Proceedings of the International Joint Conference IBERAMIA/SBIA/SBRN 2006 - 4th Workshop in Information and Human Language Technology (TIL'2006), Ribeirão Preto, Brazil, October 23–28, 2006. CD-ROM. ISBN 85-87837-11-7

A Domain Knowledge Advisor for Dialogue Systems

Porfírio Filipe[1, 2] and Nuno Mamede[1, 3]

[1] L[2]F INESC-ID, Spoken Language Systems Lab, Lisbon, Portugal
{porfirio.filipe, nuno.mamede}@l2f.inesc-id.pt
http://www.l2f.inesc-id.pt/
[2] ISEL, Instituto Superior de Engenharia de Lisboa, Lisbon, Portugal
[3] IST, Instituto Superior Técnico, Lisbon, Portugal

Abstract. This paper describes ongoing research in order to enhance our Domain Knowledge Manager (DKM) that is a module of a multi-propose Spoken Dialogue System (SDS) architecture. The application domain is materialized as an arbitrary set of devices, such as household appliances, providing useful tasks to the SDS users. Our main contribution is a DKM advisor service, which suggests the best task-device pairs to satisfy a request. Additionally, we also propose a DKM recognizer service to identify the domain's concepts from a natural language request. These services use as knowledge source a domain model, to obtain knowledge about devices and the tasks they provide. The implementation of these services allows the DKM to provide a high-level and easy to use small interface, instead of a conventional service interface with several remote procedures/methods. These services have been tested into a domain simulator. Our contributions try to reach SDS domain portability issues.

1 Introduction

This paper describes our contributions to enhance a Domain Knowledge Manager (DKM) that is a module of a generic multi-propose Spoken Dialogue Systems (SDS) architecture [1][2].

Our main contribution is an advisor service that suggests the best task-device pairs to satisfy a request formalized in a list of domain's concepts. Additionally, we also propose a recognizer service to identify the domain's concepts from a natural language request.

We propose these services to be built-in the DKM, which handles a domain model that includes representations of devices and the tasks they provide, such as household appliances. The implementation of these services allows the DKM to offer a high-level and easy to use small interface.

2 Background

SDSs have been defined as computer systems with which humans interact on a turn-by-turn basis and in which spoken natural language plays an important part in

the communication [3]. The main purpose of a SDS is to provide an interface between a user and a computer-based application such as a database or expert system. There is a wide variety of systems that are covered by this definition, ranging from question-answer systems that answer one question at a time to "conversational" systems that engage in an extended conversation with the user.

Traditionally, the SDSs have been built by expert developers, with hand-crafting of domain-specific knowledge and functionality [4]. As the underlying technologies matured, the scientific community became increasingly interested in making these systems portable and configurable by novice developers.

Only in the last decade, with major advances in speech technology, have large-scale working systems been developed and, in some cases, introduced into commercial environments. Nevertheless, many implementations of Dialogue Managers (DM) perform input interpretation, output generation, and domain dependent tasks. This approach may easily lead to situations in which the DM is a monolithic component. Monolithic components make it harder to build modular, distributed systems, and reusable components.

Fig. 1 shows a modular SDS architecture where the DM is the main actor, used to coordinate SDS's modules: Speech Recognition, Language Understanding, External Communication, Response Generation, and Speech Output (see McTear [5] for a survey).

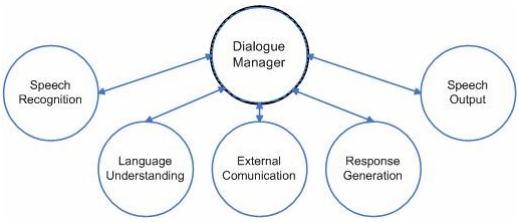

Fig. 1. Architecture for Spoken Dialogue Systems.

The role of the DM module differs slightly between different SDS architectures, but its primary responsibility is to control the flow of the dialogue by deciding how the system should respond to a user utterance. This is done by inspecting and contextually specifying the information produced by the Language Understanding module. If some information is missing or a request is ambiguous, the DM specifies clarification questions that are posed to the user. When the request is completed and unambiguous, the External Communication module access, classically, a background system and an answer is produced. As a basis for this process, the DM has a dialogue model, a dialogue history, and a domain model that typically includes a task model.

Quesada et al. [6] describe a SDS in the D'Homme project that is specifically designed for interacting with a background system composed by a set of devices including household appliances.

A quantity of commercial systems is being built to handle dialog with household appliances. Some examples include the Linguamatics Automated House, the SmartKom Home/Office, the Fluency House, and Voxi Smart Homes. As these are commercial systems, they do not report vital information about their mechanisms.

3 Portability

One of our first steps towards portability of SDSs was introduced in [7] enabling plug-and-play reconfiguration and modularity. Our recent research effort identifies the need for a more flexible and easy to use way to access the knowledge of a SDS application domain [8].

In order to develop a DM that easily can be customized to new domains and in which different dialogue strategies can be explored, the DM should only be concerned with phenomena related to the dialogue with the user. It should not be involved in the process of accessing a background system or performing domain reasoning. A separate module, the DKM, should carry out these features. A DKM is in charge for retrieving and coordinating knowledge from the different domain knowledge sources and application systems naturally named background system. The DM can deliver a request to the DKM and in return expects an answer retrieved from the background system. If a request is under-specified or contains inconsistencies from the DKM's point of view, a specification of what clarifying information is needed will be returned to the DM.

In this context, the use of a SDS architecture that includes a DKM seems to be a key enabler to achieve domain portability and easy configuration [9]. Fig. 2 shows the generic SDS architecture diagram used as reference to report our work. In this diagram, X_n represents a generic device and A, B, C, and D are other modules of the SDS.

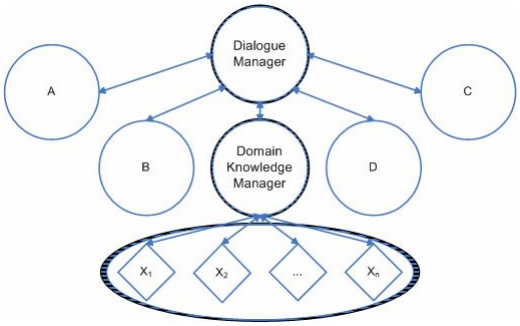

Fig. 2. Generic Architecture for Spoken Dialogue Systems.

This diagram shows the DKM working as a broker interacting with a set of heterogeneous devices. First, the user's request is processed using the recognizer service to identify the list of domain's referred concepts. Second, the DM accesses the DKM for obtaining the appropriate task-device pair. For this, the DKM uses the advisor service to deliver the best task-device pairs. Finally, when the request is completed and unambiguous the DM calls the DKM that invokes the selected task supported by the associated device.

4 Domain Model

This section tries to give an overview of the most relevant components of the domain model used as knowledge source by the advisor and recognizer services. For the sake of space, a detail description of content of the components will be omitted.

The main goal of the DKM is to support the communication interoperability between the SDS and the set of heterogeneous devices. To achieve this goal, the DKM includes a domain model with three independent knowledge components: DISCOURSE model, WORLD model, and TASK model. This domain model architecture was adapted from Unified Problem-solving Method Development Language (UPML) [10] and is explained in [11] [12].

DISCOURSE Model

A concept is an atomic knowledge unit. The DISCOURSE model defines a conceptual support, grouping concept declarations, used to describe device classes, devices, and the tasks they provide.

The concepts are organized in four groups of collections. The general group maintains all the collections. The task group contains two collections action and perception that holds the task names. The quantity group contains two collections number (integer, real, positive, integer, …) and measure (time, power, …). The attribute group contains collections of concepts that are usually attributes (color, shape, texture, …). The artifact group contains the set of device/artifact classes (artifact, equipment, application, furniture, appliance, …) that can by referred in the type hierarchy.

In order to guarantee the availability of vocabulary to designate the domain's concepts, the concept declarations include linguistic resources. This approach tries to reach the ubiquitous essence of natural language. Although, the coverage of handmade resources such as WordNet [13] in general is impressive, coverage problems remain for applications involving specific domains or multiple languages.

Unlike a terminology-inspired ontology [14], concepts are not included for complex terms (word root or stem) unless absolutely necessary. For example, an item such as "*the yellow house*" should be treated as an instance of a "*house*", having the color "*yellow*" without creating a new concept "*yellow house*".

A linguistic descriptor linked to a concept declaration holds the list of terms or more generically, the list of Multi-Word Unit (MWU) [15]. A MWU list contains linguistic variations used to refer a concept, such as synonymous or acronyms. Each word, has a part of speech tag, such as noun, verb, adjective or adverb; a language tag, such as "pt", "br", "uk" or "us"; and a group of selected phonetic transcriptions. For instance, if the language tag of a word is "pt" its phonetic transcription is encoded using the Speech Assessment Methods Phonetic Alphabet (SAMPA) for European Portuguese [16].

The concept declaration can also refers optionally semantic resources. A semantic descriptor has references to internal or external knowledge sources, for instance, an ontology or a lexical database, such as WordNet. The knowledge sources references must be unique, because we cannot have in the domain model same meaning twice.

The references to knowledge sources must be encoded using a data format allowing a unique identification of the concept in the knowledge source. The data format of

the knowledge source reference do not need to be universal, it is enough to keep the same data format for a particular knowledge source. We recommend the use of a generic Uniform Resource Identifier (URI) format to encode the references to knowledge sources.

WORLD Model

The WORLD model has two components: type hierarchy and mediator. The type hierarchy organizes device/artifact classes, for instance in a home environment, device class may be either an appliance, or a light, or a window, or a table. The mediator manages device instances linked to their classes.

TASK Model

The TASK model contains task descriptors (T_n) that are associated to device (X_n) instances through links ($T_n \Leftrightarrow X_n$). We consider two kinds of tasks: action and perception. A perception task cannot modify the state of the device, on the other hand an action task can. A task descriptor is a semantic representation of a device capability and has a name and optionally an input list, an output list, and assumptions. The task name is a concept from the predefined task group of concepts. Table 1 depicts a task descriptor where the "*" means mandatory fulfilling.

Table 1. Device Task Descriptor

Slot			Value
name*			ID-Task
input list	**input role**	**name***	ID-Attribute
		range*	ID-Attribute
		restriction	*rule*
		default	ID-Value
	other input roles …		
	pre-condition		*rule*
output list	**output role**	**name***	ID-Attribute
		range*	ID-Attribute
	other output roles …		
	pos-condition		*rule*
assumptions	**initial condition**		*rule*
	final condition		*rule*

The *input list*, that describes the task input parameters, has a set of optional input roles. An *input role*, that describes one input parameter, has a *name*, a *range*, a *restriction*, and a *default*. The *name* and *range* are concepts from the attribute group. The *restriction* is a rule that is materialized as logical formula and is optional. The *range* rule and the *restriction* rules define the set of allowed values in task parameters. For instance, if the range is a positive integer and we want to assure that the parameter is greater than 5, then we must indicate the restriction rule: "name > 5". The *default* optional slot of the input role is a concept member of the quantity group. If the default is not provided the input role must be filled.

The *output list*, that describes the output parameters, has a set of optional output roles. An *output role*, which describes one output parameter, is similar to an *input role* without *restriction* rule and *default*.

The rules in the *task descriptor* allow three kinds of validation: *restriction* rule to perform individual parameter validation, *pre-condition* to check input parameters before task execution, and *pos-condition* to check output parameters after task execution. *Restriction* can refer the associated input role, *pre-condition* can refer task input role names and *pos-condition* can refer output role names. *Assumptions* perform state validation: the *initial condition* (to check the initial state of the world before task execution) and the *final condition* (to check the final state of the word after task execution). *Assumptions* can refer role names and results of perception task calls.

The state of the world is composed by all device states. The state of each device is obtained by calling the provided perception tasks. For instance, if the request is "*switch on the light*", we have to check if the "*light*" is not already "*switched on*" and after the execution, we have to check if the "*light*" has really been "*switched on*".

5 Advisor

This section describes the DKM advisor service, proposed to suggest the best task-device pairs to satisfy a request formalized in a list of domain's concepts. The ideas behind this service are based on the relative weight of each concept that figures in the request. Two independent ranking, for tasks and devices, support the suggestion for the best task-device pairs.

The ranking points are determined considering three heuristic values: *nABase*, *nTBase*, and *nTUnit*. The *nABase* value is determined by the maximum height of the domain model type hierarchy plus (1) one. The *nTBase* value is determined by the maximum number of task roles (arguments) plus (1) one. The *nTUnit* value is constant and equal to 3 (three) that are the number of ways to reference a task role (by *name*, *range*, or *value*). The advisor service uses as input a list of pivot concept. The pivot concepts references to tasks and devices are converted, following the next rules, into points that are credited to the respective device or task rank.

The rank of a device is modified according to the rules:

1. If the pivot concept refers a device name, the value *nABase*2* is credited in the respective device rank;
2. If the pivot concept refers a device class name, the value *nABase* is credited in the respective device rank;
3. If the pivot concept refers a device super-class name, the value *nABase-n* is credited in the respective device rank, where *n* is determined by the number of classes (in the type hierarchy), between the device class and the referred super-class.

The rank of a task is modified according to the rules:

4. If the pivot concept refers a task name, the value *nTBase*nTUnit* is credited in the respective task rank;
5. If the pivot concept refers a task role name or a task role range, the value *nUnit/2* is credited in the respective task rank;
6. If the pivot concept refers a task parameter, the value *nTUnit/3* is credited in the respective task rank.

Finally, the task-device pairs are composed selecting the tasks with the best rank and the devices, which provide the tasks, with the best rank.

6 Recognizer

This section describes an additional DKM service, proposed to recognize the domain's concepts from a natural language request. The recognizer service receives a request and split its words into groups, trying to obtain a match against the linguistic descriptors in the domain model.

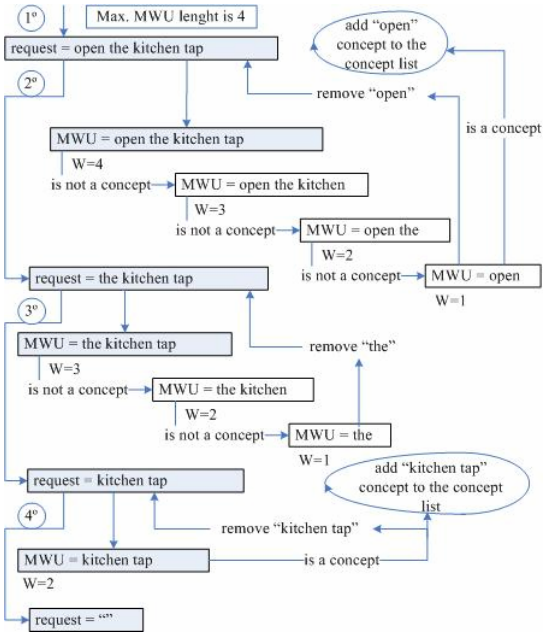

Fig. 3. "*open the kitchen window*".

Fig. 3 shows a schema with four iterations involved in the processing of the request "*open the kitchen tap*".

The list of recognized concepts can be directly used by the DM to fill the list of pivot concepts indicated as input to the advisor service. However, the DM can remove or add new concepts into the pivot concept list, according to its own dialogue strategies or knowledge sources, for instance the dialogue history.

A better version of the recognizer service can also accept annotated natural language requests, including part of speech tags and specific phonetic transcriptions of some words, in order to solve potential linguistic ambiguities.

The words of the request are grouped from the left to the right. Each group of words is processed in several interactions. The first step of an iteration uses a group of W words (MWU candidate). The second step uses a group of W-1 words, and so on. The maximum length of a group of words, represented in the domain model, determines the value of W. When a group of words matches a MWU in a concept linguistic descriptor, the concept is recognized and these words are removed form the request. When the group as only one word the word is removed form the request. The process stops when the request is empty.

7 Examples

Our current research is supported by our own environment simulator, in which we are using and testing the proposed services. This simulator allows the debug of an invoked task and the simulation of the interaction with a particular device. With this simulator, we can activate and deactivate devices, execute the tasks, obtain the answers and observe the behavior of the devices. We can also consult and print several data about the device interfaces.

Fig. 4 and 5 are screenshots (windows) of our home environment simulator, developed originally for Portuguese users.

Fig. 4. Water Faucet Simulator.

Fig. 5. Microwave Oven Simulator.

Fig. 4 presents a water faucet (tap) simulator where the selected temperature is 30 ºC and the water volume is 25 %. Fig. 5 shows a microwave oven simulator configured to defrost codfish, where the selected cooking period is 8 minutes and the power is set to defrost. According to the knowledge represented into the domain model of the simulator, we have determined the value for *nABase* that is equal to 5 and the value for *nTBase* that is equal to 4.

Fig. 6 and 7 illustrate the processing of two requests combining the use of the recognizer service with the use of the advisor service.

Fig. 6 shows a diagram where is illustrated the processing of a simple request "*defrost codfish in microwave oven*". The concept recognizer service identifies three concepts: 57 - "*defrost*", 16 - "codfish", and 91 - "*microwave oven*". In order to determine the best task-device pair, the advisor service applies its rules to each one of the pivot concepts. Rule 4 applied to concept, with the ID 57, adds 12 points in the rank of the task, with the ID 100029. Rule 6 applied to concept, with the ID 16, adds 1 point in the rank of the same task. Finally, Rule 2 applied to concept, with the ID 91, adds 5 points in the rank of the device, with the ID 9. The reference to the device class "microwave oven" becomes a reference to the device with the name "mmoulinex", because we have only this microwave oven represented in the domain model.

Fig. 7 shows a diagram where is illustrated the processing of another request "*increase 5 ºC on kitchen tap*". The concept recognizer service identifies four concepts: 13 - "*increase*", 100057 - "5", 105 - "ºC" (Celsius degrees), and 100002 - "*kitchen tap*". Rule 4 applied to concept, with the ID 13, adds 12 points in the rank of the task, with the IDs 100008 and 100010. Rule 6 applied to concept, with the ID 100057, adds 1 point in the rank of the task, with the ID 100010. Rule 5 applied to concept,

with the ID 100002, adds 1.5 points in the rank of the tasks, with the IDs 100008 and 100004. Finally, Rule 1 applied to concept, with the ID 100002, adds 10 points in the rank of the device, with the ID 2. This example, demonstrates the use of the advisor service to select the right task (100010) among others (100004, 100008, …).

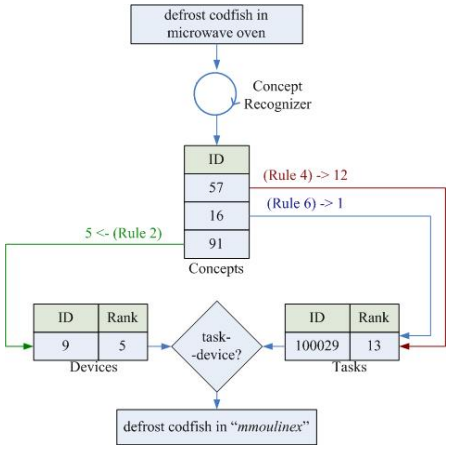

Fig. 6. "*defrost codfish in microwave oven*".

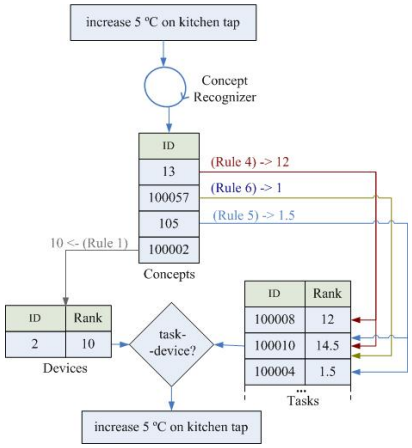

Fig. 7. "*increase 5 ºC on kitchen tap*".

8 Concluding Remarks and Future Work

The work reported in this paper is a significant contribution to improve the flexibility, and simultaneously the portability, of the SDS multi-propose architecture being developed in our lab. In this paper, we have devised two services to enhance a DKM in order to offer a high-level easy to use small interface, instead of a conventional service interface with several remote procedures/methods.

The SDS research issues should be specified according to the purpose for which the SDS is intended. If the goal is to make the system work in the field, then performance and real time operation become key factors, and the dialogue manager should drive the user to speak in a constrained way. Under these circumstances, the interaction model will be simple. In this context, the employ of the advisor service will be a bit inefficient because of the processing time wasted applying all the advisor rules to all pivot concepts. In order to get better the performance of the advisor service should be used an index table to keep the advisor rules results. For each concept, an index table maintains the points to credit to a specific device or task rank.

Our approach simplifies the interface of the DKM, which is an important feature to split dialogue (linguistic) and domain issues. We have presented this subject focusing on examples, according to the domain model maintained into our simulator. The presented ideas have been applied, with success, in a domain materialized as a set of heterogeneous devices that represents a home environment.

As future work, we expect to explorer these ideas, more deeply, applying them into a richer domain model trying to cover new perspectives.

Acknowledgments. This work is partially supported by GIATSI project "*Integração Dinâmica de Dispositivos em Ambientes Inteligentes*" (ID^2AI).

References

1. Neto, J., Mamede, N., Cassaca, R. and Oliveira, L.: The Development of a Multi-purpose Spoken Dialogue System, 8th European Conference on Speech Communication and Technology, Geneva, Switzerland (2003)
2. Glass, J., Weinstein, E., Cyphers, S., Polifroni, J., Chung, G., Nakano, N.: A Framework for Developing Conversational User Interfaces, 5th International Conference on Computer--Aided Design of User Interfaces, Funchal, Madeira Island, Portugal (2004)
3. Fraser, N.: Assessment of Interactive Systems, Handbook of Standards and Resources for Spoken Language Systems, D. Gibbon, R. Moore, and R. Winski, Eds. Mouton de Gruyter, New York, NY, 564–614 (1997)
4. Polifroni, J., Chung, G.: Promoting Portability in Dialogue Management, 7th International Conference on Spoken Language Processing, Denver, Colorado (2002)
5. McTear, M.: Spoken Dialogue Technology: Towards the Conversational User Interface, Springer Verlag, ISBN: 1-85233-672-2 (2004)
6. Quesada, J., Garcia, F., Sena, E., Bernal, J., Amores, G.: Dialogue Managements in a Home Machine Environment: Linguistic Components over an Agent Architecture, SEPLN, 89-98, (2001)
7. Filipe, P., Mamede, N.: Towards Ubiquitous Task Management, 8th International Conference on Spoken Language Processing, Jeju Island, Korea (2004)
8. Filipe, P., Mamede, N.: A Task Repository for Ambient Intelligence, (to appear) 11th International Conference on Applications of Natural Language to Information Systems, Klagenfurt, Austria (2006)
9. Flycht-Eriksson, A., Jönsson, A.: Dialogue and Domain Knowledge Management in Dialogue Systems, 1st SIGdial Workshop on Discourse and Dialogue, Hong Kong (2000)
10. Fensel, D., Benjamins, V., Motta, E., Wielinga, B.: UPML: A Framework for Knowledge System Reuse, 16th International Joint Conference on Artificial Intelligence. Stockholm, Sweden (1999)
11. Filipe, P., Mamede, N.: Ubiquitous Knowledge Modeling for Dialogue Systems, (to appear) 8th International Conference on Enterprise Information Systems, Paphos, Cyprus (2006)
12. Filipe, P., Mamede, N.: A Framework to Integrate Ubiquitous Knowledge Modeling, (to appear) 5th International Conference on Language Resources and Evaluation, Genoa, Italy (2006)
13. Fellbaum, C. (editor): WordNet: An Electronic Lexical Database, MIT Press (1998)
14. Gruber, T.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing, International Workshop on Formal Ontology, Padova, Italy (1992)
15. Daille, B., Gaussier, E., Lange, J.: Towards Automatic Extraction of Monolingual and Bilingual Terminology, 15th International Conference on Computational Linguistics, Kyoto, Japan, 515-521 (1994)
16. SAMPA (SAM Phonetic Alphabet), Spoken Language Systems Lab (L^2F), http://www.l2f.inesc-id.pt/resources/sampa/sampa.html