems needs more consideration on the issues of extensibility and reuse of dialogue models to help reduce the cost in developing new dialogue systems by adapting or extending existing models. We have discussed our work in extending an agent-based dialogue model for handling additional task domain. The modularity of the dialogue model at the level of the agent plans facilitates the reuse of discourse-level plans and the addition and extension of domain-level plans.

5. Acknowledgments

This work was funded by the CRC for Smart Internet Technology. We would also like to thank Agent Oriented Software Pty. Ltd. for the use of their JACK agents platform.

References

- [1] H. Aust, M. Oerder, F. Seide and V. Steinbiss. The Philips Automatic Train Timetable Information System. *Speech Communication*, 17:249–262, 1995.
- [2] M. P. Georgeff and A. L. Lansky. Reactive Reasoning and Planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 677–682, 1987.
- [3] L. B. Larsen and A. Baekgaard. Rapid Prototyping of a Dialogue System Using a Generic Dialogue Development Platform. In *Proceedings of the Third International Conference on Spoken Language Processing*, pages 919–922, 1994.
- [4] D. Mirkovic and L. Cavedon. Practical Plug-and-Play Dialogue Management. In *Proceedings of the Sixth Meeting of the Pacific Association for Computational Linguistics*, 2005.
- [5] A. Nguyen and W. Wobcke. An Agent-Based Approach to Dialogue Management in Personal Assistants. In *Proceedings of the 2005 International Conference on Intelligent User Interfaces*, pages 137–144, 2005.
- [6] C. Rich and C. L. Sidner. COLLAGEN: A Collaboration Manager for Software Interface Agents. *User Modeling and User-Adapted Interaction*, 8:315–350, 1998.
- [7] C. A. Sammut. Managing Context in a Conversational Agent. *Electronic Transactions on Artificial Intelligence*, 5(B):189–202, 2001.
- [8] W. Wobcke, V. Ho, A. Nguyen and A. Krzywicki. A BDI Agent Architecture for Dialogue Modelling and Coordination in a Smart Personal Assistant. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 323–329, 2005.