5

Ambient Intelligence Interaction via Dialogue Systems

Porfírio Filipe [a, b, c] and Nuno Mamede [a, d]
[a] *L2F INESC-ID – Spoken Language Systems Lab*
[b] *GuIAA – Research Group on Autonomous Environments*
[c] *ISEL – Instituto Superior de Engenharia de Lisboa*
[d] *IST – Instituto Superior Técnico*
Portugal

1. Introduction

The vision of Ambient Intelligence (AmI), that expresses a paradigm in information technology, is based on the increasing technological advances in embedding computational power, information and sensing capabilities into everyday artefacts and environments (Ducatel et al., 2001). Intelligent environment is a technological concept that, according to Mark Weiser, is "*a physical world that is richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives, and connected through a continuous network*" (Weiser, 1991). Consequently, the computational model of this kind of environments can be analysed as a large collection of networked artefacts.

The use of embedded systems to control artefacts, tools and appliances has been common practice for almost two decades now. With every new generation, these controllers provide an ever increasing list of capabilities in the form of assistance, information, and customization. However, it is the addition of communication capabilities that changes the perspectives of what such systems can do: gather information from other sensors, real objects and computers on the network, or enable user-oriented customization and operations through short-range communication (Cook & Das, 2004).

For this, AmI technological infrastructures must be able to spontaneously reconfigure themselves and grow from the available, purposeful artefacts in order to become effective in the real world. Recent approaches are based on exploiting the affordances of real artefacts by augmenting their physical properties with the potential of computer based support (Streitz, 2007). Combining the best of both worlds requires an integration of real and virtual worlds resulting in hybrid worlds.

Despite the current availability of technology, there is a notorious absence of large scale settings. In this context, the fundamental question is what kind of intelligent interfaces is needed to access a large federation of artefacts within an AmI scenario. In this scenario, each one of the hybrid artefacts or controllable resources should be designed to allow plug and play integration in an AmI multimodal architecture (Dahl et al., 2008).

According to the convincing demonstration of Byron Reeves and Clifford Nass, the interactions with computers, television, and new communication technologies are identical to real social relationships and to the navigation of real physical spaces (Reeves & Nass, 1996). In this perspective, it is reasonable to assume, for instance, that people would talk naturally with a household appliance.
The design of natural language applications that allow people to talk with machines or computers, in the same way that they talk with each other, is materialized under the form of a Spoken Dialogue System (SDS) (Zue & Glass, 2000; McTear, 2004), having constituted a natural interface, where the use of speech is privileged.
Currently, it is unrealistic to consider a real existence of an autonomic SDS embedded into each one of the environment's artefacts, because of real hardware limitations. Nevertheless, the coordination and collaboration between a set of autonomic SDS is, per se, a huge challenge.
In order to achieve AmI interaction via SDS (Minker et al., 2009) each controllable resource should implement, at least, an adequate semantic interface to expose the resource's functional capabilities at knowledge level (Newell, 1982). Nevertheless, this semantic interface is not only for exclusive use of the SDS because it can be also freely used by other concurrent systems.
Essentially, a semantic interface is a collection of task descriptors used to expose artefact's functional capabilities, and is sustained by a set of concepts that are atomic knowledge units. The mining of a concept can be previously established or can be dynamically inferred comparing its linguistic knowledge or its semantic references to internal or external knowledge source nodes (Filipe & Mamede, 2008; Filipe & Mamede 2009). This approach focus on the representation and management of the environment's knowledge aims to satisfy dialogue management needs related to the environment semantic interoperability. For this, an environment Knowledge Aggregation Process (KAP) built-in a distributed Environment Interaction Manager (EIM) manages federations of resources within an AmI holistic vision (Aartsa & Ruytera, 2009).
Summarizing, our contribution enables spontaneous reconfiguration of SDS, within an AmI holistic vision, to provide speech natural interaction. Section 2 gives an overview of the state of the art in SDS. Section 3 presents relevant issues about spontaneous portability.
Section 4 gives an overview of the knowledge modelling approach for semantic interface design. Section 5 describes the Knowledge Aggregation Process (KAP). Section 6 presents the experimental setup describing practical issues as well. Section 7 summarizes the contribution's main topics, conclusions and future work.

2. Spoken Dialogue Systems

The origins of SDS can be traced back to Artificial Intelligence (AI) research in the 1950s concerned with developing conversational interfaces. The research of SDS is commonly considered a branch of human-computer interaction, although its origins are generally rooted in the automatic speech recognition community.
However, it is only within the last decade or so, with major advances in speech technology, that large scale working systems have been developed and, in some cases, introduced into commercial environments. The integration of components into a working system is still an important key issue (McTear, 2004).

Typically, a SDS is used to access the data source of the domain often materialized under a relational database. The traditional interaction cycle starts when the user's request, which is captured by a microphone, provides the input for the Speech Recognition component. Next, the Language Understanding component receives the recognized words and builds the related speech acts. The Dialogue Manager (DM) processes the speech acts, accesses the Data Source and then calls the Response Generation component to generate a response message for the user. Finally, the message is processed by the Speech Output component to produce speech. The response of the SDS can be final or a request for clarification. When everything is acceptable, a final answer is produced based on the obtained external data.
Fig. 1 shows a typical logical flow through SDS components architecture to access the domain data source, typically sustained by a relational database.

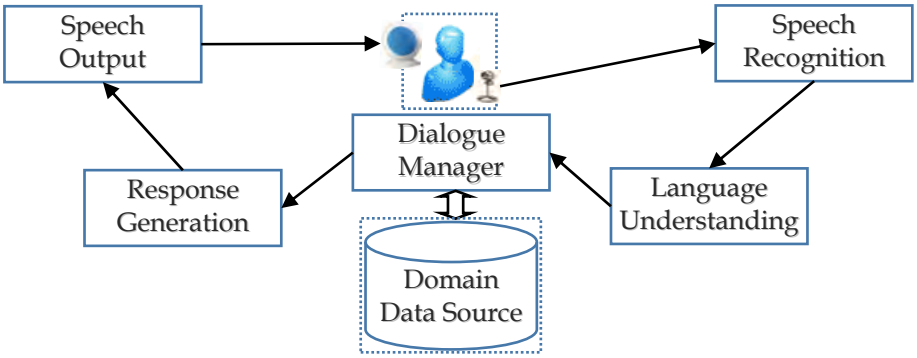


Fig. 1. Logical flow through SDS components

Current trends are putting more research emphasis on aspects of psychology and linguistics. Speech-based human-computer interaction faces several challenges in order to be more widely accepted. One of these challenges is the domain portability.

3. Spontaneous Portability

This section describes spontaneous domain portability issues, focusing a dynamic domain interaction within an AmI vision.
The design of a DM can be customized to new domains in which different dialogue strategies can be explored, concerning only 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.
This divide to conquer approach assumes that practical dialogue and domain-independent hypothesis are true (Allen et al., 2000). The reason is that typical applications of human computer interaction involve dialogue focussed on accomplishing some specific task. Assuming this, 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, is needed for reducing the complexity of SDS typical components, specially the DM.

For this, should be considered another component of SDS architecture, which handles these features, namely a Domain Knowledge Manager (Flycht-Eriksson, 2000). This component is in charge for retrieving and coordinating knowledge from the different domain knowledge sources and application systems, traditionally named background system. In these circumstances, this approach allows the customization of the DM enabling domain portability and easy configuration of the SDS architecture.
However, in this paper, the SDS is seen as a computational entity that allows universal access to AmI. In this interaction scenario, the SDS should be a computational entity that allows access to any resource by anyone, anywhere, at anytime, through any media or language, allowing its users to focus on the task, not on the tool.
Nevertheless, a traditional SDS cannot be directly used to interact with an intelligent environment, due lack of spontaneous portability, because of the fact that SDS are not ubiquitous yet. Within a ubiquitous domain, one does not know, at design time, all the resources that will be available. To address this issue, an approach for SDS architecture improvements and knowledge modelling for semantic interface design is needed (Filipe, 2007; Filipe & Mamede, 2008; Filipe & Mamede, 2009).
The SDS customization for AmI access, allowing spontaneous configuration, is supported by the proposed Environment Interaction Manager (EIM) (see Fig. 2).

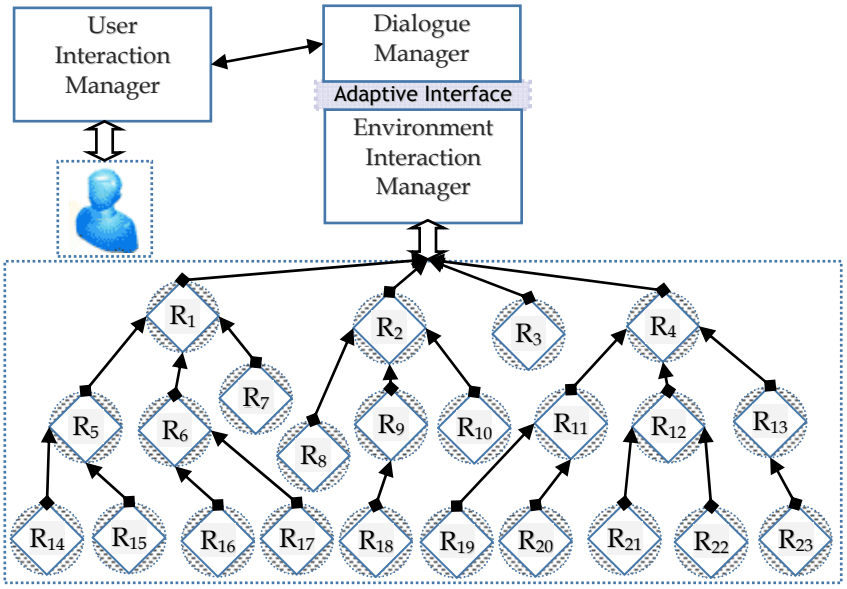


Fig. 2. SDS customization via EIM

The main goal of the EIM is to support the communication interoperability between the SDS and a set of controllable resources, performing the environment's knowledge management for allowing spontaneous configuration. For this, the EIM includes a knowledge model (see Section 4) that represents all the aggregated resource's semantic interfaces.

For instance, when it refers to an indoor environment, the knowledge reflects the plan or physical organization of the building and the SDS controllable resources. The building is modelled as a spontaneous aggregation "part-whole" of controllable resources. Each resource shares a semantic interface that exposes its functional capabilities and makes possible its manipulation by the SDS. The building itself is seen as a large controllable resource that aggregates minor resources, such as floors, rooms, entrance halls, foyers. Each one of these resources can aggregate other resources that control, for instance, doors, windows, elevators, environment controls, multimedia controls, appliance controls and so on.
When a resource, designated by "part", is activated, a discovery protocol searches the nearest resource, designated by "whole". After that, KAP is executed, in the context of the "whole", merging the knowledge built in the semantic interface of the "part" (to the "whole" knowledge model) and propagates the changes to related building parts.
In order to allow a large federation of resources, managed by the EIM, several aggregation levels are considered (see Fig. 2). A controllable resource is distinguished by a different identifier (R_1 … R_n) and can be aggregated to another resource belonging to an upper level or directly to EIM. The last aggregation level, the level of the EIM, holds all the existing resources semantic interfaces that can represent, for instance, an entire building.

3.1 Adaptive Interface

In order to allow flexible behaviour in SDS interaction, the DM must be able to handle under specified requests (when user provides only partial information). The DM must decide which task should be performed. Our approach recommends the use of domain dependent knowledge, 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 must not know the EIM's domain model, the DM must submit requests to be answered by EIM. For this, an adaptive and easy to use EIM's interface is needed (Filipe et al., 2007; Filipe et al.; 2008). The EIM presents to the Dialogue Manager (DM) an aggregated view of the environment. In this architecture (see Fig. 2), a clear separation is assumed between discourse or dialogue dependent issues (DM job) and domain dependent issues (EIM job).
The ideas behind this adaptive interface are based on the relative weight (relevance) of each concept, which is included in the request. Two independent ranking, for tasks and resources, are used to compute the best task-resource pairs. The interfaces accept as input a list of pivot concepts. The concepts reference tasks and resources, which are translated to points credited in the respective or task or resource rank.
In some cases, this ranking algorithm conduces to situations where the tasks or resources have similar or even the same ranking position. In this situation, the best task-resource pair is not clearly determined, demanding for large clarification dialogues.
One way to address this issue is by endowing the DM with the ability to interactively learn dialogue 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 the adaptive interface is not on learning complex dialogue strategies, but on dynamically adapting the relevance of task-resource pairs according to user interaction, watching the selection or rejection expressed in previous user's clarification dialogues. As an example, let's consider the case where the user is in the kitchen and selects the task "turn-on". Since "turn-on" must refer to some resource, SDS asks the user to specify the resource he/she wants to refer (e.g., a microwave oven, or the ceiling lights). Therefore, the DM must decide about which resources it should ask the user first. Consequently, some form of resource relevance is needed to enable the selection of resources according to the selected task and the previous history of user's interaction.
To allow for a dynamic adaptation considering the relevance of resources, a simple implementation of the activation potential model should be made, following the Agent Flow Model proposed by Morgado and Gaspar (Morgado & Gaspar, 2004).

4. Semantic Interface

This section gives an overview of the most relevant components of the knowledge model that holds the design of the semantic interface, which includes four independent knowledge components: the discourse model, the task model, the world model, and the events model.

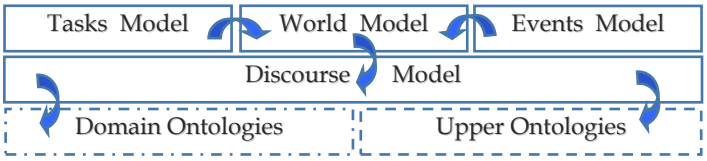


Fig. 3. Semantic Interface Knowledge Model

Additionally, external ontological knowledge components are also considered to allow the integration with domain ontologies and upper ontologies (see Fig. 3).

4.1 Discourse Model

The discourse model defines a conceptual support, grouping concept descriptions, used essentially to express functional resource capabilities. The mining of a concept is previously established or is inferred at runtime by KAP, comparing its linguistic descriptors or its semantic descriptors. For instance, a semantic descriptor can include a Uniform Resource Identifier (URI) that points to external ontology nodes.
A concept descriptor defines an atomic unit of knowledge and maps linguistic knowledge into domain knowledge. Essentially, a concept maps a set of URIs into a set of terms or more generically into a set of Multi-Word Unit (MWU). Concepts declarations include linguistic and semantic parts organized according to main kinds, which are: "task", "role", "event", "name", and "constant". Concepts of kinds "action" or "perception" hold task names. A perception task cannot modify the state of the environment, whereas, an action task can. Concepts of kinds "collection" or "quantity" hold task roles (parameters or arguments). The

kind "collection" is used to define sets of constants (represented also by concepts such as white, black, red, …) to fill task roles (colour, shape, texture, …). The kind "quantity" is about numbers (integer, real, positive, …) and the "unit" kind is for measures (time, power, …). The kind "event" holds event names. The kind "name" holds resource or class names.
In order to ensure the availability of vocabulary to refer the represented concepts, concept descriptions include linguistic properties. Each Word (or term), has a part of speech tag, such as noun, adjective, verb, adverb; a language tag, such as "pt-PT", "pt-BR", "en-UK" or "en-US"; and an optional phonetic transcription. The linguistic description holds a list of words, or more generically a MWU, referring linguistic variations associated with the concept, such as synonyms, acronyms and even antonyms.
Fig. 4 shows a mind map of a concept descriptor.

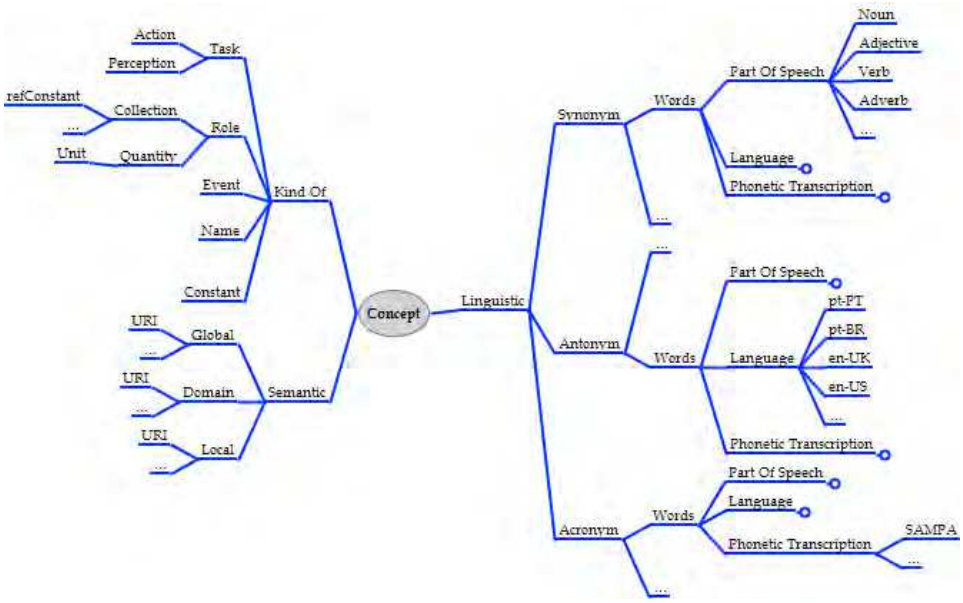


Fig. 4. Concept Descriptor

Concept descriptors can also hold semantic references typically characterized by a Universal Resource Identifier (URI). The semantic description supports references to domain knowledge sources (domain hierarchy, domain ontologies) or global knowledge sources, (upper ontologies or a lexical database, such as WordNet). A concept description must include at least one URI for local reference.
The references about knowledge sources must be unique in the same knowledge model and must be encoded using a particular data format to allow a unique identification of the referenced concept. The syntax of the knowledge source references does not need to be universal; it is enough that each particular knowledge source shares the same syntax.

4.2 Task model

The task model contains one or more task descriptions, based on concepts previously declared in the discourse model. Fig. 5 shows a mind map of a task descriptor.

A task descriptor is a semantic representation of a task that holds a task name and, optionally, a role input and/or output list. A role describes an input and/or output task argument or parameter. An input role has a name, a range, and a restriction.

The role restriction is a rule that is implemented as a regular expression and is optional. An output role is similar to an input role with an optional default constant.

The initial and final rules perform environment state validation: the initial rule (to check the initial state of the world before a task execution) and the final rule (to check the final state of the word after a task execution). These rules, also implemented as regular expressions, can refer to role names and constants returned by perception task calls.

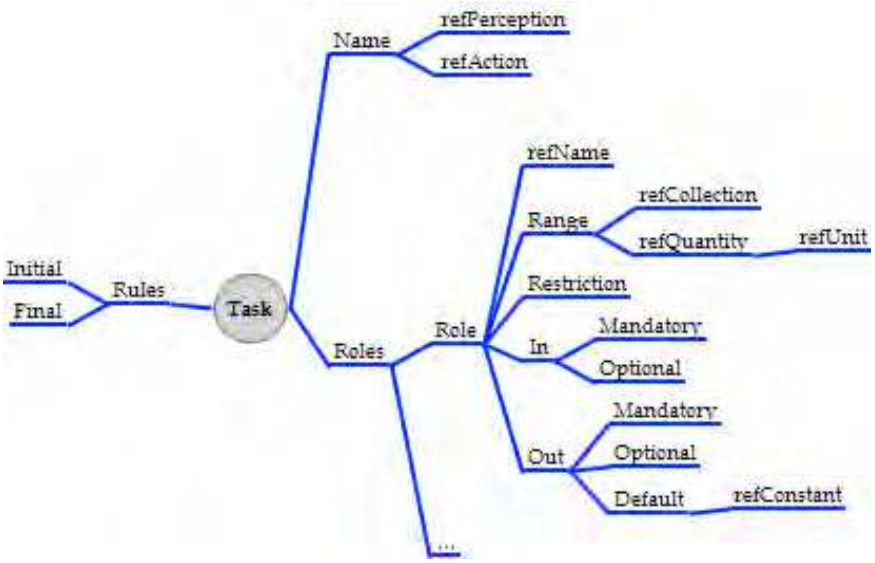


Fig. 5. Task Descriptor

4.3 World Model

The world model contains descriptions about one or more resources (the "whole" and its "parts") properties. These descriptions refer mandatorily to concepts previously declared in discourse model including, for instance, name, and optionally physical properties (colour, shape, …) that are known by the SDS user and are typically used to indentify or select a resource within a user request. Optionally, a resource description refers to one or more classes symbolized in the domain ontologies.

Fig. 6 shows a mind map of a resource descriptor.

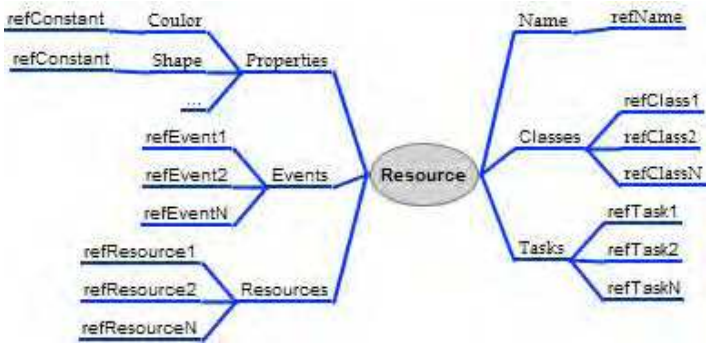


Fig. 6. Resource Descriptor

4.4 Events Model

The events model contains descriptions of events supported by concepts also declared in the discourse model. These descriptions are similar to task descriptions only with name and input roles. An event is a notification about an expected or unexpected environment state modification. The events model supports the reactive behaviour of the EIM and is used to notify the DM of the SDS about the environment changes.

5. Knowledge Aggregation Process

The main goal of the Knowledge Aggregation Process (KAP) is to update on-the-fly the knowledge model of a resource semantic interface "whole", merging the knowledge originated by one "part", that is also a resource. Generally, it is assumed that each resource holds its own built-in semantic interface. However, due to hardware limitations, the semantic interface can be maintained and virtualized by other computational entity.

At its starting point, KAP puts side by side concepts and tasks descriptions using similarity criteria:

(a) Two concepts are similar when its domain or global URIs is the same or its linguistic descriptors are literally equal. When the kind of the concepts is "collection", its member constants must be also similar;

(b) Two tasks are similar when its descriptions are literally equal;

In order to update the knowledge model of a "whole", KAP follows the next four steps:

(1) For each concept in "part", without a similar (a) in "whole", is added a new concept description to "whole" discourse model;

(2) For each task in "part" without a similar (b) in "whole" is added a new concept description to "whole" task model;

(3) A new resource description is added to "whole" world model;

(4) The resource description is linked to the updated tasks descriptions.

6. Experimental Setup

The experimental setup is based on our simulator, originally developed for Portuguese users. Fig. 7 shows a screen with a summary of the current knowledge model.

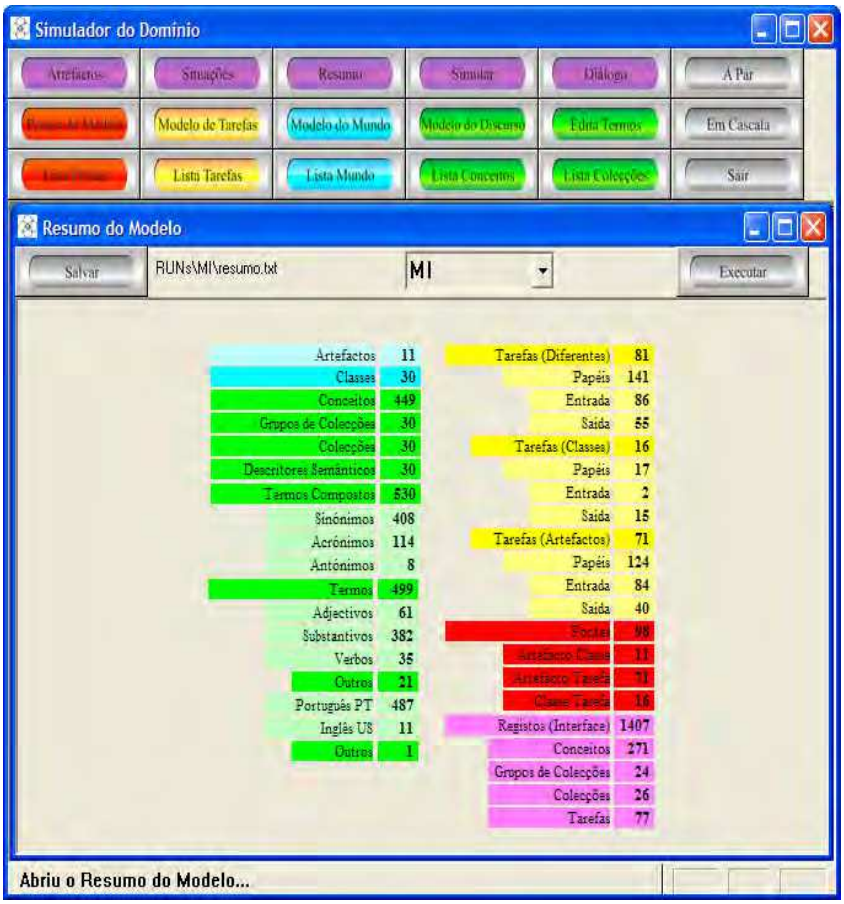


Fig. 7. Screen of the simulator with the summary of the model

The simulator incorporates an elementary dialogue manager and allows the debug of an invoked task analyzing the interaction with the target resource. Is possible to debug KAP execution, execute tasks, and observe its effects on the environment. We can also consult and print several data about each resource semantic interface. Currently, the simulator holds approximately a total of one thousand concepts and one hundred tasks.
It is possible to query the simulator about detailed descriptions of the represented concepts.
Fig. 8 shows a screen of the simulator with the description (English) of the microwave oven concept in the current knowledge model, according to its definition in WordNet (Fellbaum, 1998).

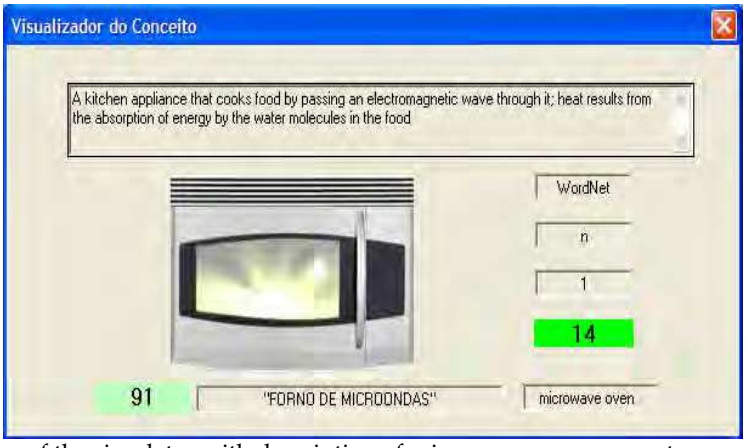


Fig. 8. Screen of the simulator with description of microwave oven concept

Considering that the environment is characterized by an arbitrary set of resources, physically supported by augmented artefacts, such as appliances, furniture, or ambient controls the simulator makes available several autonomic resource simulators, such as air conditioning, freezer, fryer, light source, microwave oven, table, water faucet, window, and window blind.
Fig. 9 shows part of the class hierarchy of resources available in the simulator.

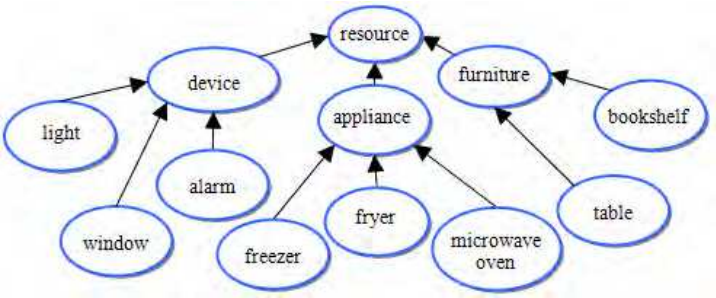


Fig. 9. Part of the simulator class hierarchy

The class hierarchy does not need to be complete because it can be improved as new resources are dynamically added.
These resources are activated to simulate and intelligent environment, composed by several floors and rooms defining aggregation levels that can be used, for instance, to model dwellings. A room is modelled as a resource that aggregates other resources physically present in that room, such as kitchen, living room, dinning room, and bedroom. A floor is a resource, which aggregates its physical parts generically named by rooms. At top level of the simulation, EIM includes a knowledge representation of the entire building.
For instance, when the SDS user demands "*turning on the light*", the simulator of the light source, aggregated by KAP as part of the kitchen simulator, executes the request turning on the kitchen light indicating by default thirty percent of luminosity.

Fig. 10 shows the screen of the kitchen light simulator, with its properties and state data, after the execution of the request "*turning on the light*".

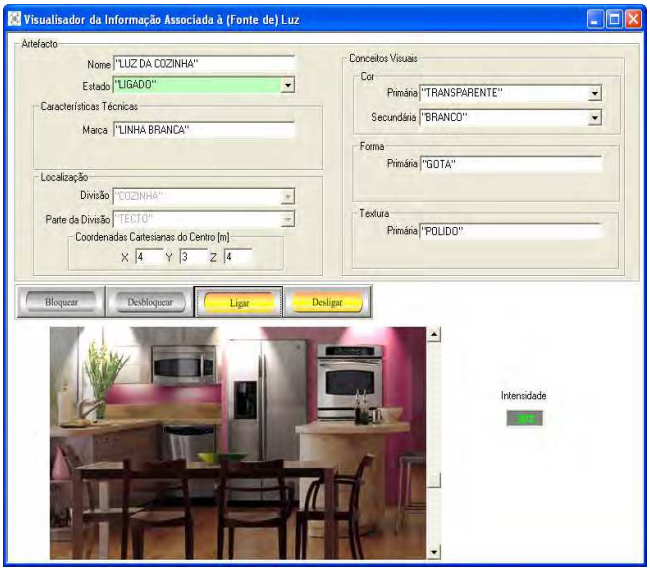


Fig. 10. Screen of the kitchen light simulator

For instance, when the SDS user demands "defrosting hamburger", the simulator of the microwave oven, aggregated by KAP as part of the kitchen simulator, executes the request indicating the automatically select power (300 watts – see symbol) and duration (8 minutes) of the defrosting process.

Fig. 11. Screen of the microwave oven simulator

Fig. 11 shows the state of the microwave oven simulator after the execution of the request "*defrosting hamburger*".

In order to start the defrosting process, after closing the door of the microwave, the user must confirm its previous intention indicating the request "*turning on the microwave*".
Fig. 12 shows 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.

Fig. 12. Screen of the deep fryer simulator

Fig. 13 shows the screen of the freezer simulator after the execution of the request "*what is the amount of carrots?*". The table in the simulator shows the selected type of food. However, in this case the dialogue manager also returns the answer "*1 package with 300 g*".

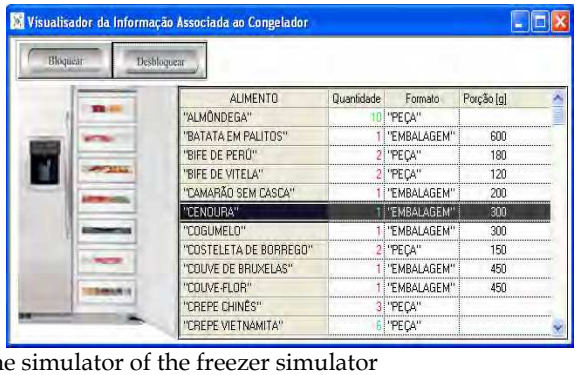


Fig. 13. Screen of the simulator of the freezer simulator

The simulator includes also the treatment of more complex user's requests, for instance, involving relational operators. For example, the request "*what is the food with amount less than five*" is redirected to the freezer simulator, by the kitchen simulator, producing the list of food with amount less than five.
The concept "carrot" (in Portuguese "cenoura") is included in the microwave oven and in the freezer simulators interfaces. However, the EIM has only one definition of the concept "carrot" in its knowledge model automatically declared by KAP at runtime.

7. Concluding Remarks

Current technologies require human intervention to solve environment reconfiguration problems. The growth in pervasive computing will require standards in real objects interoperability to achieve AmI vision. In order to face this issue, a more human like way of interaction is proposed including spoken natural language support within intelligent environments. Our proposal tries to improve the configuration features of the SDS architectures with a semantic-based approach allowing an autonomic design of semantic interfaces, which are used to describe resource capabilities. For this, was proposed EIM (Environment Interaction Manager) SDS component and KAP (Knowledge Aggregation Process) to deal, at runtime, with federations of resources. In this context, the computational environment can handle completely new resources of unknown or unseen classes, trying to cover the ubiquitous essence of natural language.

The presented ideas have been applied with success implementing the spontaneous configuration of knowledge-based resources. The proposed semantic-based approach, supported in the field by the EIM, is a significant contribution to improve the portability, scalability and, simultaneously, the robustness of the SDS being developed in our lab.

Currently, our work is based in the kitchen environment. However, we intend to generalize the use of the SDS natural interface to support inhabitants' activities, for instance, to optimize climate and light controls, item tracking and general use items, automated alarm schedules to match inhabitants' preferences, and control of media systems.

In the near future, we aim to study more deeply the knowledge replication versus knowledge integration rate. We expect to prove, for the upper aggregations levels, an interesting knowledge integration rate, due to the reuse of similar concepts and tasks within the same intelligent environment.

8. References

Aartsa, E. & Ruytera, B. (2009). New research perspectives on Ambient Intelligence, *Journal of Ambient Intelligence and Smart Environments,* Vol. 1, pp. 5–14

Allen, J.; Byron, D.; Dzikovska, M.; Ferguson, G.; Galescu, L. & Stent, A. (2000). An Architecture for a Generic Dialogue Shell, *Natural Language Engineering*, Vol. 6, 3-4, pp. 213-228, Cambridge University Press

Cook, D. & Das, S. (2004). *Smart Environments: Technology, Protocols and Applications,* Wiley Interscience, ISBN: 978-0-471-54448-7

Dahl, D.; Kliche, I.; Tumuluri, R.; Yudkowsky, M.; Bodell, M.; Porter, B.; Raggett, D.; Raman, T. & Wahbe, A. (2008). Multimodal Architecture and Interfaces, *http://www.w3.org/TR/mmi-arch/*

Ducatel, K.; Bogdanowicz, M.; Scapolo, F.; Leijten, J. & Burgelman J. (2001). Scenarios for Ambient Intelligence in 2010, *ISTAG: European Commission Report*

Dzikovska, M.; Allen, J. & Swift, M. (2003). Integrating Linguistic and Domain Knowledge for Spoken Dialogue Systems in Multiple Domains. *Proceedings of Workshop on Knowledge and Reasoning in Practical Dialogue Systems at The Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-2003)*, pp. 25-35, Acapulco, Mexico, August 2003

Fellbaum, C. (Ed.) (1998). *WordNet, an Electronic Lexical Database*, MIT Press, Cambridge, MA, United States

Filipe, P. & Mamede, N. (2008). Empirical Multi-Artifact Knowledge Modeling for Dialogue Systems, *Proceedings of 11th International Conference on Enterprise Information Systems*, pp. 286-292, Barcelona Spain, June 2008

Filipe, P. & Mamede, N. (2009). Indoor Domain Model for Dialogue Systems, *Proceedings of 13th International Conference on Human-Computer Interaction "Universal Access in HCI, Part III, LNCS 5616"*, pp. 512-520, ISBN: 978-3-642-02712-3, San Diego CA USA, July 2009, Springer-Verlag, Berlin, Heidelberg

Filipe, P. (2007). *Dynamic Integration of Artefacts in Dialogue Systems*. PhD Thesis, Technical University of Lisbon UTL/IST

Filipe, P.; Araújo, P. & Mamede, N. (2008). Dialogue Systems Domain Interaction Using Reinforcement Learning, *Proceedings of 11th Ibero-American Conference on Artificial Intelligence (IBERAMIA) - Workshop on Agreement Technologies (WAT 2008)*, Lisboa, Portugal

Filipe, P.; Morgado, L. & Mamede, N. (2007). An Adaptive Domain Knowledge Manager for Dialogue Systems, *Proceedings of 9th International Conference on Enterprise Information Systems (ICEIS 2007)*, pp. 45-52, Funchal, Portugal

Flycht-Eriksson, A. (2000). A Domain Knowledge Manager for Dialogue Systems, *Proceedings of 14th European Conference on Artificial Intelligence (ECAI 2000)*, IOS Press, Amsterdam, Amsterdam

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

McTear, M. (2004). *Spoken Dialogue Technology: Towards the Conversational User Interface,* Springer-Verlag, ISBN 1-85233-672-2, London, Berlin, Heidelberg

Minker, W.; López-Cózar, R. & McTear, M. (2009). The Role of Spoken Language Dialogue Interaction in Intelligent Environments, *Journal of Ambient Intelligence and Smart Environments,* Vol. 1

Morgado, L., Gaspar, G. (2004). Focusing Reasoning Through Emotional Mechanisms. *Proceedings of 16th European Conference on Artificial Intelligence (ECAI 2004)*. IOS Press. Valencia, Spain.

Newell, A. (1982). The Knowledge Level, *Artificial Intelligence*, Vol. 18, No. 1, pp. 87-127

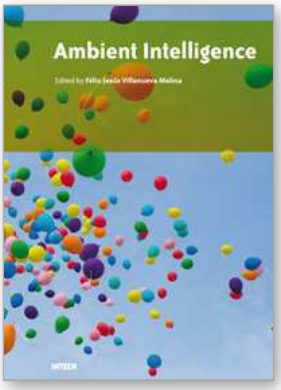
Reeves, B. & Nass, C. (1996). *The Media Equation: How People Treat Computers, Television and New Media Like Real People and Places*, Cambridge University Press, ISBN: 157586052X, Cambridge, Mass

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*, Vol. 21, No. 2, pp. 97-126

Streitz, N.; Kameas, A. & Mavrommati, I. (Eds.)(2007). *The Disappearing Computer: Interaction Design, System Infrastructures and Applications for Smart Environments,* Springer Publishers, ISBN: 3-540-72725-6, Berlin, Heidelberg, New York

Weiser, M. (1991). The Computer of the Twenty-First Century. *Scientific American,* Vol. 265, No. 3, pp. 94-104

Zue, V. & Glass J. (2000). Conversational Interfaces: Advances and Challenges. *IEEE,* Vol. 88, No. 8, pp. 1166-1180



Ambient Intelligence

Edited by Felix Jesus Villanueva Molina

ISBN 978-953-307-078-0
Hard cover, 144 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

It can no longer be ignored that Ambient Intelligence concepts are moving away from research labs demonstrators into our daily lives in a slow but continuous manner. However, we are still far from concluding that our living spaces are intelligent and are enhancing our living style. Ambient Intelligence has attracted much attention from multidisciplinary research areas and there are still open issues in most of them. In this book a selection of unsolved problems which are considered key for ambient intelligence to become a reality, is analyzed and studied in depth. Hopefully this book will provide the reader with a good idea about the current research lines in ambient intelligence, a good overview of existing works and identify potential solutions for each one of these problems.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Porfirio Filipe and Nuno Mamede (2010). Ambient Intelligence Interaction via Dialogue Systems, Ambient Intelligence, Felix Jesus Villanueva Molina (Ed.), ISBN: 978-953-307-078-0, InTech, Available from: http://www.intechopen.com/books/ambient-intelligence/ambient-intelligence-interaction-via-dialogue-systems

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821