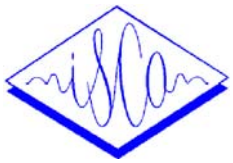
ISCA Archive
http://www.isca-speech.org/archive



INTERSPEECH 2004 - ICSLP
8th International Conference on Spoken Language Processing
ICC Jeju, Jeju Island, Korea
October 4-8, 2004

Flexible Dialogue Management Using Distributed and Dynamic Dialogue Control

Esa-Pekka Salonen[1], Mikko Hartikainen[1], Markku Turunen[1], Jaakko Hakulinen[1] and J. Adam Funk[2]

Department of Computer Sciences, Tampere Unit for Computer-Human Interaction, Speech-based and Pervasive Interaction Group, University of Tampere, Finland[1]
Department of Computation, Interactive Systems Design Group, UMIST, UK[2]
{eps,mhartikainen,mturunen,jh}@cs.uta.fi; jfunk@co.umist.ac.uk

10.21437/Interspeech.2004-117

Abstract

In spoken dialogue applications dialogue management has conventionally been realized with a single monolithic dialogue manager implementing a comprehensive dialogue control model. We present a highly distributed system structure that enables the integration of different dialogue control approaches to handle spoken dialogues. With this structure it is possible to integrate multiple dialogue control models into a single working application in a flexible fashion. The main principle is that all the processing is distributed into many compact agents that focus on single tasks. The agents process the information independently and share all the information via shared information storage. We present the principles that enable such distribution. In addition, a multilingual example application, AthosMail, is presented.

1. Introduction

Different ways to implement interaction management tasks in spoken dialogue applications are efficient in different situations. For example, some input interpretation methods are better with linguistically rich inputs, while other methods produce better results with imperfect sentences. Combining different approaches has been claimed to be more efficient than using a single approach [6]. For example, some mixed-initiative spoken dialogue systems are able to handle various types of inputs and combine different ways to interpret the user's intention [1], [5]. However, in most cases single approach, such as a monolithic dialogue control structure, is used.

We have implemented AthosMail, a multilingual speech-based e-mail system, within EU-funded DUMAS-project (Dynamic Universal Mobility for Adaptive Speech Interfaces IST-2000-29452) [12]. AthosMail has a distributed system structure including two dialogue control models. In addition, the system contains two input processing approaches: the first approach is based on linguistic analysis, and the second approach on concept spotting. Both dialogue control and input interpretation approaches are realized using a set of compact agents. The most suitable agents (implementing different dialogue control and input processing approaches) are selected for each system turn. This makes the interaction management highly dynamic and flexible.

Integration of different input processing and dialogue management approaches is crucial in multilingual applications, since every language has its specialties to be taken care of. Moreover, it is reasonable to assume that integrating different dialogue control methods leads to more robust dialogue handling in entirety. In addition, by integrating the methods as independent distributed agents we can achieve many advantages, such as truly distributed development of an application, and better the re-usability of existing components.

Next we present the underlying system architecture to explain the general principles. Then we proceed to the AthosMail application architecture, and explain how the input processing and dialogue management tasks are distributed and handled. We end the paper with discussion and conclusions.

2. Underlying System Architecture

The AthosMail application is built on top of the Jaspis architecture [10]. Jaspis is based on distributed set of managers containing evaluators and agents that share all the information via information storage (IS) that all components have access to. The main extensions to the Jaspis architecture are different methods for accessing and manipulating the AthosMail specific structures of the IS, mainly the discourse tree that describes the dialogue structure.

Jaspis is based on a highly distributed structure, where managers co-ordinate sets of evaluators to select appropriate agents that actually handle the tasks. Managers represent the high level organization of tasks that are handled within spoken dialogue applications. For example, input handling and surface generation are such tasks. However, the underlying architecture does not restrict the developer to a certain fixed set of managers. This enables the developers to introduce new managers when it is needed. Agents in Jaspis-based applications are compact software components. Ideally, agents are focused on handling single tasks. Agents enable further distribution of a system beyond the manager level. For example, an agent could be dedicated to produce a certain system output, such as a welcome message. This structure makes it possible to introduce many agents that are meant for the same task and the one that is selected depends on the situation. The system level adaptation mechanism and its possibilities is presented in more detail in [13].

Each manager uses a set of evaluators to evaluate whether the agent is suitable for the situation. Each evaluator gives a floating point score between zero and one to each agent. The product of these scores is the overall score of the agent. The agent with the highest score is selected. Table 1 illustrates the agent selection process, where agent 3 would be selected.

	Eval 1	Eval 2	...	Eval N	Overall score
Agent 1	0	-		-	0
Agent 2	1	0		-	0
Agent 3	1	1		1	1
...					
Agent n	1	0,5		0,5	0,25

Table 1: Agent selection.

3. AthosMail Application Architecture

AthosMail is a multilingual telephone based e-mail reading application, where users can access their mailboxes. AthosMail is particularly interesting because it consists of multiple distributed agents that can make dialogue decisions, and many agents that produce the information needed to make a certain decision.

3.1. System structure

The AthosMail application is distributed in many ways. First, it consists of two main applications: the dialogue system and the offline e-mail preprocessing system called MailServer. MailServer is out of the scope of this paper; here we concentrate on the dialogue system. Second, the system architecture has two layers of distributed components, namely managers and agents. These system components can be physically distributed, e.g. used in networked environments. Third, agents, managers and evaluators model logical distribution of tasks in various levels.

Figure 1 illustrates the AthosMail system and its high-level system modules from the viewpoint of interaction management.

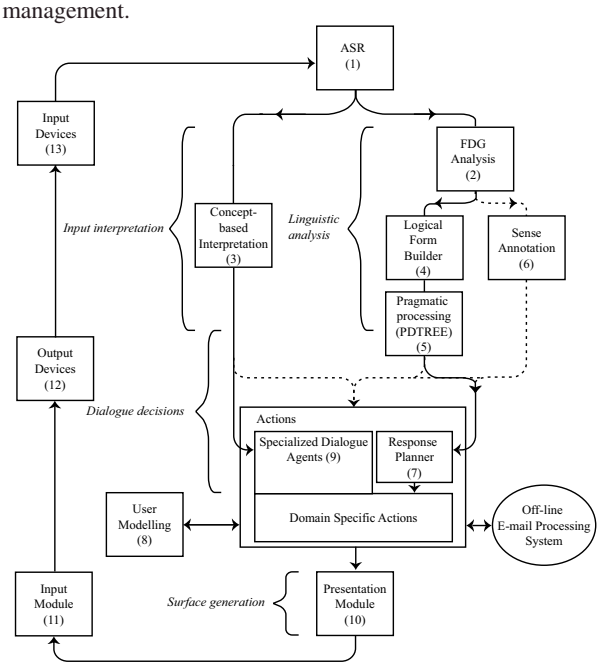


Figure 1: The overall processing structure of AthosMail

The interaction proceeds from top to down, arrows indicating possible paths. The paths starting from the speech recognition module (1) are related to the reasoning (input interpretation, dialogue management, response generation) taking place in the system. Next we will describe principles behind the components involved to the reasoning.

Interpretation of inputs is performed in two different ways. In the first, input interpretation is based on the linguistic analysis of the user input (2, 4 and 5). The linguistic analysis components build a logical form out of the user input and anchor it to the e-mail domain (pragmatic processing). In parallel, a concept spotting interpretation method is used to spot known concepts (single words, complete sentences or something in between) and solve their relations. Concepts are linked to the application domain later on by the dialogue agents.

Dialogue decisions are made using two approaches. The first approach uses a single agent called the Response Planner (7). The second approach utilizes multiple specialized dialogue agents (9). In principle, both dialogue management approaches are able to utilize interpreted inputs regardless the used interpretation method as the paths marked with dotted lines in Figure 1 suggest. Currently, AthosMail uses the Response Planner to carry out dialogue decision on the base of the anchored logical form, and specialized dialogue agents for processing spotted concepts. In addition, there are specialized dialogue agents to process system's internal states, e.g. to carry out incomplete operations.

Regardless of the result producing approach the rest of the interaction/data flow is shared, so that the domain specific actions are carried out with shared set of action agents and surface generation is done with a shared set of presentation agents. This processing flow is repeated for every dialogue turn. Next we will describe the central components and explain their functionality.

3.2. Sharing of information – the discourse tree

In Jaspis based applications all the information is kept in the shared Information Storage (IS) that every component has access to. This enables the agents to be stateless, since every piece of information can be retrieved from the IS.

The AthosMail application maintains a discourse tree in the IS regarding to user utterances and system responses. Every user and system utterance is presented as a node in the tree. Each agent that takes part in the processing places its results to the discourse tree and therefore each node is a record of all the information that is needed to carry out dialogue tasks. Most of the agents also read the information that they need from the discourse tree. The discourse tree is not only a placeholder for current state of the dialogue but it also serves as a dialogue history.

Figure 2 depicts the conceptual structure of the discourse tree and the information that is generated there by the agents; associated numbers correspond to modules illustrated in Figure 1.

Shared Information Storage

```
<DIALOGUE>
  ...
  <node>
    <speaker>user</speaker>
    <ASR_result>String representation</ASR_result> (1)
    <ASR_score>Integer value (0-1000)</ASR_score> (1)
    <FDG>The result of functional dependency analysis</FDG> (2)
    <concept_interpretation>Spotted concepts</concept_interpretation> (3)
    <logical_form>Unanchored logical form</logical_form> (4)
  </node>
  <node>
    <speaker>system</speaker> (5 || 9)
    <goal>Propositional list of dialogue solutions</goal> (5)
    <action>Abstract utterance and/or action plan</action> (7)
    <user_modelling>Explicity value for the utterance</user_modelling> (8)
    <output_event>Abstract utterance plan</output_event> (9)
    <surface_form>Realised surface form</surface_form> (10)
  </node>
</DIALOGUE>
```

Figure 2: The discourse tree

4. Dialogue Management in AthosMail

Dialogue management tasks in AthosMail are distributed to several managers that represent the high-level distribution of tasks that are done. Next we describe how the tasks are handled in AthosMail.

4.1. Input interpretation

The processing of user inputs is done in two different ways in AthosMail. The conceptual interpretation and semantic analysis are done in parallel and regardless of each other. Next we will shortly describe the main features of these methods. In the DUMAS project there have been also experiments with semantic templates [2].

4.1.1. Concept-based interpretation

Concept-based interpretation is done using a keyword recognition method. Concept agents try to identify predefined and dynamically constructed concepts from the recognized user inputs (speech and DTMF), and resolve their relations. The resulting concepts are language independent, which makes the use of language independent dialogue agents possible.

4.1.2. Semantic analysis

The logical-form-building agent (LFB), based on the Parasite system [8], [9], reads the functional dependency analysis of the user's utterance from the information storage, transforms each token into an HPSG-like sign, and assembles the list of signs into one sign that spans the utterance. LFB then uses compositional semantics and property theory to produce a logical form that represents the propositional content of the utterance as well as the explicit utterance type (statement, command, query, wh-query).

LFB is a Prolog program with a Java wrapper that interacts with AthosMail, reads from and writes to the information storage, and translates the XML form of the FDG analysis into a Prolog data structure.

4.2. Dialogue decisions

Currently AthosMail contains two ways of making dialogue decisions: specialized dialogue agents that act regarding to the recognized concepts, and the Response Planner agent that does the reasoning based on the goals produced by the PDTREE agent. Both approaches produce an abstract suggestion on what the response to the user should be. These suggestions are realized by a set of action agents that carry out domain specific tasks. The suggestions are evaluated and the best matching one is selected.

4.2.1. Specialized dialogue agents

The specialized dialogue agents represent application functionality. They act regarding to the conceptualized form of the user input. In addition, some dialogue agents take turn regarding to the internal state of the system. For example, if the user has given both the user code and the password, and the mailbox has not been retrieved, a dialogue agent is selected that consults the offline component and informs the user of this action. As an output the dialogue agents produce an abstract output request called an event.

AthosMail contains roughly one agent per high-level application function. For example, the application contains an agent that responds when a call is received, and an agent for giving context-sensitive help. The dialogue agents are implemented using the object-oriented software development paradigm, and they utilize inheritance to separate generic dialogue management from domain specific actions. This is motivated by previous research [7]. In AthosMail we have successfully re-used and inherited agents from a previous e-mail application [11]. In addition, dialogue agents from the underlying Jaspis system architecture are used to handle generic and domain independent dialogue tasks (e.g., to accept incoming calls and handle certain error situations).

4.2.2. Pragmatic processing

The pragmatic agent PDTREE (like LFB, based on Parasite [8], [9] and written in Prolog with a Java wrapper) maintains a discourse model which represents the entities in the discourse and facts about them. PDTREE is initialized with a summary of the mailbox: entities for the messages and persons in the mailbox and facts such as sender, date, read/unread, etc.

For each user utterance, PDTREE anchors the logical form from LFB by dereferencing referring expressions where possible to suitable entities existing in the model and by creating new entities needed to accommodate some expressions. PDTREE also identifies the utterance type from the logical form and uses its theorem prover to find solutions to questions (e.g. the messages that answer the user's question "What new messages do I have from Bob?") or to postulate an event that the user requests ("Delete that message"). PDTREE's output (to the information storage) represents the user's goal and the solutions.

The response planning agent RPLANNER interprets the goal from PDTREE and information from the user model using a set of declarative rules to produce a plan that describes the system's utterance as well as any appropriate non-verbal actions (such as deleting a message). [3]

RPLANNER's plan and the events produced by specialized dialogue agents are realized with a set of action agents, which are described next.

4.3. Domain specific actions

The action agents running under the Action Manager carry out domain dependent tasks that the specialized dialogue agents and the Response Planner agent suggest. Their responsibilities are: to do the needed actions to the IS (e.g. mailbox), communication with the offline system (MailServer), and conveying the dialogue suggestions to the input and output modules.

Each system action is presented in a separate agent. For example, there is an agent that updates the current chapter when a message is being read to the user, and an agent for retrieving messages from MailServer. The action agents extend the specialized dialogue agents. Inheritance is used to maximize efficiency and to separate domain specific tasks from generic ones. A similar approach is used in [7].

4.4. Surface generation

The responsibility of the presentation agents is to generate surface forms from the abstract output requests produced by action agents. AthosMail uses a number of independent agents that respond to one or more conceptual output request(s) to produce the system’s utterances. Proper agents are selected for each abstract output request to produce the surface form regarding to parameters, such as the language and amount of guidance in the surface form.

5. Discussion

By using distributed agents we have achieved the simplicity of functionality oriented approach, and we are still able to cope with a wide range of rich user inputs by using a linguistically motivated semantic analysis of the user input. Distributed approach enables the linguistically oriented agents to concentrate only on their primal purpose, and let more application oriented tasks, such as handling of generic error situations, reading of e-mail messages and interpretation of other modalities (in this case DTMF commands) to be done in other ways. Distribution enables the use of many approaches to figure out the user's intentions, which is beneficial when working with imperfect technologies.

One of the main benefits of the distributed approach is the strong support for multilingual applications. By combining both language independent and language dependent agents we can achieve truly multilingual applications. For example, one instance of AthosMail is able to handle multiple languages by selecting agents dynamically using the generic evaluation method provided by the Jaspis architecture.

In general, the distributed model could be used to benefit from different dialogue management strategies, where one set of agents handles user initiated dialogues and another set when the system needs to take the initiative. In AthosMail this is used with functions that are considered critical, such as deleting. In these cases the systems confirms the action. In addition AthosMail takes the initiative when silence or clearly partial recognition results are encountered. Otherwise, the dialogue with AthosMail is user initiated. However, system utterances are adapted to user's experience level by varying the amount of hints on what can be said in surface forms.

In addition to single domain applications, distributed design of dialogue components could benefit the design of multi-domain applications. In these applications one set of components handles the dialogues related to one domain, another set for some other domain and a separate set of components is used for overall coordination. With proper system architecture, such as the Jaspis architecture used in AthosMail, these components can be distributed to different computers and platforms. This all means better re-usability of existing components. With a distributed structure it is possible to re-use components and even complete systems.

There are also disadvantages when various components are combined to work on the same task. One obvious drawback is efficiency. We have addressed this problem using inheritance to reduce the number of object instances in the runtime application. In addition, the system level evaluation method can reduce the costs considerably when a proper set of evaluators is used.

6. Conclusions

We have presented a method to integrate different methods to manage spoken dialogue applications with a distribution of tasks. Using the approach presented it is possible to process information to be used for the same goal in parallel.

The Jaspis architecture made the reusability of existing components from various sources possible. The use of system level evaluation method and shared information management was found especially feasible. They made the integration and reuse of different components easy. Furthermore, adding altogether new ways to handle the dialogue was found efficient, and the benefits of several different approaches were realized in a single multilingual application. Preliminary results of user study are promising in means of system acceptability and user satisfaction [4].

7. References

[1] Boye, J., Wiren, M., Rayner, M., Lewin, I., Carter, D. & Becket, R., "Language-Processing Strategies and Mixed-Initiative Dialogues", *Proc. of Workshop on Knowledge and Reasoning in Practical Dialogue Systems (IJCAI-99)*, 1999.

[2] Cheadle, M. & Gambäck, B., "Robust Semantic Analysis for Adaptive Speech Interfaces", *Proc. of Universal Access in HCI: Inclusive Design in the Information Society*, 2003.

[3] Funk, J. A., "Management and modelling of discourse in the DUMAS project", *7th Annual Computational Linguistics in the UK Research Colloquium (CLUK),* 2004.

[4] Hartikainen, M., Salonen, E-P. & Turunen, M., "Subjective Evaluation of Spoken Dialogue Systems Using SERVQUAL Method", *Proc. ICSLP 2004*, 2004.

[5] Larsen, L. B., "A Strategy for Mixed-Initiative Dialogue Control", In *Proc. of Eurospeech -99,* 1999.

[6] McTear, M., Allen, S., Clatworthy, L., Ellison, N., Lavelle, C. & McCaffery, H., "Integrating Flexibility into a Structured Dialogue Model: Some Design Considerations", *Proc. of ICSLP 2000*, 2000.

[7] O'Neill, I. M. & McTear, M. F., "Object-Oriented Modelling of Spoken Language Dialogue Systems", *Natural Language Engineering*, *Best Practice in Spoken Language Dialogue System Engineering, Special Issue, 6, 3,* 2000.

[8] Ramsay, A. & Seville, H., "Models and Discourse Models", *Journal of Language and Computation, 1, 2,* 2000.

[9] Ramsay, A., *The logical structure of English: computing semantic content*, London: Pitman, 1990.

[10] Turunen, M. and Hakulinen, J., "Jaspis - A Framework for Multilingual Adaptive Speech Applications", *Proc. of ICSLP 2000*, 2000.

[11] Turunen, M. & Hakulinen, J., "Mailman - a Multilingual Speech-only E-mail Client based on an Adaptive Speech Application Framework", *Proc. of Workshop on Multi-Lingual Speech Communication (MSC 2000)*, 2000.

[12] Turunen, M., Salonen, E-P., Hartikainen, M., Hakulinen, J., Black, W.J., Ramsay, A. Funk, A., Conroy, A., Thompson, P., Stairmand, M., Jokinen, K., Rissanen, J., Kanto, K., Kerminen, A., Gambäck, B., Cheadle, M., Olsson, F., Sahlgren, M. 2004. "AthosMail – a multilingual Adaptive Spoken Dialogue System for E-mail Domain" *Proc. of the COLING Workshop Robust and Adaptive Information Processing for Mobile Speech Interfaces*, Geneva, Switzerland.

[13] Turunen, M., Salonen, E-P., Hartikainen, M., Hakulinen, J., "Robust and Adaptive Architecture for Multilingual Spoken Dialogue Systems", *Proc. ICSLP 2004*, 2004.