

# TOWARDS MULTIMODAL DIALOGUE MANAGEMENT

Scott McGlashan  
Swedish Institute of Computer Science  
Box 1263, S-164 28 Kista, Sweden  
*E-mail* scott@sics.se

## ABSTRACT

Effective dialogue management is a key issue in speech-based interfaces to information systems since it can ensure a cooperative interaction with the user. Cooperativeness requires techniques which allow the user to efficiently access information and also techniques which compensate for limitations in system knowledge and speech technology. The paper describes management techniques developed in a speech-only dialogue system and how they are being extended for a multimodal system which combines a direct manipulation interface with a spoken dialogue interface for a simple consumer information service.

## 1 INTRODUCTION

This paper describes dialogue management techniques which have been developed for spoken dialogue systems, and how these techniques are being used in a multimodal system which combines a speech interface with a graphical user interface. Our guiding principle is that dialogue is a joint activity in which user input is interpreted as instructions how to update the system's model of the evolving dialogue, and that system output transparently reflects the state of this model. The model itself encodes techniques for contextual semantic interpretation, and dialogue strategies to handle clarification, confirmation and failure-repair in manner appropriate to dialogue progression. The approach is being extended to incorporate non-speech input and output, references to visual objects, output modality selection, and adaptive navigation. This will allow us to investigate the tension between speech and graphical modalities in a system providing a simple consumer information service.

## 2 SPOKEN DIALOGUE MANAGEMENT

Over the last five years, spoken language dialogue systems have emerged for a variety of task domains including spatial navigation (Voyager), travel planning (Waxholm), and speech translation (Verbmobil). One particular area of interest is telephone-based dialogue systems which provide access to simple information services such train timetable and flight enquiries (Peckham 1993; Aust and Oerder 1995). In these systems, "the customer or client identify certain entities to the person providing the service; these entities are parameters of the service, and once they are identified the service can be provided" (Hayes and Reddy 1983: 252). A simple dialogue from the domain of flight enquiries illustrates this type of service.

- (1) S1: Welcome to British Airways flight enquiries service. How can I help you?  
U1: Can you tell me the arrival time of BA777 from Stockholm?  
S2: BA777 from Stockholm?  
U2: Yes.  
S3: BA777 leaves Stockholm at 11.17 and arrives London Heathrow terminal 1 at 13.41. That's BA777 arriving London Heathrow terminal 1 at 13.41. Do you have another enquiry?  
U3: No.  
S3: Thank you for calling. Goodbye.

The user provides parameters (the flight identifier, departure city and a request for the arrival time) and the system then provides the flight enquiry service.

These dialogue systems are inevitably compared by users to human agents offering a similar service. By analyzing human-human and (simulated) human-computer dialogues, we find that while users expect reduced linguistic competence from a system (and reflect this in

a simplification of their own linguistic behaviour), they still expect the system to retain many of the characteristics of a human service agent, whilst compensating for its own performance limitations (Giachin and McGlashan 1996; Wooffitt et al. forthcoming). Some of the main dialogue characteristics with systems include:

**global structure** the dialogue has an opening, a body and a closing. In the body, the system takes responsibility for obtaining information

**mixed initiative** while the user generally takes the initiative (such as providing task information or asking for repetition), the system can take the initiative to confirm information has been correctly understood, obtain information necessary for database access and, if the dialogue deteriorates, to constrain the form and content of user utterances (for example, by means of closed questions)

**over-informativeness** users may provide over-informative answers; for example, instead of U2, the user might have added the departure time as with *Yes, it left Stockholm late morning*

**contextual interpretation** user input may only provide partial and ambiguous information whose interpretation needs to be established in the discourse context; for example the utterance, *from Stockholm*, would function as a repetition in U2 of (1), but as a modification following *BA777 from Scunthorpe?*

**failure-repair** users expect the system to repair failures which may arise from performance limitations, especially speech recognition, as well as limitations in system knowledge — linguistic knowledge, semantic knowledge, task knowledge, and dialogue knowledge. The system needs to adopt pre-emptive strategies to avoid dialogue failure — such as confirming task information as in S2 — and reactive strategies to deal with failures when they do arise — such as asking for parameters to be spelt.

In the following sections we describe our techniques for addressing these characteristics. Other information services may exhibit different dialogue characteristics and so may require other dialogue management techniques (Bernsen et al. 1994).

## 2.1 BASIC PRINCIPLES

A minimal requirement is that the system plays the role of a *co-operative* agent so that the resulting interaction is comfortable, comprehensive and comprehensible to the user. From the user's point of view, whether the interaction is co-operative or not is judged solely on the basis of what the system says. Even if difficulties arise in the interaction, the system should still provide a response which does not lead to dialogue failure: while the goal of the interaction — providing the service — may fail, the dialogue *per se* should not (McGlashan et al. 1992). To achieve this, both interpretation of user utterances and production of system utterances must be informed by past and current states of the interaction. Co-operative dialogue management, therefore, requires the construction and maintenance of an interactional model: i.e. a model which specifies the layers of structure which can be distinguished in dialogue interactions. We distinguish linguistic structure, attentional (or discourse) structure, and intentional structure. Intentional structure is further differentiated into dialogue structure and task structure (Bunt 1989).

Of these structural layers, the characterization of the intentional is the most contentious. Three approaches have been distinguished (Cohen 1995): those based on dialogue grammars, those based on plans and intentions, and those, like ours, which treat dialogue as a joint activity where cooperating agents evolve a common model of the discourse situation. Our approach is based on the following general principles (Giachin and McGlashan 1996; Heisterkamp and McGlashan 1996):

1. Only the system's goals are explicitly represented in the dialogue model: user utterances are not assigned dialogue acts.
2. Only local transitions are modelled: the dialogue as a whole is not modelled, but global structure can still emerge.
3. Task-level information in user utterances is assigned a semantic function indicating its effects on the accessible part of the discourse model. Pragmatic functions are assigned on the basis of surface properties, including discourse markers.
4. These functions are applied to goals in the current dialogue model: they may satisfy a goal, modify it, or introduce another one. The results are then evaluated to determine which

goals provides an optimal continuation of the dialogue.

5. The system reports these goals to the user. The user is thus able to verify the system's model against their own interactional model. If verification fails, the user has an opportunity to make the problem explicit and correct it, so forestalling more serious problems which lead to irreparable breakdowns in the dialogue.

This approach is realized in a dialogue manager where each structural layer is represented in a separate model and each model, together with maintenance and update routines, is encapsulated in a semi-autonomous software module: a linguistic interface, semantics module, task module, and dialogue module. When the dialogue manager is ready to process user input, the linguistic interface calls the parser with a set of predictions, and communicates the result of recognition and parsing to the dialogue module. The parser result is either a semantic representation of the user turn, or an error message. After semantic, task and dialogue processing, a set of goals is selected as the dialogue continuation. The linguistic interface passes these to a linguistic generator for linguistic processing and synthesis.

## 2.2 SEMANTIC TECHNIQUES

The primary function of the semantics module is to interpret the representation of user turns with respect to the discourse model and assign semantic functions to each task parameter in the user input.

Input from two types of parsers is supported: those which provide a compositional semantic representation, such as Unification Categorical Grammar; and those which produce frame-based semantic representation, such as phrase-spotting and finite-state grammar approaches. Compositional semantic input is reduced to a frame representation by application of inference rules: assignment of semantic function (as well as database access) in simple information services only requires a limited set of concepts extracted from user input (cf. translation applications). The frame representation consists of instantiated concepts (with 'modus' features such as definiteness and number) organized in an inheritance hierarchy. The hierarchy includes a *meta-level* and an *object-level*: the former describe information about the dialogue including discourse-level actions such *repeat* and *open*; while the latter principally refers to ac-

tions and concepts (such as *request* and *flight*) in the task domain. The representation for *Can you tell me the arrival time of BA777 from Stockholm* is illustrated in (2).

$$(2) \quad \left\langle \left[ \begin{array}{l} \text{type : request} \\ \text{value : } \left[ \begin{array}{l} \text{type : flight} \\ \text{flightid : ba777} \\ \text{sourcecity : stockholm} \\ \text{goaltime : '?'} \end{array} \right] \end{array} \right] \right\rangle$$

Analysis of the turn consists of a single object-level representation of the type *request* whose value is a *flight* concept with two instantiated parameters and one requested parameter.

Object-level descriptions are then added to the discourse model and their semantic functions assigned; meta-level descriptions can be directly given over to dialogue interpretation since their pragmatic function is already specified. The discourse model is composed of an ordered set of discourse states. Each state is defined as a structure of the type

$$ID \times OWNER \times TYPE \times OBJECT \times PARAMETERS \times FUNCTIONS$$

where *TYPE*, *OBJECT* and *PARAMETERS* are extracted from the description and *OWNER* indicates whether the utterance was produced by the system or user. These states are ordered by recency. The semantics functions are then assigned by parameter-wise comparison with the most accessible, compatible concept; accessibility is simply based on recency, and compatibility depends on the type and modus properties of concepts. The set of semantic functions is shown in Table 1.

| Function  | Interpretation  |
|-----------|---|
| new       | a new parameter has been introduced by the user                   |
| modified  | the user has given an alternative value for an existing parameter |
| repeated  | the user repeated a value for an existing parameter               |
| negated   | the user has negated the value of an existing parameter           |
| requested | the user has requested the value of a parameter                   |
| inferred  | the system has inferred a parameter value                         |

Table 1: Semantic Function Assignments

With U1 in (1), there is no existing object of the same type, so a new id value `flight1` is instantiated and the parameters `flightid` and `sourcecity` are assigned the function *new*, while `goaltime` is assigned the function *requested*. A more interesting situation arises if, instead of U2, the user provides an over-informative response like *Yes, BA777 from Scunthorpe leaving late morning* where the system has misrecognized the user's repetition of *Stockholm* as *Scunthorpe* and the user also provides the departure time. The following discourse state would be generated:

(3)

$$\left[ \begin{array}{l} id : da \\ owner : user \\ type : inform \\ object : \left[ \begin{array}{l} id : flight1 \\ type : flight \end{array} \right] \\ parameters : \left[ \begin{array}{l} flightid : ba777 \\ sourcecity : scunthorpe \\ sourcetime : [1000, 1200] \end{array} \right] \\ functions : \left[ \begin{array}{l} flightid : repeated \\ sourcecity : modified \\ sourcetime : inferred \end{array} \right] \end{array} \right]$$

The description is compatible with an existing object `flight1`: while the flight identifiers are the same, the value of the `sourcecity` parameter has been modified, and the user has introduced a new parameter `sourcetime` whose value has been inferred as a time interval.

Comparison with an accessible, compatible object may also resolve partial descriptions (see Section 4.2 below) as well as underspecified parameters. For example, if the user replies to a system request for the departure time with the utterance *11.20*, comparison with the representation of the system utterance indicates that this time parameter needs to be contextually interpreted as a `sourcetime` parameter. Finally, since a user turn may consist of one or more utterances — either because that is how it was uttered or because recognition errors, speech disfluencies, out of vocabulary items, or gaps in the recognition grammars lead to ‘fragmentary input’ — the semantic interpretation mechanism updates the discourse model on an utterance-by-utterance basis. In this way, the representation of each utterance can be compared with the interpretation resulting from interpretation of the previous utterance in the turn. The effect is that task information in multi-utterance turns are assigned the same functions as in single utterance turns, except that a set of interpretations will be passed onto the dialogue interpretation function<sup>1</sup>.

<sup>1</sup>The dialogue interpretation updates the dialogue

## 2.3 TASK TECHNIQUES

Task parameter values are passed to the task module for updating its model. Since task structure determines many dialogue continuations, the task module embodies navigation strategies to efficiently obtain information necessary for successful database access, as well as techniques for suggesting alternative solutions and presenting the information to the user. These strategies result in goals being forwarded to the dialogue module.

The task module checks whether the task model is sufficiently instantiated for database access. This is determined by matching the task model against a set of request templates. Each template specifies obligatory parameters for a particular type of request. For example, a flight enquiry about the arrival time uses the following template:

(4)

$$\left[ \begin{array}{l} input : goaltime \\ required : \left\{ \left[ \begin{array}{l} flightid, date \\ sourcecity, goalcity, \\ date, sourcetime \end{array} \right] \right\} \\ output : \left[ \begin{array}{l} flightid, sourcecity, sourcetime, \\ goalcity, goalterminal, goaltime \end{array} \right] \end{array} \right]$$

The template can be satisfied with either the flight number and date, or the departure and arrival cities, the date, and the departure time of the flight. If the task model does not completely match the request template, then one of the required parameters is sought from the user. In dialogue (1), the `flightid` is provided by the user but the date is inferred. Default constraints are used to provide values for certain parameters, unless the user provides details to the contrary; for example, that the date of the flight is ‘today’. Necessary constraints are used to infer less specific information from more specific information; for example, if the arrival airport is known then, in many cases, so too is the arrival city.

Once database access has taken place, the solutions are filtered according to four subintervals:

$$0 \dots \text{Min} \dots \text{Max} \dots \text{Threshold} \dots$$

**Min** and **Max** describe the optimum range for the number of solutions which can be presented to the user directly. Entries within the interval from **Max** to **Threshold** will be tolerated too, but the results summarized. The numbers of solutions below **Min** or above **Threshold** are not acceptable for the presentation. One important strategy for dealing model on an utterance-by-utterance basis too.

| Type      | Function                            |
|-----------|-------------------------------------|
| open      | open the dialogue                   |
| close     | close the dialogue                  |
| request   | seek information                    |
| spell     | seek information through spell mode |
| confirm   | check information                   |
| inform    | give information                    |
| explain   | explain behaviour                   |
| terminate | force termination of dialogue       |

Table 2: Dialogue Goal Types

with the former case is to use constraint relaxation so that the value of a non-discrete task parameter is relaxed and database access retried. For example, the user may ask for information about a flight departing at 10.30 but will accept information about flights leaving just before or after that. Finally, each acceptable solution is presented according to the request template. With (4), solutions to a request for the arrival time will contain the information shown in S3 of (1)<sup>2</sup>. If there are no solutions even after constraint relaxation, then the user is informed that their enquiry has been unsuccessful.

## 2.4 DIALOGUE TECHNIQUES

On the basis of semantic and pragmatic functions assigned to user input, an interpreter in the dialogue module applies update rules to the current state of the dialogue model to derive a new state. As a side-effect, task-related information may be passed to the task module, and further goals added. This state is then evaluated to select those goals which provide the locally optimal dialogue continuation.

The dialogue model is composed of goals and contextual variables. These goals are types of dialogue acts which describe intentions of the system in the dialogue. As illustrated in Table 2, some goals are concerned with information transfer, like *request* and *inform*, while others, such as *close* and *confirm*, are concerned with dialogue control (Bunt 1989). Each dialogue goal is of the form

$TYPE \times CONTEXT \times STATUS \times COUNTER$

where *TYPE* indicates the dialogue act type,

<sup>2</sup>All arrival information is provided and repeated to minimize difficulties which may arise from synthesized speech and reduce the need for the user to ask for additional information.

*CONTEXT* the semantic function, *STATUS* whether the goal is active or pending, and *COUNTER* the number of times the goal has been realized. The contextual variables indicate the current status of dialogue strategies, such as whether parameters are to be confirmed and the type of confirmation strategy, as well as the status of various contextual parameters, including a ‘repair’ threshold value for the *COUNTER* in dialogue goals.

A dialogue state is characterized as a set of instantiated goals and contextual variables. The active dialogue goals are updated on the basis of the semantic and pragmatic functions using rules which are sensitive to contextual variables<sup>3</sup>. Each type of goal is associated with success and failure conditions as illustrated in Table 3. A goal will

| Goal    | Success                    | Failure                     |
|---------|----------------------------|-----------------------------|
| confirm | $\emptyset$                | modified,<br>repeat, reject |
| request | new, modified,<br>repeated | $\emptyset$                 |
| explain | $\emptyset$                | $\emptyset$                 |

Table 3: Success-Failure Conditions

be satisfied if (a) the semantic input matches the same parameters in its *CONTEXT* and (b) the assigned semantic and pragmatic functions either do not contradict any of its failure conditions or match one of its success conditions. In (1) the system utterance S2 *BA777 from Stockholm?* is the realization of two *confirm* goals — one for the *flightid* parameter and another for the *sourcecity* — and both are satisfied by the user’s response *yes*: this utterance is assigned the pragmatic function *accept* which is not one of its failure conditions. In this way, one function can affect more than one goal. Similarly, a *request* goal will only be satisfied if the semantic function of the relevant parameter is *new*, *modified* or *repeated*; the goal will fail if, for example, the user simply says (or is interpreted to say) *yes*. A semantic function which is relevant to a goal but does not match its *CONTEXT* can also be associated with the goal; this occurs with over-informative responses such as represented in (3). Finally, an *explain* goal will always be satisfied.

The effect on the dialogue state is determined

<sup>3</sup>A subset of pending goals are updated in a similar manner. This allows pending requests for parameters which the user has provided in over-informative answers to be removed.

using update rules for (a) the satisfaction function of each goal, and (b) the semantic and pragmatic functions now associated with them. Each rule maps from a function to a set of actions to be applied to the goal in the current dialogue state. These rules are sensitive to goal type, the specific semantic parameters as well as the status of contextual variables as illustrated in Table 4. If a goal

| Function | Conditions       | Actions                                  |
|----------|------------------|--|
| succeed  | $\emptyset$      | pop                                      |
| succeed  | type=close       | pop,<br>set(restart,yes)                 |
| fail     | rt=less          | rep:l                                    |
| fail     | rt=eq, type=open | pop,<br>set(menu,yes)                    |
| modified | rt=less, cs=yes  | post:confirm,<br>negPred,<br>set(sc,yes) |
| modified | rt=eq, cs=yes    | post:spell,<br>negPred                   |
| modified | cs=no            | $\emptyset$                              |

Table 4: Dialogue Update Rules (‘rt’ is repair threshold, ‘cs’ is confirmation strategy and ‘sc’ is single confirmation strategy)

is satisfied, then it is ‘popped’ from the dialogue state; if it is a close goal, a contextual variable is set indicating that this system should restart after the generation cycle is complete. If a goal is not satisfied and its *COUNTER* is less than the repair threshold variable, the goal is retained but its *COUNTER* is incremented; a failed open goal, whose *COUNTER* is equal to the threshold, results in the system adopting a ‘menu-driven’ interaction style. The first and second rules for the *modified* semantic function apply when the confirmation strategy is active. In the first, the repair threshold is not met, so a goal confirming the new value is ‘posted’ in the dialogue state, a negative prediction is generated, and the confirmation strategy is set to confirm parameters in separate utterances. In the second rule the repair threshold is met resulting in a spell goal being used to obtain a ‘fresh’ value for the parameter<sup>4</sup>. In the third instance, the confirmation strategy is not active and no action is taken.

It is very straightforward to change dialogue behaviour by changing these rules and contextual variables. For example, if recognition performance was so good that confirmation was unneces-

<sup>4</sup>Experience taught us that multiple modifications of a parameter are usually an artifact of poor recognition.

sary, then the confirmation strategy could be set to ‘no’. Alternatively, if we decided not to confirm *modified* information associated with request goals below the repair threshold, then we simply need to add a *modified* rule specific to this goal type.

In general, the approach allows for confirmation and clarification of user input (to minimize dialogue breakdown), as well as requests for further information (to maximize dialogue progress). Since the application of these rules are sensitive to contextual variables — which are themselves threaded through successive dialogue states — progress can be monitored, and responded to, throughout the dialogue. The effect is that the overall behaviour of the system varies with the degree of success in the dialogue (Eckert and McGlashan 1993; Heisterkamp 1993). If the dialogue is progressing well, the user is permitted considerable freedom; otherwise, the system restricts what the user can say so that the dialogue can recover.

Once the dialogue model has been updated, a subset of the current goals are selected for realization in the next system turn<sup>5</sup>. The selection algorithm uses a classification of goals as initiatives, reactions and evaluations; for example, a request goal is an initiative, inform a reaction, and confirm an evaluation. Depending on the dialogue strategies, different selections may result:

**evaluation only** select some confirmations depending upon the confirmation strategy

**initiatives and reactions only** select one initiative and any explanatory reactions

**evaluations, initiatives and reactions** select an initiative and some confirmations depending on the confirmation strategy, and any explanatory reactions

Goals are then realized in the following order: Reactions (Explanations) > Evaluations > Initiatives > Reactions (Others). This ensures that answers are realized earlier than questions (thus generating adjacency pair sequences) and that explanations precede all other output. Explaining that the system is, for example, repeating a confirmation due to ‘nothing being heard’ makes the behaviour of the system more transparent to the user and subtlety influences how they respond. Finally, predictions are calculated for each realized goal. These are used to supply top-down con-

<sup>5</sup>Those selected have the *STATUS* active; the remainder are pending.

straints on the speech and language components, thus limiting the search space and forestalling recognition errors. Depending the type of parser and speech recognizer, they can be used to limit or prioritize what can be recognized (by means of dialogue-state dependent n-grams or finite state grammars), or not recognized — predictions can be negative so as to avoid the ‘broken record’ effect of continually repeating the same recognition error.

### 3 BEYOND SPOKEN DIALOGUE MANAGEMENT

As spoken dialogue systems for simple information services begin to move into the area of technology, research interest is increasing turning to the integration of spoken dialogue interfaces with other modalities. A **multimodal** system supports interaction with the user through more than one modality, with respect to input and/or output, *and* with the capacity to interpret and/or generate with respect to the representation of content<sup>6</sup>. Various systems have been developed recently combining speech (or text) interfaces with other modalities (Maybury 1993); CUBRICON, for example, combines speech with a graphics and mouse interface in the domain of mission planning; and Alfresco combines text with hypermedia for art exploration.

The aim of combining a speech interface with a graphical interface is to provide more efficient access to the backend application than the graphical interface alone. Apart from the general advantages of allowing an alternative input and output modality, synergic effects (such as clicking on an object and speaking a command), concurrent tasking for eyes-/hands-busy situations, speech can compensate for some of the apparent limitations of a graphical interface. For example: increased speed of interaction, higher-bandwidth (attention and attitude expressed through stress and prosody, etc), descriptions of objects which are not visually present or are awkward to access in a conventional interface without negation, quantification or temporal expressions, as well as the convenience of reduced descriptions, such as anaphora and ellipsis. Conversely, the graphical interface can compensate for limitations of speech

---

<sup>6</sup>Various definitions of multimodal system have been offered, and are frequently contradictory due to *modality* being used both to refer to the sensory channel by which information is conveyed (e.g. visual) and the form of the expression (e.g. graphics).

by making immediately visible the effects of actions upon objects, and indicating through the display which objects (and by extension which actions) are currently salient for the system.

Recent empirical studies have suggested that users not only prefer to interact multimodally, but that compared with a speech-only interface a multimodal interface can reduce performance errors, spontaneous disfluencies and task completion time (Oviatt 1996). However, such results must be treated with caution. Firstly, the performance gain is not consistent across all tasks; other studies have shown that task completion time is faster with speech-only interfaces in verbal and numerical tasks. Secondly, these results were obtained using the WOZ technique: response time was always less than 1 second and there were no (simulated) recognition errors. User’s perception of, and performance with, speech recognition can be downgraded when faced with the imperfections of present-day technology (Damper and Wood 1995). Thirdly, most studies have focused on speech interfaces for command and control applications rather than more interactive applications where some degree of multimodal dialogue management is required (McGlashan and Axling 1996). Integrating a spoken dialogue interface with a graphical interface for this type of application may introduce new problems; for example, spoken dialogue interfaces are instances of indirect management interfaces (the user delegates some tasks to a software agent) while graphical user interfaces are instances of direct manipulation interfaces (the user is responsible for explicitly initiating and monitoring all tasks). Consequently, it is our belief that we need to build and evaluate such multimodal systems before we can be sure that they match the needs (and abilities) of users as well as telephone-based systems have met the needs of users for simple information services.

### 4 A MULTIMODAL SYSTEM FOR CONSUMER INFORMATION

Our dialogue management techniques are being incorporated into a multimodal system combining a spoken dialogue interface with a graphical user interface, which provides consumer information about microwave ovens<sup>7</sup>. Due to limited

---

<sup>7</sup>This work is part of the OLGA project. The partners are SICS, Stockholm University, KTH, Nada, and Nordvis AB. I am particularly grateful to Olle Sundblad, Jonas Beskow, Nikolaj Lindberg and Joel Sunnehall for their con-

resources, we adopted a ‘storyboard’ approach to the design of the system (rather than conducting a user study and running WOZ simulations to obtain user requirements) and restricted the first version of the system to allow either speech or direct manipulation input but not both in the same turn. Our design approach resulted in scripts describing how hypothetical users with different needs might interact with the system. A fragment of one script (translated from Swedish), aimed at users with a preference for the speech modality, is shown below.

- (5) S1: Whirlpool has five tested microwaves on the market. [Five ovens are shown together with their main properties]
- U1: I would like one with a grill, but they are very expensive. Is there anything cheaper with a grill?
- S2: Whirlpool has no cheaper ovens with a grill. Here you can see a selection of cheap microwaves with a grill. [A number of ovens are shown, plus a button to ‘show more’]
- U2: Okay, print them out.
- S3: You have choose ovens which don’t have digital timing. Would you like to know more about digital timing? [Olga holds up a ‘tip’ flag]
- U3: Yes.
- S4: [A movie illustrating the advantages of digital timing is shown]. Is there anything else you want to know?
- U4: Yes. Show me the Whirlpool ovens again.
- S5: [The whirlpools are shown again]

We also developed a simple mockup of the user interface in order to get a clear idea of what the system would look and sound like for the user. A snap-shot, corresponding to S2 in (5), is shown in Figure 1. In addition to speech and language components, the system is composed of three other components: a direct manipulation interface which provides graphical information and widgets for navigation; an animated talking agent whose speech is synchronized with its lip movements, and who performs gestures; and a dialogue manager for coordinating interpretation and generation in both modalities. The dialogue manager is the same dialogue manager described in Section 2 but augmented in four fundamental ways.

tributions to this project.



Figure 1: Olga User Interface

#### 4.1 NON-SPEECH INPUT AND OUTPUT

In order to manage multimodal dialogues, input and output need to be informationally compatible at the dialogue management level. A user may provide input via buttons in the interface and the system generate a spoken response; or a user may reference in speech an object which the system has realized graphically. Consequently, all input and output is represented in the semantic description language discussed in Section 2.2. Rather than ask the system to print out the products on display in U2, the user could have pressed the ‘show more’ button. This button, part of the graphical realization of the system goals, is directly associated with a show action and a semantic description of the other microwaves. Using this technique, interpretation of graphical input is simpler than interpreting speech input because there can be no recognition errors and, since a specific semantic description is already provided, no difficulties in determining what the user is referring to.

#### 4.2 REFERENCING VISUAL OBJECTS

Like simple information service dialogues, user actions can be interpreted as providing parameters for the information service as with U1 in (5), or as dialogue controls like U3. Consumer service dialogues also introduce a set of actions used to execute commands on objects and navigate around the information space. For example in U2, the user asks the system to execute a print command on visually-present objects and in U4 the user asks the system to show previously dis-

played objects. Such actions are not observed in simple information services since the service is completed once the information has been presented to the user.

In order to characterize the referential descriptions in these utterances, we use the existing *modus* feature of our semantic representation; for example, *them* will have the features *pro:pro*, *def:def*, *number:pl* indicating that it is a definite plural pronoun, and *the whirlpools* the features *pro:nonpro*, *def:def*, *number:pl* together with a *manufacturer:whirlpool* parameter. Reference resolution is still based on finding compatible, accessible concepts in the discourse model for definite expressions, but three new semantic functions are required<sup>8</sup>:

**ref\_success** appropriate antecedents have been found; this may be a single object or a set of objects which form part or all of an existing semantic description

**ref\_ambig** more than one appropriate antecedent has been found

**ref\_failure** no appropriate antecedent has been found

At the dialogue level, the first function allows the command to be executed immediately, while the second and third result in clarification goals being added to the current dialogue state. Interpretation of U2 and U4 in (5) results in *ref\_success* assignments: *them* refers to objects (from the preceding system turn) which are currently visible; and *the whirlpools* to objects (which are no longer visible) from a system turn earlier in the dialogue. While this simple technique has been successful so far, we realize that a more sophisticated approach may be necessary; for instance, an approach where the accessibility of an object is a function of the priority and temporal persistence of the modality introducing it (McGlashan and Axling 1996).

### 4.3 MODALITY SELECTION

The dialogue state is updated and goals selected for realization according to the principles discussed in Section 2.4. Extensions have been required to select between the three realization modalities; apart from speech and graphics, the Olga animated agent can perform a limited range

<sup>8</sup>With indefinite expressions, such as *show me a whirlpool oven*, a new object is created in the discourse model, and a database lookup executed.

of gestures such as pointing at the graphical display, looking at the user, and facial movements such as eyebrow-raising, smiling, looking sad, and so on. In general, modality selection is defined in terms of characteristics of the output information, and the expressiveness and efficiency of the alternative modalities for realizing it. Given the absence of many theoretically-motivated selection principles, most multimodal systems employ domain-appropriate heuristics; for example, the AIMI system uses rules based on characteristics of the user query. Similarly, we currently use rules based on the type and content of system goals<sup>9</sup>.

Goals with a control or feedback function are realized in speech and gesture: for example, success in understanding user input is indicated with a head nodding gesture, while failure is indicated by speaking an explanation of the failure together with raised eyebrows and the mouth turned down. Product information is presented in speech and graphics; detailed product information is displayed while the agent provides a spoken overview as illustrated in Figure 1. Finally, since the content and structure of the graphical display can have a significant effect on user utterances — an unstructured display can easily result in speech disfluencies and hence speech recognition problems — the realization of inform goals is followed by the agent looking at the user. In situations (like S4) where the system presents a video explanation of microwave features, the visual realization is followed by a spoken realization of a ‘dialogue continuation’ goal designed to direct the user’s attention back to the interaction with the system.

### 4.4 NAVIGATION AND FILTERING

At the task level, the system uses the techniques described in Section 2.3 to update its task model and filter database solutions. When access results in too many solutions to present on screen, they are filtered according to task-specific strategies such as whether the product has been included in consumer tests. When there are too few solutions, constraint relaxation is used to offer information about alternative products; in S2, the system realizes a goal explaining that the *manufacturer* parameter has been relaxed in order to satisfy the user’s current request parameters. This has required an extension to deal with relaxa-

<sup>9</sup>We plan to extend these rules to access contextual variables so that they are adaptive to user behaviour, especially their choice of input modalities and progress with speech recognition.

tion of discrete parameters. Another extension is the addition of functionality to allow the system to provide an explanation of desirable product features<sup>10</sup>. This behaviour is triggered when the user has selected products for printing. For example, the user has selected a product without digital timing, then the agent holds up a 'tip' flag and offers an explanation of the feature as in S3.

## 5 CONCLUSION

Effective dialogue management can compensate for limitations of speech and language processing by providing a cooperative interaction with the user. We have described techniques of spoken dialogue management for simple information services. The layering of semantic and pragmatic interpretation, combined with flexible dialogue rules, provide fine-grained control over the system's behaviour. This approach has been tested and evaluated in a number of speech-only systems with remarkably few changes.

The approach has now been incorporated in a multimodal system combining speech and graphical interfaces. Interpretation and generation of graphical input is based on the same semantic and pragmatic functions required for spoken language. Since the application domain includes navigation and command utterances not observed in the simple service applications, extensions have been made to the reference resolution algorithms. Additions have also been required for determining in which modalities system goals should be realized. However, there are a considerable number of multimodal issues which we have yet to address even within the (relatively) simple domain of consumer information services. The majority concerned our ignorance of what functionality users *actually* need, and how they will *really* react to this type of multimodal system. User trials will allow us to begin to address these issues.

## REFERENCES

- Aust, H. and M. Oerder (1995). Dialogue control in automatic inquiry systems. In *Proceedings of TWLT 9*.
- Bernsen, N. O., L. Dybkjaer, and H. Dybkjaer (1994). A dedicated task-oriented dialogue theory in support of spoken language dialogue system design. In *Proceedings of ICSLP'94*.
- Bunt, H. (1989). Information dialogues as communicative action in relation to partner modelling and information processing. In M. M. Taylor, F. Néel, and D. G. Bouwhuis, editors, *The structure of multimodal dialogue*, pages 47–71. North-Holland.
- Cohen, P. (1995). Dialogue modelling. In G. Varile and A. Zampolli, editors, *Survey of the State of the Art in Human Language Technology*. <http://www.cse.ogi.edu/CSLU/HLTSurvey>.
- Damper, R. I. and S. D. Wood (1995). Speech versus keying in command and control applications. *International Journal of Human-Computer Studies*, 42: 289–305.
- Eckert, W. and S. McGlashan (1993). Managing spoken dialogues for information services. In *Proceedings of Eurospeech'93*, pages 1653–6.
- Giachin, E. and S. McGlashan (1996). Spoken language dialogue systems. In K. Church, S. Young, and G. Bloothoof, editors, *Corpus-Based Methods in Language and Speech Processing*. Kluwer.
- Hayes, P. J. and D. R. Reddy (1983). Steps towards graceful integration of spoken and written man-machine communication. *International Journal of Man-Machine Studies*, 19: 231–84.
- Heisterkamp, P. (1993). Ambiguity and uncertainty in spoken dialogue. In *Proceedings of Eurospeech'93*.
- Heisterkamp, P. and S. McGlashan (1996). Units of dialogue management: An example. In *Proceedings of ICSLP'96*.
- M. T. Maybury, editor (1993). *Intelligent Multimedia Interfaces*. MIT Press.
- McGlashan, S. and T. Axling (1996). Talking to agents in virtual worlds. In *Proceedings of 3<sup>rd</sup> UK Virtual Reality Conference*.
- McGlashan, S., N. M. Fraser, N. Gilbert, E. Bilange, P. Heisterkamp, and N. J. Youd (1992). Dialogue management for telephone information services. In *Proceedings of ICALP'92*. Trento, Italy.
- Oviatt, S. (1996). Multimodal interfaces for dynamic interactive maps. In *Proceedings of CHI'96*.
- Peckham, J. (1993). A new generation of spoken dialogue systems: results and lessons from the SUN-DIAL project. In *Proceedings of Eurospeech'93*, pages 33–40.
- Wooftitt, R. C., N. M. Fraser, G. N. Gilbert, and S. McGlashan (forthcoming). *Designing Interaction: a conversation analytic study of human-(simulated) computer interaction*. London: Routledge.

---

<sup>10</sup>This feature is not a design decision on our part, but a requirement from our sponsors.