A Task Repository for Ambient Intelligence

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

[1] L^{2}F INESC-ID, Spoken Languages Systems Laboratory, 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 a task repository, a device semantic interface to express device capabilities, and an advice algorithm that suggests the best task-device pair to satisfy a request. The purpose of the task repository is the adaptation of a pervasive environment (Ambient Intelligence) to support natural language applications, such as a natural language interface. The task repository has a predefined group of concepts linked to linguistic and semantic resources and is updated, at runtime, with task descriptors associated with a set of heterogeneous devices. We assume that each device, belonging to the pervasive environment, holds its own semantic interface essentially composed of task descriptors. This approach tries to reach the ubiquitous essence of natural language, because the coverage of handmade lexical resources is limited, coverage problems remain for applications involving specific domains or involving multiple languages. Furthermore, we reduce the interface device problem to a database access problem. An environment simulator with the respective set of devices is depicted.

1 Introduction

This paper describes a task repository to support natural language applications, such as natural language interfaces (see Androusopoulos et al. [1] for a survey), within Ambient Intelligence (AmI) [2]. This repository includes an advice algorithm, which suggests the best task-device pair to satisfy a request. We assume that each device, spread in AmI, holds its own semantic interface essentially composed of task descriptors.

AmI is a recent paradigm in information technology. It can be defined as the merger of two important visions and trends: ubiquitous computing and social user interfaces. It is supported by advanced networking technologies, which allow robust, ad-hoc networks to be formed by a broad range of mobile devices and other objects. By adding adaptive user-system interaction methods, digital environments can be created to improve the quality of life of people by acting on their behalf. These context aware systems combine ubiquitous information, communication, and entertainment with enhanced personalization, natural interaction and intelligence.

This kind of environment can be characterized by the following basic elements: ubiquity, awareness, intelligence, and natural interaction. Ubiquity refers to a situation in which we are surrounded by a multitude of interconnected embedded systems, which are invisible and moved into the background of our environment. Awareness refers to the ability of the system to locate and recognize objects and people, and their intentions. Intelligence refers to the fact that the digital surrounding is able to analyze the context, adapt itself to the people that live in it, learn from their behavior, and eventually to recognize as well as show emotion. Finally, natural interaction refers to advanced modalities like natural speech and gesture recognition, as well as speech synthesis, which allow a much more human like communication with the digital environment than it is possible today.

Ubiquitous computing [3] or pervasive computing is an emerging discipline that brings together elements from distributed systems, mobile computing, embedded systems, human computer interaction, computer vision and many other fields. Its vision is grounded in the belief that processors are becoming so small and inexpensive that they will eventually be embedded in almost everything. Everyday objects will then be infused with computational power, enabling them as information artifacts and smart devices. By bringing computational power to the objects of the physical world, ubiquitous computing induces a paradigm shift in the way we use computers.

1.1 Pervasive Computing Environments

Pervasive or ubiquitous computing environments are far more dynamic and heterogeneous than enterprise environments. Enterprise network services operate within a network scope protected by firewalls and managed by human expert administrators. The increasing need to simplify the administration of pervasive environments introduces new requirements. A variety of new protocols has been proposed to attempt to satisfy these requirements and to provide spontaneous configuration based on service discovery.

Examples in academia include the Massachusetts Institute of Technology's Intentional Naming System (INS) [4], the University of California at Berkeley's Ninja Service Discovery Service (SDS) [5], and the IBM Research's DEAPspace [6]. Major software vendors ship their service discovery protocols with their current operating platforms, for example, the Sun Microsystems' Jini Network Technology [7], the Microsoft's Universal Plug and Play (UPnP) [8], and the Apple's Rendezvous [9].

Perhaps the most serious challenge to pervasive computing is the integration of computing devices with people. Normally, users are not prepared to deal with frequent reconfiguration problems. Unfortunately, we now spend precious time actively looking for services and manually configuring devices and programs. Sometimes the configuration requires special skills that have nothing to do with the tasks we want to accomplish.

Our research issue is exploring ways to enable non-technical people to use, manage and control their computational environment, focusing the home environment that is particularly hostile. In our standpoint, the use of natural language combined with a service discovery plays an important role in this scenario. To address this need, we propose in Section 2 a task repository that is a semantic service discovery infrastructure (that includes the environment linguistic resources), in Section 3 a device adaptation process, and in Section 4 an advice algorithm, which suggests the best task-device pair to satisfy a request.

2 Task Repository

This section describes the task repository data infrastructure that is represented by a relational data model. The main goal of the task repository is the adaptation of the pervasive environment to support semantic service discovery in order to facilitate the use of natural language applications. The task repository holds, at designed time, concept declarations that employ linguistic and semantic descriptors. The concepts that represent device classes are organized into a hierarchy or taxonomy. When a device is activated or deactivated the task repository is automatically updated using the device semantic interface. We propose a relational database to maintain the task repository. The database tables, of the task repository, are structured in three groups named: (i) DISCOURSE, (ii) WORLD, and (iii) TASK. The design of the relational database model was inspired in the *Unified Problem solving method Modeling Language* (UPML) [10] that refers decoupled knowledge components for building a knowledge-based architecture. The notation used to describe the task repository model is the *Integration DEFinition for Information Modeling* (IDEF1X). The requirements about fault tolerance, load balance, and other requirements forced by the pervasive computing environment should be supported by the adopted database implementation technology.

2.1 DISCOURSE Group

The group of tables named DISCOURSE defines a conceptual support. Fig. 1 shows the relational model of the DISCOURSE group.

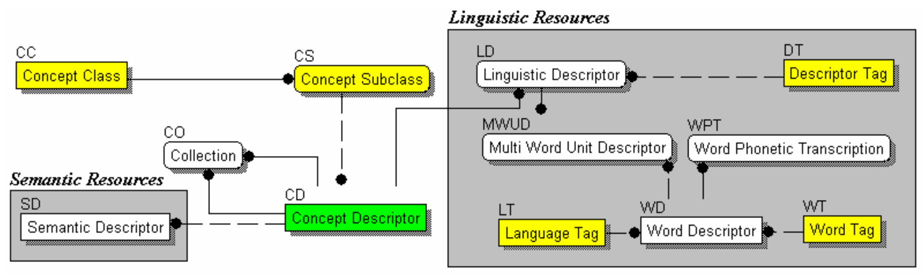


Fig. 1. DISCOURSE Group Model

The DISCOURSE group supports concept declarations used to describe device classes, devices, and the tasks they provide. A concept is an atomic knowledge unit, represented in table *Concept Descriptor* (CD). The CD table refers a unique *IDentifier* (ID), a *Concept Class* (CC), a *Concept Subclass* (CS), a *Semantic Descriptor* (SD), and a *Linguistic Descriptor* (LD). Table *Collection* (CO) links a concept used as a collection name with the respective concepts that are the collection members (*Attribute* concepts).

The concept classes (*Category*, *Class*, *Task*, and *Value*) and concept subclasses (*Collection*, *Unit*, *Active*, *Passive*, *Action*, *Acquisition*, *Attribute*, and *Quantity*), described in Table 1, allow the organization of the concept declarations.

The use of classes and subclasses are important to prevent the mismatch of concept references and to prevent task execution conflicts. The most relevant classes of concepts are (device) *Class* and *Task*. For instance, the class "*microwave oven*" is represented by a concept belonging to the *Active* subclass and the class "*thermometer*" is represented by a concept belonging to the *Passive* subclass.

Table 1. Concept Classes and Subclasses

Class	Subclass	Description
Category	**Collection**	For names of concept collections
	Unit	For physical units (International System of Units)
Class	**Active**	For subclasses of devices that provide task actions
	Passive	For subclasses of devices that only provide acquisition tasks
Task	**Action**	For task names that can modify the device state
	Acquisition	For task names that do not modify the device state
Value	**Attribute**	For property names (members of a collections)
	Quantity	For numeric values or quantities

In order to guarantee the vocabulary used to designate a concept 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 [11] in general is impressive, coverage problems remain for applications involving specific domains or multiple languages.

Each concept declaration has linguistic resources describing the concept in linguistic terms. The linguistic resources are express by linguistic descriptors in table Linguistic Descriptor (LD). 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 kitchen light*" should be treated as an instance of a "*light*", having the location "*kitchen*" without creating a new concept "*kitchen light*".

A linguistic descriptor holds a list of terms or more generically a list of Multi-Word Unit (MWU) [12]. A MWU list, see table Multi Word Unit Descriptor (MWUD), contains linguistic variations associated with the concept, such as synonymous or acronyms (classified in table Descriptor Tag (DT)). Each word, in table Word Descriptor (WD), has a part of speech tag, such as a noun, a verb, an adjective or an adverb (classified in table Word Tag (WT)); a language tag, such as "*pt*", "*br*", "*uk*" or "*us*" (classified in table Language Tag (LT)); and some phonetic transcriptions in table Word Phonetic Transcription (WPT). 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 [13]. The content of table WD can be shared with other concept declarations.

The CD table refers optionally semantic resources. Each one of the semantic resources is referred by one semantic descriptor in table Semantic Descriptor (SD). A semantic descriptor has references to other knowledge sources, for instance, an ontology or a lexical database, such as WordNet. The knowledge sources references in the task repository must be unique.

The references to knowledge sources must be encoded using a data format allowing a unique identification of the concept in the knowledge source. The syntax of the knowledge source reference do not need to be universal it is enough to keep the same syntax for a particular knowledge source. We recommend the use of a generic Uniform

Resource Identifier (URI) format to encode the knowledge sources references. In particular, could be used a Uniform Resource Locator (URL) or a Uniform Resource Name (URN). For instance, the declaration of the concept "*device*" can have the reference "*URN:WordNet21:device:noun:1*" meaning: the concept "*device*" is linked to the first sense of the noun "*device*" in WordNet 2.1, where it is described by "*an instrumentality invented for a particular purpose*" (in WordNet 2.1 this noun has five senses).

2.2 WORLD Group

The group of tables named WORLD represents a device class hierarchy and its instances that describe active devices. Fig. 2 shows an example of a device class hierarchy, in a home environment.

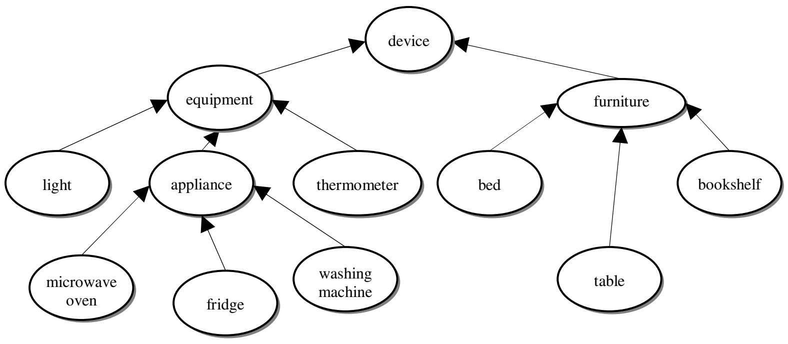


Fig. 2. Device Class Hierarchy

In a home environment, a device class may be either an appliance, or a thermometer, or a window, or a table.

Fig. 3 presents the relational model of the WORLD group. Table *Device Class Descriptor* (DDD) supports the hierarchy of the common device classes linking IDs of concepts that represent device classes. Table *Device Descriptor* (DD) keeps data about the active devices. At design time, DD table is empty. However, when a device is activated, its name (see Table 1) is automatically declare as a concept name and when the device is deactivated, this concept is removed.

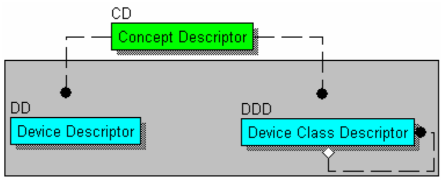


Fig. 3. WORLD Group Model

2.3 TASK Group

The group of tables named TASK represents the set of available tasks provided by the active device(s). When a device is activated, its semantic interface descriptor (see Table 2) is processed and when the device is deactivated, the respective task descriptors are removed. All concept IDs used to fill the slots of the task descriptors must exist in CD table, as we can see observing the relations in Fig. 4.

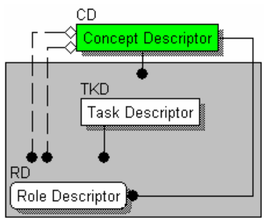


Fig. 4. TASK Group Model

Tables *Task Descriptor* (TKD) and *Role Descriptor* (RD) support the device's task descriptors. Table TKD maintains the task name and assumptions available to check the state of the world. Table RD keeps data about the allow input and output parameters of the tasks. The list of input parameters is described by an input role list. The list of output parameters is described by an output role list. One role describes one task argument.

2.4 Bridges

The task repository model is essentially based on the main descriptors represented in tables TKD, DD, and DDD. Nevertheless, these descriptors should be coupled/decoupled (when a device is activated/deactivated) to indicate the device class and the tasks they provide.

Fig. 5 presents the employ of bridges to couple the main descriptors. Table *Device Class Bridge* (DCB) couples the device descriptors in DD table to the respective device class descriptors in DDD table. Table *Device Task Bridge* (DTB) couples the device descriptors in DD table to the respective task descriptors in TKD table.

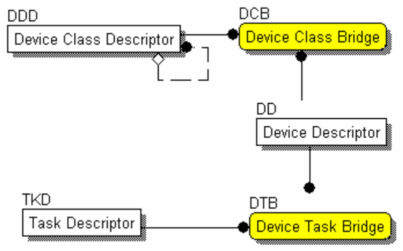


Fig. 5. Coupling of Descriptors Using Bridges

3 Device Adaptation

This section describes the adaptation of a device to work with the task repository. This adaptation is achieved by building a set of three layers, which would potentially cover all the relevant device features:

(i) The first layer is a device driver that provides an elementary abstraction of the device expressing the primitive capabilities. For instance, if the device is a light bulb we must be able, through the device driver, to turn on or to turn off the light and to ask about the associated state (on/off);
(ii) The second layer is an adapter that transforms the first layer into a more convenient interface, considering the device class. For instance, the adapter might transform labels into infrared command codes;
(iii) The third layer includes particular features of the device, bearing in mind, for instance, variations of the device commercial model.

The second and the third layer can be extended with local data about particular features of the devices. For instance, if we have a microwave oven we have to convert the power (100 watts) into a particular cooking process (defrost). In order to allow a semantic description of the device capabilities the third layer must be personalized to the user's needs, defining a device semantic interface descriptor. Table 1 represents a device semantic interface descriptor.

Table 2. Device Semantic Interface Descriptor

Slot	Value
device name	string
device class	ID-Class
task descriptor	
other task descriptors …	

A semantic interface descriptor starts with the slot *device name* followed by the slot *device class* and ends with a group of one or more *task descriptors* (detailed in Table 3). The value of the slot *device name* is a string. Other slot values, including the values of the task descriptors, refer IDs of concepts declared in the task repository, at design time.

Table 3 depicts a task descriptor where the "*" means mandatory fulfilling.

Table 3. Device Task Descriptor

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

A device task descriptor is a semantic representation of a capability or service provided by a device.

A task descriptor has a *name* (concept from the *Task* class) and optionally an *input list*, an *output list*, and *assumptions*.

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* is a concept from the *Attribute* subclass and is optional. The *range* is a concept from the Category class. 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 parameters. For instance, if the range is a positive integer and we want to assure that the parameter is greater than 3, then we must indicate the restriction rule: "name > 3". The *default* optional slot of the input role is a concept of the *Value* class. 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 of 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 acquisition 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 Advice Algorithm

This section describes an advice algorithm that suggests the best task-device pair to satisfy a request. The goal of this algorithm is to facilitate the exploration of the repository by natural language applications or directly by people (human administrators). The task repository supplies an interface to execute this algorithm. The suggestion of the task-device pair is based on the individual ranking of tasks and devices. The advice algorithm uses as input a list of terms that are compared against the linguist descriptors in order to identify the pivot concepts.

The pivot concepts references to tasks and devices are converted into points that are added to the respective individual rankings. The ranking points are determined considering two heuristic constants values: *nBase* and *nUnit*. The *nBase* value is equal to the maximum number of task roles (arguments) plus 1 (one). The *nUnit* value is equal to 3 (three) that are the number of ways to reference a task role (by *name*, *range*, and *value*).

The device ranking is modified following the heuristic rules:

1. If the pivot concept is used in a device descriptor (device name), the *nBase* value is added to the respective device ranking;
2. If the pivot concept is used in a device class descriptor (class name), the value obtained from the expression *nBase/2* is added to the respective device ranking;
3. If the pivot concept is used in a device super-class descriptor (super-class name), the value obtain from the expression *nBase/2-n*nUnit* is added to the respective device ranking, considering *n* is equal to the number of concept classes (in the hierarchy) between the device class and the device super-class.

The task ranking is modified following the heuristic rules:

4. If the pivot concept is used as a task name, the *nBase* value is added to the respective task ranking;
5. If the pivot concept is used as a task role name or as a task role range, the value obtained from the expression *nUnit/2* is added to the respective task ranking;
6. If the pivot concept is used to fill a task argument (is validate by the task role), the value obtained from the expression *nUnit/3* is added to the respective task ranking.

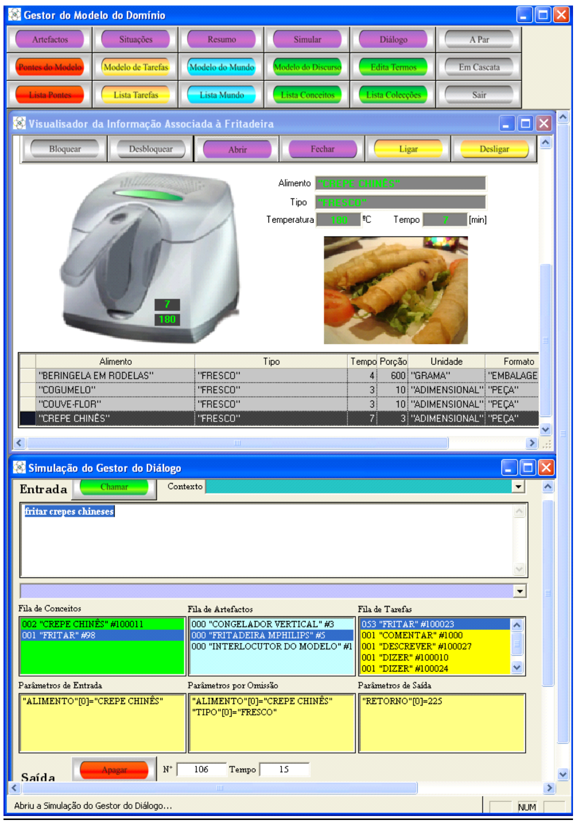
Finally, the pair task-device is formed by the task with the best ranking and by the device with the best ranking, which provides the task.

5 Experimental Evaluation

Our current work is based on an environment simulator in which we are testing the proposed task repository to represent a set of common home devices that are essentially present in the kitchen. Fig. 6 shows a screenshot of the home environment simulator, developed originally for Portuguese users.

This simulator allows the debug of the task invocation and the simulation of the interaction with a particular appliance. Using the simulator, we can activate and deactivate devices, do requests of tasks, obtain the answers and observe the devices behavior. We can also consult and print several data about the device semantic interfaces. Actually, 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 simulator table 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;
- A window blind simulator with 22 tasks using 65 concepts.



Alimento "CREPE CHINÊS"
Tipo "FRESCO"
Temperatura 180 ºC Tempo 7 [min]

	Alimento	Tipo	Tempo	Porção	Unidade	Formato
	"BERINGELA EM RODELAS"	"FRESCO"	4	600	"GRAMA"	"EMBALAGE
	"COGUMELO"	"FRESCO"	3	10	"ADIMENSIONAL"	"PEÇA"
	"COUVE-FLOR"	"FRESCO"	3	10	"ADIMENSIONAL"	"PEÇA"
	"CREPE CHINÊS"	"FRESCO"	7	3	"ADIMENSIONAL"	"PEÇA"

fritar crepes chineses

Fila de Conceitos	Fila de Artefactos	Fila de Tarefas
002 "CREPE CHINÊS" #100011	000 "CONGELADOR VERTICAL" #3	053 "FRITAR" #100023
001 "FRITAR" #98	000 "FRITADEIRA MPHILIPS" #5	001 "COMENTAR" #1000
	000 "INTERLOCUTOR DO MODELO" #1	001 "DESCREVER" #100027
		001 "DIZER" #100010
		001 "DIZER" #100024

Parâmetros de Entrada	Parâmetros por Omissão	Parâmetros de Saída
"ALIMENTO"[0]="CREPE CHINÊS"	"ALIMENTO"[0]="CREPE CHINÊS"	"RETORNO"[0]=225
	"TIPO"[0]="FRESCO"	

Nº 106 Tempo 15

Abriu a Simulação do Gestor do Diálogo... NUM

Fig. 6. Environment Simulator

On the top of the screen, is showed the Fryer simulator after the execution of the request: "*frying Chinese spring rolls*". The Fryer simulator screen shows the automatically select temperature (180 ºC) and duration (7 minutes) of the frying process. On the bottom of the screen, we can see data processed by the advice task algorithm. In order to determine the best task-device pair, the advice algorithm selects two pivot concepts associated with "*frying*" and with "*Chinese spring rolls*". The concept "*frying*" is used as a task name (rule 4). The concept "*Chinese spring rolls*" can be used to fill and argument in tasks "*frying*", provide by the "*fryer*" and "*tell*", provided by the "*freezer*" (rule 6). The task "*frying*" is the best choice with *nBase* + *nUnit/3* points. Finally, the device "*fryer*" is selected only because it provides the task "*frying*". So, the suggested task-device pair is "*frying*"-"*fryer*". In this case we do not have a ranking of devices because the request does not refer any device.

6 Discussion and Future Work

The growth in pervasive computing will require standards in device communication interoperability. These devices must be able to interact and share resources with other existing devices and any future devices across the network. The current technologies require human interventions to solve environment reconfiguration problems. We believe that these technologies must be improved with more human like ways of interaction that must include natural language support.

In this context, we have already identified two future application scenarios:

(i) Automatic generation of multilingual device descriptions. The lexical knowledge linked in a concept declaration can be combined with the task descriptors to generate a systematic report (easily understandable) of the available services of one device and even of the entire environment. The generation can be targeted to a preferred language and the part of speech tags can be used for syntactic validation. The phonetic transcription can be used to disambiguate terms (homograph vs. homophone);

(ii) Automatic build of device's graphical user interfaces. The knowledge about the class of concepts (see Table 1) and other appropriated extensions can be used to determine the convenient graphical item to represent them. For instance, when a task input role is represented by a concept of the Collection class the respective argument should be graphically represented by a combo box.

As future work, we also expect to evaluate the task repository and the advice algorithm within an improved context, integrating the progresses already achieved in our lab [15] [16] [17]. Currently, our goal is the development of a Device Integration Process (DIP) that will update automatically the DISCOURSE group. DIP will resolve ambiguities in the vocabulary that should be included into each one of the device semantic interfaces.

7 Concluding Remarks

Our approach reduces the interface device problem to a database access problem and is a definitive contribution to facilitate the use of natural language applications, namely natural language interfaces (with or without dialogue) in a pervasive computing environment. This approach tries to reach the ubiquitous essence of natural language, because the coverage of handmade lexical resources is limited, coverage problems remain for applications involving specific domains or involving multiple languages.

The task repository supports semantic service discovery, which is an important feature to achieve spontaneous configuration of a heterogeneous set of devices. The presented ideas have been applied, with success, in a set of devices that represents a home environment.

References

1. Androutsopoulos, I., Ritchie, G., Thanish, P.: Natural Language Interfaces to Databases – An Introduction, Natural Language Engineering, vol 1, part 1, 29-81, (1995)
2. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.: Scenarios for Ambient Intelligence in 2010, IPTSSeville, ISTAG (2001)
3. Weiser, M.: The Computer for the Twenty-First Century, Scientific American (1991)

4. Adjie-Winoto, W., Schwartz, E., Balakrishnan, H., Lilley, J.: The Design and Implementation of an Intentional Naming System, Proc. 17th ACM Symp. Operating System Principles (SOSP 99), ACM Press, pp. 186–201 (1999)
5. Hodes, T., Czerwinski, S., Zhao, Ben., Joseph, A., Katz, R.: An Architecture for Secure Wide-Area Service Discovery, ACM Wireless Networks J., vol. 8, nos. 2/3, pp. 213–230 (2002)
6. Nidd, M.: Service Discovery in DEAPspace, IEEE Personal Comm., pp. 39–45, Aug. 2001
7. Jini Technology Core Platform Specification, v. 2.0, Sun Microsystems (2003)
8. UPnP Device Architecture 1.0, UPnP Forum (2003)
9. Cheshire, S., Krochmal, M.: DNS-Based Service Discovery, IETF Internet draft, work in progress, (2005)
10. Fensel, D., Benjamins, V., Motta, E., Wielinga, B.: UPML: A Framework for Knowledge System Reuse, IJCAI (1999)
11. Fellbaum, C. (editor): WordNet: An Electronic Lexical Database, MIT Press (1998)
12. Daille, B., Gaussier, E., Lange, J.: Towards Automatic Extraction of Monolingual and Bilingual Terminology, COLING 94, 515-521 (1994)
13. SAMPA (SAM Phonetic Alphabet), Spoken Language Systems Lab (L^2F), http://www.l2f.inesc-id.pt/resources/sampa/sampa.html
14. Gruber, T.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing, In International Workshop on Formal Ontology, Padova, Italy (1992)
15. Filipe, P., Mamede, N.: Databases and Natural Language Interfaces. V Jornadas de Bases de Datos, Valladolid, Spain (2000)
16. Filipe, P., Mamede, N.: Towards Ubiquitous Task Management. 8th International Conference on Spoken Language Processing, Jeju Island, Korea (2004)
17. Filipe, P., Mamede, N.: Ubiquitous Knowledge Modeling for Dialogue Systems, to appear in 8th International Conference on Enterprise Information Systems, Paphos, Cyprus (2006)