Speech Centric Multimodal Interfaces for Mobile Communication Systems

KNUT KVALE, NARADA DILP WARAKAGODA AND JAN EIKESET KNUDSEN

Natural conversational man-machine interfaces should support multimodal interaction. Generally, a multimodal system combines natural input modes such as speech, touch, manual gestures, gaze, head and body movements, and searches for the meaning of these combined inputs. The response can be presented by a multimedia system.

The main focus of this paper is the technical aspects of implementing multimodal interfaces for mobile terminals. With the limited size and processing power of these terminals, we have restricted the functionality to *speech centric* multimodal interfaces with *two* input modes: speech (audio) and touch, and two output modes: audio and vision. That is, the input combines automatic speech recognition and a pen to click areas on the touch-screen, or pushing buttons on the small terminal. The output is either speech (synthetic or pre-recorded) or text and graphics.

Knut Kvale (39) is Research Scientist at Telenor R&D and Professor II at the Stavanger University College. He graduated from the Norwegian Institute of Technology as Siv.Ing. (MSc) in 1987 and Dr.Ing. (PhD) in 1993. Knut Kvale joined the Speech Technology Group of Telenor R&D in 1994. From September 1998 he has been the Research Manager of this group.

knut.kvale@telenor.com

Narada Warakagoda (38) is a research scientist at Telenor R&D. He graduated from the Norwegian University of Science and Technology as Siv.Ing. (MSc) in 1994 and Dr.Ing. (PhD) in 2001. During his work towards the PhD degree, he developed automatic speech recognition algorithms based on the theory of nonlinear dynamical systems. After joining Telenor R&D, his research began to orientate more towards speech technology based telecom applications. Recently he has been involved in several projects on speech centric multimodal systems. His current research interests are multimodal interfaces which can exhibit human-like intelligent and emotional behaviour.

narada-dilp.warakagoda @telenor.com

1 Introduction

Since small mobile terminals have limited keypads and small screens we have to use alternative communication channels when interacting with these terminals. Speech is a natural and convenient way to express complex questions to a service. Speech is also the best option when eyes and hands are busy, e.g. when driving a car. Thus, a well-designed user interface has to support automatic speech recognition (ASR). However, when the speech recogniser fails or when it is not appropriate to speak, the user may want to select icons or scroll menus by pointing at the screen or even write the commands. Input channels such as touch sensitive screens, keyboards or keypads are therefore also required. The terminals' output channels should at least support visual (text and graphics) presentation and audio. Listening may be a slow way of obtaining information, and it is difficult to remember anything more than a few names or numbers. Screens are needed to display graphics and information that is tedious to listen to.

For some tasks it is natural and convenient to ask questions orally while pointing at related objects on the screen, e.g. when asking for the distance between two points on a map. In this case we want to combine different "senses" seamlessly in the user-interface. Multimodal interfaces combine the different input signals, extract the combined meaning from them, find requested information and present the response in the most appropriate format.

Hence, a multimodal human-computer interface (HCI) gives us the opportunity to choose the *most natural interaction* pattern. If the preferred mode fails in a certain context or task, we may switch to a more appropriate mode or we can combine modalities.

In the last two decades there has been a huge research activity within multimodal user interfaces. In 1980 Bolt [1] presented the "Put That There" concept demonstrator, which processed speech in parallel with manual pointing during object manipulation. Since then major advances have been made in speech recognition algorithms and natural language processing, in handwriting and gesture recognition, as well as in speed, processing power and memory capacity of the computers. Today's multimodal systems are capable of recognizing and combining a wide variety of signals such as speech, touch, manual gestures, gaze, head and body movements. The response can be presented by a multimedia system. These advanced systems need various sensors and cameras and a lot of processing power and memory. They are therefore best suited for kiosk applications. An overview of various multimodal systems can be found in [2], [3], [4], [5].

The work presented in this paper is based on the research work at the Speech Technology group at Telenor R&D aiming at implementing a test platform for speech-centric multimodal interaction with small mobile terminals. For applications on mobile terminals, with limited size and processing power, we have restricted the functionality to *speech centric* multimodal interfaces with two input modes: speech (audio) and touch, and *two* output modes: audio and vision. Our main focus is on exploiting multimodal speech dialogues to improve dialogue success rate, dialogue completion time and user friendliness in applications basically for 3rd generation mobile communication systems (3G/UMTS).

Jan Eikeset Knudsen (41) is Research Scientist at Telenor R&D. He graduated from the Norwegian Institute of Technology as Siv.Ing. (MSc) in 1987. Jan Eikeset Knudsen joined the Speech Technology Group at Telenor R&D in 1988.

jan-eikeset.knudsen@telenor.com

2 What is Mulitmodality?

2.1 Modality, Multimodal and Multimedia

The term *modality* refers to a form of sensory perception: hearing, vision, touch, taste and smell. For our research on human-machine interaction, we define modality as a communication channel between the user and the device.

The modes above can be combined in a multimodal interface, containing audio (e.g. in the form of speech), vision (in the form of text and graphics, or moving video), and touch. We do not consider services using one particular input mode, e.g. speech, and another output mode, e.g. text/graphics as multimodal services.

We distinguish between multi*mode* and multi*media*; that is, *media* is the representation format for the information or content in a certain *mode*. For example, speech and music are two media formats in the auditory mode. Text, graphics and video are examples of media types in the visual mode.

2.2 Combining Multiple Modalities

Multiple input and output modalities can be combined in several different ways. We apply the scheme proposed by the World Wide Web Consortium (W3C) [8], which distinguishes between three different ways of combining multimodal inputs and outputs: *Sequential*, *uncoordinated simultaneous* and *coordinated simultaneous* multimodal input/output, as discussed below.

2.2.1 Multiple Input Modalities

Sequential Multimodal Input

This is the simplest type, where inputs from different modalities are interpreted separately. For each dialogue state, there is only one input mode available, but in the whole interaction more than one input mode may be used.

Sequential multimodal input is often used in system-driven applications.

Uncoordinated Simultaneous Multimodal Input

In this situation several parallel input modes are active at the same time. This means that the users can choose the input mode they prefer at each dialogue stage. However, only one of the input channels is interpreted (e.g. the first input).

Coordinated Simultaneous Multimodal Input

Here, more than one input mode is available, and *all* inputs from the multiple modalities within a given time window are interpreted. The interpretation depends on the fusion of the partial information coming from the channels.

Coordinated simultaneous modalities may be the most natural way of interacting with computers, but it is by far the most complicated scenario to implement.

2.2.2 Multiple Output Modalities

W3C [8] distinguishes between three different implementation schemes for multimodal *outputs* in a similar manner. On the output side the sequential and non-coordinated simultaneous use of modes is less apparent, because the graphical display is static: it remains visible during times when speech is played (and the graphical image cannot be changed). In coordinated simultaneous multimodal output, information may be conferred by means of a spoken message that coincides with changes in the graphical display and perhaps also with gestures of an on-screen presentation agent.

2.3 Speech Centric Multimodality

Some multimodal systems apply advanced input 'devices', such as gaze tracking and facial expression recognition, and outputs e.g. facial animation in the form of human-like presentation agents on the screen as in "Quickset" [2, 3, 4, 5], "SmartKom" [9, 10], and "Adapt" [11, 12].

However, for telecommunication services on small mobile terminals, we have constrained our multimodal research issues to *speech centric multimodality* with *two* input modes: speech (audio) and touch, and two output modes: audio and vision. That is, the input combines automatic speech recognition and a pen for clicking areas on the touch-screen, or pushing buttons on the small terminal, also called "tap and talk" functionality. The output is either speech (synthetic or pre-recorded) or text and graphics.

Speech centric multimodality utilises the fact that the pen/screen and speech are *complementary*: The advantage of pen is typically the weakness of speech and vice versa. With speech it is natural to ask one question containing several key words, but it may be tedious to listen to all information read aloud – speech is inherently sequential. With pen only, it may be hard to enter data, but it is easy to get a quick overview of the information on the screen, as summarised in Table 1.

Hence, we believe that combining the pen and speech input in a speech centric multimodal interface will lead to a more efficient dialogue through better error avoidance, error correction and error recovery, as exemplified below:

- Users will select the input mode they judge to be less prone to error;

Only pen input, screen output	Pure speech input/output
Hands and eyes busy – difficult to perform other tasks	Hands and eyes free to perform other tasks
Simple actions	Complex actions
Visual feedback	Oral feedback
No reference ambiguity	Reference ambiguity
Refer to items on screen only	Natural to refer to absent items also
No problem with background noise	Recognition rate degrades in noisy environments

Table 1 Comparison between the two complementary user interfaces: Pen-only input and screen (visual) output versus a pure speech-based input/output interface

- Users tend to use simpler language which reduces the complexity of the Spoken Language Understanding (SLU) unit;
- Users tend to switch modes after system errors and thus facilitating error recovery;
- Improved error recovery by combining N-best lists and confidence scores and selection from multiple alternatives via display;
- Improved speech recognition performance by context sensitive ASR grammars with respect to other modalities, e.g. focused field in display.

In addition, the "tap and talk"-interface explicitly provides dialogue state (i.e. the tapped field), simplifying the dialogue management compared to other multimodal systems.

Microsoft's Mipad (Multimodal Interactive Pad) [13, 14] is one example of multimodal interaction limited to speech and pen. With MiPad the user interacts with the PDA by tapping and holding on a field and uttering appropriate content to it, i.e. uncoordinated simultaneous multimodal input. MiPad is an application prototype offering a conversational, multimodal interface to Personal Information Manager (PIM) on a PDA, including calendar, contact list and e-mail.

2.4 Speech Centric Multimodality in Form-filling Applications: An Example

In order to illustrate the concepts of speech centric multimodality, we have developed a multimodal demonstrator platform with two categories of multimodal dialogues: Form filling applications [15] and a map/location-based system. The overall architecture is described in chapter 4. In this section we exemplify the benefits of speech centric multimodality in two form filling telephony applications: A train timetable information retrieval service and a "yellow pages" service. The system architecture for form-filling applications is here basically appropriate for the *Sequential* and *Non-coordinated simultaneous* multimodal input types.

Figure 1 shows the graphic user interface (GUI) in three dialogue steps of the service for the Norwegian train timetable information retrieval application.

1 This entry page appears on the screen when the service is called up. Below the text heading: "Where do you want to go?" there are five input form fields: Arrival and departure station, date and time of arrival and the number of tickets. The questions are also read aloud by text-to-speech synthesis (TTS).

2 This screen shows the result of the user request in natural language[1)]: "I want to go from Kristiansand to Bodø next Friday at seven o'clock". The key words in the utterance were recognised correctly and the corresponding fields filled in, giving the user an immediate feedback on the screen. The call

Figure 1 The GUI for the train timetable information retrieval application

1) *In Norwegian: "Jeg vil reise fra Kristiansand til Bodø neste fredag klokka sju".*

Figure 2 Yellow pages application

was made on June 10, so “next Friday” was correctly interpreted as June 15. Since all the information in the form fields on the screen is correct the user confirms by pushing the ‘OK’ button, and the system gets the requested information from the railway company web portal.

3 The result of the web request is presented on the screen. Usually three or four realistic alternatives are depicted on the screen. The user may then click on the preferred travel alternative, or say the alternative number. Then the dialogue goes on to ask how many tickets the customer wants for the selected trip, and the demonstrator service ends.

In the example in Figure 1, all the words were correctly recognised and understood and the visual presentation of information was much more efficient than audio feedback. Thus the customers efficiently obtained what they wanted. However, in real world speech-enabled telephony applications speech recognition errors will unavoidably occur. Correcting speech recognition errors in speech only mode (no visual feedback) is very difficult and reduces user satisfaction. But, with speech centric multimodal interface it is easier to correct ASR-errors in these form-filling services. If some of the information on the screen is wrong, the user corrects it by clicking on the field containing the erroneous words and then either saying the correct word once more or tapping on the correct word from the N-best list, which occurs on the right hand side of the field.

Figure 2 illustrates this situation in a “yellow pages” application:

1 On the entry page that appears on the screen when the service is called up and under the text heading “Welcome to Yellow pages”, there are two input form fields: Business sector and municipal (Norwegian: “Bransje” and “sted”).

2 When the user asked in natural language[2)] “I want bakeries in Oslo”, the ASR recognised the key words in the utterance and filled in the corresponding fields, giving the user an immediate feedback on the screen. Note that the N-best list on the right hand side of the sector field contains the alternative “Batteries”, i.e. the word “batteries” has the second best confidence score. Since all the information in the form fields on the screen is correct the user pushes the ‘OK’ button, and the system gets the requested information from the service provider.

3 The requested information is displayed on the screen. There are 25 bakeries in this listing, which would have been rather tedious to listen to. Here, the user easily gets a quick overview and clicks on the preferred baker.

The actions and benefits of multimodality in the form filling examples are summarised in Table 2.

3 General Implementation Aspects

The generic multimodal dialogue system shown in Figure 3 with several input and output channels, raises several important issues when implementing multimodal architectures, such as:

- Selection of suitable software architecture for implementing the logical elements shown in Figure 3 in a real system;
- Funnelling of multiple channels through the heart of the system, namely the dialogue manager.

[2)] *Pronounced in Norwegian as: “Jeg vil gjerne ha bakerier i Oslo”. Here the business sector (Norwegian: “Bransje”) is “bakerier” and the municipal (Norwegian “sted”) is Oslo.*

User actions	Benefits of multimodality
Natural language input, asking for several different pieces of information in one sentence.	Speech is most natural for asking this type of question. Speech is much faster than typing and faster than selecting from a hierarchical menu.
Reads information shown on the screen.	The user gets a quick overview – much quicker than with aural feedback reading sentence by sentence.
Taps in the field where the ASR-error occurs, and taps at the correct alternative in the N-best list.	Much easier to correct ASR-errors or understand rejections than in complicated speech-only dialogues. Better error control and disambiguation strategies (e.g. when there are multiple matching listings for the user query).

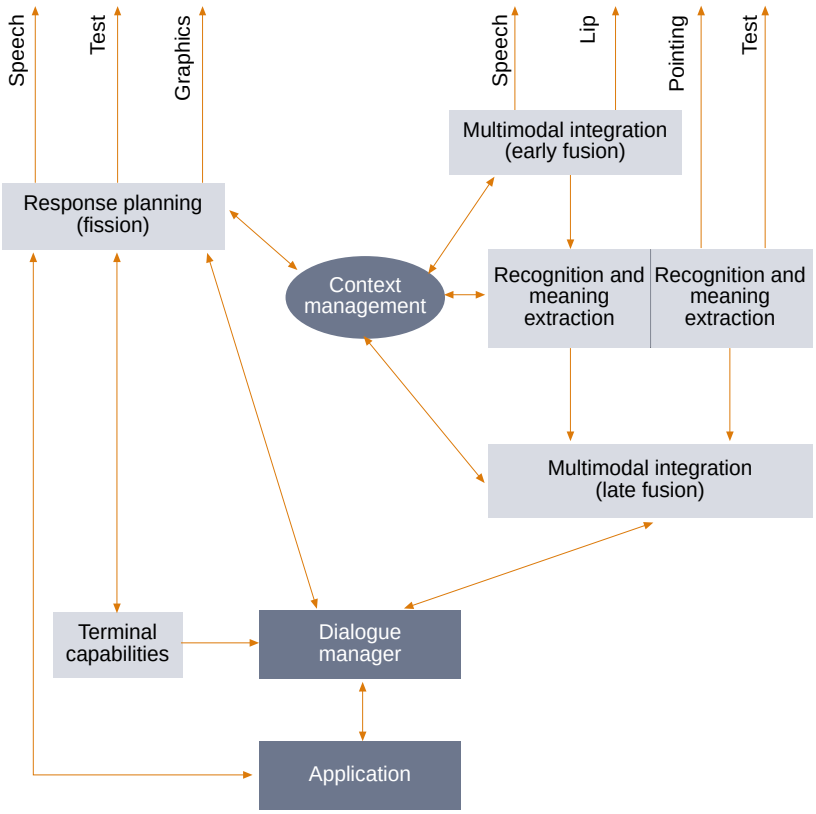
Table 2 Benefits of multimodality in the form-filling applications

3.1 Distributed Processing

In advanced multimodal systems, several input/output channels can be active simultaneously. In order to cope with these kinds of multimodality, an architectural support for simultaneous information flow is necessary. Furthermore, it is desirable to run different functional modules separately (often on different machines), in order to deal more effectively with the system's complexity. The so-called *distributed processing paradigm* matches these requirements quite nicely, and therefore most of the multimodal system architectures are based on this paradigm.

There are many different approaches to implementing a distributed software system. One can for example choose a very low level *socket*-based approach or one of the higher level approaches such as Parallel Virtual Machine (PVM) [42], Message Passing Interface (MPI) [43], RPC-XML [44], and SOAP [45]. On an even higher level, object oriented distributed processing approaches such as CORBA, DCOM, JINI and RMI are available. However, the most attractive approach to implementing multimodal systems is based on *co-operative software agents*. They represent a very high level of abstraction of distributed processing and offer a very flexible communication interface.

Figure 3 A generic multimodal dialog system architecture



3.2 Existing Agent-Based Platforms

There are several well-known agent architectures that have been used to build multimodal systems, such as GALAXY Communicator (a public domain reference version of DARPA Communicator) maintained by MITRE [19], Open Agent Architecture (OAA) from SRI international [20] and Adaptive Agent Architecture (AAA) from Oregon Graduate Institute (OGI) [21]. In these architectures a set of specialised agents are employed to perform different tasks. Two given agents can communicate (indirectly) with each other through a special agent called a *facilitator*. Usually, this inter-agent communication is a *service request* and a *response* to such a request. The facilitator performs matchmaking between a service provider and a requester.

We found that GALAXY Communicator is the most suitable agent-based platform for our purpose. A detailed description is given in section 4.2.1. In this system, messages sent between agents and the central facilitator called a *hub*, are based on a simple attribute-value type data structure. This *hub-spoke* type architecture allows easier asynchronous and simultaneous message exchange than for example a *serial* architecture does. In addition to a message passing between the hub and the agents, direct communication between any two agents is also possible. Such a direct connection is useful when large amounts of data, such as audio, have to be efficiently passed between two agents. One drawback with GALAXY Communicator, however, is its dependency on a single facilitator, whose failure will cause a complete system breakdown. In AAA this problem has been addressed by introducing many facilitators.

3.3 Fusion and Fission

Since an advanced multimodal system such as the one shown in Figure 3 has more than one input and/or output channel, there must be mechanisms to map:

- Several input channels to a single semantic stream, i.e. *fusion;*

- A single semantic stream to several output channels, i.e. *fission.*

From a technical point of view, fusion, also called *multimodal integration*, deserves a higher attention than fission, because a good fusion strategy can help reduce the recognition errors.

Usually, fusion is divided into two classes, *early fusion* and *late fusion*. Early fusion means integration of the input channels at an early stage of processing. Often this means integration of feature vectors, before they are sent through the recogniser(s). Late fusion means integration of the recogniser outputs, usually at a semantic interpretation level. Late fusion seems to have attracted more interest than early fusion, probably because it only needs the recogniser outputs and no changes of existing modules (such as feature extractors, recognisers).

In one of its simplest forms, late fusion can be performed by simple table look-ups. For example, assume that we have two input channels. Then we can maintain a (two-dimensional) table, where rows and columns correspond to alternative outcomes of the recognisers acting on channel 1 and channel 2 respectively. Each cell of the table can be marked 1 or 0, indicating whether this particular corresponding combination is valid or invalid. Then the fusion procedure for a given pair of recogniser output lists would be to scan the (recogniser) output combinations in the decreasing order of likelihood and find the first valid combination by consulting the table.

The above procedure can be extended to handle uncertainty associated with recognition by considering joint probability of the recogniser outputs from the two channels. One simple approach for computing these joint probabilities is to assume that two recognition streams are statistically independent. However, the fusion performance (i.e. multimodal recognition performance) can be enhanced by dropping this assumption in favour of more realistic assumptions [22].

Table look-up based fusion is not very convenient when the semantic information to be integrated is complicated. In such cases *typed feature structures* can be used. This data structure can be considered as an extended, recursive version of *attribute-value* type data structures, where a value can in turn be a feature structure. Typed feature structures can be used for representing meaning as well as fusion rules. Integration of two or several feature structures can be achieved through a widely studied algorithm called *feature-structure unification* [2].

In fusion, temporal relationships between different input channels are very important. In multimodal systems this issue is usually known as *synchronization*. In most of the reported systems in the literature, synchronization is achieved by considering all input contents that lie within a pre-defined time window. This can be done by employing timers and relying on the real arriving times of the input signals to the module responsible for performing fusion. However, a more accurate synchronization can be obtained by time-stamping all inputs as soon as they are generated since this approach will remove the errors due to transit delays. Note, however, that input synchronization is meaningful only for coordinated multimodality.

3.4 Dialogue Management

A dialogue manager is usually modelled as a finite state machine (FSM), where a given state S_t represents the *current context*. One problem with this modelling approach is the potentially large number of states even for a relatively simple application. This can be brought to a fairly controllable level by considering a hierarchical structure. In such a structure there are only a few states at the top level. But each of these states is thought to be consisting of several substates that lie on the next level. This can go on until the model is powerful enough to describe the application concerned.

When the user generates an event, a *state transition* can occur in the FSM describing the dialogue. The route of the transition is dependent upon the input. That means that state transition is defined by the tuple (S_t, I_t), where S_t is the current state and I_t is the current user input. Each state transition has a well-defined end state S_{t+1} and an output O_t. In other words the building-block-operation of the dialogue manager is as follows:

1 Wait for input (I_t).

2 Act according to (S_t, I_t) for example by looking up a database and getting the result (R_t)

3 Generate the output according to (S_t, I_t, R_t)

4 Set next state S_{t+1} according to (S_t, I_t)

The user input (I_t) is a vector which is a representation of the structure called concept table. This structure consists of an array of *concepts* and the values of each of these concepts. For example in a travel planning dialogue system the concept table can look as shown overleaf.

The column “value” of the concept table is filled using the values output by the speech recogniser and the other recognisers operating on the input modalities (e.g. a GUI tap recogniser). During the filling operation, input ambiguities can be resolved completing late fusion. Once filled, the

Concept	Value
<FROM_CITY>	Oslo
<TO_CITY>	Amsterdam
<DEPARTURE_TIME>	1600

concept table defines the current input I_t. More specifically, if the values in the concept table are $I_t(1)$, $I_t(2)$, ..., $I_t(n)$, then the N-tuple ($I_t(1)$, $I_t(2)$, $I_t(n)$) is the current input I_t. The number of different inputs can be prohibitively large, even if the length of the concept table (M) and the number of values a given concept can take (K) is moderate. This implies that a given state in the dialogue FSM has a large number of possible transitions.

A possible remedy for this problem is to employ a clever many-to-one mapping from the original input space to a new smaller sized input space, which exploits the fact that there are many don't-care concept values.

4 An Implementation of a Multimodal Demonstrator Platform

This chapter describes the architecture and the system aspects of the multimodal demonstrator platform developed in the EURESCOM project MUST [7], [16], [17], [18]. The main purpose of the demonstrator was to implement an experimental multimodal service in order to study and evaluate the interaction with and the appreciation of such a service by real "naïve" users. The service is a map application, i.e. a tourist guide for Paris.

4.1 System Overview

The multimodal demonstrator platform consists of a relatively complex server and a thin client.

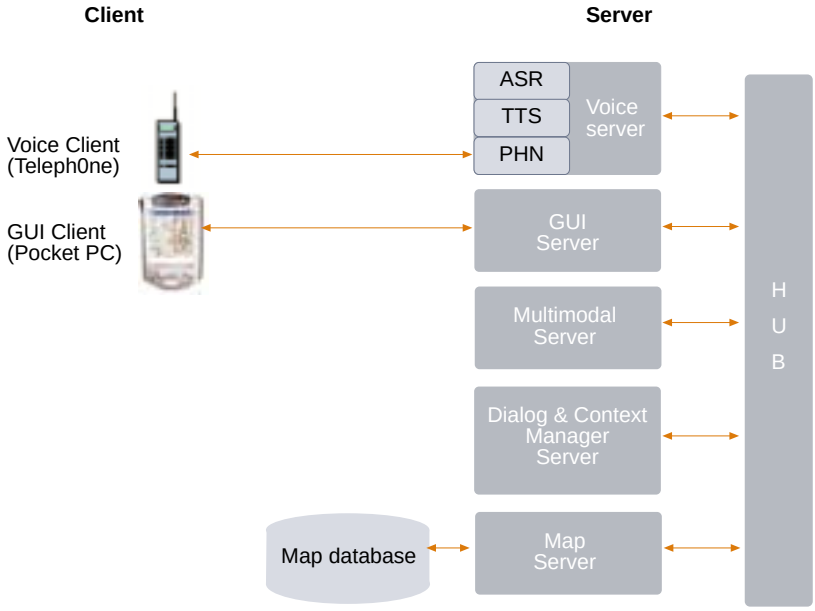


Figure 4 The architecture of the multimodal demonstrator platform

The overall architecture of the platform is shown in Figure 4.

The Application Server consists of five main autonomous modules (or servers) that inter-communicate via a central facilitator module (Hub). The modules are:

- *Voice Server* – comprises speech technological components such as Automatic Speech Recognition (ASR), Text to Speech Synthesis (TTS) and telephony (PHN) for the speech modality.
- *GUI Server* – is the gateway between the GUI Client and the server side.
- *Multimodal Server* – performs preliminary steps of multimodal integration of the incoming signals (fusion) and distributes the service response through the output channels (fission).
- *Map Server* – acts as a proxy interface to the map database.
- *Dialogue & Context Manager Server* – performs the dialogue and context management.
- *Hub* – manages the inter-communication for the modules at the server side.

The client side, or the multimodal mobile terminal, consists of two modules: The Voice Client which is a plain mobile telephone used for the voice modality, and the GUI Client which is a programmable Pocket PC (or a Personal Digital Assistant – PDA) with touch sensitive screen used for the tap and graphical modality.

All the modules in the MUST Application Server are described in section 4.2. The client side of the MUST demonstrator is described in section 4.3.

4.2 The Server Side

This section describes all the modules that comprise the Application Server. All the modules are implemented in Java, except for the Voice Server that is implemented in C++.

4.2.1 The Galaxy Hub

The modules communicate asynchronously by messages passing through a Hub. The Hub distributes the messages according to a set of rules in accordance with the service logic.

The Galaxy Communicator Software Infrastructure mentioned in section 3.2 was selected as the underlying software platform, which provides the Hub in Figure 4. The main features of this framework are modularity, distribution, seamless integration of the modules and flexibility in

terms of inter-module data exchange, i.e. synchronous and asynchronous communication through the Hub and directly between modules.

In addition to the Galaxy Communicator, two other promising software multimodal platforms have been considered: ATLAS [11, 12], and Smartkom [9]. The lack of availability of these software platforms in terms of licensing clarifications as well as available documentation in English led us to discard these two alternatives at an early stage and go for the Galaxy Communicator.

The Galaxy software infrastructure allows us to connect the modules (i.e. the GUI Server, Voice Server, Multimodal Server, Dialogue Server and the Map Server) together in different ways by providing extensive facilities for messages passing between the modules. A module, or a server, can very easily invoke a functionality that is being provided by another module without knowing which module that provides it or where it is running. This is an important feature of the modularity, distribution, and seamless integration.

Normally, a script is used to control the processing in the Hub, but it can also act as an autonomous facilitator in an agent-based environment. In our platform, we have followed the approach of script-based Hub control.

Galaxy messages that are passed between the modules are based on the key-value pair (attribute-value pair or *name* plus a *value*) format. This message format allows messages to encompass simple data structures that are sufficient to support less complex operations such as connection set-up, synchronisation, and disconnection. However, more complex operations such as database lookup results and GUI display information involve more complex data structures, necessitating an extension to the message format. Fortunately, the Galaxy message format is a framework that is flexible and extensible.

In the MUST project, an XML-based mark-up language (named MxML – 'MUST eXtensible Mark-up Language') was defined to cope with complex data structures in the Application Server. Complex data structures are represented by MxML strings and embedded in the basic key-value pair based Galaxy messages. In this way we can take the advantage of the flexible message passing mechanism provided by Galaxy Communicator and the power of XML.

The highly modular and seamless architecture of the Application Server gives a flexible system where modules easily can be removed, added or replaced. For example, we have used two alternative versions of the Voice Server (see section 4.2.2), and one can replace the other whenever necessary.

Due to the distributed nature of the architecture, many different deployment configurations are possible for the MUST demonstrator. We typically use two PCs – both running Windows 2000; one PC for the Voice Server (including a lot of firmware such as telephony drivers, ASR and TTS engines), and another PC for the other Servers. The demonstrator also supports a portable stand-alone version where all the modules (including the whole Voice Server) are running on one machine (i.e. a laptop).

4.2.2 Voice Server

The Voice Server (or Voice Platform) is the module that handles the voice modality. This module is built upon underlying speech technology such as ASR, TTS and telephony, and basically provides the API for these technology resources.

Because two of the partners in the MUST project (Portugal Telecom and Telenor) wished to use their own voice platforms, two different versions of the Voice Server have been developed for the MUST demonstrator. One is based on the "InoVox" [23] voice platform from Portugal Telecom Inovação, and the other version is based on the "Tabulib" voice platform [24] from Telenor R&D.

The Tabulib platform is currently freely available to the general public. Tabulib basically supports ISDN telephony for the input/output voice to the system, but also a proprietary VoIP (Voice over IP) solution is implemented for the development of a portable demonstrator platform without the need for a wired telephone line. The speech recogniser supported is Philips SpeechPearl 2000 [25] recogniser. For the Text-to-Speech Synthesis, Tabulib supports the Microsoft SAPI 4.0 [26] standard, and hence, all Text-to-Speech Synthesis engines supporting this standard can be used with Tabulib. We have used a TTS engine from Microsoft for English [27] and Telenor's TTS engine Talsmann [28] for Norwegian.

4.2.3 GUI Server

The GUI Server is the 'gateway' between the GUI Client (i.e. the GUI part of the multimodal mobile terminal) and the Application Server. Received data from the GUI Client is packed into Galaxy frames and transmitted to the Dialogue Server via the Multimodal Server. Data from the Dialogue Server is extracted from the received Galaxy frames, and an HTML file is generated. The generated HTML file is actually stored on an HTTP (Web) server and further

fetched by the GUI Client – which is a sort of Web browser.

The data from the Dialogue Server is in XML format and contains the contents to be presented on the GUI Client; that is, raw information such as text and images to be displayed, and coordinates for items – e.g. points of interest on a map. We use XSLT [29] in order to transform the XML body to an HTML file. It is the XSL style sheet that really defines the format of the GUI, such as the size of text fields, font types, background colours, the width of borders and list boxes. With the use of style sheets, the appearance of the GUI display can be easily altered on the fly in services where the GUI format should be dependent on the dialogue context or the user's profile. New applications can be implemented without any software upgrade on the GUI Client, since the GUI is defined by the XSL style sheets and the content in the XML body.

4.2.4 Multimodal Server

The MUST project aimed to investigate the implementation and human factor issues related to coordinated simultaneous multimodal inputs, i.e. all parallel inputs must be interpreted in combination, depending on the fusion of the information from all channels. The so-called "late fusion" approach was adopted, where the recogniser outputs are combined at a semantic interpretation level (by the Dialogue & Context Manager Server).

The Multimodal Server performs the preliminary steps of the integration of the incoming signals (pen click and voice). The temporal relationship between different input channels is obtained by considering all input signals within a pre-defined time window. The duration of this time window is a variable parameter that can be adjusted according to the dialogue state. If two input signals occur within the fusion window, then the Dialogue Server will combine the two inputs to determine the 'meaning' from the user, and act accordingly (late fusion). A drawback with this solution is that pen taps (e.g. on menu icons as "home" and "help"), which should not be combined with voice input, are also delayed through the fusion window. This slows down the system response.

In this version of the Must demonstrator the Dialogue Server performs the (early) fission, and the output modalities are forwarded to the destination channels.

4.2.5 Dialogue Server

The Dialogue Server, or the Dialogue and Context Manager module, can be seen as the mediator between the user on one side and the source of information on the other side. It is the Dialogue Server that controls the logic for the service and can be considered as the central processing unit of the system. The main tasks of the Dialogue Server is:

- Process incoming messages with textual or abstract representations from the user interactions
- Combine the multimodal inputs (late fusion)
- Context processing
- Implement the service logic rules
- Access data repository
- Handle the output information to the user (early fission)
- Error handling

The application in the MUST project (the tourist guide for Paris) is fully user driven, and it is the user who always takes the initiative in the dialogue. A big challenge in the implementation of such a dialogue system is that the context or dialogue states to a certain extent are submitted to the user's premises. The system should always be in the ready state for serving the user, i.e. receiving queries from the user at any time, which complicates the control and processing of the multimodal inputs and outputs. For system driven dialogues such as form filling applications, these implementation issues are alleviated since the Dialogue Manager takes the initiative and control in the interaction with the user.

The requests from the user are represented in terms of textual or abstract symbols by the boundary modules (here: Voice Server and GUI Server) that handle the interaction with the user. The Dialogue Server combines and processes the inputs (late fusion), and acts accordingly to fulfil the user's request (typically a query to a database). The feedback is sent to the boundary modules via the Multimodal Server in order to be presented in different modalities on the multimodal terminal (early fission).

4.2.6 Map Server

The Map Data Server is the module that receives and processes requests coming from the Dialogue Server. These queries are Galaxy messages constructed using a proprietary protocol defined to code domain information requests. These messages are parsed, interpreted and mapped to the corresponding SQL queries, which are executed to gather the requested data from the database. The data from the database is packed into a Galaxy frame, and forwarded to the Dialogue Server. An error frame is sent to

the Dialog Server if no matching data is found in the database.

4.3 The Client Side

At the time of the service specification and implementation, there were no commercial terminals available with support for the features and functionality needed for a small multimodal wireless terminal. The prototypes developed or under development in other projects were not available for third party entities.

The solution that was found to overcome this situation is the simulation of a multimodal terminal, which was done by combining a mobile GSM phone and a Pocket PC with touch sensitive screen.

4.3.1 The GUI Client

The GUI Client is implemented on a Compaq iPAQ Pocket PC running Microsoft CE 3.0/2002 (see Figure 5). The GUI Client is connected to the GUI Server via WLAN using a TCP socket connection to convey the GUI signals (tap and graphical information) back and forth between the client and server.

The GUI Client software is developed using Microsoft eMbedded Visual C++, and the main features are based on the Pocket Internet Explorer (web browser) technology. The use of ActiveX [30] controls inside the web browser gives a powerful interface that supports a variety of GUI components, such as gif image display, hotspots, push buttons, list boxes (select lists) and text fields.

The input to the GUI Client (from the GUI Server) is an HTML file. The GUI is defined and controlled by the use of Microsoft JScript [31] in the HTML code. In the MUST project, we have defined a set of JScript functions or methods for the creation of the different GUI components. This provides a powerful and flexible GUI, and it is the Application Server (here: Dialogue Manager Server and GUI Server) that defines the appearance of the GUI, and therefore no software updates are necessary on the Pocket PC in order to change the GUI. This gives a system for quick and easy design of multimodal applications.

4.3.2 The Voice Client

The speech part is for the time being handled in parallel by a mobile telephone. The users will not notice this “two terminal” solution, since the phone is hidden and the interface is transparent. Only the headset (microphone and earphones) with Bluetooth connection will be visible to the user. The headset frees the user’s hands to hold the Pocket PC and to navigate the map application with the pen giving the “illusion” that the user is interacting with a small multimodal wireless terminal.

Figure 5 The GUI Client implemented on a Compaq iPAQ Pocket PC

We have also implemented a VoIP solution for the speech part. Here we utilise the audio device on the Pocket PC. The input voice is recorded on the Pocket PC, and forwarded to the Voice Server via the WLAN connection. Likewise, the audio from the Voice Server is transmitted over the same connection, and played on the Pocket PC. The advantage with such a solution is that both the GUI and speech part are implemented on one single device, giving a better “illusion” that the users are interacting with a multimodal mobile telephone. Besides, there is no need for a telephone line for the Voice Server, and the whole demonstrator can be presented everywhere as a stand-alone device. Unfortunately, this solution is for the time being not stable, due to slow CPU performance and some problems with the audio device on the current Pocket PCs.

5 When Will Multimodal Applications Take Off in the Mobile Market?

We have discussed various aspects of multimodal interfaces for mobile communications and carefully described the technical implementation of a multimodal demonstrator platform. However, when will the multimodal services take off in the mobile market? In this chapter we look at the situation today, and address some issues that are important for deployment of regular multimodal services in the near future.

5.1 Status – Existing Multimodal Mobile Services and Platforms

Today a few commercial multimodal mobile services for GSM, GPRS and WLAN are based on a combination of existing standards such as WML (WAP), HTML, VoiceXML, SMS and MMS, or *proprietary* or *de facto* standards. The multimodal interaction is in most cases limited to one modality at a time for the input and output signals. Many of the services are expensive in use, and some involve more than one terminal for the end user, e.g. a cellular phone for the speech modality, and a Personal Digital Assis-

tant (PDA) for the graphical modality. Most often, only one network operator offers the services, i.e. the one that possesses the standard, and this limits the geographical coverage and the potential of offering a mass marketing service. Below are listed some typical examples of commercial services:

- PlatiNet in Israel [32] offers services to operators such as voice-enabled chat and conferencing applications, location-based applications such as contacting the nearest taxi service or finding a hotel, and the addition of voice menus to text-based Yellow Pages, allowing easy access to any listing.

- SFR of France [33] offers text or voice coverage of events such as the football World Cup and the Tour de France, and access to other location-based services such as sports, weather, festival details, and news. The service runs on a platform from NMS Communications [34].

- Hutchison Max Telecom and Orange in India offer a multimodal Football World Cup service, which delivers audio action replays from live matches to cellular subscribers. An SMS alert is sent to subscribers at the start of a match, and they are given a number to call for live voice coverage or text updates. The service runs on the multimodal platform from OnMobile Systems, Inc. [35] in Orange's GSM network.

- Kirusa [36] and SpeechWorks [37] have established a trial multimodal service on the French mobile operator Bouygues Telecom's GPRS mobile networks in France. The service supports simultaneous multimodality, where voice and visual information seamlessly interact.

5.2 Standardization

In the past, we have seen the appearance – and disappearance – of services based on *proprietary* or *de facto* standards, for example the so-called smart phones, portable personal assistants, email devices and many more. International standardization within the telecommunication sector is definitely an important basis for large-scale regular telecommunication services in the long term. Standards provide interoperability across different networks, and standard tools for the development and deployment of the service. Standards are also highly pertinent to the forthcoming multimodal services, and the initiation of standardization activities is needed in the field of multimodal communication, particularly for the next generation mobile network (3G/UMTS).

However, due to the recent dramatic changes of the economic environment in the telecommunication sector, many telecommunication companies are forced to seek new models to generate new revenues from the existing mobile networks, e.g. GSM and GPRS, due to the large investments. Moreover, the planned 3G/UMTS mobile network is considerably delayed for a variety of reasons, from technical problems to unpredicted large investment costs. This situation will, at least for a while, slow down the standardisation effort in the field of multimodal communication. So, what one can expect in the near future is a variety of multimodal services based on a mixture of existing standards over the existing mobile networks.

We will address two important fields where standardisation is important for the forthcoming multimodal services: the standardisation of programming languages for the implementation of service logic (e.g. multimodal dialogues), and the quality and protocols in the mobile networks.

5.2.1 Programming Languages for Developing Multimodal Dialogues

The big advantage with standard languages for the development of multimodal dialogues is that the service and third party content providers can, in a convenient and flexible way, deploy new multimodal services by exploiting common development resources. In the case of cross-platform portability, services can be distributed to different application servers. A common standard will definitely speed up the dissemination and availability of such services. A good example of this is the *VoiceXML* [38], which is an XML-based mark-up language for pure voice-driven dialogues, adopted by the W3C consortium in 2000.

As regards programming languages for multimodal dialogues, W3C already in 2000 announced plans for a multi-modal mark-up language [39], but little public documentation has been available since then. An interesting activity within W3C is the XHTML+Voice profile [40] basically designed for Web clients that support visual and spoken interaction.

In July 2002 the SALT Forum [41], backed by industry heavyweights, released version 1.0 specification of the SALT mark-up language. SALT is an extension of HTML and other mark-up languages (e.g. cHTML, XHTML and WML), which basically add a speech interface to web applications and services, for voice only (e.g. telephony) and for multimodal browsers.

Surely, the ongoing activities within W3C and the recent announcement of the SALT Forum are a significant attempt at standardising implementation languages for multimodal dialogues. However, most of these languages are focusing

on web applications with dialogues of the form-filling nature.

For more advanced multimodal dialogues, with mixed initiative (both user and system driven) dialogues, and with support for all the three types of multimodal inputs (i.e. *sequential*, *uncoordinated simultaneous* and *coordinated simultaneous*), we believe that there are still some challenges to overcome before a general standard for multimodal dialogues can be drafted.

5.2.2 Mobile Wireless Networks – Quality and Protocols

The new forthcoming multimedia and multimodal mobile services will certainly put some requirements on the mobile networks. Key parameters are bandwidth, transmission delay, simultaneous and synchronous transmission of voice and data, handover capabilities, and quality of service. Of course, such parameters are to a certain extent given for the existing mobile communication networks and the forthcoming 3G/UMTS, and one should rather estimate the opportunity as well as the limitations for deploying multimedia services in mobile networks.

Critical network scenarios are the ones that involve handover between different network segments (e.g. between GPRS and WLAN) with different properties. It is important that the quality of the connection is stable, i.e. parameters such as bandwidth, jitter and time lag should, to the extent possible, be stable throughout the handover operation. Today handover in the existing mobile networks is often a problem, especially for real time traffic such as voice and video.

Standard protocols for the network transmission are an important issue. There is a need to define multi-channel transmission protocols for the different multimedia or multimodal communications, from the simplest including only voice and pen click, to more advanced applications including other modalities such as gesture. The protocols are required to work across different types of mobile networks obtaining a transparent connection. The development of such protocols should be done within standardisation bodies such as ITU, ETSI, 3GPP and IETF.

5.3 Multimodal Mobile Terminals

Widespread use of mobile multimodal services requires suitable terminals. Technically, we can classify mobile terminals into two main classes: Pre-programmed terminals and factory and programmable terminals.

Pre-programmed terminals need to be compatible with the prevailing multimodal standards, at both application and communication levels. However, standardisation of multimodal protocols is still in its infancy and it may take some time before such standard compliant multimodal services will be taken into large-scale use.

Programmable terminals, on the other hand, allow swift introduction of multimodal services by using proprietary protocols. However, multimodal services require terminals that can be used with a network supporting simultaneous voice and data transmission at a sufficiently high bit rate. The emerging mobile networks, GPRS and UMTS, as well as WLAN available in hotspots, definitely satisfy this requirement, because they inherently support the IP protocol. Even the existing GSM network can be used by employing a modem to simulate an IP layer on top of the circuit switched net. In order to enable the use of proprietary protocols, the service provider first has to release a client program that the users can install in their terminals. This client and the service can then communicate with each other using their own application protocol.

There is already a large number of programmable mobile terminals on the market which are potential candidates for use with multimodal services. Compaq iPAQ, Toshiba Pocket PC, HP Jornada, O2 xda and Siemens PocketLoox etc. are some of the popular ones. They all come with the pre-installed Microsoft Windows CE/Pocket PC operating system for which a complete programming environment is available. These terminals have relatively large colour screens, 200 MHz or more CPU frequency, 32 MB or more RAM, audio ports and PCMCIA/CF ports. Except for O2 xda, they do not come with built-in communication hardware. However, the PCMCIA or CF (Compact Flash) port allows insertion of a communication module, such as a GPRS or WLAN card. Siemens has announced that a specially designed GPRS module for PocketLOOX will be available this year. There is reason to believe that this trend will continue and even more complete communication terminals of this kind will be available in the near future.

In spite of these developments, mobile terminals cannot be expected to be so resource-rich that sophisticated components of multimodal systems (such as speech recognition and text-to-speech synthesis modules) can be contained by the terminal itself. However, this is not an obstacle to the provision of multimodal services, as speech engines and other resource hungry components can be placed in the network. But this has significant architectural and business implications for the mobile operator, application provider, and end user, and understanding this is perhaps more important than it appears.

Even though the mobile terminals seem to have overcome the major technological barriers on the way towards multimodal services, the price of these terminals is still high. Generally a mobile terminal capable of delivering multimodal services of reasonable quality is at least ten times more expensive than a usual mobile telephone. Therefore, one cannot expect mobile multimodal services taking off until the price comes down to a level that the consumers will accept.

Human Factors

While all these engineering and technical improvements are important, the ultimate success of a multimodal system has to be assessed from the user's viewpoint. Understanding the nature and constraints of the new access interfaces is critical for commercial success of future services on mobile terminals. Also, it is important to understand why people opt for certain (combinations of) modalities in the given context, in order to develop efficient multimodal interfaces and services [3]. It is therefore necessary to run user experiments and quantify the advantages of the multimodal system in terms of user-satisfaction, dialogue success rate and dialogue completion time. These experiments are high on our agenda.

Acknowledgements

In this research activity, Telenor R&D has collaborated with France Telecom and Portugal Telecom and researchers from the Netherlands in the EURESCOM[3)] project called "MUST – Multimodal, multilingual information Services for small mobile Terminals" [6], [7].

We would like to thank our colleagues in the Speech Technology Group at Telenor R&D, and our colleagues in the MUST project for valuable discussions and cooperation.

References

1 Bolt, R. Put That There : Voice and Gesture at the Graphics Interface. *Computer Graphics*, 14 (3), 262–270, 1980.

2 Oviatt, S et al. Designing the user interface for multimodal speech and gesture applications: State-of-the-art systems and research directions. *Human Computer Interaction*, 15 (4), 263–322, 2000.

3 Oviatt, S. Ten Myths of Multimodal Interaction. *Communications of the ACM*, 42 (11), 74–81, 1999.

4 Oviatt, S. Multimodal interface research : A science without borders. *Proc. 6th International Conference on Spoken Language Processing (ICSLP 2000)*, Beijing, China, III, 1–4, 2000.

5 *Center for Human-Computer Communication*. 2002, November 8 [online] – URL: http://www.cse.ogi.edu/CHCC/index.html

6 *Eurescom*. 2002, November 8 [online] – URL: http://www.eurescom.de/

7 *MUST – Multimodal and Multilingual Services for Small Mobile Terminals*. Heidelberg, EURESCOM Brochure Series, May 2002. (http://www.eurescom.de/public/projects/P1100-series/P1104/default.asp)

8 *W3C – Multimodal Requirements for Voice Markup Languages*. 2002, November 8 [online] – URL: http://www.w3.org/TR/multimodal-reqs

9 *SMARTCOM – Dialog-based Human-Technology Interaction by Coordinated Analysis and Generation of Multiple Modalities*. 2002, November 8 [online] – URL: http://www.smartkom.org/start_en.html

10 Wahlster, W et al. SmartKom: Multimodal Communication with a Life-Like Character. *EUROSPEECH-2001*, Aalborg, Denmark, 1547–1550, 2001.

11 Gustafson, J et al. Adapt – A Multimodal Conversational Dialogue System In An Apartment Domain. *Proc. 6th International Conference on Spoken Language Processing (ICSLP 2000)*, Beijing, China, II, 134–137, 2000.

12 Beskow, J et al. Specification and Realisation of Multimodal Output in Dialogue Systems. *Proc. 7th International Conference on Spoken Language Processing (ICSLP 2002)*, Denver, USA, 181–184, 2002.

13 Huang, X et al. MiPad : A multimodal interaction prototype. *Proc. ICASSP 2001*.

14 Wang, Ye-Yi. Robust language understanding in MiPad. *Proc. EUROSPEECH-2001*, Aalborg, Denmark, 1555–1558, 2001.

15 Kvale, K, Warakagoda, N D, Knudsen, J E. Speech-centric multimodal interaction with

3) EURESCOM, the European Institute for Research and Strategic Studies in Telecommunications, performs collaborative R&D in telecommunications. The EURESCOM shareholders are mainly the major European network operators and service providers.

small mobile terminals. *Proc. Norsk Symposium i Signalbehandling (NORSIG 2001)*, Trondheim, Norway, 12–17, 2001

16 Boves, L, den Os, E (eds.). *Multimodal services – a MUST for UMTS*. January 2002. 2002, November 8 [online] – URL: http://www.eurescom.de/public/projectresults/P1100-series/P1104-D1.asp

17 Almeida, L et al. The MUST guide to Paris – Implementation and expert evaluation of a multimodal tourist guide to Paris. *Proc. ISCA Tutorial and Research Workshop (ITRW) on Multi-Modal Dialogue in Mobile Environments (IDS 2002)*, Kloster Irsee, Germany, 49–51, 2002.

18 Almeida, L et al. Implementing and evaluating a multimodal and multilingual tourist guide. *Proc. International CLASS Workshop on Natural, Intelligent and Effective Interaction in Multimodal Dialogue Systems*, Copenhagen, Denmark, 2002.

19 *GALAXY Communicator*. 2002, November 8 [online] – URL: http://fofoca.mitre.org/

20 *Open Agent Architecture*. 2002, November 8 [online] – URL: http://www.ai.sri.com/~oaa

21 *Adaptive Agent Architecture*. 2002, November 8 [online] – URL: http://chef.cse.ogi.edu/AAA/

22 Wu, L, Oviatt, L, Cohen, P R. Multimodal Integration – A Statistical View. *IEEE Trans. on Multimedia*, 1 (4), 334–341, 1999.

23 *Inovox*. 2002, November 8 [online] – URL: http://www.ptinovacao.pt/comunicacao/noticias/2002/jul2_inovox.html

24 Knudsen, J E et al. *Tabulib 1.4 Reference Manual*. Kjeller, Telenor R&D, 2000. (R&D note N 36/2000)

25 *Philips*. 2002, November 8 [online] – URL: http://www.speech.philips.com/

26 *Microsoft Speech API 4.0*. 2002, November 8 [online] – URL: http://research.microsoft.com/srg/docs/

27 *Code-it Software*. 2002, November 8 [online] – URL: http://www.code-it.com/tts_engines.htm

28 *Telenor Talsmann*. 2002, November 8 [online] – URL: http://www.telenor.no/fou/prosjekter/taletek/talsmann/

29 *XSLT*. 2002, November 8 [online] – URL: http://www.w3.org/TR/xslt

30 *ActiveX*. 2002, November 8 [online] – URL: http://www.microsoft.com/com/tech/ActiveX.asp

31 *Microsoft Jscript*. 2002, November 8 [online] – URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/js56jsoriJScript.asp

32 *PlatiNet*. 2002, November 8 [online] – URL: http://www.platinet.com/

33 *SFR*. 2002, November 8 [online] – URL: http://www.sfr.com/en/index.jsp

34 *NMS*. 2002, November 8 [online] – URL: http://www.nmscommunications.com

35 *OnMobile*. 2002, November 8 [online] – URL: http://www.onmobile.com/

36 *Kirusa*. 2002, November 8 [online] – URL: http://www.kirusa.com/

37 *SpeechWorks*. 2002, November 8 [online] – URL: http://www.speechworks.com/

38 *VoiceXML*. 2002, November 8 [online] – URL: http://www.voicexml.org/

39 *W3C Multimodal Interaction Activity*. 2002, November 8 [online] – URL: http://www.w3.org/2002/mmi

40 *W3C XHTML+Voice Profile*. 2002, November 8 [online] – URL: http://www.w3.org/TR/xhtml+voice/

41 *SALT Forum*. 2002, November 8 [online] – URL: http://www.saltforum.org/

42 *PVM home page*. 2002, November 8 [online] – URL: http://www.epm.ornl.gov/pvm/pvm_home.html

43 *MPI home page*. 2002, November 8 [online] – URL: http://www-unix.mcs.anl.gov/mpi/

44 *RPC-XML home page*. 2002, November 8 [online] – URL: http://www.xmlrpc.com

45 SOAP protocol specifications. 2002, November 8 [online] – URL: http://www.w3.org/2002/xp/Group/