Facilitating the Design of Multi-channel Interfaces for Ambient Computing

José Rouillard
LIFL Laboratory – University of Lille1
59655 Villeneuve d’Ascq Cedex - France
jose.rouillard@univ-lille1.fr

Jean-Claude Tarby
LIFL Laboratory – University of Lille1
59655 Villeneuve d’Ascq Cedex – France
jean-claude.Tarby@univ-lille1.fr

Xavier Le Pallec
LIFL Laboratory – University of Lille1
59655 Villeneuve d’Ascq Cedex - France
xavier.le-pallec@univ-lille1.fr

Raphaël Marvie
LIFL Laboratory – University of Lille1
59655 Villeneuve d’Ascq Cedex - France
raphael.marvie@univ-lille1.fr

Abstract — Ambient computing is one of the more significant recent advances in computer-human interactions. With the ambient intelligence paradigm, computers become embedded in our natural surroundings. As they are context-sensitive and adaptable, they better provide smart services to humans. But ambient computing requires communication between several heterogeneous components that are not supposed to communicate each other. This paper describes how we use a workflow to facilitate the design of multi-channel interfaces for ambient computing. Our results show that different devices (such as Wiimote, multi-touch screen, telephone, etc.) can be managed in order to activate real things (such as lamp, fan, robot, webcam, etc.). A smart digital home case study illustrates a possible implementation of our approach and shows how it allows redesigning easily some parts of the ambient system just by modifying the workflow.

Keywords — Pervasive computing, ubiquitous computing ambient intelligence, multi-channel interaction, workflow, smart digital home.

I. INTRODUCTION

Due to the arising of pervasive and ubiquitous computing, the design of human computer interfaces has to take into account the context of interactions. The objective is to allow users to interact with a smart system in low constraints through the use of multiple modalities, channels, and devices. In the future, with the availability of new devices and smart objects, ambient computing will permit the definition of services able to seamlessly interact both with environment and users.

Our current work takes place in this context of ambient computing. In order to support on-the-fly interactions with the user, services have to adapt themselves to their mutating environment – resulting of the user mobility and the variability of his/her context. This requires (a) the availability of distributed devices such as PDA, laptops, smart phones, robots, probes, and (b) the easy discovery of these devices.

Currently, development tools able to easily generate and integrate ambient services are lacking. Each piece of software is developed on its own, and then integrated in the system. This introduces additional costs as well as misconfiguration risks. This paper presents some of our research results contributing to this particular problem. It is focused on the design of multi-channel interfaces relying on a workflow engine in order to ease the realization of ambient systems. The document is structured as follows: related work is presented in section two. Section three explains the background and motivation of this project. Section four gives an overview of our approach in order to tackle the emerging problems encountered. Section five describes a case study around the smart digital home thematic. Then, a conclusion gives our roadmap for future work.

II. RELATED WORK

Some computer frameworks (e.g. W3C MMI: Multimodal Interaction Activity or OpenInterface) and languages have been proposed especially to facilitate the development of multimodal interfaces.

In the W3C MultiModal Interaction framework [13], the interaction manager invokes specific application functions and access information in a dynamic processing module. The interaction manager presents the result to the user via one or more output components. Obviously, the interaction manager of this framework is very important because it coordinates data and manages execution flow among various input and output components. It also responds to inputs from the input components, updates the interaction state and the application context, and initiates output to one or more output components. Developers use several approaches to implement interaction managers, including: Traditional programming languages such as C or C++; Speech Application Language Tags (SALT) which extends HTML by adding a handful of HTML tags to support speech recognition, speech synthesis, audio file
replay, and audio capture; XHTML plus Voice (often referred as “X+V”), in which the VoiceXML 2.0 [10] voice dialog control language is partitioned into modules that are embedded into HTML; Formal specification techniques such as state transition diagrams and Harel state charts.

The OpenInterface project [5] is dedicated to multimodal interaction. In this project, everyday objects can take part in the interaction in ubiquitous computing (including an augmented table for instance) and the user can freely switch from one modality to another one according to her/his context: running in the street, at home, in front of a big screen in an airport, etc. This project aims at design and development of an open source framework for multimodal interaction: the OpenInterface framework. Those kinds of projects are mainly devoted to the study of multimodal interactions, allowing the usage of more than one device or modality at the same time in order to interact with a main system connected to Internet.

Our researches are more oriented toward the ambient computing that is not supposed to manage only devices and modalities, but also channels, in order to allow intelligent and context-aware communications.

III. BACKGROUND

Our work tackles the ability of ambient computing to permit context-aware interactions between humans and machines. To do so, we rely on the use of multimodal and multi-channel interfaces in various fields of application such as coaching [8], learning, health care diagnosis, or in-situ marketing. For Frohlich, a channel is defined as an interface that makes a transformation of energy [2]. From a user’s point of view, he distinguished voice and movement channels, and from the system’s point of view he mentioned audio, visual, and haptic channels. In the HCI domain, the notion of channel is not used very often and there are very few references to multi-channel research with some exceptions such as the work of [3]: “Often these modalities require specialized channels to allow access modalities such as cameras, microphones, and sensors. A multi-modal multi-channel system faces the challenge of accepting information from any input method and delivering information through the appropriate output methods.”.

Using a multi-channel approach allows users to interact with several channels, and to choose the most appropriate one each time, in order to exchange with an entity. Such channels could be, for instance, plain paper, e-mail, phone, web site. Using a multimodal approach allows users to employ several modalities in order to interact with the system. It can be sequential, like first on the phone then on the web, or synergistic [1], like on the phone while on the web. This approach implies some synchronization requirements both for the interfaces and bases of knowledge used during the interactions.

There are very few tools that support the design and implementation of interfaces having such characteristics [9]. One of our goals is to study and propose infrastructures easing interactions that are both multimodal and multi-channel in an ambient context. In our work, we use the Multi-DMC referential proposed in [7]. It can identify a system based on three criteria: Device (D), Modal (M) and Channel (C). It has two positions (Mono or Multi) for each of the three criteria targeted (DMC). This represents $2^3$ opportunities positioning. For a given system, one tries to indicate the position of each decisive factor. For example, the system represented on Figure 1 is a multi-device, multimodal, and multi-channel system.

![Figure 1: Example of a multi-device, multimodal, and multi-channel system](image)

In this paper, we are actually talking about ambient systems, which are more comfortable to be used, but also more difficult to conceive and develop than traditional systems. As robust communication between the various agents involved in the system is one of the key of an achievable solution, we need a bus to exchange messages smoothly and efficiently. The IVY bus [4] is an “interaction oriented middleware”, easy to use for prototyping. We used it as a support for multimodal and multi-channel pervasive interactions. The existing middleware was adapted to our specific requirements to meet the constraints of remote use. We designed and built a delegate to interact remotely by injecting information on the bus, usable with a phone (vocally) or a smartphone (Internet connected application), for example in a mobile situation.
A major question in pervasive and ubiquitous computing is how to integrate physical objects of the world (screen, chair, coffee machine...) into multimodal applications, thanks to technology such as RFID, NFC, Barcodes (1D or 2D as QR-Codes). This will help the users to manipulate freely virtual and real objects with commands like “identify this,” “make a copy of that object, here”, “move that webcam on the left,” etc. We are using the notion of workflow in order to indicate to the user the tasks available at each point of the whole activity flow.

Common Knowledge [6] is a cross-platform business rules engine and management system that supports the capture, representation, documentation, maintenance, testing, and deployment of an organization's business rules and application logic. Common Knowledge allows the business logic to be represented in a variety of interoperable, visual formats including: Rete rules, workflows, flowcharts, decision tables, decision trees, decision grids, state maps, and scripts. The engine allows running, testing, and simulating the system behaviors. It can be used through many languages (such as Java, Delphi, VisualBasic, C#, DotNET, etc.) and platforms (Windows, Linux, UNIX).

Standard and advanced operators can be used graphically to represent tasks, task choices, split or merge actions, timers, loops, etc. Figure 2 presents an example of workflow designed graphically with the Studio Common Knowledge tool. It allows following different path in order to complete a command such as “switch on fan”, “move camera down”, “switch off lamp”, etc.

The notion of persistence is very important in this context. Indeed, we consider that a main interaction could be the result of many sub interactions between the system and one or many users. It could also be the result of a sequence of sub interactions conducted via different kind of channels and modalities. The Common Knowledge software that we are using for our research supports this persistence feature.

IV. Our Approach

In the context of interface design based on the DMC referential, we believe, as we explained previously, that meaningful global action for the system may be the result of a series of sub-actions. These sub-actions can be performed by multiple users cooperating. Several types of devices can be used (PC, Smartphone, mobile phone, etc.). Several modalities of interaction, such as direct manipulation (keyboard/mouse), voice, gesture, brain waves, can be used both in input and output. Finally, multiple communication channels can be used such as the telephone channel or the Internet channel.

We currently limit the use of an alternate multimodality (not synergistic). The triggering of a sub-action is based on the FIFO (First In, First Out) principle. Subsequently, we can use weights to prefer one method over another.

In our approach, a workflow is used in order to describe objects and actions that can be applied on those objects using one or more devices. At each step of the global activity, the workflow is queried and the different task choices returned by the workflow are proposed to the user, by the means of a software bus. When the end of the workflow is reached, a message is sent to the software bus announcing the kind of action requested by the user. Then, specialized applications of this ambient system transcribe textual commands into physical actions. To do so, they must be connected to the software bus, and also be subscribers to particular regular expressions.
V. CASE STUDY

A. Smart digital home

A smart digital home refers to a living space with devices that are connected through wired or wireless networks. The connected devices may be sensors, actors, consumer electronics, appliances, mobile and PC devices that cooperate transparently for facilitating living and improving usability in the home. Since a variety of devices are present in a smart digital home, convergence, across all the screens of TVs, PCs, appliances and mobile devices, and management of multi-channel interactions is manifestly the key for the success of residential applications.

In our example, several objects are identified in order to be driven remotely: a lamp, a fan, a Rovio robot, and a webcam. The possible actions on those objects are the following: move (up, down, left, right, and home) and switch (on/off). As we can see on Figure 2, while the interaction takes place, one of the possible paths of the workflow is followed. Once the final state is reached, a command is sent to the bus.

B. Architecture

For this smart digital home case study, we are using the Ivy software bus [4]. Figure 3 represents a part of the agents connected to the bus designed in order to interact with the user. They have been developed in C# language and they all receive and/or send textual information to the Ivy bus.

TTS_PC, X10_PC, VoiceXML_Maker and Webcam_PC applications are only subscribers. It means that they need data to prompt information to the user (a speech synthesised sentence for example), to activate appliances (micro-wave oven, washing machine, etc.), or to generate some piece of VoiceXML [11] [12] code that will be dynamically generated and used at runtime.

Some applications, like QRCode_PC, are only sending information to the bus. Some others, like ASR_PC and Workflow_Engine are using the bus both to receive and send data. Indeed, the Automatic Speech Recognition application usable on a PC (ASR_PC) needs to receive the different labels corresponding of the speakable words, and oppositely, it send to the bus the result of the speech recognition engine.

Workflow_Engine application is in charged of the connection with the persistent workflow that we use for this project. It exploits a dedicated API to send the choices of the user to the object connection engine, and to receive the next elements to be presented to the user.

Figure 3: Some agents developed to manage exchanges on the IVY bus

C. Implementation

Our global project was conceived to manage various kinds of devices, sensors, effectors and technologies such as keyboard and mouse, voice over telephone or softphone, QR-Code, multi-touch screen, Wiimote, Mirror / Reflet NanoZtag RFID, motion webcam, X10 protocol, Rovio robot, etc.

Our proposition is based on the architecture illustrated on Figure 4. Three types of elements are present: (1) Interactive components that are detectors and/or effectors, (2) IVY bus for message exchange and (3) Workflow engine. This proposal aims to provide developer the ability to associate to her/his application a multimodal dimension concerning its interactive part. Currently, interactions supported are ruled by only one principle which is "sentences triggering actions". A sentence consists of a sequence of words that can be triggered by any type of modality (voice, QR-Code, keyboard/mouse, etc.). To facilitate the writing of such sentences for an application, we use the Task Choice concept in order to factorize words. For example, a sentence may begin by "move" and then be divided into 4 sub-sentences (one for each concerned device). This avoids writing four complete sentences.
An example of path may be the following one: the user activate the button "move" from the windows application (first sub-action), presents in front of a webcam a QR-Code identifying the robot (second sub-action) and then pronounces on his/her phone the word "left" (third sub-action). This path is completed and the action "move the robot on the left" is triggered.

Once a model is loaded into the workflow engine, it is executed and the engine starts with the first task choice. Each time the engine points to a new task choice, the list of possible choices is sent to the bus. This is done by a software agent attached to the workflow engine. Thus, interactive components can subscribe to this type of message, in order, for example, to present the list of choices to the user (as graphical buttons, voice prompt, etc.).

Two other software agents were needed and developed. The first one notifies the workflow engine that a sub-action was performed. This type of agent is attached to an interactive component and translates each relevant interaction into a sub-action that is sent to the bus. The second agent allows to be notified that an action is requested (e.g. switch on fan). Such agent aims to be associated to an interactive component that will translate actions into actual commands on the component, using X10 protocol, for instance.

The three software agents previously mentioned have two roles: to subscribe/transmit on the IVY bus and to establish a protocol for discussion between the workflow engine and interactive components. This protocol is based on actions, sub-actions and possible actions. Note that in the model associated to smart digital home, we defined paths so user must first specify the command, then identify the device and finally give a possible parameter for command.

The three software agents used the workflow presented in Figure 2 to describe the objects and actions that can be applied on those objects using one or more devices. Moreover, dynamic voice grammars (or entire VoiceXML files) can be generated this way, as we can see on Figure 5.

If the designer decides to add a possible new direction, he/she can do it graphically, on the workflow, by adding an arc (called “home” for example), near the up/down/left/right already available. With no addition of code, a new possible path is potentially usable in the workflow. Consequently, one can then pronounce a sentence like “move camera home”, in order to physically make the webcam move.
the same time but with a unique model of existing actions. Thus, it will be interesting to work on the possibility to detect codes generated thanks to model (and meta-model) transformations. This will allow the possibility to detect new objects, persons or possible behaviors on the fly and to respond as soon as possible with relevant feature of the ambient system. Thus, it will be interesting to work on the possibility to manage different natural languages at the same time but with a unique model of existing actions.

We will also work on the semantic aspect of the workflow. This is an important point that will help the users that are not using the commands in the right order for example. Indeed, a smart system must be able to understand that “move up robot” is the same command as “move robot up”. We are also planning to offer the possibility to switch from a software bus to another and to manage virtual representation of tangible things (fridge, oven, etc.) in order to allow realistic simulations before real implementation.

ACKNOWLEDGEMENT

The authors would like to thank ObjectConnections, Jaxo Sytem and bewecam for special tools provided.

REFERENCES