ENHANCING A PERVASIVE COMPUTING ENVIRONMENT WITH LEXICAL KNOWLEDGE

P. Filipe[1], M. Barata[2], N. Mamede[3], P. Araújo[4]
[1]pfilipe@deetc.isel.ipl.pt, [2]mmb@isel.ipl.pt, [3]nuno.mamede@l2f.inesc-id.pt, [4]paraujo@deetc.isel.ipl.pt
[1,2,4]GIATSI – Grupo de Investigação Aplicada em Tecnologias e Sistemas de Informação, Lisbon, Portugal
[1,2,4]ISEL – Instituto Superior de Engenharia de Lisboa, Lisbon, Portugal
[1,3,4]L^{2}F INESC-ID – Spoken Languages Systems Laboratory, Lisbon, Portugal
[3]IST – Instituto Superior Técnico, Lisbon, Portugal

Abstract – A pervasive computing environment consists typically of a large heterogeneous collection of networked devices. This paper describes the use of lexical knowledge to improve a pervasive computing environment. In an ongoing research project, we are exploring ways to enable non-technical users to manage and control their home environment that is particularly hostile. We assume that each device belonging to the pervasive environment has its own knowledge model, linked to lexical resources, with the purpose of defining a semantic interface. This approach tries to reach the pervasive essence of the natural language. The coverage of handmade lexical resources is limited, coverage problems remain for applications involving specific domains or involving multiple languages. Our recent efforts are directed towards the technology development, focusing on devices that are household appliances. This work is a contribution to facilitate, specially: the generation of multilingual device descriptions, the automatic build of device's graphical user interfaces, and on the fly adaptation of a spoken dialogue system to the pervasive environment.

Keywords: Pervasive Computing, Lexical Knowledge, Spoken Dialogue System.

I. INTRODUCTION

A pervasive computing environment consists typically of a large heterogeneous collection of networked devices. For start, we have to refer the terminology typically used to designate two key concepts within a pervasive computing environment, which are: Device and Service. Devices and services are the entities that participate in environment. "*Devices*" includes conventional computers, small handheld computers (PDAs), printers, and more specialized network devices, such as a thermometer or household appliances. "*Services*" includes any sort of network service that might be available. In fact, most devices are represented on the network by one or more services. Furthermore, a single network attached device may implement several services, e.g., a network printer may provide printing and fax (and who knows what else), all in a single device. In this context, devices and services are considered essentially equivalent and interchangeable. Together, these will be termed "*Entities*" or "*Resources*" on the network.

Entities must interoperate with other entities without pre-existing knowledge. This is a key aspect of spontaneous configuration.

A pervasive computing environment offers us an interesting starting point of discussion regarding the goal of achieving Plug and Play (PnP) functionality and its subsequent application to manage and control household appliances. There are several imaginable levels of PnP, which can be summarized in the following possibilities:

(i) No PnP – The environment can handle a fixed set of devices that are always connected;
(ii) Weak PnP – The environment can handle a fixed set of devices that can be connected to or disconnected from the network;
(iii) Medium PnP – The environment can handle a fixed set of device classes and only new instances of these device classes can be plug and played;
(iv) Strong PnP – The environment can handle completely new devices of previously unknown classes.

II. AMBIENT INTELLIGENT ENVIRONMENTS

Ambient Intelligence [1] refers to 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 which 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. Natural Interaction finally refers to advanced modalities like natural speech and gesture recognition, as well as speech synthesis, which will allow a much more human like communication with the digital environment than is possible today.

Ubiquitous computing [2] or pervasive computing are emerging disciplines bringing together elements from distributed systems, mobile computing, embedded systems, human computer interaction, computer vision and many other fields. Their 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.

III. Pervasive Computing Environments

Pervasive computing environments, such as home 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.

Examples in academia include the Massachusetts Institute of Technology's Intentional Naming System (INS) [3], University of California at Berkeley's Ninja Service Discovery Service (SDS) [4], and IBM Research's DEAPspace [5].

Major software vendors ship their service discovery protocols with their current operating platforms, for example, Sun Microsystems' Jini Network Technology [6], Microsoft's Universal Plug and Play (UPnP)[7], and Apple's Rendezvous [8].

Perhaps the most serious challenge to pervasive computing is the integration of computing devices with people. Normally, the 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.

IV. Related Work

Various ubiquitous computing projects (e.g., MIT's intelligent room [16]) have considered multi-modal interfaces that include natural language.

A number of other researchers have modeled appliances and developed specification languages (e.g., in XML) for appliance interfaces [17][18][19][20]. Systems designed around these specifications have tried to create a single interface to all appliances on another, remote appliance like a PDA. The Personal Universal Controller (PUC) [21] generates an interface to appliances, using XML specifications. Unlike our proposal, the PUC executes simple commands, and does not use lexical knowledge.

TRIPS Allen, et al. [22] is an agent based architecture for handling natural conversation in the travel planning domain. In contrast to this kind of system, our proposal is to keep a simple solution to adapt on the fly a multi-propose spoken dialogue system [23]. Some related work on spoken dialogue systems within a ubiquitous domain can be seen in [24][25].

Quesada et al. [26] describe a spoken dialogue system in the D'Homme project that is specifically designed for interacting with household appliances. The D'Homme system supports a set of core devices that are essentially binary and dimmable more simple that a household appliance.

A number of commercial systems are 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.

V. Lexical Knowledge

Lexical knowledge encompasses all the information that is known about words and the relationships among them. Outside of strictly linguistic knowledge such as phonology (the study of speech sounds (phonemes) and how they are used), morphology (grammatical and other variants of words that are derived from the same root or stem), and grammatical categories, this includes conceptual knowledge, such as on various ontological categories, and pragmatic knowledge, such as conventional usages for certain words.

VI. Device Semantic Interface

The adaptation process of a physical device to a pervasive computing environment 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 services. For instance, if the device is a door we must be able, through the device driver, opening or closing the door and to ask about the associated state (opened/closed);

(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 third layer must be personalized to the user's needs, defining a device semantic interface, which integrates *concept declarations*, *class declarations*, and *task descriptors*. Table I describes a device semantic interface where mandatory slots are signaled by "*".

The concepts are organized in pre-defined groups:

(1) *attribute*, (2) *device*, (3) *general*, (4) *quantity*, and (5) *task*. A group of concepts can be seen as a controlled vocabulary.

Table I- Device Semantic Interface Descriptor

slot		value
concept declaration*	**ID***	alphanumeric
	group*	attribute, device, general, quantity, task
	lexical descriptor*	Multilingual MWU list
	semantic descriptor	Knowledge source list
	collection	ID list
other concept declarations …		
class declaration*	**name**	device
	class	device
	super classes	device list
task descriptor* (*see Table II*)		
other task descriptors …		

A semantic interface descriptor starts with one or more concept declarations. A *concept declaration* refers a unique *IDentifier* (ID) that is an alphanumeric code, a *group* in which the concept belongs, a *lexical descriptor*, an optional *semantic descriptor*, and an optional collection of related concept IDs.

The *lexical descriptor* has a list of Multi-Word Unit (MWU) [9]. This list contains linguistic variations associated with the concept, such as synonymous or acronyms, and can be replicated for each one of the supported languages. The terms or words (root or stem) that compose the MWU are linked with their part of speech tags and phonetic transcriptions. Unlike a terminology inspired ontology [10], concepts are not included for complex terms unless absolutely necessary. For example, an item such as "*the back bedroom light*" is treated as an instance of a light, having the location "*back bedroom*" without creating a new concept "*back bedroom light*"..

The *semantic descriptor* references a list of external knowledge representations, for instance, a domain ontology or a lexical database such as WordNet [11].

The *class declaration* refers optionally members of the *device* group that are *name*, *class*, and a list of super classes.

Finally, the semantic interface descriptor ends with one or more task descriptors (detailed in Table II).

Table II, presents a task descriptor where the "*" means mandatory fulfilling. A device task descriptor is a semantic representation of a service provided by a device. We consider two kinds of tasks: *action* and *perception* tasks. A perception task cannot modify the state of the device. A *task descriptor* has a *name* and optionally an *input list*, an *output list*, and *assumptions*. A *name* is a concept from the pre-defined *task* group.

Table II- Device Task Descriptor

slot			value
name*			task
input list	**input role**	**name**	attribute
		range*	attribute or quantity
		restriction	*rule*
		default	attribute or quantity
	other input roles …		
	pre-condition		*rule*
output list	**output role**	**name**	attribute
		range*	attribute or quantity
	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* is member of the *attribute* group and is optional. The *range* is member of the *attribute* or *quantity* groups. The *restriction* is a rule that is materialized as logical formula and is optional. The *range* rule and the *restriction* rule define the set of allowed parameters. For instance, if the range is a positive integer (quantity) and we want to assure that the parameter is lower than 10, then we must indicate the restriction rule: "name < 10". The *default* of the input role is a member of the *attribute* or *quantity* groups and is optional. When 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 validation at three distinct layers. First, we have the restriction rule to perform argument validation. In a restriction, only the associated input role name can be referred. Secondly, we have the pre-condition (to be checked before task execution) and pos-condition (to be checked after task execution) for parameter group validation. In a pre-condition only task input role names can be referred. In a pos-condition only output role names can be referred. Thirdly, we have assumptions for state validation: the initial condition (to check initial state before task execution) and the final condition (to check final state after task execution). In assumptions, role names and results of perception task calls can be referred.

VII. Application Scenarios

The previous described device semantic interface can be used by people or by other devices to recognize a device interface.

In this context, we have already identified the following application scenarios:

(1) Automatic generation of multilingual device descriptions. The lexical knowledge linked in the 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 target 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). With the phonetic transcription of terms, is possible to correctly interpret the represented concepts, and if needed, to synthesize speech. For Portuguese language, the phonetic transcription is obtained using SAMPA that is the phonetic alphabet for European Portuguese [12].

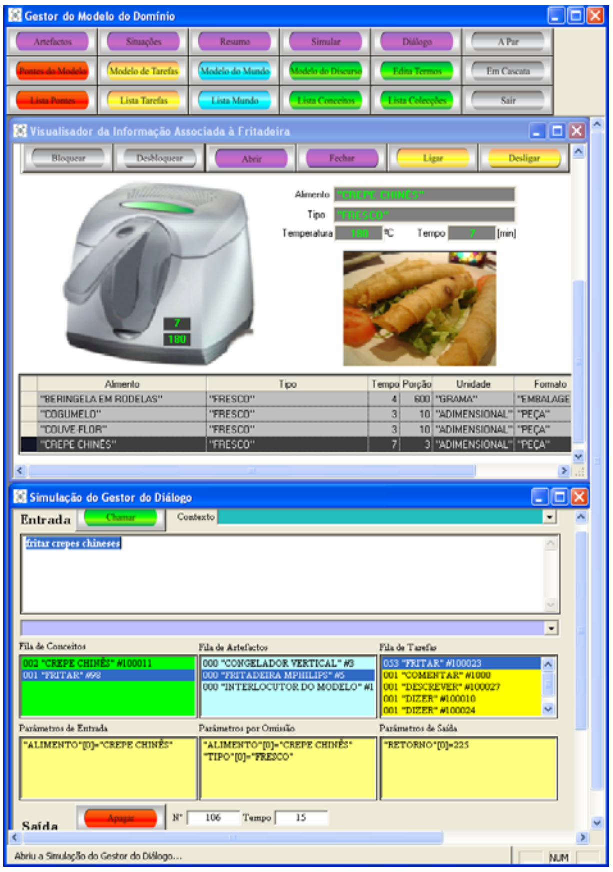
(2) Automatic build of device's graphical user interfaces. The knowledge about the group of concepts (attribute, device, general, quantity, and task) can be used to determine the convenient graphical item to represent them. For instance, when a task input role is represented by an attribute concept with a linked list o concepts this role should be represented by a combo box. Furthermore when a concept belongs to the task group should be represented graphically by a push button that is used to fire the respective task execution.

(3) On the fly adaptation of spoken dialogue system to a pervasive environment. Spoken dialogue systems 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 [13][14]. The main purpose of a spoken dialogue system is to provide an interface between a user and a computer-based application such as a database or expert system. Recently, new domain applications have been adopted by spoken dialogue designers, such as the home environment. In order to face this challenge some authors follow approaches focusing the linguistic need of the each device or appliance [15]. The proposed semantic interface allows the device integration into the spoken language dialogue system previously prepared to deal with the propose task layout representation. This integration process will allow the available task recognition on the fly for completely new devices – strong PnP.

VIII. Experimental Evaluation

Our current work is based on an environment simulator in which we are testing the proposed semantic interface layout and the need of lexical knowledge to represent a set of home appliances that are essentially present in the kitchen.

Figure 1 show a screenshot of the home environment simulator, developed originally for Portuguese users. On the top of the screen we can see the screen of the Fryer simulator after the execution of the request: "*frying chinese spring rolls*". This 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 the list of the tasks and concepts involved in treatment of the command.



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"

Contexto

fritar crepes chineses

Fila de Conceitos
002 "CREPE CHINÊS" #100011
001 "FRITAR" #98

Fila de Artefactos
000 "CONGELADOR VERTICAL" #3
000 "FRITADEIRA MPHILIPS" #5
000 "INTERLOCUTOR DO MODELO" #1

Fila de Tarefas
053 "FRITAR" #100023
001 "COMENTAR" #1000
001 "DESCREVER" #100027
001 "DIZER" #100010
001 "DIZER" #100024

Parâmetros de Entrada
"ALIMENTO"[0]="CREPE CHINÊS"

Parâmetros por Omissão
"ALIMENTO"[0]="CREPE CHINÊS"
"TIPO"[0]="FRESCO"

Parâmetros de Saída
"RETORNO"[0]=225

Nº 106 Tempo 15

Abriu a Simulação do Gestor do Diálogo... NUM

Figure 1- Environment Simulator

This simulator allows the debug of the task invocation and the simulation of the interaction with a particular appliance. Using the simulator, we can attach and detach devices, do requests of tasks, obtain the answers and observe the devices behavior. We can also consult and print several data about the several device semantic interfaces. The available device simulators are: Air Conditioner with 24 tasks and 63 concepts, Freezer with 13 tasks and 96 concepts, Fryer with 23 tasks and 92 concepts, Light Source with 20 tasks and 62 concepts, Microwave Oven with 26 tasks and 167 concepts, Kitchen Table with 13 tasks and 48 concepts, Water Faucet with 24 tasks and 63 concepts, Window with 13 tasks and 44 concepts, and a Window Blind with 65 concepts and 22 tasks.

Actually, we are migrating the simulator to a distributed architecture that is presented in Figure 2. The knowledge representation is supported by relational databases allowing an easy manipulation of data.

We are currently using Java technology, under Linux Fedora, deployed in Tomcat Application Server. In order to maintain the knowledge models we are using MySQL database to satisfy MI and device needs. The database MITemp (in Figure 2) is used as a temporary database to load data from devices.

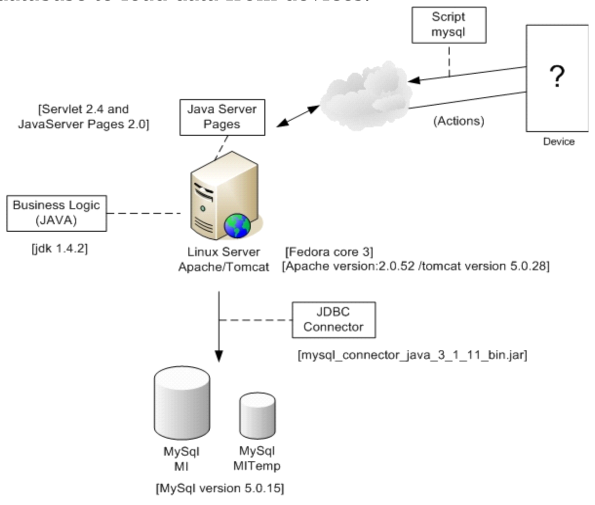


Figure 2- Environment Simulator

IX. Discussion

The growth in pervasive computing will require standards in device communications. These devices must be able to interact and share resources with other existing devices and any future devices across the network.

We considered the two extremes of technologies in terms of semantics: UPnP with all semantic aspects represented in XML and Jini with no need to represent semantic aspects.

Jini relies essentially on code upload from devices in order that clients can access their services. If the code is itself a complete User Interface, then the client need not have any prior knowledge of the device interface in order to use it. The user can interact with the device via the user interface on his client, but the client need not know anything about the device. Otherwise, in order to use a service, the device must provide a "well-known" interface that the client does know about. From an implementational view, Jini is attractive simply because the base technology is already public and reasonably well developed.

Jini attributes system has the advantage of being strongly typed, due to the fact that it utilizes the Java type system. This advantage is lessened, however, by the lack of hierarchical descriptions. Because of this, the fields of many attributes will be represented as Strings, rather than actual data structures.

UPnP is particularly interesting in providing both a simple format for device models (a set of actions of the form 'command plus parameters', an array of variables to model internal state, and an event mechanism for autonomous alerts on changes of internal state) and agreed specifications for particular devices for manufacturers to conform to. Although most of the specifications themselves generally remain in draft format and are not public knowledge from an implementational view, UPnP is less attractive at the moment simply because the base technology is less well developed. There are 'early release' versions of UPnP development kits and recent releases of the Windows operating system (ME, CE) have some support for UPnP built into them but in general the environment is not as well developed as the Jini platform.

X. Conclusions

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 includes natural language support. Although the coverage of handmade resources such as WordNet in general is impressive. Coverage problems remain for applications involving specific domains or multiple languages. Our proposal considers possible coverage problems and the pervasive essence of the natural language. Our work is a contribution to enhance the pervasive environments with a simple inclusion of incremental lexical knowledge. We expect to explorer these ideas moving our simulated devices to real pervasive environments.

XI. Acknowledgments

This work is supported by GIATSI project "*Integração Dinâmica de Dispositivos em Ambientes Inteligentes*" (ID^{2}AI). The authors thank the contribution of Professor Luís Osório and undergraduate student Ana Salgueiro.

References

[1] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J-C. Burgelman, ISTAG 2001, "Scenarios for Ambient Intelligence in 2010", IPTSSeville (Institute for Prospective Technological Studies), 2001

[2] Mark Weiser, "The Computer for the Twenty-First Century", Scientific American, 1991

[3] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley, "The Design and Implementation of an Intentional Naming System", Proc. 17th ACM Symp. Operating System Principles (SOSP 99), ACM Press, pp. 186–201, 1999

[4] Todd D. Hodes, Steven E. Czerwinski, Ben Y. Zhao, Anthony D. Joseph, and Randy H. Katz, "An Architecture for Secure Wide-Area Service Discovery", ACM Wireless Networks J., vol. 8, nos. 2/3, pp. 213–230, 2002

[5] M. Nidd, "Service Discovery in DEAPspace", IEEE Personal Comm., Aug. 2001, pp. 39–45

[6] Jini Technology Core Platform Specification, v. 2.0, Sun Microsystems, 2003

[7] UPnP Device Architecture 1.0, UPnP Forum, 2003.

[8] Stuart Cheshire and Mark Krochmal, "DNS-Based Service Discovery", IETF Internet draft, work in progress, 2005

[9] B. Daille, E. Gaussier, J. Lange, "Towards Automatic Extraction of Monolingual and Bilingual Terminology", COLING 94, 515-521, 1994

[10]T Gruber, "A Translation Approach to Portable Ontology Specifications", Knowledge Acquisition, 5(2), pp. 199-220, 1993

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

[12]SAMPA (SAM Phonetic Alphabet), Spoken Language Systems Lab (L2F), http://www.l2f.inesc-id.pt/resources/sampa/sampa.html, 1995

[13]N. Fraser, "Assessment of Interactive Systems. Handbook of Standards and Resources for Spoken Language Systems", In D. Gibbon, R. Moore & R. Winski (eds.) New York: Mouton de Gruyter, 1997

[14]M. McTear, "Spoken Dialogue Technology: Enabling the Conversational Interface", ACM Computing Surveys, Volume 34, 2002

[15]M. Rayner, I. Lewin, G. Gorrell, J. Boyce, "Plug and play speech understanding". 2nd SIGdial Workshop on Discourse and Dialogue, Aalborg. 2001.

[16]Michael Coen, Luke Weisman, Kavita Thomas, Marion Groh, "A Context Sensitive Natural Language Modality for the Intelligent Room", Proceedings of MANSE'99. Dublin, Ireland, 1999

[17]Marc Abrams, Constantinos Phanouriou, Alan L. Batongbacal, Stephen M. Williams, and Jonathan E. Shuster, "UIML: An Appliance-Independent XML User Interface Language", The Eighth International World Wide Web Conference, Toronto, Canada, 1999

[18]Kevin F. Eustice, Tobin J. Lehman, Armando Morales, Michelle C. Munson, Stefan Edlund, and Miguel Guillen, "A Universal Information Appliance", IBM Systems Journal, 38(4): pp. 575-601, 1999

[19]Jaap Haartsen, Mahmoud Naghshineh, Jon Inouye, Olaf J. Joeressen, and Warren Allen, "Bluetooth: Vision, Goals, and Architecture", ACM Mobile Computing and Communications Review, 2(4): p. 38-45, 1998

[20]Dan R. Olsen Jr., Sean Jefferies, Travis Nielsen, William Moyes, and Paul Fredrickson, "Cross-modal Interaction using Xweb", ACM SIGGRAPH Symposium on User Interface Software and Technology, pp. 191-200, 2000

[21]Jeffrey Nichols, Brad A. Myers, Michael Higgins, Joseph Hughes, Thomas K. Harris, Roni Rosenfeld, and Mathilde Pignol, "Generating Remote Control Interfaces for Complex Appliances", CHI Letters: ACM, 2002

[22]J. Allen, G. Ferguson, A. Stent, "An Architecture for more Realistic Conversational Systems" Intelligent User Interface, 2001

[23]J. P. Neto, N. J. Mamede, R. Cassaca, L. Oliveira, "The Development of a Multi-purpose Spoken Dialogue System", Eurospeech, 2003

[24]P. Filipe, and N. Mamede, "Towards Ubiquitous Task Management", In Interspeech 2004, Jeju Island, Korea, 2004

[25]P. Filipe, and N. Mamede, "Ubiquitous Knowledge Modeling for Dialogue Systems" To appear in 8th International Conference on Enterprise Information Systems, Paphos, Cyprus, 2006

[26]J. F. Quesada, F. Garcia, E. Sena, J. A. Bernal, G. Amores, "Dialogue Managements in a Home Machine Environment: Linguistic Components over an Agent Architecture", SEPLN, 89-98, 2001