Mixed-Initiative, Trans-Modal Interface Migration

Renata Bandelloni, Silvia Berti, Fabio Paternò.

I.S.T.I-C.N.R. via G.Moruzzi, 1,
56124 Pisa, Italy
{Renata.Bandelloni, Silvia.Berti, Fabio.Paterno}@isti.cnr.it

Abstract. This paper presents our solution to supporting runtime migration of Web application interfaces among devices offering different interaction modalities, in particular graphic to vocal platform migration and vice versa. Migrating between platforms implies keeping track of the user interactions in order to retrieve the runtime state of the interface and maintaining interaction continuity on the target device. The system can serve user-issued migration requests containing the identifier of the selected target device, and can also automatically start the migration procedure when environment conditions require it. In automatic migration the target platform has to be automatically selected as well. To this aim, we consider devices belonging to a restricted environment and have defined selection rules in order to identify the most suitable available target for the ongoing migration.

1 Introduction

The wide availability of new mobile devices supporting Internet access, offering various interaction capabilities, raises the need for applications able to support different interaction modalities. On any given day, users are surrounded by many different interactive platforms at work, at home, and even while walking along the road. User mobility accompanied by various devices raises the need for some sort of application interface mobility that allows users to change the device they are interacting with while moving from one environment to another, or just because the resources (such as the battery) of the current mobile device have been depleted. Such scenarios raise the need for multi-platform migration services that are able to follow users through the changing contexts by transferring amongst different devices at run time.

We analyzed the potentialities of our model-based approach to perform an adaptive interface migration [1], addressing devices with different features, able to support graphic navigation of Web sites. In this work we address two novel issues for this approach:

- introduction of migration with modality change, thus allowing users to migrate from a graphical interface to a vocal interface or vice versa;
- support for migration that can be activated not only on user request but also by the automatic detection that a system is no longer able to support the user (for example, because the battery has expired or the system is no longer connected).

The solution to these issues is important because allows users interacting with a typical graphical Web browser, to continue the interaction through a different device and a different interaction modality. This is the case of a user navigating the Web through a PDA or Desktop PC and migrating the application to a mobile phone supporting only vocal interaction. Apart from migration on user demand, we also introduce the novelty of automatic migration, which implies automatic recognition of nearby devices associated to the same user as well as the ambient conditions leading to migration.

Graphic and vocal interactions rely on different interaction techniques because of the differences between the associated media. In graphic browsers, many tasks can be supported concurrently, all at once in a page, and the user can freely decide which one to perform. Vocal navigation imposes a serialisation of dialogues. At any time only one interaction is available, even if users can choose to move at different points of the dialogue structure. Such differences imply a different way to structure the concrete interface and choose the interface elements.

Our migration service applies to Web applications whose interface has been developed through the model-based approach supported by the TERESA tool [5]. This provides semantic information associated with the user interface implementation that can be exploited at run-time to support migration in such a way as to maintain interaction continuity and consider usability design criteria.

In section 2 we discuss related work. Next, we introduce a couple of scenarios highlighting the issues that we aim to address. Then, we discuss the solution developed to obtain migratory interfaces through underlying transformations and processing. This is followed by the description of the architecture of the migration service highlighting how it can support both user-activated and system-activated migration. We also provide more detail showing how the trans-modality migration is achieved along with an example of application. Some concluding remarks and indications for future work conclude the paper.

2 Related Work

Run-time adaptation of user interfaces to different device capabilities raises many issues. A framework describing such issues is provided in [2]. PUC [6] is an environment that supports the downloading of logical descriptions of appliances and the automatic generation of the corresponding user interfaces. The logical description is performed through templates associated with design conventions, which are typical design solutions for domain-specific applications. The application area of this approach is limited to the home domain where devices require similar interfaces. Aura [3] is a project whose goal is to provide an infrastructure that configures itself automatically for the mobile user. When a user moves to a different platform, Aura attempts to reconfigure the computing infrastructure so that the user can continue working on tasks started elsewhere. In this approach, suppliers provide the abstract services, which are implemented by just wrapping existing applications and services to conform to Aura APIs. For instance, Emacs, Word and NotePad can each be

wrapped to become a supplier of text editing services. So, the different context is supported through a different application for the same goal (for example, text editing can be supported through MS Word or Emacs depending on the resources of the device at hand). Our work follows a different approach where the application is still the same but the interactive part is adapted to the new device.
A taxonomy of task for voice interfaces to Web pages [7] has been proposed with the aim to obtain a voice navigation that is not a mere substitution of graphic navigation, but is tailored on specific voice interaction features. The paper focalizes on VoiceXML interface design, in our work we take into consideration both vocal and graphic navigation and make a comparison between them in order to allow a user to change interaction modality while changing device.
An example of transformation of a Web site developed in HTML into a VoiceXML application is presented in [4]. In this paper the original Web site is analyzed and redesigned in a dialog model style, by means of a finite state diagram, then the model is implemented in VoiceXML. The model of the vocal version of the original site is manually built; authors are working on the automatic remodelling of the Web site, basing on a syntactic and semantic HTML files analysis. In our approach we consider interfaces generated from task models. This semantic information is exploited also in the migration service in order to identify what part of the target interface to activate and to associate it with the state of the user interactions performed so far.
Perez-Quinones et al. [7] describe a multimodal interface architecture that allows for combining speech, pen and touch-tone digit interaction in noisy mobile environments. The proposed system allows users to interact with an application using more than one modality at once. The system was evaluated through an example application. One result is the confirmation that some kinds of tasks are more appropriate for a specific input modality. In that work, the user can access different interaction modalities at the same time, over a single device.

3 Scenarios

In this section we present two scenarios to underline the features of the multi-modal migration service. The first one concerns a restaurant booking application and is an example of graphic to vocal migration.
Friday Morning, Louis is at home and wants to organize a dinner with his friends for the evening. He turns on his personal computer and opens the Official Web site of the town. He accesses the restaurant main page, from which he starts selecting restaurants one by one, in order to check the menu of the day. While Louis is selecting the Mermaid restaurant main page, he realizes that it is getting late and has to leave and go to work, hence requires the migration to his mobile phone. Louis can now turn off the computer keep interacting with the application in vocal mode. The vocal interface remembers the selected restaurant to Louis and tells him the different options he can go through. Louis asks to hear the menu of the day, then he asks to go back to the main restaurant options and asks for booking a table. The system asks Louis to say his name, selects the preferred menu, and specifies the date and time for booking. Finally, the system repeats all the information inserted to Louis, asking confirmation.

Louis confirms the booking and keeps walking to his office, enjoying the thought of the dinner with his dearest friends.
The second scenario concerns a typical agenda application and is an example of vocal to graphic migration. Monday morning, George is driving in his car when an accident blocks the road. George will be late for work, so he decides to access the Agenda Application through the car voice system to check his schedule for the day. The voice synthesizer welcomes George to the Agenda Application and tells him all the available operations. George says "Today's *schedule*" to check the appointments fixed for the day and under a further system request, he specifies that he wants to hear the appointments scheduled for the morning. The synthesizer says that he has two appointments scheduled in the morning and the first one is a 10:00 meeting with the project coordinator. George asks for more details, meanwhile he arrives at work. As soon as he turns off the car, the application migrates automatically from the voice car system to the PDA that George has in his pocket. George starts running to his office to collect important documents for the meeting and use his PDA to check in which room the meeting is to be held. In the above scenario, the vocal interaction is supported by a voice car system. In our analysis, we take into consideration a vocal interface accessed through a mobile phone. Diverse voice car kits connect to the car owner mobile phone, as soon as the vehicle is turned on allowing automatic call answering. With such kind of equipment, migration can take place from the PDA to the mobile phone and vice versa, giving the user the feeling that only the phone and the car are involved. In particular, the user will not hear the phone ring, announcing graphic to vocal migration, because of the automatic call answer feature, and will be able to continue interacting, without any supplementary action to receive the call.

4 Migration Service Approach

The interface migration is obtained through different interface versions (one for each platform). When the interface migrates then the migration service is able to activate the version for the target device at the point where the user left the source device and maintain the state resulting from the previous interactions in the new device.
Our migration service applies to Web applications whose interfaces have been developed through the model-based approach integrated in the tool TERESA. The interface generation through the TERESA approach, starts with the development of the nomadic task model that describes the application interface in terms of user activities. Platform specific task models are obtained analysing the nomadic one, extracting the tasks supported by the specific platform. Each refined task model is used to generate the Abstract User Interface (AUI), where the interface is described in terms of presentations. Each presentation contains: a set of Interactors, giving an abstract description of the objects that will be used to implement corresponding tasks, and composition operators, providing declarative indications on how to compose interactors (grouping, hierarchy, relation, …). Each presentation is associated with the set of tasks that it supports. The last step is the generation of the final user interface according to design criteria that take into account the platform selected. It can be generated in XHTML, XHTML Mobile Profile, Java, or VoiceXML. In this

paper we are considering XHTML and VoiceXML languages, in order to address the trans-modality migration.
The logical descriptions obtained through the interface generation process are used by the Migration Server in order to compare source and target platform version of the interface, to identify the presentation for the target platform and keeping user interaction continuity, when activating the interface on the target device. When a migration is required by the user, or automatically triggered, the server retrieves the last URL loaded on the source device, hence extracts the presentation describing the migrating page, from the AUI of the corresponding interface. At this point, the server accesses the AUI describing the interface for the target platform type.
The server uses the AUIs, to search for the target presentation that is the most similar to the source one. Similarity is calculated in terms of supported tasks: the higher number of tasks the source and target presentation share, the more similar the presentations are. This similarity criterion can lead to ambiguity in case more than one target presentations share the same number of tasks with the source one, having the same similarity degree. The conflict is solved identifying the target presentation supporting the task associated with the interaction object last modified by the user on the source device, since the user is most likely to continue interaction from that point.
Once the target presentation has been identified, the target page is immediately identified, since a one to one mapping exists between presentations and pages. Next step is to calculate the state of the objects contained in the target page, in order to keep interaction continuity. In this phase, we consider objects implementing corresponding tasks in the source and target page. In different versions of the interface obtained for different platform types, the same task can be implemented by means of different interaction objects. In particular, while comparing graphic and vocal platforms, we have VoiceXML objects in one version and HTML objects in the other one. For example, the graphical interface task that performs a selection action can be implemented by radio button while, in the vocal interface, this can be obtained through DTMF (Dual Tone Multi Frequency) menu voice, and the selection can be performed through keypads. Another interesting example is given when in the logical description of the presentations there are two or more *control* interactors that enable the access to other application pages: in the graphical interface they can be implemented by buttons or links while in the voice interface they can be combined in a menu voice.
The description of the runtime state of a graphical object has to be translated into a description of the runtime state of the corresponding vocal object and vice versa. For example, if users select an option in the graphical interface they can listen its result through the feedback of the choice in voice menu and vice versa, if users press a particular number of keypad they can see the radio button corresponding the same option selected in the graphical interface.
In graphic-to-graphic migration, the runtime state is sent to the target device in a form adapted to its resources. Applying the state to the page is the task of the migration client running on the device. The same technique can be used in performing vocal to graphic migration, but not for graphic to vocal because a typical vocal device has very limited capabilities. Hence, when the target has no computational capability other than handling phone calls, all the work must be performed on the server side.

Once the vocal target page has been identified, a new temporary page is created. Such a page is a copy of the target one, plus the suitably adapted runtime state. In this way, the original page remains available on the server for access by other users and the modified copy is removed when the target platform ends the call.

5 The Migration Service Architecture

The scenarios introduced in Section 3 can be supported by a migration service that allows trans-modal migration to be activated either by user request or automatically when the environment conditions require it. We define the first type as *on demand migration* and the second *automatic migration*.
In *on demand migration*, the user explicitly asks for the application to migrate, specifying the target device. In *automatic migration*, the system must check the environment conditions like mobile device battery energy level and device proximity, in order to decide if migration is needed and to select the target device when more than one fit the predefined requisites.
In [1] we proposed a solution to support on *demand migration* involving devices supporting graphic Web browsers. In this work, we improve the migration service adding a modal migration from graphic to vocal browsing and vice versa. We also add the *automatic migration* service by changing the runtime context manager module and introducing client devices classification and localisation mechanism. In our previous work, the runtime state of the migrating page was collected on the client side and sent to the server only when the user decided to migrate the application. In the runtime state there is the result of the user interactions (selected elements, values entered, …). In the new solution, we keep updating the server-side data structure, describing the runtime state of the application on the client. In this way, the server does not have to query the client for its runtime state, in particular, when migration is triggered because a previously available device becomes unavailable. Otherwise, it would not be possible to retrieve the runtime context of the application running on it. The new solution for the state management is discussed in 5.1.
The other important new feature is the introduction of the client device classification and localisation. Devices are classified in terms of features that guide the selection of the target device for *automatic migration*. In particular, they are considered as part of an environment as described in 5.2 Activation of the application on the target device has also been improved, in order to enable modal migration, which was not previously supported. The new solution is discussed in 5.3.

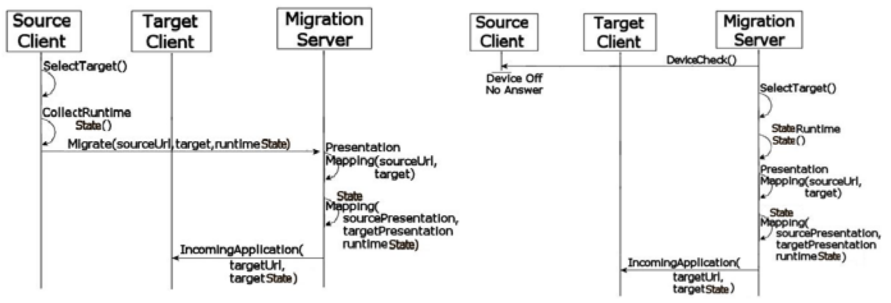

Source Client
Target Client
Migration Server
SelectTarget()
CollectRuntime State()
Migrate(sourceUrl, target, runtimeState)
Presentation Mapping(sourceUrl, target)
State Mapping(sourcePresentation, targetPresentation runtimeState)
IncomingApplication(targetUrl, targetState)
Source Client
Target Client
Migration Server
DeviceCheck()
Device Off No Answer
SelectTarget()
StateRuntime State()
Presentation Mapping(sourceUrl, target)
State Mapping(sourcePresentation, targetPresentation runtimeState)
IncomingApplication(targetUrl, targetState)

Fig. 1. Left: old migration solution, right: new migration solution

The service relies on a server machine that stores the migratable applications, as well as their logical descriptions and the mechanisms to perform device migration. The server is initialised by building correspondences between tasks and the user interface logical elements, and between the logical description of interface elements and the objects used for their implementation. Such operations are performed for all migratable interfaces and for each one of their device specific versions (see section 4). The time required for the initialisation phase increases with the number of supported applications and their complexity. However, this operation is performed only once at start-up and considerably increases the speed of runtime migration. Users who want to access the service have to load the migration client from the server onto their device. This operation allows the server to identify the devices available for migration and also enables the user device to send migration requests and work as a target for an incoming migrating interface.
Summarising, the main aspects of this improved version of the migration service are: state management, device management and target interface activation.

5.1 User Interface State Management

When a new device enters the migration service, a state collection module is activated on the client side and a corresponding one is created on the server side.
Generally speaking, a server can not access directly information inserted by users on client devices, until they are submitted. In performing migration, we need what has been inserted in the page shown to the user, for this purpose clients can provide useful support in the runtime state collection.
Any time the user interacts with an element of an interface, the client module catches the generated event and immediately sends the new state of the object to the server. The captured events relate to actions, such as objects selection and text insertion. The server keeps a description of the runtime state of the pages loaded on the client, and updates it at each new message received by the client.
When a migration request has to be served, the server analyzes the description of the runtime state, associated to the client from which the interface has to migrate to retrieve the URL of the last page visited by the user and the runtime state of each object of the interface as it was when migration was requested (or triggered).

The last visited page URL is used in the process of target page retrieval and the runtime state of the source page is elaborated to be adapted to the retrieved target page (see section 4).

5.2 Device Management

When asking for *on demand migration*, the user specifies which device has to be the target. In this case, the only information concerning the target device that is necessary, is a description of its type and its supported features. In *automatic migration*, the target is selected by the server among the devices registered to the service according to their features, settings and location.

- *Features*. When a device accesses the migration service, the server recognises its platform type like mobile phone, PDA, desktop, vocal and features such as the screen size, browser supported, etc. In particular, client devices are also recognised as mobile or stationary.
- *Settings.* When the users start the client migration module, they have to specify if the device has to be used as a personal or shared device. A device is shared when more than one user can access it, while it is personal when only the owner can use it. The availability to accept incoming migrating interface has to be declared. Users can also register to the service, specifying more devices that must be considered as potential target for migration. Such devices are those that cannot load a migration client, but can be activated directly by the server, in particular they can be fixed or mobile phones.
- *Location.* The server must keep track of the position of each active client. Devices are considered near, when they are inside the same environment. An environment can be a room, when we consider a building or a car when we consider the user moving outside. The current environment is mainly detected through the use of WLANs and infrared beacons. Stationary devices such as desktop PCs, are statically considered into a specific environment that can not change until the device is turned on, while mobile devices are subject to change position frequently and their position is kept updated.

When selecting a target device for automatically triggered migration, the server considers all the devices being in the same environment in which the source device is that are available to receive incoming applications. In order to select the final target device, among a set of available candidates the migration server analyses the interaction capabilities and energy supply matters of the available devices. For example, we can think of a user interacting with a vocal application through his mobile phone, while reaching his desktop PC and having his PDA turned on in a pocket. The mobile phone is losing battery power and turns off, the application must migrate, and both the PDA and the desktop are close enough to the user. In this case, the desktop is selected as the target device, because a PDA could also be affected by energy supply problems and offers less interaction facilities than the desktop.
Checking the environment, the migration assigns priority to the devices registered as personal and that can be automatically activated. In case the user has for example a

fixed or mobile phone in his device list, the server can make the phone ring migrating the application to one of them as soon as the user answers the call.

5.3 Target Interface Activation

There are two different modalities used to activate the interface application on the target device. In case of vocal to graphic migration, the target is required to run the client migration module. Once the server has calculated the URL of the page to be loaded on the target and adapted the corresponding runtime context, all information is coded in a formatted string and sent to the client module running on the target. The client module extracts the URL from the string, loads it into a Web browser window and also extracts the runtime state and applies it to the new page.
In case of graphic to vocal migration, if the host platform corresponds to a fixed or mobile phone then the server is instructed to send a phone call to the appropriate target, indicating which presentation has to be activated on user phone answer and how the vocal interpreter has to run the target presentation applying the runtime context obtained by the migration process.
Migrating from a modality to another one goes far beyond a simple one to one mapping among the pages of the two different versions.
The graphical interfaces do not translate well into speech interfaces for a number of reasons. For instance, graphical interfaces do not always reflect the vocabulary that people use when talking to one another in the application domain. Another important consideration concerns the information organization. In fact, presentations that work well in the graphical interface can fail in speech implementations. Reading exactly what is displayed on the screen is rarely effective. Likewise, users find it awkward to say exactly what is printed on the display. Therefore, it is necessary to analyse the logical description of the application to obtain a graphic to vocal mapping and vice versa, based on the supported task sets.

6 The Multimodal Restaurant Booking Application

In this section we introduce the Multimodal Restaurant Booking Application, a sample application built on the basis of one scenario described in Section 3. In the application the user can choose a restaurant in a specific area of the city. After selecting the Mermaid restaurant, the user fills in the form for booking a table. Let us imagine that he has filled in the first three fields and has selected menu type and then realises that it is getting late, so he decides to continue his booking by phone with the voice system in the car.
The first step is performed by the migration service in order to identify the voice presentation most similar to the source graphical presentation. Then, the migration server accesses the Abstract User Interface of the graphical interface and retrieves the presentation corresponding to the migrating page. At this point, the set of tasks performed by the presentation, and therefore supported by the migrating page, is identified and used by the mapping algorithm in order to find the right target

presentation. In the graphical application the set of tasks are composed of: *provide name*, *provide e-mail*, *provide date of reservation*, *provide time of reservation*, *provide number of people*, *select preference seating*, *select menu type*, *provide special request or comments*, *send reservation*, *and cancel reservation*.
During the mapping the migration server compares the task set of the source presentation (graphic) with the task set of the target (vocal) and identifies the most similar abstract presentation of the vocal abstract interface. During this step it may happen that some tasks supported by the source platform cannot be supported or can be performed through different interaction techniques. For example, the sample application does not support the task "*Provide special request or comments*" in the voice platform, because it would encumber the vocal interaction as it is not an essential task for booking.

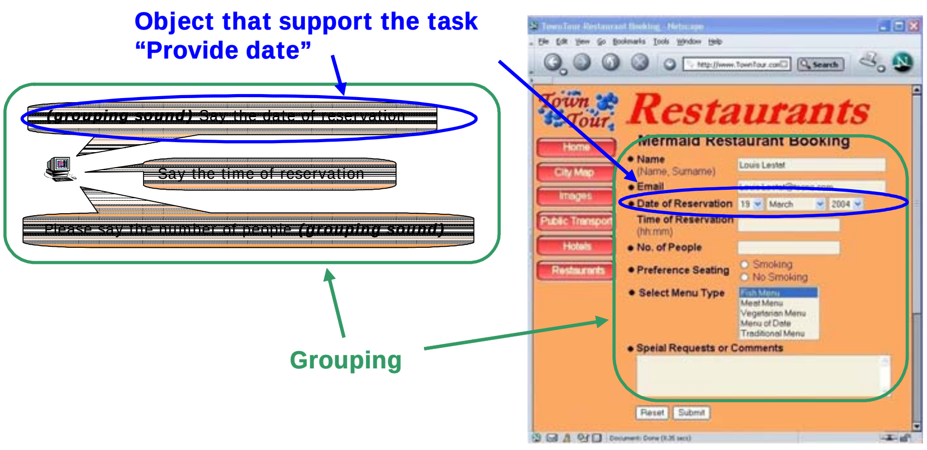
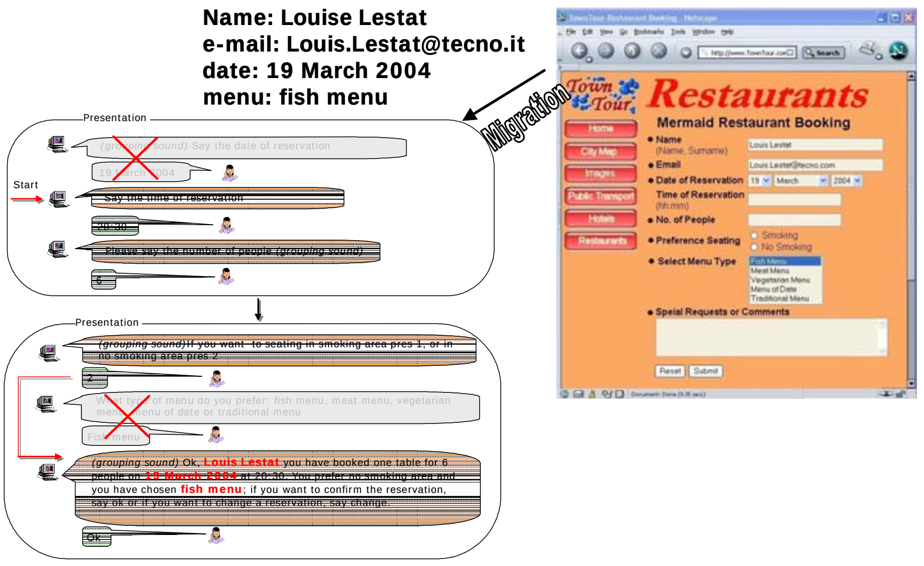


Fig. 2. Presentations of some booking information in different modalities.

Another example is the different method used for supporting the task "*Provide Date of reservation*". In the desktop interface it is implemented by three pull-down menus (day, month and year) while in the vocal system it is accomplished by a vocal input request to the user for the date of reservation without indicating any potential choice (see Figure 2).
In the example, the migration server identifies three vocal abstract presentations containing the same number of tasks of the source presentation. One task is not supported in the vocal application (*Provide special request or comments)*. The first presentation requests the user's name and e-mail, the second presentation requests the reservation date, time and the number of people, and the third presentation requests seating preferences, the type of menu and confirms or deletes the reservation.
It is also interesting to notice the different techniques adopted to combine interactors. For example, in the graphical interface the grouping operator is obtained through an unordered list, whereas the vocal interface uses a sound to delimit the grouped elements.

The second step allows the migration server to select the presentation that contains the object implementing the last task performed by the user on the target platform. During this phase, it is important to consider that the vocal channel serialises interactions, while they can be performed concurrently on a visual channel. Accordingly, once the presentation has been identified, the migration server checks if all the previous tasks have really been executed. If a negative response results for any tasks, they are performed first and then the dialog carries on from the task last executed in the source device.


Name: Louise Lestat
e-mail: Louis.Lestat@tecno.it
date: 19 March 2004
menu: fish menu

(grouping sound) Say the date of reservation
19 March 2004
Say the time of reservation
20:30
Please say the number of people (grouping sound)
6

(grouping sound)If you want to seating in smoking area pres 1, or in no smoking area pres 2
2
What type of menu do you prefer: fish menu, meat menu, vegetarian menu, menu of date or traditional menu
Fish menu
(grouping sound) Ok, Louis Lestat you have booked one table for 6 people on 19 March 2004 at 20:30. You prefer no smoking area and you have chosen fish menu; if you want to confirm the reservation, say ok or if you want to change a reservation, say change.
Ok

Restaurants
Mermaid Restaurant Booking
- Name (Name, Surname): Louis Lestat
- Email: Louis.Lestat@tecno.com
- Date of Reservation: 19 March 2004
- Time of Reservation (hh:mm):
- No. of People:
- Preference Seating: Smoking / No Smoking
- Select Menu Type: [✓ Fish Menu] Meat Menu Vegetarian Menu Menu of Date Traditional Menu
- Speial Requests or Comments:

Fig. 3. Example of Migration scheme

In the presented example, composed of three vocal presentations, the last executed task is *select menu type* and is included in the third presentation; the first presentation, which asks for the user name and the e-mail, has been performed, while the second and third presentations were not completed. In this situation, the dialog starts with the first task of the second presentation (*provide time reservation*) and skips the tasks that have already been executed through the graphical interface (see Figure 3).
With this solution, the data previously inserted in the form by the user are not lost, and can be listened to in a feedback message of the last presentation.

7 Conclusions and Future Work

We have presented a new solution to obtaining migrating interfaces that can be either initiated by the user or automatically triggered by the system when environment conditions require. Moreover, we have also added the possibility of interaction

modality changes during migration. In particular, we have addressed graphic to vocal migration and vice versa.
At this stage, our prototype of migration service supports migration of interfaces implemented by XHTML, XHTML Mobile Profile and VoiceXML developed with TERESA. We will soon support also multimodal interfaces implemented in languages such as X+V.
Further studies will address the improvement of the migration service in order to support Web interfaces developed using other tools as well. This further issue will require a different kind of interface analysis: we plan to use tools for reconstructing a logical description of the pages at runtime. The extension to such interfaces is a main goal for our future work.
Another topic for future work is the support of multimodal distributed migration, in which a user interface migrates in such a way to carry on interaction through multiple devices.

Acknowledgments

We gratefully acknowledge support from the IST EU R&D CAMELEON project (http://giove.cnuce.cnr.it/ cameleon.html) and would like to thank our colleagues in the project for useful discussions.

References

1. Bandelloni R., Paternò F., Flexible Interface Migration, Proceedings ACM IUI 2004, pp.148-157, ACM Press, Funchal, 2004.
2. Coutaz J., Balme L., Barralon N., Calvary G., Demeure A., Lachenal C., Rey G., Bandelloni R., Paternò F., the CAMELEON Run Time Infrastructure for User Interface Adaptation, October 2003, CAMELEON Deliverable D2.2.
3. de Sousa, J., Garlan, D. Aura : an Architectural Framework for User Mobility in Ubiquitous Computing Environments. IEEE-IFIP Conf. 140 on Software Architecture, Montreal, 2002.
4. Ferreras, C, G., Mancebo, E, D., Payo, V, C.. From HTML to VoiceXML: A First Approach. In Proceedings of TSD 2002, LNAI 2448, pp 441-444, 2002.
5. Mori, G., Paternò, F., Santoro, C. “Tool Support for Designing Nomadic Applications”, Proceedings ACM IUI’03, Miami, pp.141-148, ACM Press.
6. Nichols, J. Myers B. A., Higgins M., Hughes J., Harris T. K., Rosenfeld R., Pignol M.. “Generating remote control interfaces for complex appliances”. Proceedings ACM UIST’02, pp.161-170.
7. Pérez-Quiñoes, M, A., Capra, R, G., Shao, Z.. The Ears Have It: A Task by Information Structure Taxonomy for Voice Access to Web Pages. In Proceedings of INTERACT 2003.