# An Agent-Based Approach to Dialogue Management in Personal Assistants

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

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

## ABSTRACT

Personal assistants need to allow the user to interact with the system in a flexible and adaptive way such as through spoken language dialogue. In this research we focus on an application in which the user can use a variety of devices to interact with a collection of personal assistants each specializing in a task domain such as email or calendar management, information seeking, etc. We propose an agent-based approach for developing the dialogue manager that acts as the central point maintaining continuous user-system interaction and coordinating the activities of the assistants. In addition, this approach enables development of multi-modal interfaces. We describe our initial implementation which contains an email management agent that the user can interact with through a spoken dialogue and an interface on PDAs. The dialogue manager was implemented by extending a BDI agent architecture.

**Categories and Subject Descriptors:** I.2 [Artificial Intelligence]: Natural Language Processing, Distributed Artificial Intelligence

**General Terms:** Human Factors.

**Keywords:** Dialogue Modelling, Personal Assistant, BDI Agent Architecture.

## 1. INTRODUCTION

In recent years, considerable research effort has been put into the area of dialogue modelling for computer-based applications. The goal is to allow high-level, human-like user-system interaction such as through spoken language interfaces, which become feasible in domains that are sufficiently constrained to overcome the limitations of speech recognition over large vocabularies. To ensure flexible, adaptive and continuous interaction, dialogue management for such interfaces is of particular importance. The main functions of a dialogue manager are to maintain the conversational context and model strategies for controlling the dialogue structure. Our intended application is a "Smart Personal Assistant" (SPA), which is a suite of personal assistants, each specializing in a particular application domain such as email or calendar management. For example, the user should be able to do email tasks such as searching, deleting, archiving, replying to emails and requesting for notification of important email arrivals. This integrated collection of assistants can be accessible via a range of devices (e.g. PDAs, phones, desktops) and the communication is through spoken or typed-text dialogue and GUI. The application must provide a single point of contact so that the user can easily switch back and forth between different personal assistants. The user may also change physical context by using different devices, although it is unlikely that more than one device would be used at the same time. The dialogue manger should be the central component for maintaining a coherent dialogue with the user and also coordinating the activities of the personal assistants. The research in this paper aims to model the natural language dialogue needed for such applications.

Most existing spoken dialogue systems focus on simple and constrained tasks. Some examples are the telephone-based flight and travel-package booking systems by Pellom *et al.* [18], by Seneff and Polifroni [23], and by Xu and Rudnicky [28]. Earlier systems are the Philips train information system [2] or the MIMIC system for querying movie show times [6]. Dialogue structures in these systems are represented by some prescriptive grammar. This grammar can be a finite state machine/graph consisting of conversation states. Other approaches use pre-defined sets of ordered rules to constrain the possible sequences of user utterances, or a hierarchy of conversation topics each represented as a frame with several slots for related pieces of information, such as in the YPA directory enquiry system [8]. Managing the dialogue in this frame-based approach can then be cast as a slot-filling problem. A disadvantage of these approaches is that maintenance of the system's topic hierarchy and/or rule set requires significant effort. Another approach, used in the TRINDI project [15], is based on the concepts of information state and dialogue move. The information state represents the relevant aspects of information in the dialogue and is updated in every dialogue move. The updates are governed by a set of update rules.

Those approaches are not suitable for applications such as the SPA. They are fairly simple and cannot provide the desired level of sophistication required for complex domains such as remote email management, in which several interactions may be required to complete a single task. The dialogue model in the SPA must be user-independent (so that the system can be deployed for different users) but it must also adapt to the user's device, environment and preferences. For example, the system must decide not to use spoken dialogue in environments such as seminars since it would be socially inappropriate. More interesting is the potential for the character of the dialogue to change during a course of interaction. The user may change physical context by moving around and/or being connected to new devices. While full information can be shown on large-screen devices (e.g. desktops), only important information should be displayed on small devices such as PDAs. Moreover, the dialogue model should be robust enough to be able to recover from poor quality speech recognition. In the case of a recognition error, the system can open a sub-dialogue for clarification.

There has been other work on modelling dialogue for complex task domains. In the TRAINS system [1] and its successor, TRIPS [4], Allen and others view dialogue as a collaborative process in which the user and the system work together for some problem solving task (e.g. an emergency vehicle dispatch scenario). Approaches such as in these systems would be unnecessarily complicated for the SPA. The SPA application domain involves less problem solving but more continuous interactions than in those systems. Thus the dialogue manager should be light enough for this problem, and focus more on the user's interaction rather than on problem solving.

The dialogue in the case of the SPA is mainly user-driven. Nevertheless, system initiative is also essential for clarifying user requests or notifying the user of important events. Hence the dialogue management requires some degree of proactiveness as for error recovery but also reactiveness in order to fulfil the user's requests. We therefore believe it is advantageous to design the dialogue manager as an agent. The dialogue management agent will maintain coherent interaction with the user as well as being a coordinator that directs the actions of the specialist agents such as email and calendar agents. For example, the agent may have a number of plans each specifying the agent's actions for achieving a goal such as for utterance interpretation, identifying the user's intention, response generation, etc. (more details are given in Section 2.1). The agent looks through its plans to find those that are relevant to its goal and applicable to the situation and executes them. Learning can also be integrated into the agent by incorporating meta-plans as intermediate steps for plan selection. An agent-based approach is also beneficial in other respects. The issue of semantic interoperability is solvable by formulating declarative representations of the capabilities of individual agents and of the SPA as a whole, which will enable back-end integration with existing personal assistants. Agents embody the characteristic of modularity. Adapting an agent-based dialogue model for integration of more specialist agents becomes easier with this approach, requiring the addition to the agent of domain-related plans and internal knowledge. Hence, modularity can be achieved at the level of agent plans. Another important aspect is tracking the conversational context to allow the system to conduct coherent dialogue with the user.

The idea of employing an agent paradigm is also motivated by existing research on intelligent user interfaces. Initially, there were arguments over which of the two interaction metaphors would be better for user interface development: the tool metaphor or the agent metaphor. In one view, the system interface is viewed as a tool that allows the user's direct manipulation of available options in order to perform some task, whilst researchers of the interface-as-agent view suggest that intelligent interfaces should better be structured as agents. This view is strengthened by Chin's argument in [5] that an intelligent interface must exhibit to some degree autonomous and intelligent behaviour such as taking the initiative to correct user misconceptions or suggest alternative actions. A more recent proposal by Horvitz [12] was to improve user-system interaction by combining the tool and agent metaphors into a single framework. In the case of the SPA, it is essential to provide this sort of mixed-initiative interface (e.g. through natural language dialogue) in addition to the conventional GUI due to the limitations (e.g. small keyboard, stylus for input, small screen for output) of mobile devices such as phones and PDAs.

In this paper, we discuss our work in extending an agent architecture for the implementation of the SPA dialogue manager. The current system works for typed-text dialogue. We have also done experiments with the IBM ViaVoice speech recognition software. Although the dialogue manager was able to recover from some recognition errors, our experience was that ViaVoice is inconsistently unreliable, thus not adequate for our application. Currently we are testing a version of Dragon NaturallySpeaking. Initial testing suggests that the speech recognition errors will be more predictable. Our application does not require speech recognition of 100% accuracy. However, certain common keywords (e.g. "mail", "from") need to be reliably recognized. Otherwise, recognition errors (e.g. for proper names) need to be consistent to enable correction by the dialogue manager. The structure of the paper is as follows. In Section 2, we discuss how an agent architecture can be extended for dialogue modelling. Section 3 illustrates the dialogue processing in more detail which includes an overview of the SPA system. The current development status is given in Section 4. Section 5 studies the related work and finally, the conclusion and future work is in Section 6.

## 2. AGENTS FOR DIALOGUE MODELLING

### 2.1 BDI Agent Architectures

There is a lack of consensus in the agent community about what exactly constitutes an agent. According to Wooldridge and Jennings [27], an agent should have at least one *informational attitude* (e.g. knowledge or belief) to maintain information about the environment and one *pro-attitude* (e.g. desires, goals or intentions) to guide the agent's actions. Many agent architectures in the literature follow the well-known BDI model, in which an agent has three primary mental attitudes of Belief, Desire and Intention. These attitudes respectively represent the informational, motivational and deliberative states of the agent, Rao and Georgeff [19].

One instance of BDI agent architectures, the PRS (Procedural Reasoning System) architecture, was developed by Georgeff and Lansky [9]. The PRS abstract architecture is presented in Figure 1. The inputs to the agent are events from the environment and the outputs are the agent's actions. The interpreter loops and generates an action in each cycle. External and internal events are always added to an event queue. The agent's beliefs are adjusted according to those events. At the beginning of a cycle, plans are chosen from the plan library which specify courses of action that may be undertaken in order to achieve the agent's goals. Next, the deliberator, a component of the interpreter, selects a subset of these plans to be adopted and adds them to the intention structure. The agent then executes one action from one plan in the intention structure. The intention and goal structures are modified by dropping successful goals and satisfied intentions, as well as impossible goals and unrealizable intentions. Hence, due to new external events, the agent can reactively choose to drop intended plans and/or adopt others.

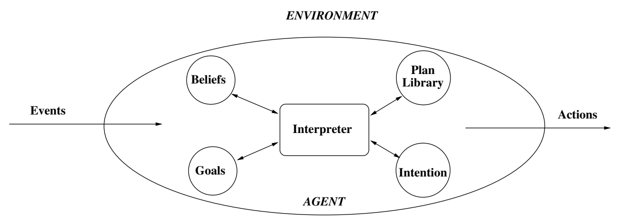


**Figure 1: PRS Architecture**

PRS has been shown to operate very effectively in continuously changing environments such as military scenarios. In such environments, the agent must respond appropriately in reacting to the situation as well as be deliberative in fulfilling its prior goals. The situation in our SPA system is probably less dynamic. However, the dialogue management must similarly balance the requirements of being reactive and pro-active, as has been discussed earlier in Section 1.

Therefore we chose a PRS-style agent platform, JACK Intelligent Agents$^{TM}$ [13], for developing the dialogue manager agent. JACK is a Java implementation derived from the PRS architecture which also includes mechanisms for agent communication by exchanging messages. In the current SPA implementation, the dialogue manager is a BDI agent that has a set of plans for:

- Identifying conversational acts
- Identifying the user's intentions
- Performing tasks by contacting appropriate PAs (e.g. email, calendar agent)
- Generating system responses

The BDI agent architecture in JACK provides a general framework for implementing the dialogue manager. In addition, this framework has also been extended for including the dialogue data structure to maintain the conversational context. Next, we describe in detail those extensions.

## 2.2 Dialogue Data Structure as Agent Beliefs

Dialogue is a kind of discourse where each participant periodically takes turns to be speaker and hearer. Discourse research has been closely related to a question of what information is being conveyed in a coherent discourse, in addition to the literal meanings of the individual utterances. There are two major lines of approaches which are referred to as *informational* and *intentional.* Informational approaches consider the coherence of discourse to derive from semantic relationships between the information expressed by successive utterances. On the other hand, according to intentional approaches, the coherence of discourse comes from the intentions of the speaker, and understanding depends on recognition of those intentions.

We follow the intentional approaches, which are consistent with early work on speech act theory by Austin [3] and Searle [22]. Briefly, the speaker in a dialogue, by making an utterance, intends to perform some action known as an *illocutionary act* (generally termed speech acts). All speech acts can be classified into five major classes: assertives, directives, commissives, expressives and declarations. The intention of the speaker can be recognized from the speech acts being performed.

Computational work in discourse by Grosz [10] has identified three components of discourse structure: the linguistic structure (the sequence of utterances), the intentional structure and the state of focus of attention. Utterances in a discourse naturally form a hierarchy of discourse segments. Each discourse segment has a discourse segment purpose (or discourse segment intention). Discourse intentions differ from other kinds of intentions that they are intended to be recognized. The attentional state has an important part which contains the salient entities that have been mentioned earlier in the discourse.

With the level of complexity in our task domain, we make an assumption that although there may be multiple speech acts being performed in any given utterance, only one of them is important in determining the user intention. We call this act a conversational act. Moreover, the intention of every highest-level discourse segment is to fulfil a domain task. Thus we separate domain-independent conversational acts from domain-dependent task goals. The defined conversational acts are shown in Table 1. In carrying out the domain tasks, the dialogue manager (DM) can also interact with other personal assistants (PAs). The interactions between the DM and PAs are also modelled as performing conversational acts with associated intentions/domain tasks.

| **Conversational Act** | **Act Description** |
|---|---|
| `Request` | ask the addressee to perform a domain task |
| `Respond` | describe the task result to the hearer |
| `Clarify` | ask the addressee to clarify ambiguities |
| `Greet` | express the speaker's greetings and/or feelings |
| `Confirm` | clarify ambiguities by expressing agreement or disagreement |
| `Ack` | express the acknowledgement |

**Table 1: Conversational Act Descriptions**

The DM internal belief state contains specific domain knowledge for help in interpreting user utterances such as a domain-specific term dictionary. Additionally, there is a discourse history and a focus stack developed to maintain the discourse context.

### 2.2.1 *Discourse History*

The *discourse history* keeps all the interactions as a stack of conversational acts. They include the DM-user as well as the DM-PAs interactions. The discourse history also keeps track of the user intention being carried out in the current discourse segment. In many cases, the user intention cannot be recognized in the first utterance but it is required that discourse subsegments to be opened for more information or clarification. The discourse history is also used in determining which acts the user has performed and what the underlying intentions are. Table 2 illustrates an example of the discourse history stack.

| | |
|---|---|
| `Greet` | `ID=122,Sender=User,Receiver=DM,Subj="..."` |
| `Request` | `ID=123,IsReplacedBy=126,Sender="User",Receiver="DM", Task=SEARCH,Object=EMAIL,Condition={From="John"}` |
| `Clarify` | `...,Subj=(From="John Lloyd"\|\|"John McAfee")` |
| `Confirm` | `InResponseTo=124,...,Subj=(From="John Lloyd")` |
| `Request` | `ID=126,AsReplaceFor=123,...,Task=SEARCH, Object=EMAIL,Condition={From="John Lloyd"}` |
| `Request` | `...,Sender=DM,Receiver=Emailer,...` |
| `Respond` | `Sender=Emailer,Receiver=DM,MailId={...}` |
| `...` | `.........` |

**Table 2: Sample Discourse History Stack Entries**

### 2.2.2 *Focus Stack*

The *focus stack* is a data structure for tracking objects that have been mentioned during the course of conversation. Information in the focus stack is used to resolve references, that is when certain words are used in current utterance to refer back to other referring expressions in the previous utterances. It also helps in generating context-sensitive natural language responses. In the email management domain, the objects to be kept in the focus stack include mail items, folders, names (of contacts) and key phrases. In the SPA application, the user interacts with the system through dialogues as well as the graphical interface on mobile devices. Hence objects involved in the interaction on the device are also put onto the focus stack. The details of how the focus stack evolve will be explained later as we go through examples in Section 3. There are separate stacks for each individual object type.

It is important to note that dialogue (spoken and written) differs from other kinds of discourse in some characteristics such as turn-taking, grounding, etc. With the proposed dialogue model, the user and the DM alternately take turns in the conversation. The user takes turn when requesting tasks or confirming information while the DM takes turn when requesting clarification or giving back the task results.

Grounding in the literature is commonly understood as a process through which the speaker and the hearer constantly establish common ground which are things that they both mutually believe, Clark [7]. This is done by the hearer acknowledging the speaker's utterances or specifying any problems arising in reaching the common ground. In our model, the `Ack`, `Clarify` and `Confirm` conversational acts are used for grounding; `Ack` and `Confirm` are for acknowledgement while `Clarify` is for resolving ambiguities.

## 3. DIALOGUE PROCESSING

## 3.1 SPA System Overview

The dialogue model is integrated within the SPA system. Currently the SPA has one personal assistant for email management but a calendar agent will be integrated in the near future. The system components are as illustrated in Figure 2, which include a PDA interface, a speech engine, a partial parser, the Dialogue Manager agent and the Email agent. The agents interact with each other using the JACK communication mechanism.

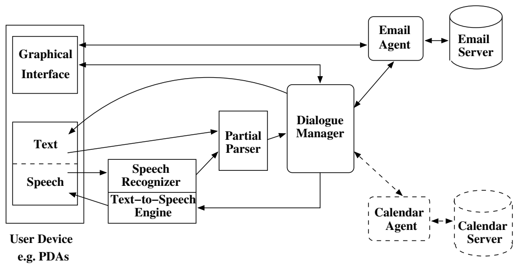


**Figure 2: The SPA System Architecture**

The Email agent is an agent wrapper built around the existing EMMA email management software enabling the Dialogue Manager to request the EMMA system to perform tasks such as searching, deleting and archiving emails, and notifying the user on the arrival of some important messages. In addition, EMMA can perform classification of mail in the Inbox into sorting folders and prioritization based on the degree of importance, urgency, etc. This information is given to the user and also used in processing email tasks. For example, the user may ask *"Do I have any new messages about meetings?"* and there exists a *meeting* virtual folder. Hence, the classification results of this virtual folder can be used, which is more appropriate than searching for emails containing the word *"meeting"*. The meaning of *"about meetings"* varies between users but should be consistent with their predefined classification rules. Details of the EMMA system can be found in Ho, Wobcke and Compton [11].

The speech engine handles speech recognition and text-to-speech synthesis. In the current implementation we use IBM ViaVoice software in dictation mode. The user is required to spend time training the voice model. However, the design of the system architecture is independent of the particular speech recognition engine used. The speech recognition engine together with the partial parser are built on top of the InCA architecture developed by Kadous and Sammut [14].

### 3.1.1 *Partial Parser*

The first stage in processing the dialogue is to parse the user utterances into some shallow syntactic structure. Full parsing for this application domain is inappropriate for the following reasons. Existing speech software can only provide a level of quality which is far less than perfect. In addition, many English speakers regularly use shortened forms of expression (abbreviated commands). Thus, it would be inefficient or even impossible to fully parse the user's utterances. Another reason for not using full parsing is that the language vocabulary (e.g. proper names, objects) in the application is unconstrained. Hence, building the dictionary for a full parser would be difficult. For example, consider the request *"Are there any new messages about ... ?"*. The missing phrase could be anything. However, because the task domains are known, a domain-specific dictionary can be constructed with a reasonable amount of effort. User utterances and sentences are partially parsed into a pre-defined shallow syntactic frame which has six components as follows:

- *type*: utterance type as one of : yes/no-question, wh-question, declaration, imperative, greeting.

- *subject*: the syntactic subject of the sentence.
- *predicate*: the main verb of the utterance as obtained from a domain-specific dictionary, e.g. delete, reply.
- *direct object*: main object of the predicate which can be email-related such as email, folder, or noun phrases.
- *indirect object*: possible second object which can also be email-related such as email, folder, or noun phrases.
- *complement phrase*: information e.g. time or location.

The partial parser is implemented using the ProBot[1] scripting language of Sammut [21].

## 3.2 Dialogue Manager

Dialogue processing for the email management domain is illustrated in Figure 3. The user input can be actions on the PDA graphical interface (such as selecting a message or folder) and/or spoken utterances or typed sentences. The user utterance is sent to speech recognition engine before its recognized text is passed to the partial parser. Next the Dialogue Manager agent receives output from the partial parser for further processing.

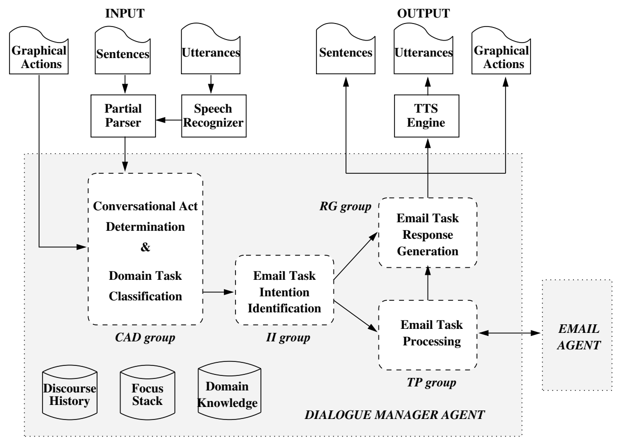


**Figure 3: Dialogue Processing Overview**

The Dialogue Manager agent has a number of plans which can be divided into four groups: conversational act determination and domain task classification (CAD group), intention identification (II group), task processing (TP group) and response generation (RG group). The input (graphical actions and parsed text) from the user is handled first by the plans in CAD group to determine the conversational act and identify the task domain. The user's specific intention is then identified by the plans in the II group. If intention identification succeeds, the requested task is handled by the plans in the TP group, which likely involves further requests to the Email agent. Otherwise, clarification is produced by the RG group. In the case of successful task processing, the plans in the RG group generate natural language text and instructions for displaying the results on the PDA. If the required reply mode has been speech then the system reply in text is sent to the Text-to-Speech (TTS) engine to generate an audio response. At the same time, other information is displayed on the interface accordingly. Thus the system supports limited multi-modal input and output.

### 3.2.1 Intention Identification

The intention of the user is considered a discourse segment purpose. In our application, a user intention corresponds to a domain task. Determining the task may require more than one user utterance, as in cases where there are ambiguities. Hence the intention is maintained in the discourse history and marked as being partially processed until it is identified successfully, i.e. when the requested task has been fully recognized. This process requires knowledge of discourse history, information in the recognized conversational act and heuristic rules. For example, suppose the user's last intention was to request a search for emails about some topic and five messages were found. If the current conversational act is another request with keyword *show*, it is likely to be a request to show one of those messages.

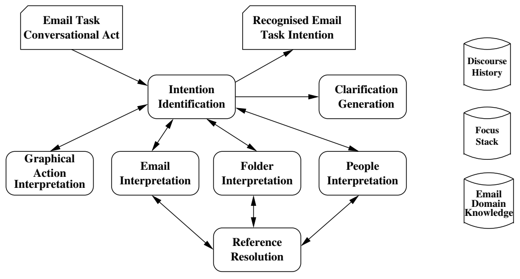


**Figure 4: Intention Identification Plans**

In order to identify the user's intention, there is one IntentionIdentification plan with a number of sub-plans. This plan can activate any of its sub-plans, namely, EmailInterpretation, FolderInterpretation, PeopleInterpretation, ReferenceResolution and GraphicalActionInterpretation. It is possible that the sub-plans can send messages to other personal assistants seeking required information for the interpretation. For example in identifying a person name, the PeopleInterpretation plan can request address book entries from the Email agent. Figure 4 shows the diagram of the plan and sub-plans. The arrows indicate the possible control flows. An example of a plan for performing email interpretation is shown in Figure 5.

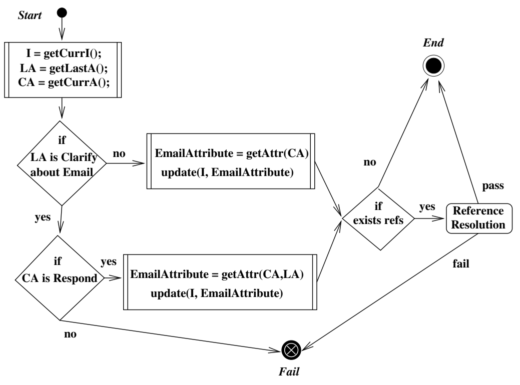

Start
I = getCurrI();
LA = getLastA();
CA = getCurrA();
if LA is Clarify about Email
no
EmailAttribute = getAttr(CA)
update(I, EmailAttribute)
yes
if CA is Respond
yes
EmailAttribute = getAttr(CA,LA)
update(I, EmailAttribute)
no
if exists refs
no
yes
Reference Resolution
pass
fail
End
Fail

**Figure 5: Example EmailInterpretation Plan**

[1]ProBot is a rule-based system embedded in a Prolog interpreter. Following is a simple example of scripting rule.

```
mail_subject::<noun-phrase> * ==> [#assert(subject(^1))
                                   #goto(mail_predicate,^2) ]
```

It indicates if a noun phrase is found when looking for the subject then assert this noun phrase to be the subject. The remainder of the utterance is passed on for determining other components.

The goal of the EmailInterpretation plan is to determine attributes of the emails mentioned by the user. In the plan diagram, there are three special points: Start, End and Fail indicating the beginning, end and failure of plan execution. When the plan fails, it triggers the execution of ClarificationGeneration plan. Diamonds in the diagram denote decision making points. Rectangles indicate calls to procedures for data manipulation. The curved rectangles are sub-plans.

The ReferenceResolution plan is designed to handle the interpretation of anaphora, i.e. the co-reference of one expression with its antecedent. More specifically, a pronoun or a definite noun phrase can be used to refer back to an entity that has been previously introduced. The GraphicalActionInterpretation plan is used to interpret user actions on the interface. The other four are to extract information about task attributes. Consider an example where the user highlights some phrase on the interface; it is likely that the user will soon refer back to the phrase. Hence, the Dialogue Manager updates its belief state by pushing the phrase onto the corresponding focus stack.

The focus stacks are maintained as follows. If an intention is successfully identified, all objects mentioned in this intention are pushed onto focus stacks. For the email management domain, object types are email, folder, person name and key phrase. If there is more than one object of the same type, they are pushed onto the focus stack as a set. A particular focus stack is searched when there is reference to some object of that type. According to the relation between the currently identified intention and the last intention, either some objects will be chosen from the set at the top of the stack or it must be popped off and the search continues. For example, consider the following conversation:

```
User     Do I have new mail about a meeting?
SPA      You have three messages about meetings,
         from Paul Compton, Supriya Singh and John Lloyd.
User     Show me the one from Paul.
SPA      <Displays message from Paul>
User     Show me the next one.
SPA      <Displays message from Supriya>
```

In the above conversation, after showing the mail from Paul to the user, this mail is at the top of email focus stack. Just below it is the set of three mails about meetings. In interpreting user's next utterance, *"Show me the next one"*, the mail from Paul Compton must be popped off the stack and *"the next one"* refers to an email in the set of three messages about meeting. As the interpretation goes on, that noun phrase must refer to the mail after the one from Paul Compton, thus it is the one from Supriya Singh.

If at some point, the Dialogue Manager fails to interpret the user utterance because of some ambiguity, the ClarificationGeneration plan will be executed and the user asked for clarification. The Dialogue Manager can also take the initiative to ask the user for more information. For example, the user requests *"Do I have any messages from John?"* and in the Inbox there are messages from two different John's. The system may seek clarification by asking *"Did you mean John Lloyd or John McAfee?"*. Note how the reply is contextually dependent on the current state of the Inbox.

### 3.2.2 *Task Processing and Response Generation*

Figure 6 shows the possible control flows between the Dialogue Manager's plans for processing domain tasks and generating responses to the user, generated as the result of the plan selection mechanism of the BDI agent interpreter.

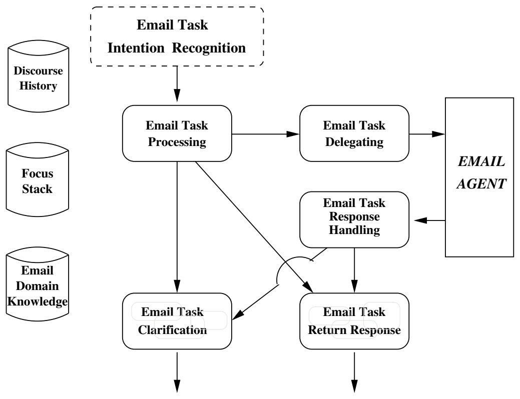


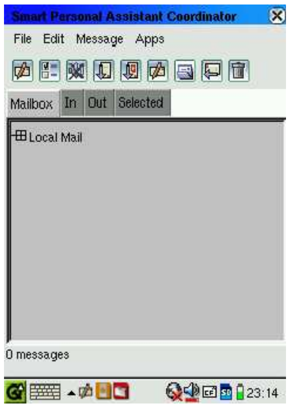
**Figure 6: Task Processing and Response Generation**

The Dialogue Manager can perform tasks immediately if the necessary knowledge is available in its internal beliefs. Otherwise, it requests information from the Email agent. Some tasks require more than one request to be sent. This is handled by the EmailTaskProcessing and EmailTaskDelegating plans. An example of a request message is shown below. The message is in XML format. Its enclosed request is a search for emails from John Lloyd.

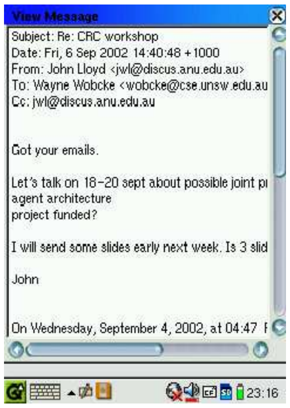
```
<object type="RequestMessage">
  <field name="sender">DM</field>
  <field name="receiver">Emailer</field>
  <field name="taskDomain">Email</field>
  <field name="task">Search</field>
  <field name="objects"><list>
    <object type="PAObject">
       <field name=type>Email</field>
       <list><object type="Condition">
         <field name="attName">From</field>
         <field name="function">contains</field>
         <field name="attValue">John Lloyd</field>
...
```

If the task is successfully processed, the resulting JACK message is enclosed in an event to be handled by the response generation plan set. This set includes different plans for generating different types of response. Depending on the task type and size of the result, the system can either display text on the device screen or speak out or choose to show either the full message content or just a brief description. There is a possibility for integrating a learning mechanism with the Dialogue Manager so that it can learn when to generate what responses. This may require maintaining the user's preferences and information about the physical context. In the case of inconsistency during task processing, one of the ClarificationGeneration plans is activated, generating an appropriate question for sending to the user.

There can be two-way information flow in the system. The user can request to be notified when particular mails arrive. Thus the Dialogue Manager registers that event with Email agent. Upon receiving notification from the Email agent, the EmailTaskResponseHandling plan processes the notification and activates ResponseGeneration plans to notify the user. The system may choose to immediately notify the user or defer the notification. However, prompt notifications might distract the user from the ongoing task. We have considered various approaches to this problem, some rule-based and some using machine learning techniques.



(a)



Subject: Re: CRC workshop
Date: Fri, 6 Sep 2002 14:40:48 +1000
From: John Lloyd <jwl@discus.anu.edu.au>
To: Wayne Wobcke <wobcke@cse.unsw.edu.au
Cc: jwl@discus.anu.edu.au

Got your emails.

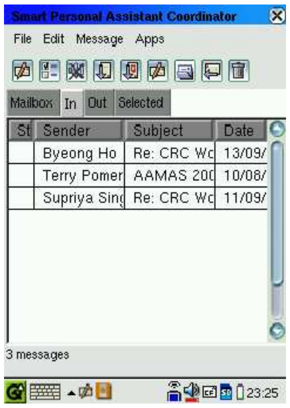
Let's talk on 18-20 sept about possible joint p
agent architecture
project funded?

I will send some slides early next week. Is 3 slid

John

On Wednesday, September 4, 2002, at 04:47

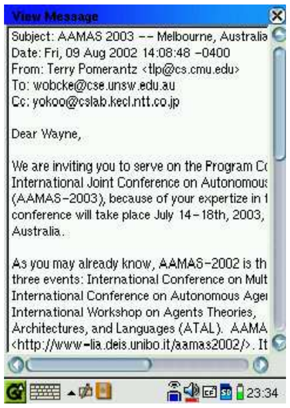
(b)



| St | Sender | Subject | Date |
|---|---|---|---|
| | Byeong Ho | Re: CRC Wc | 13/09/ |
| | Terry Pomer | AAMAS 200 | 10/08/ |
| | Supriya Sin | Re: CRC Wc | 11/09/ |

3 messages

(c)



Subject: AAMAS 2003 -- Melbourne, Australia
Date: Fri, 09 Aug 2002 14:08:48 -0400
From: Terry Pomerantz <tlp@cs.cmu.edu>
To: wobcke@cse.unsw.edu.au
Cc: yokoo@cslab.kecl.ntt.co.jp

Dear Wayne,

We are inviting you to serve on the Program C
International Joint Conference on Autonomou
(AAMAS-2003), because of your expertize in
conference will take place July 14-18th, 2003,
Australia.

As you may already know, AAMAS-2002 is th
three events: International Conference on Mult
International Conference on Autonomous Age
International Workshop on Agents Theories,
Architectures, and Languages (ATAL). AAMA
<http://www-lia.deis.unibo.it/aamas2002/>. It

(d)

**Figure 7: PDA Displays for Sample Dialogue**

## 4. CURRENT STATUS

Currently, dialogue can be used for searching, displaying and archiving emails. Our first experiments with spoken dialogue used the IBM ViaVoice speech recognition software. The SPA is able to recover from some speech recognition errors, however our testing showed that ViaVoice is unpredictably unreliable, hence not adequate for this task (on some occasions working with high accuracy, on others with low accuracy). We are now trying a version of Dragon NaturallySpeaking, and initial testing suggests that speech recognition will be more consistent and errors more predictable, potentially enabling correction by the dialogue manager.

A working scenario is given below: these utterances can be recognized by ViaVoice, though not consistently, using the ordinary microphone on a Sharp Zaurus SL-5600. Speech recognition is not required to be perfect but it is required to reliably recognize keywords such as "mail" and obtain close approximations for phrases such as proper names.

```
SPA     <Displays initial PDA interface: Fig.7(a)>
User    Do I have any mail from Paul?
SPA     You have no messages from Paul.
User    Are there any emails from John?
SPA     You have two new messages from John Lloyd.
User    Show me the first message.
SPA     <Displays first message from John Lloyd: Fig.7(b)>
User    Any mail about conferences?
SPA     You have three new messages that contain conference.
        <Displays three message headers: from Byeong Ho
        Kang about CRC, from Terry Pomerantz about Research
        and from Supriya Singh about CRC: Fig.7(c)>
User    I want to see the one from Terry please.
SPA     <Displays message from Terry: Fig.7(d)>
```

## 5. RELATED WORK

There have been several research prototypes providing spoken dialogue interfaces for remote email management. In comparison to ours, most applications were designed to deal with a single modality and support only one-way transfer of information. Email Voice Interactive System (ELVIS) by Walker *et al.* [25] supports remote access to email over a fixed-line phone based on a state machine approach, in which, different strategies (e.g. system initiative) lead to different state transitions. The emphasis of this research is on learning the strategy selection from a training corpus of collected dialogues. The users can only interact with the system in a simple question-answer fashion. Another prototype developed at BT Laboratories in 1996, MailSec [26], allows users to access email by making a telephone call. The dialogue manager in MailSec implements a dialogue theory of Games Structure in Conversation, i.e. a conversation can be considered as a sequence of games and moves. It controls the flow of information by sequentially passing the data to and receiving the processed data from its modules. User utterances are parsed and interpreted strictly into an extended logic form. Hence we believe high quality speech recognition must be used.

The TRIPS system [4], as mentioned in Section 1, is also agent-based but distributed. That is, the system consists of different agent-based components acting asynchronously and communicating with each other by message passing. Among its components are the Interpretation Manager and the Generation Manager. An advantage of the approach is enhancing modularity. However, although the authors have taken into account the synchronization problem, there are issues that have not been resolved, such as controlling the concurrent actions of the Interpretation Manager and the Generation Manager when a user barges in while the system is talking. Hence for less complex domains, centralized approaches are more suitable because modularity can be facilitated through the use of plans and synchronization is manageable since agent execution is in controllable cycles.

Issues in supporting multi-modal interfaces have been addressed in work by McGlashan [16]. The proposed system provides a combination of two forms of communicative interfaces (graphical and speech modalities) in a consumer information service about microwave ovens. The user actions are interpreted as providing parameters for the requested information service. Thus graphical objects are also included for reference resolution. In case the system fails to interpret the user's request, menu-driven interaction style is adopted. This error-recovery method does not provide natural interaction as if the system could ask the user a clarifying question.

Research by Traum *et al.* [24] follows the framework of the TRINDI project which aims to model multi-modal dialogue for multiple participant interaction. In this work, the information state is divided into several layers dealing with different coherent aspects of dialogue, e.g. the contact layer provides information on the modalities that are accessible for communication. The interpretation processes at the layers are independent. An additional negotiation layer is introduced for handling multi-party dialogue, which is a central aspect of this application.

VERBMOBIL [20] is a system for speech-to-speech translation, in which statistical language modelling methods are employed for the prediction of the next dialogue act (by computing the probability of a sequence of dialogue acts). The dialogue grammar is trained from a corpus of 300 manually tagged spoken dialogues. These statistical approaches are not suitable for our application.

Finally, Paek and Horvitz [17] aim to build a probabilistic model (using Bayesian networks) of possible uncertainties at different levels of human-computer conversation. Thus the system would be able to identify actions that maximize the expected utility of achieving mutual understanding.

## 6. CONCLUSION

We have proposed an agent-based approach to dialogue management in personal assistants. This approach is particularly useful for applications where a single point of contact should be provided for users to interact with a collection of specialist assistants. The user can access to the system by using mobile device, e.g. PDA, and through different modalities, e.g. speech, text and/or graphical interface. The dialogue manager is implemented using a BDI agent framework with extensions for maintaining the conversational context, thus enabling coherent user-system dialogue to be conducted. Future work will be the integration of additional personal assistants to demonstrate the system's capability of coordinating the activities of the multiple agents. Different speech software will be considered for replacing IBM ViaVoice. In longer term, we aim to research on mechanisms for handling speech recognition errors.

## 7. ACKNOWLEDGEMENTS

This work was funded by the CRC for Smart Internet Technology. We would like to thank Agent Oriented Software Group for the use of their JACK agents platform. Many thanks to Waleed Kadous for supplying InCA and for his help in setting up the IBM ViaVoice speech engine. We especially acknowledge Van Ho for her contribution to the project, including the design of the overall SPA system architecture, and the implementation of the PDA graphical interface and the EMMA email management system.

## 8. REFERENCES

[1] J. Allen, D. Byron, and M. Dzilovska. Towards Conversational Human-Computer Interaction. *AI Magazine*, 22(4):27–37, 2001.

[2] H. Aust, M. Oerder, F. Seide, and V. Steinbiss. The Philips Automatic Train Timetable Information System. *Speech Communication*, 17(3-4):249–262, 1995.

[3] J. L. Austin. *How to Do Things with Words*. Clarendon Press, Oxford, 1962.

[4] N. Blaylock, J. Allen, and G. Ferguson. Synchronization in an Asynchronous Agent-Based Architecture for Dialogue Systems. In *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialog*, pages 1–10, 2002.

[5] D. N. Chin. Intelligent Interfaces as Agents. In J. W. Sullivan and S. W. Tyler, editors, *Intelligent User Interfaces*. ACM Press, New York, NY, 1991.

[6] J. Chu-Carroll. MIMIC: An Adaptive Mixed Initiative Spoken Dialogue System for Information Queries. In *Proceedings of the Sixth ACL Conference on Applied Natural Language Processing*, pages 97–104, 2000.

[7] H. H. Clark. *Using Language*. Cambridge University Press, Cambridge, 1996.

[8] A. N. De Roeck, U. Kruschwitz, P. Scott, S. Steel, R. Turner, and N. Webb. YPA - An Assistant for Classified Directory Enquiries. In B. Azvine, N. Azarmi, and D. Nauck, editors, *Intelligent Systems and Soft Computing*, pages 239–258. Springer-Verlag, Berlin, 2000.

[9] 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.

[10] B. J. Grosz. Attention, Intentions, and the Structure of Discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[11] V. Ho, W. Wobcke, and P. Compton. EMMA: An E-Mail Management Assistant. In *Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology*, pages 67–74, 2003.

[12] E. Horvitz. Principles of Mixed-Initiative User Interfaces. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 159–166, 1999.

[13] N. Howden, R. Rönnquist, A. Hodgson, and A. Lucas. JACK Intelligent Agents$^{TM}$– Summary of an Agent Infrastructure. *Paper presented at the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, 2001.

[14] M. W. Kadous and C. Sammut. Mobile Conversational Character. In *Proceedings of the Human Factors Workshop on Virtual Conversational Characters*, 2002.

[15] S. Larsson and D. Traum. Information State and Dialogue Management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering*, 6:323–340, 2000.

[16] S. McGlashan. Towards Multimodal Dialogue Management. In *Proceedings of the Eleventh Twente Workshop on Lanugage Technology*, pages 13–22, 1996.

[17] T. Paek and E. Horvitz. Conversation as Action Under Uncertainty. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 455–464, 2000.

[18] B. Pellom, W. Ward, J. Hansen, K. Hacioglu, J. Zhang, X. Yu, and S. Pradhan. University of Colorado Dialog Systems for Travel and Navigation. In *Proceedings of the 2001 Human Language Technology Conference*, 2001.

[19] A. S. Rao and M. P. Georgeff. BDI-Agents: From Theory to Practice. In *Proceedings of the First International Conference on Multiagent Systems*, pages 312–319, 1995.

[20] N. Reithinger, R. Engel, M. Kipp, and M. Klesen. Predicting Dialogue Acts for a Speech-To-Speech Translation System. In *Proceedings of the Fourth International Conference on Spoken Language Processing*, pages 654–657, 1996.

[21] C. Sammut. Managing Context in a Conversational Agent. *Linköping Electronic Articles in Computer and Information Science*, 3(7):1–13, 2001.

[22] J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, 1969.

[23] S. Seneff and J. Polifroni. Dialogue Management in the Mercury Flight Reservation System. In *Proceedings of ANLP-NAACL Workshop on Satellite Dialogue*, pages 1–6, 2000.

[24] D. Traum, J. Rickel, J. Gratch, and S. Marsella. Negotiation Over Tasks in Hybrid Human-Agent Teams for Simulation-Based Training. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 441–448, 2003.

[25] M. Walker, J. Fromer, and S. Narayanan. Learning Optimal Dialogue Strategies: A Case Study of a Spoken Dialogue Agent for Email. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 1345–1351, 1998.

[26] S. Williams. Dialogue Management in a Mixed-Initiative, Cooperative, Spoken Language System. In *Proceedings of the Eleventh Twente Workshop on Language Technology*, pages 199–208, 1996.

[27] M. J. Wooldridge and N. R. Jennings. Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

[28] W. Xu and A. Rudnicky. Task-Based Dialog Management Using an Agenda. In *Proceedings of the ANLP/NAACL 2000 Workshop on Conversational Systems*, pages 42–47, 2000.