Using context information to generate dynamic user interfaces

Xavier Alamán, Rubén Cabello, Francisco Gómez-Arriba, Pablo Haya, Antonio Martinez, Javier Martinez, Germán Montoro

Departamento de Ingeniería Informática
Universidad Autónoma de Madrid
Ctra. Colmenar Viejo, km. 15 Madrid 28049. SPAIN
Xavier.Alaman@ii.uam.es

Abstract

This paper deals with the use of context information to generate dynamic user interfaces. Our framework is a real environment composed of a heterogeneous set of components. The nature of each component can range from a physical device to an abstract concept such as the number of persons in the environment. A middleware, that provides an unified environment model and communicates context changes, is used by two different modal interfaces. This allows to manage environment components without interfering each other.

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 components, such as sensors, switches, appliance, webcams… to legacy software, such as voice recognizers, multimedia streaming servers, mail agents… On one hand, all of these components have to be seamless 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 components can be added, removed or temporally stopped, and the user interface should be aware of these changes. In our approach, contextual information, gathered from the environment and its components, is used to generate dynamic user interfaces in order to address the problems explained above. This contextual information answers the main questions: who is the user, which of the components can be controlled, and how a user prefers to do it.

A working prototype is described, including: a middleware, that provides an unified context information model and a simple mechanism to communicate context changes, and two different user interfaces, that use it to manage the components of a real environment.

2 Framework description

We have developed a middleware layer. That 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 devices changes they are interested. This communication architecture allows to maintain a loose coupling among layers, facilitating the dynamic reconfiguration of environment components 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 people in the room, the task they are performing and the status of several physical devices (lights, heating, video/audio displays) are represented in the blackboard, and are used by the natural language dialogue-management interface.

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

The blackboard is a graph where each node is a component, and relationships are arcs. In order to navigate this graph we have implemented two independent naming mechanisms. There is a basic namespace in which each component has a unique numerical identifier. Applications can directly access to a component and its properties by this number. Otherwise, a component can be referenced by concatenating the name of all of its parents components. An application can define its own relationship between components and locate a component using its own namespace. Moreover, it is allowed to use wildcards to reference more than one component at the same operation. For instance, it is possible to obtain the state of all the lights inside the environment.

Every environment component publishes a XML description of its features in the central repository. This repository is used as a proxy context information server. Applications may ask the blackboard to obtain information about to the state of any component and change this state. Components descriptions can be added and removed to the blackboard in run-time, and the new information can be reused for the rest of applications. In summary, applications can perform the following operations in the blackboard: retrieve information about the entities and its properties, control the state of components 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 component 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 is a standard industry language to interchange information between applications.

3 Applications

3.1 Jeoffrey

Jeoffrey is a web based interface developed to control environment's devices and appliances. It is a custom and partial view of the contextual information stored in the blackboard. Jeoffrey is programmed to be used in a home environment. At start up, it creates a list of the house rooms, and for each room it generates a map that includes the location of the physical devices. Each device is represented by an image. A custom widget is showed when a user clicks on a device image, allowing to control the device. The layout is composed overlapping a fixed image background with each device representation image and is generated every time Jeoffrey is loaded.

The blackboard contains generic information regarding the number of rooms and the devices they host. Each device is represented in the blackboard, and its representation includes the properties required to control it. Thus, Jeoffrey gathers this information to generate rooms views dynamically. This is not Jeoffrey-specific information but it also can be used by any other application. Besides, device representation can include specific information required by Jeoffrey, such as layout coordenates or image representation url. This information is also held by the blackboard but not interferes in other applications performance.

The device control panel is also generated dynamically and it depends on device properties. A set of generic widgets has been defined, such as text areas, buttons, sliders, etc. Therefore, a one-to-one translation between properties and widgets has been established. When device management is required, Jeoffrey reads its properties description from the blackboard, translates properties to widgets and generates a custom control panel composed by the aggregation of simple widgets. Jeoffrey has its own blackboard name-space to facilitate devices naming. The blackboard is a set of devices organized by rooms where each device is located appending the device name to the room name.

When Jeoffrey sets up, it queries the blackboard to obtain all the rooms. For each room, it asks for all the available devices. If a device has been Jeoffrey annotated, the applet reads its particular configuration and generates the web representation and the widgets needed to control it.

Moreover, Jeoffrey uses the blackboard as a proxy to manage the physical devices, like changing the volume speaker, switching the lights, etc., and to receive the changes occurred in the environment. Jeoffrey is subscribed to all the devices events, thus every change in a device state is reflected in the user interface. For instance, a widget named alarm has been defined. If a property has associated a widget alarm, when its value changes, the blackboard will notify to Jeoffrey and it will modify the color of the web image representation. Thus, a Jeoffrey instance can easily coordinate with other applications or Jeoffrey instances.

Left side of Figure 1. shows a Jeoffrey user interface screenshoot. Displayed windows correspond to device control panels. At the right side, a webcam shot shows the experimental environment.

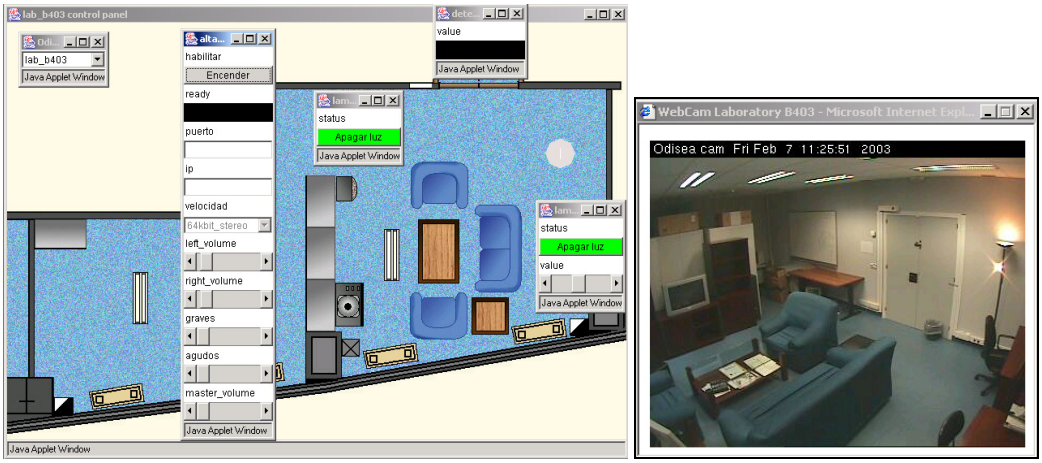

Figure 1: Jeoffrey's Screenshot and webcam laboratory view

3.2 Odisea

An alternative interface to control the environment is provided by means of natural language spoken dialogues. This is called the Odisea system. It uses the contextual information stored in the blackboard to carry on conversations related to device control and user information. Blackboard representation does not have to suffer any modification and it is independent of the Jeoffrey-specific representation.

Odisea is running several environment-specific dialogues that 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 recogniser) and the contextual information from the blackboard. This supervisor can also activate and deactivate some of the dialogues when they come in or out of scope.

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.

Every active dialogue will inform to the supervisor of the state of its template (how much it is completed) after a user utterance. Supervisor collects all this information and gives the control to the most appropriate dialogue.

Every dialogue is focused on a specific task. For instance, the lights dialogue controls the lamps of a room. It has full access to read or modify the values of the lamps stored in the blackboard, as much as other information from the blackboard that can become useful in the dialogue.

Since speak recognition is not completely accurate, context information from the blackboard plays an important rule in the supervisor decisions. When a speaker sentence (or recognizer output) may deal with ambiguities, the supervisor accesses to the information stored in 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 the word "lights", the supervisor can check the lights state. If they were off, it could directly offer the user to turn them on.

Odisea dialogues are fully compatible with Jeoffrey's interface. They can be used independently or at the same time. They show two ways of accessing and control to the environment. New interfaces can be added easily and it will not be necessary to modify the blackboard information.

4 Environment

A running prototype has been developed. It 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-card, etc. Multimedia devices, such as speaker, microphones, TV and an IP video-camera, are accessible through a backbone IP. Control devices are connected to a EIB (EIBA) network, and a gateway joins the two networks. The blackboard that accesses to the physical layer is harmonised through a SMNP (Simple Management Network Protocol) layer that is described elsewhere (Martinez et al., 2003). Both interfaces, Jeoffrey (web-based and automatically generated from the blackboard) and Odisea (natural-language based) can be used to interact with the environment.

5 Conclusions

The user interface required by a smart environment depends on user preference and environment context. Both of them can change over the time, thus the user interface should adapt dynamically. A middleware has been developed in order to facilitate context changes communication. This middleware has been tested in a real environment composed by several devices. Two independent applications allow users to interact with environment devices without interfere between them.

6 Future Work

Our current work is focused on what can be controlled. Future researching is oriented to the other two questions, how and who. For this reason, we will develop new user interfaces using mobile devices, such hand-held PCs, laptop PCs and cell phones. Moreover, user modeling and dialogues management will be improved.

References

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

EIBA. European Installation Bus Association. http://www.eiba.com.

Engelmore, R. And Morgan, T. (1988). *Blackboard Systems*, Addison-Wesley.

Martinez, A.E., Cabello, R., Gómez, F.J. and Martínez, J. (2003). Interact-DDM: A Solution for the Integration of Domestic Devices on Network Management Platforms. IFIP/IEEE International Symposium on Integrated Network Management Colorado Springs, Colorado, USA.

Schank, R. and Abelson, R. (1977). Scripts, Plans and Goals. Erlbaum, Hillsdale, New Jersey.

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