Facts and challenges in a dialogue system for Smart Environments

Germán Montoro, Xavier Alamán and Pablo A. Haya
Universidad Autónoma de Madrid
Departamento de Ingeniería Informática
Ctra. Colmenar Viejo, km. 15. Madrid 28049, Spain
{German.Montoro, Xavier.Alaman, Pablo.Haya}@ii.uam.es

Abstract

This paper presents two dialogue spoken interfaces for smart environments. A smart environment is an interactive space that may communicate with the user by multiple modalities. A domain ontology has been defined based on XML. This smart environment ontology is implemented in a middleware layer, called blackboard. It allows to modify and get the state of the environment, providing environmental information that is used by the dialogue systems (as well as other interfaces) to carry on a conversation and enhance their performance. The first dialogue system is Odisea. It has a dialogue for every task and a supervisor that decides which dialogue takes the control. The second dialogue system is Odisea II. It is a continuation of the first system, where dialogues are entity oriented. They can be generated automatically and are represented on one or several trees. The load of the dialogues is supported by the supervisor. They adapt more easily to different environments and integrate better with other modal interfaces.

1 Introduction

Our work is related to the research area known as smart environments or active spaces. A smart environment is a "highly embedded, interactive space that brings computation into the real, physical world". It allows computers "to participate in activities that have never previously involved computation" and people "to interact with computational systems the way they would with other people: via gesture, voice, movement, and context" [Coen, 1998]. Thus, these new environments present new challenges [Shafer, 1999] that must be addressed by the research community. Different and highly heterogeneous technologies can be found inside a smart environment, from hardware devices, such as sensors, switches, appliances, webcams… to legacy software, such as voice recognizers, multimedia streaming servers, mail agents… On the one hand, all of these entities have to be seamlessly integrated and controlled using the same user interface. For instance, a user has to be able to start a broadcasting music server as easily as to turn off the lights. On the other hand, user interaction has to be kept as flexible as possible. It should be based on multiple and distinct modalities, such as web, voice, touch… so that user preferences and capabilities can be considered. Moreover, the environment configuration is highly dynamic and it may change from one environment to another. New entities can be added, removed or temporally stopped, and the user interface should be aware of these changes.

A working prototype has been developed, including: an ontology that provides a simple mechanism to communicate context changes, and two different user interfaces (a web-based user interface and a spoken dialogue user interface) that interact with and manage the entities of a real environment.

This real environment consists of a laboratory separated in two rooms. Several devices have been spread out across the rooms. There are two kinds of devices: control and multimedia. Control devices are lighting controls, door mechanism, presence detector, smart-cards, etc. Multimedia devices, such as speakers, microphones, a TV and an IP video-camera, are accessible through a backbone IP. Control devices are connected to an EIB (EIBA) network, and a gateway joins the two networks. The blackboard that accesses to the physical layer is harmonized through a SMNP (Simple Management Network Protocol) layer.

Web and dialogue interfaces interact with the environment and cause changes in the laboratory. In the same way, a user can get real information from the environment. Figure 1 shows a snapshot of the laboratory where the environment is implemented.

This paper is organized as follows: section two describes the ontology; section three presents the spoken dialogue user interface; in section four we describe the new challenges for this user interface and; finally, in section five we discuss the conclusions and future work.

Figure 1. Snapshot of the laboratory

2 Ontology description

The ontology is implemented in a middleware layer, which is the glue between user interfaces and the environment. The interaction between them is based on an event-driven protocol. The contextual changes are published in a common repository, called blackboard [Engelmore and Morgan, 1988], Applications subscribe to the blackboard for the entity changes that they are interested in. This communication architecture allows to maintain a loose coupling among layers, facilitating the dynamic reconfiguration of environmental entities and user interfaces. Interfaces employ the information provided by the blackboard to adapt dynamically to the environment configuration and to interact with it. For example, the number of persons in the room, the task they are performing and the status of several physical devices (lights, heating, video/audio displays) are represented on the blackboard and used by the spoken dialogue user interface.

The blackboard holds a formal representation of all the environment active entities. The nature of each entity can range from a physical device to an abstract concept, such as the number of persons in the environment. Each entity comprises a set of properties that can be common or specific. All entities of the same type share a set of common properties that describe universal accepted features of an entity. Specific properties represent custom application information. Each application can annotate the entity representation, so that it can customize the blackboard to its own requirements. The blackboard is not only a set of entities but also stores the relationships among them. A relationship can be of any kind (association, aggregation...) and any direction (unidirectional or bidirectional). It has not a explicit semantic associated, hence a relationship representation can range from location information (a device is inside a room) to the flow of multimedia information among physical devices (microphones, speakers, cameras, displays, etc.).

The blackboard is a graph where each node is an entity, and the relationships are arcs. This structure allows to use several abstraction levels to organize the context information. The deepest nodes represent more concrete properties, while top nodes may reflect structural relationships among components. The blackboard is a snapshot of the state of the environment at every specific moment.

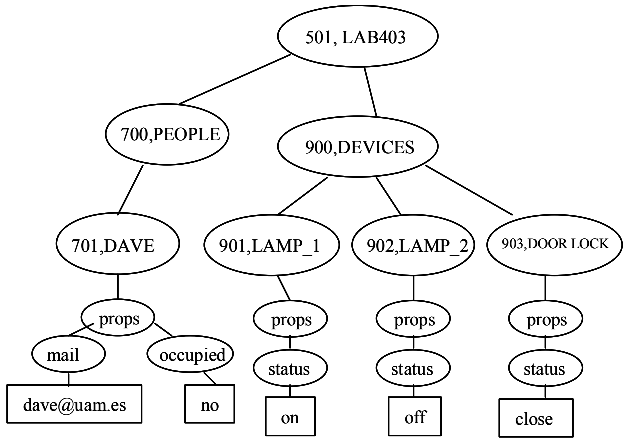

Figure 2. A schematic blackboard graph example

Every node that belongs to the blackboard graph has one or more node identifiers (nid) and a name. In order to navigate this graph we have implemented two independent naming mechanisms. There is a basic namespace in which each entity has a unique numerical identifier. Applications can directly access to an entity and its properties by this number. Otherwise, an entity can be referenced by concatenating the name of all of its parent entities. E.g.: /lab403/devices/lamp_1/props/status. Moreover, it is allowed to use wildcards to reference more than one entity at the same operation. For instance, it is possible to obtain the state of all the devices inside the environment. E.g.: /lab403/devices/*/props/status.

Every environment entity publishes an XML description of its features in the central repository. This repository is used as a proxy context information server. Applications and interfaces may ask the blackboard to obtain information about the state of any entity and change this state. Entities descriptions can be added to and removed from the blackboard in run-time, and the new information can be reused for the rest of applications or interfaces. In summary, applications and interfaces can perform the following operations in the blackboard: retrieve information about the entities and their properties, control the state of the entities and subscribe to changes in the environment.

Every blackboard is a server that can be accessed using client-server TCP/IP protocols. The interaction between applications and the blackboard uses the HTTP protocol, technologically independent of the entity nature. HTTP has been chosen as the transport protocol because it is simple and widely spread. XML has been chosen as the lingua franca among layers, because it is an industry standard language to exchange information among applications.

3 Dialogue system, so far

One of the user interfaces developed to control the environment is provided by means of natural language spoken dialogues. This is called the Odisea system. It uses the ontology represented on the blackboard to carry on conversations related to entity control and user information. Blackboard representation does not have to suffer any modification and it is independent of other user interfaces.

The system is composed of a specific dialogue for every task. For instance, the light dialogue controls the lamps of a room. Since dialogues are focused on tasks, a light dialogue must control all the lights in the same environment. This means that if we add or remove a lamp in the environment we will have to modify that specific dialogue to be consistent with the new situation.

Since access and permit protection has not been developed yet, every dialogue has full access to read or modify any value stored on the blackboard.

Dialogues implementation is similar to Schank's scripts concepts [Schank and Abelson, 1977]. A dialogue is formed by a template with gaps that must be filled. The dialogue will guide the user through the script until the template is fulfilled.

Dialogues are independent from each other, so they do not exchange information directly and they are not aware of the state of the other dialogues. The way dialogues exchange information with other dialogues and with any other application is by means of the blackboard.

A dialogue template does not have to be fulfilled in one single sentence. In that case, the dialogue system keeps its state, so that it can be completed in next sentences. In the template completion process the dialogue may modify or read the blackboard information, so that it can inform to the rest of the world of a new state or it can get support to finish the current task successfully.

When a dialogue is completed, it is in charge of performing the actions associated to it. They can be as different as making a physical action in the environment, providing information to the user or answering a user question. For instance, after a "turn on the lights" sentence the dialogue script may finish by modifying the lights state on the blackboard, so that it will turn the laboratory lights on, or by producing the sentence "lights are already on", after checking on the blackboard the lights state. In any case, if a dialogue needs to know or modify the state of the environment it will do it by accessing to the blackboard and never by communicating with the environment entities directly.

Since speak recognition is not completely accurate, the blackboard ontology plays an important rule in the dialogues. When a speaker sentence (or recognizer output) may deal with ambiguities, the dialogue accesses to the information stored on the blackboard to try to solve it. Moreover, it can offer solutions to the user depending on the current context. For instance, if the recognizer output is only formed by the word "lights", the dialogue can check the state of the lights. If they were off, it could directly offer the user to turn them on.

Every dialogue comes with an associated grammar. The grammar defines the possible sentences that may be uttered by a user. These grammars tend to be wide, allowing the user to interact with the environment in a more natural way and avoiding the use of fix sentences or commands. Dialogues can be activated or deactivated by enabling or disabling their associated grammar. This may help to improve the recognition accuracy by decreasing the number of possible sentences that the recognizer has to consider.

Odisea is running every active dialogue at the same time in different execution threads. However only one of these dialogues can finally take the control. For this, dialogues have to compete to be the one that deals with the current conversation. A dialogue supervisor is in charge of choosing the most accurate dialogue depending on the user input (provided by the speak recognizer) and the contextual information from the blackboard. It can also activate and deactivate some of the dialogues when they come in or out of scope.

This supervisor receives the user utterance. Then it sends it to all the active dialogues (that are in a waiting state). Dialogues receive the utterance and process it in order to find out how many gaps from their template it would complete and how many would keep empty. Every dialogue answers to the supervisor with the percentage of its template completion, considering the gaps filled in previous sentences and the gaps that the new sentence would fill. Supervisor collects all this information from every dialogue and sees which dialogue has the biggest percentage. If this percentage is the same in more than one dialogue the supervisor will consider if it was already selected in the preceding iteration, in order to continue with the task that the user carried on in the previous utterance. After this decision, only the dialogue selected by the supervisor can go on with its script and perform some task. The other dialogues will have to wait for the next utterance.

Given that after a new sentence the supervisor can jump from one dialogue to another, the user can start performing a task and, whether or not it was concluded, change to a different one, start a new dialogue script or go back to a previous one. This provides a lot of flexibility to the process of interacting with the environment and allows the supervisor to recover easily from previous misunderstandings.

As a precaution to avoid confusions with a regular user utterance (for instance, if she is speaking with another person), the supervisor is not always "listening" to the users utterances. To activate the system, she must pronounce the word "Odisea". This makes that the supervisor wakes up, prompts the user and activates the grammars associated to the dialogues. After that moment, she can start interacting with the environment and Odisea will consider that all the utterances are addressed to it. If a user does not utter anything eight seconds after a dialogue was processed, the supervisor will go to "sleep" again.

Given that the supervisor and every different dialogue run in a different execution thread, once the supervisor has chosen the most accurate dialogue it is ready to receive a new user utterance from the recognizer, even though the selected dialogue is in the middle of its process. In that case the supervisor waits until the current dialogue ends its script and uses this new utterance to select the next running dialogue. This is specially useful to allow the user to answer before a synthesizer utterance is over. Although the supervisor does not interrupt the dialogue process (and the synthesizer sentence will not be interrupted) the user utterance is stored to be used as soon as the dialogue has finished its process.

The speech recognizer and synthesizer employed in Odisea were not specially designed for the system but are widely spread commercial tools. Recognizer is speaker-independent and has not been trained to be used with any specific subject. This provides more flexibility to the system and the recognition accuracy does not degrade dramatically thanks to the use of grammars and contextual information from the blackboard ontology.

Odisea has been tried for four days in a general public fair. People could use the dialogue system and change or ask for the physical state of the laboratory. They could see the results of their utterances through a real-time webcam connected with the laboratory. Users from any age or gender utilized the system without receiving special instructions. In most of the cases the experience was fully satisfactory and the users could interact with the environment from the first sentence.

4 The new dialogue system

The Odisea system has become a testbed to analyze the necessities and challenges in the development of a spoken dialogue system for smart environments. As a result, a new dialogue system, denominated Odisea II, is under construction.

One of the main drawbacks of the Odisea system was that the dialogues were task oriented. This meant that if we added or removed a entity on the blackboard we had to modify the dialogue that grouped all the entities performing similar tasks. On the one hand this might raise the complexity of a dialogue when similar entities were added. On the other hand if an entity was removed, to adapt the dialogue to the new situation could not be a trivial task. It also made very difficult to create or modify dialogues automatically and it always required human supervision.

In the new system the blackboard has a set of possible entity definitions. Every entity stored on the blackboard has to belong to one of these entity definitions. Some entity definitions have associated information that is useful to the interfaces. This information comprises the actions that can be taken with the entity (what is called the verb part), the name it can be given (the object part), where it is in the environment (the location part) or other specific information (additional information part).

As we already said above, the nature of these entities can range from a physical device to an abstract concept

Every dialogue is focused on a single entity and is created automatically by the supervisor. At startup, the supervisor reads all the entities with interface information from the blackboard. With this information it creates a tree that serves as the new dialogue structure. For everyone of these entities the supervisor adds a new structure to the tree that corresponds to a new dialogue. For instance, let us suppose that our environment is composed of one fluorescent ceiling lamp and one floor lamp (that we can turn on and off) and a door (that we can open or close). The process to generate the dialogues follows the next steps:

- Firstly, Odisea II reads from the blackboard that there is an entity with the "light_ceiling" entity definition and with interface information associated. Now the system knows that any "light_ceiling" entity can be turned on or off (what corresponds with the verb part), that it can be named as fluorescent (what corresponds with the object part) and, only if there are several entities of the same kind, where it is in the environment (what correspond with the location part). The system also counts with a grammar template for all the grammars of the kind "light_*" and dictionaries of synonym and verb conjugations for the object and verb parts. With all this information it is able to create a specific grammar for this entity (if no previous entity of the same kind has created the grammar before). After that, the supervisor can add the branches that determinate the new dialogue to the tree. Figure 3 shows a simplified example of part of the generated tree.

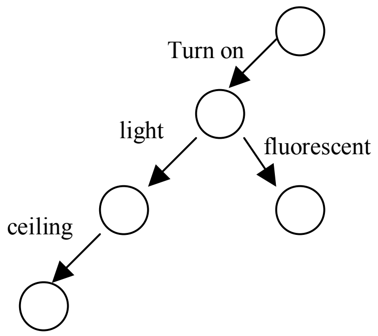

Figure 3. Partial tree after first iteration

This example tree is generated after getting that a fluorescent can also be referenced as a ceiling light from the synonyms dictionary. We have omitted the "turn off" branch and other branches to simplify the tree representation.

- Secondly the supervisor reads a "light_lamp" entity from the blackboard and employs the same process to generate a new grammar based on the "ligh_*" grammar template and add the new parts to the tree.
- Finally, the supervisor reads a "door" entity from the blackboard. Following the same process it creates a new grammar and completes the tree. A simplified final tree is shown in figure 4.

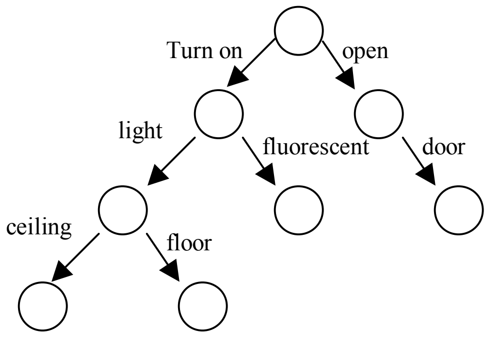

Figure 4. Simplified final tree

Notice that only the leafs determinate final states for a dialogue and that any other node implies that the user utterance or utterances were not enough to complete a task.

In Odisea II dialogues do not take the control until the dialogue supervisor has reached a leaf. In that case, the dialogue can start a specific script related to the task. Some of these specific scripts only have to perform a single action on the blackboard (for instance, to control a device) and the others become much simpler than in the previous system. This is because the supervisor is now in charge of performing many of the common tasks that were carried by the dialogues previously. For instance, if the speak recognizer produces the sentence "turn on the light" the supervisor, based on the tree information, will prompt "the ceiling or the floor light?" automatically. Furthermore, the leaves specific script dialogue can be common to the same entities definitions. The leaves script dialogue is already pre-defined and we will only have to create a new one in special situations. This means that most or all dialogue interfaces based on basic interaction with the environment can be created automatically.

This tree information is not the only information available for the supervisor. It can also employ the blackboard information to make dialogues more precise. In the previous example, the supervisor could have read the state of the lamp and ceiling lights from the blackboard and built an utterance with only those lights that were off at that moment.

More trees can also increase the dialogues accuracy. Besides the tree we have seen above, the supervisor can also create another tree based on the same information. In this new tree the object parts hang from the root (instead of the verb parts) so that the supervisor can also navigate through this tree and, if necessary, consider and offer the user all the possible actions that can be performed with an entity.

The supervisor always memorizes the state of the tree from the last utterance, so that when a user says something, the supervisor goes down through the tree from the root and from the place were the supervisor finished in the last sentence (if it was not a leaf). If any of both gets a leaf the supervisor will start the specific script task associated to the leaf. If not, it will take into account the dialogue state in a lower level.

In the old dialogue system was uneasy to integrate other modalities with a dialogue. In Odisea II, the information employed to generate the dialogues is not intrinsic to the supervisor but it is available for any other interface that requires it. This makes easy to create a multimodal interface. For instance, we can see what happens with a radio control dialogue. If the user utters "turn on the radio", the supervisor, by means of the synthesizer, can offer her all the possible radio stations represented on the tree information. However, if the number of alternatives is high, it can also call a graphical module that shows them on a display. The user can choose the station by answering back or by pointing one on the display. Since both interfaces have access to the same radio entity information stored on the blackboard, interfaces can be generated automatically and used independently.

5 Conclusions and future work

In this paper we have presented Odisea and Odisea II, two dialogue spoken interfaces for smart environments. A real smart environment has been implemented and users can interact with it through these interfaces (and through a web-based interface that was out of the scope of this paper). An ontology of the smart environment has also been created and it is represented on a blackboard. Dialogue systems employ this ontology to create the dialogues, enhance their performance and manage the conversations.

Odisea is a working dialogue system that runs task oriented dialogues simultaneously. Process load is on the dialogues side, what made harder their maintenance, adaptation to environmental changes and integration with different modalities.

Odisea II is the continuation of the initial dialogue system. It is still under development and tries to enhance the weak points of the previous system. Odisea II is entity oriented. Information related to the entities represented on the blackboard ontology is available for every interface. This allows to simplify the creation or modification of dialogues. Moreover, it permits to create dialogue interfaces or other modal interfaces automatically. As a consequence, the integration of multimodal interfaces with the dialogues also becomes much easier.

In the future we still have to solve others challenges in the new dialogue system. Some of them are already on the table and others will appear once Odisea II is fully completed. Trees are based on the idea of verb and object parts but new dialogues can extend this idea to use more parts. In that case, these new dialogues can create new complementary trees or they can be integrated in the already existing trees. Based on the blackboard information, dialogues are generated automatically or almost automatically and become part of a common tree. Nevertheless, the system may also be composed of other kind of dialogues that are not covered yet. For instance, those dialogues that need information from more than one entity or those that are not necessarily focused on blackboard entities. We still have to explore the possibilities of merging Odisea and Odisea II to support any possible dialogue that can take place in a smart environment.

Acknowledgments

This paper has been sponsored by the Spanish Interdepartmental Commission of Science and Technology (CICYT), project number TIC2000-0464.

References

[Coen, 1998] Coen, M.H. Design Principles for Intelligent Environments. In Proceedings of the AAAI Spring Symposium on Intelligent Environments (AAAI98). Stanford University in Palo Alto, California, 1998.

[Engelmore and Morgan, 1988] Engelmore, R. And Mogan, T. *Blackboard Systems*, Addison-Wesley, 1988.

[Schank and Abelson, 1977] Schank, R. and Abelson, R. Scripts, Plans and Goals. Erlbaum, Hillsdale, New Jersey, 1977.

[Shafer, 1999] Shafer, S. Ten dimensions of ubiquitous computing. In Proceedgins of 1st International Workshop on Managing Interactions in Smart Environments (MANSE'99). Dublin. Ireland, 1999.