

A BDI Agent Architecture for Dialogue Modelling and Coordination in a Smart Personal Assistant

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

Abstract

In this paper, we discuss the architectural aspects of a Smart Personal Assistant (SPA) system that enables users to access a range of applications from a range of devices using multi-modal natural language dialogue. Each back-end application is a personal assistant specializing in one specific task such as e-mail or calendar management, and typically each has its own user model, enabling it to adapt to the user's changing preferences. The PDA interface to the SPA must present the system as a single unified set of back-end applications, enabling the user to conduct a dialogue in which it is easy to switch between these applications. Furthermore, the system's interaction with the user must be tailored to their current device. The SPA is implemented using an agent platform and includes a special BDI coordinator agent with plans both for coordinating the actions of the individual assistants and for encoding the system's dialogue model. The plan-based dialogue model is at a high level of abstraction, enabling the domain-independent plans in the dialogue model to be reused in different SPA systems.

1. Introduction

The notion of a personal assistant was promoted by Maes [8] as a way of coping with information and communication overload. The idea was that a software agent should collaborate with the user, perform tasks on the user's behalf and hide task complexity, learn the user's interests, and adapt to the user's changing preferences in performing some specific tasks. A successful personal assistant needs *competence* (ease of acquiring the knowledge required to perform tasks) and *trust* (willingness of users to delegate tasks to the system). To these we can add *usability* (the ease of interaction between the user and the assistant, including the ease with which knowledge is acquired by the assistant).

Our research is focused on personal assistant systems for use on mobile devices such as PDAs and mobile phones that address the problems of communication and time management. The intended user is an office worker who needs to keep in contact while travelling. We aim to develop a Smart Personal Assistant (SPA) that combines a number of task-specific assistants into a single system that the user can access either from a desktop computer using a conventional graphical interface or remotely using a multi-modal interface. Since PDAs and phones have limited graphical capability, of much interest is the use of speech for input and output, which has become feasible with recent advances in speech recognition technology. Also of interest is the personalization of services and the context-sensitive presentation of information.

A SPA application typically includes multi-modal user interfaces to a collection of specialized assistants that the user can access using range of devices. The interfaces should be independent of the "back-end" functions of the assistants so that regardless of the device, the user interacts with the same set of applications. However, the style of interaction with the SPA should be tailored to the user's current device and context (e.g. for a PDA, presenting large quantities of information should be avoided and speech input and output may be preferred). Moreover, the SPA architecture should be general in that the same architecture should be able to be employed for different SPA systems (different collections of specialist task assistants).

Our current SPA application is a personal information management assistant that provides users with integrated access to e-mail and calendar information. The e-mail assistant, which has been described in Ho, Wobcke and Compton [5], enables users to easily define a rule base that enables messages to be associated with a sorting folder for display, a priority and/or an action to be taken. The calendar assistant enables users to define rules for suggesting various attributes of appointments. The SPA must present a unified interface to the user so that they can switch seamlessly

between the different applications; ideally, the user should simply issue requests to the SPA in natural language and the system should determine how to process those requests. Thus processing the user's requests requires *coordination* of a collection of specialist task assistants (especially when more than one assistant is needed to fulfil a user's request), and *dialogue modelling* since the SPA will need to maintain a coherent conversation with the user and tailor responses to the user's current device. In addition, though speech recognition systems have greatly improved in recent years, performance is still not perfect so the dialogue model must include mechanisms for recovering from speech recognition errors.

In this paper, we describe our rationale for the design of the SPA architecture as a multi-agent system with a special Coordinator agent based on a BDI architecture, and discuss how coordination and dialogue management are implemented in the Coordinator using plans. The discussion of our agent architecture is motivated in Section 2 by comparison with the alternative "blackboard" style approach used in the Open Agent Architecture [9] and the Automated Office demonstrator built using this platform, Cohen *et al.* [3]. In Section 3, the system architecture of the SPA is described, while Section 4 focuses on the internal architecture of the Coordinator, including the dialogue model and the use of plans for coordination (an earlier version of the dialogue manager handling only the e-mail domain was described in Nguyen and Wobcke [10]). A sample scenario is presented in Section 5, and related work is described in Section 6.

2. Motivation

There are two broad approaches to building systems of autonomous agents, differing in how the overall control of the system is specified. One approach is to use a "blackboard" mechanism where a global data store (the blackboard) is used for sharing information. The only means of communication between agents is by reading and writing (posting events) to the blackboard. This is the approach taken in the Open Agent Architecture (OAA) [9]. In this approach, some control flow is specified in a special agent, the facilitator (basically a problem solving agent that responds to events taking the form of goals to be achieved by reducing those goals to collections of subgoals), but otherwise the control flow of the system "emerges" from the interactions between agents through the blackboard.

There are various ways that dialogue management can be implemented within the OAA. One solution, exemplified by the Automated Office demonstrator, originally described in Cohen *et al.* [3], is to use specialized speech recognition and natural language agents that perform the tasks of converting speech to text and converting text to logical form (respectively). The logical form of a user request is a goal that the

facilitator can decompose, and when the final result is provided on the blackboard by the relevant task-specific agents, other notification or user interface agents can present the information to the user in a suitable form. However, because the dialogue model is highly distributed, it is difficult to specify the overall system behaviour to guarantee robust behaviour, especially the dialogue action(s) to be taken on failure of any one of the agents.

For applications such as the SPA, therefore, more explicit control flow information is needed, so that the system is robust with respect to a variety of types of failure. Therefore to implement the SPA, we use an alternative agent-based architecture with a special Coordinator agent that mediates communication between the user and the specialist task assistants, and which is responsible for handling both coordination of these assistants and dialogue management. The Coordinator is based on a BDI (Belief, Desire, Intention) agent architecture in which the dialogue model is encoded as a set of plans, and in which coordination is also achieved by using plans. Thus both dialogue management and coordination are treated uniformly in the agent architecture as instances of goal-directed activity.

The role of the Coordinator is to present a single point of contact for the user to interact with the SPA, to maintain the dialogue context (both information about the physical context of the user and the dialogue history), to route commands to the relevant task assistants to fulfil a user's request, to notify the user of any important events, and to learn the user's preferences for interaction on particular device types.

The main advantage of using a centralized coordinator agent for both dialogue modelling and coordination is that it enables a common framework for expressing problem solving and dialogue plans, avoiding the problem of having to control the interaction between otherwise separate agents. Concerning dialogue management, a BDI architecture offers the advantages of a complex but modular approach to designing dialogue systems, in that the plans used by the BDI agent are independent chunks of knowledge associated with particular communicative goals. The BDI approach makes it easy to extend the agent's dialogue model as expressed in the set of plans, and also enables learning to be easily added to the agent in a general way but for specific learning tasks. Finally, the modularity of the plan-based approach enables domain-independent discourse-level plans to be reused for different applications.

3. SPA Agent Architecture

In this section, we describe the design of the SPA system architecture in more detail. As indicated above, the architecture is based on having a set of agents including a special Coordinator agent and agents for each of the back-end ap-

plications (specialist assistants such as e-mail and calendar assistants) that communicate using message passing. The SPA system is built using the JACK Intelligent Agents™ platform [6], which enables point-to-point communication between agents that run on different machines to be encoded at a high level of abstraction.

The user interacts with the current version of the SPA using a desktop computer with a standard graphical user interface or using a PDA over a wireless network using a specially designed interface capable of invoking speech recognition and generation as well as interpreting actions on the graphical interface (such as selecting a particular e-mail message or highlighting a portion of text). The SPA system consists of two main clusters of components as shown in Figure 1: user interfaces (SPA clients) operate on a PDA or a desktop, while the SPA itself (SPA server) runs on desktop machine(s). The SPA client for the PDA is a lightweight component that can operate on handheld devices with limited resources: all major processes are run on the SPA server, including speech processing, partial parsing, dialogue management, and the execution of plans by back-end assistants in fulfilment of the user's requests. The SPA client communicates with the SPA server components through JACK message passing.

The SPA includes a number of JACK agents, including a User Interaction Agent to handle interaction with the user and provide an interface to the speech recognition and partial parsing procedures, the Coordinator agent that incorporates a plan-based dialogue model and plans for handling user requests by coordinating the specialist assistants, and a set of agents that are “wrappers” for the back-end assistants to enable them to communicate with the Coordinator using JACK message passing. An overview of the system of agents and their communication links are shown in Figure 1. Although all the agents are implemented using JACK and hence are capable of decision making in the interpretation cycle, only the Coordinator makes significant use of this reasoning capability.

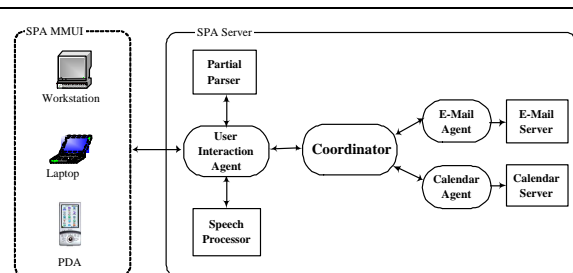


Figure 1. SPA Agent Architecture

The role of the User Interaction Agent is to establish and maintain the connection between the user interface on the

device (e.g. the PDA) and the Coordinator (and through the Coordinator, the set of specialist assistants). Typically the User Interaction Agent runs on the same machine as the Coordinator, though the speech services it uses may run on different machines. When the user initiates a spoken utterance, the speech processor and partial parser are called and the User Interaction Agent receives the partially parsed input, from which it creates a JACK message for communication to the Coordinator that represents the logical form of the utterance. The User Interaction Agent also handles the conversion of the system's reply to a format (in XML) that can be used by the interface on the device to present information on the graphical display and/or speak out some response (in this case, the speech processor is called to convert the textual response of the Coordinator into speech).

The current version of the User Interaction Agent is based on InCA, the Internet Conversational Agent developed by Kadous and Sammut [7], which employs Dragon NaturallySpeaking in dictation mode for speech recognition and Lernout and Hauspie TTS for speech synthesis. The User Interaction Agent also uses the ProBot scripting language of Sammut [12] for computing a partial syntactic analysis of the input. Due to the modularity of the architecture, it is straightforward to replace both the speech processor and partial parser by other equivalent systems.

The core of the SPA is the Coordinator that is responsible not only for coordinating the actions of the task-specific assistants in fulfilling a user's requests, but also for maintaining a coherent dialogue between the user and the system. To support coordination, the Coordinator has a series of simple plans that represent the “capabilities” of each individual specialist assistant, and a series of more complex plans that are used to “coordinate” multiple specialist assistants in achieving more complex goals. The Coordinator also has a set of plans that form its dialogue model. As the system is run, the Coordinator maintains knowledge of the dialogue context in its belief state, and uses the plans in the dialogue model to determine the user's communicative acts, identify the user's intentions, satisfy the user's goals and generate appropriate responses.

The other agents in the system are task-specific agents that provide wrappers around specific back-end applications such as e-mail and calendar assistants. The E-Mail Agent in the current implementation provides a wrapper for our existing E-Mail Management Assistant (EMMA), Ho, Wobcke and Compton [5]. EMMA uses a rule base defined by the user to classify messages into sorting folders, prioritize messages and suggest actions to be taken. In the SPA architecture, the E-Mail Agent provides a set of services for manipulating the user's mail box through requests sent by the Coordinator, such as searching for messages, archiving and deleting messages, and retrieving new messages from the mail server, while the Calendar Agent pro-

vides services such as creating and viewing appointments. The plans used by these task-specific agents are fairly simple, and these agents do not require the decision making procedures used by the Coordinator.

4. Dialogue Management and Coordination

The two main aspects addressed in design of the SPA Coordinator are (i) the coordination of a variety of specialist personal assistants, and (ii) the conduct of a coherent natural language dialogue with the user. As described above, the SPA Coordinator is a single JACK BDI agent that incorporates a plan-based dialogue model and plans for coordinating the actions of the set of individual personal assistants. Under the BDI interpretation cycle, the execution of these plans are tightly integrated, hence the coordination aspects of the SPA are best described together with a discussion of dialogue management. As argued above, even though this approach combines dialogue modelling and problem solving within a single agent, modularity is preserved at the level of plans, i.e. the plans for dialogue management are separate from those for coordination, thus enabling the domain-independent plans in the dialogue model to be reused for other SPA applications.

4.1. Dialogue Model

The dialogue model is based on the theory of speech acts developed by Searle [13]. The speaker in a dialogue, by making an utterance, intends to perform an action known as an *illocutionary act* (or speech act). The intention of the speaker can be identified from the speech acts being performed. In addition, computational work in discourse by Grosz and Sidner [4] has argued that discourse structure is composed of the linguistic structure, the intentional structure and the state of focus of attention. Utterances in a discourse naturally form a hierarchy of discourse segments, each of which has a corresponding intention. Discourse segment intentions differ from other kinds of intentions in that they are intended to be recognized. The attentional state 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's intention. We call this act a conversational act; the conversational acts used by the SPA are as defined in Table 1. While carrying out domain tasks, the Coordinator also performs conversational acts with other personal assistants. However, we separate the domain-independent conversational acts from the domain-dependent task goals.

In the SPA, the dialogue data structure constitutes the agent's beliefs. The Coordinator's internal belief state con-

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/feelings
Confirm	clarify ambiguities by expressing agreement/disagreement
Ack	express an acknowledgement

Table 1. Conversational Act Descriptions

Greeting	ID=122, Sender="User", Receiver="Coor", Subject="OpenConversation"
Request	ID=123, IsReplacedBy=126, Sender="User", Receiver="Coor", Task=SEARCH, Object=EMAIL, Condition={From="John"}
Clarify	ID=124, OriginateFrom=123, Sender="Coor", Receiver="User", Subject=(From="John Lloyd" or From="John McAfee")
Confirm	ID=125, InResponseTo=124, Sender="User", Receiver="Coor", Subject=(From="John Lloyd")
Request	ID=126, AsReplaceFor=123, Sender="User", Receiver="Coor", Task=SEARCH, Object=EMAIL, Condition={From="John Lloyd"}
Request	ID=127, InResponseTo=126, Sender="Coor", Receiver="Emailer", Task=SEARCH, Object=EMAIL, Condition={From="John Lloyd"}
Respond	ID=128, InResponseTo=127, Sender="Emailer", Receiver="Coor", Result=(MailId=...)
...

Table 2. Sample Discourse History

tains specific domain knowledge used for interpreting user utterances, such as a domain-specific term dictionary, together with a *discourse history* and a *focus stack* for maintaining discourse context. The discourse history keeps all the interactions as a stack of conversational acts. It also keeps track of the user intention being carried out in the current discourse segment. In many cases, the user intention cannot be identified 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 the underlying intentions. Table 2 shows an example discourse history.

The focus stack is a data structure for tracking objects that have been mentioned during the course of conversation. This information is used to resolve references, i.e. when certain words are used in the current utterance to refer back to other referring expressions in previous utterances. The focus stack is also used in generating context-sensitive natural

language responses. As an example, the information about e-mail stored in the focus stack includes e-mails, folder names, personal names and key phrases. As the user interacts with the Coordinator through dialogue as well as the graphical interface on mobile devices, objects involved in the interaction on the device (such as highlighted text) are also put on to the focus stack.

Dialogue (spoken and written) differs from other kinds of discourse in some characteristics such as turn-taking and grounding. In our model, the user and the Coordinator alternately take turns in the conversation. The user takes turn to request tasks or response to a confirmation while the Coordinator takes turn to request clarification or provide the task results. Grounding is commonly understood as a process in which the speaker and the listener constantly establish common ground (i.e. mutual beliefs). In our model, this is done by using `Ack` and `Confirm` acts for acknowledgement and `Clarify` for specifying problems.

4.2. Dialogue Plans and Coordination

The Coordinator uses a set of plans that specify the actions to be performed in order to achieve its goals. These plans are divided into four groups according to their purpose: conversational act determination and domain task classification (CAD group), intention identification (II group), task processing (TP group) and response generation (RG group). Each plan instance is relatively short-lived, lasting only for the duration of the system's turn, updating the dialogue history and focus stack rather than suspending execution for resumption on the next turn. This method of encoding plans is chosen because of the dynamic nature of the dialogue; the SPA can in this way respond more directly to the user's inputs rather than relying on its plans to predict the user's responses.

Input from the user is processed as follows (there is no need to explicitly specify the control flow; this behaviour is the automatic result of running these plans under the JACK BDI interpretation cycle). First, the user's input (graphical actions and parsed text) is handled by the plans in CAD group to determine the conversational act and identify the relevant task domain. Then, the user's specific intention is 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 specialist agents. Otherwise, a clarification act is generated by the RG group. When a task is successfully processed, the plans in the RG group generate multi-modal output including natural language text and information to be displayed or spoken out on the user's device. This organization of the plans in the Coordinator is as shown in Figure 2.

Perhaps the most complex of these plan groups is that for intention identification. In this group, there is one

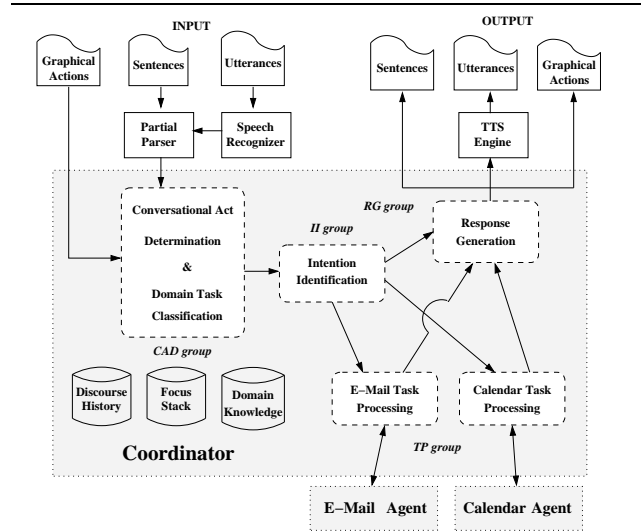


Figure 2. Coordinator Agent Plans

main `IntentionIdentification` plan with a number of sub-plans. The `IntentionIdentification` plan can activate any of its subplans, namely `EmailInterpretation`, `FolderInterpretation`, `AppointmentInterpretation`, `ToDoInterpretation`, `PeopleInterpretation`, `ReferenceResolution` and `GraphicalActionInterpretation`. The goal of these plans is to determine the attributes of objects (e-mails, folders, to-do items, etc.) mentioned by the user. These subplans can send messages to other personal assistants seeking information required to complete the interpretation. For example, in interpreting a proper name, the `PeopleInterpretation` plan can request address book entries from the E-Mail Agent (see the example scenario below). The `ReferenceResolution` plan is designed to handle the interpretation of anaphora, i.e. the coreference 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. This plan results in the Coordinator updating its belief state by pushing the phrase on to the corresponding focus stack.

A plan for determining an attribute of an e-mail message is shown in Figure 3. Diamonds in the diagram denote decision points, rectangles indicate calls to procedures for data manipulation, and the curved rectangles are subplans. There are three special points in the plan: Start, End and Fail indicating the beginning, end and failure of plan execution. The plan first checks the previous conversational act; if this was a `Clarify` act, the plan takes the user's response

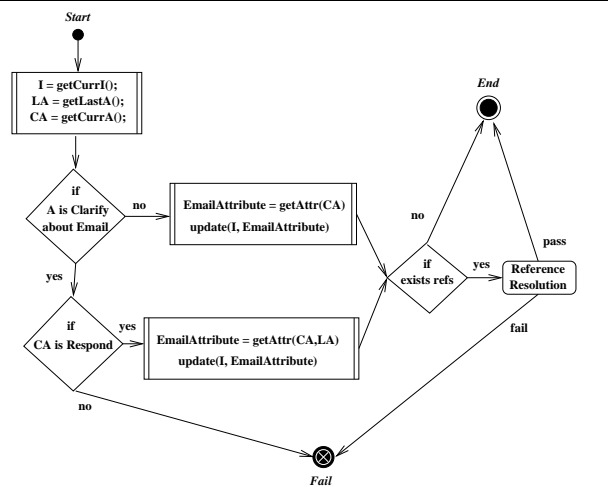


Figure 3. EmailInterpretation Plan

to update the e-mail attribute, otherwise the plan takes the user's input directly to interpret the attribute. Then if references need to be resolved, the ReferenceResolution plan is invoked, which may cause this entire plan to fail. When the plan fails, it triggers the execution of the Clarification-Generation plan.

5. Example Scenario

In this section, an example is briefly presented to illustrate the functioning of the SPA architecture in a scenario where the user engages in a conversation using a PDA to check e-mail and create an appointment. Actions of the SPA corresponding to points in the dialogue are written in italics and relate to the screen shots displayed in Figure 4.

```

User  Is there any new mail from John?
SPA  You have two new messages from John Lloyd.
     Displays two message headers: Fig. 4(a)
User  Show me the one about slides please.
SPA  Displays correct message from John Lloyd: Fig. 4(b)
User  I need to see him at 5 pm this Friday about
     workshop 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.
     Displays calendar showing new appointment: Fig. 4(c)
User  Anything from Paul?
SPA  You have no new messages from Paul.
User  Notify me if I have mail from him.
SPA  Sure.
User  Show me appointments with him next week.
SPA  You will meet Paul Compton at 11 am next Tuesday.
     Displays calendar showing the appointment: Fig. 4(d)
SPA  You just have a new message from Paul Compton.
  
```

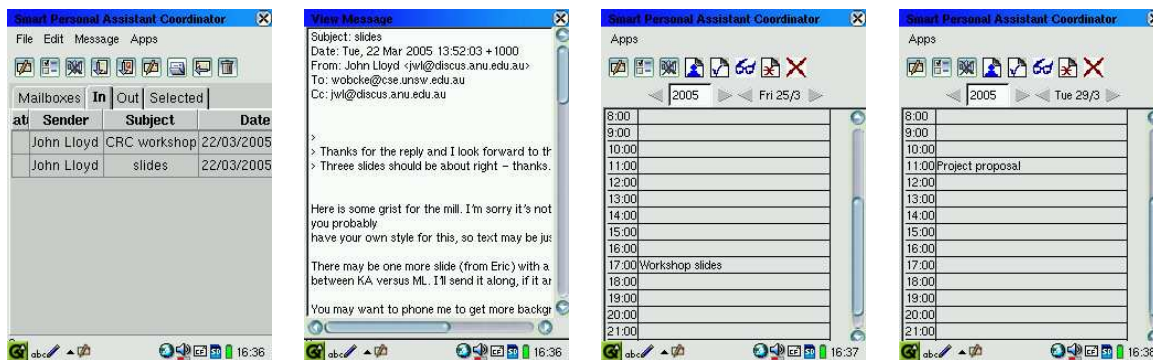
Notice particularly how with the query *Is there any new mail from John?*, information from the E-Mail Agent is used to determine the appropriate response. The Coordinator has determined that there are messages from only one *John* in

the Inbox, so generates a response that states the fact that there are two such messages, in the process further grounding the term *John* as *John Lloyd*. Here the user accepts this grounding and continues the dialogue; an alternative continuation could have been for the user to reject this interpretation, e.g. *No, I meant John McAfee*. If there are messages in the Inbox from two different people named *John*, the system seeks clarification from the user by asking which *John* is meant, e.g. *Do you mean John Lloyd or John McAfee?*. In the response to the next request, *Show me the one about slides please*, the SPA has used the current context to identify the correct message and display its content on the interface. The user then changes the conversational context from e-mail to the calendar domain, suggesting a new appointment. After the new appointment has been successfully created, the conversation is shifted back to the e-mail domain. The coordination mechanism of the SPA allows the user to easily switch contexts between assistants while still being in a continuing conversation with the whole system.

6. Related Work

The Open Agent Architecture (OAA) [9] and the ZEUS agent platform, Nwana *et al.* [11], provide frameworks for building multi-agent applications such as the SPA, as has been done for OAA with the Automated Office demonstrator, Cohen *et al.* [3], and with the Intelligent Assistant, Azvine *et al.* [1]. Another related application is Electric Elves, Chalupsky *et al.* [2]. The Electric Elves system is a team of heterogeneous agents, including humans (via proxy agents), that support coordination activities in organizations such as scheduling meetings and nominating speakers for regular presentations. Electric Elves focuses on coordinating the activities of users through their proxy software agents rather than on coordinating multiple applications through dialogue, and while users interact with the system using PDAs, there is comparatively little attention paid to the usability of the system.

Finally, a similar application to the SPA that provides a speech interface to a combined e-mail and calendar system on a desktop has recently been described by Sidner [14]. That system was developed using the Collagen platform, which is based on the metaphor of collaborative problem solving between user and system. The Collagen approach to dialogue management is to use a general plan recognition algorithm to interpret the user's utterances with reference to a library of domain-specific plans, the discourse state consisting of a partially executed hierarchical task plan. Thus the only plans used with Collagen are task plans; there is no notion of treating dialogue management as the rational selection of dialogue plans, as in our model.



(a)

(b)

(c)

(d)

Figure 4. PDA Displays for Sample Dialogue

7. Conclusion

We have described the architecture and dialogue model for a Smart Personal Assistant application that addresses coordination of a range of specialist assistants and natural language interaction tailored to the user's devices. The architecture is a BDI architecture with plans for coordination and dialogue actions, and point-to-point communication between agents. The plan-based approach provides a high degree of modularity, making the domain-independent aspects of the dialogue model reusable across applications. The agent-based approach also makes it easy to integrate learning into the system so that the user's preferences for interaction can be acquired automatically.

Acknowledgements

This work was funded by the CRC for Smart Internet Technology. We would like to thank Agent Oriented Software Pty. Ltd. for the use of their JACK agents platform. Many thanks also to Waleed Kadous for supplying InCA and for helping us to integrate InCA into the SPA system.

References

- [1] B. Azvine, D. Djian, K. C. Tsui and W. R. Wobcke. The Intelligent Assistant: An Overview. In B. Azvine, N. Azarmi and D. D. Nauck, editors, *Intelligent Systems and Soft Computing*, pages 215–238. Springer-Verlag, Berlin, 2000.
- [2] H. Chalupsky, Y. Gil, G. A. Knoblock, K. Lerman, J. Oh, D. V. Pynadath, T. A. Russ and M. Tambe. Electric Elves: Agent Technology for Supporting Human Organizations. *AI Magazine*, 23(2):11–24, 2002.
- [3] P. R. Cohen, A. J. Cheyer, M. Wang and S. C. Baeg. An Open Agent Architecture. In *Software Agents: Papers from the 1994 Spring Symposium*, pages 1–8, March 1994.
- [4] B. J. Grosz and C. L. Sidner. Attention, Intentions, and the Structure of Discourse. *Computational Linguistics*, 12:175–204, 1986.
- [5] V. H. Ho, W. R. Wobcke and P. J. Compton. EMMA: An E-Mail Management Assistant. In *Proceedings of the 2003 IEEE/WIC International Conference on Intelligent Agent Technology*, pages 67–74, 2003.
- [6] N. Howden, R. Rönquist, A. Hodgson and A. Lucas. JACK Intelligent Agents™ – Summary of an Agent Infrastructure. Paper presented at the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS, May 29, 2001.
- [7] M. W. Kadous and C. A. Sammut. InCA: A Mobile Conversational Agent. In C. Zhang, H. W. Guesgen and W. K. Yeap, editors, *PRICAI 2004: Trends in Artificial Intelligence*, pages 644–653. Springer-Verlag, Berlin, 2004.
- [8] P. Maes. Agents that Reduce Work and Information Overload. *Communications of the ACM*, 37(7):31–40, 1994.
- [9] D. L. Martin, A. J. Cheyer and D. B. Moran. The Open Agent Architecture: A Framework for Building Distributed Software Systems. *Applied Artificial Intelligence*, 13:91–128, 1999.
- [10] A. Nguyen and W. R. 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.
- [11] H. S. Nwana, D. T. Ndumu, L. C. Lee and J. C. Collis. ZEUS: A Toolkit for Building Distributed Multiagent Systems. *Applied Artificial Intelligence*, 13:129–185, 1999.
- [12] C. A. Sammut. Managing Context in a Conversational Agent. *Electronic Transactions on Artificial Intelligence*, 5(B):189–202, 2001.
- [13] J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, 1969.
- [14] C. L. Sidner. Building Spoken-Language Collaborative Interface Agents. In D. Dahl, editor, *Practical Spoken Dialog Systems*, pages 197–226. Kluwer, Dordrecht, 2004.