

An approach to Multi-Strategy Dialogue Management

Shiu-Wah Chu, Ian O'Neill, Philip Hanna, Mike McTear

School of Computer Science
Queen's University of Belfast, Belfast, U.K.
{s.chu, i.oneill, p.hanna}@qub.ac.uk

School of Computing and Mathematics
University of Ulster, Newtownabbey, U.K.
mf.mctear@ulster.ac.uk

Abstract

Existing dialogue systems typically use only one dialogue strategy: finite-state based, frame-based or agent-based. This paper describes research done on a dialogue manager (DM) that uses all three dialogue strategies. This DM can determine the most suitable dialogue strategy to be used according to circumstances, and change to a different strategy whenever required. The research puts emphasis on how such a multi-strategy DM improves the naturalness of human-computer interaction. Determining best use of different dialogue strategies, rather than tailoring dialogue management to a particular business domain, motivates the research.

1. Introduction

Current dialogue systems typically use one of the three following existing dialogue strategies [1]:

- Finite-state – The dialogue path is represented as a finite state machine, in which transitions to new dialogue states are determined by the user's selections from fixed sets of responses to system questions.
- Frame-based – The system constructs the dialogue in order to fill in the slots of a pre-determined frame.
- Agent-based – The system constructs the dialogue as it recognizes and attempts to satisfy the user's objectives.

The aim of the current research is to develop a dialogue manager (DM) that, like a human, has more than one information elicitation strategy and is able to change strategy according to circumstances. Implemented in Java, the DM will be integrated into the Queen's Communicator [2]. Due to the object-oriented nature of the Queen's Communicator, it is possible to add or modify some components of the dialogue manager while retaining other components.

The goal of this research project is to investigate whether a dialogue manager can select the most appropriate dialogue strategy according to the discourse state and the system's understanding of the user's utterances.

2. Architecture of the Dialogue Manager

The dialogue manager consists of two main components: the Dialogue Strategy Manager (DSM), which is responsible for choosing and changing dialogue strategy, and the Problem-Solving Manager (PSM), which is responsible for the problem-solving aspect of the system. As shown in Figure 1, the DSM and PSM and their respective components compose the DM: the PSM decides what dialogue acts are required to further the discourse, and the DSM decides the manner in which the dialogue acts should be expressed. The DSM and PSM are responsible for different aspects of dialogue

production and together they produce the dialogue required for the interaction.

2.1. DSM

The DSM can provide different dialogue strategies by using its different components: instances of the Finite State Manager (FSM) class, instances of the Frame class, or for an agent-driven strategy, instances of the FreeForm class.

2.1.1. FSM

Each FSM instance contains a model of a partial dialogue that is represented as a finite-state machine. The FSM instance will be responsible for asking the user specific questions and determining whether the user input is valid or not.

2.1.2. Frame

Each Frame has a set of slots that it wants to fill with data captured from the user. The Frame component will determine the questions that need to be asked in order to fill slots and capture any useful data from each user input.

2.1.3. FreeForm

The FreeForm component does not apply any restriction on the interaction. It asks open-ended questions and accepts any user input. Once it has captured the user input, FreeForm will pass the input directly to the PSM, which provides the agent-based aspect of the DM and proceeds with the dialogue.

2.2. PSM

The PSM and its problem-solving (PS) objects are designed in similar fashion to the problem-solving model described by Blaylock et al. [3]. A user request will be represented by an *Objective*, which can be fulfilled by a corresponding *Recipe*. The *Recipe* may use one or more *Resources*. The design of the problem-solver is not the primary motivation of this research: however, as well as satisfying the user's objectives, the system may well have objectives of its own, which must be reconciled with user objectives.

2.3. Relationship of DSM and PSM

Whenever the PSM requires a system utterance, it sends a request to the DSM. The DSM will choose the most appropriate dialogue strategy for the requested utterance. The DSM will call the appropriate FSM, Frame or FreeForm object.

If the chosen dialogue strategy is successful, the DSM will return control back to the PSM. However, if the chosen dialogue strategy is unsuccessful, the DSM will need to select an alternative strategy. If there is no available strategy, the DSM will declare failure and return control back to the PSM;

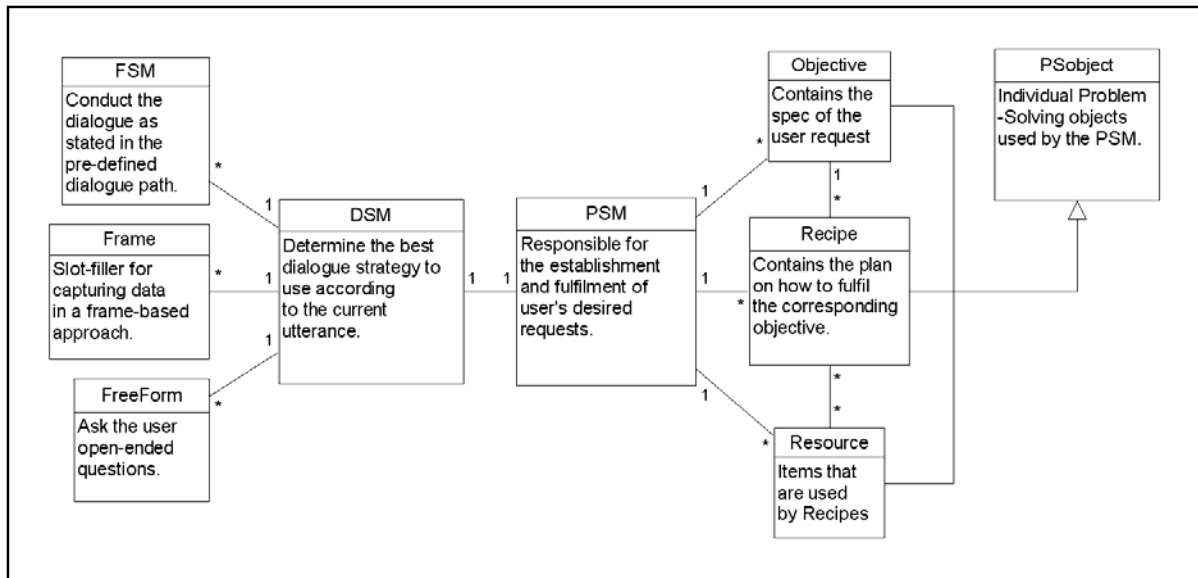


Figure 1: Relationships between the DSM and PSM

the PSM will then decide the next appropriate action, e.g. re-confirm the current objective with the user.

3. Using the Dialogue Strategies

3.1. Finite-State (FS)

FS strategies depend on the developer's knowledge of required inputs. The rigidity of FS strategies make them useful for restricting the user's replies.

There are different types of FS-based strategies available. One example is to have the system to list out all the possible inputs to the user and have the user reply with the actual input, as shown in the example in Table 1. Alternatively, in the example in Table 2, the system lists out all the possible choices with numbers and the user replies with the number that corresponds to their selected option.

System	Do you want to go to Belfast, Dublin or London?
User	Belfast.

Table 1: A FS-based strategy example

System	What is your destination? Tell me the number before your desired destination. One-Belfast, Two-Dublin, Three-London
User	One.

Table 2: Another FS-based strategy example

3.2. Frame-based

Frame-based strategies have no fixed dialogue path, but they have a frame of slots that are filled by capturing data from the user. Frame-based strategies can capture multiple data in one

prompt, so they are generally used whenever the system requires a set of related data from the user, e.g. in ticket-booking systems such as the Mercury Flight Reservation System [4]. Users have the flexibility to fill in the slots in any order and can fill more than one slot per dialogue turn.

3.3. Free-Form

The agent-based strategy can deal with multiple domains. Therefore it is always prepared to accept any relevant data, even if the data is relevant to a different domain than the current one. This strategy can accept any user inputs at any time, but it is also capable of taking the initiative whenever required [5]. Since it can deal with complex interaction, the free-form strategy can be used in interactions for establishing and recognizing objectives.

3.4. Multi-Strategy

Each type of strategy has its own disadvantages. Finite-state strategies are only efficient for performing simple tasks in a rigid manner. A frame-based strategy is flexible and efficient for capturing a set of data, but its context can only be fixed on one particular domain at any given time.

The free-form strategy allows the user to reply with anything that is within the scope of the system domain, but it is questionable whether such freedom is required if the current domain is refined enough such that all the system requires is to obtain some standard data from the user. While the agent-based strategy provides a wide degree of flexibility and freedom to the user, it also increases the chance of misunderstanding the user responses. Hence, the frame-based strategy is often a better strategy to use when the system wants to capture a set of data from the user.

Although free-form based and framed-based strategies are flexible, there are at least two circumstances where a rigid finite-state strategy is preferable:

- When a system provides a wide degree of flexibility and freedom to the users, it also increases the chance for the

user to stray out of the current domain [6]. The system may decide to restrict the user's freedom in this respect.

- When an error occurs at the recognition level, the less complex finite-state strategy will impose greater restriction and hence have a better chance of obtaining data successfully from the user.

If a dialogue manager has all three types of dialogue strategy available, it can provide more varieties of dialogue interaction and can choose the best strategy for each specific utterance.

3.5. Context of usage

The system will use the above dialogue strategies as follows:

- Frame-based strategies are mainly used for capturing sets of data.
- Finite-state strategies are mainly used whenever a frame-based or agent-based strategy fails to capture a specific datum, or if the system wants tighter control on the user prompts.
- Free-form strategy is used whenever the user's reply is not relevant to the current objective or whenever the current objective has still not been established.

4. Dialogue Strategy Selection

4.1. Criteria for selecting a Dialogue Strategy

Whenever the DSM receives a request from the PSM, it considers a number of criteria before it determines the most suitable strategy:

- One of the goals of this project is to create a dialogue system that can engage in natural interactions. Therefore, the DSM will be more likely to choose a flexible strategy rather than a rigid one.
- The DSM considers the *number of data* to be captured. If the PSM wants to capture many data, it is more appropriate to choose strategies that can capture multiple data per dialogue turn.
- The DSM also considers the *set of expected inputs*, or the type of data it is expected to capture. For example, FS-based strategies can be used if the input set is small and fixed, whereas if the expected inputs are new objectives then the agent-based strategy will be the preferred choice.
- If there is any *special requirement* on choosing dialogue strategy, then the PSM should pass it to the DSM. For example, the system may require the data to be requested in a certain order, or it may require a very high level of accuracy in the recognition of data.
- An important criterion is the system's *level of understanding* of the user's responses. The DSM is less likely to choose a particular strategy if the same strategy has failed many times previously.

4.2. Strategy Transition Criteria

If the dialogue strategy chosen by the DSM does not receive the expected user reply for the current utterance, the DSM will change to a different dialogue strategy. Each type of

dialogue strategy has its own criteria for determining when it has failed to capture the expected user input. Change of strategy may not follow immediately upon failure to elicit required information. However, failure makes transition more likely. We are currently developing criteria by which transition is triggered.

Every dialogue node of the FSM has a fixed number of possible pathways. If the user replies with a response that does not match any of the possible pathways then the user response will be regarded as an erroneous input.

Every Frame has a set of slots that it wants to fill in by capturing the corresponding data from the user. A system prompt will be constructed to ask for the datum for one specific slot, even though the user can supply data that were not specifically requested. The prompt is regarded as successful if the user's reply contains usable data. If the user's reply does not contain any useful data for any slot, then the elicitation strategy has failed for that turn.

FreeForm could receive user input that is gibberish or is outside the scope of the domain of the system. If the current objective of the PSM remains unchanged, the DSM should try a different dialogue strategy. However, repeated failure of a specific prompt may mean that the PSM must re-confirm its current objective.

4.3. An example

Table 3 shows a dialogue example that uses all three types of dialogue strategy and Figure 2 shows how the different components of the DM co-operate. System utterances are denoted by the letter 'S' followed by a number and user utterances are denoted by the letter "U" followed by a number. Though this sample is largely system-led, the overall dialogue may be mixed initiative. However, system-led segments clearly illustrate the system's information elicitation strategies in action.

Table 3: Example of a typical dialogue

S1	Welcome to the City Visit Planner. How can I help you?
U1	I want to book a hotel in Belfast.
S2	What type of room do you want?
U2	One double room at the Hilton Hotel.
S3	When do you want to check-in?
U3	I don't know yet. I have to buy my train ticket first.
S4	Do you want to buy the train ticket now, before booking the hotel?
U4	Yes, please.
S5	How many train tickets do you want?
U5	Two adult tickets, please.
S6	What ticket class do you want?
U6	Normal class will do.
S7	Sorry, I didn't understand. Do you want first class or second class tickets?
U7	Second class.
S8	What travel day do you want?
U8	Tomorrow.
S9	You want to book two second class train tickets to Belfast tomorrow. Is this correct?
U9	Yes.
S10	Ok, that's done. Do you want to book the

hotel now?
 U10 Yes, please.
 S11 When do you want to check-out?
 U11 This Friday.
 S12 You want to book a double room at the Europa Hotel from tomorrow until Friday. Is this correct?
 U12 Yes.
 S13 Ok, that's done. Do you also want to book train tickets leaving Belfast this Friday?
 U13 Yes please.
 S14 You want two second class train tickets leaving Belfast this Friday. Is this correct?
 U14 That's right.
 S15 Ok, that's done. Is there anything else I can help you with?
 U15 No, that's everything. Goodbye.

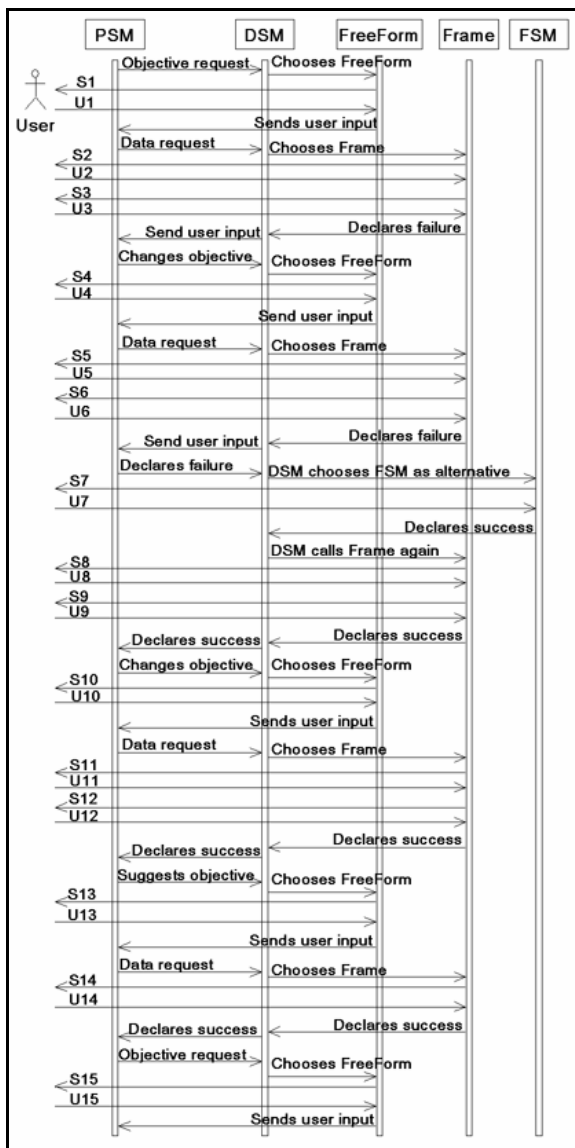


Figure 2: Sequence Diagram of the dialogue

In this example the different strategies have been used as follows:

- Frame-based strategy is used for capturing and confirming the sets of required data for each different objective. In the example it is used when the system wants to capture data for *booking hotel* (S2 to U3), *booking train tickets* (S5 to U6, S8 to U9) and *booking return train tickets* (S14 to U14).
- Finite-state strategies have been used once. This occurs when the Frame fails to capture the *ticket class* (S6 and U6) and the DSM uses the FSM instead (S7). When the FSM has captured the *ticket class* successfully the DSM changes back to the Frame again.
- Free-form strategy is used whenever the system demands a new objective from the user (S1 and S15), or whenever it is suggesting a new objective (S13). It is also used whenever its interaction with the user concerns multiple objectives, e.g. when it receives a new objective from the user when it is currently working with a different objective, or when it requires confirmation of changes in objectives (U3 to U4).

5. Concluding Remarks

Since a multi-strategy DM can change its strategy according to circumstances, it can provide more adaptability whenever the system misinterprets a user response. By changing its strategy the system can also adjust restrictions on the user's response. The system can cope with multiple objectives and can change its current objective as requested by the user, while still maintaining a high level of control on the interaction if desired.

Future work will explore the extent to which the DSM can be independent of the domain of the PSM.

6. References

- [1] McTear, M.F., "Spoken Dialogue Technology: Enabling the Conversational User Interface", *ACM Computing Surveys*, Vol. 34, No.1, pp. 90-169. March 2002.
- [2] O'Neill, I., Hanna, P., Liu, X., McTear, M., "The Queen's Communicator: An Object-Oriented Dialogue Manager", *Proceedings of Eurospeech 2003*, Geneva, Switzerland. September 2003.
- [3] Blaylock, N., Allen, J., Ferguson, G., "Managing Communicative Intentions with Collaborative Problem Solving", *Current and New Directions in Discourse and Dialogue*, Kluwer Academic Publishers, Dordrecht, November 2003.
- [4] Seneff, S., Polifroni, J., "Dialogue Management in the Mercury Flight Reservation System", *Proceedings of ANLP/NAACL 2000 Workshop on Conversational Systems*, Seattle, WA. May 2000.
- [5] Allen, J.F., Byron, D.K., Dzikovska, M., Ferguson, G., Galescu, L., Stent, A., "Towards Conversational Human-Computer Interaction", *AI Magazine*, Vol. 22, No. 4, pp. 27-37. Winter 2001.
- [6] Zue, V.W., Glass, J.R., "Conversational Interfaces: Advances and Challenges", *Proceedings of the IEEE*, Vol. 88, No. 8, pp. 1166-1180. August 2000.