AN ADAPTIVE DOMAIN KNOWLEDGE MANAGER FOR DIALOGUE SYSTEMS

Porfírio Filipe[1, 2], Luís Morgado[2, 4] and Nuno Mamede[1, 3]
[1] *L²F INESC-ID - Spoken Languages Systems Laboratory, Lisbon, Portugal*

[2] *GIATSI, ISEL – Instituto Superior de Engenharia de Lisboa, Lisbon, Portugal*
[3] *IST – Instituto Superior Técnico, Lisbon, Portugal*
[4] *LabMAG, FCUL – Faculdade de Ciências da Universidade de Lisboa, Lisbon, Portugal*

Keywords: Human–Computer Interaction, Spoken Dialogue System, Dialogue Management, Domain Model.

Abstract: This paper describes the recent effort to improve our Domain Knowledge Manager (DKM) that is part of a mixed-initiative task based Spoken Dialogue System (SDS) architecture, namely to interact within an ambient intelligence scenario. Machine-learning applied to SDS dialogue management strategy design is a growing research area. Training of such strategies can be done using human users or using corpora of human computer dialogue. However, the size of the state space grows exponentially according to the state variables taken into account, making the task of learning dialogue strategies for large-scale SDS very difficult. To address that problem, we propose a divide to conquer approach, assuming that practical dialogue and domain-independent hypothesis are true. In this context, we have considered a clear separation between linguistic dependent and domain dependent knowledge, which allows reducing the complexity of SDS typical components, specially the Dialoguer Manager (DM). Our contribution enables domain portability issues, proposing an adaptive DKM to simplify DM's clarification dialogs. DKM learns, through trial-and-error, from the interaction with DM suggesting a set of best task-device pairs to accomplish a request and watching the user's confirmation. This adaptive DKM has been tested in our domain simulator.

1 INTRODUCTION

This paper describes our recent effort to improve our Domain Knowledge Manager (DKM) that is part of a mixed-initiative task based Spoken Dialogue System (SDS), such as described by Neto *et al.* (2003), namely to support generic dialogue management strategies within an Ambient Intelligence (AmI) vision (Ducatel et al., 2001), (Filipe and Mamede, 2006b).

A SDS should be a computational entity that allows access to any device by anyone, anywhere, at anytime, through any media, allowing its user to focus on the task, not on the tool. Only in the last decade, with major advances in speech technology, have large-scale working SDS been developed and, in some cases, introduced into commercial environments (McTear, 2004).

Speech-based human-computer interaction faces several challenges in order to be more widely accepted. One of the challenges in dialogue management is to control the dialogue flow (dialogue strategy) in an efficient and natural way. Dialogue strategies designed by humans are prone to errors, labour-intensive and non-portable, making automatic design an attractive alternative. Learning dialogue strategies from real users is a very expensive and time-consuming process, making automatic learning an attractive alternative.

Machine-learning applied to SDS dialogue management strategy design is a growing research area. Training of such strategies can be done using human users or using corpora of human computer dialogue. However, the size of the state space grows exponentially according to the state variables taken into account, making the task of learning dialogue strategies for large-scale SDS very difficult.

In this paper, we propose a divide to conquer approach assuming that practical dialogue and domain-independent hypothesis are true (Allen et al. 2000). The reason is that all applications of human-computer interaction involve dialogue

Filipe P., Morgado L. and Mamede N. (2007).
AN ADAPTIVE DOMAIN KNOWLEDGE MANAGER FOR DIALOGUE SYSTEMS.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - HCI*, pages 45-52
DOI: 10.5220/0002384200450052
Copyright © SciTePress

focussed on accomplishing some specific task. We consider the bulk of the complexity in the language interpretation and dialogue management is independent of the task being performed. In this context, a clear separation between linguistic dependent and domain dependent knowledge allows reducing the complexity of SDS typical components, specially the DM.

Our contribution enables domain portability issues, proposing an adaptive DKM to simplify DMs clarification dialogs, reducing the amount of user's interactions to solve dialogue ambiguities. In this scenario, the DKM learns through trial-and-error from the interaction with the DM, suggesting the best task-device pairs to accomplish a request and watching the user's confirmation.

Summarizing, our contribution enables domain portability issues reducing the amount of user's interactions wasted to solve dialogue ambiguities. In Section 2, we describe some domain portability issues focusing DM and DKM interaction. Section 3 gives an overview of the most relevant components of the domain model built-in the DKM. Section 4 describes the advisor service proposed to enhance the DKM covering performance issues and the adaptive approach. Section 5 makes a description of our domain simulator. Finally, in Section 6, we present concluding remarks and future work.

2 DOMAIN PORTABILITY

This section describes some domain portability issues focusing the DM and DKM interaction.

While some are concerned with the generation of systems from on-line content (Feng et al., 2003), others have addressed portability issues within the DM (Denecke, 2002) and the understanding components (Dzikovska et al., 2003). Nevertheless, many implementations of DMs perform input interpretation, output generation, and domain specific tasks. These tasks are typically domain dependent because its designs consider particular requirements of each domain.

The design of a DM that can be easily customized to new domains and in which different dialogue strategies can be explored, should only concern phenomena related to the dialogue with the user, focusing on dialogue and on discourse strategies. The DM component should not be involved in the process of accessing a background system or performing domain reasoning. For this, should be considered another component of SDS architecture, the DKM, which handles these features (Flycht-Eriksson and Jönsson, 2000). The DKM is in charge for retrieving and coordinating knowledge from the different domain knowledge sources and application systems, traditionally 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 the problem will be returned to the DM that will start a clarification dialogue. In these circumstances, the DKM allows the customization of the DM enabling domain portability and easy configuration of the SDS architecture.

Within AmI vision, the DKM supports the communication interoperability between the SDS and a set of heterogeneous devices. Figure 1 shows the generic SDS architecture diagram used as reference to report our work.

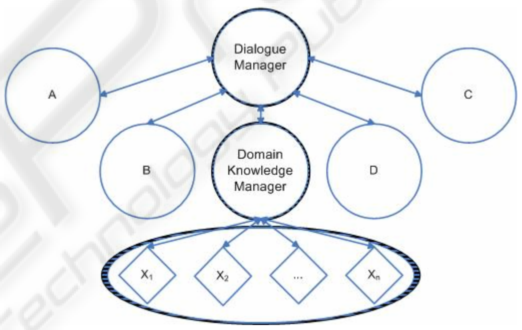


Figure 1: Generic SDS architecture in AmI.

In Figure 1, X_n represents a generic device and A, B, C, and D are other components of the SDS architecture. Figure 1 shows the DKM working as a broker: the DKM uses the advisor service to deliver the best task-device pairs. When the user's request is unambiguous, the DM calls the DKM that invokes the selected task supported by the associated device.

3 DOMAIN MODEL

This section gives an overview of the most relevant components of the domain model built-in the DKM and used as domain knowledge source by the DM.

In order to achieve its 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 the Unified Problem solving Method Development Language (UPML) (Fensel et al., 1999). For the sake of space, a detailed description of the content of the components will be omitted. For a complete description see (Filipe e Mamede, 2006a) and (Filipe e Mamede, 2006c).

3.1 DISCOURSE Model

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

Concept declarations are organized in four groups of collections. The general group maintains all the collections. The task group contains two collections: action and perception that hold 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, 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 (Fellbaum, 1998) in general is impressive, coverage problems remain for applications involving specific domains or when multiple languages are used. Unlike a terminology inspired ontology (Gruber, 1992), 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 the concept "*yellow house*".

A linguistic descriptor linked to a concept declaration holds a list of terms or more generically, a list of Multi Word Unit (MWU) (Daille et al., 1994). 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.

A concept declaration can also refer optionally semantic resources. A semantic descriptor has references to internal or external knowledge sources, for instance, an ontology or a lexical database, such as the WordNet. The knowledge sources references must be unique, because each meaning must be unique.

The references to knowledge sources must be encoded using a non-ambiguous data format. 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.

3.2 WORLD Model

The WORLD model has two components: a type hierarchy module and a mediator module. The type hierarchy organizes the device/artifact classes, for instance in a home environment, the 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.

3.3 TASK Model

The TASK model contains task descriptors (T_n) that are associated to device (X_n) instances through links ($T_n \Leftrightarrow X_n$). 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 …		
	post-condition		*rule*
assumptions	**initial condition**		*rule*
	final condition		*rule*

We consider two kinds of tasks: actions and perceptions. 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 competence and has a name and optionally an input list, an output list, and assumptions. The task name is a concept previously declared in the task group of the DISCOURSE model.

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 declared in the attribute group of the DISCOURSE model. The *restriction* is a rule that is materialized as a logical formula and is optional. The *range* rule and the *restriction* rules define the set of task parameter allowed values. 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 an *input role* is a concept member of the quantity group of the DISCOURSE model. 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 a *restriction rule* and with no *default* value.

The rules belonging to the *task descriptor* allow three kinds of validation rules: (i) *restriction* rules to perform individual parameter validation; (ii) *pre-condition* rules check input parameters before task execution; and (iii) *post-condition* rules check output parameters after task execution. A *restriction* rule can refer the associated *input role*, the *pre-condition* rule can refer task *input role* names and the *pos-condition* rule can refer output role names. An *assumption* performs state validation: the *initial condition* (to check the initial state of the world before a task execution) and the *final condition* (to check the final state of the word after a task execution). The *assumption* rules can refer role names and values returned by 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*".

4 ADVISOR

This section describes an advisor service proposed to suggest the best task-device pairs that satisfy a request formalized in a list of domain's concepts. This section also covers performance issues and the adaptive approach presenting local examples.

In order to allow flexible behaviour in user interaction, a DM must be able to handle under-specified requests (the user provides only partial information). The DM must decide which task should be performed. Our approach is to use domain dependent information, for instance, in a particular domain the user's preferred choice may be "*turn-on the light*" for the request "*turn-on*", but in another domain without lights, the best choice is certainly different.

Since the DM does not know the domain model, the DM submits a list of pivot concepts that represent a request and the advisor service returns the best task-device pair. This service allows the DKM to offer a high level and easy to use small interface instead of a conventional service interface with several remote procedures/methods.

4.1 Advisor Basic Algorithm

The ideas behind this service are based on the relative weight (relevance) of each concept that are present in the request. Two independent ranking, for tasks and devices, are used to compute 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 one. The *nTBase* value is determined by the maximum number of task roles (arguments) plus one. The *nTUnit* value is constant and equal to 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 into points that are credited in the respective device or task rank.

The rank of a device is modified according to the following 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 following 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, task-device pairs are composed selecting the tasks with the best rank and the devices, which provide the tasks, with the best rank.

4.2 Performance

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 robust performance and real time operation become key factors, and the DKM should drive the user to speak in a constrained way. Under these circumstances, the interaction model will be simple. In this context, the advisor service use will be inefficient because of the processing time wasted applying all the advisor rules to all pivot concepts.

In order to improve the advisor service performance we propose the use of an index table to keep the advisor rules results. For each concept that figures in the DISCOURSE model, an index table maintains the points credited to each specific device and task.

Table 2: Index table example.

Concept ID	Device ID	Points	Task ID	Points
13	-	-	100008	12
13	-	-	100010	12
16	-	-	100029	1
57	-	-	100029	12
91	9	5	-	-
105	-	-	100010	1.5
105	-	-	100004	1.5
100002	2	10	-	-
100057	-	-	100010	1
…	…	…	…	…

Table 2 presents an example of an index table. For instance, Rule 6 always adds 1 point to tasks, IDs 100029 and 100010, referred by concepts, IDs 16 and 100057.

An index table can be build at design time or at runtime. When the index table is build at runtime we propose an incremental approach, if a pivot concept does not exists in index table all the advisor rules are applied to the concept provoking an index table update. The advisor service, instead of applying all the rules for each concept, consults the index table to obtain the respective points. After, the points are credited in the respective device or task rank.

4.3 Adaptive Approach

In some cases the advisor basic algorithm conduces to situations where the tasks and/or devices have similar or even the same ranking points. In this situation, the best task-device pair is not clearly determined, demanding for a clarification dialogue.

One way to address this problem is by endowing the DM with the ability to interactively learn dialog strategies, namely by using reinforcement learning approaches (Henderson et al., 2005), (Schatzmann et al., 2006). However, although this kind of approaches present interesting characteristics in what concerns learning of dialog strategies, they suffer from well-known drawbacks for online operation, especially in what concerns the number of interactions needed for convergence, which restricts their application mainly to offline processing. On contrary, in online dynamic environments, the user interactions are relatively scarce and the SDS must be able to adapt its operation taking advantage of these limited interactions.

The main focus of our adaptive approach is not on learning complex dialog strategies, but on dynamically adapting the relevance of task-device pairs according to user interaction, watching the selection or rejection expressed in previous user's clarification dialogues. As an example, lets consider the case where the user is in the kitchen and selects the task "*turn-on*". Since "*turn-on*" must refer to some device, SDS asks the user to specify the device he/she wants to refer (e.g., a microwave oven, or the ceiling lights).

Therefore, the SDS must decide about which devices it should ask the user first. Consequently, some form of device relevance is needed to enable the selection of devices according to the selected task and the previous history of user interaction.

4.4 Dynamic Adaptation of Device Relevance

To allow for a dynamic adaptation of device relevance, we propose a model where device relevance is seen as a form of activation potential, following the *Agent Flow Model* proposed by Morgado & Gaspar (2004).

The idea is "*a fixed amount of activation potential is shared by all the devices associated to a task*". To reflect the fact that a device d_s was selected for a task t, some amount of activation potential will flow into it. However, since the total activation potential is fixed, this flow is provided by each non-selected device d_{ns} in the proportion of their actual activation potential. Formally, the activation flow $\varphi_\tau(t,d_s) \in \Re$ is defined as:

$$\varphi_\tau(t,d_s) = \alpha(\varepsilon^{\max} - \varepsilon_\tau(t,d_s)) \qquad (4.4.1)$$

where $\varepsilon_\tau(t,d_s) \in \Re$ represents the activation potential of device d_s relative to task t at some discrete time instant $\tau \in \aleph$, $\varepsilon^{\max} \in \Re$ represents the maximum activation potential and $\alpha \in [0, 1]$ represents a coefficient that regulates de flow of activation potential. By regulating the flow of activation potential, the coefficient α controls the rate at which adaptation occurs. Low values of α result in slow changes to user feedback, while high values of α result in fast changes.

The increase of activation potential in a selected device d_s due to the accumulation of activation flow, is defined as:

$$\varepsilon_{\tau+1}(t,d_s) = \varepsilon_\tau(t,d_s) + \varphi_\tau(t,d_s) \qquad (4.4.2)$$

Reciprocally, a non-selected device will release activation flow in the proportion of their actual activation potential, leading to the decrease of its activation potential, defined as follows:

$$\varepsilon_{\tau+1}(t,d_{ns}) = \varepsilon_\tau(t,d_{ns}) - \frac{\varepsilon_\tau(t,d_{ns})}{\sum_{d \neq d_{ns}} \varepsilon_\tau(t,d)} \varphi_\tau(t,d_s) \qquad (4.4.3)$$

In this way, the proposed mechanism enables a dynamic adaptation based on the feedback provided by the user, through the adjustment of the distribution of activation potentials associated to devices. It acts like a continuous memory mechanism, which aim is not to learn a long-term dialog strategy but instead to provide short-term adaptation ability. This mechanism can be seen as a form of temporal difference adaptation (Sutton, 1988; Sutton & Barto, 1998; Staddon, 2001).

4.5 Dynamic Adaptation Example

To illustrate the operation of the proposed mechanism lets consider a simple scenario where the user selects a task t_1 several times with five available devices to choose (d_1 to d_5). In the simulation, the configuration parameters of the adaptive mechanism were set to $\alpha = 0.5$, which is a medium range adaptation rate, the initial activation potential $\varepsilon_0(t,d) = 1$ for all task-device pairs, and $\varepsilon^{\max} = 2$, which corresponds to the maximum activation potential.

As shown in Figure 2, initially all the devices have the same activation potential. However, when the user selects d_2 for the first time, the user feedback leads to an increase of the activation potential of the selected device and to the decrease of the activation potential of non-selected devices, as previously described.

At the third interaction, the user changes its choice from device d_2 to device d_3. This action produces a readjustment of the activation potentials, leading to the increase of d_3 activation potential. However, as can be observed in Figure 2, the increase of device d_3 activation potential asymptotically converges to $\varepsilon^{\max}$, therefore avoiding possible runways. This enables a prompt readjustment of activation potentials in the case of user choice change.

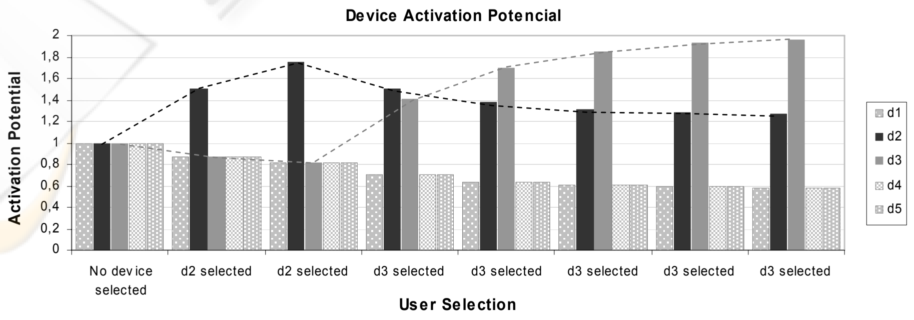


Figure 2: Evolution of the device activation potential according to user interaction.

4.6 Discussion

As illustrated in the previous example, the adaptive mechanism enables short-term adaptation to user interaction, therefore simplifying DMs clarification dialogs, reducing the amount of user's interactions to solve dialogue ambiguities.

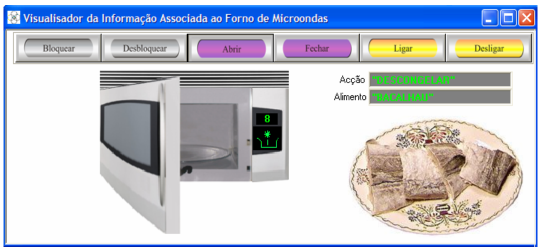
The proposed advisor service combines (i) an heuristic rule-based algorithm with (ii) an adaptive mechanism. The first one produces device and task ranks. The second one enables the reordering of the top ranking elements when they have similar ranks, according to previous user interactions. This combined operation allows the advisor service to discriminate among different choices even in the case of under-specified requests.

This kind of operation is particularly important when using spoken language, because the exhaustive enumeration of all the possible choices is typically inefficient, at least it will be boring for the user.

5 EXPERIMENTAL SETUP

Our current research is supported by our own domain simulator, in which we are using and, among others the proposed advisor service. 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. Currently, the environment simulator incorporates nine device simulators: an air conditioner simulator with 24 tasks using 63 concepts, a freezer simulator with 13 tasks using 96 concepts, a fryer simulator with 23 tasks using 92 concepts, a light source simulator with 20 tasks using 62 concepts, a microwave oven simulator with 26 tasks using 167 concepts, a kitchen table simulator with 13 tasks using 48 concepts, a water faucet simulator with 24 tasks using 63 concepts, a window simulator with 13 tasks using 44 concepts and a window blind simulator with 22 tasks using 65 concepts.

Figure 3 shows a screenshot of our home environment simulator that presents a microwave oven simulator selected by user's request "*defrost codfish*", where the selected cooking period is 8 minutes and the power is set to defrost.



Acção "DESCONGELAR"
Alimento "BACALHAU"

Figure 3: Microwave oven simulator.

The example presented in Section 4.5 corresponds to a situation where devices of the same class, which are similar, can be activated by the same task (e.g. "*turn-on*"), as is the case of choosing among several light sources or to choose between turning on some light sources or the microwave oven.

6 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 an adaptive approach 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.

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. 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 some examples.

An interesting aspect of our advisor service is its intrinsically ability to deal with the "*concept drift*" issue, since it is able to work at runtime with unforeseen changes in user preferences.

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 address a multi-user interaction scenario by extending the advisor adaptive mechanism to deal with different user's profiles. We also expect to explore the presented ideas, more deeply, applying them into a richer domain model trying to cover new perspectives.

ACKNOWLEDGEMENTS

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

REFERENCES

Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L., Stent, A., 2000. An Architecture for a generic dialogue shell. *Natural Language Engineering*, 6(3–4):213–228.

Daille, B., Gaussier, E., Lange, J., 1994. Towards Automatic Extraction of Monolingual and Bilingual Terminology. In *15th International Conference on Computational Linguistics*. Kyoto, Japan.

Denecke, M., 2002. Rapid Prototyping for Spoken Dialog Systems. In *COLING'02, 19th International Conference on Computational Linguistics*. Taipei, Taiwan.

Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J. and Burgelman, J-C., 2001. Scenarios for Ambient Intelligence in 2010. *IST Advisory Group Report*. IPTSSeville (Institute for Prospective Technological Studies).

Dzikovska, M., Allen, J., Swift, M., 2003. Integrating linguistic and domain knowledge for spoken dialog systems in multiple domains. In *IJCAI'03, 18th International Joint Conference on Artificial Intelligence*. Acapulco, Mexico.

Fellbaum, C. (editor), 1998. WordNet: An Electronic Lexical Database. MIT Press.

Feng, J., Bangalore, S., Rahim, M., 2003. Webtalk: Mining Websites for Automatically Building Dialog Systems. In *IEEE ASRU, Automatic Speech Recognition and Understanding Workshop*. United States, Virgin Islands.

Fensel, D., Benjamins, V., Motta, E., Wielinga, B., 1999. UPML: A Framework for Knowledge System Reuse. In *16th International Joint Conference on Artificial Intelligence*. Stockholm, Sweden.

Filipe, P., Mamede, N., 2006a. A Framework to Integrate Ubiquitous Knowledge Modeling. In *5th International Conference on Language Resources and Evaluation*. Genoa, Italy.

Filipe, P., Mamede, N., 2006b. Hybrid Knowledge Modeling for Ambient Intelligence. In *9th Workshop User Interfaces for All (ERCIM-UI4ALL) Special Theme: "Universal Access in Ambient Intelligence Environments"*. Königswinter, Germany. Springer Verlag.

Filipe, P., Mamede, N., 2006c. Ubiquitous Knowledge Modeling for Dialogue Systems. In *ICEIS 2006, 8th International Conference on Enterprise Information Systems*. Paphos, Cyprus.

Flycht-Eriksson, A., Jönsson, A., 2000. Dialogue and Domain Knowledge Management in Dialogue Systems. In *1st SIGdial Workshop on Discourse and Dialogue*, Hong Kong.

Gruber, T., 1992. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In *International Workshop on Formal Ontology*. Padova, Italy.

Henderson, J., Lemon, O., Georgila, K., 2005. Hybrid Reinforcement/Supervised Learning for Dialogue Policies from Communicator Data. In *IJCAI'05, 19th International Joint Conference on Artificial Intelligence Workshop on KRPDS*. Edinburgh, Scotland.

McTear, M., 2004. *Spoken Dialogue Technology: Towards the Conversational User Interface*. Springer Verlag.

Morgado, L., Gaspar, G., 2004. Focusing Reasoning Through Emotional Mechanisms. In *ECAI'04, 16th European Conference on Artificial Intelligence*. IOS Press. Valencia, Spain.

Neto, J., Mamede, N., Cassaca, R. and Oliveira, L., 2003. The Development of a Multi-purpose Spoken Dialogue System. In *Eurospeech 3003, 8th European Conference on Speech Communication and Technology*, Geneva, Switzerland.

Schatzmann, J., Weilhammer, K., Stuttle, M., Young, S., 2006. A Survey of Statistical User Simulation Techniques for Reinforcement-Learning of Dialogue Management Strategies. *The Knowledge Engineering Review*, 21(2):97 126.

Staddon, J., 2001. Adaptive dynamics: the theoretical analysis of behavior. *MIT Press*.

Sutton, R., 1988. Learning to Predict by the Method of Temporal Differences. *Machine Learning*, 3(1):9-44.

Sutton, R., Barto, A., 1998. Reinforcement Learning. *MIT Press*.